**Universidad Central "Marta Abreu" De Las Villas.**

**Facultad de Ingeniería Eléctrica**

**Departamento de Telecomunicaciones y Electrónica**


# TRABAJO DE DIPLOMA

## "Aplicaciones de Integración Telefonía Ordenador"

**Autor: Humberto Sierra Martínez.**

**Tutor: MSc Carlos A. Ferrer Riesgo**

**Santa Clara**

**2003-2004**

**"Año del 45 aniversario del triunfo de la Revolución"**

# Resumen

La integración telefonía-ordenador surge con el objetivo de integrar las computadoras actuales a la telefonía tradicional. Mediante su uso no solo se logran hacer y atender llamadas telefónicas sino que también se pueden implementar diversos servicios muy útiles en la actualidad.

La herramienta de integración telefonía-ordenador es un software hecho en DELPHI que usando las TAPI de WINDOWS, exactamente la versión 3.0 de las mismas, permite crear una aplicación capaz de brindar servicios de menú telefónico, como contestadora telefónica y despertador automático. Esta herramienta será soportada por los sistemas operativos actuales (XP o superior). Para la implementación del software y sus respectivos servicios se requiere de una línea telefónica convencional conectada a un voice-modem.

La aplicación tiene una interfaz visual que la convierte en una cómoda herramienta para los diseñadores de menús telefónicos, además de contar con una ayuda que facilita su uso.

*Índice*

# Introducción

La integración telefonía-ordenador conocido por sus siglas en ingles CTI (Computer Telephony Integration) es usado cada vez mas, haciendo más competitivas a las instituciones que se sirvan de ellas. Estas pueden dar un servicio más completo y apropiado a sus clientes sin que ello requiera de inversiones en personal, como pudo ser en un pasado. El mismo desarrollo de las telecomunicaciones conlleva a un mejor servicio a los usuarios y la CTI no se ha alejado de ello.

El desarrollo del sistema operativo ***Windows*** ha traído consigo el desarrollo de las TAPI (Telephony Aplication Programming Interface) que permite aplicar las CTI. Con el uso de las TAPI es posible colgar, descolgar, discar tonos DTMF y por ende hacer y atender llamadas.

Existe por parte del CEETI (Centro de Estudio de Electrónica y Tecnologías de la Información) la necesidad de realizar encuestas para diferentes usuarios usando telefonía conmutada, por lo que resulta muy conveniente una aplicación de software para el desarrollo de las mismas. Además esta aplicación puede ser útil para muchas empresas que deseen ofrecer a sus usuarios o clientes los servicios de contestadora telefónica, despertador automático, etc.

Este trabajo tiene como objetivo desarrollar una aplicación de las herramientas de integración Telefonía-Ordenador soportadas en los sistemas operativos actuales, que permita su configuración para aplicaciones múltiples de menús telefónicos como encuestas telefónicas, servicios de despertador automático, contestadota automática, etc.

Las tareas propuestas para la investigación son las siguientes:

- Búsqueda bibliográfica.
- Selección de Entorno de Desarrollo.
- Programación y puesta a punto.
- Confección de Manual de Usuario.

- Confección de Informe.

Su aplicabilidad estaría dada en todas las entidades o instituciones que atiendan a muchos usuarios por la red pública telefónica conmutada o que quieran brindar algunos de los servicios que ofrece el mismo, como pueden ser hoteles, restaurantes, centros de investigación, ETECSA, etc.

Con el proyecto se evita la adquisición de un software para realizar las tareas que este debe cumplir, también se garantiza la atención de todas las llamadas entrantes, de no existir personal para atender las mismas en ese momento. Este logra un valor agregado, en una mejor atención a los usuarios o clientes, permite servicio de menú telefónico y esta construido de forma flexible permitiéndoles a los usuarios configurar previamente los servicios a utilizar.

Este trabajo estará dividido en tres capítulos, donde en el capitulo I se abordaran los antecedentes históricos, el entorno teórico y el fundamento de la tecnología CTI. El capitulo II estará dedicado a la metodología y los procedimientos de los algoritmos a realizar. En el capitulo III se realiza la descripción del software y su interfaz con el usuario.

El software posee una ayuda del mismo, en donde se explica de forma clara y concisa como realizar la programación y el diseño de las aplicaciones. Como parte de la ayuda vienen una serie de ejemplos para que el usuario se familiarize con el software.

# Capitulo I: Generalidades

## *1.1 Reseña histórica*

Durante años los sectores de la informática y las telecomunicaciones han estado buscado la convergencia de sus tecnologías, lo que llevo a la aparición de las CTI (Computer Telephony Integration). Esta tecnología surge con la idea de combinar las ventajas de los sistemas telefónicos e informáticos. Con el incremento de los servicios telefónicos controlados por la PC (Personal Computer) las compañías se han dado la tarea de construir aplicaciones para esta tecnología. Este sistema es capaz de controlar las llamadas, es decir, responder, colgar, descolgar, transferir, además de introducir información en la línea en forma de voz, datos, fax o extraer información de la línea en forma de tonos de multifrecuencia (DTMF), o por reconocimiento de voz, existen aplicaciones capaces de recibir datos de un modem, fax etc. Algunas de las aplicaciones de la tecnología pueden ser fax por ordenador, contestadota telefónica, o cuando llamamos a un servicio de atención al cliente y nos responde un sistema automático que nos guía con menús telefónicos, llamados también portales telefónicos. Con la integración entre las tecnologías informáticas y de telecomunicaciones se busca un mayor dinamismo en los servicios ofrecidos por ambos sectores, en los últimos años el crecimiento de los servicios ha tenido grandes avances.

Históricamente esta integración ha sido asociada a soluciones para PBX (Private Branch Exchange), siendo este un equipamiento de alto costo, y en el que el vendedor de estos equipos suministra el software para su uso. En estos momentos los modem son unos de los periféricos más comunes en las PC, lo que facilita dicha integración. Microsoft diseñó las TAPI (Telephony Application Programming Interface) para ofrecer una alternativa en la solución de las CTI en sus sistemas operativos. La meta de las TAPI es presentar independencia entre el software y el equipamiento usado, en este caso la aplicación no necesita saber nada sobre el hardware, sino que ella trabajara directamente sobre las TAPI.

## *1.2 Tecnología CTI*

La tecnología actúa sobre diferentes dispositivos telefónicos o informáticos, entre ellos se encuentran:

- *Las llamadas centralitas o PBX* (Private Branch Exchange), estas permiten todo tipo de conexiones al exterior como son acceso RDSI básico y primario, líneas analógicas y conexiones a Internet (teniendo un modulo de voz sobre IP).
- *Teléfonos con conexión a PC*, estos dispositivos además de estar siempre conectado a una línea telefónica de forma opcional se pueden conectar a una PC, y desde ella se pueden ejecutar varias funciones como pueden ser configurar el teléfono, marcar y responder activando manos libres.
- *Tarjetas propietarias de telefonía*, estas fueron echas para la tecnología CTI se conectan a la PC por el puerto serie o directamente al bus ISA o PCI, también se conectan a una o varias líneas para realizar funciones especificas.
- *Modem*, con una apropiada programación se logra que su funcionamiento sea similar a las tarjetas telefónicas que tienen un mayor costo en el mercado, un modem con capacidad de voz (estos módems permiten colocar señales de audio en la línea y de grabar lo que se recibe por ella, además de la modulación/demodulación de datos normal) permite emitir audio por la línea y grabar lo que se recibe de ella. Otra capacidad seria la reconocer tonos DTMF que son la señales de numeración que reciba por la línea[2].

### *1.2.1 Aplicaciones.*

La aplicación directa de la tecnología ha significado un paso importante en los sistemas telefónicos actuales, permitiendo a estos una flexibilidad en el control de llamadas, tarificación, etc.

Se puede decir que la tarificación es la aplicación CTI más antigua que existe. La misma consiste en obtener información de todas las llamadas realizadas y asignarle de alguna forma un precio a cada una. La introducción de los ordenadores ha permitido un gran avance en los procesos de tarificación, permitiendo la creación de ofertas específicas y variadas de precios que ha traído consigo la mejora de los servicios.

El control de llamadas es la capacidad de realizar operaciones telefónicas desde un programa informático (software), también este debe ser capaz de obtener toda la información posible que tengan nuestros dispositivos telefónicos sobre el transcurso de la llamada dentro del sistema es decir tanto las generadas por la PC como las que fueron generadas manualmente por los usuarios, esta información pudiera ser por ejemplo el colgado o descolgado del teléfono. Existen diversos medios para programar aplicaciones de este tipo entre las que se encuentran las TAPI interfaz que se utiliza en este trabajo y que será explicada mas adelante en este capítulo. La CTI permite actuar en cualquier momento sobre una llamada tanto por software como manualmente. El control de llamadas tiene dos conceptos básicos:

- Métodos que permitan realizar acciones sobre las llamadas como pudieran ser marcar, colgar, responder, retener, transferir, llamadas.
- Eventos para que los dispositivos telefónicos informen lo ocurrido, como llamadas entrantes, salientes, etc.

