ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

" Diseño e implementación de un sistema de envío de mensajes de texto bajo demanda utilizando Asterisk y Chan mobile "

INFORME DE MATERIA DE GRADUACIÓN

Previa a la obtención del Título de:

INGENIERO EN TELEMÁTICA

Presentado por

DANILO ANDRÉ LITUMA CASANOVA

ERIKA SOFIA GALLARDO TOLEDO

Guayaquil - Ecuador

2012

AGRADECIMIENTO

A Dios por permitirnos alcanzar nuestras metas luego de mucho esfuerzo y dedicación.

A nuestros padres por su apoyo incondicional en nuestras vidas, en especial durante los años de estudio dedicados a nuestras carreras.

DEDICATORIA

A mi hija Valentina y mi esposa, a quienes siempre las llevo en mi corazón.

A mi madre, quien ha sido incondicional en cada paso de mi vida brindándome todo su apoyo y siendo para mí la persona que más admiro en este mundo.

Danilo Lituma Casanova

A mis padres, que siempre estuvieron a mi lado dándome todo el apoyo y buen ejemplo para alcanzar mis metas.

A mis abuelos por su dedicación y cariño hoy puedo ver alcanzada la culminación de mi carrera, ya que siempre estuvieron impulsándome en los momentos más difíciles.

Erika Gallardo Toledo

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de este Trabajo de Grado, nos corresponde exclusivamente; y el patrimonio intelectual de la misma a la Escuela Superior Politécnica del Litoral"

_____

Danilo André Lituma Casanova

_____

Erika Sofía Gallardo Toledo

RESUMEN

Con el objetivo de implementar un sistema de envío de mensajes de texto bajo demanda utilizando asterisk y su módulo Chan Mobile se realizó el diseño de un sistema, sobre el cual se hicieron pruebas para descartar compatibilidad de software y hardware, para luego proceder con su implementación. Este sistema sigue el modelo cliente-servidor y sirve para realizar compras de mensajes de texto acerca de noticias del día a día y el horóscopo diario.

Para implementar este sistema se utilizó un servidor Linux, sobre el cual se instaló una base de datos MySQL y la PBX Asterisk. Además de esto se instaló un adaptador Bluetooth USB con su respectivo controlador para realizar una conexión Bluetooth con un teléfono celular que funciona como GW hacia la red GSM. La interacción entre el servidor Asterisk y la GSM se hace a través del modulo Chan Mobile que se instaló y configuró correctamente para el reconocimiento de los dispositivos involucrados en la conexión, en este caso el adaptador Bluetooth y el teléfono GW.

La implementación de este sistema tiene un costo económico muy bajo y puede ser mejorado añadiendo uno o 2 dispositivos extras de bajo costo para brindar un servicio más eficiente. El sistema diseñado puede ser puesto en producción tomando en cuenta las recomendaciones especificadas en este documento en la sección conclusiones y recomendaciones.

INDICE GENERAL

ABREVIATURAS

AGI:	Interface de entrada de Asterisk
ALSA	Arquitectura de sonido avanzada para Linux
AMI	Interface de administración de Asterisk
CDR	Registros de llamadas de datos
CEL	Registro de eventos de canal
DSCP	Código de diferenciación de servicios
FXO	Oficina de intercambio externo
FXS	Estación de intercambio externo
GSM	Sistema global para comunicaciones móviles
GW	Puerta de enlace
IAX	Intercambio entre servidores Asterisk
ISDN	Red digital de servicios integrados
MAC	Control de acceso al medio
OSS	Sistema de sonido abierto
PBX	Centralita privada
RTPC	Red de telefonía pública conmutada

RAM	Memoria de acceso aleatorio
RTCP	Protocolo de control en tiempo real
RTP	Protocolo de transporte de tiempo real
SIP	Protocolo de inicio de sesión
SMS	Servicio de mensajería instantanea
VoIP	Voz sobre IP.

INDICE DE FIGURAS

INDICE DE TABLAS

INTRODUCCION

Este proyecto consistióenel diseño e implementación de un sistema de envío de mensajes de texto bajo demanda, utilizando el software libre Asterisk y el módulo Chan Mobile, el cual será capaz de acceder a la red GSM usando un teléfono celular como gateway por medio de una conexión bluetooth.

El proceso funciona de la siguiente manera: el usuario realizaunallamada al teléfono que se conectacon el servidor asterisk, donde dispone de unmenú de voz para escoger la categoría que desea, las cuales son: noticias y horóscopo diario. El usuario escoge una opción y como respuesta recibirá un mensaje de texto.

La información de los mensajes de texto y comprases almacenada en una base de datos. Esta información es ingresada a diario manualmente a través de una aplicación desarrolladapor los autores de este proyecto. Esta aplicación permite realizaractualizaciones y consultas en la base de datos del sistema. Esta aplicaciónfue desarrolladacon el objetivo de que un usuario administrador del sistema sin conocimientos técnicos de base datos y SQL pueda actualizar la información de los mensajes y realizar consultas en el sistema.

CAPÍTULO 1

1. Antecedentes y justificación

1.1 Antecedentes

En la actualidad las comunicaciones son el eje principal del desarrollo mundial. Hoy por hoy, el ser humano cuenta muchas alternativas para comunicarse, las cuales facilitan la vida, eliminado distancias y multiplicando los horizontes de la información.

El teléfono celular es uno de los medios de comunicación que más impacto ha tenido sobre la forma de vivir de las personas, ya que a través de este dispositivos es posible establecer comunicación ya sea por un mensaje de texto, llamada telefónica o internet.

El uso del software libre en el campo de las comunicaciones cada vez es mayor. Con el desarrollo de las telecomunicaciones, sobre todo de las redes de datos, se empezó a transmitir voz sobre

el protocolo IP (VoIP). En sus inicios VoIP era tan solo aplicaciones gratuitas que permitían enviar pequeños segmentos de voz. De inmediato se desarrollaron aplicaciones que permitían transmitir no solo voz sino también video, aunque con una baja resolución de video, lo que hoy se conoce como videoconferencia.

En la actualidad el desarrollo de la telefonía ha sido muy importante para las empresas, entidades financieras, educativas, comerciales y todo el mundo corporativo en general. La implementación de las centralitas telefónicas o PBXs que ofrecen varios fabricantes como Nortel, Panasonic, entre otros es hoy en día indispensable. Además de este tipo de centralitas se han desarrollados centralitas en software que brindan mucha flexibilidad y beneficios a nivel de administración. Con el desarrollo de este tipo de centralitas aparece el software libre introduciendo la PBX Asterisk, que permite obtener todas las funcionalidades de una central telefónica, desde conectar varios teléfonos entre sí, servicio de buzón de llamadas, distribución automática de llamadas hasta servicios RDSI.

Asterisk ha tenido una gran acogida en el mercado corporativo por ser software libre y brindar todos los servicios que cualquier otra central ofrece. Asterisk puede conectarse a la red pública de

telefonía usando enlaces troncales. También puede conectarse a la red GSM usando teléfonos celulares como gateways conectados al servidor a través de bluetooth.

1.2 Justificación

Las telecomunicaciones y en particular la telefonía móvil han tenido un avance muy notorio en cuanto sus aplicaciones y servicios. Hoy en día es posible acceder a mucha información a través del celular, esta es una muestra clara de los avances que tienen las telecomunicaciones. Además de que el teléfono móvil ha cambiado la forma de comunicación entre las personas, ya que no solo es un dispositivo utilizado para realizar llamadas y enviar mensajes de texto. Hoy en día la transmisión de videollamadas y datos, forman parte de la nueva forma de comunicación, facilitando así el estilo de vida del ser humano.

El ofrecer la facilidad a las personas de acceder a mucha información, como noticias, deportes, horóscopo, concejos, entretenimiento, alertas, entre otros, tan solo desde su celular es un servicio que a medida que pasa el tiempo, se está convirtiendo en una necesidad. Existen empresas que no cuentan con los recursos necesarios para implementar un

servicio como este. Una alternativa para solucionar este problema es utilizar el canal Chan_mobile de Asterisk, el cual brinda una administración sencilla, eficiente y a un bajo costo ya que es software libre.

1.3 Descripción del proyecto

Para la implementación de este sistema de envió de mensajes de texto bajo demanda usando Asterisk, se tienen los siguientes objetivos.

1.3.1 Objetivos Generales

- Implementar un sistema de envió de mensajes de texto bajo demanda, de forma eficaz, rápida y a un bajo costo, utilizando el Software Libre Asterisk y un teléfono móvil como puerta de enlace a la red GSM.

1.3.2 Objetivos Específicos

- Diseñar un sistema capaz de enviar información a través de mensajes de texto a todos los móviles que hacen

requerimientos. El sistema debe contar con una base de datos donde se registran solicitudes de mensajes de texto o compras realizadas por los usuarios. El funcionamiento del sistema debe ser automático. El administrador del sistema solo debe ingresar la información de los mensajes texto que se van a enviar y llevar un control del funcionamiento del sistema a través de la consola de administración de Asterisk y de una aplicación java desarrollada para la actualización de la información y consultas del sistema.

- Proporcionar un servicio confiable, rápido y con información actualizada para optimizar los requerimientos de los usuarios.
- Utilizar una base de datos para registrar la información de las entidades involucradas con el objetivo de llevar una mejor administración del sistema.
- Utilizar un teléfono móvil como gateaway para conectar o comunicar el servidorAsterisk con la red GSM. Este teléfono debe ser compatible con el canal chan mobile y con el adaptador bluetooth que se conecta al servidor.

- Aprender a instalar y administrar un sistema Asterisk y utilizar el modulo Chan Mobile de Asterisk para la implementación del servicio que se quiere brindar.
- Investigar que teléfonos móviles y dispositivos bluetooth USB son compatibles con el módulo Chan Mobile.

Se desea implementar un sistema que brinde un servicio de información a través de mensajes de texto, mediante el cual los usuarios puedan recibir información sobre noticias y horóscopo diario como consecuencia de un requerimiento previo de los usuarios.

- El servidor Asterisk enviará la información requerida por el usuario con el número del destinatario al teléfono Gateway, para que luego este lo reenvié a su destinatario respectivo de la GSM. Este teléfono debe ser compatible con Chan mobile para lo cual se deben realizar pruebas con diferentes teléfonos y adaptadores bluetooth, para luego analizar el porcentaje de compatibilidad que ofrece Asterisk en general con todos los fabricantes de telefónos y adaptadores bluetooth USB.

Se utilizará un servidor con una distribución Linux Ubuntu, motor de base de datos SQL, Asterisk con el módulo Chain Mobile, teléfono móvil en modo audio Gateway y un dispositivo bluetooth USB para la comunicación entre el servidor Linux y el teléfono móvil. El esquema del sistema que se desea desarrollar se detalla en la Figura 1.1.

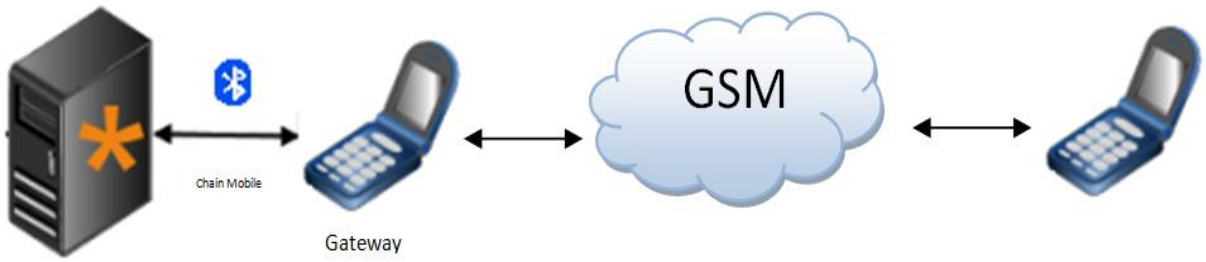


Figura 1.1 Esquema general del sistema

1.4 Metodología

El proyecto implementado es un sistema de envío de mensajes de texto bajo demanda. El sistema tienela capacidad de enviar mensajes a través de la red GSM por medio del módulo Chan Mobile de Asterisk, que fué implementado sobre una distribución de Linux, Ubuntu.

Además se dispone de un menú de voz para realizar las peticiones o requerimientos de los usuarios implementado sobre Asterisk. Los usuarios disponen de este menú al llamar al teléfono Gateway.

Cabe recalcar que se utilizó un dispositivo bluetooth USB para conectar el teléfono Gateway con el servidor Asterisk.

1.5 Perfil de la tesis

El objetivo principal de nuestra tesis es implementar un sistema que comunique una centralita Asterisk con la red GSM usando el módulo Chan_mobile. Esto con el objetivo específico de brindar un servicio de envío de mensajes de texto bajo demanda. Para lo cual se dispondrá de un menú de voz que permitirá al usuario realizar las peticiones al servidor. En los próximos capítulos se explicacómo funciona Asterisk con su módulo Chan Mobile y como se implementó el sistema.

CAPITULO 2

2. Asterisk y Chan Mobile

2.1 Asterisk

2.1.1 Historia de Asterisk

Las primeras centrales telefónicas se crearon para interconectar un conjunto de líneas telefónicas de un punto hacia otro y la conmutación de una línea a otra se realizaba manualmente. Con el desarrollo de la tecnología y la telefoníase consiguió automatizar este sistema, de manera que la conmutación se realizara automáticamente y no se necesitaba tener un gran número de operadores realizando este trabajo. Hoy en día, tenemos la red de telefonía pública conmutada o RTPC que es un conjunto de centrales telefónicas analógicas conectadas entre sí y a través de la cual la conmutación de circuitos permite interconectar las líneas telefónicas de los usuarios. Esto facilitó el crecimiento de la red.

A medida que pasaban los años, el teléfono se convirtió en algo muy necesario y pasó a ser considerado como un servicio básico.

Entonces se crearon las centralitas telefónicas secundarias automáticas o PBXs. Mediante estas centralitas las empresas se conectaban a la PSTN y lograban tener una mejor administración y control de sus teléfonos internos asignando un número de n dígitos para cada teléfono de la empresa denominado extensión. Empresas como Panasonic, Samsung, Nortel, entre otras, empezaron a fabricar y vender este tipo de centrales.

Luego de varios años, con el desarrollo de las redes de datos surgió la transmisión de voz sobre el protocolo IP. Esto trajo consigo el desarrollo de centralitas telefónicas para VoIP que en ciertos casos han pasado a remplazar a las centralitas analógicas debido a que actualmente las empresas utilizan enlaces de datos para comunicarse entre sucursales y estos enlaces pueden ser aprovechados no solo para transmisión de datos sino también para voz o video. Actualmente existen varios protocolos de VoIP como SIP, IAX, H.323, MGCP o SCCP.

Asterisk fue desarrollada por Mark Spencer utilizando lenguaje C debido a la necesidad de adquirir una central telefónica de bajo costo para su empresa inicial. Spencer en ese entonces era un estudiante de ingeniería y había desarrollado previamente proyectos como el cliente de chat GAIM que actualmente se lo conoce como PIDGIN y además de este, otros proyectos de

software libre. Este joven emprendedor inició lo que hoy conocemos como Asterisk. [2]

Spencer junto con otros programadores que contribuyeron a la corrección de errores y adición de ciertas funciones desarrollaron la PBX Asterisk inicialmente para el sistema operativo GNU/Linux. Sin embargo las versiones actuales son compatibles con los sistemas operativos BSD, Mac OS X, Solaris, Microsoft Windows. Aunque para la plataforma Linux que es la nativa existe un mejor soporte que para el resto de las plataformas. La compañía que desarrolla hardware y software para Asterisk se llama Digium.Para la implementación de este proyecto se escogió la distribución Ubuntu del sistema operativo Linux y la versión 1.8 de Asterisk. Es decir que la descripción de la arquitectura del software que se hace en este documento hace referencia al sistema operativo Linux – Ubuntu.

2.1.2 Asterisk. Definición y descripción funcional

Asterisk es un software que funciona como una PBX (**P**rivate **B**ranch E**x**change) o PABX (**P**rivate **A**utomatic **B**ranch E**x**change). En otras palabras, podemos describir a Asterisk como una central telefónica secundaria privada que se conecta a la red de telefonía pública a través de líneas troncales para gestionar llamadas internas, llamadas entrantes y salientes con autonomía sobre cualquier otra central telefónica [1]. Es importante mencionar que está basado en el protocolo IP. Sin embargo, podemos conectar Asterisk a la RTPC (Red de telefonía pública conmutada) que es una red analógica, a través de líneas troncales y utilizarla como cualquier centralita telefónica. También se puede conectar Asterisk a la red de telefonía móvil GSM a través enlaces bluetooth entre la PBX y uno o más teléfonos móviles que funcionan como gateways.

Si se quiere conectar Asterisk a una red como la RTPC necesitamos tarjetas o módulos FXO (Foreign Exchange Office). Estas tarjetas, usualmente son tarjetas PCI o PCI Express que se añaden a la tarjeta madre del servidor Asterisk y contienen uno o más puertos FXO, en los cuales se puede conectar líneas telefónicas de la red pública de telefonía, es decir son interfaces que funcionan como puerta de enlace a la RTPC.

También existen las tarjetas FXS que proporcionan puertos para conectar teléfonos analógicos a la PBX. Es decir que si se tiene un teléfono analógico es posible conectarlo con un servidor Asterisk a través de un puerto FXS. Hoy en día, se puede encontrar en el mercado tarjetas con puertos FXO y FXS integrados (ver Figura 2.1). La compañía Digium fabrica este tipo de tarjetas compatibles con Asterisk.

Fuente: http://www.gsm-gprs-modem.com/china-gsm_gateway_call_termination_800p-134395.html

Figura 2.1ETROSS-800P, Tarjeta PCI con puertos FXS / FXO

En este proyecto se realiza una conexión a la red GSM a través de un canal bluetooth. Aunque Asterisk es una PBX que transmite voz sobre IP, puede conectarse a cualquier tipo de red de telefonía usando el

hardware adecuado. Esto convierte a Asterisk es una PBX muy versátil y con muchas ventajas sobre cualquier otra centralita.

