Diseño e Implementación de un Sistema Telefónico para la Lectura de Repositorios Digitales de Información con la Finalidad de Prestar el Servicio a Personas con Discapacidades Visuales

Ma. Fernanda Molina M.[(1)], Luis Sánchez Loor[(2)], Gabriel Astudillo[(3)]
Facultad de Ingeniería en Electricidad y Computación
Escuela Superior Politécnica del Litoral (ESPOL)
Campus Gustavo Galindo, Km 30.5 vía Perimetral
Apartado 09-01-5863. Guayaquil-Ecuador
mmolina@fiec.espol.edu.ec[(1)], lesanche@fiec.espol.edu.ec[(2)]
Escuela Superior Politécnica del Litoral (ESPOL)[(3)], Ingeniero en Electrónica y Telecomunicaciones[(3)],
gastudil@fiec.espol.edu.ec[(3)]

Resumen

El objetivo principal del presente proyecto fue diseñar e implementar un IVR basado en Asterisk, el cual junto con otras tecnologías fue capaz de leer repositorios digitales como fueron las páginas web de periódicos, diccionarios, la wikipedia, blogs y libros.

El sistema inicia recogiendo los datos de páginas web estandarizadas para generar archivos con un formato que permite ser leído por otra herramienta que es capaz de generar archivos de voz, los cuales fueron reproducidos y así el usuario escuchará la información, además de la recepción por medio de comandos de voz para la elección entre los cinco repositorios digitales disponibles.

La central telefónica Asterisk permite la comunicación entre las diferentes aplicaciones como son Festival TTS, VXI y Sphinx mediante scripts en lenguaje PHP y Perl. Esta herramienta puede ser adquirida por un bajo costo, es de fácil instalación y mantenimiento, por lo cual podría ser implementada sin problema por alguna institución del gobierno y así las personas obtendrían un nuevo medio de información

Palabras Claves: *IVR, Asterisk, repositorio digital, discapacitado*

Abstract

The main objective of this project is to design and implement an IVR based on Asterisk, which along with other technologies is able to read digital repositories, such as websites of newspaper, dictionaries, wikipedia, blogs and books.

The system starts by collecting data from standardized web pages to generate files with a format which can be read by another tool that can generate voice files. The generated voice files are played, thus the user will hear the information, as well as the reception by voice commands for the selection of the five available digital repositories.

The Asterisk PBX enables communication between different applications such as Festival TTS, VXI and Sphinx through PHP and Perl. This tool can be purchased with a low cost and has easy installation and maintenance, for that reason it could be implemented without problem by any government institution, so people would get a new medium.

Keywords: *IVR, Asterisk, digital repositories, handicapped*

1. Introducción

En la actualidad existe un sinnúmero de programas, aplicaciones y dispositivos que las personas usan para permanecer comunicados, pero todos estos conllevan un costo de adquisición y de uso, sin embargo existe un porcentaje alto de personas que no pueden gozar de estos avances por varias razones como por ejemplo, el costo económico o que sufren de discapacidad visual.

Por ello, se ha sacado provecho a varias herramientas para implementar un sistema que será capaz de leer de forma clara y comprensible el contenido de cinco repositorios digitales; con la finalidad de mantener informada e integrada a las personas con discapacidad visual leve, media o severa.

La finalidad de este proyecto es aprovechar las diferentes tecnologías que se encuentran disponibles para el beneficio de la comunidad, ya que este sistema puede ser de gran utilidad para muchas personas pero sobre todo las personas con discapacidad visual que se encuentran limitadas a solo escuchar su entorno y no tienen acceso a internet.

2. Descripción

El proyecto a realizar consiste en la implementación y diseño de un IVR basado en Asterisk junto con otras tecnologías capaz de leer repositorios digitales como las páginas web de periódicos, diccionarios, la wikipedia, blogs y libros.

