Universidad del Azuay

Facultad de Ciencias de la Administración

Escuela de Ingeniería de Sistemas y Telemática

Sistema de información de telefonía IP para la asistencia de cartera, utilizando librerías AGI (Asterisk Gateway Interface)

Trabajo de graduación previo a la obtención del título de

Ingeniero en Sistemas y Telemática

Autor: Alvaro José Avila Alvarado

Director: Ing. Marcos Orellana Cordero

Cuenca, Ecuador

2015

Dedicatoria

Dedico este trabajo a mis padres, que con gran esfuerzo y dedicación me guiaron a lo largo de mi carrera hasta cumplir esta meta. A mis profesores, familiares y amigos que me brindaron su valioso apoyo en todo momento.

Agradecimientos

Agradezco a mi director de tesis Ingeniero Marcos Orellana Cordero, por su confianza y ayuda depositada en mí, a lo largo del tiempo que tomo culminar este trabajo de graduación.

A todos mis profesores y compañeros de curso, con los que día a día nos preparamos para llegar a ser profesionales.

Índice de Contenidos

Índice

Índice de Imágenes

Índice de Tablas

Resumen

El trabajo consiste en implementar un I.V.R. (*Interactive Voice Response*) en el software Asterisk, para realizar consultas en una base de datos, mediante tonos de marcado, a través del desarrollo de un script en el lenguaje de programación PHP.

Para esta interacción se utilizó una funcionalidad de Asterisk llamada AGI (*Asterisk Gateway Interface*), la que permite el uso de diferentes lenguajes de programación externos tales como: Perl, PHP, C/C++, Pascal, Bourne Shell y Pyton.

Finalmente se realizó pruebas de toda la implementación mediante el uso un *softphone* con el que se realizó consultas a una base de datos a través de tonos de marcado.

ABSTRACT

This research paper aims to implement an I.V.R. (Interactive Voice Response) for data lookup in the Asterisk software, using dial tones through the development of a script in PHP programming language. For this interaction, an Asterisk functionality called AGI (Asterisk Gateway Interface) was used, which allows the use of different external programming languages such as Perl, PHP, C / C ++, Pascal, Bourne Shell and Python. Finally, the testing of the entire implementation was done by using a softphone with which queries to a database using dial tone were performed.

UNIVERSIDAD DEL AZUAY
Dpto. Idiomas

Translated by,
Lic. Lourdes Crespo

Capítulo 1. Introducción y Definiciones

Introducción

Con el paso de los años Asterisk se ha convertido en una de las centralillas telefónicas privadas más populares del mundo, y es una tecnología muy aceptada en la industria de las telecomunicaciones. En la última década, la industria de las telecomunicaciones ha ido perdiendo hegemonía, los métodos de comunicación han cambiado bastante, antes la mayoría de comunicaciones se hacían mediante llamadas telefónicas, la tendencia actual es el uso de mensajes de texto, correo electrónico u otros servicios de mensajería instantánea.

Asterisk es una tecnología impresionante y se cree que es una de las mejores alternativas para la integración entre las telecomunicaciones y cualquier otra tecnología de una empresa.

Es muy raro encontrar en estos días empresas que no tengan que reinventarse con el pasar de los años. Es igualmente raro encontrar un negocio que pueda darse el lujo de reemplazar su infraestructura o equipos de comunicaciones, cada vez que estos cambian de ubicación. Las empresas de hoy necesitan flexibilidad en toda su tecnología, incluyendo las telecomunicaciones.

La nueva tecnología digital que ahora se ofrece en el mercado para las telecomunicaciones, hacen que la integración de la telefonía con el computador (C.T.I.) sea más fácil, las interacciones entre la telefonía y el computador pueden ser atendidas de manera automática con la creación de procesos necesarios, como por ejemplo, la integración de una base de datos que se encuentra en un computador con un sistema telefónico que también está en el mismo computador. (Elastix Tech)

1.1 Definición de Asterisk

Asterisk es una plataforma de telefonía de distribución libre, que está diseñada principalmente para funcionar sobre Linux. Asterisk combina muchos años de conocimientos en telefonía, en un conjunto robusto de aplicaciones de telecomunicaciones, que se encuentran estrechamente ligadas. El poder de Asterisk radica en su naturaleza adaptable, complementado por el cumplimento de ciertos estándares. (Landívar, Comunicaciones unificadas con Elastix, 2009)

Las aplicaciones como correos de voz, conferencias, cola de llamadas, música en espera, aparcamiento de llamadas y agentes, son algunas de las características por defecto integradas en el software. Más aun, Asterisk puede integrarse con otras tecnologías empresariales sin mayor problema.

1.2 Motivación de la Investigación

En los últimos años se han desarrollado programas basados en software libre que realizan todas las funciones de una central telefónica (P.B.X.). Uno de estos programas de distribución libre es Elastix basado en un entorno de programación llamado Asterisk. Entre las funciones de Elastix está incluida AGI (Asterisk Gateway Interface) la cual permite programar en algunos lenguajes no nativos; uno de los más usados es el PHP (Hypertext Pre-processor), este permite la conexión a cualquier base de datos. En nuestro medio este tema no ha sido ampliamente difundido por lo que es oportuna la construcción de una interface de consulta con las características que se describirán más adelante.

1.3 Problemática

Muchas empresas de nuestro medio todavía utilizan centrales telefónicas análogas para la comunicación, las que tienen un costo bastante elevado comparado con las centrales PBX de telefonía IP, que ofrece un notable cambio en cuestión de costos y opciones.

Es muy beneficioso para las empresas adoptar este tipo de tecnología ya que ofrece la posibilidad de recibir y enviar información a los usuarios por medio del teléfono.

Instituciones que utilizan un servidor de comunicaciones como Elastix o Asterisk no conocen muchas de las opciones que estos ofrecen, como por ejemplo los IVR (*Interactive Voice Response*). Los IVR son aplicaciones de voz interactivas que aceptan como entrada tonos marcados por el usuario, entregando distintos tipos de respuesta según la programación del sistema.

Los IVR son comúnmente implementados en empresas que reciben grandes cantidades de llamadas, a fin de reducir la necesidad de personal y los costos que el servicio ofrecido represente para dicha entidad. Entre otras, se puede mencionar a la banca telefónica.

De esta manera, se plantea utilizar la opción IVR para enrutar una llamada a una base de datos, sin la necesidad de intervención humana, reduciendo el tiempo de espera, en línea, de sus clientes.

1.4 Objetivos

1.4.1 Objetivo General

Implementar un I.V.R. (*Interactive Voice Response*) con el servicio de consulta automatizada para una base de datos de cartera a través de librerías AGI (*Asterisk Gateway Interface*).

1.4.2 Objetivos Específicos

- Realizar la instalación de componentes y configuraciones en el servidor Asterisk para el correcto funcionamiento del sistema de consultas de cartera.
- Implementar un I.V.R. que re-direccione las llamadas.
- Valorar la librería PHP-AGI y demostrar el funcionamiento de AGI (*Asterisk Gateway Interface*).
- Desarrollar un script en lenguaje PHP y construir una base de datos de cartera.
- Adaptar un conversor de texto a voz para presentar los resultados de las consultas a los usuarios.
- Establecer los casos de uso del sistema.

1.5 Metodología

La metodología que se usará se divide en 3 fases, cada una con sus actividades.

La primera fase es la de investigación en la que constan: revisiones bibliográficas, se sintetizarán conceptos y se estudiarán las herramientas a utilizar.

La segunda fase es la de implementación, la que está conformada por las configuraciones básicas de Asterisk, la instalación de todos los componentes necesarios, el desarrollo del IVR (*Interactive Voice Response*), el desarrollo del script en el lenguaje de programación PHP y por último se pondrán en práctica todos los temas anteriormente investigados para unir el script con el IVR mediante el uso de la librería AGI (*Asterisk Gateway Interface*).

La fase final es la de pruebas en el sistema mediante el uso de *softphones* (teléfonos implementados por software) analizando el funcionamiento del I.V.R. mediante la consola de Asterisk.

1.6 Alcance y resultados esperados

El presente trabajo busca diseñar y desarrollar un I.V.R. para un sistema de consultas de una base de datos de cartera de cualquier empresa a través de un servidor Asterisk.

El desarrollo del sistema partirá desde un servidor Asterisk ya creado, y la programación del script se desarrollará en el lenguaje PHP el que interactuará con Mysql que es motor de base de datos por defecto que trae Asterisk.

Para verificar el funcionamiento del I.V.R. se realizarán pruebas con un *softphone*, y el sistema I.V.R. que se pretende implementar solo ejecutará consultas de la base de datos a través de un *softphone*, no realizará ningún otro tipo de operación.

1.7 Supuestos y riesgos

Supuestos:

- Contar con todo el material bibliográfico propuesto para el desarrollo del trabajo de investigación o el necesario.
- Obtener artículos y estudios previamente realizados de los casos propuestos.
- Contar con cronograma de actividades.

Riegos:

- El tiempo estimado para el desarrollo no sea el suficiente.
- No encontrar el material bibliográfico necesario para el desarrollo.
- No tener la orientación adecuada por parte del director y no avanzar en el trabajo.
- Pérdida de material digital por causas de cualquier fenómeno natural.

Capítulo 2. I.V.R.

Introducción

El propósito de un sistema de respuesta de voz interactiva (I.V.R), es obtener información de una persona que llama realizando una acción basada en esa entrada, por lo general es realizada mediante una búsqueda de datos en un sistema externo, como lo es una base de datos, o se re-direcciona la llamada con un operador u operadora, devolviendo un resultado de voz a la persona que llama. (Sharif, 2008)

Tradicionalmente los sistemas de I.V.R han sido caros y difíciles de implementar. Asterisk cambia todo eso.

2.1 Arquitectura

El I.V.R (*Interactive Voice Responce*) basa su arquitectura de la siguiente manera:

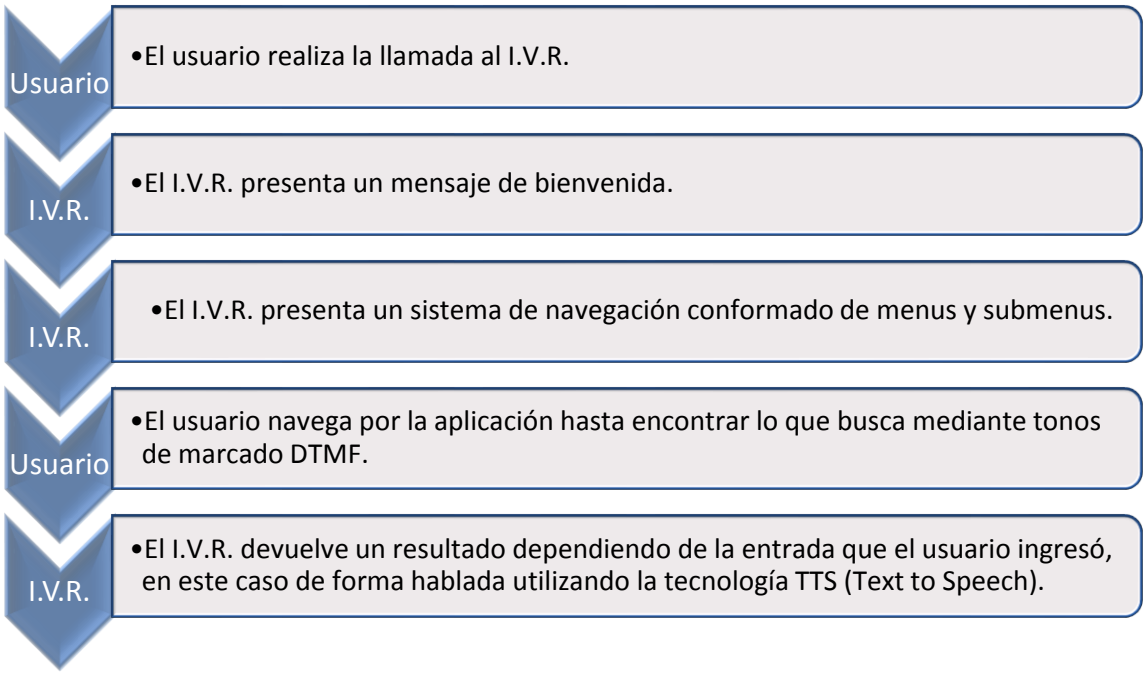


Tabla 1. Arquitectura del I.V.R.

2.1.1 I.V.R fuera de la P.B.X

En este tipo de arquitectura, los usuarios usan medios de comunicación, tales como las líneas telefónicas fijas y móviles, las que pasan primero por la P.B.X. (*Private Branch Exchange*). Después de que esto sucede, la P.B.X se encarga de re direccionar la llamada a un fax, a una operadora o a un servidor I.V.R. que este a su vez puede conectarse con una Base de datos.

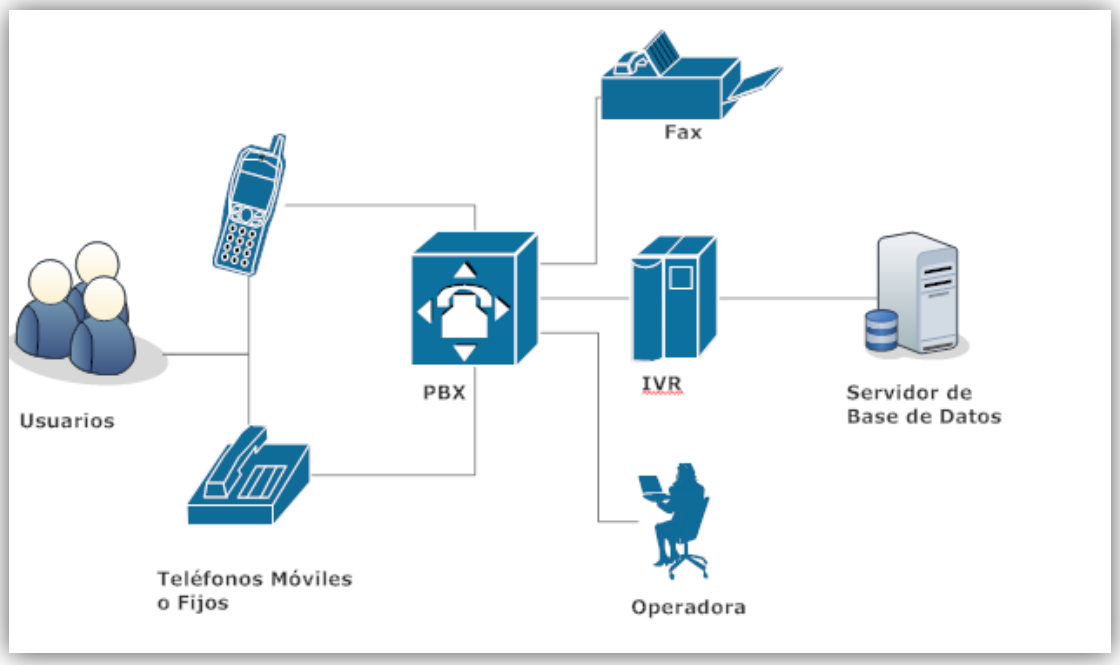


Imagen 1. I.V.R fuera de la P.B.X.

2.1.2 I.V.R dentro de la P.B.X

En este tipo de arquitectura los usuarios usan medios de comunicación, tales como las líneas telefónicas fijas y móviles, las que primero pasan por la P.B.X (*Private Branch Exchange*), la cual en este caso ya contiene al servidor I.V.R, que este a su vez maneja un plan de marcado que permite la comunicación con un fax, con una operadora, o como en este caso de este proyecto de graduación, con un servidor de Bases de Datos.

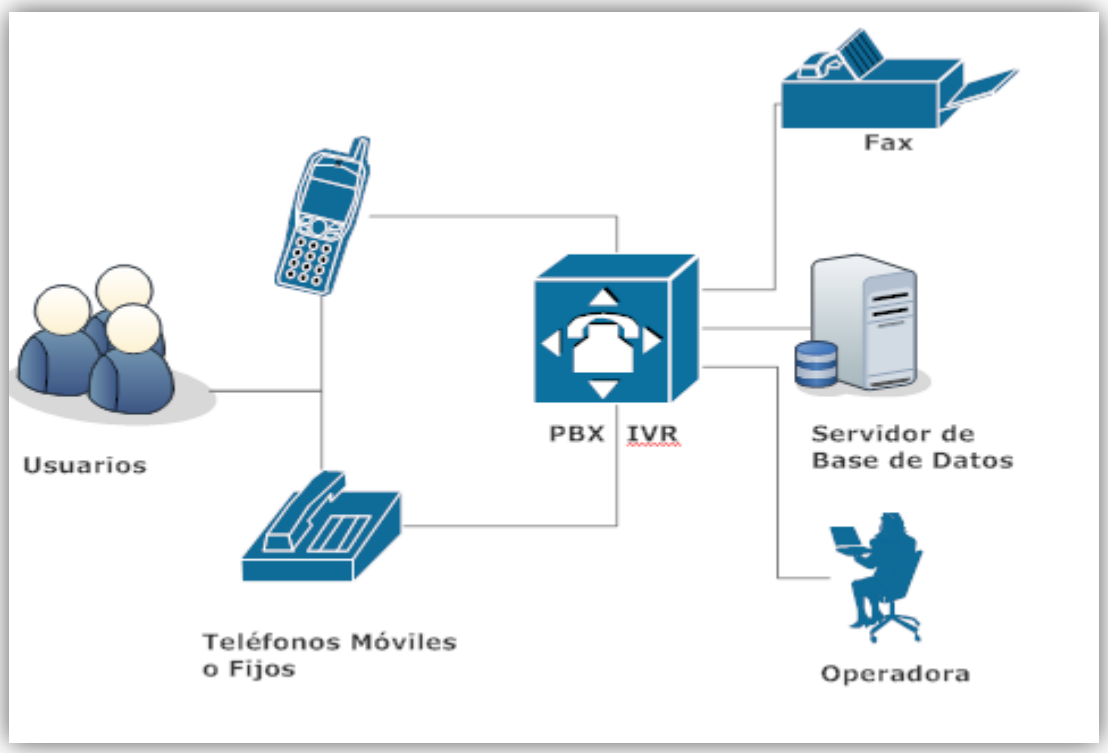


Imagen 2. I.V.R. dentro de la P.B.X.

2.2 Servicios

Los servicios de una I.V.R son empleados en las empresas o lugares en donde trabajan varios empleados en diferentes departamentos, los empleados reciben llamadas por parte de los clientes para concretar alguna consulta o cualquier tipo de transacción.

De acuerdo al tipo de requerimiento por parte de los usuarios los I.V.R presenta los siguientes servicios:

- Call Center
- Sistema de Gestión de Base de Datos

2.2.1 Call Center

En el caso de los *Call Center*, las llamadas entrantes son direccionadas a un menú que ofrece estas opciones: elegir el departamento o persona con la que el usuario desea comunicarse; el usuario ingresa el número de opción de acuerdo al menú presentado por el I.V.R y este re-direcciona la llamada para completarse la transacción. (Muñoz, 2010)

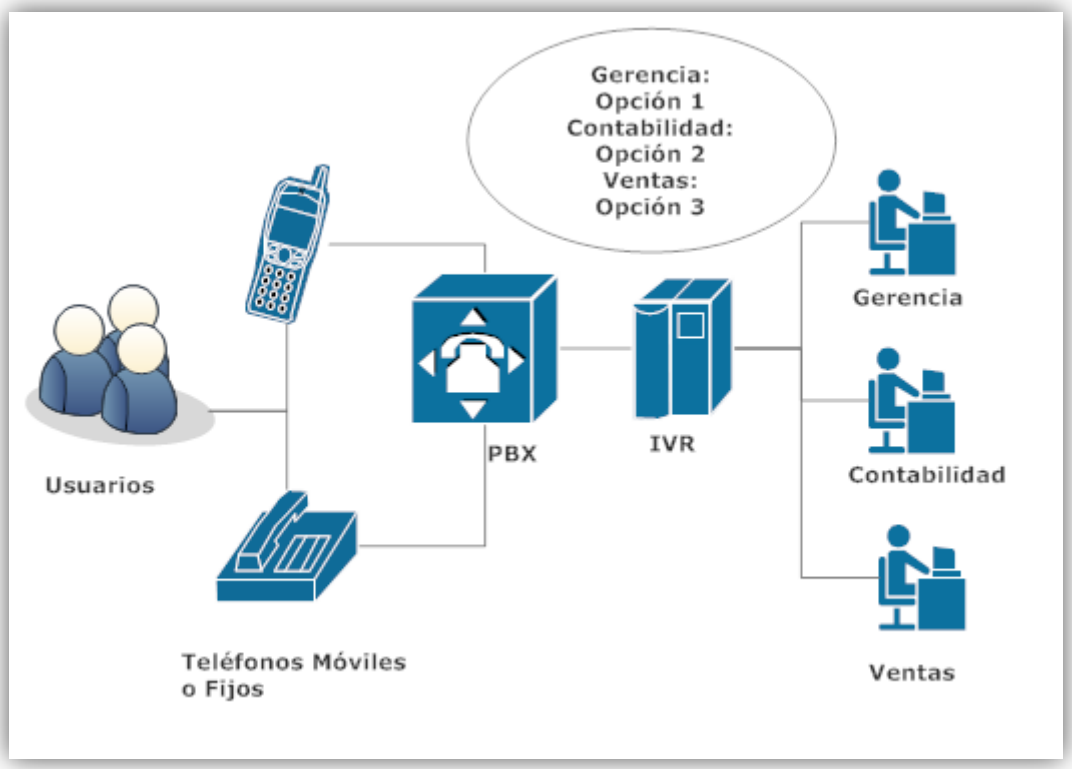


Imagen 3. Funcionamiento de un *Call Center*.

2.2.2 Sistema de Gestión de Base de Datos – D.B.M.S. (*Data Base Management System*)

En este caso el I.V.R trabaja directamente con el usuario que desea realizar una consulta en la base de datos de la empresa, sin necesidad de contactarse primero con un operador o departamento. En el momento que el usuario realiza la llamada el I.V.R presenta un mensaje de bienvenida y a continuación pedirá que ingrese un código de acceso o autentificación mediante tonos de marcado D.T.M.F. (*Dual-Tone Multi-Frequency*), en el momento que comprueba que el usuario existe en los registros de la base de datos el I.V.R presenta un menú con submenús ofreciendo diferentes tipos de consultas.

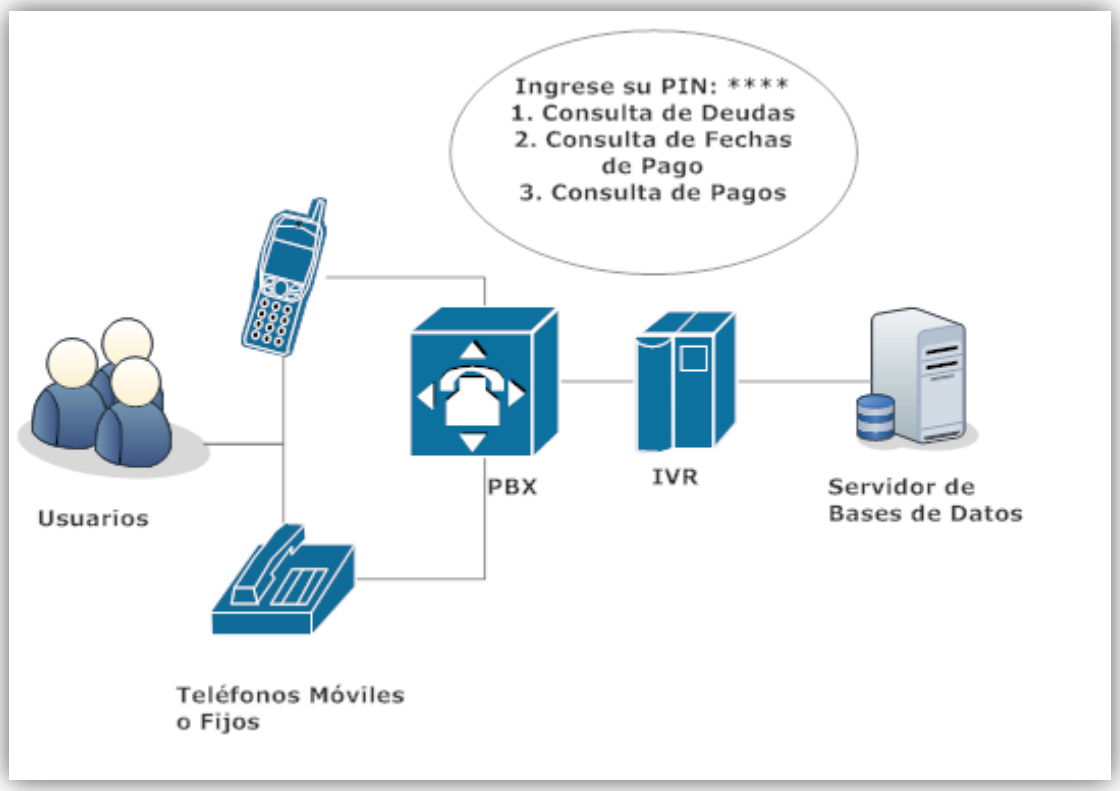


Imagen 4. Funcionamiento de un sistema de gestión de base de datos.