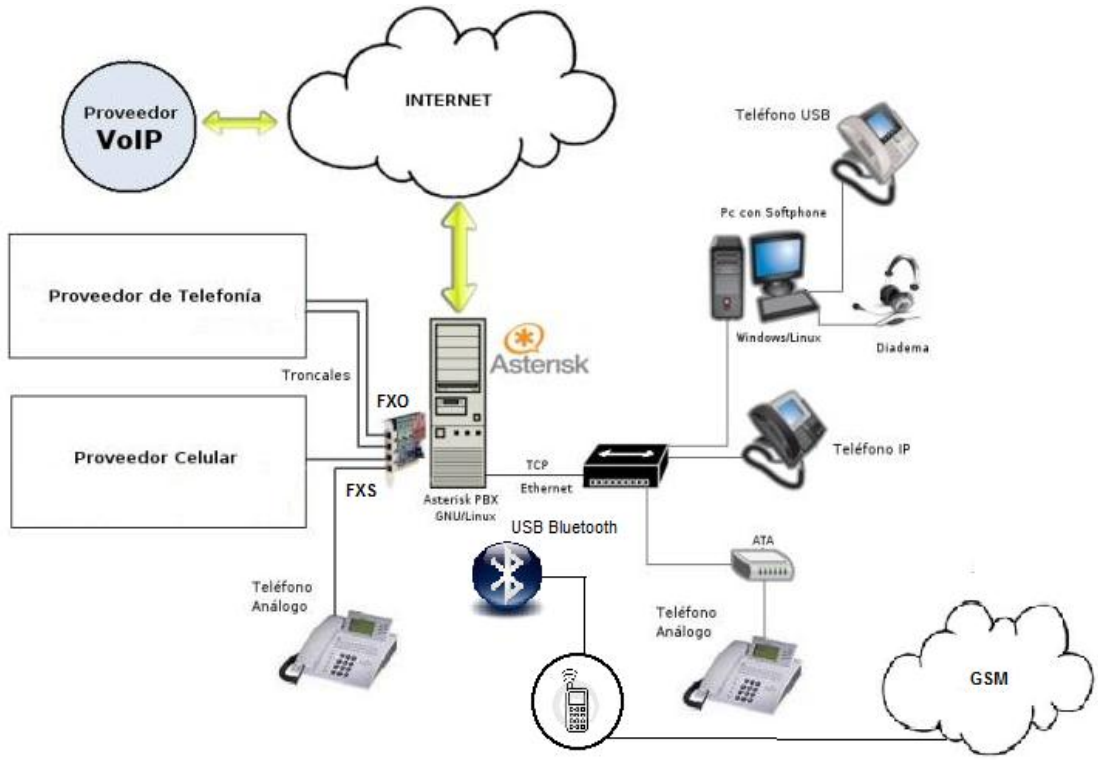


Figura 2.2 Asterisk y su interacción con diversas redes de telefonía y dispositivos terminales

Tal como se indica en el subcapítulo 2.1.1 Asterisk fue desarrollado en un principio para plataforma Linux y por ende es la opción más segura y con mejor soporte, por ser la plataforma nativa. Las versiones actuales pueden ser instaladas sobre Windows, Solaris, BSD o MAC, sin embargo la mayor parte de los servidores Asterisk están bajo plataforma Linux. Para este proyecto se utilizó un servidor Linux con la

distribución Ubuntu. A continuación se describe la arquitectura del software basada en el sistema de archivos que utiliza Ubuntu.

Asterisk presenta una arquitectura basada en módulos. Los módulos son componentes que son cargados para cumplir una función específica. Estos se guardan en el archivo modules.conf el cual se guarda en la ruta /etc/asterisk/modules.conf.

Existen varios tipos de módulos entre los principales tenemos: módulos CDR (Call detail recording), módulos CEL (Channel event logging), applications, channel drivers, códec translators, dialplan functions, addon modules, resource modules, PBX modules, entre otros. Si no se carga alguno de estos módulos o ninguno, es posible ejecutar Asterisk sin embargo no se podría realizar acciones ya que cada módulo está diseñado para cumplir alguna función específica. En este proyecto se utilizaron los módulos: chan_mobile.so que es un addon module, chan_oss.so que es un BPX module y el módulo chan_sip.so para pruebas con un softphone.

Es muy importante mencionar los addon modules o módulos adicionales que brinda el software. A continuación se hace un descripción breve de cada addon module en la tabla I.

TABLA I. Módulos adicionales de Asterisk.

Nombre del módulo	**Descripción**
app_mysql	Ejecuta queries MySQL con una aplicación de plan de marcado
cdr_mysql	Almacena logs de llamadas detallados en una base de datos MySQL
chan_mobile	Permite enviar y recibir llamadas de teléfonos móviles a través de un canal bluetooth.
chan_ooh323	Permite el envío y recepción de llamadas de VoIP usando el protocolo H.323
format_mp3	Permite a Asterisk la reproducción de archivos de audio MP3
res_config_mysql	Utiliza una base de datos MySQL como backend de configuración en tiempo real

Asterisk está compuesto de varias fuentes que utilizan el sistema de archivos de Linux para su arquitectura de archivos. Entre los principales archivos de configuración tenemos: extensions.conf, sip.conf y modules.conf [1]. Para este proyecto se hizo uso de los 3 archivos previamente mencionados,

además de los archivos oss.conf y chan_mobile.conf. Todos estos archivos se guardan en ruta /etc/asterisk.

El archivo extensions.conf es uno de los archivos más importantes de la PBX, ya que es el archivo en el cual se describe el plan de marcado. La configuración de este archivo es bastante sencilla y permite al administrador configurar extensiones puntuales, MACROS, ejecutar scripts a través de las funciones AGI (Asterisk Gateway Interface) y AMI (Asterisk Manager Interface) e inclusive enviar mensajes de texto a través de un dispositivo conectado a travésdel canal chan_mobile. En este archivo se definen todas las extensiones de la centralita agrupadas por contexto. Si se marca a la extensión de un usuario y esta no está agregada al contexto que le corresponde o no existe dicho contexto, entonces no se genera la llamada y se mostrará el log en la consola de Asterisk indicando el inconveniente. Cuando se define un usuario SIP, IAX, de consola, o cualquier tipo de usuario que se conecte con el servidor Asterisk se define el contexto que utilizará. Estos usuarios son definidos en su archivo de configuración correspondiente. Por ejemplo un usuario SIP será definido en el archivo sip.conf y en dicho archivo se define el contexto del usuario.

Es importante mencionar que Asterisk almacena todo tipo de logs, los cuales son mostrados en la consola. Estos logs se almacenan en la ruta /var/log/astersisk. En este directorio podemos encontrar logs de debug, logs de encolamiento, CDRs, CEL, mensajes, errores, entre otros.

Asterisk contiene ciertos recursos que requieren fuentes de datos externas. Estos archivos o datos externos son almacenados en el directorio /var/lib/asterisk. Dentro de este directorio existen varios subdirectorios donde se almacena archivos de audio como por ejemplo, música de llamada en espera o mensajes de audio, también se almacenan scripts que son utilizados por el AGI (Asterisk Gateway Interface) y algunos archivos extras que son utilizados por otros recursos de Asterisk [1].En general, podemos resumir los directorios usados por la PBX Asterisk, los cuales presentan la siguiente arquitectura (ver Figura 2.3)

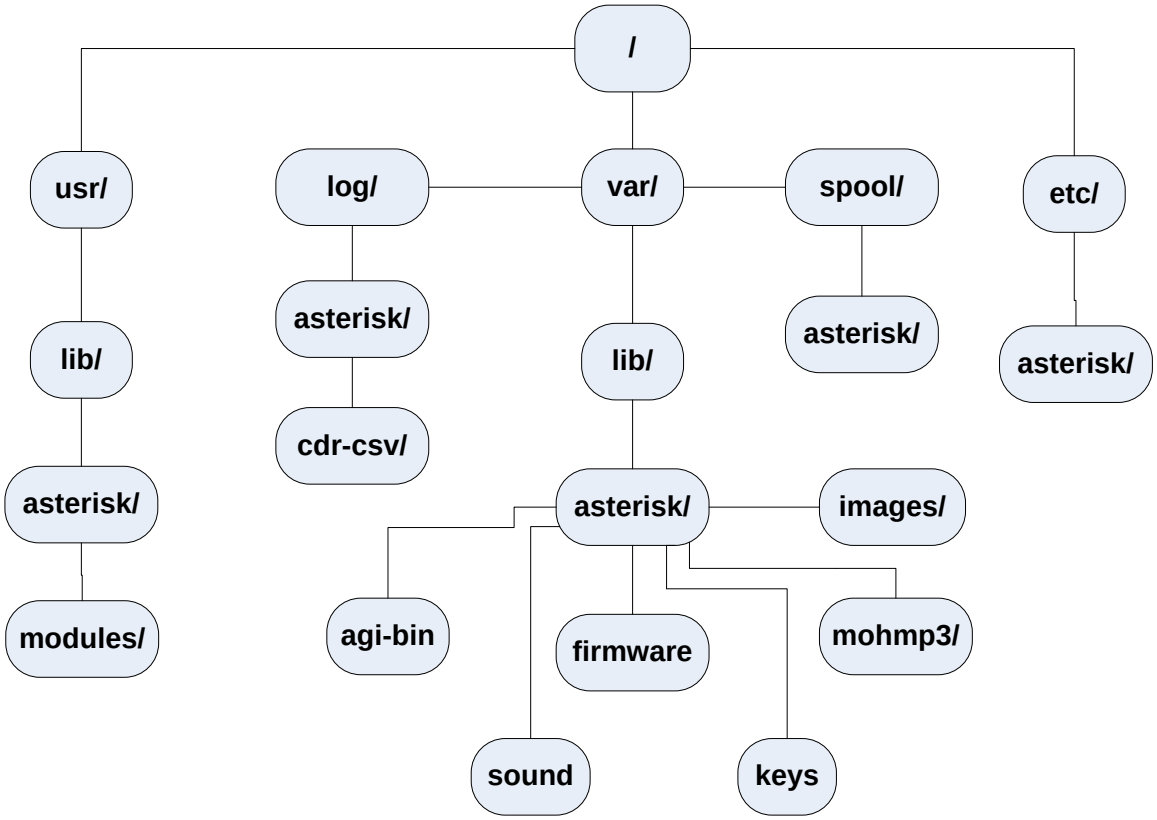


Figura 2.3. Directorios y subdirectorios usados por Asterisk

A continuación se muestra la tabla II que contiene la respectiva descripción de cada directorio o subdirectorio:

Tabla II. Descripción de directorios usados por Asterisk

Directorio (Ruta)	**Descripción**
/etc/asterisk	Contiene la mayor parte de archivos de configuración de Asterisk como extensions.conf, sip.conf, modules.conf, entre otros
/usr/lib/asterisk/modules/	Contiene todos los módulos de la PBX.
/var/lib/asterisk	Contiene varios subdirectorios que almacenan archivos que son usados por ciertos recursos de Asterisk.
/var/lib/asterisk/agi-bin	Contiene scripts que son usados por la función AGI
/var/lib/asterisk/sounds	Contiene archivos de audio que son usados para mensajes de voz, alertas, contestadoras, entre otros.
/var/lib/asterisk/firmware	Contiene imágenes binarias de firmware de varios dispositivos compatibles con Asterisk
/var/lib/asterisk/images	Contiene imágenes
/var/lib/asterisk/keys	Almacena claves RSA que se usan en conversaciones IAX2
/var/lib/asterisk/mohmp3	Contiene archivos de audio para llamadas en espera

Tabla II. Descripción de directorios usados por Asterisk

/var/log/asterisk	Contiene subdirectorios donde almacena todo tipo de logs
/var/log/asterisk/cdr-csv	Almacena CDRs en format comma-separated value o CSV
/var/spool/asterisk	Contiene los directorios outgoing/, qcall/, tmp/, y voicemail/, en los cuales Asterisk busca archivos de texto que contienen información respecto a requerimientos de llamdas

2.1.3 Protocolos de VoIP

Para establecer comunicación entre dos dispositivos terminales se necesita definir un conjunto de reglas para la negoción que se realiza en la comunicación. En el caso de la telefonía, si un teléfono quiere comunicarse con otro, debe existir un conjunto de reglas que definan como realizar una petición de llamada, como recibir una petición de llamada, como contestarla, como colgar la llamadas, entre otras acciones que se realizan para establecer una llamada. Este conjunto de reglas o parámetros que permiten establecer la comunicación se denomina señalización o señalización de llamada específicamente para telefonía.

En la telefonía IP, la voz se transmite codificada en paquetes. Los codificadores más utilizados hoy en día son G.729, G.711 y GSM. Es importante mencionar que en la mayor parte de los casos estos paquetes se transportan sobre segmentos UDP, lo que hace necesario la negociación de los puertos UDP donde el receptor espera recibir el audio. Para el intercambio de este tipo de información se definen los protocolos de control de señalización de llamada (Call control signaling). Luego de establecida la comunicación el audio viaja codificado en paquetes IP.

La transmisión de voz sobre IP tiene ciertas desventajas, ya que al ser transmitida en paquetes IP existe la posibilidad de tener un retardo, paquetes perdidos o paquetes desordenados, o inclusive degradación si los enlaces están saturados. Esto ocurre porque las redes IP no fueron diseñadas para transmitir voz. Sin embargo, existe un campo del paquete IP llamado DSCP o código de servicios diferenciados, el cual permite diferenciar el tipo de servicio y prioridad del paquete. En el caso de la voz, todos los paquetes tiene el valor hexadecimal EF en el campo DSCP. Esto permite que la transmisión de paquetes de voz sea identificada y tenga mayor prioridad.

Para controlar los efectos mencionados que puede ocasionar la transmisión de voz en paquetes IP existen los protocolos de transporte de media que están asociados con los protocoles de control de

transporte de media, los cuales envían a las terminales involucradas estadísticas de jitter, paquetes enviados, recibidos, perdidos, etc. Actualmente los protocolos mas utilizados para control de transporte de media son RTP y RTCP que están definidos por el RFC3550.

Utilizando los protocoles RTP o RTCP y alguno de los codificadores mencionados es posible transmitir voz sobre IP, sin embargo esto no es suficiente para una red que posee un gran número de terminales y dispositivos que realizan enrutamiento, trascoding de señalización o localización de dispositivos. Por otro lado es necesario establecer un protocolo para la conversación entre gateways. Por estas y otras razones se crearon protocolos de registración y control.

Para tener una red escalable y con un mejor control es necesario utilizar uno de estos protocolos. Los principales protocolos de VoIP utilizados son: SIP, H.323, MGCP, MEGACO / H.248 y el protocolo IAX (Inter-Asterisk-eXchange) definido por el mismo creador de Asterisk Mark Spencer. A continuación se describen brevemente algunos de los principales protocolos.

- **SIP** o Session Initiation Procolos, en español protocolo de inicio de sesiones desarrollado por la IETF, el cual define la iniciación, modificación y finalización de sesiones interactivas de usuario donde intervienen elementos multimedia como el video, voz, mensajería instantánea, entre otros. Este protocolo se encarga de la señalización

de llamada, registración y control pero se complemente con el protocolo SDP, el cual describe el contenido multimedia de la sesión, por ejemplo qué direcciones IP, puertos y códecs se usarán durante la comunicación. En otras palabras, SDP se encarga del control de señalización de llamada. Por otro lado del transporte de audio se encarga el protocolo RTP. Es decir que para establecer una sesión SIP, se utilizan también los protocolos SDP y RTP. SIP es un protocolo que adopta el modelo cliente – servidor y es transaccional. A nivel de capa 4, SIP utiliza por defecto el puerto 5060 en TCP y UDP.

- **H.323** es el stack de protocolos recomendado por la ITU-T que se utiliza para establecer los protocolos usados para la transmisión de voz y video sobre paquetes de red. Es utilizados por las aplicaciones Microsoft Netmeeting y Ekiga. Para las sesiones H.323 se utiliza el protocolo H.225 o Q.931 para la señalización de llamada. El protocolo H.225 también se utiliza para la registración y control, mientras que el protocolo H.245 se encarga del control de señalización de llamada. El transporte de audio se hace a través de RTP. Estos son los protocolos que usualmente se utilizan en una conversación H.323. aunque dentro del stack H.323 tambien se encuentran los protocolos H.235, H.450, H.239 y H.281. Es muy importante mencionar que este protocolo no

garantiza una buena calidad de servicio y el transporte de audio y video no es confiable.

- **IAX** o Inter-Asterisk eXchange es un protocolo creado por Mark Spencer de Digium, desarrollado específicamente para la comunicación entre servidores Asterisk. Es un protocolo robusto que maneja un gran número de codecs y streams. En comparación con otros protocolos de VoIP es bastante sencillo y su última versión es IAX2. Este protocolo utiliza un único puerto UDP que es generalmente el 4569 para comunicación entre terminales VoIP. IAX se encarga de la señalización de llamada, registración y control. A diferencia de otros protocolos que usan RTP para transporte, en IAX el tráfico es transmitido in-band. El principal objetivo de IAX ha sido minimizar el ancho de banda utilizado en la transmisión de voz y vídeo a través de la red IP.

  Asterisk soporta todos los protocolos antes mencionados. Aunque es recomendable utilizar IAX2 para la comunicación entre servidores Asterisk.

2.2 AGI. The Asterisk Gateway Interface

La interface de entrada de Asterisk,mejor conocida como AGI es una interface que permite a la PBX Asterisk interactuar con programas o scripts externos desarrollados en diversos lenguajes de programación. AGI permite que el plan de marcado pueda ser controlado por programas o scripts externos. Es importante saber que si el script realiza una llamada, se mantendrá la ejecución del mismo hasta que termine la llamada. Es decir que una vez que se cierra la llamada, el script deja de ejecutarse.

Una de las aplicaciones más comunes de AGI es la interacción de Asterisk con una base de datos externa. En este proyecto se utiliza AGI para la ejecución de scripts que realizan queries en una base de datos MySQL.