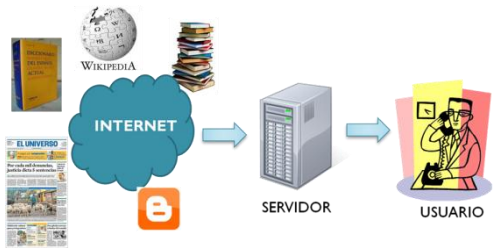

Figura 1. Esquema del sistema

3. Metodología

Los pasos necesarios a seguir para cumplir con el principal objetivo es:

a) Instalar Asterisk sobre un servidor con sistema operativo CentOS Linux sobre el cual se desarrolla el IVR.
b) Desarrollar scripts en PHP que reciben la dirección de diferentes páginas web como son: periódico, blog, wikipedia, diccionario y libros.
c) Traducir en un lenguaje común para poder ser leídas a través del AGI. De esta forma, el usuario obtiene la información deseada.

4. Asterisk

Asterisk es un software libre que proporciona funcionalidades de una central telefónica IP (IPBX) conectada directamente a la red pública de teléfono por medio de líneas troncales. Mark Spencer de Digium, desarrolló este sistema que se ha caracterizado por su flexibilidad y además con otros programadores han contribuido en añadir funcionalidades como buzones de voz, conferencias, IVR (Respuesta Automática Interactiva), distribución automática de llamadas, directorios, grabación de llamadas y muchas otras más de manera gratuita y consumiendo pocos recursos de hardware.

4.1 Arquitectura de Asterisk

4.1.1 Plan de Marcado

El plan de marcado o dialplan, como se lo conoce en inglés, es el corazón de Asterisk donde reside la lógica de la central telefónica (PBX). Dado que es único y se configura en el archivo /etc/asterisk/extensions.conf, debe seguir una lista de instrucciones que Asterisk emplea como respuesta a peticiones externas.

4.1.2 IVR

La Respuesta de Voz Interactiva de Asterisk, es un conjunto de aplicaciones, contextos y elementos de un plan de marcado. Es un sistema automatizado para la interacción de llamadas entrantes, donde existen grabaciones de voz y reconocimiento de respuestas simples como SI, NO y otros.

4.1.3 AGI

La Interfaz de Puerta de Enlace, agrega funcionalidades a Asterisk mediante diferentes tipos de lenguajes de programación, como Perl, PHP, C, Pascal. Se comunica con Asterisk mediante comandos que controlan las peticiones y respuestas del script.

5. Componentes del Sistema

5.1 Hardware

Una de las ventajas del sistema es que requiere de tan solo un componente físico, el servidor, el cual debe tener las características necesarias para receptar, procesar y enviar los datos del registro telefónico.

5.1.1 Servidor

Las características mínimas requeridas para el buen funcionamiento del servidor están detalladas en la tabla 1.

Tabla 1. Características Mínimas del Servidor

CPU	Pentium III 2Ghz
MEMORIA RAM	1.5 GB
TARJETA DE RED	10/100Mbps
DISCO DURO	20 GB

De acuerdo al libro Asterisk, The Future of Telephony[1], estas características deberán ser más robustas y con mayor velocidad dependiendo de la finalidad que el usuario le vaya a dar al sistema. Como se muestra en la tabla 2.

Tabla 2. Guía para requerimientos del sistema

Propósito	**Número de Canales**	**Mínimo Recomendado**
Sistema para ocio	No más de 5 canales	400MHz x86, 256MB RAM
Sistema SOHO	De 5 a 10	1 GHz x86, 523MB RAM
Sistema para pequeña empresa	Hasta 25	3 GHz x86, 1GB RAM
Sistema para mediana y grande	Más de 25	Procesador dual, posiblemente múltiples servidores en una arquitectura distribuida

5.2 Software

La finalidad del presente proyecto es conocer y aplicar los beneficios que ofrece el software libre, además de disminuir los costos de implementación. Las aplicaciones que se detallan en la tabla 3 fueron elegidas de acuerdo al nivel de compatibilidad con Asterisk, funcionalidades y con licencia gratuita.