Para cada llamada existen una serie de datos que son proporcionados por el sistema, estos datos sirven para la identificación de la llamada, así como conocer en que estado se encuentran algunos de estos datos son:

- *identificador*: Número único para la llamada que se usa para cada función que vaya a modificar la llamada y que se obtiene a través de eventos, salvo cuando se crea una llamada que en este caso el identificador lo crea la misma función.
- *Número llamante*: Obtenido en las llamadas externas.

- *Estado*: Una llamada puede estar en uno de varios estados, como son llamada retenida, finalizada, etc.

Existe también el control de contenido pues resulta que también existe control sobre el tipo de información que fluye sobre la llamada. Dependiendo del tipo de información se pueden generar diferentes servicios, pueden enviarse y recibirse datos a través de un modem, utilizar modem de voz para grabar y reproducir mensajes creando aplicaciones de mensajería vocal, reconocer señales enviadas por el usuarios del otro lado de la línea como tonos DTMF creando aplicaciones portales telefónicos[4].

Las aplicaciones de mensajería integrada pueden integrar mensajería vocal, convirtiendo los mensajes de voz en correo electrónico. También existen sistemas que permiten al usuario autentificarse mediante una clave conformada por dígitos DTMF y mediante un sintetizador de voz su correo sea leído.

Casi todas las aplicaciones CTI hacen uso de bases de datos ya sea para hacer búsquedas en las bases de datos con la información obtenida del sistema telefónico o viceversa, por ejemplo la tarificación es un uso frecuente de las bases de datos por las CTI

### *1.2.2 Arquitectura hardware*

La mayoría de las aplicaciones CTI poseen dos tipos de arquitecturas: la de acceso directo o (first party) y acceso indirecto o (third party). El modelo de acceso directo consiste en que cada ordenador realiza sus funciones CTI solamente accediendo a los dispositivos telefónicos conectado a él. El modelo de acceso indirecto opera bajo el formato cliente-servidor, es decir existe un servidor de telefonía poseedor de los dispositivos telefónicos, y todas las PC que son en este caso los clientes pueden conectarse al servidor y hacer uso de él, dependiendo de la política de permiso del servidor[2].

## *1.3 TAPIs*

Las TAPI les permiten a los programadores desarrollar aplicaciones que soportan transmisión de voz y datos sobre varios dispositivos terminales, soporta diferentes tipos de conexiones y técnicas de administración de llamadas. Estas aplicaciones logran un manejo más eficiente de las llamadas de voz y el control de las operaciones de transferencia de datos. Las TAPI brindan acceso a la red telefónica e incorporan un grupo de herramientas como son:

- Conectar directamente una aplicación a la red telefónica.
- El discado de números automáticamente.
- La transmisión de documentos, fax o correo electrónico.
- Establecer y administrar llamadas de conferencia.
- Usar el caller ID para manipular las llamadas entrantes de forma automática.
- Control de operaciones de una computadora remota.
- Computo cooperativo sobre la línea telefónica.

### *1.3.1 Modelo*

Existen cuatro tipos de modelos telefónicos LineApp, línea, teléfonos, y llamadas. Estos modelos también son conocidos en la literatura como objetos TAPI[1].

El modelo LineApp es la base de todos los otros objetos de TAPI, el no tiene una correspondencia directa con algún dispositivo físico. Cada LineApp representa una sección TAPI diferente, se crea cuando se llama a las funciones de inicialización y se destruye al llamar funciones para terminar. LineApp es el propietario de los otros objetos de TAPI, esta propiedad es exclusiva y no transferible y cada objeto de LineApp tiene su manipulador.

El modelo línea es una abstracción de la línea telefónica representado por TSP (Telephony Service Provider), que es una dll que soporta el control de los dispositivos de comunicación. En algunas ocasiones puede ser la línea telefónica pero a menudo el

modelo línea de TAPI no se corresponde con la línea sino que es una representación lógica de un dispositivo que funciona como un medio de transporte para el flujo de datos telefónicos. El TSP determina como se representa un dispositivo físico en términos de líneas de TAPI. Los dispositivos de línea son comúnmente usados con el propósito de control de llamadas.

El modelo teléfono es una representación lógica del equipo terminal, el puede ser usado sin llamar, es decir pudiera emplearse como una interfase de un sistema de correo de voz grabando y recuperando los mensajes del correo sin realmente tener que hacer una llamada. Este modelo esta íntimamente ligado con los dispositivos físicos telefónicos, incluye los conceptos de micrófono, altavoces, teléfonos, etc.

El modelo llamada puede ser creado tanto por TAPI como por TSP. Cada modelo de llamada es identificado por su manipulador de llamadas. El modelo llamada es creado en una línea y siempre esta asociado con el modelo línea, una llamada esta siempre poseída por una línea y su relación no es necesario que sea uno a uno, un modelo línea puede tener varias llamadas como por ejemplo una conferencia.

### *1.3.2 TAPI y WOSA*

El modelo WOSA (Windows Open Services Architecture) esta compuesto por tres piezas, cada uno de ellas juega un papel importante en la programación de servicios de las aplicaciones. Sus tres componentes son:

- Cliente API (Application Programming Interface) usado por el programa que solicita el servicio.
- Servidor SPI (Service Provider Interface) para los programas que posean servicios extendidos como e-mail, telefonía, entre otros.

- La interfaz API/SPI que es un modulo de enlaces a llamadas API y SPI. La interfaz usualmente esta implementada como DLL (Dynamic Library Link) en el entorno Windows.

Cada componente tiene un papel importante cumpliendo sus tareas de forma independiente, pero haciendo que la interfaz funcione como un todo. Para que TAPI sea capaz de cumplir su trabajo, divide sus tareas en dos capas diferentes, el cliente API y el servidor SPI. Cada uno de ellos posee sus funciones para completar las tareas telefónicas como: abrir línea, chequear tonos de discado, discado telefónico, chequear señales de ocupado o timbre entre otras. El cliente API envía una solicitud de la aplicación al servidor SPI, este completa la tarea y pasa los resultados a la llamada del programa pedida por API[3].

### *1.3.3 Configuraciones típicas de TAPI*

El modelo de telefonía TAPI esta diseñado en función a las diferentes configuraciones físicas, donde cada una de ellas tiene sus ventajas y desventajas. Existen cuatro configuraciones generales, ellas son:

- Orientado a teléfonos.
- Orientado a PC.
- Línea unida o compartida.
- Multilínea.

**La configuración orientada a teléfonos** es usada cuando el proceso de la llamada esta orientado a voz y en la que los equipos telefónicos estándares son usados mas frecuentemente Esta configuración es más útil cuando se usa el teléfono como equipo primario para el acceso a la línea telefónica. El teléfono se apoya en la PC para el acceso a la línea telefónica, aquí al PC no puede ser capaz de compartir todas las actividades en la línea. Esta configuración no excluye de usar la PC para originar llamadas ya que permite el discado, y el teléfono puede identificar la llamada en cualquier momento.

**La configuración orientada a PC** es usada cuando el proceso de la llamada esta orientado a datos, las PC son usadas con mas frecuencia para voz y datos. Esta configuración es útil cuando se usa la PC como equipo primario para acceder a la línea telefónica. En esta configuración es la PC la que con más frecuencia origina la llamada. La PC puede manejar lista de números telefónicos y manipular el discado, dependiendo del modo exacto de la llamada, ella puede ser usada como un display digital en pantalla mientras es manipulada la información de voz. La configuración también permite que las llamadas sean generadas por el teléfono pero en este caso la llamada puede ser compartida por la PC mientras el flujo de datos esta pasando por ella.

La PC puede actuar como un manipulador de llamadas para el teléfono, donde puede recibir voz, datos y fax en la misma dirección. Si una llamada entra a la línea la PC puede responder y determinar el modo de la misma, si es un fax enrutarlo a la maquina de fax, los datos pueden ser procesado directamente y si es voz enviar al teléfono.

Los mensajes que sean recibidos para los usuarios pueden ser grabados y puestos en un lugar para revisarlos posteriormente, se pueden reenviar las llamadas a otra dirección. Con la adición de el servicio de caller ID la PC puede hacer un filtrado de las llamadas, y explorando las mismas designar diferentes accesos a los llamantes, PC o teléfonos.

**La configuración de línea unida o compartida** es un compromiso entre las dos anteriores, permite una operación equilibrada de los servicios de línea entre los diferentes equipos. Esta configuración permite un acceso simultáneo a la línea por parte de la PC y el teléfono, tiene como ventaja que cualquiera de los dos puede originar la llamada, como desventaja que ambos tienen igual acceso a las llamadas entrantes y al llegar una llamada los dos timbran y puede que ambos atiendan la misma llamada en línea. Con esta configuración se puede acceder directamente a la línea telefónica desde la PC, con el uso de un micrófono para la entrada y los altavoces o audífonos se puede hacer un teléfono virtual que funcione igual que uno convencional.