2.2.1 Funcionamiento de AGI

Los lenguajes de programación comúnmente utilizados para AGI son: Perl, PHP, Python, JAVA, Boune Shell, C, entre otros. Para este proyecto se desarrollaron scripts en PHP; este lenguaje fue una selección personal del desarrollador de los scripts.

Los scripts AGI se comunican con Asterisk usando los canales STDIN, STDOUT y STDERR que son los canales estándares

(standard input, standard output y standard error) utilizados por programas o aplicaciones para recibir información, enviar información y manejo de errores. Estos son los estándares utilizados en la mayoría de los ambientes Unix. En el caso de AGI funciona de la siguiente manera:

- Un script AGI lee desde un canal STDIN para obtener información de Asterisk
- Un script AGI escribe data sobre un canal STDOUT para enviar información a Asterisk
- Un script AGI podría escribir data sobre un canal STDERR para enviar información de logs o debug a la consola de Asterisk.

Un script AGI se ejecuta escribiendo la sentencia**AGI(nombre_del_script)** en el caso en que el script esta almacenado en el directorio de la siguiente ruta: /var/lib/asterisk/agi-bin. En el caso contrario, se deberá especificar la ruta. Por ejemplo: AGI(/home/user1/scrip1.sh).

Existe la alternativa de pasar parámetros al script colocándolos a continuación del nombre del script y separados por coma. Por ejemplo: AGI(script1.php,DATO1,DATO2). Para utilizar estos parámetros en el script se puede obtenerlos desde un arreglo de parámetros que se crea automáticamente. Haciendo

referencia al ejemplo anterior, en PHP por ejemplo, obtenemos los parámetros del arreglo argv de la siguiente manera:

DATO1 = $argv[1]

DATO2 = $argv[2]

Además de AGI() existen otras aplicaciones que básicamente cumple la misma función pero en circunstancias diferentes. Estas son: EAGI(), DeadAGI() y FastAGI(). A continuación se describen cada una:

- **EAGI()** conocida como enhanced AGI o AGI mejorado, funciona igual que AGI() con la diferencia que le permite al script leer el stream de audio entrante en el archivo descriptor número tres.
- **DeadAGI()** aplicación que funciona como AGI() pero sobre canales muertos, es decir canales en los que la llamada fue colgada. Con la aplicación AGI() no es posible realizar esto. En otras palabras, con DeadAGI() el script continuará ejecutándose aunque se cierre la llamada.
- **FastAGI()** es una aplicación que permite que un script AGI sea llamado desde cualquier servidor de la red. Es decir, que un solo archivo o script localizado sobre un servidor asterisk puede ser ejecutado desde otros servidores en red.

2.2.2 Características de AGI

AGI proporciona un conjunto de variables y funciones o métodos sumamente importantes que se deben tomar en cuenta cuando se ejecuta un script AGI. En primer lugar se tiene las variables que son pasadas por Asterisk, las cuales pueden ser utilizadas por el script a través del canal STDIN. Existen varias de estas variables que como estándar en su nombre se les antepone “agi_”.

Las principales variables de Asterisk usadas por el script son las siguientes:

- **agi_request** – Nombre del script
- **agi_channel** – El canal de origen
- **agi_language** – El código del lenguaje usado
- **agi_type** – El tipo de canal de origen (ej: SIP)
- **agi_uniqueid** – El UNIQUE ID de la llamada
- **agi_version** – La versión de Asterisk (desde Asterisk 1.6)
- **agi_callerid** – El número del CALLER ID (ó "unknown")
- **agi_calleridname** – El nombre del CALLER ID(ó "unknown")l
- **agi_callingani2** – El número definido en ANI2

- **agi_callington** – El tipo de números usados en PRI Channels
- **agi_callingtns** – Un número opcional de 4 dígitos (Transit Network Selector) usado en PRI Channels
- **agi_dnid** – El ID del número llamado (ó "unknown")
- **agi_context** – Contexto de origen en el archivo extensions.conf
- **agi_extension** – La extensión llamada

También existen variables que apuntan a directorios del sistema de archivos de Asterisk. Entre estas tenemos:

- AST_CONFIG_DIR
- AST_CONFIG_FILE
- AST_MODULE_DIR
- AST_SPOOL_DIR
- AST_MONITOR_DIR
- AST_VAR_DIR
- AST_DATA_DIR
- AST_LOG_DIR
- AST_AGI_DIR
- AST_KEY_DIR
- AST_RUN_DIR

También existe una variable de retorno del script AGI que es la variable “AGISTATUS” la cual retorna 3 tipos de status que son: SUCCESS, FAILURE, HANGUP.

Luego de conocer las variables que permiten la interacción entre Asterisk y otro lenguaje de programación, se mencionan las funciones, comandos o métodos que proporciona AGI. Entre los principales comandos tenemos:

- **answer**: Contesta llamada
- **channel status**: Retorna status del canal conectado
- **database del**: Borra un key/value de la base de datos
- **database deltree**: Borra un keytree/value de la base de datos
- **database get**: Obtiene un valor de la base de datos
- **database put**: Actualiza o añade valores en la base de datos
- **exec**: Ejecuta una aplicación.
- **get data**: Obtiene data de un canal
- **get variable**: Obtiene una variable del canal
- **hangup**: Cuelga el canal que se está usando
- **noop**: No ejecuta acción alguna
- **receive char**: Recibe un caracter de un canal que lo soporte
- **receive text**: Recibe texto de canales que lo soportan
- **say alpha**: Dice un caracteralfanúmerico

- **say date**: Dice un dato tipo fecha
- **say digits**: Dice un dígito
- **say number**: Dice un número
- **say phonetic**: Dice un caracter
- **say time**: Dice un dato tipo time
- **send image**: Envía imagenes a los canales lo soportan
- **send text**: Envía texto a canales que lo soportan
- **set callerid**: Estable el caller ID para el canal que se está usando
- **set context**: Establece el contexto del canal
- **set extension**: Cambia la extensión del canal
- **set music**: Habilita o deshabilita música en espera
- **set priority**: Da prioridad al canal
- **set variable**: Establece una variable del canal
- **speech activate grammar**: Activa una gramática (Desde version 1.6)
- **speech deactivate grammar**: Desactiva una gramática (Desde version 1.6)
- **speech load grammar**: Carga una gramática (Desde version 1.6)
- **speech recognize**: Reconoce un speech o parlamento.
- **speech unload grammar**: Descarga una gramática (Desde version 1.6)
- **stream file**: Envía un archivo de audio en el canal..
- **verbose**: Registra un mensaje en Asterisk

- **wait for digit**: Espera que se presione un digito.

Todas las variables y funciones mencionadas permiten que la interface AGI sea muy útil. Cabe recalcar que dependiendo del lenguaje escogido para implementar el script es necesario utilizar las respectivas librerías para poder utilizar todas las funciones y variables de AGI. Por ejemplo, en el caso de PHP, es necesario invocar a la librería phpagi que se la puede obtener a través del sitio web de phpagi [5].

Para este proyecto se implementaron scripts en PHP, por lo que a continuación se describe el encabezado de un script AGI en php. Las primeras 2 líneas que debe llevar el script son:

```
#! /usr/bin/php –q
<?php
```

La opción –q es muy importante ya que desactiva los mensajes de error HTML. No deben existir líneas extras entre la primera línea y la línea de inicio del código php.

Luego de creado el archivo de script se deben asignar permisos de ejecución. Esto lo podemos hacer ingresando al directorio en donde está el archivo y ejecutando el siguiente comando en la consola de Linux:

```
chmod 755 *.php
```

Luego de esto podemos ejecutar el AGI desde nuestro plan de marcado. En el capítulo 3 se describe la implementación de scripts en php utilizados para este proyecto.

2.3 CANAL DE ASTERISK CHAN MOBILE

Chan mobile es un módulo adicional de Asterisk o addon module que permite la interacción entre Asterisk y la red GSM (Global System for Mobile communications). En otras palabras, permite a la PBX Asterisk realizar y recibir llamadas hacia y desde teléfonos móviles conectados a la GSM, así como también mensajes de texto sms. El modulo Chan Mobile debe ser instalado cuando se realiza la instalación de Asterisk desde el Menuselect. En el capítulo 3 se describe en detalle la instalación de este módulo. Chan mobile conecta nuestra PBX Asterisk con uno o más teléfonos móviles que funcionarán como puertas de enlace de Asterisk hacia la red GSM. La conexión entre el servidor Asterisk y el o los teléfonos móviles es a través de bluetooth, por lo que es necesario utilizar un adaptador bluetooth USB en el servidor.

La configuración del módulo es sumamente sencilla. Existe solo un archivo de configuración que es denominado chan_mobile.conf a partir de la vesión 1.6. En versiones anteriores a las 1.6 existen módulos previos a chan_mobile. En las primeras versiones de utilizaba el paquete

chan_bluetooth de Theo Zourzouvillys que con el pasar del tiempo quedo como un paquete desactualizado y entonces se empezó a desarrollar el módulo chan_cellphone que luego se convirtió en módulo chan_mobile que es el que actualmente se utiliza. Este módulo fue escrito por David Bowerman y está disponible vía SVN. La documentación actualizada sobre chan mobile se puede obtener de http://svn.digium.com/svn/

Como ya se indicó, la configuración del archivo es bastante sencilla, sin embargo los problemas que pueden presentarse al utilizar este módulo son problemas de compatibilidad a nivel hardware o software (driver del adaptador bluetooth).

2.3.1 Descripción funcional de Chan Mobile

Luego de instalado el módulo, es necesario asegurarse de que el módulo este cargado. Para cargar el módulo se usa el comando: **module load chan_mobile.so.** Luego, se debe verificar que exista el archivo chan_mobile.conf dentro del directorio /etc/asterisk. En este archivo se describe el hardware que se va a utilizar para que pueda ser reconocido. Este hardware incluye al adaptador bluetooth utilizado y todos los teléfonos que se conectarán al servidor Asterisk. Si el adaptador bluetooth no está descrito en el archivo de configuración, no se reconocerá teléfono alguno, ya que es a través del mismo como Asterisk los reconoce. Los parámetros

que se definen en el archivo de configuración son básicos, entre ellos los más importantes, el nombre del dispositivo y su dirección MAC. Si no existen problemas de compatibilidad con el driver del bluetooth y con los dispositivos a utilizar basta con describirlos en el archivo chan_mobile.conf para empezar a utilizarlos. Luego, solo se debe configurar el plan de marcado. A continuación se describen casos de uso de este módulo.

Si se quiere llamar desde un teléfono móvil conectado a la red GSM podemos marcar a uno de los teléfonos Gateway conectados a la PBX y entonces el teléfono Gateway pasará el requerimiento a la PBX y se buscará el contexto por defecto “default” en el cual se buscará la extensión “s” por defecto. Conociendo hacia que contexto y extensión se dirige la llamada podemos realizar varias acciones en el plan de marcado, como por ejemplo: redirigir la llamada a una extensión interna, enrutar la llamada hacia otra red IP, ejecutar algún script, entre otras.

También es posible generar llamadas hacia la red GSM desde alguna extensión interna. Por ejemplo: se puede definir una extensión cualquiera en el plan de marcado que cuando se marque desde un teléfono interno se redirija la llamada a la red GSM a cierto número. Por otro lado, podemos hacer que al marcar una

extensión se envíe un mensaje de texto o sms a cierto número de la red de telefonía móvil. Realmente existen muchos casos en los que podemos utilizar este módulo adicional.

Chan mobile es una herramienta bastante útil y dependiendo del propósito con que se la quiere usar se configura el plan de marcado.

2.3.2 Compatibilidad de Características

Antes de mencionar la compatibilidad de Chan mobile, se describen sus principales características, entre las cuales tenemos:

- Conexión permitida a múltiples teléfonos celulares
- Conexión permitida a múltiples adaptadores bluetooth. Cabe recalcar que en la definición de cada teléfono en el archivo chan_mobile.conf se describe el adaptador bluetooth que usará para conectarse. Es decir que cada celular tiene un adaptador bluetooth destinado (varios teléfonos pueden trabajar con el mismo bluetooth).
- Puede hacer una búsqueda de dispositivos móviles con bluetooth disponibles a través de un adaptador bluetooth desocupado. En esta búsqueda muestra la dirección MAC de cada dispositivo y el canal que usa en la conexión bluetooth cada uno.

- Identificación de llamadas a través de llamadas entrantes
- Envío y recepción de mensajes de texto sms. El envió lo realiza con la aplicación MobileSendSMS.
- Las llamadas entrantes son manejadas por Asterisk, lo que permite realizar cualquier acción en el plan de marcado con estas llamadas.
- Existe la aplicación cellstatus para ser utilizada en el plan de marcado, la cual permite conocer el status de un teléfono móvil conectado a la PBX, es decir si está disponible, ocupado o en el peor de los casos desconectado.
- Se puede utilizar un auricular bluetooth como extensión de acceso telefónico en el plan de marcado

Todas las características mencionas son válidas siempre y cuando no existan problemas de compatibilidad.

No todos los adaptadores bluetooth USB son compatibles con Asterisk y Chan Mobile. Así como también los drivers utilizados. Para este proyecto se utilizó el driver **BlueZ**que es para plataforma Linux**.** También aplica lo mismo para los teléfonos con bluetooth. No todos los teléfonos con bluetooth son compatibles con chan mobile, y algunos, son parcialmente compatibles [3]. Se puede encontrar una lista de compatibilidad en la web de voip-info [3], sin embargo la

alteración del hardware en ciertos modelos que aparentemente son compatibles puede cambiar esto; esto curre porque muchos de los modelos listados están descontinuados y son reparados o reconstruidos. En esta lista se detalla que tipo de compatibilidad poseen ciertos teléfonos. Por ejemplo: El Nokia 6131 puede enviar mensajes de texto, realizar llamadas y recibir llamadas pero no puede recibir mensajes de texto.

El tema de los sms es realmente crítico ya que la mayor parte de los teléfonos compatibles solo permiten hacer y recibir llamadas más no mensajes sms, y ciertos teléfonos permiten solo enviar mensajes.

2.4 MóduloOSS

OSS (Open Sound system)es una interfaz estándar de captura y envío de sonidos en sistemas operativos tipo UNIX. En otras palabras, oss es utilizado como driver de audio en sistemas tipo UNIX.

En Asterisk existe el módulo OSS o Chan_oss que puede ser utilizado siempre y cuando el servidor utilice el driver de audio oss. Este módulo es utilizado para realizar y recibir llamadas desde la consola de Asterisk usando dispositivos de audio como audífonos y micrófonos en lugar de un teléfono. Hoy en día existen los auriculares tipos diadema que son

audífonos con un micrófono incorporado, y son muy útiles para este caso. Para cargar este módulo se usa el comando **module load chan_oss.so**desde la consola de asterisk.

Es un módulo bastante sencillo cuyo archivo de configuración se denomina oss.conf. En este archivo se define la extensión, con su correspondiente contexto que puede ser marcada desde la consola de Asterisk.

2.4.1 Descripción funcional y características

Este módulo cuenta con comandos para contestar, colgar o realizar llamadas desde la consola de Asterisk. En este proyecto fue necesario utilizar este módulo para ejecutar una tarea automática que marque una extensión desde la consola de Asterisk con el fin de correr un script AGI.

Chan Oss permite utilizar audífonos tipo diadema como un teléfono siempre y cuando el usuario este dentro de la consola de Asterisk. Es decir que la consola de Asterisk actúa como un softphone, pero utilizando el canal chan_oss donde se transmite el audio a través del driver OSS instalado en el servidor Asterisk.

Si se quiere marcar una extensión desde la consola de Asterisk, la extensión debe estar definida en el archivo oss.conf que se

encuentra dentro del directorio /etc/asterisk. En este archivo se definen todas las extensiones de consola con su respectivo contexto, luego deben ser configuradas en el plan de marcado, es decir el archivo extensions.conf para poder ser utilizadas. Para marcar una extensión desde la consola se ejecuta el comando **console dial "extension"**. Por ejemplo: console dial 555,y si esta definida en el plan de marcado y el archivo oss.conf, entonces se ejecutará la acción que se configuró en extensions.conf.

A continuación se menciona una aplicación muy útil para este módulo. Utilizando chan_oss se puede ejecutar extensiones del plan de marcado desde la consola de Linux, ya que existe el comado **asterisk –rx '*comando_de_asterisk*'.** Por ejemplo: asterisk.-rx 'console dial 100'. Esta sentencia ejecutada desde la consola de Linux realiza la ejecución del comando console dial 100 como si estuviese dentro de la consola de Asterisk, en otras palabras llama a la extensión de consola 100 del plan de marcado, en la cual se puede redirigir la llamada a cualquier otra extensión o ejecutar alguna acción. En este proyecto se aplicó exactamente lo mismo, con la diferencia que la ejecución del comando en la consola de Linux es automática y se realiza cada 5 minutos. En esto caso lo que se hace en plan de marcado es ejecutar un script AGI escrito en PHP.

CAPITULO 3

3. Implementación del sistema

La implementación de este sistema en general se divide en 2 partes que son: hardware y software. Respecto al hardware se utilizó un ordenador de escritorio con un procesador de 2.66 GHz, 1 GB de memoria RAM y 40 GB de disco duro, esto se detallará en el subcapítulo 3.2.1. Además del servidor se utilizó un adaptador Bluetooth que se conecta a un puerto USB del servidor, y un teléfono celular Nokia 6131, el cual dispone de una capacidad de conexión Bluetooth.

En lo referente al software, para el servidor se instaló la distribución Ubuntu versión 11.04 del sistema operativo Linux. Sobre este sistema se instaló una base de datos MySQL, y como driver de audio del sistema se instaló OSS. En lo referente al adaptador Bluetooth, se utilizó el driver BlueZ que es un driver Bluetooth diseñado para sistemas Linux. Luego de tener el sistema operativo instalado con la base de datos, el driver de audio y el driver Bluetooth, se procedió a instalar el software Asterisk versión 1.6 con su respectivo módulo Chan Mobile. Además de esto se

desarrolló un programa sencillo en JAVA para manejar el ingreso de datos y consultas en la base de datos del sistema. En los siguientes subcapítulos se describe en detalle todo el hardware y software de este sistema.

