Desarrollo de interfaces de voz para aplicaciones web

J. Quiané-Ruiz [1], J. Manjarrez-Sánchez [2]

Centro de Investigación en Computación
Instituto Politécnico Nacional
Unidad Profesional "Adolfo López Mateos" Zacatenco
Col. Linda Vista, C.P. 07738
México D.F.
ufo@correo.cic.ipn.mx [1], jorgerms@cic.ipn.mx [2]

Resumen. La gran popularidad de los teléfonos celulares digitales y de los asistentes personales digitales (PDA), han impactado en la forma de desarrollar aplicaciones sobre Internet, por lo cual se están demandando más procedimientos sobre Internet. Pero el problema de estos dispositivos es su pequeña capacidad de visualización. Las aplicaciones basadas en Web de hoy en día están diseñadas para equipos de escritorio, lo cual las hace difícil de visualizarse en los dispositivos ya mencionados. Este artículo presenta una solución a este problema al desarrollar aplicaciones web basadas en VoiceXML, y así poder acceder a nuestra información desde cualquier lugar utilizando cualquier tipo de teléfono.

1 Introducción

El VoiceXML es un lenguaje para aplicaciones de voz en Internet implementando los servicios de telefonía, y está basado en la norma XML (Extensible Markup Language). Este tipo de aplicaciones son ejecutadas sobre un servidor de voz (intérprete VoiceXML) mediante cualquier tipo de teléfono, y dicho servidor contiene los servicios de telefonía y de Speech, y la comunicación con el al servidor web o de servicios es por medio del protocolo http (la información puede estar almacenada en el mismo servidor de voz o en el servidor web).

En este artículo se explica el desarrollo de aplicaciones VoiceXML utilizando un emulador y enlazando el código de voz con una página jsp.

2 Componentes de Speech

Las aplicaciones VoiceXML interactúan con el usuario por medio de diálogos de voz, lo cual implica servicios de reconocimiento de voz (ASR), de Text-to-Speech (TTS), reproducción de archivos de audio, reconocimiento de tonos de marcado (DTMF), y de telefonía. Y los componentes de Speech son los que le permiten a las

A. Gelbukh, M. Hernández Cruz (Eds.) Avances en la ciencia de la computación en México, CORE-2003, pp. 135–148, 2003.

aplicaciones VoiceXML hacer todo lo anterior. Lo importante de esta tecnología, es de qué separa totalmente a estos componentes de la aplicación, lo cual hace que desarrollador no se preocupe por la programación de los mismos (programación de bajo nivel).

Hay dos tipos de Speech, que son:

Speech input: este se refiere a la entrada de información por parte del usuario, cual puede ser por medio de voz o por tonos de marcado. La información entrante procesada por un reconocedor de voz, para su uso posterior dentro del sistema.

Speech output: en éste se encuentran tres tipos de salida hacia el usuario, el pregrabado, el streaming y el sintetizado. Donde el pregrabado son archivos de audio familiares para los formatos de las computadoras de escritorio, tal como los WAV. El streaming audio es muy similar al pregrabado, a diferencia que en este el audio proporcionado en tiempo real. Y por último el sintetizado, que es por medio de un TTS Engine [1].

3 Intérprete VoiceXML

Éste es el componente más importante en esta tecnología, ya que es el que funciona como servidor de voz, contiene todos los componentes Speech, las aplicaciones VoiceXML y JSP ó ASP. Y es el encargado de ejecutar las aplicaciones VoiceXML, recibir las llamadas telefónicas, de llevar el dialogo con el cliente, de procesar peticiones del cliente, etc.