**La configuración de multilínea** se diferencia con las anteriores porque la PC actúa como servidor o centro de conmutación que conecta la línea exterior a una o más PC o teléfonos con la principal ventaja de que no es necesario una relación directa entre la línea telefónica y el dispositivo final (PC o teléfonos). Las configuraciones anteriores han sido para modelos de una única línea, lo que normalmente se nombra acceso directo (first-party). Las TAPI también soportan configuraciones de varias líneas, acceso indirecto (thrid-party). En el ambiente de multilínea este control actúa como una centralita telefónica, cuando la llamada llega, ella acepta la llamada y determina el destino de la misma, la enruta para la extensión correcta y abandona el control de la llamada. Existen dos configuraciones básicas de multilínea:

- Servidor voz: Usado para proveer correos de voz y otros servicios de forma centralizada accesible por cualquier teléfono. Las TAPI se usan para atender las llamadas.
- Servidor PBX: Usado para proveer control de llamadas para las líneas interiores y exteriores. Las TAPI son usadas para aceptar la llamada y llevar la misma a su destino final.

### *1.3.4 Niveles de servicio de TAPI*

#### 1.3.4.1 Servicio de telefonía asistida

Este es el más sencillo de los servicios TAPI. Bajo esta interfase se pueden hacer llamadas externas y chequear los discados de la PC. Puede ser usado por un simple botón de dial para aplicaciones existentes o para añadir capacidades de discado a nuevas aplicaciones que usen la telefonía como un valor añadido. Con este modelo solo se podría dar acceso a los programas de colocar llamadas externas. El discado de la llamada es manipulado por otra aplicación de Windows que es Dialer.exe.

### 1.3.4.2 Servicios básicos de telefonía

Este es el siguiente nivel en los modelos de servicio de las TAPI. Este modelo permite crear aplicaciones que sean capaces de hacer llamar externas de voz y datos sobre líneas telefónicas analógicas. Este nivel permite usar líneas mas sofisticadas como ISDN o líneas digitales.

El nivel se enfoca en el uso de dispositivos de línea como transporte de información de un lugar a otro. Este dispositivo de línea para TAPI puede ser un teléfono, un modem de datos, o cualquier tipo de equipamiento físico que pueda ser incluido en la línea telefónica, pero es considerado como un equipo virtual, no físico. El dispositivo de línea no esta asociado directamente a la línea telefónica, TAPI puede ver múltiples equipos en la misma maquina, pero tan solo existe una línea telefónica.

La interfaz TAPI es la encargada de manipular los diferentes servicios solicitados a la PC, es posible que existan varias aplicaciones corriendo sobre la PC y que estas pidan servicio al mismo tiempo, entonces la aplicación de control de llamadas Dialer.exe acepta cada una de las solicitudes y las pone en una cola para procesarlas en el orden pedido.

Este modelo tiene varias funciones para la manipulación de las solicitudes, estas se pueden agrupar en grupos lógicos, los cuales les mostramos a continuación:

- *Manipulación básica de la línea*: Manipula la inicialización, el abierto y cerrado de la línea de TAPI.
- *Configuración y status de la línea*: Manipula el leído y la escritura de varios valores en parámetros de control del comportamiento de la línea.
- *Función interior y exterior*: Manipula los detalles para llamadas de entrada o salida, ya sea de datos o voz.
- *Función de direccionamiento*: Manipula los detalles de reconocimiento, traducción y/o construcción de direcciones telefónicas (números telefónicos).

- *Características diversas*: Manipula otras funciones relacionadas con TAPI, como pudieran ser la administración y el monitoreo de los privilegios de las llamadas.

Además del grupo de funciones, el modelo define varias estructuras de datos que serán usadas para pasar la información entre TAPI y la aplicación que solicita el servicio, la capa de la estructura contiene variables y datos fijos. Estos permiten que las funciones de API contengan información de tamaño no determinado sin previo conocimiento del contenido de la estructura.

La telefonía API usa mensajes de Windows para comunicarse con la aplicación que solicita el servicio. Cada mensaje retorna el mismo grupo de parámetros, usualmente el manipulador de llamadas puede ser el manipulador de línea.

#### 1.3.4.3 Servicio de telefonía suplementario

Este servicio provee funciones avanzadas al manipulador del dispositivo de línea. Para este servicio el acceso es dependiente del tipo de línea telefónica conectada a la PC, las funciones permiten a los programadores manipular los servicios solicitados por la telefonía multilínea, con sus funciones se manipulan los diferentes teléfonos usando una o varias líneas telefónicas, los dispositivos telefónicos y de línea no necesitan tener una correspondencia directa entre el dispositivo telefónico y el dispositivo de línea definida. En este caso se puede usar las TAPI para crear una aplicación que funcione como una centralita telefónica para manejar los servicios telefónicos. El servicio de telefonía suplementario define y manipula los dispositivos telefónicos. Para TAPI un dispositivo telefónico es cualquier dispositivo que pueda recibir o efectuar llamadas. Se pudiera registrar la PC como un dispositivo telefónico, por lo que se puede usar los recursos de la PC para efectuar o recibir llamadas sin tener un dispositivo telefónico, en ese caso la PC tiene que tener entrada y salida de audio. Existe un grupo de funciones para el control de llamadas y están divididas en los siguientes grupos:

- *Funciones de dígitos y tonos:* Permiten generar y detectar los dígitos o tonos sobre la línea.
- *Funciones de manipulación de línea:* Permiten aceptar, rechazar y redireccionar llamadas, entre otras.
- *Funciones de características de llamadas*: Provee llamadas en espera, transferencias, enrutamientos, entre otras.
- *Funciones de diversas características*: Permite añadir características especificas para las solicitudes de servicio a TAPI, como el monitoreo de la línea y los parámetros de la configuración de la llamada.

Las API brinda funciones para la manipulación de los dispositivos telefónicos. Estas funciones les permiten a los programadores crear sus propios dispositivos telefónicos en código, y es que se puede crear un teléfono virtual usando le dispositivo telefónico TAPI, este tiene la propiedad de que la PC actué como un teléfono de simple o múltiple línea. La telefonía suplementaria API para dispositivos telefónicos es dividida en los siguientes grupos de funciones

- *Funciones básicas de manipulación telefónica:* Permite la inicialización y la terminación básica, el abierto y el cerrado del dispositivo telefónico, y el timbrado al abrir el dispositivo.
- *Funciones de status y configuraciones telefónicas:* Permite leer y escribir las diferentes configuraciones del dispositivo telefónico como el volumen, ganancia, etc.

Los mensajes de Windows son usados para las funciones de retorno. Cada mensaje retorna los mismos grupos de parámetros. El primero de estos es el manipulador de servicios telefónicos, el segundo es el valor de retorno. Este siempre será manipulado por la aplicación que este corriendo. Los siguientes valores dependerán del mensaje.

#### 1.3.4.4 Servicios de telefonía extendida

Este es el último de los servicios telefónicos, permite a los vendedores de equipamiento definir sus propias funciones y servicios específicos del equipo y seguir operando bajo el modelo de servicio TAPI. Para adicionar los mismos Microsoft permite que los vendedores brinden servicios que no están previamente definidos por TAPI. El modelo TAPI define los dispositivos telefónicos y de línea para la telefonía extendida. Los proveedores de la telefonía extendida definen los parámetros de las llamadas y sus significados, esta información es publicada para los programadores.

### *1.3.5 Trabajo con las TAPI*

#### 1.3.5.1 Comenzar una sección TAPI

Para comenzar una sección TAPI el primer paso a seguir es llamar a la subrutina *lineInitialize* para la versión 1.3 o *lineInitializeEx* para versiones 2.x o superior. Estas llamadas son cargadas en el espacio de memoria de la aplicación, de TAPI.DLL con esta llamada se logra la inicialización del enlace entre la aplicación y el proveedor de servicios TAPI esta subrutina incluye punteros para funciones de retorno en el código, buscando manipular los mensajes[1]. Después de la inicialización retornan valores del parámetro manipulador de línea el valor será usado a través de la sección identificando dicha sección dentro de la aplicación, también se obtendrá el número de líneas definidas por la PC. Si ocurre un error es recibido un valor diferente a cero. Cuando se llama a la función de inicialización la aplicación selecciona y provee a TAPI con métodos de notificación de eventos, estos reportes hacia la aplicación incluyen cambios en llamadas existentes, nuevas llamadas entrantes, conexiones y desconexiones.

#### 1.3.5.2 Seleccionar una línea

Podemos encontrar más de un dispositivo de línea telefónica en una PC. Con TAPI se puede escoger el dispositivo dependiendo de donde ellos se encuentren y de lo que puedan hacer, la función *lineGetDevCaps* permite averiguar sobre las capacidades de los

dispositivos[1]. Una aplicación puede usar la información devuelta para escoger el dispositivo de línea que posea las características deseadas.