2.3 Tecnologías Usadas

El I.V.R ofrece diferentes métodos para que el usuario pueda ingresar sus datos, entre ellas se encuentran:

- D.T.M.F.
- T.T.S.
- A.S.R.

2.3.1 D.T.M.F. (*Dual Tone Multi Frecuency*)

D.T.M.F significa en español Tonos Duales de Multi-frecuencia, este es el término técnico para los sonidos que produce un teléfono análogo cuando son presionadas sus teclas.

Cada tecla tiene su propio sonido, cada sonido en realidad son 2 tonos que se reproducen al mismo tiempo. Uno de estos tonos es un tono de baja frecuencia, y el otro es un tono de alta frecuencia. Existen cuatro tonos de baja frecuencia que corresponden a cada fila de teclas, y cuatro tonos de alta frecuencia que corresponden a cada columna de teclas; debido a que cada tecla utiliza dos tonos, el sonido es casi imposible de

reproducir por la voz humana, por lo que existen pocas posibilidades de que esta pueda ser interpretada como un tono D.T.M.F. (Kouhfallah, 2012)

A través de líneas telefónicas regulares los tonos D.T.M.F se envían como audio, mientras tanto los teléfonos IP están diseñados para trasportar solo voz, pero estos pueden enviar el tono D.T.M.F como un paquete de datos por medio de la red. Este paquete de datos contiene con tono con gran similitud a los tonos D.T.M.F, pero no una coincidencia exacta. Utilizando teléfonos IP los tonos pueden crear ya sea un digito multiplicado o en el peor de los casos ningún digito en absoluto en el extremo del receptor debido a que pueden darse problemas de eco o un problema de pérdidas de paquetes en el momento de la transmisión.

D.T.M.F

Tabla de combinación de frecuencias

Grupo de baja frecuencia [Hz]	Grupo de alta frecuencia [Hz]				
	1209	1336	1477	1633	
697	1	2	3	A	Fila 1
770	4	5	6	B	Fila 2
852	7	8	9	C	Fila 3
941	*	0	#	D	Fila 4
	Columna 1	Columna 2	Columna 3	Columna 4	

Tabla 2. Funcionamiento de los tonos duales de multi-frecuencia.

2.3.2 T.T.S. *Text to Speech*

La utilidad de texto a voz (T.T.S.) es necesaria para la conversión de cadenas a caracteres en audio, utilizando una voz artificial que será reproducida a los usuarios que llaman.

Esta utilidad de texto a voz se encuentra ya en muchos lugares, tales como:

- I.V.R
- Portales de Voz
- Call Centers
- Correos Electrónicos
- Recordatorios
- Relojes para personas con discapacidad visual
- Aplicaciones para personas con discapacidad visual
- Teléfonos celulares y Smartphones

Existen muchos tipos de alternativas para la sintetización de voz, algunas son de bajo costo y otras son de distribución libre, como es el caso de Festival que viene por defecto en Asterisk.

2.3.2.1 Festival

Festival es una aplicación que convierte de texto a voz, que además de ser de uso libre viene incluida en Asterisk.

Festival usa un proceso de 3 pasos para convertir el texto a voz.

1. Análisis de Texto

   Esta es la primera etapa en la que Festival se encarga de normalizar el texto, esto quiere decir pasarlo a palabras más entendibles para ser usadas en la siguiente etapa.

2. Análisis Lingüístico

   Esta es la segunda etapa en la que Festival se encarga de convertir el texto normalizado en la primera etapa en fonemas. (El fonema es la unidad fonética que compone un lenguaje hablado).

3. Generación del Audio

   En esta última etapa, una vez que Festival tiene el texto en forma de fonemas, esta lo traduce en audio. Se realiza la concatenación del audio y la intercalación con los espacios que existen entre cada una de las palabras.

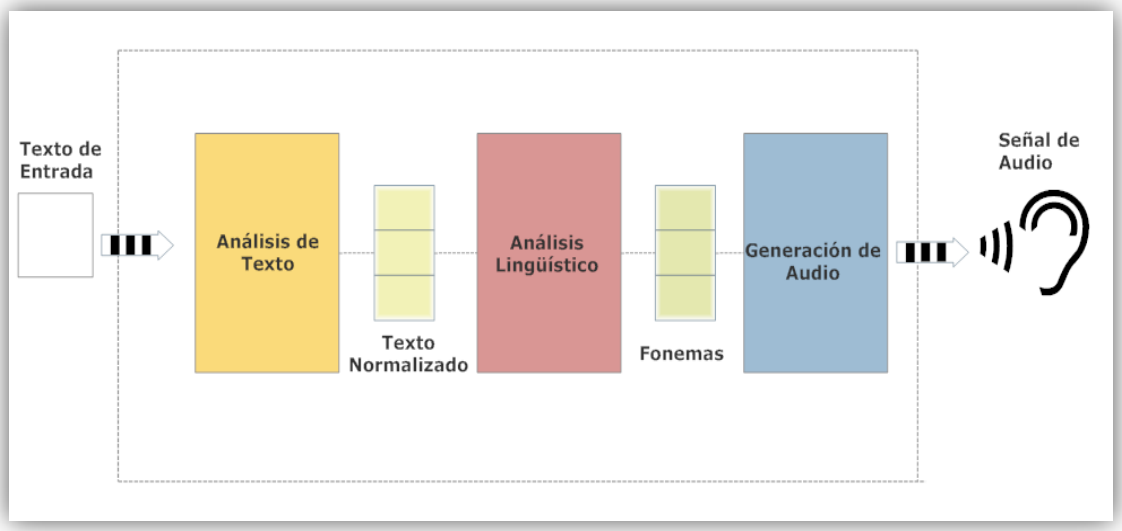


Imagen 5. Pasos para convertir de texto a voz que usa Festival.

2.3.3 A.S.R (*Automatic Speech Recognition*)

Este tipo de tecnología en español significa reconocimiento automático de voz, como su nombre lo dice reconoce las instrucciones que el usuario ingresa mediante un teléfono y las envía hacia el I.V.R. en vez de pulsar las teclas como en el caso de la D.T.M.F.

No es un sustituto perfecto de la anterior tecnología porque el reconocimiento se limita a ciertos factores tales como: el dialecto de la persona, el ruido de fondo, la calidad de comunicación, problemas de lenguaje del usuario, pronunciación, etc., mientras que en los tonos D.T.M.F tiene 16 tonos distintos que se combinan para que el usuario realice cualquier operación de manera más confiable. (Wiki Asterisk, 2015)

De manera similar a la tecnología D.T.M.F existe una entrada de una señal sonora por parte del usuario, la cual es analizada y reconocida mediante diferentes métodos, para que después pase esa misma entrada sonora en forma de texto donde el I.V.R interpretara y realizara alguna acción.

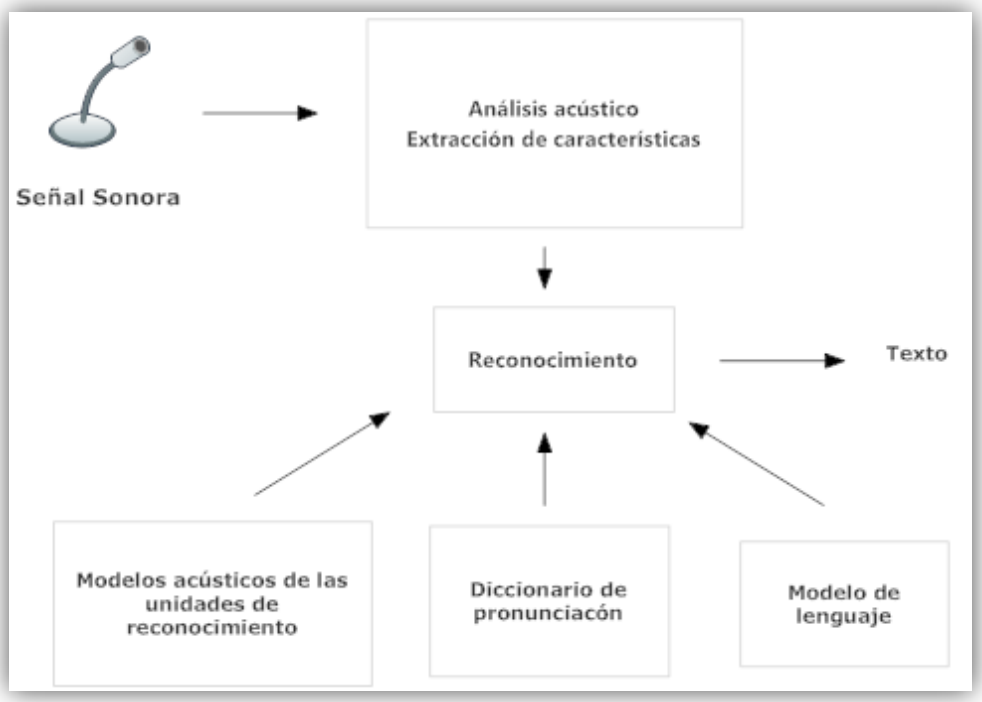


Imagen 6. Funcionamiento de un sistema de reconocimiento automático de voz

2.4 Módulos de Asterisk para construir una I.V.R.

El "Front End" de la I.V.R., la parte que interactúa con los usuarios, se puede manejar en el plan de marcado (dialplan) o mediante una fuente externa. En teoría, es posible construir un I.V.R mediante el plan de marcado. En la práctica, el plan de marcado utiliza funciones las que permite comunicarse con algo externo a ella, como por ejemplo una base de datos.

2.4.1 Curl

Esta función de Asterisk permite conectarse mediante el *dial-plan* a una página web entera mediante los métodos GET y POST con el fin de recuperar datos.

```
; Queremos obtener el identificador de llamadas de una página web.
   exten => 123,1,Curl(http://localhost/test.php,callerid=${callerid})
; Se usa la aplicación Noop( ) para presentar el resultado en la consola de Asterisk.
   exten => 123,2,Noop(${CURL})
```

2.4.2 A.G.I. (*Asterisk Gateway Interface*)

Esta función de Asterisk permite a los usuarios conectarse a aplicaciones externas, con el uso de parámetros, mediante el uso de lenguajes de programación.

En Asterisk viene por defecto instalado esta función, permitiendo el uso de lenguajes de programación tales como:

- PHP (Hypertext Pre-processor)
- C/C++
- Pascal
- Bourne Shell y
- Pyton.

A continuación, se encuentra un pequeño ejemplo de cómo llamar a un script desde el plan de marcado, el que está desarrollado en el lenguaje de programación P.H.P., enviando *n* parámetros.

```
exten => 123,1,Answer()

exten => 123,2,AGI(mi_script.php,arg1,..,argn)
```

2.4.3 A.M.I. (*Asterisk Manager Interface*)

Esta función de Asterisk significa "Administrador de la Interfaz de Asterisk", A.M.I permite la conexión de un cliente a una instancia de Asterisk, para la ejecución de comandos o la lectura de eventos basados en TCP/IP.

2.5 Conclusiones

Asterisk se ha convertido en una plataforma I.V.R muy popular en los últimos tiempos, mientras que mucha gente solo la toma en cuenta como una P.B.X libre, la realidad es que Asterisk está entrando con gran fuerza en la industria de las telecomunicaciones. Esta plataforma es adaptable fácilmente sin importar el tamaño de la empresa resolviendo así problemas que antes eran irresolubles o muy costosos de implementar.

Capítulo 3. Plan de Marcado (Dial Plan)

Introducción

El Plan de Marcado es un grupo de reglas que indican a la central telefónica cómo manejar los números que son marcados por los usuarios.

Si ha instalado los archivos de configuración de ejemplo (*sample configuration files*), en el momento de la instalación de Asterisk, entonces lo más probable es que tenga un archivo llamado "extensions_custom.conf", lo más recomendable es que usted cree su archivo de "extensions_custom.conf" desde cero. Comenzando con el archivo de ejemplo no es la mejor forma de aprender a crear un plan de marcado, pero si es un recurso muy bueno, lleno de ejemplos e ideas prácticas que se puede utilizar después de que haya aprendido a utilizar los conceptos básicos. Si la instalación de Asterisk fue como indica la mayoría de manuales, entonces el archivo de "extensions_custom.conf" se encuentra en la ruta "/etc/asterisk/" junto con muchos más archivos de configuración de ejemplo.

El plan de marcado en Asterisk está compuesto de cuatro conceptos principales: los contextos, las extensiones, las prioridades y las aplicaciones. Después de la revisión de estos conceptos, se tendrá la capacidad de construir un plan de marcado básico pero funcional. (Goncalves, 2007)

3.1 Contextos

Los contextos en Asterisk son utilizados para agrupar o categorizar procedimientos y extensiones en función de la marcación que hace el usuario.

En el momento que el usuario crea una extensión, ya sea esta con un protocolo SIP (*Session Initiation Protocol*) o lAX (*Inter-Asterisk eXchange protocol*), se le asigna un contexto que indica a qué lugar en el dial plan se direccionara la llamada. En el caso de que se no se haya asignado a ningún contexto, Asterisk lo pondrá como defecto en "default".

Los contextos se definen colocando su nombre dentro de corchetes (**[ ]**). El nombre puede estar formado por letras de la A-Z (mayúsculas y minúsculas), números del 0 al 9 y el guion bajo.

Un contexto para las llamadas entrantes puede tener esta forma:

[**Entrantes**]

Como un simple ejemplo, imaginemos que existen 2 empresas que comparten el servidor de Asterisk. Si se asigna la atención automática de cada compañía en su propio contexto, esto las haría completamente separadas la una de la otra. Este tipo de opciones permite al usuario definir de forma independiente lo que ocurre cuando, por ejemplo, se marca la extensión 0: Las personas que marcan 0 en el menú de voz de la compañía A, recibirá la llamada la recepcionista de la compañía A, mientras que las personas que marquen 0 al menú de voz de la compañía B, recibirá la llamada la recepcionista de la empresa B (esto se supone de antemano que se ha dicho a Asterisk que transfiera las llamadas a la recepcionista cuando se marque 0). (Madsen, Meggelen, & Bryant, 2011)

El gráfico a continuación muestra el uso del contexto "prueba" utilizado en un usuario SIP con el plan de marcado ubicado en extensions.conf.

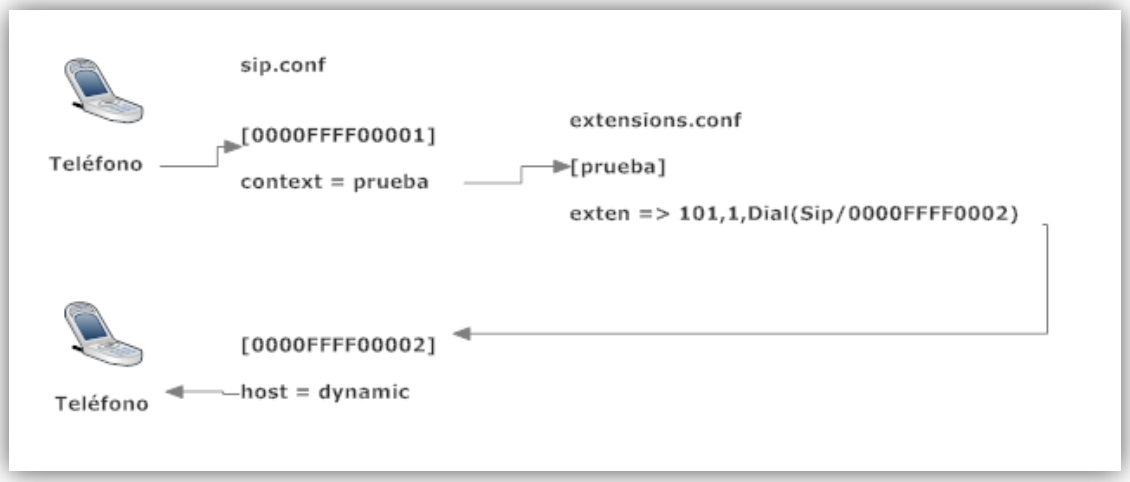


Imagen 7. Funcionamiento de los contextos en Asterisk

Un uso importante de los contextos, quizá el más importante, es el de proporcionar seguridad. Al utilizar correctamente los contextos, se puede otorgar a ciertos usuarios accesos a funciones (como llamadas a larga distancia) que no están a disposición de otros. Si no se diseña el plan de marcado con cuidado, sin darse cuenta se está dando acceso a que los usuarios usen de manera abierta el sistema.

3.2 Extensiones

En el mundo de las telecomunicaciones la palabra extensión, por lo general, se refiere a un identificador numérico, que si se es marcado hará sonar un teléfono o algún otro recurso del sistema como el correo de voz. En Asterisk, una extensión es mucho más poderoso que eso, ya que define una serie única de pasos (cada paso contiene una aplicación), a través la cual el sistema podrá tomar esa llamada.

Dentro de cada contexto, pueden ser definidas muchas o pocas extensiones según sean necesarias. Cuando es activada una extensión en particular (por una llamada entrante) Asterisk seguirá los pasos definidos por esa extensión. Son las extensiones, por lo tanto, que especifican lo que sucede con las llamadas ya que ellas hacen su camino en el plan de marcado (dialplan).

La sintaxis que se usa para asignar una extensión es la palabra “exten” seguido por una flecha formada por el signo igual y el signo mayor que, de esta manera:

```
exten  =>
```

Esto es seguido por el nombre (o número) de la extensión. Cuando se trata de sistemas de telefonía tradicionales, se tiende a pensar que las extensiones son los números que se marcan para hacer timbrar otro teléfono.

Cada paso en una extensión se compone de 3 componentes:

- El nombre (o número) de la extensión
- La prioridad (cada extensión puede tener varios pasos; el número de paso se le conoce como prioridad)
- La aplicación (o comando) que se llevara a cabo.

Estos 3 componentes deben separarse con el uso comas, por ejemplo:

```
exten => nombre,prioridad,aplicación()
```

Aquí un ejemplo simple:

```
exten => 123,1,Answer()
```

En el ejemplo anterior el nombre de la extensión es 123, la prioridad es 1, y la aplicación es responder (answer).

3.3 Prioridades

Cada extensión puede tener varias etapas llamadas prioridades. Las prioridades se numeran secuencialmente, comenzando con el número 1 como el más importante. Cada prioridad ejecuta una aplicación específica.

Como ejemplo: La siguiente extensión contestaría el teléfono (en el número de prioridad 1), y luego colgaría (en el número de prioridad 2).

```
exten => 123,1,Answer()
exten => 123,2,Hangup()
```

3.3.1 Prioridades no numeradas

En las versiones anteriores de Asterisk, la numeración de las prioridades causó muchos problemas. Imagine crear una extensión con más de 20 prioridades todas numeradas manualmente.

En las versiones actuales de Asterisk se solucionó este problema. Se introdujo el uso de la prioridad “n”. Cada vez que se encuentre con una prioridad “n” se toma el número de la prioridad anterior y suma 1, teniendo en cuenta que siempre se debe tener por lo menos escrito la prioridad número 1. Este método hace que sea más fácil hacer cambios en el plan de marcado, ya que no se tiene que mantener una numeración de todos sus pasos. Como ejemplo podría ser algo como lo siguiente:

```
exten => 123,1,Answer()
exten => 123,n,aplicación1
exten => 123,n,aplicación2
exten => 123,n,aplicación3
exten => 123,n,Hangup()
```

3.3.2 El operador “same =>”

En los esfuerzos de simplificar la codificación del plan de marcado Asterisk creo el operador “same”. Esta es utilizada para ya no escribir el nombre de la extensión

completa, si no, solo se escribe el operador “same” seguido de la prioridad y la aplicación:

```
exten => 123,1,Answer()
  same => n,aplicación1
  same => n,aplicación2
  same => n,aplicación3
  same => n,Hangup()
```

El sangrado no es requerido, pero ayuda a entender mejor la lectura. Con el uso de este operador en el plan de marcado hará que sea más fácil copiar el código de una extensión a otra.

3.3.3 Etiquetas de prioridad

Las etiquetas de prioridad permiten asignar un nombre a cualquiera de las prioridades dentro de una extensión. Esto es realizado para asegurar que se puede hacer referencia a una prioridad por medio de algo más que su número (el cual probablemente no se conoce, ya que los planes de marcado de ahora ya casi no utilizan prioridades numeradas).

Para asignar una etiqueta de texto a cualquier prioridad, solo se tiene que añadir la etiqueta dentro de paréntesis después de la prioridad de la siguiente forma:

```
exten => 123,n(etiqueta),aplicación()
```

Un error muy común al escribir las etiquetas de prioridad es insertar una coma entre la “n” y “(”, así:

```
exten => 123,n,(etiqueta),aplicación()          <- Esto no va a funcionar.
```

Este tipo de errores hará que el plan de marcado no pueda utilizar esta línea.

3.4 Aplicaciones

Las aplicaciones en el plan de marcado es uno de los elementos más importantes. Cada aplicación puede realizar acciones diferentes tales como:

- Answer() .- Aceptar una llamada entrante.
- Playback() .- Reproducir un mensaje de bienvenida.
- Hangup() .- Terminar con una llamada.
- AGI().- Buscar algo en una base de datos (AGI).

En los ejemplos anteriores se mostraron 2 aplicaciones muy utilizadas: Answer() y Hangup(), las cuales son usadas para contestar y terminar una llamada respectivamente.

Para el uso de las 2 aplicaciones nombradas anteriormente no se necesitan otras instrucciones para que realicen su trabajo. Sin embargo, la mayoría de aplicaciones necesitan información adicional. Estos elementos adicionales se les conocen con el nombre de parámetros, los cuales pasan del plan de marcado hacia la aplicación para efectuar algún tipo de operación específica. Para realizar ese proceso, se debe colocar entre paréntesis a continuación del nombre de la aplicación.

3.5 Conclusiones

Es imprescindible que este capítulo quede claro porque es la base fundamental de este trabajo de graduación. Hay que tener claro todos los conceptos aplicados para poderlos poner en práctica al momento de crear el plan de marcado.

Capítulo 4. A.G.I. (*Asterisk Gateway Interface)*

Introducción

El plan de marcado de Asterisk ha evolucionado hasta convertirse en una interfaz sencilla de programación, pero muy potente al momento de gestionar las llamadas. Sin embargo, muchas personas, especialmente los que tienen experiencia previa en programación, prefieren implementar su propio plan de marcado personalizado en un diferente lenguaje de programación. El uso de otro lenguaje de programación también puede permitir que el usuario utilice el código para la integración con otros sistemas, como lo es una base de datos. AGI (*Asterisk Gateway Interface*) de Asterisk permite el desarrollo de un script para el control de llamadas, codificado en el lenguaje de programación de su elección. (Simionovich, [s.a.])

El siguiente diagrama U.M.L. describe los pasos que ocurren cuando un script AGI es ejecutado desde el plan de marcado de Asterisk.

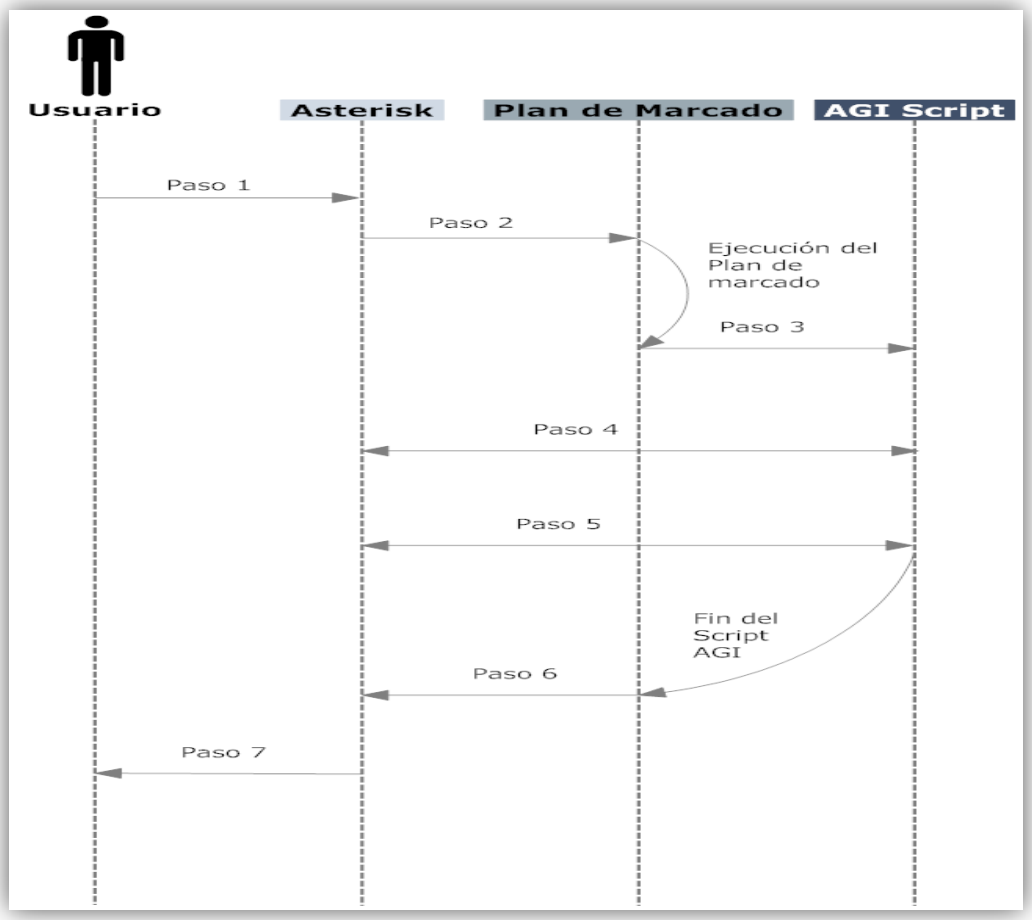


Imagen 8. U.M.L. de la transmisión de datos del usuario hacia el script A.G.I.