Tabla 3. Aplicaciones del sistema

Plataforma	Linux
Distribución	Centos 5
Servidor Web	Apache
Software IP PBX	Asterisk versión 1.8.4.2
Aplicación Text-to-Speech	Festival TTS
Estándar Voice XML	OpenVXI
Reconocedor de voz	Sphinx

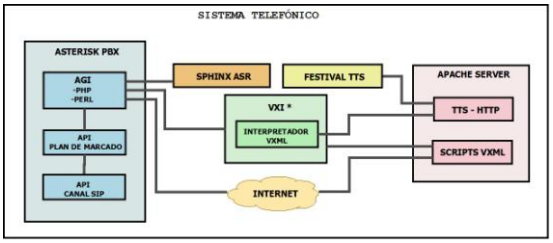

Figura 2. Sistema Telefónico

5.2.1 Servidor Web - Apache

Apache es el servidor web HTTP de código abierto multiplataforma más usado para enviar páginas web estáticas y dinámicas en aplicaciones web, que son desarrolladas sobre todo en PHP, el cual viene a ser el lenguaje que se implementó en el sistema para desarrollar los scripts que recogen y administran la información desde las páginas web.

5.2.2 VOICEXML (Voice Extensible Markup Language version 2.0, Lenguaje de Marcado Extensible de Voz versión 2.0)

VXML es un formato estándar XML de la W3C diseñado para interactuar entre el humano y la computadora creando diálogos de voz mediante un navegador de voz o por teléfono.

Entre sus principales características están: sintetizador de voz, audio digitalizado, reconocimiento de voz, registro de llamadas entrantes y entrada DTMF (marcación por tonos).

Las grandes ventajas del desarrollo de aplicaciones con VXML son: una implementación flexible, el re-uso de infraestructura web existente, plataforma independiente y la existencia de productos disponibles.

5.2.2.1 Arquitectura del VoiceXML

El modelo de aplicación distribuido del navegador VXI por Asterisk asumido por VoiceXML es un sistema basado en tres niveles con la siguiente arquitectura:

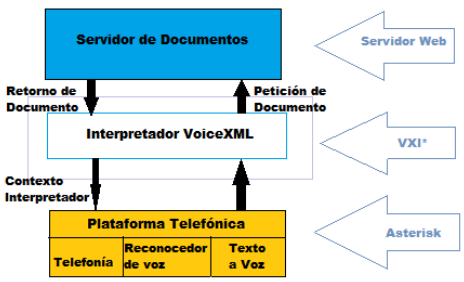

Figura 3. Arquitectura del VoiceXML

Cuando una llamada es recibida, la plataforma telefónica, Asterisk, envía un evento al interpretador VoiceXML, el cual a su vez se ve en el contexto de la URL del documento inicial a buscar. Una petición es enviada al servidor de documentos para el documento inicial. El servidor de documentos a la vez envía el documento al interpretador VoiceXML que interpreta el documento y lo ejecuta usando los servicios de la plataforma telefónica.

La interpretación de acuerdo a la aplicación puede resultar de varias formas: enviando un mensaje al usuario, esperando una entrada por parte del usuario, grabando una voz, entre otros.

5.2.3 Navegador VXI

Entre los productos que ofrece la compañía europea i6net soluciones y tecnologías está el navegador VXI* VoiceXML para Asterisk[2], este software corre sobre VoIP para crear nuestro propio IVR (Respuesta de Voz Interactiva) o plataforma IVVR (Respuesta de Voz y Video Interactiva) a través de IP, PSTN y redes 3G-324M.

OpenVXI es un kit de herramientas interpretador de VoiceXML portátil de código abierto, es un componente del navegador VXI que provee APIs para servicios de plataforma como reconocimiento de voz, síntesis de habla y servicios de telefonía.

VXI intérprete, elemento del navegador VXI, trabaja directamente con el software PBX Asterisk con el apoyo de Digium y está escrito en lenguaje C sobre el sistema operativo Linux. La licencia gratuita ofrecida para el usuario final solo permite una sesión a la vez, es decir, solo una llamada puede ser atendida; para fines comerciales es necesario tener la licencia comercial que ya tiene un costo.

5.2.3.1 Características

- Solución basada en software, empaquetado 100%.
- Disponible para varias distribuciones de Linux OS en 32 o 64 bits aplicaciones interoperables simultánea de voz y de video.
- Soporta 3G-324M que incluye H.263, H.263, H.264.
- VoiceXML 2.0 compatible con etiquetas extendidas añadidas recientemente.
- Transcodificación de video y adaptación de tasa en tiempo real.
- Complementos disponibles para extender su plataforma VXI*.