Una vez que el dispositivo de línea este seleccionado se debe negociar la versión de API a utilizar usando la función *lineNegotiateAPIVersion* que permite al llamador especificar la versión a utilizar siendo conveniente especificar la versión en que la aplicación ha sido escrita[1][8]

#### 1.3.5.3 Abrir una línea

Una vez seleccionada la línea se puede abrir la misma. La función *lineOpen* se usa para solicitar de cada línea un apropiado nivel de servicio. Por ejemplo si se quisiera un servicio especificado *lineOpen* localiza la línea que soporte dicho servicio. Puede que una PC brinde varios servicios pero para el servicio solicitado no exista dispositivo disponible o el existente este en estado de ocupado, en este caso se recibirá un mensaje de error. Si una línea apropiada para el servicio requerido esta disponible se recibirá como código de retorno un cero, y un valor que le indicara al manipulador que la línea se encuentra abierta, este valor es usado para las siguientes llamadas[8].

#### 1.3.5.4 Realizar llamadas

Se pueden iniciar llamadas llamando a la función *lineMakeCall.* La misma requiere que la línea se encuentre abierta y tenga su manipulador de llamadas (handle). Es necesario que se le sea especificado el número telefónico en forma de una cadena de caracteres. El número telefónico se le pudiera pasar completo o llevar el mismo a un número más simple mediante una “traslación”. De existir algún problema con la realización de la llamada el código devuelto es diferente de cero[8].

La traslación antes mencionada consiste en que al instalar un modem en una PC que soporte TAPI deben ser llenados cajas de diálogos especificando código de área y de

país, pulsar un número para obtener una línea externa etc. Esa información es usada por el proceso de traslación de números de destino, las TAPI provee funciones como *lineTranslateAddress* para lograr lo anterior. La dirección entonces es pasada mas simple en la forma +código de país (código de área) número.

En ocasiones es deseable romper el discado en múltiples partes, se usa este método cuando es necesario insertar demoras en el discado entre grupos de dígitos. Un punto y coma anexado a la cadena le dice a TAPI que el discado es incompleto y que más dígitos seguirán, los dígitos que siguen se le pueden pasar llamando a la función *lineDial*. El estado del discado es incompleto mientras tenga añadido un punto y coma a la cadena se puede completar el discado omitiendo el punto y coma o llamando a la función *lineDial* pasándole un número nulo[8].

#### 1.3.5.5 Progreso de llamadas

Una vez que el discado ha sido completado, TAPI comenzara reportando el estado de la llamada usando mensajes *LINE_CALLSTATE*. El primer estado de la llamada pasa por los procedimientos *LINE_CALLSTATE* que indica que la llamada comienza a rutearse hacia su destino, si la llamada es contestada cambia hacia el estado *LINECALLSTATE_CONNECTED* indicando que la llamada se ha establecido. Dependiendo de la capacidad del proveedor de servicio es común encontrar algunos estados antes de alcanzar el estado de *LINECALLSTATE_CONNECTED*[8].

#### 1.3.5.6 Llamadas entrantes

La manipulación de las llamadas entrantes es más simple que la manipulación de las llamadas salientes, para que una aplicación responda a una llamada entrante se debe tener la línea abierta con privilegios de propietario. Una vez que la línea esta abierta la aplicación espera por una llamada, cuando una llamada entra el mensaje de estado

*LINECALLSTATE_OFFERING* indica que una llamada esta siendo ofrecida por la aplicación.

### *1.3.6 Requerimientos para TAPI*

Los servicios TAPI son ruteados a través de modem, que son usados para comunicarse entre el programa y el dispositivo. Los datos existentes en las PC están de forma binaria, y las líneas telefónicas no tienen la capacidad de transportar estos datos binarios, así que la información tiene que ir en forma de señales analógicas. Para la conversión de datos binarios a señales analógicas y viceversa se usan los modem (*mo*dulador-*dem*odulador). Los modem de datos básicos lo que hacen es utilizar la línea telefónica y mandar datos digitales, estos modem tienen un grupo de códigos de control llamados AT que se encargan de enviar el control directamente al hardware.

Enviar los datos requiere de tres pasos principales, primero establecer la conexión, segundo paso modular la información a trasmitir, y como último paso demodular la información que viene de la línea. Esta claro que si el modem es el que recibe la llamada, después de establecer la conexión tiene que primero demodular y luego modular los pasos 2 y 3 pueden intercambiar su orden pero siempre después que la conexión esta establecida.

Las TAPI también requieren del TSPI (Telephony Service Provider Interface) que es usualmente suministrado por los fabricantes, esta interfase lo que logra es transformar nuestras solicitudes de servicio TAPI en comandos que sean entendidos por el modem.

#### 1.3.6.1 Modem de voz

Estos modem tienen la capacidad de ofrecer los mismos servicios que los modem de datos además de soportar servicios de voz. Su hardware esta equipado para soportar servicios telefónicos como caller ID, llamada en espera, redireccionamiento de llamadas,

etc. Estos modem también poseen su grupo de códigos control llamados AT+V (AT mas voz). Los mismos requieren de un driver para su trabajo con TAPI, este driver es suministrado por el fabricante, también pudiera utilizar UniModemV driver suministrado por la Microsoft.

## *1.4 Software CTI*

Existen a nivel mundial, múltiples aplicaciones que implementan servicios sobre la tecnología CTI. Es de interés del CEETI contar con un programa propio que también sea capaz de generar servicios que son ampliamente conocidos. Con el programa y por supuesto con su código fuente, algo imposible de obtener de las aplicaciones ya realizadas sobre este tema, se brindarían servicios a diferentes clientes. A medida que el software sea probado e implementado podrá tener cambios que favorezcan a los más exigentes clientes.

Unos de los programas más completo que implementan servicios sobre la tecnología CTI es el WinIVR Interactive Voice Response, esta aplicación es capaz de realizar casi todos los servicios de fax, correo de voz, y soporta varias llamadas por línea, es decir soporta servicio de llamada en espera, conferencia, etc.[7]. Nuestro software no implementa los servicios anteriores. Como aspecto positivo podemos destacar que la programación de nuestro software resulta más sencilla para el operador encargado de implementar los servicios.

Para la programación del Interactive Voice Responce es necesario declarar variables y trabajar con estructuras de flujo, para la programación de nuestro software esto se resuelve de manera sencilla, se declaran hijos que responden a dígitos tecleados por el usuario que accede al servicio, hijos por defectos que se ejecutan inmediatamente después que su padre, y se retorna con la creación de un tipo de nodo llamado retorno que permite retornar a un punto anterior que el operador puede definir. Ambos software

tienen en común que trabajan sobre un *treeview*, orientan su programación a nodos en forma de árbol no binario[7].

# Capítulo II: Algoritmos y metodologías en la implementación del PhoneMenuDesigner.

A continuación se abordaran los aspectos relacionados con el funcionamiento interno del programa, los formatos y tipos de datos empleados. Se describen los algoritmos y herramienta de programación utilizadas.

## *2.1 Nodos y sus propiedades*

En las aplicaciones de menú telefónicos el usuario accede a servicios que pueden estar organizados en forma jerárquica. Por este motivo se decidió emplear como representación del menú telefónico la estructura de un árbol no binario, donde a partir de un nodo, que representa un estado dentro de la llamada, se pueden recorrer las distintas opciones del menú.

Como representación gráfica del árbol no binario se decidió emplear el *TTreeView*, que es un objeto gráfico del sistema operativo *Windows*, ampliamente utilizado en ese entorno para visualizar las estructuras de directorios de los dispositivos de almacenamiento. Esto evita las complicaciones que implicaría el programar dicha representación visual. Entre las facilidades que brinda el *TTreeView* están la posibilidad de asignar íconos diferentes a cada nodo, desplegar o plegar sus descendientes, así como un fácil movimiento por sus nodos, ya sean el ancestro, los hijos o los nodos adyacentes (hermanos).
A pesar de esto, el *TTreeView* no satisface todas las necesidades de un menú telefónico en cuanto a poder almacenar las configuraciones específicas requeridas de cada nodo. Con este fin se creó una estructura paralela en memoria, equivalente al *TTreeView*, donde los nodos son del tipo *record* del lenguaje Pascal, con campos correspondientes a las distintas características de cada nodo. El tipo de dato se declaró *TNode*, y sus campos se enumeran a continuación:

- Name: (Nombre) de tipo *String*, usada para la identificación del nodo tanto en el árbol como en el *TTreeView*.
- Type: (Tipo) de tipo *TNodeType* (definido en esta aplicación), que permite al programa tomar decisiones sobre las acciones a ejecutar al llegar al nodo. La variedad de tipos de nodos permite gran flexibilidad para implementar los servicios de menú telefónico.
- Digit: (Dígito) de tipo *Integer*. Es el dígito telefónico con el que se activa el nodo, cuando el ancestro (padre) recibe el mensaje de que dicho dígito ha sido pulsado.
- PhoneNumbers: de tipo *String*, Contiene la dirección en donde estará el fichero con los números telefónicos.
- WavFile: de tipo *String*, Contiene la dirección en donde estará el fichero .WAV a reproducir o grabar al entrar al nodo.
- WaitTime: (Tiempo de espera por dígito) de tipo *Integer*. Define cuánto tiempo en segundos el nodo estará esperando un dígito, antes de pasar al hijo por defecto, o colgar si este no existe. Por defecto tiene un valor de 10 segundos.
- WavTime: (Tiempo de espera del fichero .WAV) de tipo *Integer*: Define cuánto tiempo el nodo estará grabando para un fichero .WAV en caso de que no lleguen nuevos cambios de estado en la línea. Valor por defecto de 30 segundos.
- DefaultDigit: de tipo *Integer*. Define cual va a ser su nodo hijo por defecto. En caso que no se desee un hijo por defecto su valor es -1, que es el valor por defecto de este campo (ver *WaitTime*).
- Index: de tipo *Integer*. Contiene el índice del nodo dentro de la lista de hermanos en que se encuentra.
- Father: de tipo *Pointer*. Apunta al ancestro del nodo dentro del árbol.
- Children: de tipo *Array of Pointer*. Contiene los apuntadores a los hijos del nodo.
- StepCount: de tipo *Integer*: Indica la cantidad de saltos hacia arriba en la jerarquía que provoca el nodo al terminar su ejecución. Si no requiere salto su valor es -1, y así se inicializa por defecto.
- WaitRings: de tipo *Integer*: contiene la cantidad de timbres que esperará el nodo para contestar si es contestador, o el umbral en decenas de segundos entre timbres de retorno para considerar que el usuario remoto descolgó si el nodo es llamador.