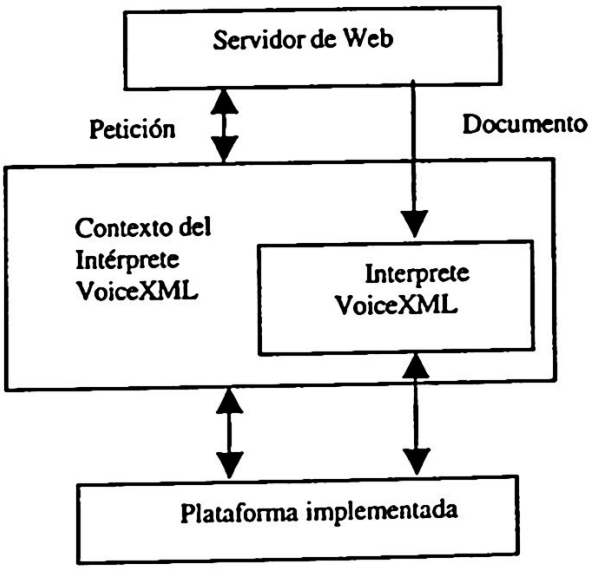

Fig. 1. Esta es la arquitectura presentada por el W3C (World Wide Web Consortium), donde se pueden ver los elementos principales de dicha arquitectura así como la interacción que hay entre ellos.

Donde un servidor del documento (ejemplo: un Web Server) procesa peticiones cliente, a través del intérprete VoiceXML, con el contexto del intérprete de VoiceXML. El servidor elabora los documentos de VoiceXML en la respuesta, que

procesados por el intérprete VoiceXML. El contexto del intérprete de VoiceXML puede supervisar entradas del usuario en paralelo al intérprete VoiceXML. Por ejemplo, un contexto del intérprete de VoiceXML puede esperar a escuchar siempre una frase especial de escape que lleve al usuario a un ayudante personal de alto nivel, y otra puede esperar a escuchar las frases del escape que alteran preferencias del usuario como: volumen ó características Text-to-Speech.

La plataforma implementada es controlada por el contexto del intérprete VoiceXML y por el intérprete VoiceXML. Por ejemplo: en un uso interactivo de la respuesta de la voz, el contexto del intérprete VoiceXML puede ser responsable de detectar una llamada entrante, de adquirir el documento inicial de VoiceXML, y de contestar a la llamada, mientras que el intérprete VoiceXML conduce el diálogo después de la respuesta.

La plataforma implementada genera acontecimientos en respuesta a acciones del usuario (ejemplo: la entrada hablada ó por carácter recibido) y a acontecimientos del sistema (ejemplo: expiración del contador de tiempo). Algunos de estos acontecimientos son actuados sobre y por el intérprete VoiceXML, según lo especificado por el documento de VoiceXML, mientras que otros son actuados sobre y por el contexto del intérprete VoiceXML. [10]

Actualmente, los intérpretes utilizan por lo menos un dispositivo de hardware sofisticado (costoso), lo cual los hace difíciles de adquirir por los desarrolladores individuales, pequeñas y medianas empresas, y por el sector académico. También ofrecen sus servicios sobre la Internet para la realización de pruebas de las aplicaciones VoiceXML, pero dichos servicios son ofrecidos en la unión americana ya que son productos americanos (BeVocal Café, Tellme Studio, VoiceGenie Developer Workshop, por no mencionar todos).

En este caso para poder ejecutar nuestros ejemplos se puede utilizar un emulador para este tipo de aplicaciones, tales como (solo algunos de ellos):

- Cambridge Voice Studio
- WebSphere Voice Server SDK
- Motorola Mobile ADK

4 Introducción a las aplicaciones VoiceXML

En esta sección nos enfocaremos a su desarrollo, viendo la estructura y los el ementos principales de una aplicación VoiceXML, en esta misma sección veremos una pequeña comparación de dos aplicaciones simples, como sería en HTML y como en VoiceXML.

Antes de pasar a ver VoiceXML debemos saber que XML es una norma estándar de marcado para darle una estructura a los documentos de texto en la Internet, en sí podemos decir, que es un metalenguaje para describir otros lenguajes. Una de las características principales de los documentos XML es que deben de ser documentos bien formados sin excepción alguna, para que la información pueda ser interpretada

y transformada a otro lenguaje. XML no sustituye al HTML mas bien es un formato complementario para este tipo de aplicaciones [7].

4.1 Estructura básica de las aplicaciones VoiceXML

Como VoiceXML es un lenguaje que surge a partir de la norma XML, por lo tanto son documentos bien formados, detalle que debemos tener siempre presente al desarrollar este tipo de aplicaciones.