Como se puede ver, la mayor parte de la interacción entre Asterisk y el script AGI sucede entre el tercer y quinto paso. Examinemos que pasa en esos pasos con el siguiente ejemplo:

Como primer paso se agrega la siguiente línea en esta ruta: /etc/Asterisk/extensions_custom.conf

```
exten => 9999,1,AGI(consulta.php)
```

A continuación, se debe crear el script “consulta.php” en la ubicación: /var/lib/Asterisk/agi-bin/

```
#!/usr/bin/php -q

<?php

set_time_limit(30);

require_once "phpagi.php";    //Se incluye la clase de PHP que utiliza AGI
error_reporting(E_ALL);


$agi = new AGI();                    //Se crea un objeto tipo AGI
$agi->answer();                      //Contesta la llamada

tts($agi, "Hola Mundo");   //Utiliza la función tts para reproducir un texto

$agi->hangup();                      //Fin de la llamada

function tts($agi, $texto) {
        if ($texto!="") {
                $agi->verbose("$texto");     //Muestra en la consola de Asterisk
                $tts=iconv("UTF-8","ISO-8859-1", "$texto");
                $agi->text2wav($tts);      //Transforma de texto a voz
        }
        else {
                $agi->verbose('No se recibió ningún texto. Intentelo nuevamente');
```

```
        }
}
?>
```

Tabla 3. Ejemplo "Hola Mundo" usando Asterisk

El momento que se realiza una llamada a la extensión 9999, ingresa al script y se reproduce el mensaje “hola mundo”.

Para le reproducción del mensaje se utiliza la función llamada “tts”, cuyo rol es el de transformar un texto a una reproducción de audio.

Con el uso de la consola de Asterisk se visualiza la operación desde que el usuario realiza la llamada hasta que esta finaliza.

Para el ingreso a la consola de Asterisk se digita el siguiente comando en la terminal del sistema operativo:

```
asterisk –r
```

Una vez que se encuentre en la consola, se realiza la llamada telefónica a la extensión 9999 y el resultado será el siguiente:

```
*CLI> agi set debug on

AGI Debbuggin Enabled

 == Using SIP RTP CoS mark 5

   -- Executing [9999@from-internal:1] AGI("SIP/1002-00000014", "consulta.php") in new stack

   -- Launched AGI Script /var/lib/asterisk/agi-bin/consulta.php

<SIP/1002-00000014>AGI Tx >> agi_request: consulta.php

<SIP/1002-00000014>AGI Tx >> agi_channel: SIP/1002-00000014

<SIP/1002-00000014>AGI Tx >> agi_language: en
```

<SIP/1002-00000014>AGI Tx >> agi_type: SIP

<SIP/1002-00000014>AGI Tx >> agi_uniqueid: 1409686636.20

<SIP/1002-00000014>AGI Tx >> agi_version: 1.8.20.0

<SIP/1002-00000014>AGI Tx >> agi_callerid: 1002

<SIP/1002-00000014>AGI Tx >> agi_calleridname: device

<SIP/1002-00000014>AGI Tx >> agi_callingpres: 0

<SIP/1002-00000014>AGI Tx >> agi_callingani2: 0

<SIP/1002-00000014>AGI Tx >> agi_callington: 0

<SIP/1002-00000014>AGI Tx >> agi_callingtns: 0

<SIP/1002-00000014>AGI Tx >> agi_dnid: 9999

<SIP/1002-00000014>AGI Tx >> agi_rdnis: unknown

<SIP/1002-00000014>AGI Tx >> agi_context: from-internal

<SIP/1002-00000014>AGI Tx >> agi_extension: 9999

<SIP/1002-00000014>AGI Tx >> agi_priority: 1

<SIP/1002-00000014>AGI Tx >> agi_enhanced: 0.0

<SIP/1002-00000014>AGI Tx >> agi_accountcode:

<SIP/1002-00000014>AGI Tx >> agi_threadid: -1219470448

<SIP/1002-00000014>AGI Tx >>

<SIP/1002-00000014>AGI Rx << ANSWER

<SIP/1002-00000014>AGI Tx >> 200 result=0

<SIP/1002-00000014>AGI Rx << VERBOSE "Hola Mundo" 1

consulta.php: Hola Mundo

<SIP/1002-00000014>AGI Tx >> 200 result=1

<SIP/1002-00000014>AGI Rx << STREAM FILE

```
/var/spool/asterisk//tmp//text2wav_d501194c987486789bb01b50dc1a0adb "" 0

  --
```

Se observa que el comando “verbose” muestra en pantalla lo que el usuario escucha durante la llamada, indicando que se conectó correctamente el plan de marcado el script desarrollado en el lenguaje de programación PHP.

4.1 Variantes de A.G.I. (*Asterisk Gateway Interface*)

Existen algunas variantes de A.G.I. que principalmente difieren en el método usado al momento de comunicarse con Asterisk. Es una buena idea estar al tanto de todas las opciones para poder tomar la mejor decisión en base a las necesidades que requiere el usuario al momento de crear su aplicación.

4.1.1 Procesos basados en A.G.I. (*Asterisk Gateway Interface*)

Los procesos basados en A.G.I. son simples variantes de A.G.I. En el ejemplo anterior se mostró como invocar a un script mediante el uso de AGI() desde el plan de marcado de Asterisk. La aplicación a ejecutar es especificada como el primer argumento de AGI (aplicación.php), en el caso de que la aplicación este creada en otra ruta a la predeterminada “/var/lib/asterisk/agi-bin” se indica la ruta completa en AGI (ruta/aplicación.php). Los argumentos que se pasan a la aplicación se llaman “argumentos adicionales”.

```
exten => 123,n,AGI(aplicacion.php,CALLERID(num))
```

En este caso se pasó el argumento “CALLERID”, el que puede ser recuperado en la aplicación llamada “aplicación.php”.

Como recomendación hay que tener en cuenta que para poder usar un archivo fuera del *dial plan*, se necesita darle los respectivos permisos para poderlo ejecutar.

4.1.2 FastAGI

FastAGI es el término utilizado para el control de llamadas utilizando AGI a través de una conexión T.C.P. (Protocolo de Control de Transmisión).

FastAGI se utiliza al invocar la aplicación AGI() en el plan de marcado de Asterisk, pero en lugar de tener el nombre de la aplicación a ejecutar, se usara una dirección agi://. Por ejemplo:

```
exten => 500,1,AGI(agi://127.0.0.1)
```

El puerto predeterminado para la conexión FastAGI es el 4573.

```
exten => 500,1,AGI(agi://127.0.0.1:4573)
```

De la misma forma que lo hace AGI, FastAGI también permite pasar argumentos a una aplicación.

```
exten => 500,1,AGI(agi://127.0.0,argumento1,argumento2,argumento3)
```

4.2 Comunicación con A.G.I.

En esta sección se muestra con más detalle, acerca de cómo una aplicación se comunica con Asterisk una vez que AGI() es invocada en el plan de marcado.

4.2.1 Configuración de una sesión de AGI.

Cuando una aplicación AGI() es invocada desde el plan de marcado de Asterisk, alguna información se pasa a la aplicación AGI para configurar la sesión de AGI. A continuación, se muestra qué medidas se adoptan al principio de una sesión AGI.

Para una aplicación AGI basada en procesos o para una conexión a un servidor con FastAGI, las variables que se muestran a continuación serán la primera información enviada desde Asterisk hacia la aplicación.

Variable	Valor / Ejemplo	Descripción
agi_request	Consulta.php	Este es el primer argumento que se pasa. En este caso es el nombre de la aplicación que ha sido ejecutada. Si se usa FastAGI cambiaría el nombre de la aplicación por la dirección del servidor.
agi_channel	SIP/1002-00000014	El nombre del canal que ha ejecutado la aplicación AGI().
agi_languaje	en	El lenguaje asignado en el agi_channel.
agi_type	SIP	El tipo de protocolo usado en el agi_channel.
agi_uniqueid	1409686636.20	El número de identificación único usado en el agi_channel.
agi_version	1.8.20.0	La versión de Asterisk en uso.
agi_callerid	1002	La cadena completa de identificación de llamadas usada en el agi_channel.
agi_calleridname	device	El nombre de la cadena de identificación usada en el agi_channel.
agi_callingpres	0	La presentación de llamadas asociado con el identificador en el agi_channel.
agi_callingani2	0	El ANI2 llamante asociado

		en el agi_channel.
agi_callington	0	El identificador de llamadas TON (tipo de número) asociado en el agi_channel.
agi_callingns	0	El numero marcado TNS (Selector de Tránsito de Red) asociado con el agi_channel.
agi_dnid	9999	El número de extensión marcada asociada con el agi_channel.
agi_rdnis	unknown	El número de redireccionamiento asociado con el agi_channel.
agi_context	from-internal	El nombre del contexto del plan de marcado en el cual fue ejecutado la aplicación AGI().
agi_extension	9999	La extensión en el plan de marcado que el agi_channel estaba ejecutando cuando se corrió la aplicación AGI().
agi_priority	1	La prioridad de la agi_extension en el agi_context que tenía la aplicación AGI().
agi_enhanced	0.0	Indica quien utilizo esta aplicación en el plan de marcado. En el caso de que se use agi el valor será 0.0.

agi_accountcode	myaccount	La cuenta asociada con el agi_channel.
agi_threadid	1219470448	El threadid del hilo de Asterisk que ejecuta la aplicación AGI().
agi_arg_<argument number>	my argument	Estas variables proporcionan el contenido de los argumentos adicionales prestados a la aplicación AGI().

Tabla 4. Variables de sesión del ejemplo "Hola Mundo"

Las variables mostradas en la tabla anterior fueron tomadas del ejemplo “Hola mundo”, y obtenidas desde la consola de Asterisk en el momento de la ejecución de la aplicación “consulta.php”.

4.2.2 Comandos y Respuestas.

Una vez que la sesión de AGI está establecida, Asterisk comienza a realizar el procesamiento de la llamada en respuesta a los comandos enviados desde la aplicación AGI(). Tan pronto como un comando AGI ha sido emitido a Asterisk, ningún otro comando u orden será procesada en ese canal hasta que el comando u orden se haya completado. Cuando termina de procesar un comando u orden, Asterisk responderá con un resultado.

Estos comandos son los responsables de realizar todas las operaciones de AGI, desde capturar los tonos de teclado marcados por los usuarios, hasta la transformación de texto a voz para la representación de los resultados.

Una lista completa de comandos de AGI disponibles puede ser recuperada desde la consola de Asterisk, con la ejecución del siguiente comando en la consola:

```
agi show commands
```

answer	Contesta una llamada entrante.
asyncagi break	Termina una sesión de AGI asíncrono y

	tiene el canal de retorno para el plan de marcado de Asterisk.
channel status	Recuperar el estado del canal. Esto se utiliza para recuperar el estado actual del canal, tal como: contestado, colgado o timbrando.
control stream file	Transmitir los contenidos de una archivo a un canal, también permite al canal controlar el flujo.
database del	Elimina una llave/valor de la base de datos.
database deltree	Elimina un árbol de llaves/valores incorporada en la base de datos.
database get	Recupera el valor de una clave en la base de datos.
database put	Establece el valor de una llave en la base de datos.
exec	Ejecuta una aplicación determinada.
get data	Lee dígitos de la persona que está llamando.
get full variable	Evalúa una expresión del plan de marcado. Usted puede enviar una cadena de caracteres que contienen variables o funciones del plan de marcado, y Asterisk devolverá el resultado después de hacer las sustituciones apropiadas.
get option	Trasmitir un archivo de sonido a la espera de que la persona que llama presione un digito.
get variable	Recuperar el valor de una variable de canal.
hangup	Colgar el canal.
noop	No hacer nada. Recibirá un resultado de la

	respuesta de este comando.
receive char	Recibe un solo carácter. Esto solo funciona para los tipos de canales que lo apoyan, como IAX2 utilizando marcos de texto o SIP utilizando el método de mensaje.
receive text	Recibe un mensaje de texto. Esto solo funciona en los mismos casos que receive char.
record file	Graba el audio de la persona que llama en un archivo.
say alpha	Dice una cadena de caracteres dada.
say date	Dice la fecha.
say datetime	Dice una fecha y hora determinada utilizando un formato específico.
say digits	Dice una cadena de dígitos dada. Por ejemplo, el número 100 se diría como: uno, cero, cero.
say number	Dice un número. Por ejemplo, el número 100 se diría como: cien.
say phonetic	Dice una cadena de caracteres, pero usando una palabra común de cada letra (Alpha, Bravo, Charlie).
say time	Dice la hora.
send image	Envía una imagen hacia un canal.
send text	Envía mensajes de texto a un canal que tenga soporte (IAX2, SIP).
set autohangup	Programa el canal que va a ser colgado en un punto determinado del futuro.
set callerid	Establece el nombre de la persona que llama y su número de identificación en el canal.
set context	Establece el actual contexto del plan de

	marcado en el canal.
set extension	Establece la actual extensión del plan de marcado en el canal.
set music	Inicia o detiene la música de espera en el canal.
set priority	Establece la prioridad del plan de marcado en el canal.
set variable	Establece una variable de canal a un valor dado.
speech activate grammar	Activa una gramática que se ha cargado.
speech create	Inicializa el reconocimiento de voz. Esto se debe realizar antes de otro comando de habla.
speech deactivate grammar	Desactiva una gramática.
speech destroy	Destruye los recursos que se asignaron para hacer el reconocimiento de voz. Este comando debe ser el último comando de voz utilizado.
speech load grammar	Carga una gramática.
speech recognize	Suena un aviso y comienza el reconocimiento por voz, así como esperar a que presione los dígitos del teclado.
speech set	Estable los ajustes del motor de reconocimiento. Los ajustes que están disponibles son especificados para el motor de reconocimiento en uso.
speech unload grammar	Descarga una gramática.
stream file	Transmite el contenido de un archivo a un canal.
verbose	Envía un mensaje detallado al canal registrador. Los mensajes detallados aparecen en la consola de Asterisk.
wait for digit	Espera a que la persona que llame

	presione un dígito.

Tabla 5. Comandos de A.G.I. disponibles en Asterisk.

Para la obtención de información más detallada de un comando en específico, por ejemplo “ANSWER” se ejecuta el siguiente comando:

```
agi show commands topic ANSWER
```

En el anexo número 1 se encuentran detallados cada uno de los comandos.

4.2.3 Terminar una sesión de AGI.

La aplicación AGI puede salir o terminar su sesión en cualquier momento. Mientras que el canal no haya sido colgado antes de que finalice la sesión, el plan de marcado se seguirá ejecutando. Si se cuelga el canal mientras la sesión de AGI este activa, Asterisk dará una notificación del problema, para que la aplicación pueda ajustar su funcionamiento según corresponda.

La próxima cosa que pasa después de que un canal se cuelga es que una notificación explicita del “hangup” se envía hacia la aplicación en uso. Para AGI basado en procesos, la señal sighup será enviada al proceso para notificarle que será colgada. Para una conexión FastAgi, Asterisk envía una línea que contiene la palabra Hangup.

4.3 Marcos de Desarrollo

Existe un número de marcos o librerías que hacen la programación en AGI más fácil. A continuación, se encuentran listados algunos de ellos con su respectiva página web para obtener más información.

Marco de Desarrollo	**Lenguaje**	**URL**
Adhearsion	Ruby	http://adhearsion.com/
StarPy	Python	http://starpy.sourceforge.net/
Asterisk-Java	Java	http://asterisk-java.org/
Asterisk-Perl	Perl	http://asterisk.gnuinter.net/
PHP AGI	PHP	http://phpagi.sourceforge.net/

Tabla 6. Marcos de desarrollo para A.G.I.

En este proyecto se hará uso del marco de desarrollo PHP-AGI, el cual en su página web contiene toda la información necesaria junto a su clase que interactúa con el

lenguaje de programación PHP a la perfección, la cual se usara en la programación del script de este proyecto de graduación.

4.4 PHP AGI

PHP AGI es una clase que contiene todos los comandos AGI previamente definidos de Asterisk, permitiendo que la programación en el lenguaje de programación P.H.P. sea más fácil, debido a que todas las funciones y comandos se encuentran centralizadas en un solo archivo. (Landívar, Comunicaciones unificadas con Elastix, 2009)

Esta clase es descargable desde su propia página web, y debe ser ubicada dentro del directorio **/var/lib/Asterisk/agi-bin/** para poderla usar. En la misma ruta también se ubica la aplicación AGI desarrollada en PHP, la que será invocada desde el plan de marcado.

Para el uso de esta clase con cualquier aplicación, solo se necesita agregar esta línea al comienzo del código PHP.

```
require_once "phpagi.php";
```

De esta manera se podrá ya utilizar esta clase y los comandos de Asterisk de forma más rápida y directa.

4.5 Conclusiones

AGI ofrece una potente interfaz de Asterisk que permite implementar un control de llamadas en el lenguaje de programación a su elección. Existen múltiples enfoques que se pueden tomar a la hora de implementar una aplicación AGI. Algunos enfoques pueden proporcionar un mejor rendimiento, pero a costa de una mayor complejidad. AGI proporciona un entorno de programación que puede hacer más fácil la integración de otros sistemas con Asterisk, o simplemente proporcionar un entorno de programación de control de llamadas más cómodo para cualquier programador experimentado.

Capítulo 5. Implementación de la Aplicación

Introducción

En este capítulo se realizará la implementación de un servicio de consultas automatizadas a una base de datos Mysql de cartera a través de la librería phpAGI (*Asterisk Gateway Interface*).

Serán detalladas paso a paso las instrucciones para la instalación de los componentes necesarios con sus debidas configuraciones.

5.1 Instalación y configuración de los servicios de Asterisk

AsteriskNOW es una distribución desarrollada en Linux lista para usar. Contiene código abierto de Asterisk, es totalmente gratis y puede ser descargada directamente desde la página oficial de Asterisk.

http://www.asterisk.org/downloads

Esta distribución se encuentra disponible en un formato ISO que a su vez instala: Centos, Asterisk, Mysql y la GUI FreePBX.

5.2.1 Servicios de Asterisk

Existen comandos que se utilizarán para manejar los servicios de Asterisk, estos deberán ser ingresados mediante terminal o consola.

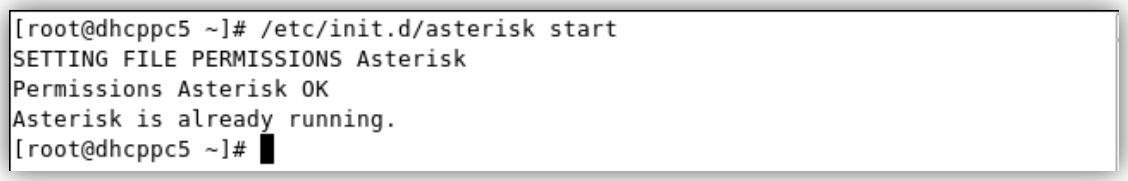


Imagen 9. Iniciar el servicio Asterisk

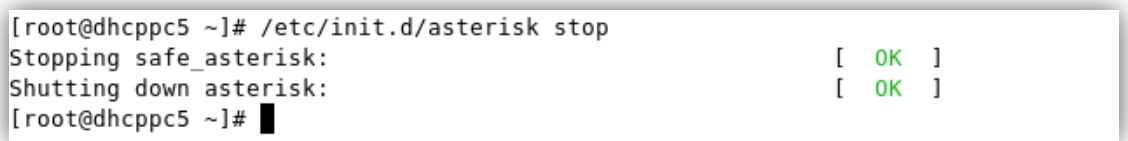


Imagen 10. Detener el servicio Asterisk

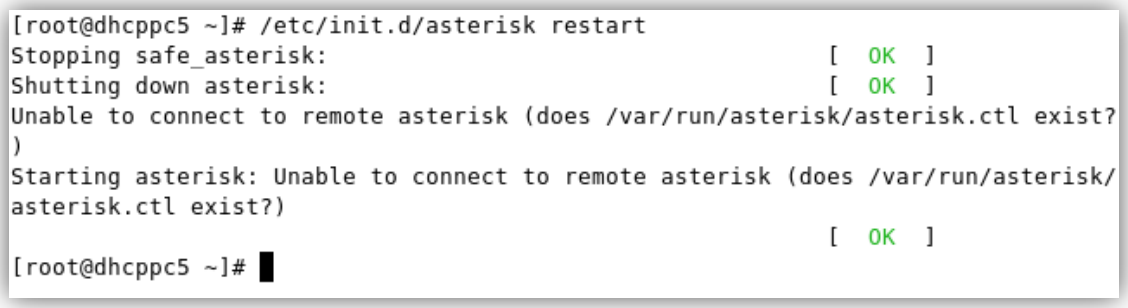


Imagen 11. Detener e iniciar el servicio Asterisk

Los siguientes comandos son también ingresados a través de la terminal de Centos, y son usados para ingresar a la consola CLI de Asterisk, la que es usada para la revisión detallada de toda la actividad que se realiza en el momento que es ejecutado el script P.H.P.

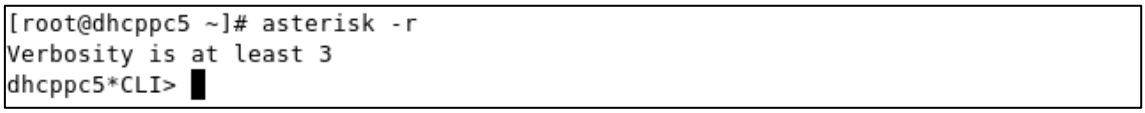


Imagen 12. Ingresar a la consola CLI de Asterisk

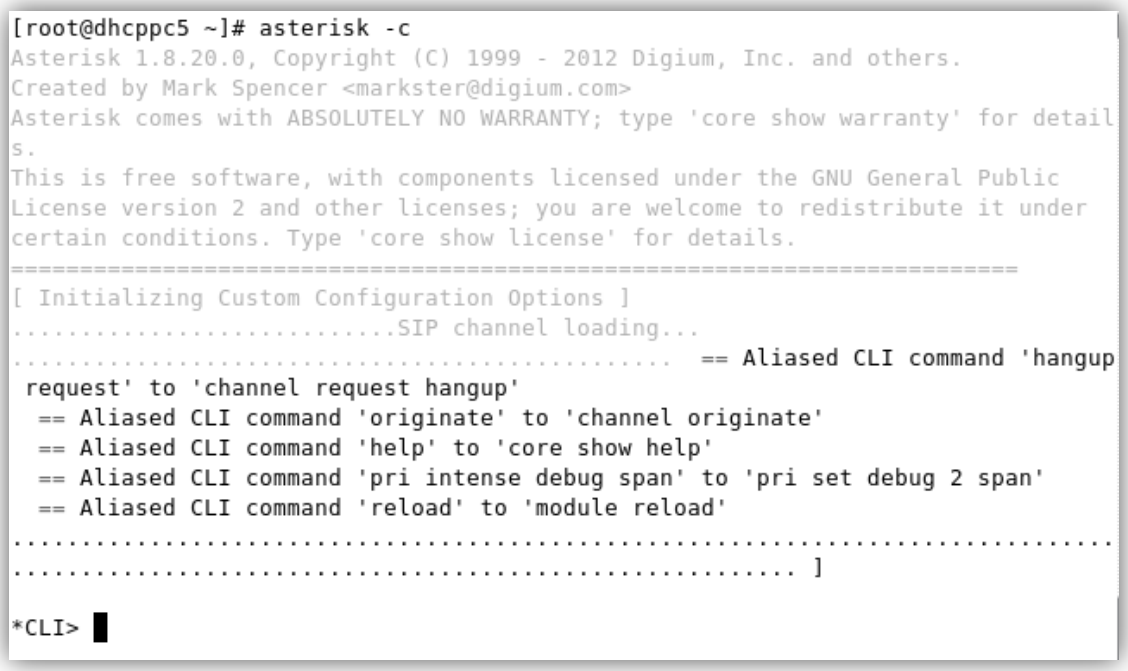


Imagen 13. Inicia el servicio de Asterisk e ingresa a la consola CLI

Imagen 14. Ejecuta un comando sin la necesidad de ingresar a la consola CLI de Asterisk

Dentro de la consola CLI de Asterisk pueden ser usados varios comandos, entre los más populares se tienen los siguientes:

Mostrar información sobre las aplicaciones:

CLI> core show applications CLI> core show applications <nombre de la aplicación>

Mostrar información sobre las funciones:

CLI> core show functions

```
CLI> core show function <nombre de la función>
```

Cargar, descargar y mostrar los módulos de Asterisk:

```
CLI> module show
CLI> module load <nombre del módulo>
CLI> module unload <nombre del módulo>
```

Mostrar los usuarios SIP/IAX

```
CLI> sip show peers

CLI> iax2 show peers
```

Mostrar los canales activos:

```
CLI> core show channels

CLI> sip show channels

CLI> iax2 show channels

CLI> zap show channels
```

Parar y reiniciar Asterisk:

```
CLI>  stop now

CLI>  stop when convenient

CLI> stop gracefully

CLI> restart now

CLI> restart when convenient
```

```
CLI> restart gracefully
```

5.3 Configuración y creación del plan de marcado

Si se ha realizado la instalación de AsteriskNow los archivos de configuración del plan de marcado se encontrarán en la siguiente ruta:

```
/etc/asterisk/extensions_custom.conf
```

Para configurarlo se abre con cualquier editor de texto.