- TransferIndex: Índice de la variable de transferencia que emplea el nodo para comunicarse con otros (ver epígrafe 2.4).

Algunas de estas propiedades son exclusivamente para ser manipuladas por el programa en tiempo de ejecución (como *Father* y *Children*) otras tienen un objetivo plenamente visual (*Name*) y no se emplean durante la ejecución del servicio, y la mayoría es modificable en tiempo de diseño del servicio y empleada en tiempo de ejecución. En muchos tipos de nodo hay propiedades que resultan innecesarias para su funcionamiento, pero se han mantenido por uniformidad. Por ejemplo, *PhoneNumber* esta especificada para los nodos "llamadores", *WavTime* es usada para el nodo "grabador", *StepCount* es usado para el nodo "retorno". Por otro lado propiedades como *Name*, *WavFile* y *Digit* son usadas por prácticamente todos los nodos

Para el programa se definen dos variables globales del tipo *PNode* (apuntador a *TNode*) que son de gran utilidad. La primera será el nodo que esté seleccionado en el *TTreeView*, útil para seguir la correspondencia entre este y el árbol. La segunda será el nodo que se está ejecutando en la aplicación, y resulta de utilidad en la ejecución del servicio. Las variables se declararon globales debido a que son ampliamente utilizadas por varias funciones y procedimientos, y no deben perder su valor al pasar de un procedimiento a otro.

## *2.2 Salvado y cargado de la configuración de los nodos*

Las configuraciones de los servicios se pueden salvar y cargar en un fichero del tipo .INI[5]. En este existirá una sección para cada nodo. El nombre de la sección en el fichero .INI estará dado por el la secuencia de índices desde el nodo raíz hasta el nodo que representa dicha sección. Las *Keys* (llaves) de cada sección serán los nombres de las propiedades asociadas al nodo (en dependencia de su tipo, ya que no se salvan las propiedades innecesarias), y los valores de estas llaves son los valores de las propiedades correspondientes.

Las propiedades de nombre y dígito asignado son comunes para todos los nodos. Para el nodo "llamador" además de las propiedades comunes es necesario *PhoneNumbers y WavFile*, que también es usada en los nodos "contestador", "reproductor", "reproductor de números", "grabador" colector de hora, colector de dígitos. Para el nodo "grabador" además necesita *WavTime*. Para los nodos colector de dígitos y colector de hora se necesita el *WaitTime*. Para el nodo "retorno" se necesita que *StepCount* tenga un valor menor al de la cantidad de niveles jerárquicos que existan por encima del mismo.

El formato del fichero de configuración permite que la confección de los servicios no sea necesariamente realizada a través de la interfaz de usuario de diseño, sino que puede efectuarse manualmente en forma de texto, ya que los ficheros .INI son fácilmente interpretables con un editor de texto (como el *Notepad*).

## *2.3 Comunicación entre Nodos*

Existen situaciones en que se requiere de determinada transferencia de información entre los diferentes tipos de nodos. A este fin se ha implementado un método de comunicación que consiste en la existencia de una variable global denominada *Transfers* de tipo arreglo de *Strings*. A través del campo *TransferIndex* del *TNode* se le especifica al nodo el índice dentro de *Transfers* de la variable de transferencia que debe usar. No todos los nodos emplean esta propiedad, sólo aquellos que necesiten almacenar información útil para otro nodo, o emplear datos colectados por otro.

## *2.4 Tipos de nodos*

Con los diferentes tipos de nodos se trata de brindar la máxima flexibilidad para la configuración de servicios telefónicos.

*Nodo llamador*: Disca el número especificado en *PhoneNumbers* y reproduce el .WAV asociado en la variable *WavFile*. Aunque el fichero de *PhoneNumbers* puede contener varios números, pero este nodo sólo llama al primero.

*Nodo llamador de lista*: Similar al anterior, pero recorre todos los números contenidos dentro del fichero en *PhoneNumbers*. Dicho fichero debe contener los números en el formato

**hhmm = number**

donde en *hhmm* los dos primeros dígitos representan la hora (militar) y los últimos representan los minutos de la hora en que debe llamarse a *number*. Este nodo resulta de gran utilidad en servicios de llamada a múltiples usuarios. El control de progreso dentro de *PhoneNumbers* se realiza empleando la variable *StepCount*.

*Nodo contestador*: Este nodo espera una llamada entrante, cuando esto ocurre espera la cantidad de timbres especificado en *WaitRings* y entonces descuelga y reproduce el .WAV especificado en *WavFile*.

*Nodo grabador*: Grabar de la línea telefónica hacia el fichero del tipo .WAV especificado en *WavFile*, hasta que el usuario remoto cuelgue o transcurra el tiempo especificado en *WavTime*.

*Nodo reproductor*: La función de este nodo consiste en reproducir el fichero. WAV especificado en *WavFile*.

*Nodo reproductor de números*: Este nodo pronuncia números de varios dígitos. Esta función es necesaria en múltiples casos, siempre que se requiera alguna confirmación del usuario. En este caso *WavFile* contiene la dirección de un .WAV introductorio, mientras que los .WAV individuales de cada dígito se encuentran dentro de la carpeta “Números” que se encuentra en el directorio de la aplicación.

*Nodo colector de dígitos*: Su función es la de guardar dígitos entrados por el usuario. Los dígitos se almacenan en *Transfers[TransferIndex]* a menos que *TransferIndex* sea -1, en cuyo caso se almacena en el fichero referenciado por *PhoneNumbers*. Los dígitos se

almacenan en forma de cadena de caracteres, y la adquisición cesa cuando transcurre *WaitTime* o llega el digito "*". No existe reproducción de .WAV.

*Nodo colector de hora*: Su función es la de guardar dígitos entrados por el usuario que representará una hora dada en formato de hora militar. Almacena cuatro dígitos en forma de cadena en *Transfers[TransferIndex]* a menos que *TransferIndex* sea -1, en cuyo caso se almacena en el fichero referenciado por *PhoneNumbers*. No existe reproducción de .WAV.

*Nodo retorno*: Este nodo es el encargado de mover la posición del nodo activo a un punto anterior. El retorno está medido en pasos jerárquicos hacia atrás, y estos pasos se especificarán en el campo *StepCount*.

*Nodo terminador*: Su objetivo será la terminación de la ejecución del menú. Básicamente cuelga la línea.

## *2.5 Controles KDTeleTools*

Para la confección del software se utilizan una serie de controles de Delphi llamados KDTeleTools cuya función principal es la de encapsular las funciones de TAPI y permitir un ahorro de tiempo en la programación.

La KDTeleTools contiene un grupo de funciones compatibles con Delphi para el desarrollo de aplicaciones. Ella esta dividido fundamentalmente en ocho componentes, los cuales son:

- *KDPhone*
- *KDVoice*
- *KDWaveEditor*
- *KDFax*
- *KDSerialPort*

- *KDFile*
- *KDListen*
- *KDSpeak*

De estas componentes para el software nos enfocaremos en el uso de la *KDPhone* y la *KDVoice* por contener las principales funciones a utilizar en el desarrollo de la aplicación.

La *KDPhone* permite básicamente hacer llamadas y responder las mismas, desconectar llamadas en progreso, detectar tonos DTMF o pulsos que hayan sido presionados en el teléfono por un usuario, generar tonos DTMF, obtener el *caller ID*, controlar el colgado y descolgado del teléfono local, además de soportar características para varias llamadas por línea ya sea llamadas en espera, conferencia etc., estas últimas particularidades no serán usadas.

La componente *KDPhone* responde a eventos de las TAPI. Entre los más importantes está el *OnCallState* que se genera cuando algún estado de la llamada ha cambiado. En uno de los parámetros se pasa una variable que puede tener varios significados en dependencia del estado que haya cambiado. El evento *OnDigitDetected* se invoca cuando las TAPI detectan un dígito por pulsos o DTMF y pasa como parámetro en ASCII el carácter detectado. El evento *OnCallerID* se invoca cuando recibe un *caller ID*, y pasa como parámetros nombre y numero telefónico de la parte remota.