La estructura general de un documento VoiceXML es la siguiente [2]:

```
<? xml version = "1.0" ?>
<vxml version = "2.0">
    [Body]
</vxml>
```

donde la primer etiqueta que va en nuestra aplicación "<? xml versión = "1.0" ?>" es la que nos indica que vamos a trabajar sobre un documento XML y sobre que versión de la misma trabajaremos y esta siempre debe de ir como encabezado de nuestro documento VoiceXML.

La segunda etiqueta "<vxml versión = "2.0">" es la que nos indica que el documento con el cual trabajamos es un documento VoiceXML, y en esta misma etiqueta de igual forma indicamos sobre que versión es la que vamos a trabajar.

Dentro de las etiquetas de "<vxml versión = "2.0">" y "</vxml>" se encontrará todo el cuerpo de nuestro programa.

Como podemos ver la etiqueta "<vxml versión = "2.0">" tiene su respectiva etiqueta de cierre "</vxml>", y podemos ver que la primer etiqueta de nuestra aplicación no tiene su respectiva etiqueta de cierre, y esta será la única etiqueta que no tendrá su etiqueta de cierre en nuestras aplicaciones VoiceXML.

4.2 Desarrollo de una aplicación VoiceXML de ejemplo

En este punto veremos lo principal para el desarrollo de aplicaciones VoiceXML, haciendo la comparación a como sería en HTML la misma aplicación.

Una aplicación en donde en HTML sería como la siguiente:

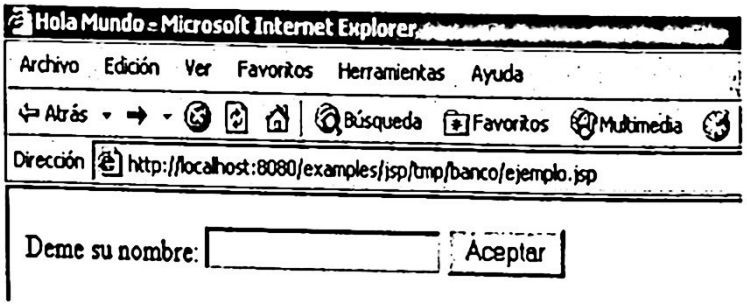

Fig. 2. Esta pantalla es generada al momento de ejecutar nuestra aplicación, y es una aplicación que contiene tres elementos, una etiqueta, un cuadro de texto, y un botón.

En este programa se le pide al usuario su nombre, donde el mismo lo escribe en el cuadro de texto y a terminar de escribirlo se lo indica al programa haciendo un click sobre el botón aceptar.

Y en caso de que usuario haya introducido un nombre nulo se le muestra una pantalla donde se le pide de nuevo su nombre, y al momento de que el usuario introduzca un nombre se le muestra la pantalla siguiente:

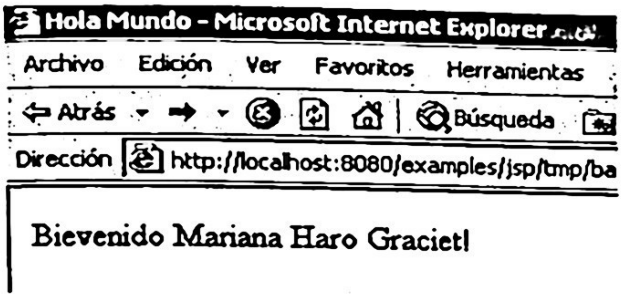

Fig. 3. Mensaje de bienvenida

Esta aplicación presentada, desarrollada en VoiceXML, él código quedaría de la siguiente forma:

```
<!-- Parte 1 -->
<?xml version="1.0"?>
<vxml version="2.0" mode="TTS">
    <form>
    <!-- Parte 2 -->
    <noinput>
          <prompt>
          El nombre no puede ser nulo
          </prompt>
    </noinput>
    <!-- Parte 3 -->
    <field name="nombre" type="spelling">
          <prompt>
          Deme su nombre
          </prompt>
          <!-- Parte 4 -->
          <filled>
                <prompt>
                ¡Bienvenido <value expr="nombre"/>!
                </prompt>
                <exit/>
          </filled>
    </field>
    </form>
</vxml>
```

En la primera parte del código tenemos 3 etiquetas, la primera es la definición de un documento "xml" y definimos la versión de xml a trabajar con el atributo "version"de dicha etiqueta. La segunda etiqueta es la que se utiliza para indicar que tra-

bajaremos con un documento "VoiceXML" (como la etiqueta "html" en HTML), y de igual forma se indica la versión del documento con el atributo "version", y con el atributo "mode" definimos la manera en que el intérprete trabajará con el documento, en nuestro caso será en modo "TTS" que es modo Text-to-Speech. Y la última etiqueta de la primer parte del código es para crear una forma de trabajo en el documento, esta etiqueta es muy parecida a la etiqueta "form" de HTML, y mayormente los documentos VoiceXML son divididos en una o mas formas.