Imagen 15. Archivo extensions_custom.conf sin configurar

Este archivo contiene por lo pronto una demo que es utilizada para verificar que la conexión del plan de marcado se está estableciendo. Se la puede acceder marcando, desde un teléfono o softphone configurado, el número de extensión “1234”.

Se puede observar que el nombre del contexto es [from-internal-custom], por lo general cuando se crea una cuenta sip o iax este contexto ya viene por defecto.

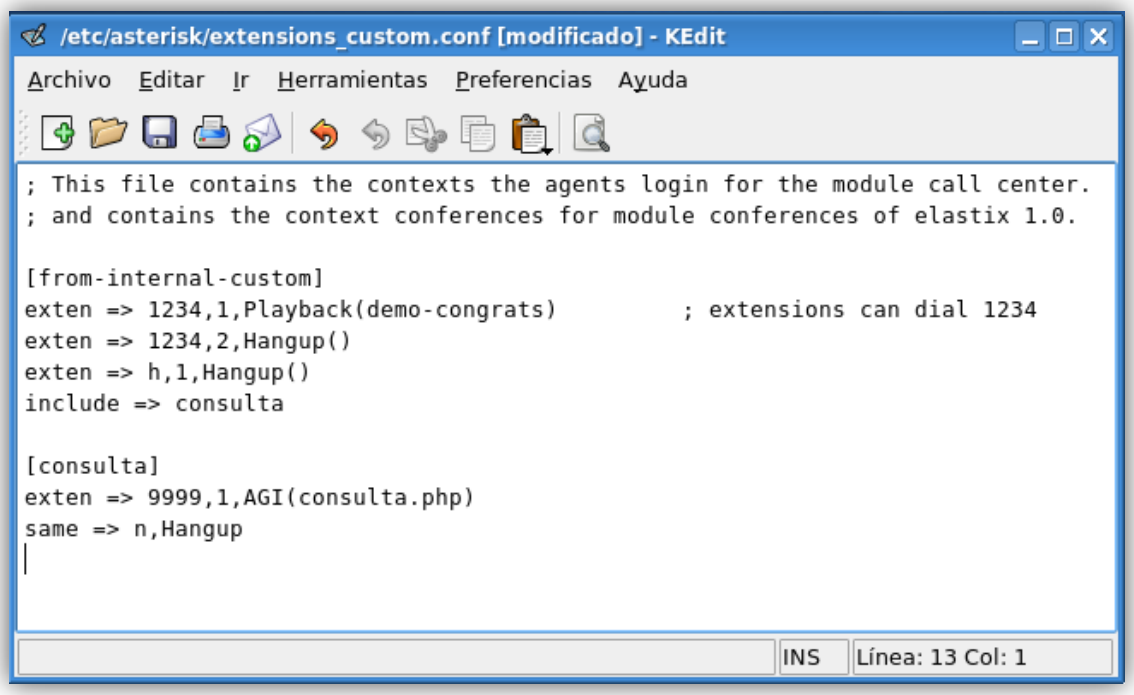


Imagen 16. Archivo extensions_custom.conf configurado

El siguiente paso es crear un contexto, en este caso se lo llamará [consulta], y debajo de él se colocará el número de extensión que debe ser marcada para acceder al script de sistemas de consultas.

```
exten => 9999,1,AGI(consulta.php)

Same => n,Hangup
```

Esta configuración indica que en el momento que se marque la extensión “9999” el plan de marcado ejecutará el script llamado “consulta.php”, el cual contiene el sistema de consultas a una base de datos de cartera.

Una vez que se termine de ejecutar el script, pasará a la siguiente prioridad, la cual es “hangup”, haciendo que la llamada llegue a su fin.

5.4 Diseño de la base de datos

La base de datos para este sistema fue creada en Mysql, y consta de 3 tablas las cuales serán usadas para las consultas, y para mostrar los resultados del sistema de cartera.

A continuación, el diagrama entidad-relación que será usado.

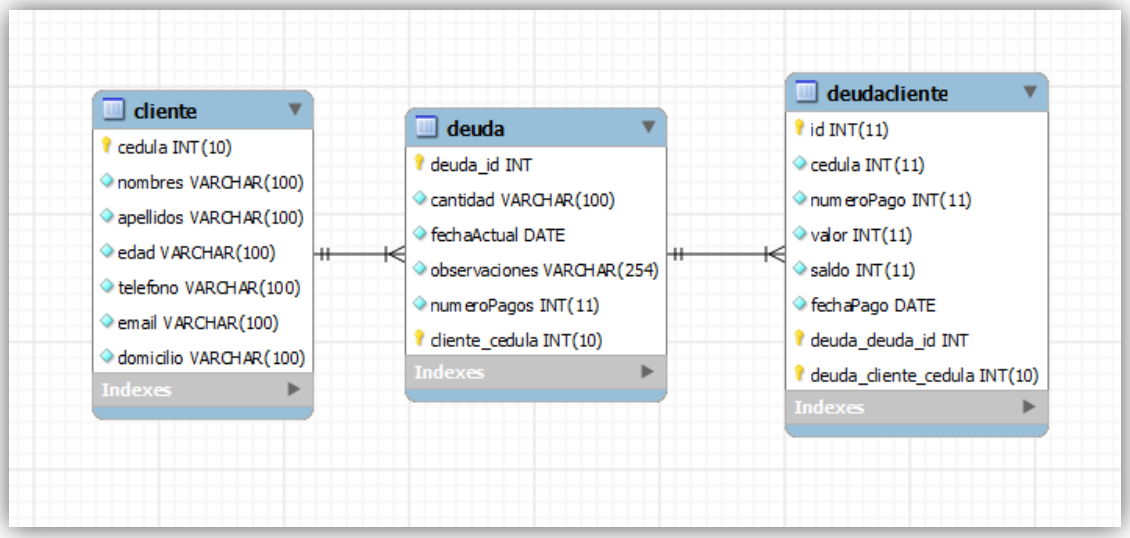


Imagen 17. Diagrama entidad-relación base de datos

La creación del diagrama Entidad-Relación fue desarrollada en la herramienta *MySQL Workbench*, y mediante el proceso de "*forward engineer*" se obtuvo un archivo de texto con extensión ".sql", el cual contiene todas las tablas para la creación de la base de datos.

Una vez terminado el paso anterior, se debe ingresar a la base de datos *MySQL* desde la consola de Centos e importar la base de datos con la siguiente instrucción.

```
mysql -u root -p < sistema.sql
```

En el momento que es ejecutado este comando, el sistema pedirá el ingreso de la contraseña de MySQL que por lo general es configurada cuando es realizada la instalación del sistema Asterisk.

El paso siguiente es otorgar todos los privilegios a el usuario "root", para que no exista ningún tipo de problema a futuro, ya que con esto el usuario es libre de acceder con total libertad a las tablas creadas en MySQL.

```
GRANT ALL PRIVILEGES ON *.* TO root@"localhost" IDENTIFIED BY
"ContraseñaDelUsuarioSql";
```

```
mysql> show full tables from sistema;
+-------------------+------------+
| Tables_in_sistema | Table_type |
+-------------------+------------+
| cliente           | BASE TABLE |
| deuda             | BASE TABLE |
| deudacliente      | BASE TABLE |
+-------------------+------------+
3 rows in set (0.02 sec)
```

Y por último se comprueba que se importó correctamente la base de datos a *MySQL*.

5.5 Instalación y configuración de Festival (Text to Speech)

Festival es una herramienta que permite traducir de texto a voz en varias aplicaciones de Linux, su principal característica es que dispone de varios idiomas y las voces en los 2 géneros (masculino y femenino).

Festival en Asterisk ya viene instalado por defecto, pero no dispone de las voces en español, para configurarlo en español es necesario descargar algunos archivos y configurarlos. A continuación, se detallan los pasos.

1. El primer paso es la descarga del paquete de voces en español, en este caso será configurada la voz femenina.

   Las siguientes instrucciones deberán ser ingresadas a través de la terminal de Centos.

```
wget http://kaplah.org/system/files/field/files/festival-es.tar.gz
```

2. Después de haber descargado, se tiene que extraer en la carpeta donde se tiene instalado festival.

```
tar xzvf festival-es.tar.gz --directory=/usr/share/festival/voices/
```

Se tiene que configurar que Festival use ahora la voz descargada. Para esto se debe descargar un parche de la siguiente forma.

```
wget http://kaplah.org/system/files/field/files/siteinit.scm_.patch
```

Y aplicarlo.

```
patch /usr/share/festival/siteinit.scm siteinit.scm_.patch
```

De esta forma ya se tiene configurada como voz principal la que fue descargada anteriormente.

La librería PHPAgi viene por defecto configurado con un path que apunta incorrectamente hacia la función de Festival llamada *text2wav* la que se encarga de la transformación de texto a voz.

Para la solución de este problema también será utilizado un parche que se encarga de apuntar correctamente a la función antes mencionada.

Se descarga el parche.

```
wget http://kaplah.org/system/files/field/files/phpagi.conf_.patch
```

Y se aplica.

```
patch /etc/asterisk/phpagi.conf phpagi.conf_.patch
```

De esta manera Festival queda totalmente funcional y con voces en español.

5.6 Creación del Script.

El script es una de las partes más importantes del sistema de cartera, ya que desde aquí son realizadas todas las consultas hacia la base de datos.

Existen 2 formas para la elaboración de un I.V.R. (Interactive Voice Response), la una es desde el plan de marcado, y la otra es desde el script, todo dependerá de las habilidades del programador.

El I.V.R. de este trabajo de graduación será elaborado en su totalidad en el script mediante el uso de una librería PHP llamada PHPAgi. Esta clase es contenedora de todos los comandos que pueden ser usados por Asterisk para la interacción con el usuario. Cabe recalcar que esta librería es de libre distribución y puede ser descargada desde su página web.

http://phpagi.sourceforge.net/

Una buena práctica para los desarrolladores no tan experimentados es descargar los ejemplos también disponibles en la página web.

Una vez descargada la librería se debe descomprimir, y ser ubicada en una ruta específica para que pueda ser utilizada por el script desde el plan de marcado. Esta ruta es la siguiente.

```
/var/lib/asterisk/agi-bin/
```

El script desarrollado en código PHP se ubica en la misma ruta donde se encuentra la librería de PHPAgi. (Ver código desarrollado en el anexo 1)

5.7 Creación de una extensión SIP.

Asterisk incorpora una PBX con la cual se puede crear extensiones SIP mediante una interfaz gráfica. A continuación, se indica la forma crear una extensión que se pueda conectar con el plan de marcado.

El primer paso es conocer la dirección IP del servidor, para lo que se deberá ingresar en una terminal de Centos, e ingresar el siguiente comando:

```
ifconfig
```

En la siguiente imagen se puede observar la dirección IP resaltada de color negro.

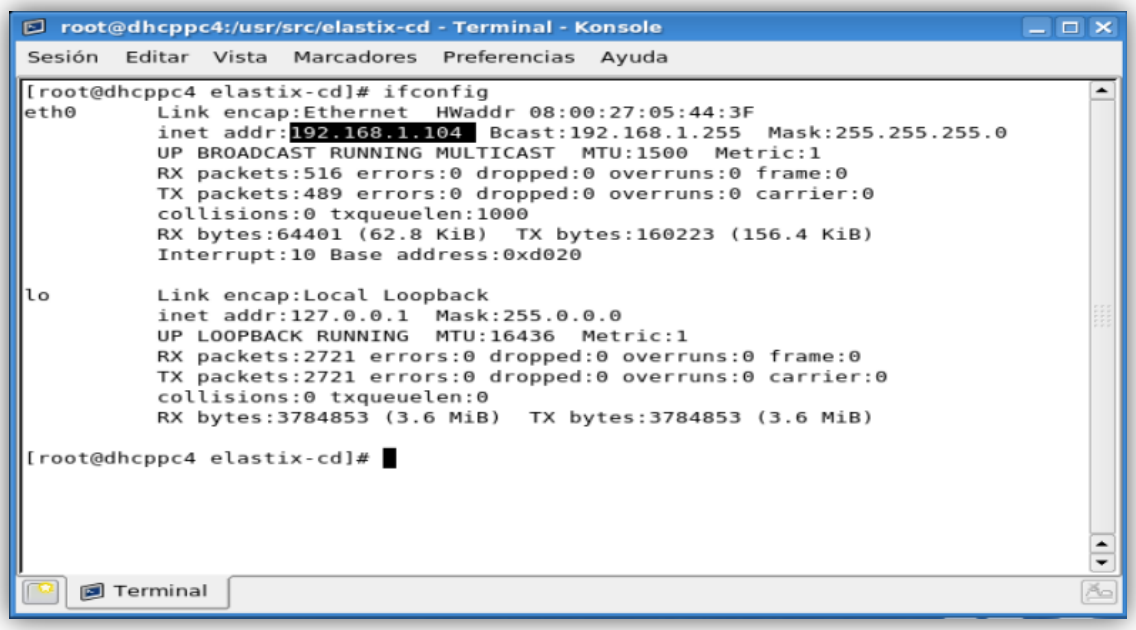


Imagen 18. Consola de Centos

A continuación, se ingresa en el navegador la dirección IP del servidor, mostrándonos una página en la cual se debe ingresar el usuario y contraseña que fueron configurados al momento de la instalación de Asterisk.

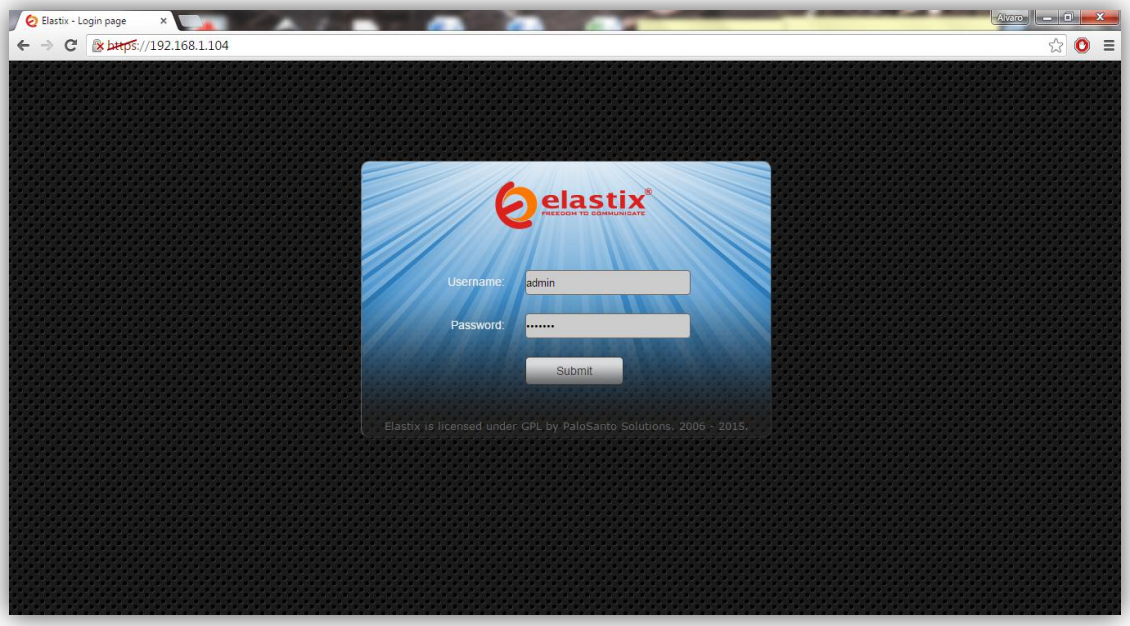


Imagen 19. Página de autentificación para ingresar al P.B.X.

Una vez autentificado, se indica al usuario la pantalla principal de la P.B.X., en la que se puede observar un *dashboard* (tablero) indicando las siguientes opciones: los recursos del sistema, el estado de los procesos de Asterisk, el uso del disco duro, y un gráfico del rendimiento en general.

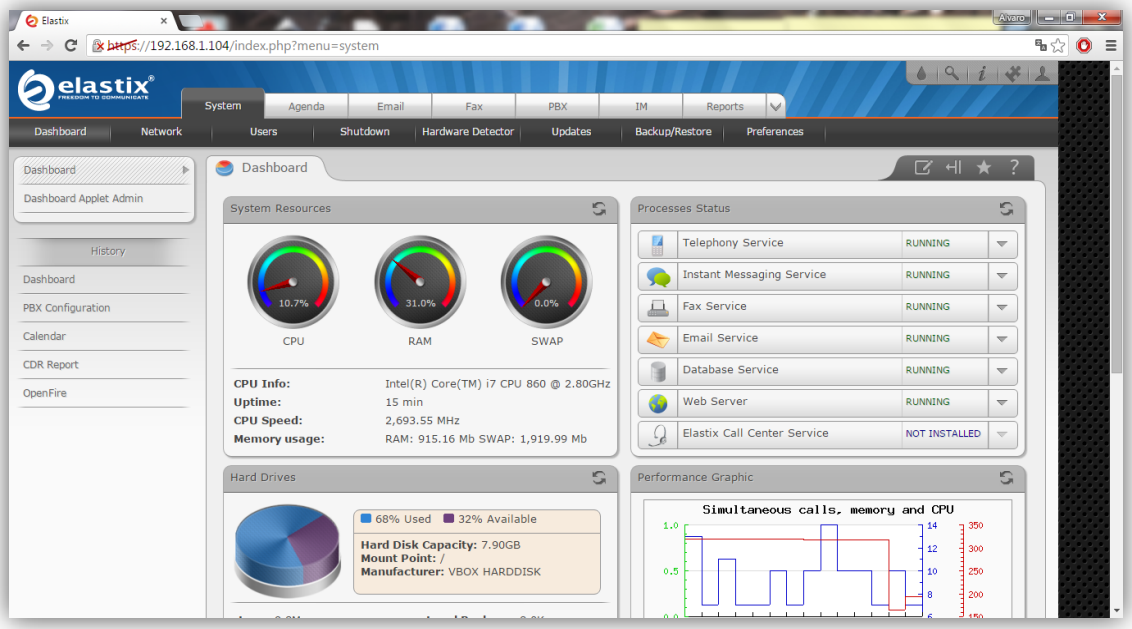


Imagen 20. Dashboard de la P.B.X.

Para acceder al área de configuración de extensiones, el usuario debe dar clic en el menú superior con título “P.B.X.”, seguido de la opción *Extensions* ubicada en la parte izquierda de la pantalla.

En el menú desplegable se elige la opción *Generic SIP Device* y damos clic en el botón *Submit*

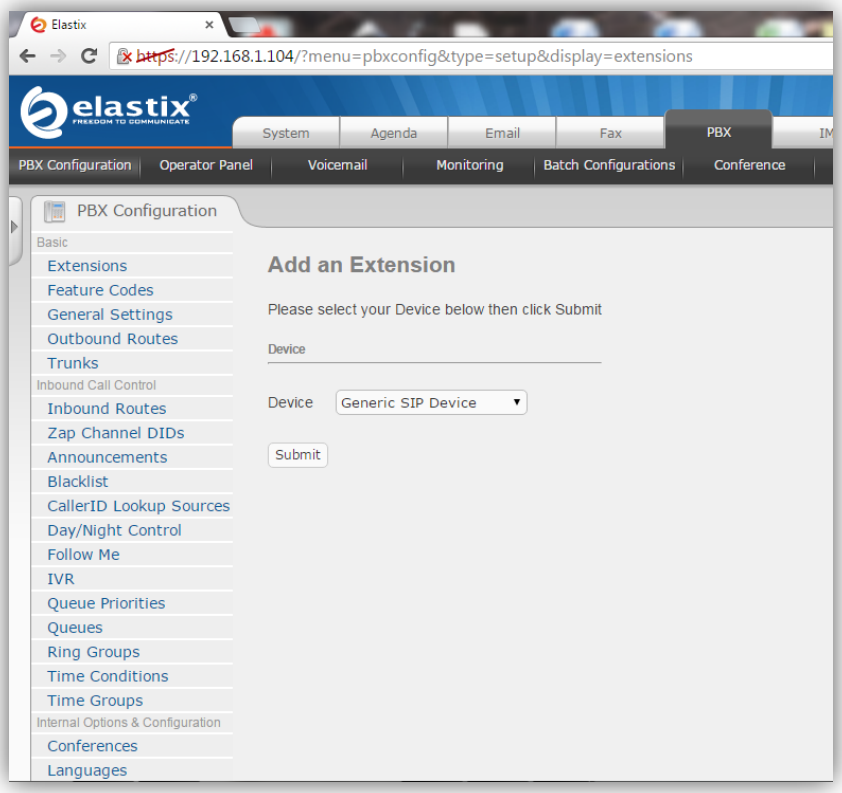


Imagen 21. Pantalla principal para la configuración de las extensiones

Para la creación de una extensión SIP básica, se tomarán en cuenta solo los siguientes parámetros:

- Tipo de extensión
- Nombre de extensión
- Numero de extensión
- Nombre de contexto
- Contraseña de extensión

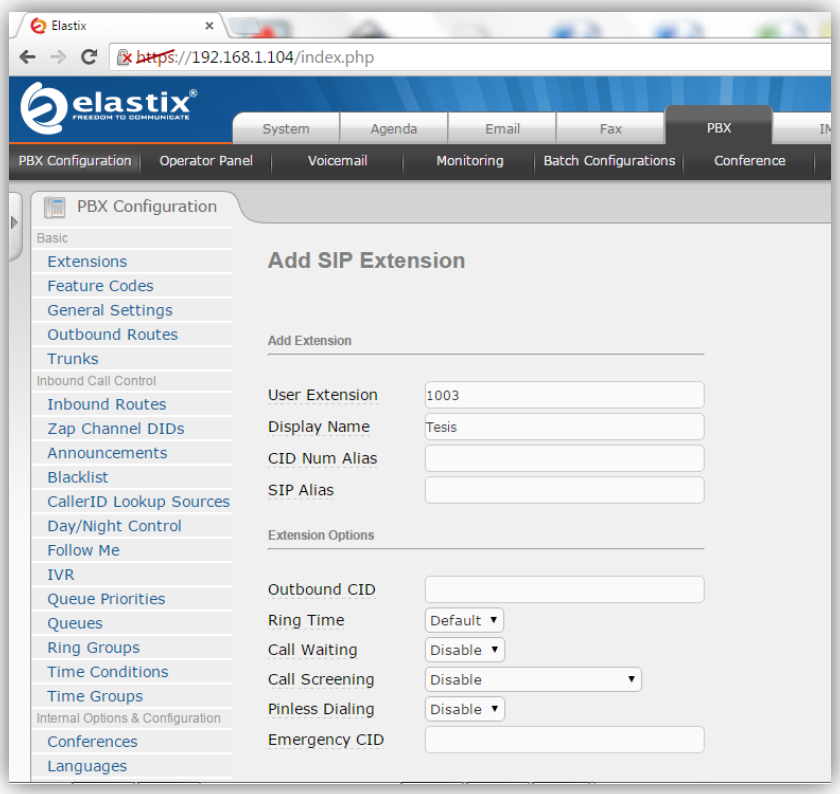


Imagen 22. Nombre y número de la extensión

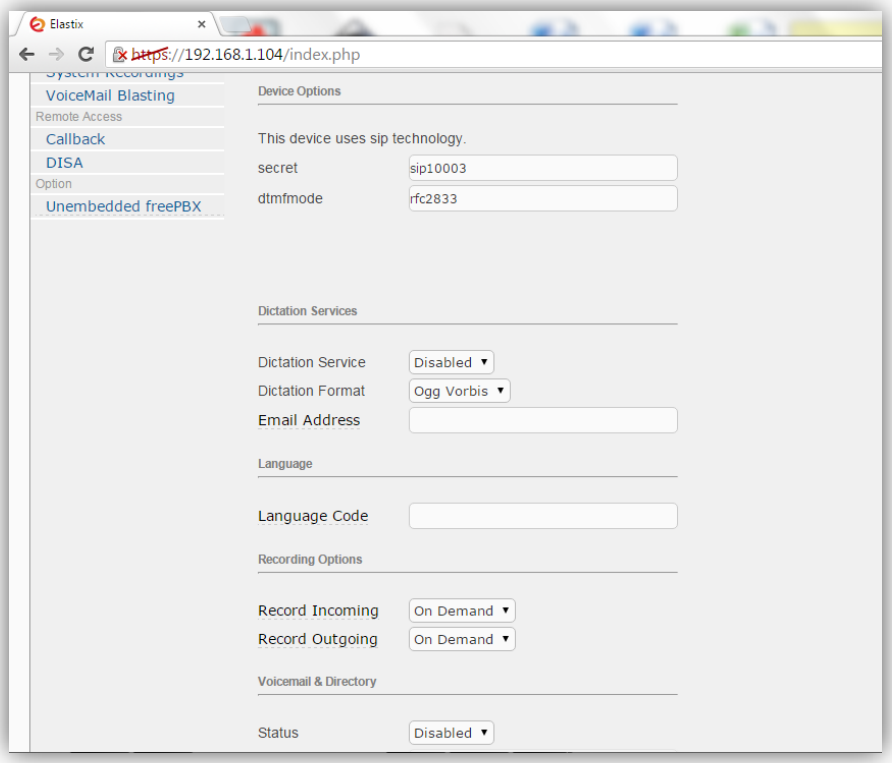


Imagen 23. Contraseña de la extensión

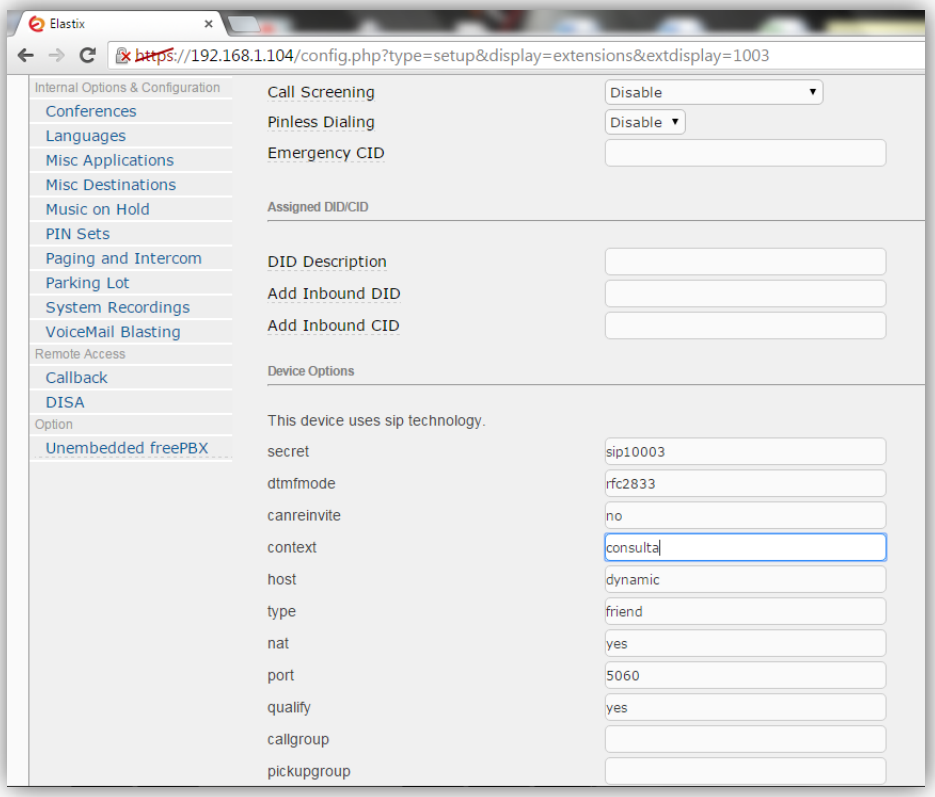


Imagen 24. Nombre del contexto

Cabe recalcar que el nombre del contexto ingresado para esta extensión, debe coincidir exactamente con el nombre del contexto creado anteriormente en el plan de marcado.