3.1 Hardware del sistema

3.1.1 Hardware del servidor

Antes de entrar en detalle del hardware que posee el servidor utilizado para la implementación de este sistema es importa conocer los requerimientos de hardware de Asterisk de acuerdo al tipo de sistema a implementar. Esto se detalla en la tabla III.

Tabla III. Requerimientos mínimos de hardware para Asterisk

Propósito	Número de canales	Hardware mínimo requerido
Sistema sencillo	Máximo 5	400 Mhz x86, 256 MB RAM
Sistema SOHO (small office/home office)	5 a 10	1 Ghz x86, 512 MB RAM
Sistema para empresas pequeñas	Máximo 25	3 Ghz x86, 1 GB RAM
Sistema para empresas medianas o grandes	más de 25	dual CPUs, posiblemente varios servidores en arquitectura distribuida

Tomando en cuenta la tabla 3 se puede concluir que bastaría con el hardware mínimo para un sistema sencillo, ya que el número de canales utilizados esmenor a 5. En este caso, se utilizan 2 canales, que son el canal de consola que se utiliza constantemente y el canal chan mobile en cada llamada que se realiza al Gateway y cada envío de mensaje.

Dado que el hardware mínimo hace referencia a un procesador de 400 MHz y 256 MB de RAM, lo recomendable sería un hardware superior a este. Por ejemplo: procesador de 1 GHz y 512 MB de RAM. Con estas características bastaría para que Asterisk funcione sin problemas, sin embargo es importante tomar en cuenta el resto de procesos que debe realizar el procesador como por ejemplo la ejecución de tareas automáticas cada 5 minutos, la ejecución del programa para manejar el ingreso de datos y consultas en la base de datos, la ejecución de scripts desarrollados en PHP y uso de la base de datos del sistema. Debido todos estos parámetros se decidió utilizar un procesador que opere a una frecuencia superior a 1 GHz.

En este caso se utilizó un procesador Intel(R) Celeron(R) que trabaja a 2.66 GHz. Un procesador de un solo núcleo que

cuenta con 128 KB de cache. En sistemas más complejos es importante utilizar procesador duales o de varios núcleos.

La memoria RAM del servidor es de 1 GB y el disco duro es de 40 GB. La capacidad de disco duro es importante porque se maneja una base de datos que va creciendo constantemente. Para poner en producción a este sistema es recomendable utilizar un disco duro de mayor capacidad, esto se mide de acuerdo a la demanda que tendrá el sistema. En realidad, lo recomendable es un usar un hardware diseñado para trabajar como servidor.

3.1.2 Adaptador Bluetooth USB

Este adaptador permite al servidor realizar una conexión vía bluetooth con otro dispositivo. Bluetooth es un estándar utilizado para redes inalámbricas de área personal o WPAN, a través del cual se transmite voz y datos mediante un enlace por radiofrecuencia en la banda ISM de 2.4 GHz para distancias cortas. Existen 4 versiones de este estándar que varían por la tasa de transferencia máxima que brindan que va desde 1 Mbps hasta 24 Mbps de acuerdo a la versión. El adaptador se conecta en un puerto USB.

Hoy en día, se puede varios de estos adaptadores bluetooth USB en el mercado fabricados por distintas marcas. Para este

proyecto se utilizó un adaptador D-link modelo DBT-120 (ver Figura 3.1). A continuación se describen las características del mismo:

- Conexión USB 2.0
- Conexión bluetooth de hasta 9 metros de alcance sin obstáculos o interferencias.
- Para mantener segura la información que se transmite, utiliza un cifrado de 128 Kbits y la técnica de modulación FHSS (Frecuency Hopping Spread Spectrum).
- Diseño ultra compacto y portable
- Versión de bluetooth 2.0. Es versión permite una tasa de transferencia es hasta 3 Mbps.

Fuente: http://www.dlink.com/products/?pid=34

Figura 3.1. Adaptador Bluetooth USB D-link DBT-120

3.1.3 Teléfono Gateway

Este dispositivo es un teléfono móvil o celular que funciona como puerta de enlace de Asterisk hacia la red GSM. Este teléfono celular cuenta con capacidad de conexión bluetooth lo cual es necesario para establecer la conexión con el servidor Asterisk.

Durante el desarrollo de este sistema se realizaron varias pruebas con distintos teléfonos, encontrando muy pocos compatibles con Chan Mobile y que permitan enviar mensajes de texto. Entre los teléfonos que se probaron y que cuentan con la capacidad de enviar sms tenemos:

- Nokia 6131
- Nokia 6021
- Nokia 6230i

Estos teléfonos a diferencia de otros como el nokia E71 o el nokia c3 tiene la capacidad de enviar mensajes sms. Aunque, también pueden recibir y generar llamadas al igual que los modelos nokia E71 y nokia c3. Por otro lado, no se encontró un teléfono compatible con Chan Mobile que permita recibir mensajes sms.

El teléfono que se utilizó en el sistema implementado es el **nokia 6131**(ver Figura 3.2) que es un modelo que ya no se fabrica. Esta etapa de pruebas con distintos dispositivos nos hizo concluir que la

compatibilidad del módulo Chan Mobile es bastante pobre, aunque basta con tener 1 o 2 teléfonos compatibles para poder utilizar el módulo. Es importante mencionar que se debe configurar el teléfono para que la conexión bluetooth con el servidor sea automática. Esto se hace cuando se establece la primera conexión entre el teléfono y el servidor.

Fuente: http://cellphones.about.com/library/bl-pi-nokia_6131.htm

Figura 3.2 Teléfono NOKIA 6131

3.2 Software del sistema

3.2.1 Sistema Operativo del servidor

Debido a que Asterisk fue desarrollado originalmente para plataforma Linux y existe un mejor soporte sobre la misma, se tomó la decisión de utilizar un sistema Linux. Se instaló un Linux versión 2.6.35-32. En este caso se instaló la distribución Ubuntu versión 11.04.

Ubuntu está basado en la distribución Debian y se enfoca más en la facilidad de uso para el usuario. En otras palabras, es más amigable para el usuario. El sistema es mantenido por Canonical y la comunidad de programadores. Cada 6 meses se pública una nueva versión de Ubuntu que recibe su respectivo soporte de Canonical.

Sobre este sistema se instaló el driver de audio OSS, cuya instalación se detalla en el subcapítulo 3.3.4. Adicionalmente, previo a la instalación de Asterisk se instaló el driver bluetooth y la base de datos MySQL. Es importante instalar primero los drivers de los canales a utilizar y la base de datos antes de instalar Asterisk. Si la instalación de drivers o bases de datos se realiza después de instalar Asterisk, se pueden tener problemas con la activación de los canales adicionales que se desea utilizar y va ser

necesario desinstalar Asteirsk para rehacer el procedimiento siguiendo el orden correcto.

3.2.2 Driver de adaptador Bluetooth

El driver Bluetooth utilizado para este proyecto se denomina **BlueZ.**Este es un driver desarrollado para las conexiones Bluetooth en los sistemas Linux. De hecho, es el stack de protocolos Bluetooth oficial de Linux. Está desarrollado para poder ser utilizado con múltiples adaptadores Bluetooth. Actualmente, funciona casi que con cualquier adaptador bluetooth. Esto se comprobó cambiando de adaptador bluetooth con el mismo driver instalado. El driver puede ser descargado desde la web de Bluez [6].

Para instalar este driver, se debe instalar los siguientes paquetes:

- Gnome-bluetooth
- Bluez-utils
- Bluez-pin
- Bluez-passkey-gnome

Para esto se puede ejecutar el siguiente comando desde la consola de Ubuntu:

- **sudo apt-get install gnome-bluetooth bluez-utils bluez-passkey-gnome bluez-pin**

Luego de instalar el driver BlueZ se puede conectar el adaptador USB y se verifica que sea reconocido por el sistema. Una forma rápida es identificar el ícono con el logotipo de Bluetooth en la barra de tareas de Ubuntu (ver Figura 3.3). Cuando el dispositivo está conectado y es reconocido por el sistema operativo debe aparecer este ícono.

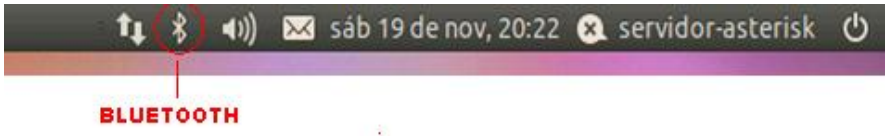


Figura 3.3 Ícono de Bluetooth en la barra de tareas de Ubuntu

Luego para confirmar que el dispositivo ha sido aceptado por el sistema y está funcionando se puede revisar la configuración del bluetooth y realizar un escaneo desde la consola de Linux, o desde la ventana de preferencias de bluetooth para buscar dispositivos con capacidad bluetooth disponibles.

Para revisar la configuración de bluetooth en Ubuntu, debe hacer click sobre el menú Sistema → Preferencias → Bluetooth y entonces aparecerá una nueva ventana de preferencias de bluetooth (ver Figura 3.4) donde se muestran las opciones de configuración del Bluetooth. A través de esta ventana de configuración también podemos añadir dispositivos bluetooth disponibles para establecer una conexión.

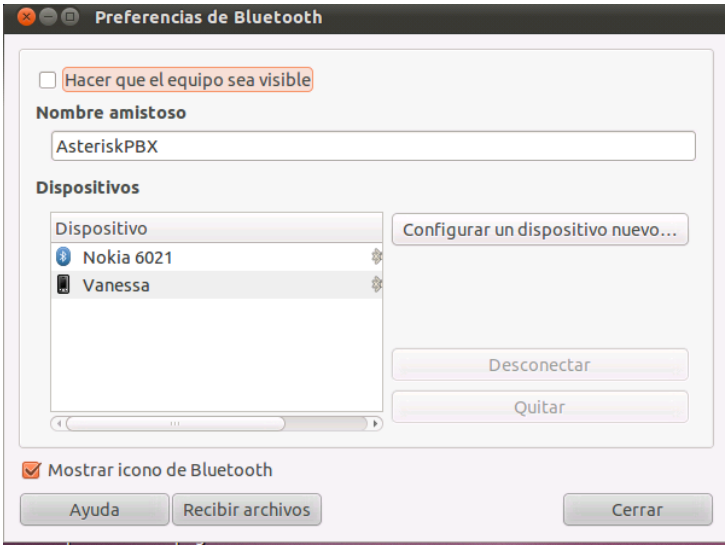


Figura 3.4 Ventana de preferencias de bluetooth en Ubuntu

Para realizar un escaneo desde la consola de Linux se debe escribir el siguiente comando:

- hcitool scan

Luego de esto aparecerán los dispositivos disponibles con su respectiva dirección MAC. Podemos realizar este escaneo varias veces (ver Figura 3.5).

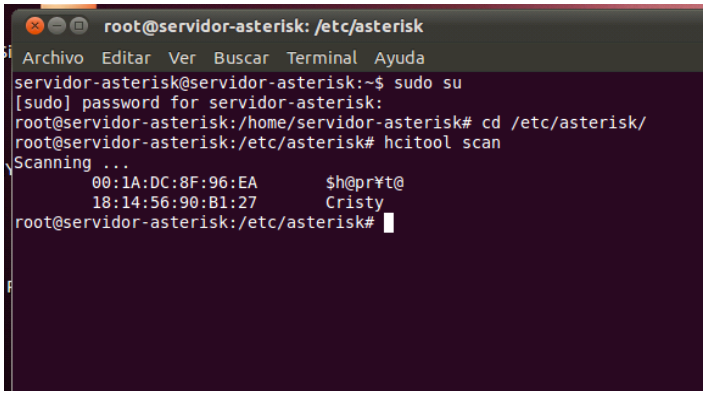


Figura 3.5. Escaneo de dispositivos bluetooth disponibles desde consola

Este escaneo es muy útil ya que para realizar la configuración de Chan mobile se necesita la dirección MAC del dispositivo, la cual se obtiene luego del escaneo.

La conexión entre el servidor y el teléfono Gateway se hace con la ventana de preferencias de bluetooth. Esta conexión se detalla en el subcapítulo 4.1

3.2.3 Base de datos del sistema

Para llevar un control y mejor administración del sistema fue necesario utilizar una base de datos. Se escogió utilizar una base MySQL. Esta base de datos es bastante sencilla pero es fundamental para el funcionamiento del sistema ya que almacena todas las compras realizadas por los clientes así como también los mensajes de texto a ser enviados y los números de los clientes. A través de esta base de datos el sistema sabe que compras están pendientes, es decir que

peticiones de clientes no han sido respondidas para luego responderlas usando un script que se ejecuta automáticamente desde la consola de Asterisk. Esta base de datos presenta el modelo descrito en la figura 3.6.

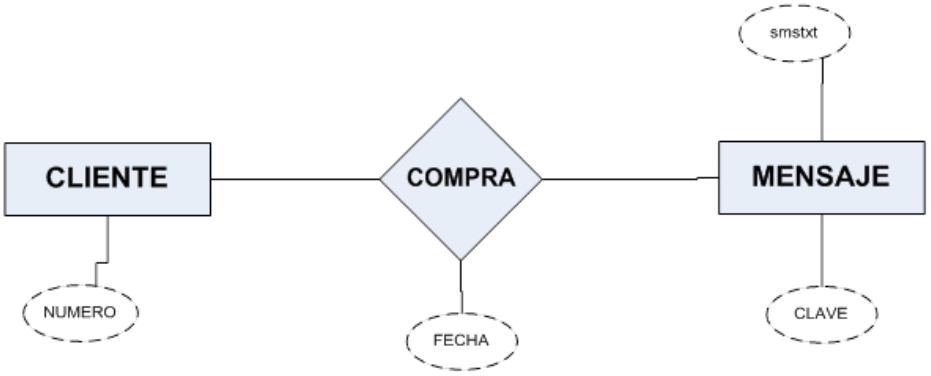


Figura 3.6 Modelo E/R la de base de datos del sistema

La relación entre el cliente y el mensaje es de N:M, por lo tanto se crearon 3 tablas. Desde la consola de Linux se puede ingresar a la base de datos, mostrar sus tablas y describir cada una de ellas. A continuación se muestra dicha información:

Tablas de la base de datos "test"

```
mysql> show tables;
+----------------+
| Tables_in_test |
+----------------+
| cliente        |
| compra         |
| mensaje        |
+----------------+
3 rows in set (0.00 sec)
```

Descripción de la tabla cliente

```
mysql> describe cliente
    -> ;
+--------+------------+------+-----+---------+-------+
| Field  | Type       | Null | Key | Default | Extra |
+--------+------------+------+-----+---------+-------+
| numero | varchar(9) | NO   | PRI | NULL    |       |
+--------+------------+------+-----+---------+-------+
1 row in set (0.00 sec)
```

Descripción de la tabla compra

```
mysql> describe compra;
+-----------+-------------+------+-----+-------------------+----------------
+
| Field     | Type        | Null | Key | Default           | Extra
|
+-----------+-------------+------+-----+-------------------+----------------
+
| id_compra | int(11)     | NO   | PRI | NULL              | auto_increment
|
| numero    | varchar(9)  | NO   | MUL | NULL              |
|
| clave     | varchar(11) | NO   | MUL | NULL              |
|
| fecha     | timestamp   | NO   |     | CURRENT_TIMESTAMP |
|
| status    | varchar(9)  | NO   |     | NULL              |
|
+-----------+-------------+------+-----+-------------------+----------------
+
5 rows in set (0.01 sec)
```