En la segunda parte del código vemos dos nuevas etiquetas, que son "noinput" y "prompt". Donde la primer etiqueta es un evento que se ejecuta únicamente cuando se introduce una cadena vacía como nombre, y la segunda etiqueta es utilizada para indicarle al Parser que el texto que se encuentre entre la etiqueta de abertura y cierra de la misma debe ser reproducido por el TTS Engine, esta etiqueta sería como en HTML escribir en el browser (<h1>¡Hola!</h1>).

La tercera parte contiene una etiqueta nueva, es la "field". Esta nos permite esperar por alguna respuesta por parte del usuario. Dicha respuesta se almacen en una variable que sería la del nombre del campo (field), donde el nombre se le asigna al campo por medio de su atributo "name". Además, podemos definir el tipo del campo, eso se hace por medio el atributo "type". Esta etiqueta viene siendo como la etiqueta "field" de HTML, y esta parte del código sería como el respectivo código en HTML que genera la pantalla mostrada en la Fig. 2.

En la última parte del código se ve la etiqueta "filled", esta etiqueta va ligada a la etiqueta "field" y se ejecuta únicamente cuando se almacena algún valor en la etiqueta "field", viene siendo como el botón aceptar de la aplicación HTML.

Las interfaces de interacción con el usuario de esta aplicación no son visuales como las mostradas por la aplicación HTML, si no, son interfaces auditivas, al ejecutar la aplicación se escucha un dialogo inicial como el que se muestra en la Fig. 4, y dicho diálogo inicial es generado por el código etiquetado como tercera parte. En la figura vemos la transcripción del diálogo, donde *system* es la aplicación y *user* es el usuario.

```
System: Déme su nombre
User: [espera respuesta]
```

Fig. 4. Este diálogo es generado al momento de ejecutar la aplicación VoiceXML.

En caso de que la respuesta sea nula (cadena vacía), se lanza el evento "noinput" de la aplicación, y el diálogo escuchado por dicho evento es el ilustrado en la Fig. 5.

```
System: El nombre no puede ser nulo
System: Déme su nombre
User: [espera respuesta]
```

Fig. 5. Este diálogo se genera al momento de introducir un nombre nulo.

Y finalmente si el usuario realmente introduce su nombre (Mariana Haro Graciet), la etiqueta "filled" entra en acción.

```
System: ¡Bienvenido Mariana Haro Graciet!
```

Fig. 6. Este diálogo se genera al momento de introducir un nombre correcto.

5 Transformación de la aplicación web

En esta sección veremos como se puede adaptar una aplicación web existente en una aplicación VoiceXML (con interfaces de voz).

5.1 Model–View–Controller (MVC)

Toda aplicación contiene código de presentación, de procesamiento de datos y de almacenamiento de datos, y la arquitectura de las aplicaciones difiere según como se distribuye este código.

La arquitectura que la mayoría de las aplicaciones cumple es la MVC, que es una arquitectura de tres capas basadas en componentes, donde la meta es unificar las aplicaciones cliente servidor y las aplicaciones basadas en la Web [8].

Donde el modelo (Model) es el que encapsula el estado de una aplicación, expone la funcionalidad de una aplicación, y responde a las consultas del estado de la aplicación. Estos vienen siendo componentes bean, jsp, servlets, etc.

La vista es la presentación de los datos al usuario, esta es la que suministra al modelo, permite al controlador seleccionar una vista, y este manda los datos del usuario al controlador. Estas son las páginas HTML tradicionales.

Y por último el controlador, define el comportamiento de la aplicación, selecciona una vista para responder cada una de las funcionalidades, y es el que se encarga de actualizar al modelo.