El momento en que todos los parámetros necesarios hayan sido ingresados, se procede a dar clic en el botón “Submit” para crear la extensión.

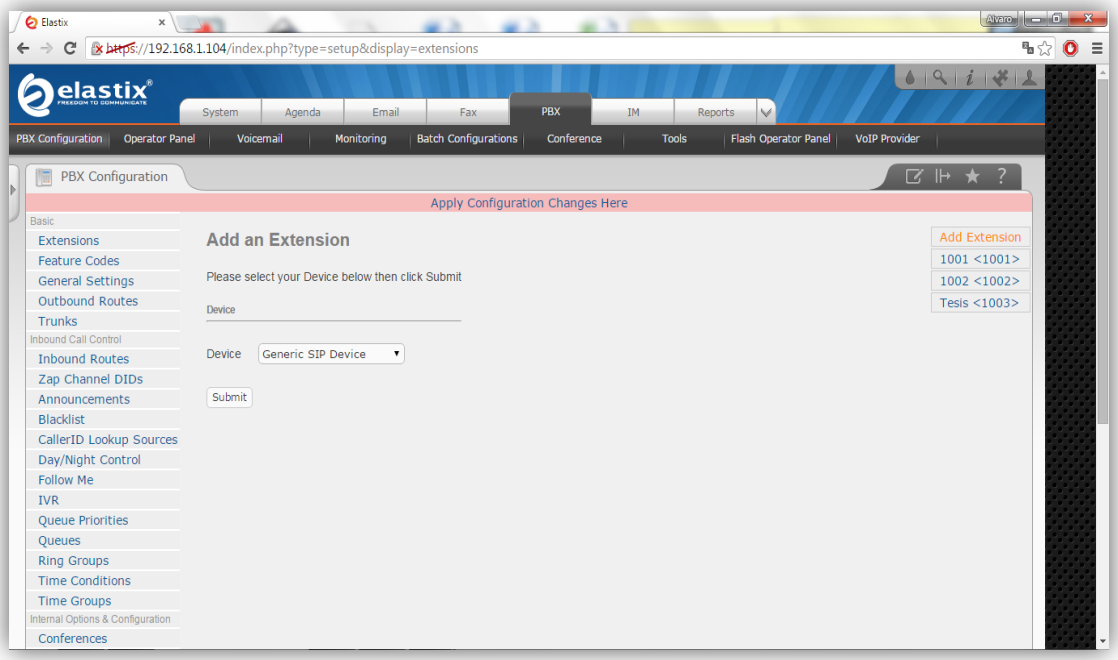


Imagen 25. Pantalla de creación con éxito de una extensión SIP

Como paso final se da clic en el link que dice *Apply Configuration Changes Here*, y con eso la extensión queda lista para su uso.

5.8 Configuración de un *softphone*.

El *softphone* es una aplicación de software que puede ser ejecutada tanto en un ordenador portátil o como de escritorio. El audio debe pasar a través del sistema de sonido de la PC, por lo que se necesita un auricular que funcione con aplicaciones de telefonía. Recientemente este software ha sido escrito para usarlo en teléfonos inteligentes, los cuales permiten conectarse a otras redes que no sean necesariamente para la red celular. Las interfaces de los muchos programas que existen son a menudo elaboradas para que se parezcan a un teléfono físico.

5.8.1 Protocolos de VoIp

Los protocolos más conocidos y usados son: SIP e IAX2.

El protocolo SIP cuyo significado es “Protocolo de inicio de sesiones”, permite establecer sesiones de tipo cliente servidor para la transmisión de contenido multimedia. Este protocolo usa un puerto para la señalización (puerto 5060), y 2 puertos RTP (*Real-time transport Protocol*) para enviar el audio.

El protocolo IAX2 cuyo significado es “Protocolo de conexiones entre servidores Asterisks”, es de libre distribución, es decir que puede ser modificado a conveniencia del usuario. Este protocolo usa solo un puerto, tanto para la señalización y el envió de audio.

Para las pruebas que se realizan más adelante se utilizara el protocolo SIP, ya que es compatible con la mayoría de *softphones* del mercado.

5.8.2 Zoiper

El *softphone* que se eligió para hacer las pruebas es Zoiper, ya que este es libre distribución, y disponible para: Windows, Os x, Linux, iOS y Android; además soporta los protocolos SIP, lo que le hace 100% compatible con Asterisk.

Como primer paso se procederá a ingresar a la página web del software para descargarlo.

En un navegador se ingresa al siguiente link: http://www.zoiper.com/en/voip-softphone/download/zoiper3

Se elige la plataforma y se procede con la descarga.

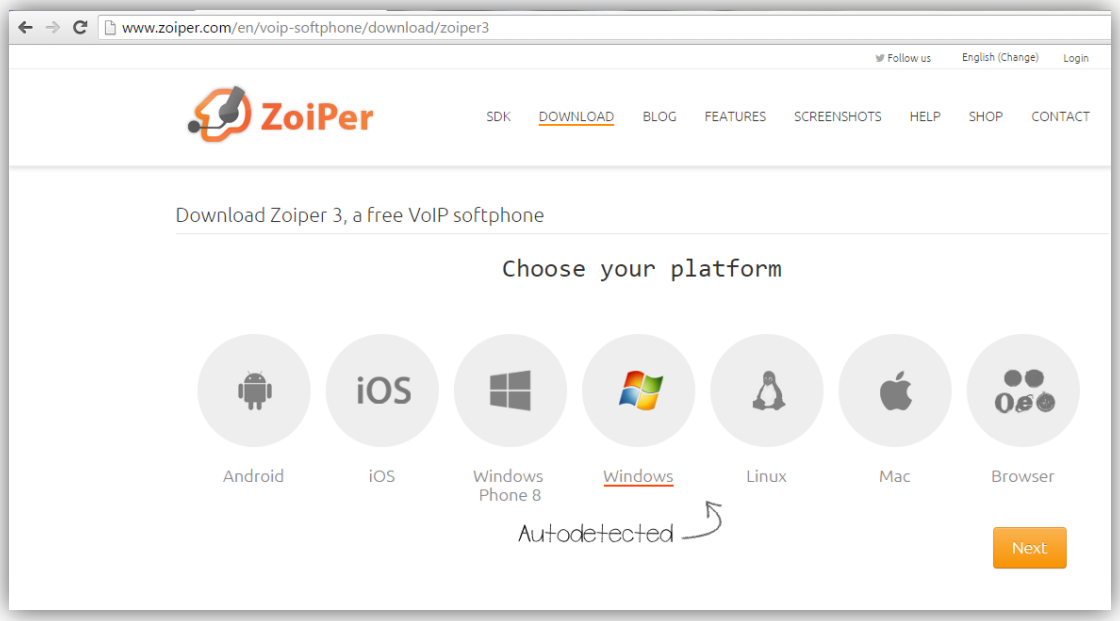


Imagen 26. Página web del softphone Zoiper

Terminada la descarga se procederá con la instalación del software, siguiendo los siguientes pasos.

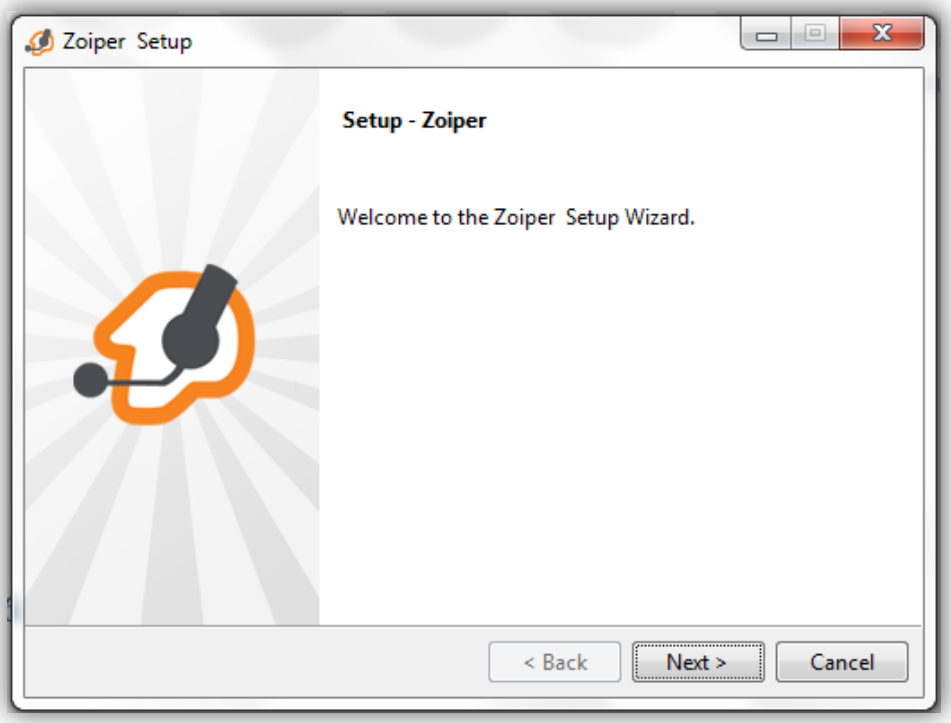


Imagen 27. Instalación de Zoiper - Paso 1

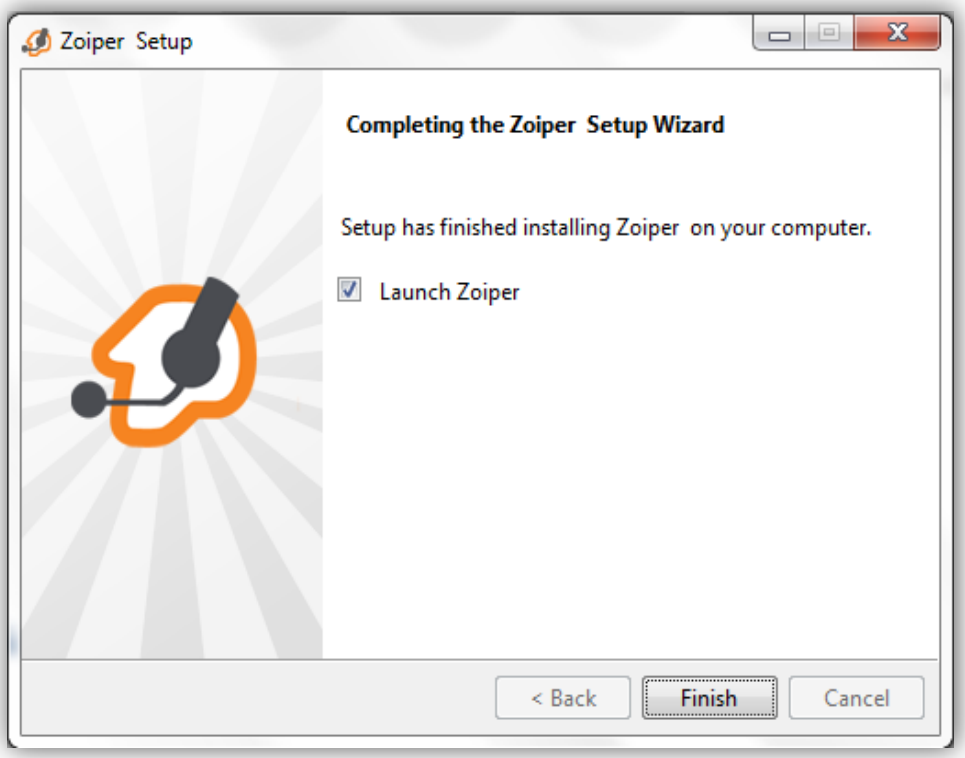


Imagen 28. Instalación de Zoiper - Paso

Como paso final se da clic en el botón “Finish”, y si se tiene activada la opción “Launch Zoiper”, se iniciará el programa automáticamente.

Imagen 29. Pantalla de inicio del software Zoiper

Para agregar una cuenta hay que ingresar al menú: Configuración -> Preferencias

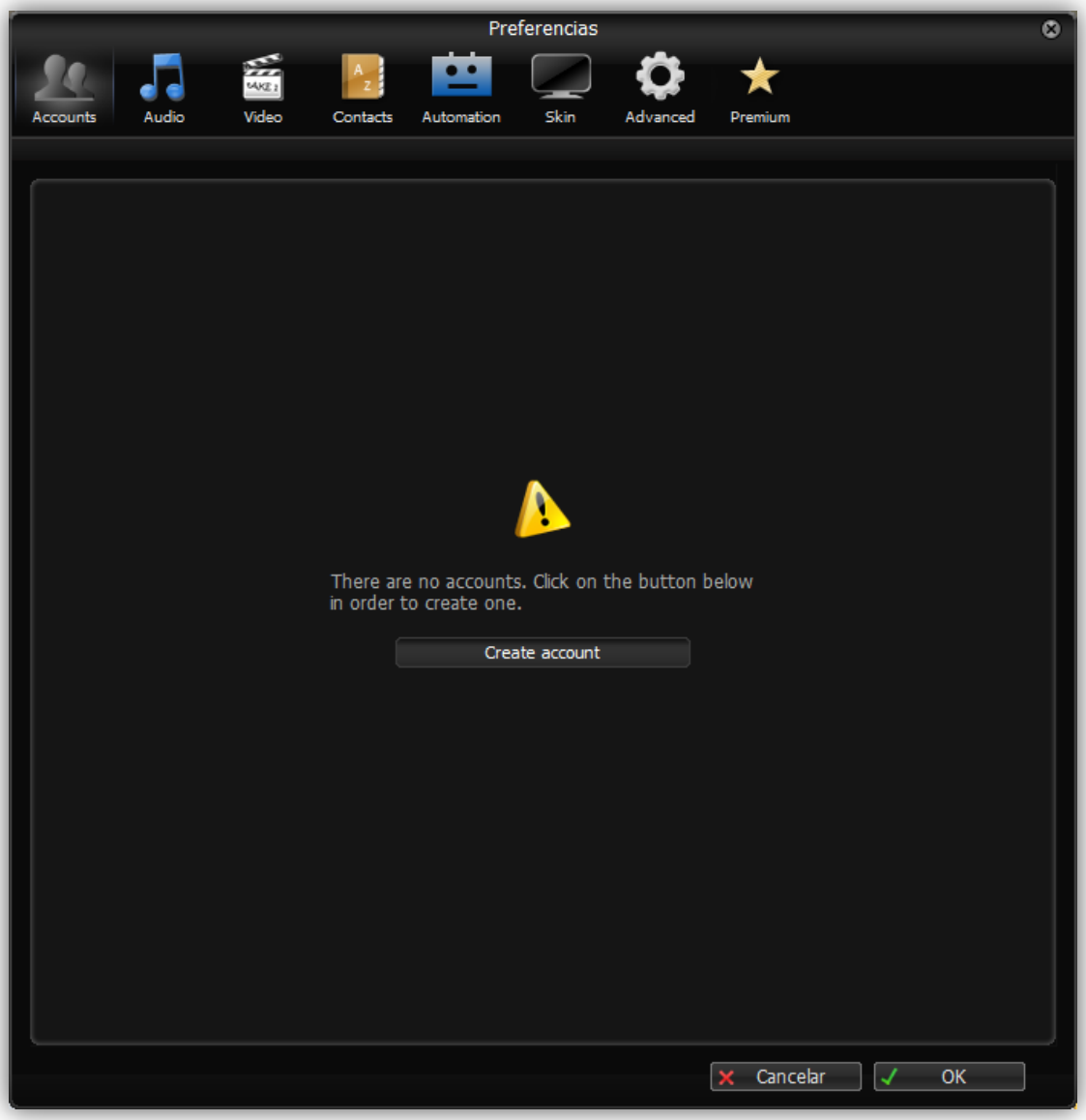


Imagen 30. Ventana para la creación de la cuenta SIP en Zoiper

Se da clic en el botón que dice "*Create account*", y se elige el protocolo SIP.

Aparecerá una ventana para el asistente de cuentas, en la cual se debe ingresar lo siguientes parámetros:

- Nombre del usuario
- La contraseña
- El dominio (IP del servidor Asterisk. Ver imagen 19)

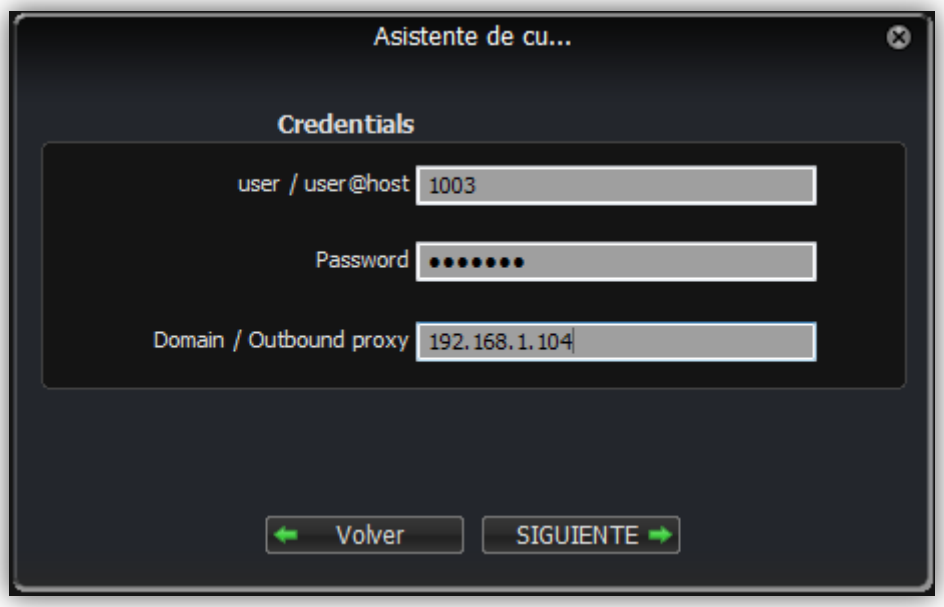


Imagen 31. Ventana del asistente de cuentas de Zoiper

Se debe tomar en cuenta que la información suministrada proviene la cuenta SIP creada anteriormente en el PBX de Asterisk. Si la información requerida es ingresada correctamente, entonces aparecerá un mensaje informando al usuario que la cuenta esta lista para usar.

5.9 Conclusiones.

En este capítulo se observa como en Asterisk, todo proceso es fácil de configurar, desde la creación del plan de marcado, hasta la creación de una extensión para realizar las pruebas, de tal forma que hasta el usuario menos experimentado pueda realizar estas configuraciones sin problemas.

6. Capítulo 6: Funcionamiento y Pruebas

Una vez realizado todos los pasos de los capítulos anteriores, solo queda por realizar las pruebas en el sistema. Lo único necesario es activar ciertos servicios, y conocer de forma exacta como debería comportase el sistema por lo que se han especificado los casos de usos que veremos más adelante.

6.1 Activación de servicios

Para poder hacer consultas a una base de datos Myql desde Asterisk, se debe iniciar los servicios de Asterisk y Mysql. Por lo general el servicio de Mysql se inicia junto con Asterisk, pero se realizaran los pasos en caso de que no este.