Como ya mencionamos anteriormente el evento *OnCallState* pasa una variable con varios posibles significados. Esta es del tipo *TxCallState*, y tiene dieciocho posibles valores, donde cada valor esta asociado a un cambio de estado diferente en la línea. A continuación mostramos los valores y el estado asociado a ellos.

- *csDisconnected* =0: La llamada ha sido desconectada de la parte remota.
- *csConnected* =1: La llamada saliente ha sido conectada a su destino.

- *csProceeding* =2: La llamada esta procediendo por la red. Esto ocurre después del discado y antes que la llamada esta conectada.
- *csDialing* =3: El teléfono esta discando los dígitos para haces la llamada.
- *csRemotePartyDisconnected* = 4: La parte remota desconecto la llamada.
- *csBusy* = 5: Indica la detección del tono de ocupado.
- *csOffering* = 6: Una nueva llamada le esta siendo ofrecida al dispositivo.
- *csRing* = 7: Es detectado un timbre (El número de timbres es pasado como Param1).
- *csError* = 8: Cuando un error ocurre (El número de error pasado en el Param1).
- *csOfferedCallDropped* = 9: La parte remota ha abandonado la llamada que estaba ofreciendo.
- *csWaitingDisconnection* = 10: El dispositivo telefonico espera desconexion.
- *csNoDialTone* = 11: Cuando no se detecta tono después del discado.
- *csDialTone* = 12: Indica que se detecto tono.
- *csRingBack* = 13: Indica que se detecto un timbrado en la parte remota (ocurre cada vez que timbre).
- *csOnHold* = 14: Indica que la llamada esta en espera.
- *sUnhold* = 15: Indica que la llamada no esta en espera.
- *csInConference* = 16: Actualmente la llamada esta siendo miembro de una conferencia.
- *csRemovedFromConference* = 17: La llamada fue quitada de la conferencia.

- *csWaitingForFaxCommand* = 18: Se espera por un comando de fax antes de establecer la conexión.

La interpretación del *Param1* depende del valor del *OnCallState* Si es *csRing*, *Param1* contiene el número de timbrado. Si es *csConnected Param1* es 0 para conexión de llamadas de voz, 1 para datos y 2 para fax. Si es *csError Param1* contiene el número de error ocurrido.

Otro evento de gran importancia en la componente es *OnDigitDetected*. Este evento se va a generar siempre que sea detectado un digito y devolverá el carácter ASCII del digito detectado.

La componente *KDVoice* permite reproducir y grabar tanto de la línea telefónica como de la tarjeta de audio de la PC, Soporta música de fondo (*Background*), puede reproducir tanto de disco como de memoria, genera eventos de tiempo para chequear el progreso de la grabación y la reproducción, detecta silencio en la grabación, mide en niveles la entrada de audio etc.

Como ya se mencionó, *KDVoice* permite reproducir y grabar audio de la línea telefónica (así como de cualquier dispositivo que lo permita como la tarjeta de sonido). Para reproducir y grabar a través de la línea telefónica es necesario que la llamada telefónica se encuentre en el estado de conectado. *KDVoice* es compatible con los ficheros PCM .WAV y soporta diferentes variantes de frecuencia de muestreo y resolución. Por parte de la KDTeleTools se recomienda que para la reproducción y grabación a través de un *voice modem* se use el formato 8 Khz –16 bit mono[6].

Además de la reproducción y la grabación, la KDTeleTools provee mecanismos para lograr la detección de silencios, lo que es útil para las aplicaciones que necesiten que al detectarse un silencio terminen la grabación como en nuestro caso. Para lograr la detección de silencios se debe habilitar primeramente la propiedad KDVoice.SilenceDetectionEnabled y esperar el evento KDVoice.OnSilenceDetected[6].

## *2.6 Funciones, procedimientos y eventos*

Para la realización del software es necesaria la creación de varios procedimientos y funciones que interactúan de forma directa o indirecta con los nodos tanto del *TTreeView* como los nodos internos del árbol. La necesidad de salvar y cargar las configuraciones hechas por el operador también nos lleva a la necesidad de crear procedimientos para

ejecutar dichas operaciones. Como es lógico existe una alta relación entre la parte visual y los procedimientos internos del programa por lo que gran parte de estas funciones o procedimientos se ejecutan una vez qua se hayan presionados los botones de la interfaz gráfica.

A continuación se brinda una breve descripción de la interfaz de la mayoría de las funciones. Primero se muestra el nombre de cada función o procedimiento, a continuación se brinda su interfaz en Delphi, seguido de los parámetros, valor de retorno de ser una función, y un comentario que describe lo que hace la función o el procedimiento. También se presentan los manipuladores de los eventos que requiere la KdTeleTools.

1. *AddEqNodeInArbol*

**function** AddEqNodeInArbol(Node:TTreeNode) : PNode;

*Parámetros:*

***Node***: Nodo en el *TTreeView*.

*Valor de retorno:*

***Result*** : Apuntador a un nodo en árbol.

*Comentario:*

El procedimiento recibe un nodo del *TTreeView* y agrega un nodo en la variable árbol en la misma posición que el nodo del *TTreeView* que le fue pasado como parámetro. La función devuelve un apuntador al nodo que ella misma creo. Esta función es la usada en la creación de nuevos nodos y subnodos.

2. *DeleteNode*

**procedure** DeleteNode(Node:PNode);

*Parámetros:*

***Node***: Apuntador a un nodo en árbol.

*Comentario:*

El procedimiento recibe un apuntador a un nodo en árbol y su función es borrar el mismo es decir eliminar este nodo de la variable antes mencionada liberando la memoria que ocupada el nodo, si el nodo tiene hijos estos serán eliminados de la misma forma.

3. *GetPath:*

**function** GetPath(Node: TTreeNode) : TIndexPath;

*Parámetros:*

***Node***: Nodo en el *TTreeview*.

*Valor de retorno:*

***Result***: Secuencia de índices de los ancestros hasta el nodo *Node*.

*Comentario:*

Es una función que recibe como parámetro un nodo del tipo *TTreeview* y que devuelve una variable del tipo *TIndexPath* nuevo tipo definido por nosotros. Este arreglo va a contener la dirección desde el nodo principal hasta el nodo que es pasado como parámetro. El valor de la última posición contiene el índice del nodo en el *Treeview*, la antepenúltima posición contiene el índice del padre del nodo que es pasado como parámetro y así sucesivamente hasta llegar a la primera posición que seria el índice del menú principal, como el menú principal esta diseñado de tal forma que no tiene hermanos siempre va a tener como índice 0. Es comprensible entonces que el *TIndexPath* obtenido de cualquier nodo va a tener como valor en la primera posición 0. La función es usada cuando se adicionan los nodos en el árbol o para cualquiera de las actualizaciones en los nodos.

4. *GetArbolPath:*

**function** GetArbolPath(Node: PNode) : TIndexPath;

*Parámetros:*

***Node***: Apuntador a un nodo en árbol.

*Valor de retorno:*

***Result***: de índices de los ancestros hasta el nodo *Node*.

*Comentario:*

Es una función que recibe como parámetro un apuntador a un nodo en árbol y que devuelve una variable del tipo *TIndexPath* que ya explicamos anteriormente. Esta función tiene una utilidad similar a la función anterior solo que su objetivo es obtener la dirección desde el nodo principal hasta el nodo pasado como parámetro a árbol. Esta función es usada en el salvado de los nodos.

5. *SaveTree*

**procedure** SaveTree(Node:PNode;FileIni:TIniFile);

*Parámetros:*

***Node***: Apuntador a un nodo en árbol.

***FileIni***: Fichero de tipo ini.

*Comentario:*

El procedimiento recibe un apuntador a un nodo en árbol y un fichero del tipo ini, su función es el de realizar el guardado de los nodos. A partir del nodo que se le es pasado es guardado este junto con sus hijos en el fichero .INI en el cual las secciones serán la dirección del nodo, y las llaves de cada sección serán las propiedades del nodo.

6. *StrtoTNodeType*

**function** StrtoTNodeType(cadena:String):TNodeType;

*Parámetros:*

***Cadena***: Cadena que tiene una relación directa con un tipo de nodo en especifico.

*Valor de retorno:*

***Result***: Tipo de Nodo.

*Comentario:*

Es una función que recibe como parámetro una cadena. Ella convierte de cadena a tipo de nodo. Necesaria para efectuar actualizaciones de la interfaz gráfica al árbol.

7. *TNodeTypetoStr*

**function** TNodeTypetoStr(NodeType:TNodeType):String;

*Parámetros:*

***NodeType***: Tipo de nodo.

*Valor de retorno:*

***Result***: Cadena que tiene una relación directa con un tipo de nodo en especifico.

*Comentario:*

Es una función que recibe como un tipo de nodo. Ella convierte de tipo de nodo a cadena. Necesaria para efectuar actualizaciones del árbol a la interfaz gráfica.