Si la aplicación a modificar está estructurada de esta manera, la parte a sustituir es la correspondiente a la vista, en lugar de código HTML y JSP tendremos diálogos de voz mediante VoiceXML.

5.2 Implementación de interfaces de voz para una aplicación legada

El ejemplo de aplicación legada que utilizaremos, tiene las funcionalidades típicas de un banco, donde se puede realizar lo siguiente (ver Fig. 7):

- Consultas
- Retiros
- Depósitos
- Altas a clientes
- Modificaciones a clientes
- Bajas a clientes

- Creación de cuentas
- Registro de clientes online
- Entre otras...

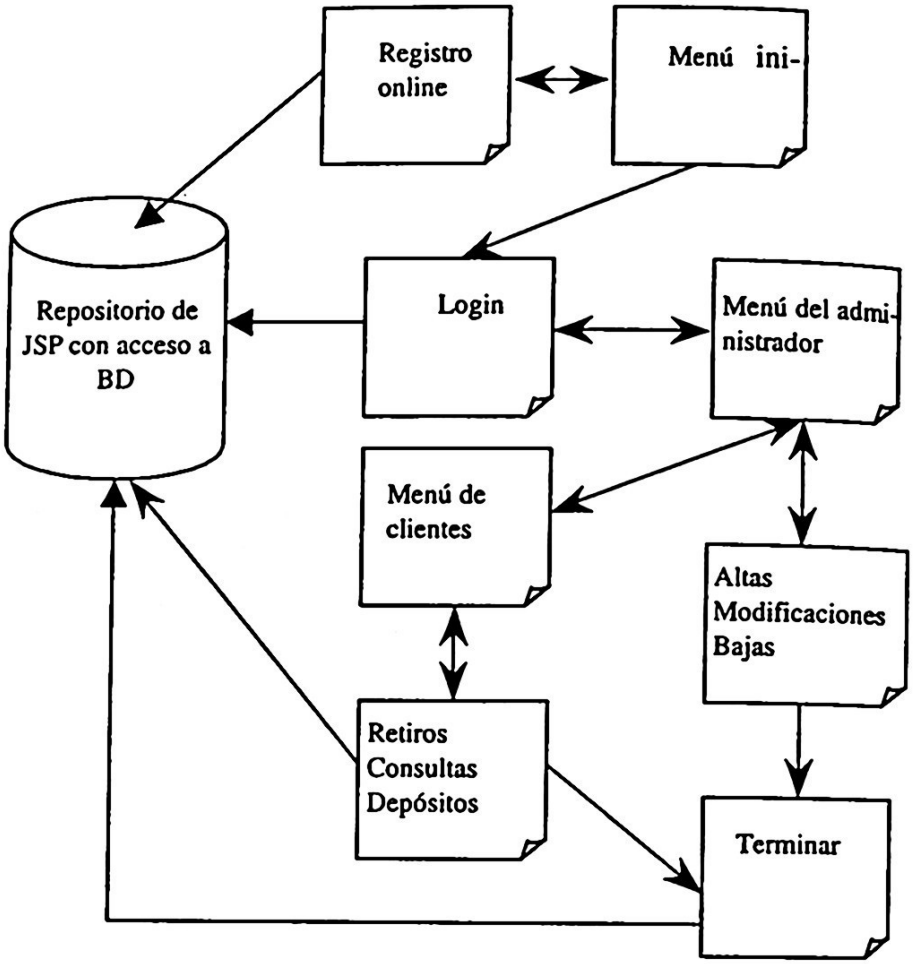

Fig. 7. Bosquejo general de la aplicación básica de bancos

Esta aplicación funciona normalmente desde cualquier navegador, pero ahora requerimos proporcionar acceso mediante voz. Como mencionamos, la parte que cambia es la que correspondía a todas las páginas que veía el usuario, y de hecho, virtualmente siguen existiendo, sólo que ahora además de verse, se escuchan. De modo que la aplicación proporciona acceso dual: el tradicional vía Web y ahora el telefónico.