Desde una consola de CentOS se ingresa el siguiente comando para iniciar el servicio Mysql.

```
service mysqld start –u root –p
```

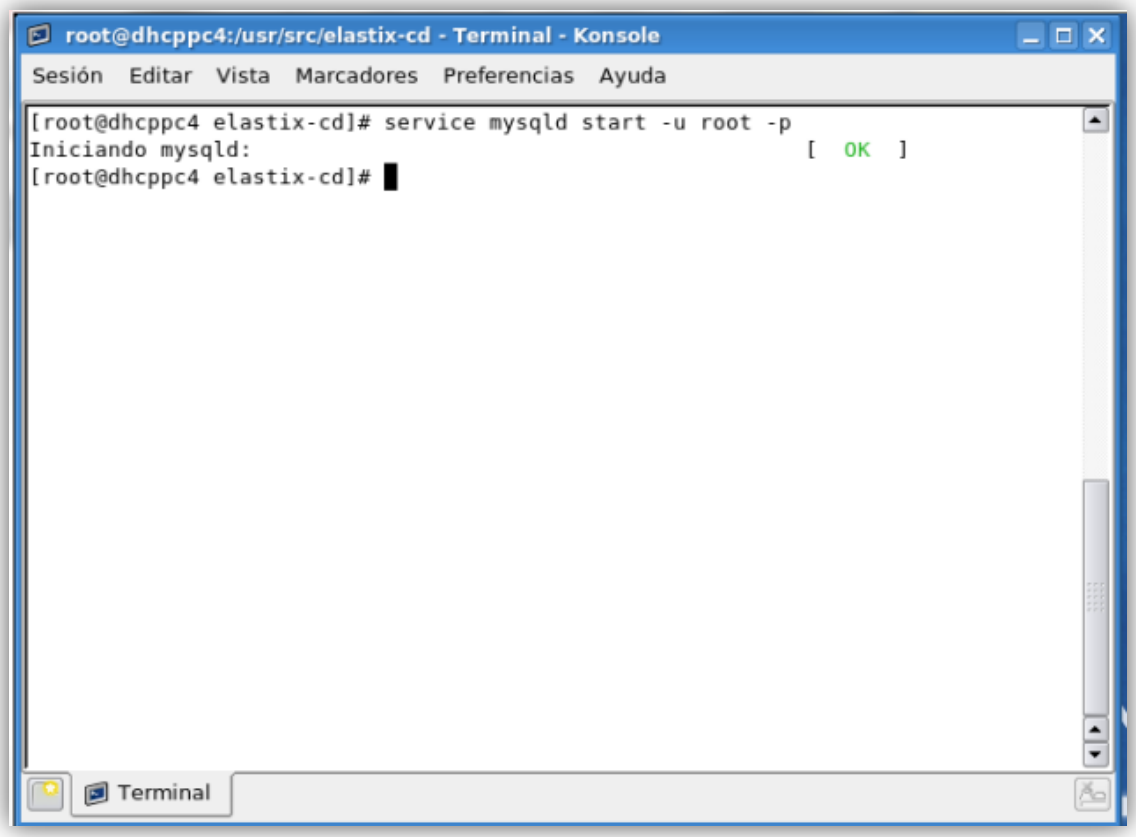


Imagen 32. Iniciar servicio de Myql

Igualmente, desde una consola se debe escribir el siguiente comando para iniciar el servicio de Asterisk.

```
service asterisk start
```

Si aparece un mensaje informando que el servicio ya se encuentra iniciado, se ejecuta el siguiente comando para ingresar a la consola de Asterisk y así comprobar que ya se puede comenzar a utilizar el software.

```
asterisk -r
```

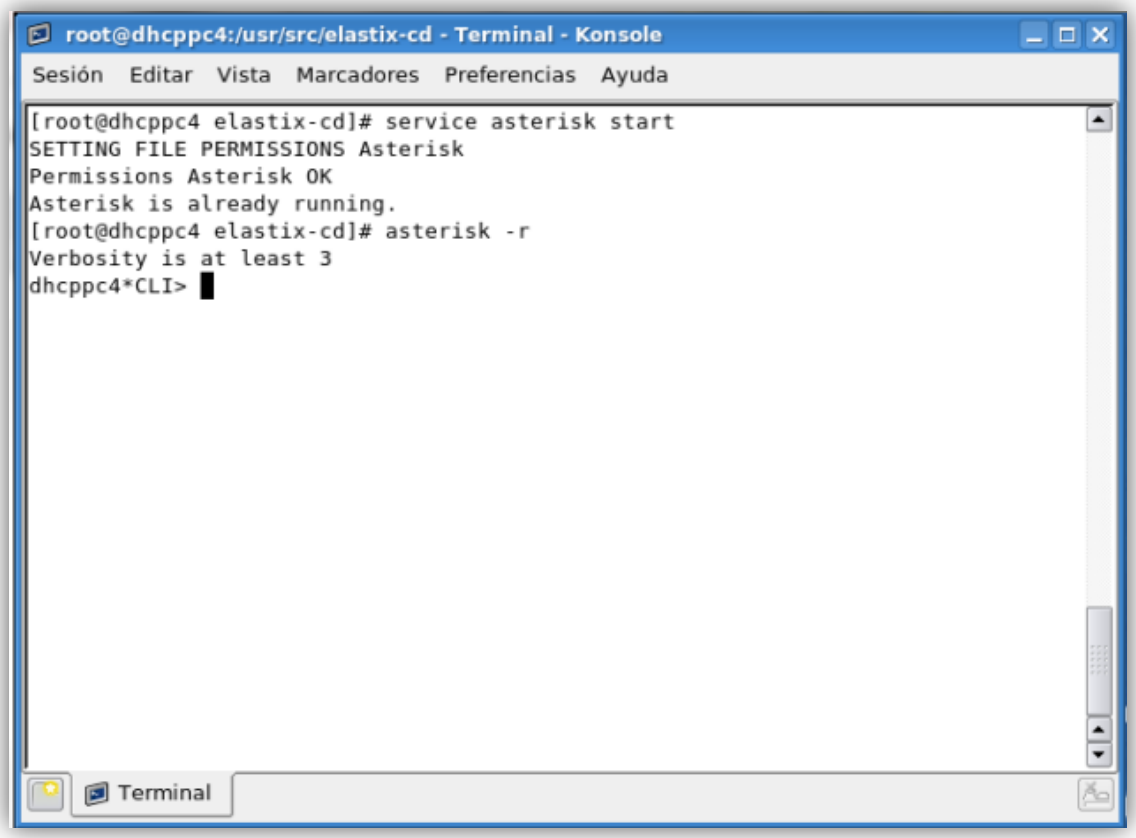


Imagen 33. Iniciar servicio de Asterisk

6.2 Ingreso de información en la base de datos

Antes de comenzar con las pruebas, se debe ingresar información a la base de datos de cartera, para poder realizar las consultas desde el *softphone*.

Para completar estos pasos de forma gráfica, se creó un prototipo, desarrollado en código PHP, para el ingreso de los datos personales del cliente, con sus respectivas deudas.

Se realizan los siguientes pasos para ingresar un cliente con sus respectivas deudas.

1. Se ingresará al cliente en la base de datos.

Imagen 34. Ingreso de un usuario a la base de datos de pruebas

2. Se le ingresará al cliente una deuda, de la que se dividirá en 4 pagos.

Imagen 35. Ingreso de una deuda a un usuario en la base de datos de pruebas

Los pagos de las cuotas quedaran de la siguiente manera:

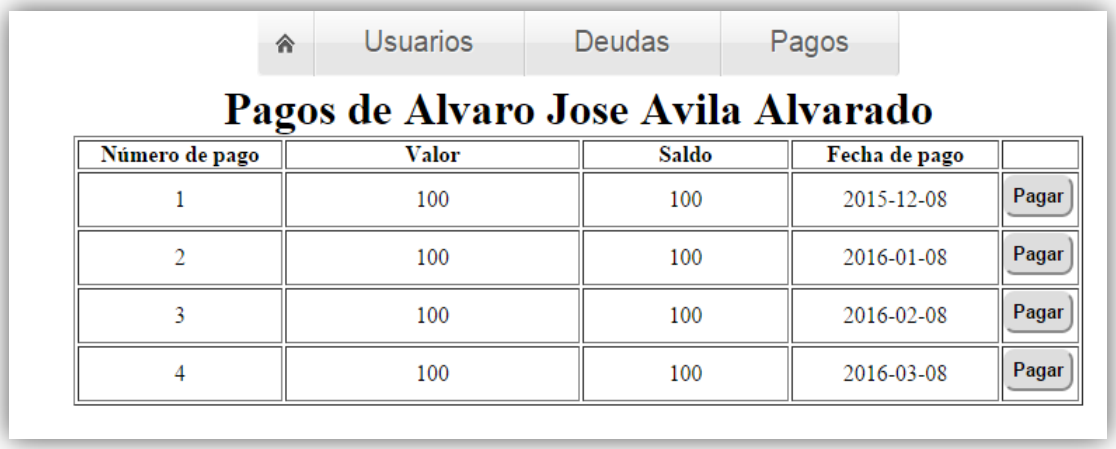


Imagen 36. Cuadro de pagos del usuario

6.3 Casos de uso del sistema

Se utilizará la consola de Asterisk para observar cada caso de uso que ocurre durante una llamada al sistema de consultas.

En el momento que se realice la llamada a la extensión configurada (extensión 9999), el sistema puede realizar diferentes acciones, las cuales son:

1. El usuario no ingresa ningún número de su cédula. - El sistema solicitará como primer paso que el usuario ingrese su número de cédula seguido del signo numeral, si el usuario no ingresa nada, el sistema le avisará, reproduciendo un mensaje que indica que no ha ingresado ningún número y que lo intente nuevamente, o que presione el número cero para salir.

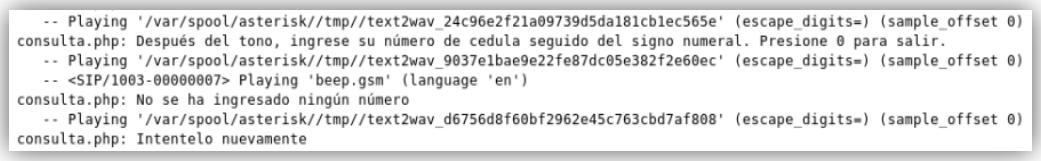


Imagen 37. Caso de uso número 1

2. El usuario ingresa un número de cédula que no está registrado en la base de datos. - El sistema realizara una consulta en la base de datos, y al no encontrar ningún resultado, reproducirá un mensaje con el número de cédula ingresado, y solicitará al usuario que ingrese nuevamente su número de cédula.

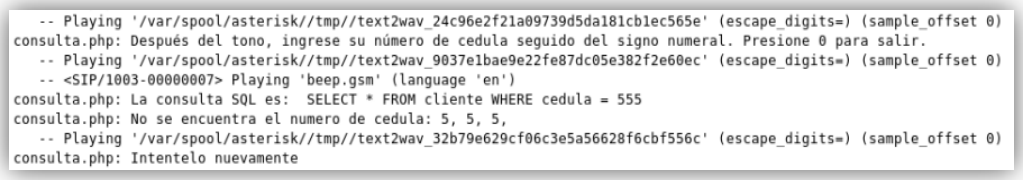


Imagen 38. Caso de uso número 2

3. El usuario ingresa un número de cédula que si se encuentra en la base de datos. - El sistema verificará en la base de datos si el usuario tiene un código de seguridad. Si es que no tiene uno le pedirá que ingrese, caso contrario el sistema le pedirá que ingrese su código de seguridad ya creado.

   En la primera imagen se muestra cuando no tiene un código de seguridad, en una segunda imagen se muestra cuando tiene un código de seguridad, y

en la tercera imagen muestra cuando el usuario si tiene un código de seguridad, pero es ingresado un código incorrecto.

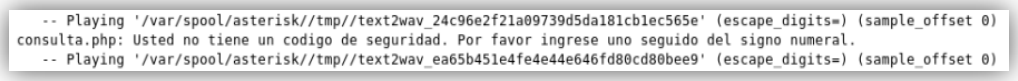


Imagen 39. Caso de uso número 3 (opción 1)

Imagen 40. Caso de uso número 3 (opción 2)

Imagen 41. Caso de uso número 3 (opción 3)

En cualquiera de las 2 opciones, si el usuario no ingresa ningún número, el sistema reproducirá un mensaje pidiendo que lo intente nuevamente.

Imagen 49. Caso de uso número 3 (opción 4)

4. El usuario ingresa su código de seguridad por primera vez. - Cuando esto ocurre, se grabará el código de seguridad en la base de datos, y el sistema le pedirá al usuario que nuevamente ingrese su número de cédula, seguido de su código de seguridad.

5. El usuario ingresa con su código de seguridad creado anteriormente. - Si el usuario se identificó con su número de cédula y su código de seguridad válido, el sistema le dará la bienvenida.

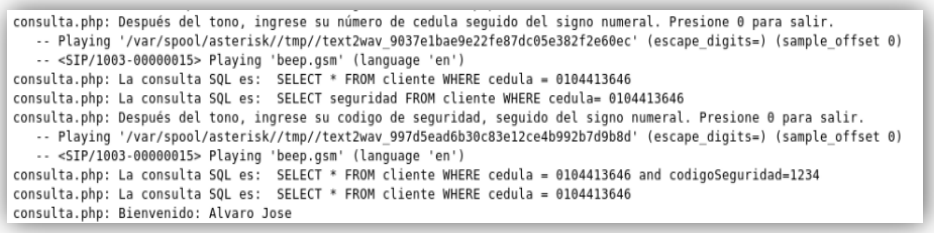


Imagen 50. Caso de uso número 5 (Opción 1)

Al usuario se le presentara un menú con 3 opciones:

- Para escuchar su deuda presione 1 seguido del signo numeral.
- Para escuchar su saldo vencido presione 2 seguido del signo numeral.
- Para escuchar su saldo por vencer presione 3 seguido del signo numeral.
- Presione 0 para salir.

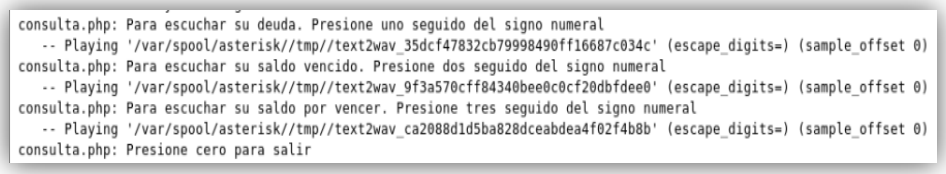


Imagen 51. Caso de uso número 5 (Opción 2)

6. El usuario presiona la opción 1 del menú principal. - Con esta opción el usuario podrá escuchar el valor de su deuda total, para salir de esta opción hacia el menú principal, el usuario tiene que presionar el número cero.

Imagen 42. Caso de uso número 6

7. El usuario presiona la opción 2 del menú principal. - Con esta opción el usuario podrá escuchar el valor de su saldo vencido, para salir de esta opción hacia el menú principal, el usuario tiene que presionar el número cero.

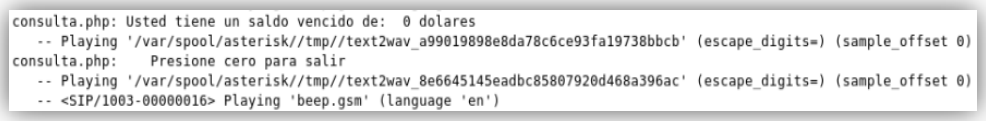


Imagen 43. Caso de uso número 6

8. El usuario presiona la opción 3 del menú principal. - Con esta opción el usuario podrá escuchar el valor de su saldo por vencer y la fecha máxima de pago de toda la deuda, para salir de esta opción hacia el menú principal, el usuario tiene que presionar el número cero.

Imagen 44. Caso de uso número 8

9. El usuario presiona el número cero en el menú principal. - Al presionar cero, el usuario indica al sistema que realizó sus consultas y está listo para salir, finalizando así la llamada.

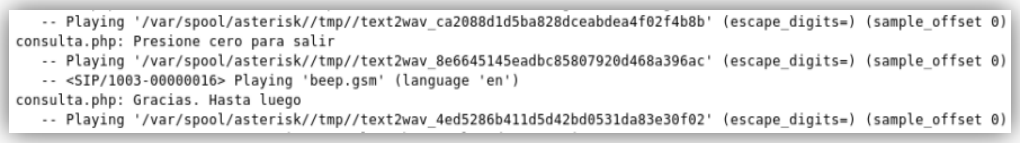


Imagen 45. Caso de uso número 9

6.4 Conclusiones

Antes de comenzar a desarrollar el script, es recomendable establecer todos los casos de uso del sistema, de forma que al desarrollador le sirva como guía, y le ayude a verificar si las especificaciones que se plantearon se cumplan al momento de testear el sistema.

Conclusiones y recomendaciones

Conclusiones

- Asterisk al ser un software de libre distribución, está al alcance de cualquier usuario común que quiera investigar el mundo de la telefonía IP. Asterisk presenta una gran cantidad de servicios que son totalmente configurables debido a que ofrece una interfaz muy amigable con el usuario.
- Debido a la escalabilidad que ofrece Asterisk, no hubo problema al momento de integrar la base de datos con el sistema I.V.R., permitiendo al usuario usar el lenguaje de programación con el que este más familiarizado.
- El uso de la clase phpAGI es de gran ayuda al momento de integrar una base de datos con el plan de marcado, ya que solo se necesita de una línea de código en el plan de marcado para realizar este procedimiento.
- El conversor de texto a voz llamado Festival, es una de las mejores opciones al momento de elegir uno, ya que es de libre distribución y existe la posibilidad de descargar muchos paquetes de voces en cualquier idioma sin costo alguno.
- Esta solución, que fue implementada para la consulta a una base de datos mediante tonos de marcado, puede ser adaptada a cualquier necesidad que tenga el cliente, y en cualquier empresa, dándole un valor agregado, ya que en el medio no existe muchas empresas que ofrezcan este servicio a excepción de las grandes, sin la necesidad de una gran inversión, ya que como se dijo anteriormente, Asterisk es totalmente de libre distribución.

Recomendaciones

- Se recomienda a los usuarios que no tienen conocimientos previos en Asterisk descargar una versión que incluya una interfaz gráfica para realizar las configuraciones.
- Cuando se descargue la librería de phpAGI sirve de gran ayuda hacerlo también con los ejemplos que se ofrecen en la página web.
- Al momento de elegir un *softphone* para realizar las pruebas, es recomendable investigar si es compatible con el tipo de extensión creada.

- Tener los casos de uso especificados antes de comenzar con el desarrollo del script, es de gran ayuda para el desarrollador, ya que con esto se puede conocer exactamente como se debe comportar el sistema.
- Los mensajes que serán reproducidos al usuario en el I.V.R. tienen que ser claros y puntuales.

Bibliografía

Capa tres soluciones. (2007). *Asterisk: Formación práctica sobre sistemasde voz ip basados en Asterisk.* [Fecha Último Acceso : 11-2-2015].

Cornu, S. (2011). *Asegurando Elastix.* Argentina: [Fecha Último Acceso : 10-5-2015].

Elastix Tech. (s.f.). *ElastixTech*. Recuperado el 02 de 05 de 2015, de http://elastixtech.com/fundamentos-de-telefonia/pbx-central-telefonica/

Fryer, B. (2011). *Elastix Network & Security Guide.* [Fecha Último Acceso : 10-11-2015]: Blue Packets (ACT, Australia).

Goncalves, F. E. (2007). *Asterisk PBX: Guía de configuración.* Rio de Janeiro: [Fecha Último Acceso : 20-08-2015].

Gorrotxategi, G., & Baz, I. ([s.a.]). *Voz sobre IP y Asterisk.* [Fecha Último Acceso : 10-11-2015]: Irontec.

Kouhfallah, H. (2012). *Elastix Easy.* [Fecha Último Acceso : 2-10-2015]: VoIp-Iran.

Landívar, E. (2008). *Comunicaciones unificadas con Elastix - Primera Edición.* [Fecha Último Acceso : 11-10-2015]: FDL.

Landívar, E. (2009). *Comunicaciones unificadas con Elastix.* [Fecha Último Acceso : 14-10-2014].

Madsen, L., Meggelen, J. V., & Bryant, R. (2011). *The future of telephony is now : Asterisk the definitive guide.* [Fecha Último Acceso : 10-08-2015]: O'Reilly.

Muñoz, A. (2010). *Elastix al ritmo del merengue.* [Fecha Último Acceso : 2-5-2015].

Sharif, B. (2008). *Elastix without tears.* [Fecha Último Acceso : 22-4-2015].

Simionovich, N. ([s.a.]). *Asterisk Gateway Interface 1.4 and 1.6 Programming.* [Fecha Último Acceso : 12-7-2014]: Packt publishing.

Solutions, P. (22 de 10 de 2012). *Elastix.* Recuperado el 2 de Marzo de 2015, de www.elastix.org

Wiki Asterisk. (10 de 02 de 2015). *Wiki Asterisk*. Obtenido de http://www.wikiasterisk.com/index.php/TTS_y_ASR

Anexos

Anexo 1. Tabla detallada de los comandos de Asterisk.

ANSWER

Sinopsis
Canal de Respuesta
Descripción
Contesta una llamada entrante. Devuelve 0 en caso de éxito, y -1 en caso de fallo.
Sintaxis
Answer()

ASYNCAGI BREAK

Sinopsis
Interrumpe el asyncagi de Asterisk.
Descripción
Termina una sesión de AGI asíncrono y tiene el canal de retorno para el plan de marcado de Asterisk.
Sintaxis
Asyncagi_break()

CHANNEL STATUS

Sinopsis
Devuelve el estado del canal conectado.
Descripción
Recuperar el estado del canal. Esto se utiliza para recuperar el estado actual del canal, tal como: contestado, colgado o timbrando. Valores de retorno: 0 - Canal caído y disponible. 1 - Canal caído, pero se reservó.

2 - Canal está descolgado.

3 - Dígitos (o equivalentes) se han marcado.

4 - Línea está sonando.

5 - Extremo remoto está sonando.

6 - Línea está para arriba.

7 - La línea está ocupada.

Sintaxis

channel_status($channel)

CONTROL STREAM FILE

Sinopsis

Envia un archivo de sonido en el canal y permite al oyente controlar su flujo.

Descripción

Transmitir los contenidos de un archivo a un canal, también permite al canal controlar el flujo. Devuelve 0 si la reproducción se completó sin interrupciones, o se devuelve el valor ASCII del dígito que se presionó, o -1 en el caso de que no se completara la reproducción.

Sintaxis

control_stream_file($filename,$escape_digits)

DATABASE DEL

Sinopsis

Elimina la llave de una base de datos.

Descripción

Elimina una llave/valor de la base de datos. Devuelve 1 si tienes éxito, y 0 de lo contrario.

Sintaxis

database_del($key)

DATABASE DELTREE

Sinopsis Elimina el árbol de llaves de una base de datos.
Descripción Elimina un árbol de llaves/valores incorporada en la base de datos. Devuelve 1 si tienes éxito, y 0 de lo contrario.
Sintaxis database_deltree($keytree)

DATABASE GET

Sinopsis Obtiene un valor de una base de datos.
Descripción Recupera el valor de una clave en la base de datos. Devuelve 0 si no se establece la clave, y 1 en el caso contrario.
Sintaxis database_get($key)

DATABASE PUT

Sinopsis Añade/actualiza un valor en la base de datos.
Descripción Establece el valor de una llave en la base de datos. Devuelve 1 si tiene éxito y 0 en caso contrario.
Sintaxis database_put($key)

EXEC

Sinopsis Añade/actualiza un valor en la base de datos.

Descripción

Ejecuta una aplicación determinada. En caso de éxito devuelve lo que la aplicación realice, en caso de que no encuentre la aplicación devuelve -2.

Sintaxis

exec($aplicacion,$opciones)

GET DATA

Sinopsis

Recibe tonos DTMF de un canal.

Descripción

Lee dígitos de la persona que está llamando. El usuario tiene la oportunidad de pulsar una tecla en cualquier momento durante el mensaje o durante el silencio posterior al mensaje. Si el usuario presiona una tecla mientras el mensaje se está reproduciendo, el mensaje deja de reproducirse.

Asterisk busca el archivo de audio a reproducirse en la ruta: /var/lib/asterisk/sounds/.

Cuando se pulsa la primera tecla un temporizador empieza a contar para comparar con el tiempo de espera en milisegundos. Cada vez que el usuario pulsa otra tecla del temporizador se reinicia. El comando termina cuando el contador llega a cero o cuando se introduzcan el número máximo de dígitos, lo que ocurra primero.

Si no se especifica el tiempo de espera, un tiempo predeterminado de 2000 se utiliza después de que se presione el último digito. Si no presionaron dígitos después de 6 segundos, el mensaje continúa con normalidad.

Si no se especifica el número máximo de dígitos, el usuario puede introducir la cantidad de dígitos que el desee.

Sintaxis

get_data($ArchivoDeAudio,$TiempoEspera,$MaximoDigitos)

GET FULL VARIABLE

Sinopsis

Evalúa la expresión de un canal.

Descripción

Evalúa una expresión del plan de marcado. Usted puede enviar una cadena de caracteres que contienen variables o funciones del plan de marcado, y Asterisk devolverá el resultado después de hacer las sustituciones apropiadas. Devuelve 0 si el nombre de la variable no está creado o si no existe el canal. Devuelve 1 si el nombre de la variable existe.

Sintaxis

get_full_variable($NombreVariable,$Canal)

GET OPTION

Sinopsis

Pide D.T.M.F. con tiempo de espera.

Descripción

Trasmitir un archivo de sonido a la espera de que la persona que llama presione un digito.

Sintaxis

get_option($NombreArchivo,$TeclaEscape,$TiempoMaximo)

GET VARIABLE

Sinopsis

Obtiene una variable del canal.

Descripción

Recuperar el valor de una variable del canal. Devuelve 0 si el nombre de la variable no está establecido, o devuelve 1 en el caso contrario más la variable entre paréntesis.

Sintaxis

get_variable($NombreVariable)

HANGUP

Sinopsis

Colgar el canal.

Descripción
Cuelga el canal especificado. Si no se da el nombre del canal, entonces cuelga el canal actual.
Sintaxis
hangup ($NombreCanal)

NOOP

Sinopsis
No hacer nada.
Descripción
No hacer nada. Recibirá un resultado de la respuesta de este comando.
Sintaxis
noop ()

RECEIVE CHAR

Sinopsis
Recibe un carácter del canal que lo soporta.
Descripción
Recibe un solo carácter. Esto solo funciona para los tipos de canales que lo apoyan, como IAX2 utilizando marcos de texto o SIP utilizando el método de mensaje. Devuelve el valor en decimal del carácter si se recibe, o 0 si el canal no soporta la recepción de texto. Devuelve -1 en el caso de que se cuelgue la llamada o que se dé un error.
Sintaxis
receive_char ($TiempoEspera)

RECEIVE TEXT

Sinopsis
Recibe texto del canal que lo soporta.
Descripción

Recibe un mensaje de texto. Esto solo funciona en los mismos casos que receive char. Devuelve -1 en caso de error, o 1 en el caso de éxito, más el texto entre paréntesis.

Sintaxis

receive_text ($TiempoEspera)

RECORD FILE

Sinopsis

Graba un archivo dado.

Descripción

Graba el audio de la persona que llama en un archivo. Devuelve -1 en caso de error. El formato especifica qué tipo de audio se va a grabar. El tiempo de espera es el número máximo en milisegundos. El parámetro silencio es el número de segundos de silencio permitidos antes de que se devuelva a la función por falta de un tono D.T.M.F.

Sintaxis

record_file($filename,$format,$escape_digits,$timeout,$offset,$samples,$beep,$s=silence)

SAY ALPHA

Sinopsis

Reproduce una cadena de caracteres.

Descripción

Dice una cadena de caracteres dada, terminándola el momento que se interrumpe por la pulsación de un digito D.T.M.F. Devuelve 0 si la reproducción se completó sin interrupción alguna, en caso o de colgar devolverá -1.

Sintaxis

say_alpha($numero,$digito_escape)

SAY DATE

Sinopsis

Reproduce la fecha.

Descripción

Dice la fecha, terminándola el momento que se interrumpe por la pulsación de un digito D.T.M.F. Devuelve 0 si la reproducción se completó sin interrupción alguna, en caso o de colgar devolverá -1.

Sintaxis

say_date($fecha,$digito_escape)

SAY DATETIME

Sinopsis

Reproduce la hora en un formato especificado.