5.2.4 Festival TTS

Es el sistema sintetizador de voz multilingüístico y fue desarrollado por CSTR (Centro de Investigación de Tecnologías del Lenguaje)[3], ofrece un completo sistema de conversión de texto a voz mediante API's que se integran a Asterisk.

Festival está escrito en lenguaje C++ y está implementado como un intérprete de comandos para un control general sobre software libre con licencia MIT-X11 para usar el código fuente sin restricciones.

Se ha implementado Festival en idioma español y se han agregado módulos que permiten obtener una voz con un léxico más claro.

5.2.5 SPHINX

Sphinx es un reconocedor de voz que se integra con Asterisk, cuenta con un gran vocabulario en inglés y trabaja bajo la licencia estilo de *Berkele*[4].

También es una colección de herramientas de código abierto y recursos que permite a los desarrolladores construir sistemas de reconocimiento de voz. Trabaja en dos módulos, cliente y servidor, los cuales fueron modificados para ser implementados en nuestro proyecto.

5.2.6 Expresiones Regulares

Se les conoce como patrones, son textos especiales conformados por una serie de signos y caracteres especiales que permiten realizar búsquedas y reemplazar porciones de texto específicos dentro de una cadena de caracteres más grande.

Las expresiones regulares son la herramienta clave para obtener la información desde las páginas web, recogiendo los datos dentro de documentos HTML y XML. Muchos lenguajes de programación admiten el uso de expresiones regulares para la manipulación de textos.

6. Configuraciones

6.1 Configuración de Asterisk

Los archivos de configuración existentes en la carpeta **/etc/asterisk/** pueden ser modificados con el propósito de cumplir las exigencias de la implementación de una central telefónica. Entre ellos, nos compete los siguientes: sip.conf y extensions.conf a los cuales creamos copias de respaldo. Al finalizar la realización de cambios, para que tomen efecto se ejecutan en la consola de Asterisk los siguientes comandos:

CLI> module reload chan_sip.so

CLI> dialplan reload

6.1.1 Configuración de SIP.CONF

En el archivo sip.conf se configuran las opciones y parámetros por defecto de canal para los usuarios o peers registrados en el archivo. Además se configuran de manera particular los parámetros para cada usuario, a continuación se muestra un ejemplo.

```
[general]
context=noautenticadas
allowguest=no
srvlookup=yes
udpbindaddr=0.0.0.0
tcpenable=no  qualify=yes
language=es
[userX]
type=friend
context=internos
host=dynamic
nat=yes
secret=1234
dtmfmode=auto
disallow=all
allow=ulaw
```

Figura 4. Configuración del archivo sip.conf

6.1.2 Configuración de EXTENSIONS.CONF

También conocido como el plan de marcado se configuran los contextos y extensiones, los cuales definen el comportamiento de la central telefónica.

```
[general]
autofallthrough=yes
clearglobalvars=no
static=yes
priorityjumping=no

[internos]
exten => 1,1,Answer()
exten => 1,n,Playback(bienvenida)
exten => 1,n,AGI(welcome.agi)
exten => 1,n,WaitExten()
[periodico]
exten=>start,1,AGI(eluniverso.agi,http://www.eluniverso.com/rss/portada.xml)
[diccionario]
exten => start,1,Answer()
same => n,AGI(decode.agi)
[blog]
exten=>start,1,AGI(blogsalud.agi,http://www.saludynutricion.es/feed/)
[libro]
exten => start,1,Answer()
same => n,AGI(decodeLetra.agi)
same => n,Hangup
[wikipedia]
exten => start,1,Answer()
same => n,AGI(decodewiki.agi)
same => n,Hangup
```

Figura 5. Configuración del archivo extensions.conf

6.2 Configuración del Navegador VXI

Los archivos de configuración a modificar para cumplir con la implementación son: client.cfg y defaults.xml los cuales se encuentran en el directorio /etc/openvxi/. De igual manera es recomendable hacer copias de respaldo. Luego de la realización de cambios, para que tomen efecto se reinicia la aplicación con los siguientes comandos:

service openvxi restart

6.2.1 Configuración del archivo CLIENT

Contiene la configuración del cliente del Navegador VXI, donde incluyen parámetros base para la interacción con otros componentes y complementos de la aplicación. Modificaremos la sección del TTS Server, reemplazando la ruta donde se encuentra el recurso encargado de convertir el texto en voz, el archivo tts.php. VXI trabaja con un servidor web para sus componentes, en nuestro caso utilizando Apache la ruta será http://localhost/tts.php.