Para poder ingresar a dicho sistema el usuario debe de proporcionar su user y su password del registro online, el diálogo encargado de pedirle la información es generado por la forma (form) "login" (name="login") del documento principal.vxml, cuyo código es presentado a continuación:

```
<?xml version="1.0"?>
<vxml version="2.0" mode="TTS">
    <!-- Parte 1 -->
    <form>
        <block>
            <prompt>
            Welcome to the CIC bank! Whenever you need
            help for navigate this application, just
            say help.
            </prompt>
        </block>
    </form>
    <!-- Parte 2 -->
    <menu>
        <noinput>
            <prompt>
            I didn't hear you
            </prompt>
        </noinput>
        <nomatch>
            <prompt>
            I didn't understand you
            </prompt>
        </nomatch>
        <help>
            Just follow the intructions in this menu
        </help>
        <prompt>
            <break msecs="5000"/>
            Select one of this options<enumerate/>
        </prompt>
        <choice next="#login">
            customers
        </choice>
        <choice next="#register">
            not customers
        </choice>
        <choice next="#end">
            exit
        </choice>
    </menu>
    <!-- Parte 3 -->
    <form name="login">
        <noinput>
            <prompt>I didn't hear you</prompt>
        </noinput>
        <nomatch>
            <prompt>
                I didn't understand you
            </prompt>
        </nomatch>
        <field name="user" type="spelling">
            <help>Just say your user</help>
            <prompt>Tell me your user</prompt>
        </field>
        <field name="pwd" type="spelling">
```

```
                    <help>Just say your password</help>
                    <prompt>Tell me your password</prompt>
                    <filled>
                          <!-- Parte 4 -->
                          <if cond="user=='administrator'">
                            <if cond="pwd=='system2003'">
                                <goto next="menu3.vxml"/>
                            <else/>
                                <submit
                                next="http://localhost:8080/ex
                                am-
                                ples/jsp/tmp/banco/login.jsp"/
                                >
                            </if>
                          <else/>
                            <submit
                            next="http://localhost:8080/exampl
                            es/jsp/tmp/banco/login.jsp"/>
                          </if>
                    </filled>
              </field>
              <!-- Parte 5 -->
              <block name="register">
                    <goto next="menu2.vxml"/>
              </block>
              <block name="end">
                    <prompt>
                          Thank you for visit us!, good bye!
                    </prompt>
                    <exit/>
              </block>
        </form>
  </vxml>
```

La primera parte del código anterior, es la que da la bienvenida al usuario y aquí podemos ver una etiqueta nueva par nosotros, que es "block", esta se utiliza para establecer un grupo de directivas para ser ejecutadas en orden. En este caso únicamente se encuentra la etiqueta "prompt" que es la encargada de decir el texto de bienvenida siguiente:

System: Welcome to the CIC bank! Whenever you need help for navigate this application, just say help.

Fig. 8. Diálogo de bienvenida de la aplicación

En la segunda, se genera el menú de opciones las que el usuario puede acceder (ver Fig. 9), y en esta misma parte podemos ver varias etiquetas no conocidas hasta este momento, que son: "menu", "nomatch", "help", "break", "enumerate", y "choice". La primer etiqueta es la que nos permitirá crear una lista de opciones para elección del usuario. La segunda etiqueta es muy parecida a la etiqueta "noinput",

diferencia de que esta se ejecuta al momento de que el usuario introduce una cadena que no coincida con el tipo esperado por el sistema. La tercer etiqueta, es de gran utilidad ya que esta nos permitirá que el usuario sepa siempre que hacer dentro del sistema, en el momento que el usuario diga la palabra clave 'help', el texto que se encuentre dentro de estas etiqueta se dictará en ayuda al usuario. La etiqueta "break", es la etiqueta que nos permitirá realizar una pausa en cualquier momento del sistema, el tiempo de la pausa es definido por el atributo "msecs" y el tiempo se encuentra en milisegundos. La etiqueta "enumarate", es la encargada de numerar ascendentemente y automáticamente las opciones del menú, entonces como podemos ver, esta siempre se encontrará dentro de la etiquetas "menu" y antes de las opciones de dicho menú. Y la última etiqueta es la encargada de definir las opciones del menú, y siempre tendrían definido el atributo "next", que es la acción que se ejecutará al seleccionar dicha opción, en el caso de esta aplicación se hace referencia a elementos del mismo documento, por lo tanto deben de ir precedidos por el carácter #, si la referencia es a otro documento no va precedido por dicho carácter.