Descripción de la tabla mensaje

```
mysql> describe mensaje;
+--------+--------------+------+-----+---------+-------+
| Field  | Type         | Null | Key | Default | Extra |
+--------+--------------+------+-----+---------+-------+
| clave  | varchar(9)   | NO   | PRI | NULL    |       |
| smstxt | varchar(160) | NO   |     | NULL    |       |
+--------+--------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Como se puede observar en la tabla compra existe el campo status, el cual permite llevar un control de las compras realizadas como estado ok o estado pendiente. Esta base de datos permite llevar una mejor administración y control total del sistema porque además ayudar a su funcionamiento me permite hacer consultas interesantes. Por ejemplo, si un cliente quiere saber acerca de todos los mensajes que ha comprado durante los últimos 15 días, o el administrador del sistema quiere saber cual es el cliente que más compra o cual es el mensaje más comprado, entre otras. Todo esto lo puedo saber realizando simples consultas SQL. Por esta razón, se decidió implementar una aplicación sencilla que permita hacer estas consultas sin necesidad de tener conocimientos técnicos de SQL o base de datos, es decir que cualquier usuario pueda realizar consultas e ingresar nueva información de los mensajes de texto. Esta aplicación fue desarrollada en JAVA y puede realizar las siguientes acciones:

- Actualización de los mensajes de texto de cada palabra clave. La aplicación valida que el mensaje sea máximo de 160 caracteres que es la cantidad de caracteres estándar que soporta un sms.
- Consultas por fecha. Ver tabla de compras por fecha. Buscar compras de un cliente por fecha, palabras clave por fecha.
- Consultas por cliente. Ver tabla de clientes. Buscar compras realizas por un cliente. Palabras clave solicitadas por cliente.

- Consultas de mensajes. Ver la tabla mensajes. Buscar mensaje por palabra clave.
- Consultas extras: cliente que más compras realiza en un determinado rango de fecha, palabra clave más comprada en un determinado rango de fecha.
- Configuración de tiempo de expiración de búsqueda de los últimos mensajes pendientes. Se puede configurar un valor de tiempo en minutos que por defecto es 60 minutos para que la búsqueda de mensajes pendientes busque aquellas compras pendientes de los últimos n minutos.

Para instalar la base de datos MySQL sobre Ubuntu se deben instalar los paquetes: mysql-sever y mysql-client. Para esto se ejecuta el siguiente comando desde la consola de Ubuntu:

sudo apt-get install mysql-server mysql-client

Mientas se realiza la instalación, se pide una contraseña de administrador la cual debemos escribir para que termine la instalación y listo. Se puede utilizar MySQL desde la consola de Linux o también se puede instalar la interfaz gráfica de administración con el paquete mysql-admin que es bastante amigable.

Otra manera de instalar MySQL en Ubuntu 11.04 es utilizando el administrador de paquetes Synaptic y buscando el paquete MySQL

server, luego se escoge el paquete y lo aplicamos, así mismo, se pide ingresar una contraseña de administrador para la base.

Cabe recalcar que al crear la base de datos se debe crear un usuario con todos los privilegios para acceder a la base y realizar cualquier acción sobre la misma. Este usuario será el que utilizarán los scripts AGI para conectarse a la base de datos y poder ejecutar consultas o actualizaciones.

3.2.4 Driver de audio OSS

Cuando se instala Ubuntu, el sistema operativo instala un driver de audio por defecto, este se denomina ALSA (Advanced Linux Sound Architecture). En la mayoría de los casos ALSA funciona sin problemas. En este caso, para el servidor utilizado en este proyecto el driver ALSA presentaba problemas de compatibilidad con la tarjeta de audio instalada. Es por esto que se decidió desinstalar ALSA y en lugar de este instalar el driver OSS, siglas que en español significan “Sistema de Sonido Abierto”, el cual fue desarrollado como driver de audio para sistemas Unix. OSS es un driver de nivel bajo comparado con ALSA, sin embargo para este proyecto no se necesita de un driver de audio complejo, sino un driver que se adapte bien a la tarjeta de audio y funcione. Para Asterisk no existe problema ya que contiene un

módulo para ALSA y otro para OSS, pero cualquiera de los dos funcionará siempre y cuando el driver sea compatible con el hardware instalado. En este caso, fue necesario recurrir a OSS y el cambio se realizó con éxito, es decir luego de instalar OSS la tarjeta de audio del servidor funcionó correctamente.

Para reemplazar el driver ALSA por OSS se siguieron los siguientes pasos:

I. Remover PulseAudio que es el servidor de audio por defecto instalado en Ubuntu a partir de la versión 8.10. Para esto ejecutamos el siguiente comando:

sudo apt-get purge pulseaudio gstreamer0.10-pulseadio

II. Remover los paquetes del driver ALSA con el siguiente comando:

sudo /etc/init.d/alsa-utils stop

sudo apt-get remove alsa-base alsa-utils

III. Configurar OSS como el driver de audio por defecto de Ubuntu con el siguiente commando:

sudo dpkg-reconfigure Linux-sound-base

Aparecerá un cuadro con un mensaje que debemos aceptar para luego escoger OSS como driver de audio y aceptar el cambio dentro de la ventana que aparece

IV. Instalar paquetes previos necesarios para OSS con el siguiente comando:

sudo apt-get install –y esound esound-clients esound-common libesd0

En este caso son los paquetes que necesita GNOME a partir de la versión 2.24. Si Ubuntu está trabajando con GNOME 2.22 o versiones inferiores se quieren de otros paquetes.

V. Instalar driver OSS. Para esto se descarga el archivo .DEB del repositorio [7]. Luego se ingresa al directorio donde está guardado el archivo .DEB y se ejecuta el siguiente comando para instalarlo:

sudo dpkg –i oss-linux*.deb

Luego de instalarlo se reinicia el servidor y se prueba el funcionamiento de la tarjeta de audio reproduciendo un archivo de audio y si se tiene algún micrófono se puede grabar algo de prueba.

3.3 Instalación de Asterisk

Antes de instalar la PBX Asterisk es recomendable instalar la base de datos MySQL, el driver del adaptador bluetooth y el driver de audio OSS para no tener problemas de reconocimiento de los diferentes canales y módulos a utilizar de Asterisk. La instalación de Asterisk se hace desde la consola de Ubuntu, es decir vía línea de comandos, es por esto que todos los comandos que son mencionados, son ejecutados desde la terminal o consola de Ubuntu. A continuación se detalla la instalación de Asterisk 1.8 sobre Ubuntu 11.04.

Primero se debe realizar una actualización del sistema y luego reiniciar el servidor para hacer efectivo cualquier cambio en la actualización. Para esto se ejecuta el siguiente comando desde la consola:

```
sudo apt-get update && sudo apt-get upgrade &&
sudo reboot
```

3.3.1 Instalación de componentes básicos

Para la instalación de Asterisk es necesario instalar previamente ciertos componentes básicos. Primero se debe sincronizar el tiempo del sistema e instalar el servicio NTP (Network Time Protocol), para lo cual se ejecuta el siguiente comando:

```
sudo apt-get install ntp
```

Luego se instalan las dependencias de Asterisk que son componentes, los cuales necesita la PBX para funcionar. Se ejecuta el siguiente comando:

```
sudo apt-get install build-essential
subversion libncurses5-dev libssl-dev libxml2-
dev vim-nox
```

3.3.2 Instalación de la PBX Asterisk

Existen tres archivos fuentes creados por Digium que deben ser instalados para que Asterisk funcione sin problemas, estos archivos debe ser instalados en el siguiente orden:

1) LibPRI
2) DAHDI
3) Asterisk

Estos son los archivos fuentes que deben ser descargados e instalados en el orden propuesto. También se pueden instalar vía SVN.

Para proceder con la instalación de la PBX primero se crean directorios para descargar e instalar los archivos de Asterisk. Estos

pueden ser creados en la ruta que el usuario desee, sin embargo se acostumbra a crearlos en la siguiente ruta:

```
$ mkdir –p ~/src/asterisk-complete
```

Luego nos cambiamos al directorio creado ejecutando el comando:

```
$cd ~/src/asterisk-complete
```

LibPRI es una librería que da soporte a Asterisk para la conexión con la red digital de servicios integrados o ISDN (Integrated Services Digital Network) y sus interfaces PRI y BRI. LibPRI es una librería opcional, que en este caso no es necesaria instalarla, sin embargo es recomendable instalarla aunque no se lo vaya a utilizar para que nuestra PBX sea una solución escalable. La instalación de este componente es rápida. Se puede realizar vía SVN (Subversión). Para esto, se ejecutan los siguientes comandos:

```
$cd ~/src/asterisk-complete/
$mkdir libpri
$cd libpri
$svn co
http://svn.asterisk.org/svn/libpri/tags/1.4
$cd 1.4
$make
```

```
$sudo make install
```

Luego, se procede a instalar la interface DAHDI (Digium Asterisk Hardware Interface), que es el software que Asterisk utiliza para interactuar con el hardware de telefonía. Se recomienda instalarla aunque no se tenga hardware instalado, debido a que DAHDI es una dependencianecesaria para la construcción del módulo de sincronización res_timing_dahdi y se utiliza para aplicaciones del plan de marcado de Asterisk como MeetMe().

Para instalar DAHDI se debe verificar que la versión del kernel que está siendo utilizada coincida con la fuente de kernel que está siendo instalada. Se puede utilizar el comando: **`uname –a`** para ver la versión que está siendo utilizada. Para la instalación de la fuente de kernel se utiliza el siguiente comando:

```
sudo apt-get install linux-headers-`uname -r`
```

Usando el comando **`uname –r`** entre comillas simples se verifica que la fuente de kernel coincida con la versión que está siendo utilizada. Luego de esto descargamos el archivo fuente vía SVN (Subversión) de DAHDI y lo instalamos ejecutando lo siguiente:

```
$cd ~/src/asterisk-complete/
```

```
$mkdir dahdi

$cd dahdi

$svn co http://svn.asterisk.org/svn/dahdi/linux-complete/tags/2.4.0+2.4.0

$cd 2.4.0+2.4.0

$make

$sudo make install

$sudo make config
```

Una vez instalados LibPRI y DAHDI, se procede con la instalación del archivo fuente de Asterisk. Para lo cual se obtiene el código fuente vía SVN (Subversion) con los siguientes comandos. Es posible que tome algunos minutos realizar las descargas respectivas.

```
$cd ~/src/asterisk-complete/

$mkdir asterisk

$cd asterisk

$ svn co
http://svn.asterisk.org/svn/asterisk/branches/1.8/
```

Entonces, se ingresa al nuevo directorio creado sobre el directorio asterisk previamente creado y se procede a instalar el software utilizando los siguientes comandos:

```
$cd ~/src/asterisk-complete/asterisk/1.8
./configure
make
sudo make install
sudo make config
```

Cabe recalcar que la instalación puede tomar varios minutos y se debe ejecutar cada comando después que termine de ejecutarse el comando anterior. A continuación, se instalan los addons o módulos adicionales de Asterisk ejecutando los siguientes comandos:

```
$cd ~/src/asterisk-complete/asterisk/1.8
sudo make menuselect
```

Luego de ejecutar el último comando mencionado aparece una interfaz gráfica donde se muestran los módulos adicionales de Asterisk y se debe seleccionar los módulos deseados, en este caso se debe seleccionar Chan Mobile. Luego, para ejecutar la instalación de los módulos adicionales se utiliza el comando:

```
sudo make install
```

Finalmente, se tiene el software instalado y se debe cambiar los permisos de los archivos de los directorios de Asterisk que fueron instalados. Para esto se ejecutan los siguientes comandos:

```
sudo chown –R asteriskpbx:asteriskpbx /usr/lib/asterisk/
sudo chown –R asteriskpbx:asteriskpbx /var/lib/asterisk/
sudo chown –R asteriskpbx:asteriskpbx /var/spool/asterisk/
sudo chown –R asteriskpbx:asteriskpbx /var/log/asterisk/
sudo chown –R asteriskpbx:asteriskpbx /var/run/asterisk/
sudo chown asteriskpbx:asteriskpbx /usr/sbin/asterisk/
```

Con esto se ha concluido la instalación de la PBX Asterisk y se puede proceder a modificar los archivos de configuración necesarios. Para acceder a la consola de asterisk se ejecuta el comando **`asterisk –r`** sobre una terminal de Ubuntu.

3.4 Configuración de archivos de Asterisk

En el capítulo 2 se describe toda la arquitectura de archivos y directorios que posee Asterisk. Existe una gran cantidad de archivos de configuración de Asterisk que se utilizan según sea el caso. Esto depende de los canales y módulos que se van a utilizar, sin embargo el archivo donde se describe el plan de marcado siempre será configurado, este es el archivo extensions.conf. Para este proyecto se han configurado los archivos oss.conf, chan_mobile.conf y por su puesto el archivo extensions.conf. A continuación se describe la configuración de cada uno de estos archivos en detalle.

3.4.1 Configuración de extensions.conf

Tal como se mencionó en el subcapítulo anterior este archivo describe el plan de marcado de la PBX. Este archivo se divide en contextos, los cuales se detallan como encabezados entre corchetes. Por ejemplo: [contexto-1]. Cada uno de estos encabezados junto con sus contenidos son detallados y explicados a continuación. La configuración del archivo empieza con el encabezado general, donde se describen algunas características generales del plan de marcado. En el encabezado general solo agregamos una línea de la siguiente manera.

```
[general]
static=yes
```

La línea **static=yes** indica que la operación "**dialplan save**" no es automática, es decir que cuando se realicen cambios se debe guardarlos manualmente y recargar el plan de marcado. El valor por defecto es **static=no**, es decir que guarda los cambios realizados al archivo automáticamente.

Luego se agrega el encabezado globals en el cual se declaran las variables globales y constantes del plan de marcado de la siguiente manera:

```
[globals]
CELLSMS=nokia6131
```

Como podemos observar solo se declara una variable global. No se necesitan más variables globales en este plan de marcado. La variable CELLSMS indica el nombre del dispositivo que utilizaremos como gateway a la red GSM, en este caso un teléfono celular Nokia 6131. Es necesario que el nombre que contiene esta variable sea el mismo nombre con el cual se declaró el dispositivo dentro del archivo chan_mobile.conf.

Los siguientes encabezados son los contextos del plan de marcado. Existen varios contextos configurados y no importa el orden en que

se los escriba en el archivo. Lo importante, es el orden que se lleva localmente en cada contexto. Se empieza describiendo el contexto script, en el cual se declara la extensión de consola 999 que utiliza el canal oss. Esta extensión se utiliza para realizar una ejecución automática desde el crontab de Ubuntu utilizando el comando: **asterisk –rx 'console dial 999'.** A continuación se detalla la configuración de dicho contexto:

```
[script]
exten => 999,1,Answer()
same => 2,AGI(script2.php)
same => 3,GotoIf($[${encolados} = 0]?4:6)
same => 4,Verbose(No existen mensajes encolados)
same => 5,Hangup()
same => 6,MobileStatus(${CELLSMS},CELLSTATUS)
same => 7,GotoIf($[${CELLSTATUS}=2]?10:8)
same => 8,Verbose(Existen ${enconlados} mensajes
encolados. Telefono Gateway NO DISPONIBLE! =( )
same => 9,Hangup()
same =>10,MobileSendSMS(${CELLSMS},${numero},${mensaje})
same => 11,AGI(script3.php,${id_compra})
same => 12,Verbose(Se envia mensaje al ${numero})
same => n,Goto(script,999,2)
```

Como se puede observar para declarar una extensión se excribe**exten => NUM_EXT,PRIORIDAD,ACCION** y luego se puede escribir **same => PRIORIDAD,ACCION** para continuar

configurando dicha extensión; también se puede escribir todas las líneas empezando con **exten => NUM_EXTEN** y variar solo la prioridad y acción que se realiza.

En este contexto, solo se configuró la extensión 999 la cual funciona como un conjunto de lazos basados en condiciones. Para empezar se contesta la llamada con la funciónAnswer() y luego se ejecuta el script2 con AGI. Este script busca en la tabla compra de la base datos todas las compras de los últimos 60 minutoscon status pendiente y cuenta la cantidad de compras que se obtiene luego de esta búsqueda. Es posible cambiar el número de minutos que por defecto es 60. Por ejemplo: buscar las compras pendientes de los últimos 20 minutos. La cantidad de registros que retorna la consulta se guarda en la variable encolados, la cual se utiliza en la siguiente prioridad con la sentencia GotoIf. En la prioridad 3 se ejecuta una acción condicional, en la cual se pregunta si el contenido de la variable encolados es igual a cero. Si la respuesta es verdadera la siguiente prioridad será el número 4, en la cual se mostrará un log o registro indicando que no existen mensajes encolados y luego se cerrará llamada, en caso contrario, la siguiente prioridad será la número 6.

En la prioridad número 6 se ejecuta la función MobileStatus sobre el dispositivo CELLSMS y el valor que retorna dicha función se

almacena en la variable CELLSTATUS. Este valor puede ser 0, 1 o 2 dependiendo el status del dispositivo. Luego en la prioridad 7 se ejecuta la condición CELLSTATUS igual a 2, lo cual equivale a consultar si el dispositivo CELLSMS está conectado y disponible. Entonces, si la respuesta es verdadera, la siguiente prioridad a ejecutarse será la número 10, en caso contrario será la número 8, en la cual se mostrará el log indicando que existe n mensajes encolados y el teléfono Gateway no está disponible donde n es el contenido de la variable encolados, y a continuación se cuelga la llamada.

Por otro lado, en la prioridad 10 se envía un mensaje de texto a través del dispositivo CELLSMS y basado en los contenidos de las variables número y mensaje, las cuales son establecidas después de la ejecución del script2 en la prioridad 2. Los valores que toma el script2 para establecerlas variables mensaje y número corresponden siempre a la primera compra de la lista resultante al realizar la búsqueda en el script2, es decir toma el contenido del campo número y mensaje de esa compra. Luego de esto se ejecuta el script 3 con AGI y se genera el log indicando que se envió un sms al número contenido en la variable número. El script 3 hace una actualización a la compra pendiente que fué respondida cambiando su status a ok. En la prioridad final de la extensión se retorna a la prioridad 2 del mismo contexto y la misma extensión 999 y todo se

repite hasta que no existan mensajes encolados o el teléfono Gateway no esté disponible.

Esta extensión es llamada cada 5 minutos automáticamente, es por esto que la búsqueda de compras pendientes solo se hace para los últimos 60 minutos por defecto. Es importante tener en cuenta que todas las variables mencionadas durante la ejecución de la extensión 999 son variables locales, es decir que solo tiene validez mientras la llamada no haya sido colgada.

El siguiente contexto se denominada default y es el contexto donde se configura las extensiones que el usuario tiene disponible al llamar para realizar la compra de un sms. En el menú de voz, estas extensiones corresponden a las opciones mencionadas por la contestadora.

Al llamar al teléfono Gateway desde otro teléfono celular, la llamada se transmite al servidor Asterisk y busca ejecutar la extensión “s” dentro del contexto “default”. En este contexto se configura la extensión mencionada y adicional a esta las extensiones 1, 2, 9, incorrecta y tiempo fuera que funcionan como opciones del menú de voz.