6.2.2 Configuración del archivo TTS

Es un script creado en PHP, el cual recibe entradas en texto desde el interpretador de VXI y retorna salidas en audio. Este archivo originalmente no viene implementado para trabajar con Festival TTS, pero está basado en script de la documentación de VXI, para una versión más ligera de Festival como es Flite (Festival-light). Partiendo de esto reemplazamos y editamos líneas de código, esencialmente:

$text2wave_cmd = sprintf("text2wave -o %s -scale 50 %s –F 8000", $wave_file, $speech_file);

exec($text2wave_cmd);

El comando text2wave, es una utilidad de Festival TTS la cual crea un archivo de extensión .wav, a partir de un archivo de texto. Adicionalmente, se especifica la frecuencia (-F) y la escala de volumen (-scale).

6.2.3 Configuración del archivo DEFAULTS

Este archivo XML contiene propiedades y opciones por defecto del interpretador de VXML del navegador VXI. Estas propiedades pueden ser sobrescritas particularmente en cada archivo vxml. Se modificaron los mensajes predeterminados de información y ayuda, así como los mensajes de error adicionando como atributo **xml:lang="es".**

6.3 Configuración de Festival y Asterisk

Se verifica que se instaló correctamente, accediendo a Asterisk y ejecutando los siguientes comandos:

CLI> module unload app_festival

CLI> module load app_festival

Se modifica el archivo festival.conf del directorio /etc/asterisk/ agregando los siguientes parámetros para activar Asterisk con festival. Se recargan los módulos una vez hecho los cambios.

[general]
host=localhost
port=1314
usecache=yes
cachedir=/var/lib/asterisk/festivalcache/
festivalcommand=(tts_textasterisk "%s" 'file)(quit)\n

6.4 Configuración de Sphinx

Los componentes a configurar son los archivos sr_server.dat y los archivos del diccionario de palabras a reconocer por Sphinx. Luego se verifica la respuesta de interpretación del servidor.

6.4.1 Configuración de SR_SERVER

Es el archivo servidor de Sphinx, encargado de reconocer los archivos de audio enviados como parámetros, retornando la palabra interpretada siempre que se encuentre en el diccionario. Se encuentra implementado en Perl, pero produce error de compilación por el encabezado. Por lo que se lo editó reemplazando con:

 # !/bin/sh
exec perl -w -x $0
 # !perl

Luego de guardar los cambios, se inicia el siguiente script para evitar errores al cargar el server, y dejamos corriendo el sr_server en espera de algún requerimiento.

 # export LD_LIBRARY_PATH=/usr/local/sphinx/lib
#/usr/local/bin/sr_server &

6.4.2 Configuración del Diccionario

Se necesitan una serie de archivos para el reconocimiento del audio. Estos se generan con una aplicación web conocida como lmtools. Partiendo por la creación de un archivo de texto conteniendo las palabras deseadas en inglés. Para nuestra implementación serán: OK, SEE, NO, separadas por un salto de línea.

La aplicación genera una carpeta con los archivos de extensión .dic, .log_pronounce, .lm, .sent y .vocab a cuales se les cambiará en nombre de la carpeta, así como el nombre de los archivos dentro de ella, por "confirm".

La carpeta confirm deberá ser movida a /usr/local/sphinx/share/sphinx2/model/lm/ de donde el server de Sphinx usa los archivos para el reconocimiento del habla.