La tercer parte del código, es la que realiza el primer diálogo con el usuario y es donde se pide el user y el password del usuario (ver Fig. 10), en caso de que la entrada sea nula o incorrecta se vuelven a pedir los datos anteriores (user y password).

```
System: Select one of these options
        For customers, say customers or press one.
        For not customers, say not customers or
        press two.
        For exit, say exit or press three
User: [espera de respuesta]
```

Fig. 9. Menú generado por el documento principal.vxml, donde podemos ver palabras agregadas por el intérprete en el menú, como "For customers, say or press one" donde la única palabra que fue definida en el documento fue "customers".

```
System: Tell me your user
User: ufo
System: Tell me your password
User: xxxxxx
```

Fig. 10. Diálogo donde se pide el user y el password

En caso de que sean ingresados correctamente el user y el password se ejecuta la parte 4 del código. En esta parte podemos identificar cuatro etiquetas nuevas, dos corresponden a la sentencia de condición 'if' y son "if" y "else", las otras dos etiquetas son "goto" y "submit" que corresponden a las etiquetas de referencias a toros elementos o documentos. Las etiquetas de condición ("if" y "else") son del mismo funcionamiento que en cualquier otro lenguaje de programación (if (var==cond) en C) y la condición a evaluar se define por medio del atributo "cond" donde se evalua a

la variabale, esto es lo mismo tanto para la etiqueta "if" y "else", a diferencia que el "else" se ejecuta cuando no entra al "if" correspondiente.

El llamado a la aplicación JSP se hace por medio de la siguiente etiqueta:

"<submit next = " http://localhost:8080/examples/jsp/banco/login.jsp "/>"

La página JSP será la encargada de realizar el login a la base de datos y devolver el resultado por medio de otra página VoiceXML. La etiqueta "goto" es utilizada para hacer referencia aun elemento del mismo documento o a otro documento y el "submit" es para hacer referencias a otros documentos, pero la diferencia entre los dos radica en que el "submit" manda todas las variables de la forma al que corresponde como referencia al documento referenciado y el "goto" no, al referencia lo hacen ambos por medio del atributo "next".

La última parte del código, es la que define los bloques ("block") para las otras dos opciones del menú (Register y Exit), donde podemos ver una nueva etiqueta, que es la etiqueta "exit" y esta es la que cierra por definitivo la aplicación en curso, y el diálogo escuchado como despedida al usuario es el mostrado enseguida.

```
System: Thank you for visit us! good bye!
```

Fig. 11. Este es último diálogo escuchado por el usuario y es el que despide al mismo.

6 Acceso a base de datos con VoiceXML

Todo el acceso a base de datos, no lo hace VoiceXML en sí, si no lo hace por medio de aplicaciones externas, que pueden ser JSP o ASP. En la aplicación básica de bancos utilizamos Java Server Pages para el acceso a bases de datos.

6.1 Java Server Pages (JSP)

Esta es la tecnología presentada por SUN para el desarrollo de aplicaciones dinámicas basadas en Web, y esta nos permite separar la parte dinámica de la parte estática (HTML: Hyper Text Markup Language). Este tipo de aplicaciones son desarrolladas con código 100% Java, gran ventaja para los desarrolladores de aplicaciones Java ya que no tienen que introducirse en un nuevo lenguaje para desarrollar este tipo de aplicaciones. Y una de las grandes ventajas de este tipo de aplicaciones es que reutilizan todas las librerías (API's) de Java, lo cual les da una gran robustez y funcionalidad [5].

Este tipo de aplicaciones es totalmente compatible con las aplicaciones VoiceXML así como también por parte de Microsoft lo son las Active Server Pages (ASP), y por medio de estas es que las aplicaciones VoiceXML hacen la mayoría de sus operaciones, logrando con eso una gran robustez como aplicaciones web. En nuestro caso utilizaremos las JSP para el acceso a las base de datos en nuestras aplicaciones VoiceXML.

6.2 Integración de un documento JSP

VoiceXML hace referencia a las JSP, por la etiqueta ya vistas ("goto" y "submit"), para el caso del login del usuario se empleo la etiqueta "submit", ya que se envían parámetros al documento JSP [4].