Descripción

Dice la hora, terminándola el momento que se interrumpe por la pulsación de un digito D.T.M.F. Devuelve 0 si la reproducción se completó sin interrupción alguna, en caso o de colgar devolverá -1.

Formato:

A: Día de la semana

B: Mes (Texto)

m: Mes (Numero)

d: Día del mes

Y: Año

I: Hora (12-horas formato)

H: Hora (24-hour formato)

M: Minutos

P: AM/PM

S: Segundos

T: Zona horaria

Sintaxis

say_datetime($hora,$dígitos_escape,$formato,$zona_horaria)

SAY DIGITS

Sinopsis
Reproduce una cadena de dígitos.
Descripción
Dice una cadena de dígitos dada. Por ejemplo, el número 100 se diría como: uno, cero, cero. Terminándola el momento que se interrumpe por la pulsación de un digito D.T.M.F. Devuelve 0 si la reproducción se completó sin interrupción alguna, en caso o de colgar devolverá -1.
Sintaxis
say_digits($numero,$digito_escape)

SAY NUMBER

Sinopsis
Reproduce un número dado.
Descripción
Dice un número. Por ejemplo, el número 100 se diría como: cien. Terminándola el momento que se interrumpe por la pulsación de un digito D.T.M.F. Devuelve 0 si la reproducción se completó sin interrupción alguna, en caso o de colgar devolverá -1.
Sintaxis
say_number($numero,$digito_escape)

SAY PHONETIC

Sinopsis
Reproduce caracteres dados usando la fonética.
Descripción
Dice una cadena de caracteres, pero usando una palabra común de cada letra (Alpha, Bravo, Charlie). Devuelve 0 si la reproducción se completó sin interrupción alguna, en caso o de colgar devolverá -1.

Sintaxis

say_phonetic($CadenaCaracteres,$digito_escape)

SAY TIME

Sinopsis

Reproduce la hora.

Descripción

Dice la hora. Devuelve 0 si la reproducción se completó sin interrupción alguna, en caso o de colgar devolverá -1.

Sintaxis

say_time($hora,$digito_escape)

SEND IMAGE

Sinopsis

Envía imágenes a un canal que lo soporte.

Descripción

Envía una imagen hacia un canal. La mayoría de canales no son compatibles con la trasmisión de imágenes. Devuelve 0 si la imagen es enviada, y cuando el canal receptor no es compatible. Devuelve -1 en el caso de error o se finalice la llamada. Los nombres de las imágenes deben enviarse sin su extensión.

Sintaxis

send_image($imagen)

SEND TEXT

Sinopsis

Envía texto a un canal que lo soporte

Descripción

Envía texto hacia un canal. La mayoría de canales no son compatibles con la trasmisión de

texto. Devuelve 0 si el texto es enviado, y cuando el canal receptor no es compatible. Devuelve -1 en el caso de error o se finalice la llamada.
Sintaxis send_image($imagen)

SET AUTOHANGOUT

Sinopsis Auto colgado en algún momento de la llamada.
Descripción Programa el canal que va a ser colgado en un punto determinado del futuro en segundos.
Sintaxis set_autohangout($tiempo)

SET CALLERID

Sinopsis Asigna nombre e identificación de usuario.
Descripción Establece el nombre de la persona que llama y su número de identificación en el canal.
Sintaxis set_callerid ($numero)

SET CONTEXT

Sinopsis Establece contexto del canal.
Descripción Establece el actual contexto del plan de marcado en el canal.
Sintaxis set_context ($ContextoDeseado)

SET EXTENSION

Sinopsis
Cambia la extensión del canal.
Descripción
Cambia la extensión del plan de marcado en el canal al finalizar la aplicación.
Sintaxis
set_extension ($NuevaExtension)

SET MUSIC

Sinopsis
Activa/Desactiva música de espera.
Descripción
Inicia o detiene la música de espera en el canal. Si no se especifica el archivo entonces se utilizará la clase que viene por defecto. Siempre devuelve 0.
Sintaxis
set_music ($on/off,$Clase)

SET PRIORITY

Sinopsis
Establece la prioridad del plan de marcado en el canal.
Descripción
Cambia la prioridad para el siguiente paso que viene después de salir de la aplicación.
Sintaxis
set_priority ($priority)

SET VARIABLE

Sinopsis Establece una variable de canal.
Descripción Establece una variable de canal a un valor dado.
Sintaxis set_variable ($NombreVariable,$ValorVariable)

SPEECH ACTIVATE GRAMMAR

Sinopsis Activa una gramática.
Descripción Activa una gramática que se ha cargado.
Sintaxis speech_activate_grammar ($NombreGramatica)

SPEECH CREATE

Sinopsis Crea un objeto de voz.
Descripción Inicializa el reconocimiento de voz y crea un objeto para ser utilizado por otros comandos de voz AGI.
Sintaxis speech_create ($Motor)

SPEECH DEACTIVATE GRAMMAR

Sinopsis Desactiva una gramática.

Descripción
Desactiva una gramática especificada en el objeto de voz.
Sintaxis
speech_deactivated_grammar ($NombreGramatica)

SPEECH DESTROY

Sinopsis
Destruye un objeto de voz.
Descripción
Destruye los recursos que se asignaron para hacer el reconocimiento de voz. Este comando debe ser el último comando de voz utilizado. Destruye objetos creados por speach_create.
Sintaxis
speech_destroy ($Motor)

SPEECH LOAD GRAMMAR

Sinopsis
Carga una gramática.
Descripción
Carga una gramática específica con un nombre específico.
Sintaxis
speech_load_grammar ($NombreGramatica,$PathGramatica)

SPEECH RECOGNIZE

Sinopsis
Reconoce la voz.
Descripción

Suena un aviso y comienza el reconocimiento por voz, así como esperar a que presione los dígitos del teclado.

Sintaxis

speech_recognize ($Prompt,$TiempoEspera,$Offset)

SPEECH SET

Sinopsis

Ajustes del motor de reconocimiento de voz.

Descripción

Estable los ajustes del motor de reconocimiento. Los ajustes que están disponibles son especificados para el motor de reconocimiento en uso.

Sintaxis

speech_set ($Nombre,$Valor)

SPEECH UNLOAD GRAMMAR

Sinopsis

Descarga una gramática.

Descripción

Descarga una gramática específica.

Sintaxis

speech_unload_grammar ($NombreGramatica)

STREAM FILE

Sinopsis

Transmite el contenido de un archivo a un canal.

Descripción Envía un archivo dado, reproduciendo el archivo con la posibilidad de que este sea interrumpido con un dígito asignado. Devuelve 0 si la reproducción se completó sin interrupciones, o el valor del dígito presionado. Devuelve -1 en el caso de error o que se termine la llamada.
Sintaxis stream_file ($NombreArchivo,$DigitoEscape)

VERBOSE

Sinopsis Muestra mensajes en la consola de Asterisk.
Descripción Envía un mensaje detallado al canal registrador. Los mensajes detallados aparecen en la consola de Asterisk.
Sintaxis verbose ($Mensaje)

WAIT FOR DIGIT

Sinopsis Muestra mensajes en la consola de Asterisk.
Descripción Espera a que la persona que llame presione un dígito. Devuelve -1 en caso de fallo o si ningún digito se recibe dentro del tiempo de espera.
Sintaxis wait_for_digit ($Mensaje)

<u>Anexo 2.</u> Código desarrollado en PHP para consultar en la base de datos.

```
//Esta línea se usa para ejecutar archivos php desde consola

#!/usr/bin/php -q

<?php

//Se limita el tiempo de conexión a 30 segundos.

set_time_limit(30);

//Se llama a la librería phpagi.

require_once "phpagi.php";

//Se activa la notificación de todos los errores por consola

error_reporting(E_ALL);

//Se crea una nueva instancia de la clase AGI

$agi = new AGI();

//Se ejecuta el comando de Asteriks "Answer", que es utilizado para contestar una llamada.

$agi->answer();

do {

        //Se utiliza la función creada "tts" para enviar un mensaje hablado al usuario.

        //Se pide al usuario que ingrese su número de cédula seguido del signo numeral.

        tts($agi, "Después del tono, ingrese su número de cédula seguido del signo numeral. Presione 0 para salir.");

        //Se envía un sonido "beep" y se tiene un tiempo de espera máximo de 20 segundos.

        $resultado = $agi->get_data('beep', 3000, 20);

        //Se obtiene los tonos pulsados desde el teléfono y se almacena en una variable.

        $id = $resultado['result'];
```

```
        //Se verifica si se ha ingresado algún valor.
        if ($id=="") {
                //Si no es ingresado nada, se emite un mensaje de error.
                tts($agi, "No se ha ingresado ningún número");
                tts($agi, "Intentelo nuevamente");
        }
        //Si el usuario digito el número "0" entonces el script llegara a su fin.
        elseif ($id==0) {
//Se emite un mensaje de fin.
                tts($agi, "Gracias. Hasta luego");
        }
        else {
//Se ingresa en el caso de que se ha ingresado alguna información.
//Es utilizado "pre_replace" para generar una búsqueda y sustitución de una expresión regular.
                //En este caso para obtener el id.
                $idtts=preg_replace("/(.)/i", "\${1}, ", $id);
// Deben ingresarse los valores para la conexión de la base de datos.
                $usuario = 'root';
                $clave = 'elastix';
                $servidor = 'localhost';
//Este es el nombre de la base de datos a usarse.
                $basededatos = 'sistema';
```

```
//Se establece la conexión con la base de datos.

        $conexion=mysql_connect($servidor, $usuario, $clave);

        mysql_select_db($basededatos, $conexion) or die("could not open
        database");

        mysql_query("SET character_set_results = 'utf8', character_set_client =
        'utf8', character_set_connection = 'utf8', character_set_database = 'utf8',
        character_set_server = 'utf8'", $conexion);

// COMIENZO DEL QUERY PRINCIPAL.

//Se realiza una consulta de toda la información de un cliente

//utilizando su "id" o cédula de identidad.

        $consultatxt=" SELECT * FROM deuda WHERE cedula = $id";

//El comando "verbose" es usado para enviar a la consola de Asterisk

//la consulta realizada.

        $agi->verbose('La consulta SQL es: '.$consultatxt);

// SE PROCESA EL QUERY PRINCIPAL.

        $consulta=mysql_query($consultatxt, $conexion) or die(mysql_error());

//Se toma el número de campos de la consulta

        $campos = mysql_num_rows($consulta);

//Si verifica si existen campos que coincidan

        if ($campos==0)

                {

// Si no existen entonces es desplegado un mensaje de error.

                tts($agi,"No se encuentra el número de cedula: $idtts");

                tts($agi, "Intentelo nuevamente");
```

```
                }
//Caso contrario
                else
                {
//Es realizada esta consulta con el objetivo de obtener el nombre del usuario mediante su número de cédula.
                        $consultatxt2=" SELECT * FROM cliente WHERE cedula = $id";
//Es enviada la consulta hacia la consola de Asterisk.
                        $agi->verbose('La consulta SQL es: '.$consultatxt2);
                        $consulta2=mysql_query($consultatxt2, $conexion) or die(mysql_error());
                        while ($datos2 = mysql_fetch_assoc($consulta2))
                        {
//Es almacenado en una variable el nombre del cliente
                                $nombres=$datos2['nombres'];
                        }
//Es desplegado un mensaje de bienvenida al usuario.
                        tts($agi,"Bienvenido: $nombres");
                        while ($datos = mysql_fetch_assoc($consulta))
                        {
                                do{
// Conocer lo que debe el cliente hasta la fecha actual
```

```
                $consultatxt3=" SELECT sum(saldo)as deuda FROM
                deudacliente WHERE saldo<>'0' and fechaPago<curdate()";
//Es enviada la consulta a la consola de Asterisk.
                $agi->verbose('La consulta SQL es: '.$consultatxt3);
                $consulta3=mysql_query($consultatxt3, $conexion) or
                die(mysql_error());
                while ($datos3 = mysql_fetch_assoc($consulta3))
                {
//Es almacenada la deuda del usuario en una variable.
                        $deuda=$datos3['deuda'];
                }
// Es realizada una consulta para conocer el saldo vencido del cliente hasta la fecha
actual
                $consultatxt4=" SELECT (sum(deudacliente.valor)-
sum(deudacliente.saldo))as vencido FROM deudacliente WHERE
fechaPago<curdate()";
//Es enviada la consulta hacia la consola de Asteriks.
                $agi->verbose('La consulta SQL es: '.$consultatxt4);
                $consulta4=mysql_query($consultatxt4, $conexion) or
                die(mysql_error());
                while ($datos4 = mysql_fetch_assoc($consulta4))
                {
//Es almacenada en una variable el valor del saldo vencido.
                        $vencido=$datos4['vencido'];
                }
```

```
//Es realizada una consulta para conocer el saldo por vencer del cliente desde la fecha actual

                $consultatxt5=" SELECT sum(deudacliente.saldo)as SaldoVencer FROM deudacliente WHERE fechaPago>curdate()";

//Es enviada la consulta hacia la consola de Asterisk.

                $agi->verbose('La consulta SQL es: '.$consultatxt5);

                $consulta5=mysql_query($consultatxt5, $conexion) or die(mysql_error());

                while ($datos5 = mysql_fetch_assoc($consulta5))

                {

//Es almacenada en una variable el valor del saldo por vencer.

                        $SaldoVencer=$datos5['SaldoVencer'];

                }

//Es realizada una consulta para conocer la fecha máxima de pago para la consulta de saldo por vencer

                $consultatxt6=" SELECT fechaPago as FechaMaxima FROM deudacliente WHERE fechaPago>curdate() order by fechaPago desc limit 1";

//Es enviada hacia la consulta hacia la consola de Asterisk.

                $agi->verbose('La consulta SQL es: '.$consultatxt6);

                $consulta6=mysql_query($consultatxt6, $conexion) or die(mysql_error());

                while ($datos6 = mysql_fetch_assoc($consulta6))

                {

//Es almacenada en una variable la fecha máxima de pago.
```

```
                $FechaMaxima=$datos6['FechaMaxima'];

            }

//Es desplegado un menú para seleccionar una consulta.

                tts($agi, "Para escuchar su deuda. Presione uno
                seguido de signo numeral");

                tts($agi, "Para escuchar su saldo vencido. Presione
                dos seguido de signo numeral");

                tts($agi, "Para escuchar su saldo por vencer.
                Presione tres seguido de signo numeral");

                tts($agi, "Presione cero para salir");

//Es reproducido un "beep"

                $resultado2 = $agi->get_data('beep', 3000, 20);

//Es almacenado en una variable el valor introducido.

                $menu1 = $resultado2['result'];

//Si el menú es "1" se reproduce la deuda.

                if($menu1==1)

                {

                    tts($agi, "Usted tiene una deuda de:  $deuda
                    dolares");



                }

//Si el menú es "2" se reproduce el saldo vencido.

                if($menu1==2)

                {
```

```
                                        tts($agi, "Usted tiene un saldo vencido de:
$vencido dolares");


                                    }
//Si el menú es "3" se reproduce el saldo por vencer.
                                    if($menu1==3)
                                    {
                                        tts($agi, "Usted tiene un saldo por vencer
                                        de:  $SaldoVencer dolares");
                                        tts($agi, "Y tiene que pagarlo hasta la fecha:
                                        $FechaMaxima");


                                    }

//Se comprueba si se ha ingresado la opción "0"
//Si se da esa condición entonces se termina la llamada.
                                }while ($menu1 != '0');
//Es terminada la conexión MySQL
                                mysql_close($conexion);
//Se cuelga la llamada.
                                $agi->hangup();
                            }
                        }
                    }
```

```
//Se comprueba si se ha ingresado la opción "0"
//Si se da esa condición entonces se termina la llamada.
} while($id != '0');
//Es terminada la conexión MySQL
mysql_close($conexion);
//Se cuelga la llamada.
$agi->hangup();


//Función para pasar de texto a voz
function tts($agi, $texto) {
//Es comprobado que el valor enviado hacia la función no sea vacío.
        if ($texto!="") {
//Se muestra en la consola el texto enviado.
                $agi->verbose("$texto");
//Se convierte de "UTF-8" a "ISO-8859-1" para una mejor reproducción.
                $tts=iconv("UTF-8","ISO-8859-1", "$texto");
//Se utiliza el comando de Festival "text2wav" para transformar un texto a voz.
                $agi->text2wav($tts);
        }
        else {
//Si no se ingresó nada es enviado un mensaje de alerta a la consola de Asterisk.
                $agi->verbose(No se recibió nada);
        }
```

```
}
?>
```

Anexo 3. SQL de la base de datos

```
-- Base de datos: `sistema`
CREATE TABLE IF NOT EXISTS `cliente` (
  `cedula` int(10) NOT NULL,
  `nombres` varchar(100) NOT NULL,
  `apellidos` varchar(100) NOT NULL,
  `edad` varchar(100) NOT NULL,
  `telefono` varchar(100) NOT NULL,
  `email` varchar(100) NOT NULL,
  `domicilio` varchar(100) NOT NULL,
  `seguridad` tinyint(4) NOT NULL default '0',
  `codigoSeguridad` varchar(20) NOT NULL,
  PRIMARY KEY  (`cedula`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Estructura de tabla para la tabla `deuda`
--

CREATE TABLE IF NOT EXISTS `deuda` (
```

```
  `cedula` int(10) NOT NULL,
  `cantidad` varchar(100) NOT NULL,
  `fechaActual` date NOT NULL,
  `observaciones` varchar(254) NOT NULL,
  `numeroPagos` int(11) NOT NULL,
  PRIMARY KEY  (`cedula`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Estructura de tabla para la tabla `deudacliente`
--

CREATE TABLE IF NOT EXISTS `deudacliente` (
  `cedula` int(11) NOT NULL,
  `numeroPago` int(11) NOT NULL,
  `valor` int(11) NOT NULL,
  `saldo` int(11) NOT NULL,
  `fechaPago` date NOT NULL,
  `id` int(11) NOT NULL auto_increment,
  PRIMARY KEY  (`id`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=5 ;
```

DOCTOR ROMEL MACHADO CLAVIJO,

SECRETARIO DE LA FACULTAD DE CIENCIAS DE LA ADMINISTRACION

DE LA UNIVERSIDAD DEL AZUAY,

C E R T I F I C A:

Que, el H, Consejo de Facultad de Ciencias de la Administración en sesión del 27 de junio de 2014, conoció la petición del señor **ALVARO JOSE AVILA ALVARADO (50487)** que denuncia su trabajo de titulación denominado: **"SISTEMA DE INFORMACIÓN DE TELEFONÍA IP PARA LA ASISTENCIA DE CARTERA, UTILIZANDO LIBRERÍAS AGI (ASTERISK GATEWAY INTERFACE)"**, presentado como requisito previo a la obtención del Grado de Ingeniero de Sistemas y Telemática. El Consejo acoge el informe de la Junta Académica y aprueba la denuncia. Designa como Director de dicho trabajo al ingeniero Marcos Orellana Cordero y como miembros del Tribunal examinador a los ingenieros Esteban Crespo Martínez y Juan Córdova Ochoa; De conformidad con la disposición general tercera del Reglamento de Régimen Académico, el peticionario tiene un plazo equivalente a dos períodos académicos ordinarios (semestres) para desarrollar y terminar su trabajo de titulación, esto es hasta el 27 de junio de 2015.-

Cuenca, junio 27 de 2014

UNIVERSIDAD DEL AZUAY
FACULTAD DE ADMINISTRACION
SECRETARIA

Doctora Jenny Ríos Coello, Secretaria de la Facultad de Ciencias de la Administración de la Universidad del Azuay,

C E R T I F I C A:

Que, el H. Consejo de Facultad en sesión realizada el 18 de junio de 2015, conoció la petición del estudiante **ALAVARO JOSE AVILA ALVARADO,** con código 50487, quien solicita prórroga para la presentación del trabajo de titulación "Sistema de Información de Telefonía IP para la asistencia de cartera, utilizando librerías AGI (ASTERSK GATEWAY INTERFACE). El Consejo de Facultad, de conformidad con las disposiciones reglamentarias, le concede la prórroga de seis meses, esto es hasta el **27 de diciembre de 2015.**

Cuenca, junio 25 de 2015

CONVOCATORIA

Por disposición de la Junta Académica de Ingeniería de Sistemas y Telemática **CONVOCO** a los Miembros del Tribunal Examinador, a la sustentación del Protocolo del Trabajo de Titulación denominado: **"SISTEMA DE INFORMACION DE TELEFONIA IP PARA LA ASISTENCIA DE CARTERA, UTILIZANDO LIBRERIAS AGI (ASTERISK GATEWAY INTERFACE)"** presentado por el señor **ALVARO JOSE AVILA ALVARADO (50487),** previa a la obtención del grado de Ingeniera de Sistemas y Telemática, para el día **MIERCOLES 25 DE JUNIO DE 2014, a las 19h00**

Cuenca, 19 de junio de 2014

Dr. Romel Machado Clavijo
Secretario de la Facultad

Ing. Marcos Orellana Cordero

Ing. Esteban Crespo Martínez

Ing. Juan Córdova Ochoa

CONVOCATORIA

Comunicado

Oficio Nro. 049-2014-DIST-UDA

Cuenca, 21 de Mayo de 2014

Señor Ingeniero
Xavier Ortega Vázquez
DECANO DE LA FACULTAD DE CIENCIAS DE LA ADMNISTRACIÓN
Presente.-

De nuestras consideraciones:

La Junta Académica de la Escuela de Ingeniería de Sistemas y Telemática, reunida el día 21 de Mayo del 2014, revisó el proyecto de tesis titulado "Sistema de Información de Telefonía IP para la Asistencia de Cartera, utilizando librerías AGI (Asterisk Gateway Interface)", presentada por el estudiante Álvaro Ávila, estudiante de la Escuela de Ingeniería de Sistemas, previo a la obtención del título de Ingeniero de Sistemas.

La Junta considera que el diseño de trabajo de titulación cumple con los requisitos normados en la "Guía de Elaboración y Presentación de la Denuncia/Protocolo de Trabajo de Titulación", razón por la cual solicita, por su digno intermedio, notificar al tribunal designado y determinar lugar, fecha y hora de sustentación.

Por lo expuesto, y de conformidad con el Reglamento de Graduación de la Facultad, recomienda como director y responsable de aplicar cualquier modificación al diseño del trabajo de graduación posterior al Ing. Marcos Orellana, y como miembros del Tribunal al Ing. Esteban Crespo e Ing. Juan Córdova.

Atentamente,

Ing. Marcos Orellana Cordero
Director Escuela de Ingeniería de Sistemas y Telemática
Universidad del Azuay

ACTA

SUSTENTACIÓN DE PROTOCOLO/DENUNCIA DEL TRABAJO DE TITULACIÓN

1.1. Nombre del estudiante: ALVARO JOSE AVILA ALVARADO

1.1.1 Código 50487

1.1.2 **Director sugerido:** Ing. Marcos Orellana Cordero

1.1.3 **Codirector (opcional):** _____

1.2 **Tribunal:** Ings. Esteban Crespo y Juan Ochoa

1.3 **Título propuesto:** SISTEMA DE INFORMACION DE TELEFONIA IP PARA LA ASISTENCIA DE CARTERA, UTILIZANDO LIBRERIAS AGI (ASTERISK GATEWAY INTERFACE)

1.4 **Resolución:**

1.4.1 Aceptado sin modificaciones ✓

1.4.2 Aceptado con las siguientes modificaciones:

_____

_____

_____

1.1.1 Responsable de dar seguimiento a las modificaciones (designado por la Junta Académica de entre los Miembros del Tribunal): Ing. Marcos Orellana Cordero

1.1.2 No aceptado

- Justificación:

_____

_____

_____

Tribunal

.....	.....
Ing. Marcos Orellana C.	Ing. Esteban Crespo M.
.....	.....
Ing. Juan Córdova O..	Sr. Álvaro J. Ávila A.
.....	.....
	Secretario de Facultad

Fecha de sustentación: 25 de Junio de 2014

RÚBRICA PARA LA EVALUACIÓN DEL PROTOCOLO DE TRABAJO DE TITULACIÓN

1.1Nombre del estudiante: ALVARO JOSE AVILA ALVARADO

1.1.1. Código (50487)

1.1.2. **1.2 Director sugerido**: Ing. Marcos Orellana Cordero

1.3 **Codirector (opcional)**:

1.4. **Título propuesto:** SISTEMA DE INFORMACION DE TELEFONIA IP PARA LA ASISTENCIA DE CARTERA, UTILIZANDO LIBRERIAS AGI (ASTERISK GATEWAY INTERFACE)

1.1 **1.5 Revisores (tribunal)**: Ings. Esteban Crespo y Juan Ochoa

1.6 Recomendaciones generales de la revisión:

	Cumple totalmente	Cumple parcialmente	No cumple	Observaciones (*)
Línea de investigación				
1. ¿El contenido se enmarca en la línea de investigación seleccionada?	✓			
Título Propuesto				
2. ¿Es informativo?	✓			
3. ¿Es conciso?	✓			
Estado del arte				
4. ¿Identifica claramente el contexto histórico, científico, global y regional del tema del trabajo?	✓			
5. ¿Describe la teoría en la que se enmarca el trabajo	✓			
6. ¿Describe los trabajos relacionados más relevantes?		✓		
7. ¿Utiliza citas bibliográficas?	✓			
Problemática y/o pregunta de investigación				
8. ¿Presenta una descripción precisa y clara?	✓			
9. ¿Tiene relevancia profesional y social?	✓			
Hipótesis (opcional)				
10. ¿Se expresa de forma clara?				
11. ¿Es factible de verificación?				
Objetivo general				
12. ¿Concuerda con el problema formulado?	✓			
13. ¿Se encuentra redactado en tiempo verbal infinitivo?	✓			
Objetivos específicos				
14. ¿Concuerdan con el objetivo general?	✓			
15. ¿Son comprobables cualitativa o cuantitativamente?	✓			
Metodología				

16. ¿Se encuentran disponibles los datos y materiales mencionados?				
17. ¿Las actividades se presentan siguiendo una secuencia lógica?				
18. ¿Las actividades permitirán la consecución de los objetivos específicos planteados?				
19. ¿Los datos, materiales y actividades mencionadas son adecuados para resolver el problema formulado?				
Resultados esperados				
20. ¿Son relevantes para resolver o contribuir con el problema formulado?				
21. ¿Concuerdan con los objetivos específicos?				
22. ¿Se detalla la forma de presentación de los resultados?				
23. ¿Los resultados esperados son consecuencia, en todos los casos, de las actividades mencionadas?				
Supuestos y riesgos				
24. ¿Se mencionan los supuestos y riesgos más relevantes?				
25. ¿Es conveniente llevar a cabo el trabajo dado los supuestos y riesgos mencionados?				
Presupuesto				
26. ¿El presupuesto es razonable?				
27. ¿Se consideran los rubros más relevantes?				
Cronograma				
28. ¿Los plazos para las actividades son realistas?				
Referencias				
29. ¿Se siguen las recomendaciones de normas internacionales para citar?				
Expresión escrita				
30. ¿La redacción es clara y fácilmente comprensible?				
31. ¿El texto se encuentra libre de faltas ortográficas?				