6.4.3 Verificación de Reconocimiento

Se verifica que el sr_server funciona corriendo el archivo sr_client con archivos de audio wav como parámetro, y observando la respuesta del server. Los archivos de audio deberán tener grabado palabras pertenecientes al diccionario, y usamos el comando:

$ /usr/local/bin/sr_client OK.wav
$ /usr/local/bin/sr_client NO.wav

En el sr_server deberá respectivamente aparecer:

SERVER RESULT: OK
SERVER RESULT: NO

7. Pruebas y Funcionamiento

Ingresar al terminal para inicializar los servicios necesarios:

1. service httpd restart
2. service openvxi restart
3. /usr/bin/festival –server
4. export LD_LIBRARY_PATH=/usr/local/sphinx/lib
5. /usr/local/bin/sr_server &
6. Asterisk –r

La reproducción de los repositorios digitales disponibles se inicia al marcar el número de prueba 1800-12345-6789, continuación en la figura 7 se visualiza la ejecución del script welcome.agi que espera un comando de voz por parte del usuario para acceder a una de las siguientes opciones:

1. Periódico
2. Diccionario
3. Wikipedia
4. Blog
5. Libros

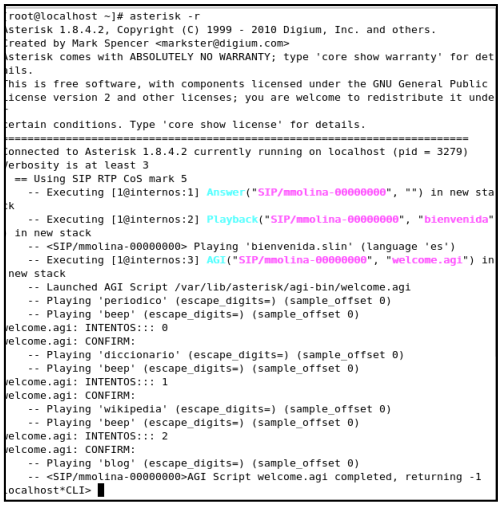

Figura 6. Ejecución del script welcome.agi

Por motivo que las funcionalidades de las opciones son similares se ha elegido la opción del diccionario ya que es una de las más complejas.

Se ingresa al diccionario de la Real Academia Española consultando su página web http://www.rae.es/rae.html.

Primero se ejecuta el script decode.agi donde el usuario debe ingresar mediante el teclado numérico la palabra a buscar en el diccionario, como por ejemplo, para consultar la palabra "tierno", debe teclear los siguientes números:

T = una vez el número 8

I = tres veces el número 4

E = dos veces el número 3

R = tres veces el número 7

N = dos veces el número 6

0 = número cero para distinguir las letras que se encuentran en el mismo número.

O = tres veces el número 6

* = para finalizar la palabra

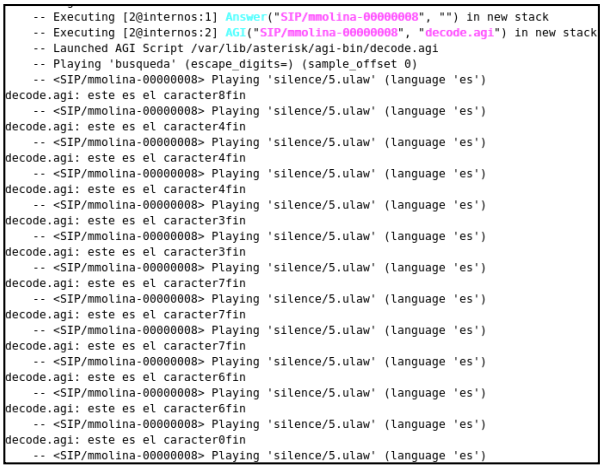

Figura 7. Ingreso de la palabra a buscar

Luego de terminar el ingreso, se pedirá la confirmación de su entrada para ejecutar el script diccionario.agi caso contrario el sistema le permitirá el ingreso de una nueva palabra. En la figura 8 se aprecia la palabra ingresada.

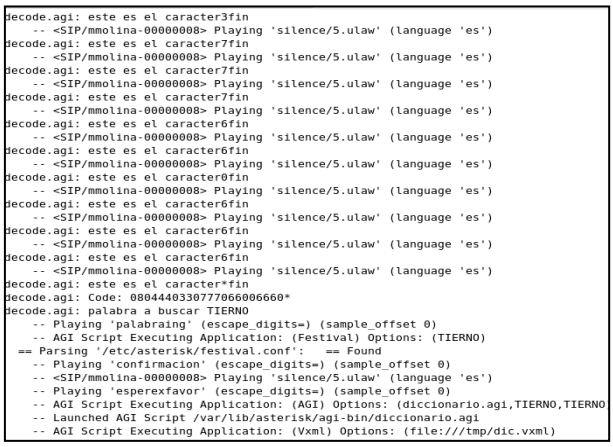

Figura 8. Ejecución de script diccionario.agi

El script diccionario.agi generará un archivo xml llamado diccionario.xml el contenido de la página web que contiene el o los significados de la palabra que consulta como se puede ver en la figura 9.