El fragmento de código que realiza el acceso a la base de datos para realizar el login es el siguiente (login.jsp):

```
// Declaración de variables de la conexión a la BD
Connection con;
Statement stmt;
ResultSet rs;
// Se guardan los valores de los parámetros enviados
// por el documento VoiceXML
String usuario = request.getParameter("user");
String contraseña = request.getParameter("pwd");
String query   = "";
// Se carga el controlador de JDBC y se hacen los
// objetos de conexión.
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
con = DriverManager.getConnection("jdbc:odbc:Banco", "","");
stmt = con.createStatement();
rs   = stmt.executeQuery("SELECT * FROM logs");
// Se hace la búsqueda del usuario en la base de datos
while(rs.next())
{
     if(usuario.equals(rs.getString("User")))
     if(contraseña.equals(rs.getString("Pwd")))
     {
      // Si se encuentra al usuario se guarda su id y
      // su user
      int tmp = rs.getInt("idCliente");
      Integer id = new Integer(tmp);
      session.setAttribute("idCliente", id.toString());
      session.setAttribute("Nombre", usuario);
      Statement stmt2;
      stmt2 = con.createStatement();
      // De igual forma se cargan sus cuentas (Tipo y
      // numero)
      ResultSet rs2;
      rs2 = stmt2.executeQuery("SELECT * FROM cuentas WHERE
      idCliente = " + tmp );
      while(rs2.next()){
       String tipo = rs2.getString("Tipo");
       int numero = rs2.getInt("Numero");
       Integer num = new Integer(numero);
       session.setAttribute("Tipo", tipo);
       session.setAttribute("Numero", num.toString());
      }
     // Finalmente se carga el documento principal de
     // la aplicación VoiceXML
     %><goto next="http://server/banco/menu.vxml" />
   <%}
}
```

En la última parte del código anterior, podemos observar una etiqueta de VoiceXML ("goto"), esto se puede hacer siempre y cuando se estructure el documento JSP como un documento VoiceXML (ver subtema 4.1), y así podemos mezclar código JSP con código VoiceXML y tener una mayor robustez en nuestras aplicaciones.

7 Conclusiones

En este artículo, se dieron a conocer las principales características de las aplicaciones VoiceXML, así como las principales etiquetas que nos permiten desarrollar documentos VoiceXML, y además de cómo tener acceso a bases de datos. Los ejemplos fueron ejecutados sobre el emulador "Voice Studio de Cambridge", este no tiene la capacidad de probar nuestras aplicaciones por vía telefónica (ningún emulador la tiene), los únicos que la tienen son los servicios ofrecidos sobre Internet ya mencionados, pero todos estos en la unión americana, por lo tanto no son óptimos para los que no pertenecemos a la unión americana. Debido a esto, en el CIC estamos trabajando en la implementación de un intérprete VoiceXML y un ambiente de desarrollo, los cuales no requieren de dispositivos de hardware costosos y superan a la mayoría de los existentes en que además se apegan a la última versión de VoiceXML (2.0), y permiten interactuar con un teléfono desde el emulador, es decir, probar las aplicaciones en un ambiente real.

Referencias

1. Bob Edgar: The VoiceXML handbook, CMP Books, New York, U.S., 2001
2. Sharma Chetan, Kunins Jeff: VoiceXML, WILEY, Indianapolis, Indiana, U.S., 2002
3. Beasley Rick, Farley Mike, O'Reilly John, Squire Leon: Voice Application Development with VoiceXML, SAMS, New York, U.S., 2002
4. Nordin A. Brandon: VoiceXML 2.0 Developer's Guide, McGraw-Hill/Osborne, Berkeley, California, U.S., 2002
5. Tom Myers, Alexander Nakhimovsky: Professional Java XML programming with servlets and JSP, Wrox Press, Birmingham, UK, 1999
6. Simon Brown: Professional JSP, Wrox Press, Birmingham, UK, 2001
7. David Megginson: Structuring XML documents, Prentice Hall PTR, Upper Saddle River, NJ, 1998
8. Erich Gamma: Design Patterns, Adison – Wesley, U.S., Abril 1996
9. Beat Signer, Moira C. Norrie, Peter Geissbuehler and Daniel Heiniger
10. http://www.w3.org/Voice
11. http://www.voicexml.org