(*) Breve justificación, explicación o recomendación.

- Opcional cuando cumple totalmente,

- Obligatorio cuando cumple parcialmente y NO cumple.

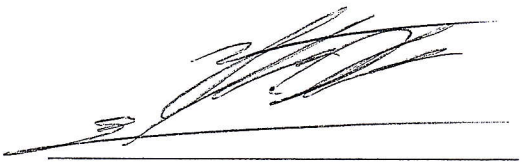
.....

.....

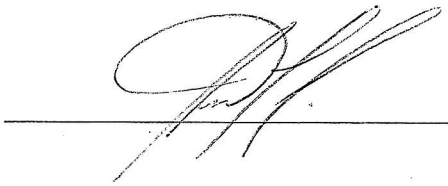
.....

.....

Ing. Marcos Orellana Cordero



Ing. Esteban Crespo Martínez



Ing. Juan Córdova Ochoa

Sustentación del Diseño de Tesis (Doctor Romel Machado Clavijo)

Fecha: 17-06-2014

ESCUELA DE INGENIERIA DE SISTEMAS

Diseños de Tesis
Escuela de Sistemas
Estudiante: Álvaro José Ávila Ávila Alvarado con código 50487.
Tema: "Sistema de Información de telefonía IP para la Asistencia de Cartera, utilizando librerías AGI (Asterisk Gateway Interface)"
Para: La obtención del título de Ingeniero en Sistemas
Director: Ing. Marcos Orellana.
Tribunal: Ing. Esteban Crespo
Tribunal: Ing. Juan Córdova.

DIA: Miercoles
FECHA: 25 Junio
HORA: 19h00.

comunicado alumno.

Oficio Nro. 069-2014-DIST-UDA

Cuenca, 26 de Junio de 2014

Señor Ingeniero

Xavier Ortega Vázquez

DECANO DE LA FACULTAD DE CIENCIAS DE LA ADMNISTRACIÓN

Presente.-

De nuestras consideraciones:

La Junta Académica de la Escuela de Ingeniería de Sistemas y Telemática, reunida el día 26 de Junio del 2014, revisó la documentación del proyecto de tesis denominado "Sistema de información de telefonía IP para la asistencia de cartera, utilizando librerías AGI(Asterisk Gateway Interface)", presentado por el estudiante Álvaro Ávila, estudiante de la Escuela de Ingeniería de Sistemas, previo a la obtención del título de Ingeniero de Sistemas.

La Junta considera que la documentación cumple con las normas legales y reglamentarias de la Universidad y de la Facultad de Ciencias de la Administración y avala la aprobación por parte del tribunal designado, así por su digno intermedio, el conocimiento y aprobación por parte del Consejo de Facultad.

Atentamente,

Ing. Marcos Orellana Cordero
Director Escuela de Ingeniería de Sistemas y Telemática
Universidad del Azuay

UNIVERSIDAD DEL AZUAY

Cuenca, 17 de junio de 2014

Ing.

Xavier Ortega Vásquez, MBA.

DECANO DE LA FACULTAD DE CIENCIAS DE LA ADMINISTRACION

Ciudad

De mis consideraciones:

Álvaro José Ávila Alvarado con código 50487, egresado de la Escuela de Ingeniería de Sistemas y Telemática de la facultad de Ciencias de la Administración, solicito a usted de la forma más comedida y por su intermedio al Consejo de Facultad, la aprobación del diseño de tesis con el tema *"Sistema de Información de telefonía IP para la Asistencia de Cartera, utilizando librerías AGI (Asterisk Gateway Interface)"*, previo a la obtención del título de Ingeniero de Sistemas y Telemática.

Me permito sugerir el nombre del Ing. Marcos Orellana como director de tesis, puesto que he recibido asesoramiento y cuento con su aprobación.

Atentamente,

Álvaro José Ávila Alvarado

Edición autorizada de 20.000 ejemplares
Del 618.501 al 638.500 Nº 0637212

Cuenca, 17 de junio de 2014

Ing.

Xavier Ortega Vásquez, MBA

Decano de la Facultad de Ciencias de la Administración

Presente

De mi consideración:

Por la presente, me permito informarle que he revisado el diseño de tesis presentado por la estudiante **Álvaro José Ávila Alvarado** con el tema *"Sistema de Información de telefonía IP para la Asistencia de Cartera, utilizando librerías AGI (Asterisk Gateway Interface)"*, como requisito previo para la obtención del título de Ingeniero de Sistemas y Telemática.

Al respecto, el diseño de tesis presenta una estructura teórica, metodológica y técnica coherente, cuyo objetivo es comprender cómo utiliza las librerías AGI(Asterisk Gateway Interface) para obtener información de una base de datos mediante el uso de telefonía IP. Por lo expuesto, emito informe favorable y recomiendo su aprobación.

Atentamente

Ing. Marcos Orellana

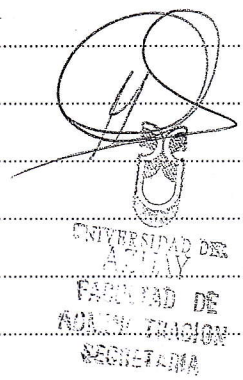
UNIVERSIDAD DEL AZUAY

DOCTORA JENNY RIOS COELLO, SECRETARIA DE LA FACULTAD DE CIENCIAS DE LA ADMINISTRACION DE LA UNIVERSIDAD DEL AZUAY

C E R T I F I C A:

Que, el Señor **Álvaro José Ávila Alvarado**, registrado con código **50487** perteneciente a la Escuela de Ingeniería de Sistemas tiene aprobado más del 80% de pensum de estudios.

Cuenca, Junio 04 de 2014



Derecho 56302
vcf.-

Edición autorizada de 20.000 ejemplares
Del 618.501 al 638.500
N° 0635471

UNIVERSIDAD DEL AZUAY

Universidad del Azuay

Facultad de Ciencias de la Administración

Escuela de Ingeniería de Sistemas y Telemática

DENUNCIA DE TESIS

ALUMNO:

Álvaro Ávila

FECHA:

17/06/2014

Edición autorizada de 20.000 ejemplares
Del 638.501 al 658.500
Nº 0638793

CONTENIDO

UNIVERSIDAD DEL AZUAY

1 DATOS GENERALES

1.1 DATOS ESTUDIANTE

Álvaro José Ávila Alvarado

Código: 50487

Celular: 0995832019

Teléfono: 4097520

Correo Electrónica: alvaroavila19874@gmail.com

1.2 DIRECTOR SUGERIDO

Ingeniero Marcos Orellana

Celular:

Correo Electrónico: marore@uazuay.edu.ec

1.3 CODIRECTOR SUGERIDO

1.4 ASESOR METODOLÓGICO

1.5 TRIBUNAL DESIGNADO

Ing. Marcos Orellana

Ing. Esteban Crespo

Ing. Juan Córdova

1.6 APROBACIÓN

Junta Académica:

Consejo de Facultad:

1.7 Línea de Investigación

1.7.1 Código UNESCO

Línea: 1203 Informática de Computadoras

Programa: 1203.99 Sistemas de Asistencia al usuario

1.7.2 Tipo de Trabajo

Investigación Formativa

1.8 Área de Estudio

Telemática

1.9 Título Propuesto

Sistema de Información de telefonía IP para la Asistencia de Cartera, utilizando librerías AGI (Asterisk Gateway Interface)

1.10 Estado del Proyecto

Trabajo Nuevo

2 Contenido

2.1 Motivación de la Investigación

En los últimos años se han desarrollado programas basados en software libre que realizan todas las funciones de una central telefónica (PBX). Uno de estos programas de distribución libre es Elastix basado en un entorno de programación llamado Asterisk. Entre las funciones de Elastix está incluida AGI (Asterisk Gateway Interface) la cual permite programar en algunos lenguajes no nativos; uno de los más usados es el PHP (Hypertext Pre-processor), este permite la conexión a cualquier base de datos. En nuestro medio este tema no ha sido ampliamente difundido por lo que es oportuna la construcción de una interface de consulta con las características que se describirán más adelante.

2.2 Problemática

Muchas empresas de nuestro medio todavía utilizan centrales telefónicas análogas para la comunicación, las que tienen un costo bastante elevado comparado con las centrales PBX de telefonía IP, que ofrece un notable cambio en cuestión de costos y opciones.

Es muy beneficioso para las empresas adoptar este tipo de tecnología ya que ofrece la posibilidad de recibir y enviar información a los usuarios por medio del teléfono.

Instituciones que utilizan un servidor de comunicaciones como Elastix o Asterisk no conocen muchas de las opciones que estos ofrecen, como por ejemplo los IVR (Interactive Voice Response). Los IVR son aplicaciones de voz interactivas que aceptan como entrada tonos marcados por el usuario, entregando distintos tipos de respuesta según la programación que le demos en el sistema.

Los IVR son comúnmente implementados en empresas que reciben grandes cantidades de llamadas, a fin de reducir la necesidad de personal y los costos que el servicio ofrecido represente para dicha entidad. Entre otras, se puede mencionar a la banca telefónica.

De esta manera, se plantea utilizar la opción IVR para enrutar una llamada a una base de datos, sin la necesidad de intervención humana, reduciendo el tiempo de espera, en línea, de sus clientes.

2.3 Pregunta de Investigación

¿Cómo se realiza la conexión del IVR con una base de datos a través de AGI?

2.4 Resumen

El trabajo consistirá en la implementación de un IVR en Elastix que haga consultas a una base de datos mediante la programación de scripts en lenguaje PHP, de modo que se obtenga la información requerida por el cliente.

Para esta interacción se utilizará AGI (Asterisk Gateway Interface), la que permite el uso de lenguajes de programación externos como lo son: Perl, PHP, C/C++, Pascal, Bourne Shell y Pyton.

2.5 Marco Teórico

El propósito de un sistema de respuesta de voz interactiva (IVR) es el de obtener información por parte del usuario mediante una acción basada en la entrada de datos que puede ser mediante voz (ASR) o tonos de marcado (DTMF)); comúnmente se realiza la búsqueda de datos en un sistema externo, como una base de datos, que devuelve el resultado a la persona que llama.

Tradicionalmente los sistemas IVR son costosos y difíciles de implementar. Elastix cambia todo eso ya que la distribución de este servidor de comunicaciones es gratuita y la creación de los planes de marcado se realiza mediante una interfaz gráfica.

Los elementos básicos de una IVR son muy similares a los de una operadora automática, aunque el objetivo es diferente. Con este sistema se necesita por lo menos una acción de entrada para saber lo que la persona que llama espera del IVR.

Lo más importante en un IVR, además de la correcta elección de una plataforma robusta y confiable, es el diseño del script. Con esto se hace referencia a que en el script están ubicados los mensajes que el IVR reproducirá cuando se realice una llamada a la extensión configurada. Este script debe ser práctico, entendible, fácil de usar y eficiente, de tal manera que los clientes prefieran usarlo en vez de comunicarse con un agente, permitiendo así a la empresa generar una mejor experiencia con el usuario.

Los lugares más comunes donde nos podemos encontrar estos sistemas son las empresas grandes como por ejemplo el área bancaria, donde el caso más común es la consulta de saldos. La empresa asigna un número de PIN con el que el usuario ingresa a través de su teléfono y posteriormente le reproduce el saldo automáticamente sin importar la hora o el día de la semana.

Para realizar todas estas operaciones es necesario el uso de lo que se denomina Dial-Plan (Plan de Marcado). "El Dial-Plan es el verdadero corazón de Asterisk y de cualquier sistema VoIP. El plan de marcado es una colección ordenada de acciones que se ejecutan cuando alguien marca un número dentro de nuestro Asterisk. El ejemplo más trivial sería que cuando alguien marca la extensión de otra persona, por ejemplo "3001", suene el teléfono de ese usuario. Sin embargo, se pueden hacer cosas mucho más avanzadas, como por ejemplo gestionar las llamadas en función de un horario, crear una centralita automática de recepción de llamadas, grabar conversaciones, poner música en espera, etc." (Axelko)

Mediante este plan de marcado se puede redirigir la llamada a un script desarrollado en un lenguaje de programación el que podrá contener: menús, consultas a bases de datos o información de cualquier tipo, dependiendo de lo que se programe.

AGI (Asterisk Gateway Interface) es la forma que Asterisk permite a los programadores de software crear aplicaciones que se usan a lo largo del plan de marcado. A través de AGI se puede leer las entradas del usuario mediante voz o tonos de marcado, reproducir archivos de sonido, controlar las llamadas y su flujo, y casi todo lo necesario para realizar una IVR (Interactive Voice Response) en prácticamente cualquier lenguaje de programación no nativo de Asterisk.

En nuestro medio son muy pocas empresas que usan AGI (Asterisk Gateway Interface) en sus servidores Asterisk debido a la poca información que existe y la falta de conocimiento, perdiendo así la posibilidad de explotar nuestro IVR (Interactive Voice Response) al máximo.

(Landívar)

2.6 Objetivo General

- Implementar un IVR (Interactive Voice Response) con el servicio de consulta automatizada para una base de datos de cartera a través de librerías AGI (Asterisk Gateway Interface).

2.7 Objetivos Específicos

- Realizar la instalación de componentes y configuraciones en el servidor Asterisk para el correcto funcionamiento del sistema de consultas de cartera.
- Implementar un IVR que re-direccione las llamadas.
- Valorar la librería PHP-AGI y demostrar el funcionamiento de AGI (Asterisk Gateway Interface).
- Desarrollar un script en lenguaje PHP y construir una base de datos de cartera.
- Adaptar un conversor de texto a voz para presentar los resultados de las consultas a los usuarios.
- Establecer los casos de uso del sistema.

2.8 Metodología

La metodología que se usará se divide en 3 fases, cada una con sus actividades.

La primera fase es la de investigación en la que se realizarán: revisiones bibliográficas, se sintetizarán conceptos y se estudiarán las herramientas a utilizar.

La segunda fase es la de implementación, la que está conformada por las configuraciones básicas de Asterisk, la instalación de todos los componentes necesarios, el desarrollo del IVR (Interactive Voice Response), el desarrollo del script en el lenguaje de programación PHP y por último se pondrán en práctica todos los temas anteriormente investigados para unir el script con el IVR mediante el uso de la librería AGI (Asterisk Gateway Interface).

La última fase es la final, en la que se realizarán pruebas en el sistema usando softphones (teléfonos implementados por software) y se analizará el funcionamiento del IVR mediante la consola de Asterisk.

2.9 Alcances y Resultados esperados

El presente trabajo busca diseñar y desarrollar un IVR para un sistema de consultas de una base de datos de cartera de cualquier empresa a través de un servidor Asterisk.

El desarrollo del sistema partirá desde un servidor Asterisk ya creado.

La programación del script se desarrollará en el lenguaje PHP el que interactuará con Mysql que es motor de base de datos por defecto que trae Asterisk.

Para verificar el funcionamiento del IVR se realizarán pruebas con un softphone.

En el sistema IVR que se pretende implementar solo ejecutará consultas de la base de datos a través de un softphone, no realizará ningún otro tipo de operación.

2.10 Supuestos y Riegos

Supuestos:

- Contar con todo el material bibliográfico propuesto para el desarrollo del trabajo de investigación o el necesario.
- Obtener artículos y estudios previamente realizados de los casos propuestos.
- Contar con cronograma de actividades.

Riegos:

- El tiempo estimado para el desarrollo no sea el suficiente.
- No encontrar el material bibliográfico necesario para el desarrollo.
- No tener la orientación adecuada por parte del Director y no avanzar en el trabajo.
- Pérdida de material digital por causas de cualquier fenómeno natural.

2.11 Presupuesto

Rubro-Denominación	Costo USD	Justificación
Útiles de Oficina	$150	- Hojas de Papel - Material de Escritorio - Copias - Empastado - Varios
Derechos Universitarios	$300	- Derecho de Tesis - Otros
Internet	$150	- Pago mensual al proveedor de internet hasta la finalización de la tesis.
Imprevistos	$100	- En caso de que se presente un gasto no previsto
Material Bibliográfico	$50	- Compra de libros.
Total	**$750**	

2.12 Financiamiento

Propio: 100%

2.13 Esquema Tentativo

1. **Capítulo 1: Introducción y Definiciones**
   - 1.1. Introducción
   - 1.2. Asterisk
     - 1.2.1. Definición
     - 1.2.2. Motivación de la Investigación
     - 1.2.3. Problemática
     - 1.2.4. Resumen
     - 1.2.5. Objetivos
       - 1.2.5.1. Objetivo General
       - 1.2.5.2. Objetivos Específicos
     - 1.2.6. Metodología
     - 1.2.7. Alcance y resultados esperados
     - 1.2.8. Supuestos y riesgos
2. **Capítulo 2: IVR**
   - 2.1. Interactive Voice Responce – IVR
   - 2.2. Arquitectura
     - 2.2.1. IVR fuera de la PBX
     - 2.2.2. IVR dentro de la PBX
   - 2.3. Servicios
     - 2.3.1. Call Center
     - 2.3.2. Data Base Management System –DBMS
   - 2.4. Tecnologías Usadas
     - 2.4.1. DTMF (Dual Tone Multi Frecuency)
     - 2.4.2. TTS (Text to Speech)
     - 2.4.3. ASR (Automatic Speech Recognition)
     - 2.4.4. PBX (Private Branch Exchange)
   - 2.5. Aplicaciones de la IVR
     - 2.5.1. Entrada de datos
     - 2.5.2. Salida de datos
   - 2.6. Conclusiones
3. **Capítulo 3: Plan de Marcado (Dial-Plan)**
   - 3.1. Contextos
   - 3.2. Extensiones
   - 3.3. Prioridades
     - 3.3.1. Prioridades Numeradas
     - 3.3.2. El 'same=>' operador
   - 3.4. Etiquetas de prioridad
   - 3.5. Aplicaciones

3.5.1.Aplicaciones: 'Answer()', 'PlayBack()' y 'Hangup()'
3.6. Includes
3.7. Conclusiones
4. **Capítulo 4: AGI (Asterisks Gateway Interface)**
4.1. Introducción
4.2. Variantes AGI
4.2.1.Procesos basados en AGI
4.2.2.FAST AGI (AGI sobre TCP)
4.2.3.ASYNC AGI
4.3. Descripción general de comunicación AGI
4.3.1.Configuración de una sesión AGI
4.3.2.Comandos y respuestas
4.3.3.Terminar una sesión AGI
4.3.4.Estructuras de desarrollo
4.4. PHP-AGI
4.5. Conclusiones
5. **Capítulo 5: Integración de MYSQL**
5.1. Instalación y configuración de Mysql
5.1.1.Instalación de Mysql para Centos
5.1.2.Configurar Mysql
5.2. Instalación y configuración de ODBC
5.2.1.Configurar ODBC para Mysql
5.3. Gestión de bases de datos
5.3.1.Instalación de PHPMyAdmin
5.4. Conclusiones
6. **Capítulo 6: Diseño**
6.1. UML
7. **Capítulo 7: Programación**
7.1. Configuración de PHP-AGI
7.2. Instalación y configuración del TTS (Text to Speech)
7.3. Configuración de las extensiones
7.4. Creación y configuración del Script PHP
7.5. Configuración un Softphone
8. **Capítulo 8: Funcionamiento y Pruebas**
9. **Capítulo 9: Conclusiones y Recomendaciones**
9.1. Conclusiones
9.2. Recomendaciones
9.3. Trabajo a futuro

2.14 CRONOGRAMA

Objetivo Específico	Actividad	Resultado Esperado	Tiempo(Semanas)
Realizar la instalación de	-Instalación PHP -Instalación del soporte y	-Tener instalado todo lo necesario	3

componentes y configuraciones en el servidor Asterisk para el correcto funcionamiento del sistema de consultas de cartera.	dependencias para la base de datos Mysql -Instalación de PHPMyAdmin -Configuraciones de los componentes instalados	para el correcto funcionamiento del sistema de consultas	
Implementar un IVR que re-direccione las llamadas.	-Realizar las diferentes configuraciones en el archivo que se encuentra en la ruta: /etc/asterisk /extensions_custom.conf -Asignar en el plan de marcado el número de la extensión que se conectará con el AGI	-El momento que el usuario marque el número de extensión asignado se realice una llamada al archivo PHP	4
Valorar la librería PHP-AGI y demostrar el funcionamiento de AGI (Asterisk Gateway Interface).	-Descargar la clase php-AGI y ubicarla correctamente donde corresponde para que se pueda usar el archivo .PHP	-Que se logre conectar el plan de marcado con la librería AGI	3
Desarrollar un script en lenguaje PHP y construir una base de datos de cartera.	-Tener los conocimientos para programar en el lenguaje PHP y así poder conectar nuestro IVR con una base de datos Mysql	-El resultado de la consulta a la base de datos debe ser el correcto.	5
Adaptar un conversor de texto a voz para presentar los resultados de las consultas a los usuarios.	-Elegir un TTS (Text to Speech) que se adapte a Elastix. -Investigar todas las opciones que nos ofrece el TTS y adaptarlas al	-El usuario una vez que realice su consulta los resultados se le presentarán de forma auditiva.	5

	código PHP.		
Establecer los casos de uso del sistema.	-Tener un plan para cada caso que pueda ocurrir el momento que el usuario realice una llamada a la extensión configurada.	-Con este plan se conocerá a fondo todas las opciones que ofrece el sistema de consultas.	3

2.15 Referencias

2.15.1 Referencias Bibliográficas

Kouhfallah, Haamed. Easy Elastix. 2012.

Landívar, Edgar. Comunicaciones Unificadas con Elastix. 2008.

M., Andrés Junge. Asterisk PBX - Guia de Configuracion . Rio de Janeiro: Titulo Independiente, 2007.

Muñoz, Alfio. Elastix al ritmo del merengue. Republica Dominicana , 2009-2010.

Purdy., Kevin. «"The complete Android Guide" » 1. USA, 2009.

Sharif, Ben. Elastix without tears. Malaysia, 2008.

Unificadas, Comunicaciones. Comunicaciones Unificadas con Elastix - Volumen 2. 2008 - 2009.

2.15.2 Referencias Electrónicas

Auronix. Mejores practicas para el diseño de IVR. 2014. 10 de 05 de 2014 <www.auronotix.com/Home/mejores-practicas-para-el-diseño-de-ivrs>.

Axelko. Curso Asterisk (IV): El Dialplan. 11 de Noviembre de 2013. 09 de 05 de 2014 <http://www.axelko.com/techblog/2013/11/curso-asterisk-iv-el-dialplan/>.

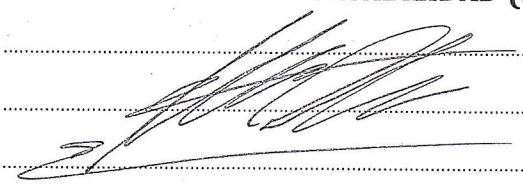
Wikiasterisk. Introducción Dialplan. 26 de Mayo de 2012. 08 de 05 de 2014 <http://www.wikiasterisk.com/index.php/Introducci%C3%B3n_Dialplan>.

Voip-info. Asterisk cmd Festival. 10 de 05 de 2012. 10 de 05 de 2014 <http://www.voip-info.org/wiki/view/Asterisk+cmd+Festival>.

UNIVERSIDAD DEL AZUAY

2.16 Firma de Responsabilidad (Estudiante)

2.17 Firma de Responsabilidad (Director Sugerido)



2.18 Fecha de Entrega

17/06/2014

RECIBIDO 26 JUN 2014