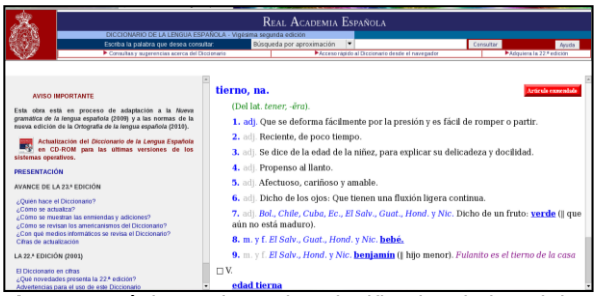

Figura 9. Página web con los significados de la palabra

Como se puede apreciar en el archivo diccionario.xml sólo se encontraron 6 definiciones porque el sistema no leerá las definiciones que contengan vínculos hacia otras páginas.

Para salir del diccionario el usuario ingresará la opción cero cuando desee y volverá al script principal como se puede apreciar en la figura 10.

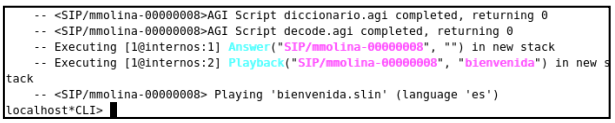

Figura 10. Finaliza el diccionario

8. Conclusiones

- Ha sido posible la implementación de un sistema IVR con acceso a repositorios web basado en tecnologías libres, con un bajo costo de inversión e implementación.
- Mediante el estándar VoiceXML fue posible desarrollar varios diálogos de voz interactivos de manera automática y personalizada.
- Mediante los scripts AGI escritos en Perl y PHP se ha logrado comprobar que con Asterisk es posible integrarlo con varias aplicaciones.
- Las expresiones regulares son una herramienta fundamental para acceder a información de texto que se encuentra en cualquier archivo de origen, tal como html y xml.
- La presencia de un servidor web local como Apache, dio soporte al interpretador de VXI, posibilitando la respuesta de voz y la interacción entre distintos vxml.

9. Recomendaciones

- Se recomienda trabajar con páginas web estandarizadas porque de esta manera se facilita el acceso a los datos y se obtiene una menor probabilidad de errores, debido a que en el transcurso del desarrollo del proyecto se tuvo que hacer cambios de sitios web por este motivo.
- Implementar una aplicación de reconocimiento de voz con lenguaje español en vez de uno en inglés para proveer del servicio en su totalidad y no de manera parcial como ocurre en el proyecto, debido a que no se encontró una aplicación de reconocimiento de voz en lenguaje español con licencia gratuita.
- Desarrollar un sistema de reconocimiento de voz en la universidad en lenguaje español para que pueda ser implementado por los estudiantes en sus proyectos de manera gratuita.
- A partir de este proyecto es recomendable integrar una base de datos para almacenar los sitios más visitados por cada usuario para garantizar mayor rapidez en la ejecución del sistema.
- Para mejorar el manejo de la memoria caché es recomendable almacenar los archivos de audio generados con mayor frecuencia.

10. Referencias

[1] Van Meggelen J., Madsen L., Smith J., *Asterisk: The Future of Telephony, O´Reilly*, fecha de consulta junio 2011

[2] I6net*, *VXI* VoiceXML Browser*, http://www.i6net.com/products/vxi/, fecha de consulta julio 2011

[3] Wikipedia, *Festival TTS*, http://es.wikipedia.org/wiki/Festival_(TTS), fecha de consulta julio 2011

[4] Voip-info.org, *Descripción de Sphinx*, http://www.voip-info.org/wiki/view/Sphinx, fecha de consulta julio 2011