```
[default]
```

```
exten =>s,1,Verbose(Llamada entrante de
${CALLERID(num)})
same => 2,Answer()
same => 3,GotoIf($[${CALLERID(num):1:1} = 8 |
${CALLERID(num):1:1} = 9]?7:4)
same => 4,GotoIf($[${CALLERID(num):1:2} = 69 |
${CALLERID(num):1:2} = 59 | ${CALLERID(num):1:2} =
39]?7:5)
same => 5,Playback(invalida)
same => 6,Hangup()
same => 7,Background(intro)
same => n,WaitExten(8)

exten => 1,1,Goto(horoscopo,start,1)

exten => 2,1,set(clave=noticias)
same => 2,Playback(oknoticias)
same => 3,AGI(script1.php,${clave})
same => n,Hangup()

exten => 9,1,Hangup()

exten => i,1,Playback(invalida)
exten => i,n,Goto(default,s,3)

exten => t,1,Hangup()
```

Como se puede observar en la primer prioridad mostramos el log “llamada entrante de ${CALLERID(num)}” donde CALLERID(num) es una variable del sistema que identifica y guarda el número de la llamada entrante.

Luego, en la prioridad 2 se contesta la llamada y en la prioridad 3 se utiliza una condición para saber si el segundo dígito del número de llamada entrante es igual a 8 o 9. Si es verdadera, la siguiente prioridad es la 7, caso contrario es la 4. En la prioridad 4 se utiliza otro condicional donde se verifica si los dígitos 2 y 3 del número entrante son iguales a 69 o a 59 o a 39. Si es verdadera, entonces la siguiente prioridad es la 7, caso contrario, es la 5. En la prioridad 5 se reproduce el archivo de audio invalida.gsm que indica opción no valida. En resumen, esto se reproduce cuando se llama desde un número convencional. En la prioridad 6, se cuelga la llamada. En la prioridad 7 se reproduce el archivo de audio intro.gsm, en el cual se le da la bienvenida al cliente y se le indican las opciones generales que posee. Este archivo de audio debe estar guardado en el directorio /var/lib/asterisk/sound/en, caso contrario se debe especificar la ruta. En la última prioridadde la extensión “s”, se espera 8 segundos después de terminado el intro para que el usuario escoja una de las opciones mencionadas que son las extensiones que se declaran posteriormente. Estas opciones son:

- Opción 1 para acceder a categoría horóscopo
- Opción 2 para comprar mensaje de noticias
- Opción 9 para salir (cuelga la llamada)

- Opción incorrecta. Esta reproduce un archivo de audio indicando opción no valida y retorna a la prioridad 3 de la misma extensión.

La extensión 1 corresponde a la opción 1. Al ejecutarse esta extensión simplemente se redirige la llamada al contexto horóscopo en la extensión start y prioridad 1.

Luego, tenemos la extensión 2, en la cual como prioridad 1, se establece la variable local “clave” con el contenido “noticias”. En la prioridad 2, se reproduce el archivo de audio oknoticias.gsm que le indica al usuario que ha seleccionado noticas y que le llegará su mensaje de texto, luego ejecuta el script1 con AGI y finalmente cuelga la llamada.

La extensión 9, simplemente cuelga la llamada. Por otro lado, la extensión “i” se interpreta como opción incorrecta, es decir cual opción diferente a 1,2 o 9. Esta extensión como prioridad 1 reproduce el archivo invalida.gsm que le indica al usuario que presionó una opción no válida y luego retorna la llamada a la extensión “s” del contexto default en la prioridad 3.

Finalmente tenemos la extensión “t” que se interpreta como tiempo fuera, es decir que se ejecuta si el usuario no presionó opción alguna durante 8 segundos a partir del fin de reproducción del archivo

intro.gsm. En esta extensión al igual que la extensión 9, solo se cuelga la llamada.

El siguiente contexto se denomina “horoscopo”. En este contexto se muestran todos los signos zodiacales como opciones, es decir cada opción del 1 al 12 es un signo tomando comoreferencia la tabla IV que se muestra en el anexo B.

A continuación se muestra la configuración del contexto horóscopo:

```
[horoscopo]
exten => start,1,Background(signos)
same => n,WaitExten(8)

exten => 1,1,set(clave=acuario)
same => 2,Playback(acuario)
same => 3,Goto(final,start,1)


exten => 2,1,set(clave=piscis)
same => 2,Playback(piscis)
same => 3,Goto(final,start,1)

exten => 3,1,set(clave=aries)
same => 2,Playback(aries)
same => 3,Goto(final,start,1)

exten => 4,1,set(clave=tauro)
same => 2,Playback(tauro)
```

```
same => 3,Goto(final,start,1)

exten => 5,1,set(clave=geminis)
same => 2,Playback(geminis)
same => 3,Goto(final,start,1)

exten => 6,1,set(clave=cancer)
same => 2,Playback(cancer)
same => 3,Goto(final,start,1)

exten => 7,1,set(clave=leo)
same => 2,Playback(leo)
same => 3,Goto(final,start,1)

exten => 8,1,set(clave=virgo)
same => 2,Playback(virgo)
same => 3,Goto(final,start,1)

exten => 9,1,set(clave=libra)
same => 2,Playback(libra)
same => 3,Goto(final,start,1)

exten => 10,1,set(clave=escorpion)
same => 2,Playback(escorpion)
same => 3,Goto(final,start,1)

exten => 11,1,set(clave=sagitario)
same => 2,Playback(sagitario)
same => 3,Goto(final,start,1)
```

```
exten => 12,1,set(clave=capricornio)
same => 2,Playback(capricornio)
same => 3,Goto(final,start,1)

exten => 0,1,Goto(default,s,3)

exten => i,1,Playback(invalida)
same => n,Hangup()

exten => t,1,Hangup()
```

Para comenzar se tiene la extensión start en la cual como prioridad 1 se reproduce el archivo de audio signos.gsm que le indica al usuario cuales son las opciones que tiene para escoger desde 0 hasta el 12, donde las opciones del 1 al 12 corresponden a los signos zodiacales siguiendo el orden de la tabla IV y la opción 0 es para volver a escuchar el menú principal. Al igual que en el contexto default declaramos las extensiones “i” y “t”, en las cuales colgamos la llamada con la diferencia que en la extensión “i” antes de colgar la llamada se reproduce el archivo invalida.gsm.

Como ya se mencionó, la extensión 0 sirve para volver a escuchar el menú principal, por lo que al presionar esta extensión solo se re direcciona la llamada al contexto default en la prioridad 3.

Las opciones del 1 al 12 siguen un patrón que consta de 3 prioridades. Lo que se hace como prioridad 1 es establecer el

contenido de la variable “clave” con el nombre del signo correspondiente a cada extensión. Luego, en la prioridad 2 se reproduce un archivo de audio distinto por cada signo, en el cual se le indica al usuario el signo que escogió y se le dice que para continuar presione 0 o en caso contrario espere en la línea. Como última prioridad se re direcciona la llamada al contexto “final” en la extensión start y prioridad 1.

El contexto “final” comienza con la extensión start, la cual simplemente espera 5 segundos para que el usuario confirme la compra presionando 0. Si la opción que presiona el usuario es incorrecta entonces se re direcciona a la llamada la extensión start del mismo contexto en la prioridad 1. Si el usuario espera los 5 segundos entonces re direcciona la llamada al contexto horóscopo en la extensión start y prioridad 1 con lo cual se vuelve a reproducir el menú de horóscopo y el usuario puede volver a escoger su signo zodiacal. A continuación se muestra la configuración del contexto “final”.

```
[final]
exten => start,1,WaitExten(5)

exten => 0,1,AGI(script1.php,${clave})
same => 2,Playback(ok)
same => n,Hangup()
```

```
exten => i,1,Playback(invalida)
exten => i,n,Goto(final,start,1)

exten => t,1,Goto(horoscopo,start,1)
```

En la extensión 0 de este contexto como prioridad 1 se ejecuta el script1 con AGI. Luego, se reproduce el archivo de audio ok.gsm que le indica al usuario que le llegará un mensaje de texto y finalmente se cuelga la llamada. Con este contexto, se finaliza la configuración del archivo extensions.conf

3.4.2 Configuración del archivo chan_mobile.conf

En este archivo se describen todos los dispositivos que se conectaran con el servidor Asterisk utilizando el canal chan mobile. Para este sistema, estos son el adaptador bluetooth USB marca Dlink y el teléfono celular marca Nokia modelo 6131. Para definir cada dispositivo se les asigna un nombre que se escribe entre corchetes, de la misma manera en que se declara un contexto en el archivo extensions.conf. En el caso de ser un adaptador se escribe el nombre adapter pero se le asigna un id. Al igual que en el archivo extensions.conf también se debe declarar el contexto general donde se definen opciones general que se aplican para todos los dispositivos. En el contexto general solo se ha defino el intervalo de

conexión con los dispositivos que es de 30 segundos. Esto se define de la siguiente manera:

```
[general]
interval=30
```

Luego de esto, se define el adaptador bluetooth USB de la siguiente manera:

```
[adapter]
id=dlink
address=00:1C:F0:EE:C3:9B
forcemaster=yes
```

Como se puede observar, se ha declarado el adaptador colocando el encabezado adapter, luego le asignamos un id que en este caso es “dlink”. Adicional a esto se escribe la dirección MAC del equipo con el comando **`address=DIRECCION_MAC`**. Para obtener esta dirección MAC se ejecuta el comando hcitool scan desde una terminal de Ubuntu (ver Figura 9).

Los comandos mencionados hasta ahora (id y address) son necesarios, es decir con esto el adaptador bluetooth puede trabajar, sin embargo existen opciones extras que son configuradas de acuerdo a los requerimientos del administrador del sistema. En este sistema, se añadió el comando **`forcemaster=yes`**. Este comando intenta forzar el adaptador en modo master o maestro, el valor por defecto es

no. Esto se hace porque nuestro adaptador debe ser el master en la conexión bluetooth con el teléfono celular.

Para finalizar la configuración del archivo chan mobile, se define el teléfono celular que se conectará vía bluetooth de la siguiente manera:

```
[nokia6131]
address=00:18:0F:B2:03:D0     ; NOKIA 6131
port=13               ; the rfcomm port number
context=default; dialplan context for incoming calls
adapter=dlink  ; adapter to use
group=1         ; this phone is in channel group 1
sms=yes
```

Como se puede observar para definir un dispositivo se escribe el nombre que le vamos a asignar entre corchetes como un encabezado. En nuestro caso, lo denominamos nokia6131. Luego se debe declarar la dirección MAC del dispositivo y el puerto rfcomm del servidor que utiliza para la conexión. Para saber en qué puerto se conecta el dispositivo se utiliza el comando mobile search desde la consola de asterisk (ver Figura 3.7). Es necesario que el adaptador esté declarado previamente en el archivo chan mobile.

```
servidor-asterisk*CLI> mobile search
Address           Name                           Usable Type    Port
00:18:0F:B2:03:D0 Vanessa                        Yes    Phone   13
```

Figura 3.7 Ejecución del comando mobile search desde la consola de asterisk

Como se puede observar el dispositivo con nombre Vanessa (nombre que se configura en el teléfono) y dirección MAC 00:18:0F:B2:03:DD está conectado al puerto 13. Luego de conocer el puerto lo declaramos en el contexto del dispositivo con el comando **`port=13.`** Luego se declara el contexto del plan de marcado que utilizará el dispositivo para llamadas entrantes con el comando **`context=CONTEXTO`**. En la siguiente línea se declara el adaptador que utilizará el dispositivo para conectarse, esta línea es sumamente importante, en este caso escogemos el adaptador con id “dlink” usando el comando **`adapter=dlink.`** Finalmente se añaden los comando group=1 y sms=yes. El primero indica el grupo de canales que utilizará el dispositivo, y el segundo indica que dispositivo si soporta SMS. El último comando es necesario para nuestro sistema porque la configuración por defecto de SMS es no. Con esto se concluye la configuración del archivo chan mobile.

Como se mencionó en el capítulo 2 chan mobile soporta múltiples adaptadores y múltiples dispositivos celulares. Para añadir dispositivos seguimos el mismo patrón del contexto nokia6131 y para añadir adaptadores seguimos el patrón de configuración con encabezado adapter.

3.4.3 Configuración del archivo oss.conf

El archivo oss.conf es un archivo donde se configuran extensiones de consola. Solo las extensiones declaras en este archivo podrán ser marcadas desde la consola de asterisk. Para la configuración tenemos el contexto general donde declaramos la única extensión de consola que vamos a utilizar e indicamos el contexto al que pertenece esta extensión, este es el contexto script, el cual se define en el archivo extensions.conf. Además de esto, se añade el comando **autoanswer=no**, esto indica que la llamada realizada desde consola no será contestada automáticamente. Con estos comandos es suficiente para poder utilizar nuestra extensión de consola. Debemos recordar que las acciones que se realizan al marcar esta extensión se configuran en el plan de marcado dentro su contexto respectivo, en este caso dentro del contexto script. A continuación se muestra el contenido del archivo oss.conf.

```
[general]
autoanswer = no      ; no autoanswer on call
extension = 999
context = script
```

Para marcar una extensión desde consola debemos utilizar el comando **console dial EXTENSION**, para este sistema sería **console dial 999** y listo. Para este sistema no es necesario

utilizar parlantes y micrófono para utilizar la consola de asterisk como un teléfono, ya que la ejecución de esta extensión es para ejecutar scripts AGI.

3.5 Implementación de scripts

Debido a la interacción de la PBX Asterisk con una base de datos fue necesario implementar 3 scripts que son ejecutados en diferentes casos y que cumplen distintas funciones. Estos scripts son ejecutados desde el plan de marcado de la PBX utilizando la interface AGI.

Los 3 scripts denominados script1, script2 y script3 fueron implementados en el lenguaje de programación orientado a objetos PHP. La interacción entre Asterisk y PHP funciona muy bien, debido a esto se decidió desarrollar los scripts en PHP. Para la implementación de estos scripts fue necesario obtener la librería phpagi la cual se puede descargar desde la web [5]. Esta librería es una clase de PHP desarrollada para la interface AGI de Asterisk.

A continuación se detalla el código fuente de cada script con su respectiva descripción.

3.5.1 Implementación de Script1

Este script fue desarrollado con el objetivo de ingresar compras nuevas a la base de datos MySQL que maneja el sistema. La ejecución de este script se realiza cuando un usuario realiza la compra de un mensaje de texto de cualquiera de las categorías llamando al teléfono Gateway.

Como se indicó en la introducción del capítulo, se requiere de la librería phpagi, la cual debe estar guarda en el directorio /var/lib/asterisk/agi-bin/.

Para comenzar el script, se crea una instancia de la clase AGI y a través de este objeto podemos obtener el número de la llamada entrante, esto se hace con el método request y la variable agi_callerid de la siguiente manera **`$agi->request[agi_callerid].`** En el código se valida que el número solo tenga caracteres numéricos y se lo guarda en una variable denominada “num”. Por otro lado la palabra clave escogida por el usuario la se obtiene del parámetro pasado por AGI cuando se ejecuta el script. Este parámetro se almacena en el arreglo argv[]. En la posición 0 del arreglo se tiene el nombre del script y a partir de la posición 1 los parámetros. Se guarda la clave dentro de la variable “key” con el comando **`$key=$argv[1].`** Luego se realiza la conexión con la base datos cuyo nombre es “test”. Para esta

conexión disponemos del usuario asterisk con contraseña asterisk, y no necesitamos de la dirección IP del servidor SQL porque es local. Todos estos datos son almacenados en las variables db, dbuser y dbpass respectivamente. Entonces se realiza la conexión a MySQL y se escoge la base de datos test.

Después de escoger la base de datos, se procede a realizar el ingreso del registro del cliente en la tabla respectiva. Para esto solo se necesita el número que está almacenado en una variable. En PHP se utiliza el método mysql_query(STRING) para ejecutar consultas, ingresos de datos o actualizaciones. El ingreso a la tabla cliente se hace con el comando:

```
mysql_query("INSERT INTO cliente VALUES('$num')");
```

Luego, se realiza el ingreso de una compra en su respectiva tabla. Para esto se necesita, el número y la clave, ya que para la fecha se utiliza el valor por defecto que es la fecha actual y para el status se coloca pendiente. En el caso del id es un valor que se incrementa automáticamente tal como se la definió en la base de datos. El comando ejecutado para esta acción es el siguiente:

```
mysql_query("INSERT INTO compra
(numero,clave,fecha,status)
VALUES('$num','$key',DEFAULT,'pendiente')");
```

Luego de esto finaliza el script y ya tenemos una compra ingresada con status pendiente. La cual será atendida en la próxima ejecución automática de la extensión de consola 999.

3.5.2 Implementación de Script2

Este script se desarrolló para buscar las últimas compras pendientes y obtener sus datos de los campos id, número y clave que luego son utilizados por el script3 al cual se los pasa como parámetros desde el plan de marcado. Este script se ejecuta cada 5 minutos automáticamente y cada vez que existen compras con status pendiente de los últimos 60 minutos, se intenta responderlas una por una con ayuda del script 3.

Al igual que el script1 se empieza añadiendo la librería phpagi y creamos un objeto AGI. Luego, se realiza la conexión con la base de datos “test” y se procede a realizar una consulta. El resultado de esta consulta es un arreglo con todas las compras de status pendiente de los últimos 60 minutos. Es importante realizar una búsqueda limitada porque esta base de datos crece constantemente y si no se limita a buscar las compras de los últimos 60 minutos aumentaremos latencia y el tiempo de ejecución de la consulta aumentará. Cabe recalcar que este tiempo en minutos es un valor parametrizable a través de la

aplicación JAVA implementada. La consulta SQL se realiza con el siguiente código:

```
$row=mysql_query("SELECT id_compra,numero,clave FROM
(SELECT id_compra,numero,clave,status FROM compra
WHERE fecha > DATE_SUB(now(), INTERVAL '$tdb'
MINUTE)) AS ultimos WHERE status = 'pendiente'");
```

Luego de esto se cuenta el número de registros almacenados en el arreglo row y se lo guarda en la variable “i”, para luego establecer la variable “encolados” usada en la descripción de la extensión de consola 999 con el valor que contiene “i”. Esto se hace con el método set_variable de la siguiente manera:

```
$agi->set_variable("encolados",$i);
```

El resto del script se ejecuta de acuerdo a una sentencia condicional if. En la cual se maneja la condición i>0. Si la respuesta es falsa entonces se finaliza la ejecución del script. En caso contrario, se toma el primer registro del arreglo row y se almacena en un arreglo llamado compra para luego obtener los valores de los campos id_compra, número y clave. Con estos valores se establece las variables utilizadas en el plan de marcado para poder enviar el mensaje texto. La variable idcno se utiliza para enviar el mensaje, pero si como parámetro para ejecutar el script3. Adicional a las variables número y

clave necesitamos obtener el contenido de la variable mensaje que es el cuerpo del mensaje que se va a enviar. Para esto, se realiza una consulta buscando el registro del mensaje con palabra clave igual al contenido de la variable “key” y se guarda este registro en el arreglo “rowsms”. Laconsulta SQL es la siguiente:

```
$rowsms=mysql_query("SELECT smstxt FROM mensaje
WHERE clave like '$key' LIMIT 1");
```

Del arreglo rowsms se guarda el contenido del campo smstxt en la variable “msgbody” para luego establecer la variable “mensaje”.

Con esto, se consigue establecer las variables número, clave y mensaje que son utilizadas en el plan de marcado para enviar el mensaje de texto. Vale la pena recordar que la variable idc se pasa como parámetro en el script3 y es por esta razón que debemos establecer su valor en el script2.

3.5.3 Implementación de Script3

Este script fue desarrollado para actualizar las compras que son atendidas. Es decir que cuando existen compras pendientes y se envía el sms correspondiente a una de estas compras, en seguida se

ejecuta el script3 pasando como parámetro el id de la compra respondida para luego actualizar su status a ok.

Este script, al igual que los scripts 1 y 2 comienza añadiendo la librería phpagi y creando un objeto AGI. Seguidamente se conecta a la base de datos test, luego ejecuta una actualización SQL y finaliza su ejecución. Se realizauna actualización sobre la compra con el id_compra igual al contenido de la variable “idc”.En la actualización se cambia el valor del campo status a “ok”. El valor de la variable “idc” se pasa como parámetro a través del AGI, y en el código PHP se lo obtiene del arreglo argv de la siguiente manera:**`$idc=$argv[1].`**

Luego de ejecutar la actualización SQL finaliza el script.

3.6 Crontab. Implementación de tarea automática.

Tal como se ha mencionado en este documento, fue necesario implementar una tarea automática que ejecute la marcación de una extensión de la consola de asterisk con el fin ejecutar un conjunto de procedimientos para responder mensajes de compras pendientes. Para la configuración de esta tarea automática se ha utilizado el servicio crontab de Ubuntu.

En los sistemas basados en Unix, CRON es un administrador de procesos tipo daemon, los cuales son ejecutados en segundo plano. Es decir son procesos que se están ejecutando sin ser vistos por el

administrador del sistema. Estos procesos se ejecutan automáticamente de acuerdo a la administración del CRON. En Ubuntu se cuenta con el Crontab, que podríamos decir que es un equivalente a “Tareas programadas” en Windows.

Para configurar el archivo de Crontab se ejecuta desde la consola de Ubuntu el comando crontab –e y se escoge un editor para realizar los cambios respectivos en el archivo. Cada tarea se declara en una línea donde se define la fecha y hora de ejecución seguida del comando en Shell. Es posible configurar una ejecución constante de cada hora, cada 5 minutos, cada semana, cada mes, todos los 15 de cada mes, entre otras. En este caso, se ha configurado una tarea que se ejecute cada 5 minutos. El formato de configuración para una tarea es el siguiente:

M H DM M DS COMANDO

Donde, M son los minutos, es un valor entre 0 y 59. H es la hora y es un valor entre 0 y 23. DM es el día del mes y es un valor entre 1 y 31. M es el número del mes y es un valor entre 1 y 12, finalmente DS es el día de la semana y es un número entre 1 y 7 tomando como referencia lunes=1 y domingo=7. Si no quiere un valor específico para alguno de estos parámetros se coloca el valor * que significa cualquiera, es decir cualquiera de los valores permitidos según sea el caso. Por ejemplo si se quiere que a las 11 pm de todos los días 15 de

cada mes se ejecute el script quincena.sh que está guardado en la ruta /home/user la configuración de la tarea sería la siguiente:

0 23 15 * * ./home/user/quincena.sh

Después de configurar el archivo crontab, se guardan los cambios y listo. Para revisar las tareas configuradas ejecutamos el comando **`crontab -l`** desde la consola de Ubuntu.

La tarea para este sistema se configuró de la siguiente manera:

***/5 * * * * asterisk –rx 'console dial 999'**

El comando**asterisk –rx 'console dial 999'** se ejecutará cada 5 minutos.

CAPITULO 4

4. Pruebas de funcionamiento

4.1 Conexión del Teléfono Gateway y envío de mensajes sms.

Es importante realizar pruebas de conexión entre asterisk y el teléfono celular que actúa como Gateway. Como prueba previa se debe comprobar la operatividad de la conexión bluetooth entre el servidor y el teléfono. Para esto se puede transferir archivos en cualquier sentido, en es decir desde el teléfono hacia el servidor o viceversa. Si la conexión bluetooth entre ambos equipos funciona ahora se debe comprobar la conexión con Asterisk. Se debe recordar que para que exista conexión entre asterisk y el teléfono se debe configurar correctamente el archivo chan mobile tal como se detalla en el subcapítulo 3.5.2. Luego se puede comprobar la conectividad del dispositvo utilizando el comando **`mobile show devices`**desde la consola de asterisk tal como se muestra en la Figura 4.1.

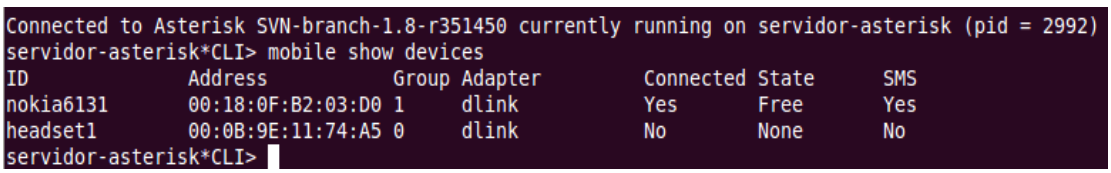


Figura 4.1. Ejecución del comando mobile show devices desde la consola de Asterisk

Como se puede observar este comando muestra un listado de todos los dispositivos declarados en el archivo chan_mobile.conf. En este caso se ha configurado el dispositivo “nokia6131” y nos indica su dirección MAC, el grupo de canales al que pertenece, el adaptador que usa para conectarse y su estado. En el ejemplo de la figura 12 el teléfono está conectado, está libre o disponible y si soporta mensajes de texto. Existen casos de dispositivos que no soportan mensajes de texto, en estos casos si ejecutados el mismo comando, debe mostrarse “no” en la columna SMS. Otra forma de darnos cuenta que el dispositivo no soporta SMS es intentando enviar un SMS con lo cual se mostrará automáticamente un log en la consola de asterisk (ver Figura 4.2).

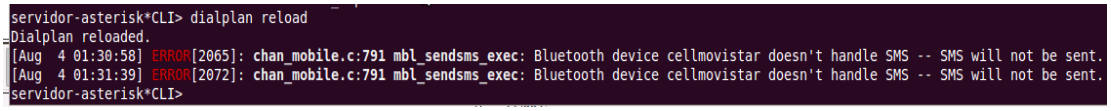


Figura 4.2 Log indicando que el dispositivo “cellmovistar” no soporta SMS

A continuación se muestra la tabla V, con el estado de conexión de un dispositivo en los principales casos posibles. Solo se muestran las columnas Connected, State, SMS.

Tabla V. Ejemplos de estados de dispositivos al ejecutar el comando mobile show devices desde la consola de asterisk.

Connected	State	SMS	Descripción
Yes	No service	Yes	El dispositivo está conectado con asterisk y si soporta SMS pero no capta señal de la red GSM (sin cobertura)
Yes	Free	No	El dispositivo está conectado con asterisk y está disponible pero no soporta SMS
Yes	Busy	Yes	El dispositivo está conectado con asterisk y si soporta SMS pero no está disponible.
No	-	No	El dispositivo está desconectado de asterisk. Probablemente esté conectado con el servidor pero no lo está con la PBX

Con esto se puede determinar si un dispositivo está correctamente configurado en el archivo chan_mobile.conf verificando su dirección MAC, puerto de conexión y adaptador, y a la vez podemos verificar su estado de conexión.

4.2 Ejecución y funcionamiento de scrip1.php

Tal como se detalla en el subcapítulo 3.6.2 el script1 ingresa registros en las tablas compra y cliente. Para verificar el correcto funcionamiento de script se realizaron las siguientes pruebas:

1. Se ejecutó el script desde la consola de Ubuntu con datos constantes en ciertas variables definidas por el programador y luego se verificó en la base de datos que se hayan ingresado correctamente. Para esta prueba se alteró el código colocando un valor fijo para la variable “num” ($num=099555123) y un valor fijo para la variable “key” ($key=sagitario). Luego se lo ejecutó desde una terminal de Ubuntu de la siguiente manera: **/var/lib/asterisk/agi-bin/scrip1.php**. Luego de la ejecución se revisaron las tablas cliente y compra en la base de datos ejecutando los siguientes consultas SQL: **select * from cliente**, **select * from compra.**Finalmente, se verificó que todos los campos se hayan ingresaron de manera correcta.
2. Se creó una extensión SIP con ayuda de un Softphone. Para esto se configuró un usuario SIP en el archivo sip.conf y en el plan de marcado se definió que cuando se llame a dicha extensión se ejecute el script1 usando AGI y pasando la clave como parámetro. Para este caso solo se configuró un valor fijo en la variable “num”

ya que la clave se la pasó como parámetro. Así mismo, se verificó en la base datos que el registro se haya ingresado correctamente.

Con estas pruebas podemos asegurarnos que el scrip1 funciona de manera correcta.

4.3 Ejecución y funcionamiento de scrip2.php

Para comprobar el correcto funcionamiento de este script que realiza varias acciones se realizaron pruebas similares a las realizadas con el script1. Es decir, primero una ejecución directa con valores fijos en ciertas variables para asegurarse de que no existan errores de sintaxis. Este se hizo ejecutando **/var/lib/asterisk/agi-bin/script2.php** desde una terminal de Ubuntu. Luego se realizó una prueba con ayuda del softphone. La extensión SIP utilizada para probar el script1 se reconfiguró en el plan de marcado para ejecutar el script2 con AGI e imprimir el contenido de las variables que son establecidas para ser usadas por el script3 y la función MobileSendSMS. Estas variables se imprimieron con el comando Verbose(VARIABLE). Por ejemplo: Verbose(${encolados}). En este caso lo más importante es comprobar que las variables idc, número, clave y mensaje del contexto script de la extensión de consola 999 son correctamente establecidas imprimiendo su contenido. No es necesario revisar los registros de alguna tabla de la base de datos.

4.4 Ejecución y funcionamiento de script3.php

Para este script que realiza la actualización de registros de la tabla compra, también se utilizó un procedimiento de pruebas similar al utilizado en los scripts anteriores.

En este caso se reconfiguró la extensión SIP para comprobar la correcta ejecución de la actualización SQLrealizando una llamada desde el softphone. La extensión SIP se configuró para ejecutar el script3 con AGI pasando como parámetroun valor constante de la variable idc, la configuración de la extensión SIP fue la siguiente:

```
Exten => 100,1,Answer()
Same => 2,AGI(scrip3.php,10)
Same=> n,Hangup()
```

Luego de esto se verificó el registro de la tabla compras que fue actualizado con el siguiente comando: **select * from compra where id_compra=10.**Se debe comprobar que el campo status fue actualizado a “ok”. Con esto comprobamos el correcto funcionamiento de este script.

Al final, se realizaron pruebas generales del plan de marcado realizando compras de mensajes llamando al teléfono Gateway y escogiendo cualquiera de las palabras claves para luego esperar la respuesta del

mensaje. El sistema funciona muy bien. Como recomendación podemos conectar otro teléfono celular y destinarlo solo para enviar los mensajes de texto, de tal forma que el teléfono que recibe las llamadas podría estar siempre ocupado recibiendo llamadas e igual se enviarían los mensajes.

CONCLUSIONES Y RECOMENDACIONES

Las conclusiones son:

1) Se implementó un sistema capaz de enviar mensajes de texto a teléfonos celulares conectados a la GSM usando como puerta de enlace un teléfono celular. Este sistema se implementó a un bajo costo y funciona bastante bien, sin embargo la disponibilidad del servicio que brinda este sistema no es realmente buena debido a la falta de recursos, pero se podría mejorar notablemente añadiendo uno a más teléfonos celulares compatibles con Asterisk y chan mobile al sistema.
2) En lo referente a la compatibilidad del módulo Chan Mobile con teléfonos que soporten envió y recepción de SMS utilizando Asterisk se puede concluir que no es muy buena y la mayor parte de modelos compatibles y que soportan esta característica son modelos de celulares descontinuados en su fabricación, lo que los hace más difíciles de conseguir, sin embargo son modelos económicos.
3) Se implementó una aplicación en JAVA que facilita la administración de este sistema permitiendo llevar un control de registros de compras, clientes y mensajes en una base de datos. Además, a través de este sistema se realizan actualizaciones periódicas y permite editar el

tiempo de expiración de las compras pendientes que por defecto es 60 minutos.

Las recomendaciones son:

1) Para realizar una implementación más eficiente de este sistema es recomendable utilizar 2 o más teléfonos celulares y no es necesario que todos soporten SMS. Se puede utilizar un teléfono que pueda recibir llamadas conectado a Asterisk y otro teléfono dedicado solo a enviar SMS desde Asterisk, el cual si debe soportar SMS. Con estos 2 teléfonos se tendría un sistema más eficiente, ya que la mayor parte del tiempo el teléfono que envía los SMS estaría disponible, así como también se da mayor disponibilidad al teléfono que recibe las llamadas. Otra alternativa es usar 4 teléfonos, 2 que reciban llamadas y 2 que envíen SMS pero 2 conectados a la red de Movistar y 2 conectados a red de Claro que son las redes de mayor número de usuarios en Ecuador. Es decir se tendría un teléfono movistar para recibir llamadas y otro para enviar SMS. Y lo mismo con Claro, un teléfono Claro para recibir llamadas y otro para enviar SMS. Estás son posibles opciones para una implementación más eficiente de este sistema.
2) Otra opción interesante es conseguir un modelo de teléfono que soporte recibir SMS y transmitirlos al servidor Asterisk para que el

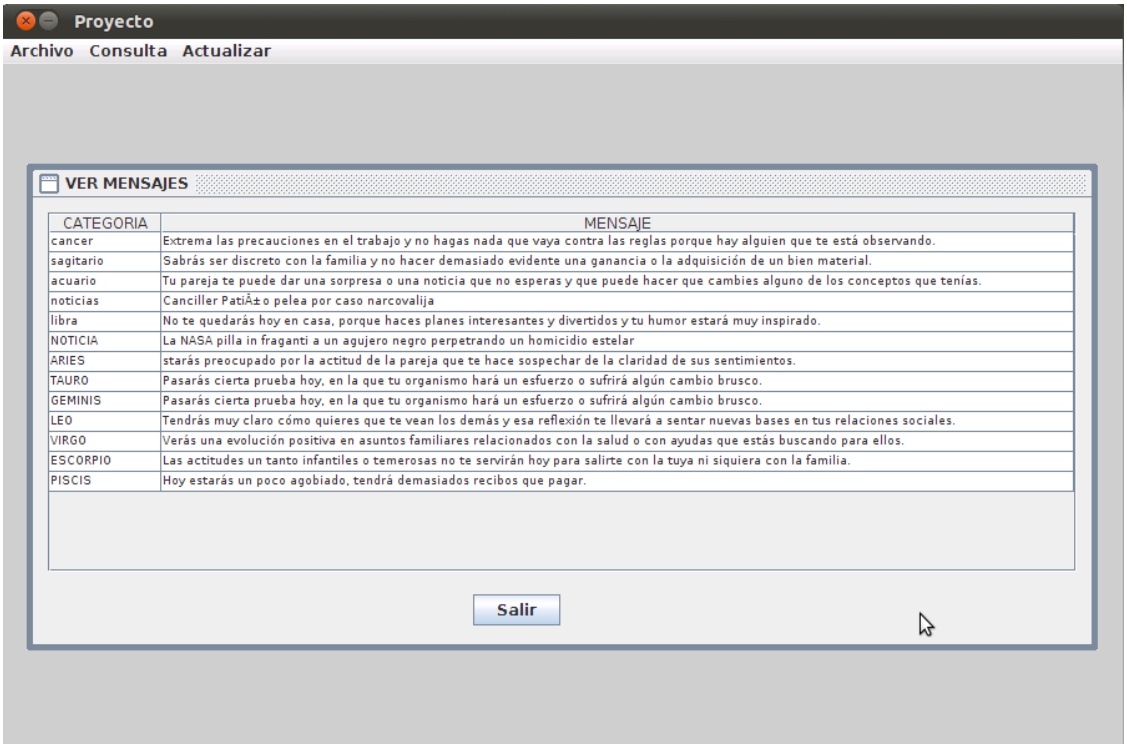
usuario pueda realizar su petición vía SMS en lugar de llamar. En realidad existen varias alternativas para implementar ese sistema utilizando Chan Mobile de Asterisk.

3) Una de las mejores ventajas de este sistema, es que el servidor no necesita de un hardware muy robusto y los teléfonos celular compatibles son económicos. Dado que el hardware utilizado no es muy costoso y el software Asterisk es libre, podemos concluir que este sistema es realmente económico y funciona bien, vale la pena recordar que con las recomendacionesmencionadas se tendrá mas eficiencia.

ANEXO A:

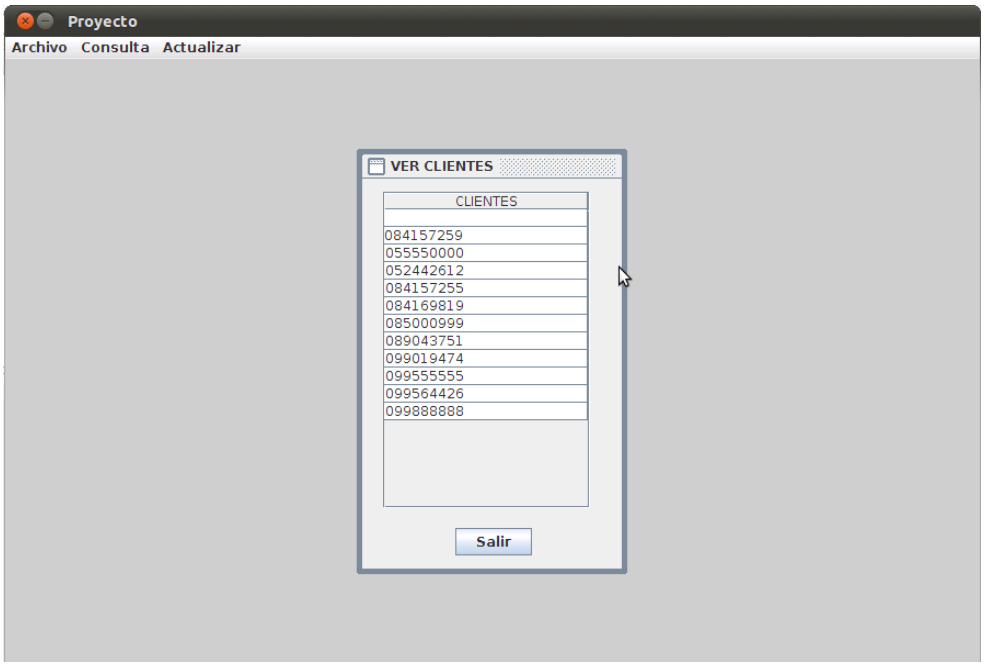
Aplicación JAVA implementada para actualizaciones y consultas en la base de datos del sistema:

- **Consulta de la tabla mensaje:** Muestra los 13 registros de mensajes que debe tener el sistema.

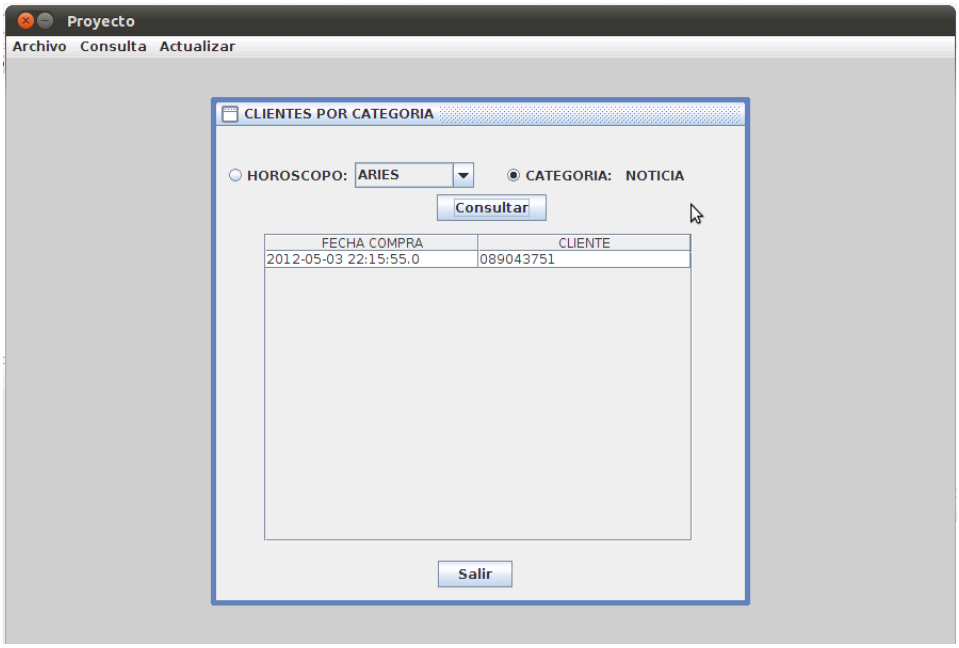


CATEGORIA	MENSAJE
cancer	Extrema las precauciones en el trabajo y no hagas nada que vaya contra las reglas porque hay alguien que te está observando.
sagitario	Sabrás ser discreto con la familia y no hacer demasiado evidente una ganancia o la adquisición de un bien material.
acuario	Tu pareja te puede dar una sorpresa o una noticia que no esperas y que puede hacer que cambies alguno de los conceptos que tenías.
noticias	Canciller PatiÃ±o pelea por caso narcovalija
libra	No te quedarás hoy en casa, porque haces planes interesantes y divertidos y tu humor estará muy inspirado.
NOTICIA	La NASA pilla in fraganti a un agujero negro perpetrando un homicidio estelar
ARIES	starás preocupado por la actitud de la pareja que te hace sospechar de la claridad de sus sentimientos.
TAURO	Pasarás cierta prueba hoy, en la que tu organismo hará un esfuerzo o sufrirá algún cambio brusco.
GEMINIS	Pasarás cierta prueba hoy, en la que tu organismo hará un esfuerzo o sufrirá algún cambio brusco.
LEO	Tendrás muy claro cómo quieres que te vean los demás y esa reflexión te llevará a sentar nuevas bases en tus relaciones sociales.
VIRGO	Verás una evolución positiva en asuntos familiares relacionados con la salud o con ayudas que estás buscando para ellos.
ESCORPIO	Las actitudes un tanto infantiles o temerosas no te servirán hoy para salirte con la tuya ni siquiera con la familia.
PISCIS	Hoy estarás un poco agobiado, tendrá demasiados recibos que pagar.

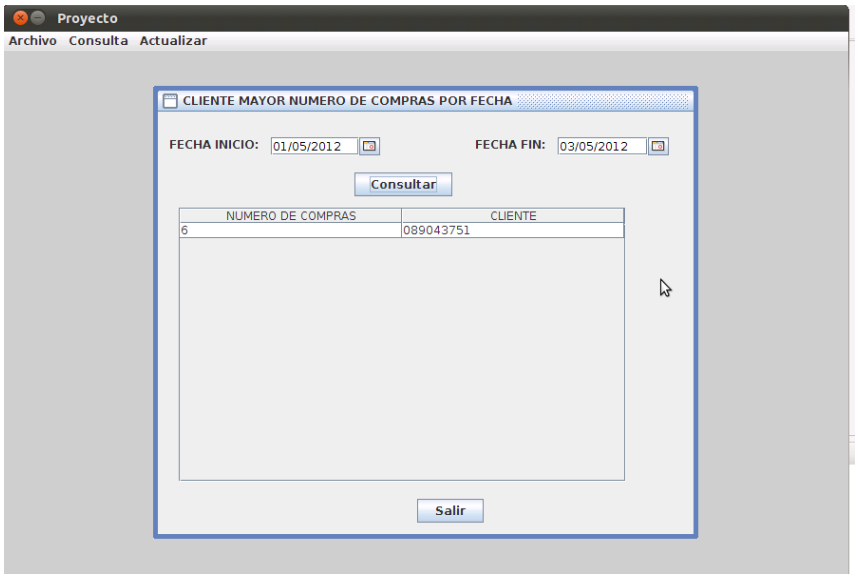
- **Consulta de la tabla clientes:** Muestra la tabla clientes



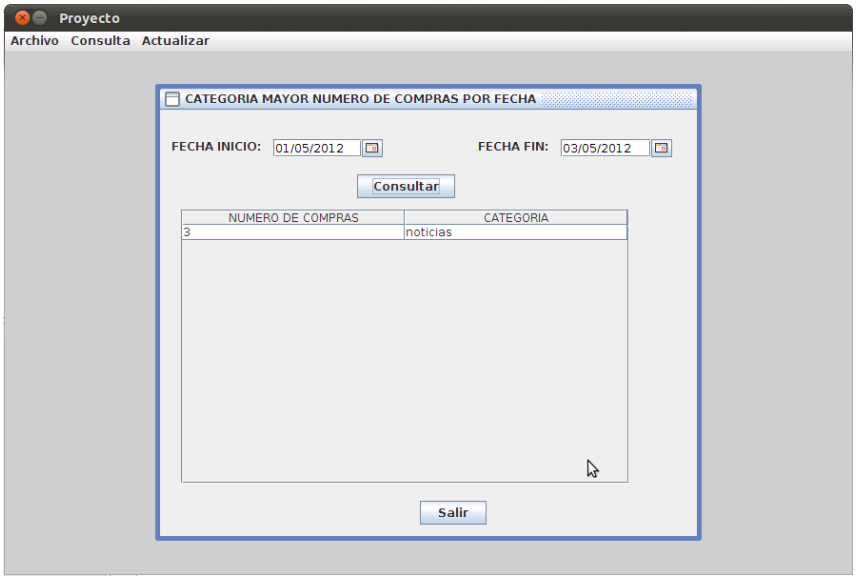
- **Consulta clientes por categoría:** Muestra el número de cliente y la fecha de compra de acuerda a la categoría seleccionada.



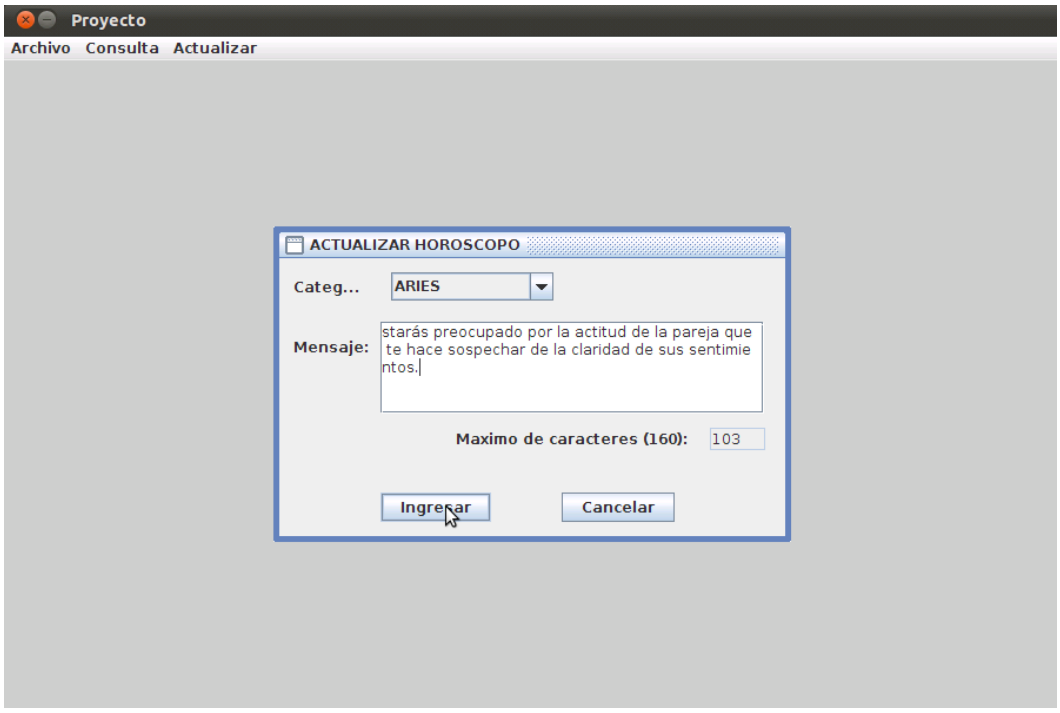
- **Consulta de cliente con mayor número de compras por fecha:** Muestra el número de cliente que mas compras ha realizado en un rango de fecha con su respectivo número de compras

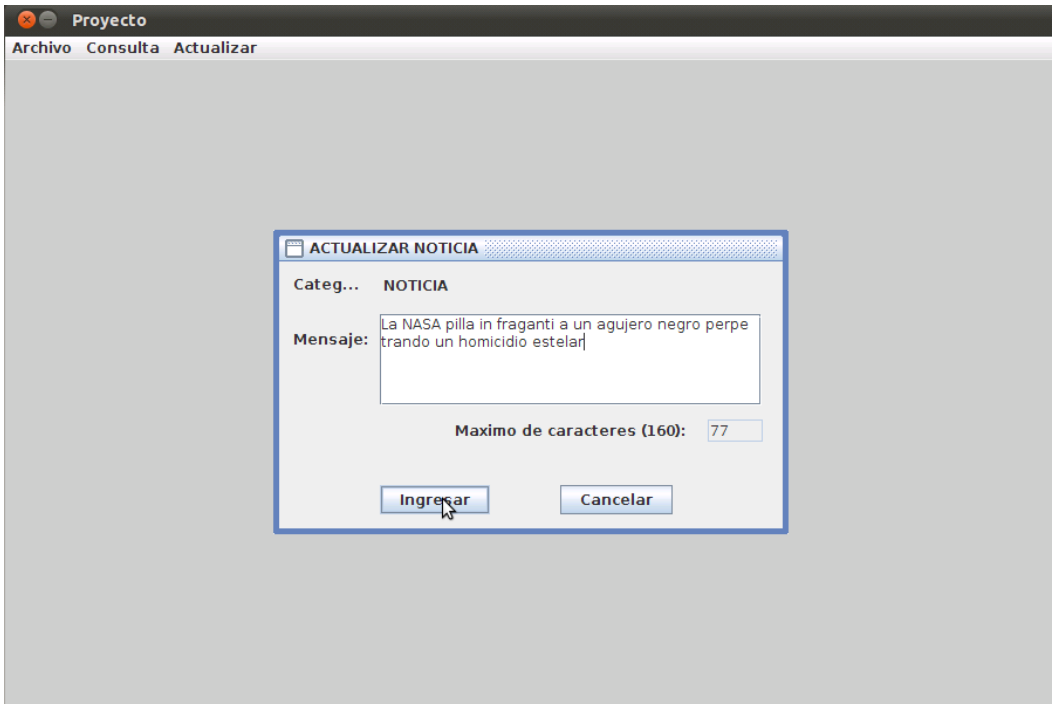


- **Consulta categoría con mayor número de compras por fecha:** Muestra la clave que mayor demanda ha tenido en un rango de fecha con el respectivo número de compras realizadas a esa palabra.



- **Actualización de mensajes para signos o noticia.**





ANEXO B:

Tabla IV. Relación de signos zodiacales a extensiones del contexto horóscopo del plan de marcado

EXTENSION / OPCION	SIGNO
1	Acuario
2	Piscis
3	Aries
4	Tauro
5	Geminis
6	Cáncer
7	Leo
8	Virgo
9	Libra
10	Escorpión
11	Sagitario
12	Capricornio

ANEXO C:

Códigos en PHP de los scripts utilizados:

Script1.php

```
#!/usr/bin/php -q
<?
require("phpagi/phpagi.php");
$agi = new AGI();

$num=preg_replace("#[^0-9]#","",$agi-
>request[agi_callerid]);//obtengo el número y
                         //elimino cualquier
                         //caracter no numérico
$key=$argv[1];  //Obtengo palabra clave que se pasa
                //como parámetro en el dialplan

/*
Conexion con la Base de datos Asterisk
*/
$db = 'test';
$dbuser = 'asterisk';
$dbpass = 'asterisk';
$dbhost = 'localhost';
```

```
mysql_connect($dbhost,$dbuser,$dbpass);
mysql_select_db("$db");

//Ingreso de datos en tabla cliente

mysql_query("INSERT INTO cliente VALUES('$num')");


//Ingreso de datos en tabla compra

mysql_query("INSERT INTO compra
(numero,clave,fecha,status)
VALUES('$num','$key',DEFAULT,'pendiente')");

?>
```

Script2.php

```
#!/usr/bin/php -q
<?
require("phpagi/phpagi.php");
$agi = new AGI();

/*
Conexion con la Base de datos Asterisk
*/
$db = 'test';
$dbuser = 'asterisk';
$dbpass = 'asterisk';
```

```
$dbhost = 'localhost';

$tdb = “60”;
//Tiempo de expiración de búsqueda por defecto es 60

mysql_connect($dbhost,$dbuser,$dbpass);
mysql_select_db("$db");

//Obtengo arreglo de compras pendientes de los
//ultimos 60 minutos
$row=mysql_query("SELECT id_compra,numero,clave FROM
(SELECT id_compra,numero,clave,status FROM compra
WHERE fecha > DATE_SUB(now(), INTERVAL '$tdb'
MINUTE)) AS ultimos WHERE status = 'pendiente'");

$i=mysql_num_rows($row);

$agi->set_variable("encolados",$i); //seteo variable
//encolados

if($i>0)
{
     $compra=mysql_fetch_array($row);
     if($compra[id_compra]) $idc=$compra[id_compra];
     $agi->set_variable("id_compra",$idc); //seteo
     //variable id_compra

     if($compra[numero]) $num=$compra[numero];
     $agi->set_variable("numero",$num); //seteo
     //variable numero
```

```
        if($compra[clave]) $key=$compra[clave];

        $rowsms=mysql_query("SELECT smstxt FROM mensaje
    WHERE clave like '$key' LIMIT 1");

        if (mysql_num_rows($rowsms)==1)
        {    $rowsms=mysql_fetch_array($rowsms);
             if ($rowsms[smstxt])
    $msgbody=$rowsms[smstxt];
             $agi->set_variable("mensaje",$msgbody);
    //Guarda cuerpo del mensaje a ser enviado
        }

    }
    ?>
```

Script3.php

```
    #!/usr/bin/php -q
    <?
    require("phpagi/phpagi.php");
    $agi = new AGI();

    $idc=$argv[1]; //id_compra


    /*
    Conexion con la Base de datos Asterisk
```

```
*/
$db = 'test';
$dbuser = 'asterisk';
$dbpass = 'asterisk';
$dbhost = 'localhost';

mysql_connect($dbhost,$dbuser,$dbpass);
mysql_select_db("$db");

//ACTUALIZO STATUS DE LA COMPRA

mysql_query("UPDATE compra SET status='ok' WHERE
(id_compra = '$idc' AND status = 'pendiente')");

?>
```

BIBLIOGRAFÍA

[1] Madsen Leif, Van MeggelenJim,Bryant Russell, **Asterisk The Future of Telephony**, Editorial O'Relly Media, 3ra Edición, 2011

[2] Long Johnny, Chaffin Larry, **Asterisk Hacking**, Editorial Elsevier, 2007

[3] Voip-Info.org, Chan Mobile,http://www.voip-info.org/wiki/view/chan_mobile , Fecha de consulta: 7-mayo-2011

[4] Voip-Info.org, Configuration OSS,http://www.voip-info.org/wiki/view/Asterisk+config+oss.con, Fecha de consulta: 20-marzo-2012

[5] PHPDocumentor v 1.4.2, PHPAGI,http://phpagi.sourceforge.net/phpagi22/api-docs/, Fecha de consulta: 18-febrero-2012

[6] Bluez Project, BlueZ - Official Linux Bluetooth protocol stack, http://www.bluez.org/download/ , Fecha de consulta: 20-junio-2011

[7] 4Front, Open Sound System Driver Download page, http://www.opensound.com/download.cgi,Fecha de consulta: 20-marzo-2012