8. *LoadTree*

**procedure** LoadTree(Node:PNode;FileIni:TIniFile);

*Parámetros:*

***Node***: Apuntador a un nodo en árbol.

***FileIni***: Fichero de tipo ini.

*Comentario:*

El procedimiento recibe un apuntador a un nodo en árbol y un fichero del tipo INI, su función es el de realizar el cargado de los nodos a partir de un fichero .INI pasado como parámetro. Todos los nodos que se encuentran en el .INI serán cargados a partir del nodo pasado como parámetro.

9. *UpdateNodeFromScreen*

**procedure** UpdateNodeFromScreen(TreeNode:TTreeNode);

*Parámetros:*

***TreeNode***: Nodo en el *TTreeview*.

*Comentario:*

El procedimiento recibe un nodo del *TTreeview* y actualiza las propiedades del nodo con los datos existentes en la interfaz grafica.

10. *UpdateScreenAndSelNode*

**procedure** UpdateScreenAndSelNode(TreeNode:TTreeNode);

*Parámetros*:

***TreeNode***: Nodo en el *TTreeview*.

*Comentario:*

El procedimiento recibe un nodo del *TTreeview* y actualiza la interfaz grafica con los datos existentes en las propiedades del nodo y actualiza la variable *SelNode* a *TreeNode.*

*11. ExecuteNode*

**procedure** ExecuteNode(TempNode:PNode);

Parámetros:

TempNode: Apuntador a un nodo en árbol.

*Comentario:*

En este procedimiento cada tipo de nodo realiza la función para la que fueron definidos. A este procedimiento se le pasa como parámetro el nodo a ejecutar. Teniendo en cuenta que los que rigen los cambios necesarios para que existan nuevos nodos a ejecutar son los eventos de cambio de línea o detección de dígitos estos eventos son los que van a llamar al procedimiento.

*12. KDPhone.OnCallState*

Evento KDPhone.OnCallState (CallState: TCallState);

*Parámetros*:

**CallState**: TCallState

*Comentario:*

Este evento se va ha ejecutar cada vez que ocurra un cambio de estado en la línea. Para cada evento se busca que nodo esta en ejecución, dependiendo de ello se realiza una tarea específica. Para todos los eventos que impliquen desconexión de la línea en cualquiera de los dos extremos, se paran todos los procesos de reproducción o grabación que se ejecutan en ese instante y se lleva el nodo de ejecución a la variable raíz en el árbol. Para la señal de timbrado si el nodo en ejecución es contestador se comprueba con el número de timbre definido por el operador en la interfaz gráfica, cuando el número de timbre coincida con el definido previamente se llama a *ExecuteNode* pasándole como parámetro el nodo contestador.

*13. KDPhone.OnDigitDetected*

Evento KDPhone. OnDigitDetected (Digit: Char; CallHandle: Integer);

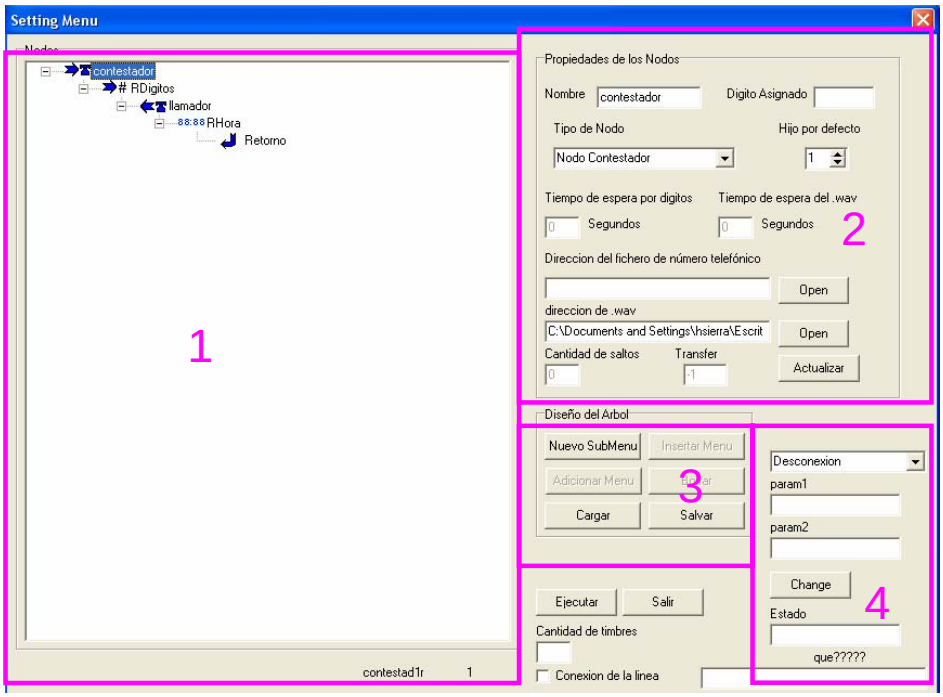
*Parámetros*:
Digit: Contiene el carácter ASCII del dígito detectado.
CallHandle: Manipulador de llamadas usados para los dispositivos que usen sistemas de varias llamadas por línea, de no tener que usar lo anterior se debe ignorar [KDTeleTools]
*Comentario:*
Este evento es ejecutado cada vez que llegue un dígito. Al llegar un digito se busca que nodo esta en ejecución, si este tiene hijos, se busca el hijo que corresponda con el digito marcado por el usuario y se pasa a ejecutar el mismo, esto se logra a través de la propiedad *Digit* de los hijos. De no encontrar ningún hijo que se corresponda con el entrado por el usuario si tiene definido hijo por defecto se pasa a ejecutar este. Lo anterior funciona para nodos de tipo contestador, llamador reproductor, y reproductor de números. Los nodos recolectores de dígito y hora deberán almacenar los dígitos que entre el usuario. Como el nodo retorno cada vez que se ejecuta pasa a un punto anterior nunca estará ejecutándose cuando lleguen dígitos.

Debemos señalar que todos los procedimientos y funciones presentados anteriormente no conforman la totalidad del código fuente de la aplicación a ellos debemos añadir todos los eventos de clic de los botones de la interfaz gráfica, ellos hacen llamados a estos procedimientos.

# Capítulo III: Interfaz grafica del PhoneMenuDesigner.

En este capitulo se aborda cómo funciona el software desde el punto de vista del diseñador de menús telefónicas, o sea, su interfaz de usuario. por fuera, es decir todo lo necesario para que el operador realice sus propias configuraciones, la programación del mismo, quedara claro el uso de los hijos e hijos por defecto en la programación para implementar diferentes servicios. Hemos elaborados algunos ejemplos que le darán claridad al operador de cómo debe implementar sus servicios al clientes. Los temas de este capitulo también se encuentran en la ayuda del programa.

La interfaz de usuario se muestra a continuación, donde se señalan las principales regiones de la misma.




1) Visualización de la estructura jerárquica del menú.
2) Contenido de las propiedades del nodo seleccionado.
3) Edición del menú.
4) Facilidades de simulación.

## *3.1 Configuración de menús*

El programa da la posibilidad de configurar los menús de forma libre, esta configuración estará dada por los servicios que este prestando el programa y las configuraciones que quiera crear el operador en si. Los menús estarán dispuestos en forma de árbol no binario compuestos por nodos que pueden tener a su vez subnodos.

La configuración de los menús se realiza en los botones dentro del recuadro llamado “Diseño del árbol”. Este recuadro posee seis botones donde cada uno tiene una función específica para el diseño del árbol.

### *3.1.1 Adicionar menú*

Permite la adición de un nodo dentro de un mismo padre, es decir permite la adición de nodos hermanos. Cada nodo que es adicionado se pondrá al final de la lista de hermanos. Para que el nodo se adicione en el lugar adecuado uno de los nodos hermanos tiene que ser el nodo previamente seleccionado.

### *3.1.2 Insertar menú*

Este adiciona un nodo en la lista de hermanos solo que no lo pondrá al final sino encima del nodo seleccionado previamente.

### *3.1.3 Adición de submenús*

Permite añadir un menú hijos al nodo seleccionado. Si el nodo seleccionado no tuviera hijos el nuevo submenús creado es el primero y único en su lista de hijos pero de tenerlos el nuevo nodo se adicionaría al final de la lista.

### *3.1.4 Borrar nodos*

Permite borrar los nodos, significa que el nodo seleccionado se borra junto con todos sus nodos hijos de tenerlos. Los nodos siguientes al nodo borrado ocuparan su lugar.

## *3.2 Salvado y cargado de las configuraciones de los menús*

En todo momento dentro del proceso de configuración o después de él, se puede salvar la configuración hecha por el operador. A partir del nodo que sea seleccionado se salva este con sus respectivos hijos, si se quisiera salvar todo el árbol se tendría que seleccionar el menú principal. Al correr esta opción saldrá la ventana clásica de Windows de salvar el fichero, el formato del fichero a salvar es .INI. Conjuntamente con el salvado de los nodos existe también la carga de los mismos cuando se elija esta opción aparecerá la ventana de Windows de abrir fichero y el operador busca el lugar en el disco donde se encuentren el fichero que contiene la configuración deseada. El fichero será cargado en el nodo que este seleccionado en ese momento, si el nodo seleccionado tiene hijos estos también son borrados, es decir se perderían. Si se quiere salvar desde el nodo principal pues solo hay que seleccionarlo y buscar que configuración se quiere salvar.

### *3.2.1 Configuración y refrescamiento de las propiedades de los nodos*

Como se explicó en el capítulo anterior los nodos tienen varias propiedades que son perfectamente visibles y modificables por el operador desde la interfaz gráfica. Para que el usuario pueda ver en todo momento las propiedades asociadas a un nodo en específico, el software muestra dichas propiedades en los cuadros de diálogos correspondientes a las propiedades del mismo, si el operador selecciona otro nodo estas propiedades serán refrescadas en los cuadros de diálogos, mostrando las del nuevo nodo seleccionado.

En todo momento los valores asociados a las diferentes propiedades de los nodos pueden ser cambiados. El primer momento donde estas propiedades pueden ser modificadas es en la creación del nodo, pues si antes de crear el nodo se colocan los valores apropiados en

los cuadros de diálogos correspondientes a cada propiedad, al crearse el nodo, este nace con los valores dados por el operador a sus propiedades. Si por parte del operador es conveniente cambiar una o más propiedades de un nodo ya existente lo que hay que hacer es lo siguiente, primero se selecciona el nodo que desea modificar, segundo colocan en los cuadros de diálogos correspondientes a las propiedades los valores deseados, y como último y tercer paso da un clic en el botón actualizar, una vez hechos estos pasos las propiedades están actualizadas.

## *3.3 Diseño de Menús*

Programar el software es el aspecto más importante para un operador, ello le permitirá la creación de nuevos servicios que se adapten a los pedidos de los más exigentes clientes. El operador tiene que tener en cuenta que cada servicio tiene sus particularidades, debe tener claridad del lugar donde se encuentran los ficheros de audio que usará para sus mensajes. En cada mensaje el operador debe indicar con claridad lo que desea que el usuario haga, un mensaje ambiguo repercutiría en una inapropiada calidad de servicio y un mal funcionamiento de todo el sistema.

Cada nodo como ya se explico tiene su función en específico, su uso esta íntimamente ligado con lo que se quiera lograr por parte del operador. Si nuestra necesidad es hacer una llamada a un número telefónico es necesario poner un nodo llamador, si lo necesario es tener un llamador constantemente según la hora pondremos a un llamador de lista, para contestar llamadas el nodo correspondiente es un contestador, para poner mensajes de audio se necesita un nodo reproductor, para almacenar dígitos y hora se usaran, un nodo recolector de dígitos y un nodo recolector de hora respectivamente. Un nodo retorno seria útil para retornar a un punto anterior.

## *3.4 Mensajes de audio*

Existen varios tipos de nodos que reproducen ficheros .WAV por lo tanto es recomendable que los ficheros se encuentren organizados en una misma carpeta, que este

dentro de la carpeta donde se encuentra el programa. Se sabe que el operador puede tener varias configuraciones almacenadas en ficheros .INI estos ficheros recomendamos que estén en una carpeta dentro de la carpeta del programa. Estas recomendaciones son importantes, pues la falta de organización por parte del operador puede traer como consecuencia que las asignaciones de los ficheros de audio no sean las correctas y que para los cambios de configuraciones sean de manera rápida.

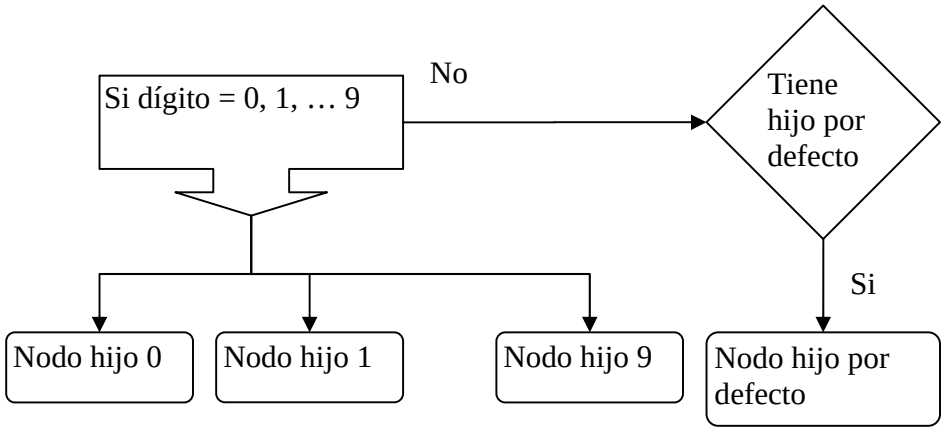
Dentro del programa existe una carpeta llamada Números que contiene diez ficheros cuyos nombres son los números del cero al nueve, el contenido de cada fichero es la reproducción del número que le da nombre. Es importante destacar que esta carpeta no puede estar en otro lugar, si es deseo del operador cambiar la reproducción de los números en las confirmaciones de los nodos recolectores de dígito y hora, o en el nodo reproductor de números, el operador puede poner sus grabaciones siempre que las ponga en la misma carpeta y mantenga los mismos nombres a los ficheros.

## *3.5 Hijos*

Como ya es sabido a cada nodo se le pueden designar varios hijos, esto tiene como objetivo que al usuario que se le esta prestando servicio tenga varias opciones en cuanto a escoger cual de ellos ejecutar. Cuando se tiene varios hijos ellos deben responder a un dígito determinado, así cuando el nodo que se esta ejecutando le llega un dígito pase la ejecución a el nodo hijo que le corresponda ese dígito.

La otra opción es designar un hijo por defecto, cuando este es designado no tiene que llegar ningún dígito para que se ejecute. En caso de que un nodo posea varios hijos y entre ellos un hijo por defecto, si en la ejecución del nodo llega un dígito que corresponda a algún nodo que tenga dígito asignado se ejecuta este, si este dígito no llegara se ejecutara el hijo por defecto.

### *3.5.1 Diagrama lógico para el acceso a los hijos*




## *3.6 Dígitos tecleados por el usuario*

El software es capaz de detectar los dígitos tecleados por el usuario, siendo útil para diversas aplicaciones como la navegación de un usuario por un portal telefónico, configurar el servicio de despertador telefónico etc. En el caso que estos dígitos quieran ser almacenados, existen dos formas de hacerlo como dígitos telefónicos usados para entrar el número de un teléfono por ejemplo o como hora para teclear la hora que quiere ser llamado un usuario por un servicio. Para almacenar en forma de dígitos se pone un nodo recolector de dígitos que estará almacenando dígitos hasta que llegue el digito "*"en caso que lo que se quiere almacenar es una hora se usa un nodo recolector de hora que estará esperando dígitos hasta que lleguen a cuatro dígitos pues la entrada de la hora es en formato militar. Si el usuario dejara de marcar dígitos sin seleccionar el "*" o no terminara de marcar la hora dejándola incompleta el sistema desconectaría la llamada pondría el nodo en ejecución al nodo principal, pasado el tiempo definido por la propiedad tiempo de espera por dígitos asociada al nodo recolector ya sea de hora o de dígitos.

## *3.7 Edición y ejecución*

El software consta de dos ventanas fundamentales, una de edición que es donde se programan los servicios, se le dan valores a las propiedades de los nodos, se carga y se salva configuraciones y la ventana de ejecución que permite ver los servicios que se están ejecutando al mismo tiempo. Por ejemplo si se quisiera montar un servicio de portal telefónico junto con el servicio de despertador telefónico es necesario que estos corran al mismo tiempo pues uno estará guiando al usuario por sus menús y el otro estará encuestando constantemente la hora de llamar a sus usuario, de coincidir la hora de llamar a un usuario con la conexión de otro usuario por el portal, habrá que esperar que el usuario que se encuentra en el portal termine para empezar a realizar las llamadas el servicio de despertador automático.

## *3.8 Ejemplos de servicios*

En esta sección se explica brevemente la implementación de algunos servicios. Esto es para que exista por parte del operador una mayor comprensión y claridad a la hora de implementar cualquier servicio en el software. Todos los servicios que explicaremos están montados de forma individual, pero igual pueden formar parte de un portal telefónico. Es necesario aclarar que el usuario solo tendrá una absoluta claridad de lo que tiene que hacer cuando los mensajes de audio que reproducirán cada uno de sus nodos expliquen claramente los pasos a seguir.

Todos los ejemplos a explicar excepto el despertador automático que por sus características no lo necesita, comienzan con nodos del tipo contestador, están realizados así con la intención de que estos se ejecutaran al llegar una llamada. El nodo contestador responderá cuando la cantidad de timbre coincida con la indicada en la cuadro de dialogo que corresponda a cantidad de timbre en la interfaz gráfica. Si estos servicios formaran parte de portales telefónicos no tendrían nodos contestadores como nodo principal sino que serian una opción más del portal.

### *3.8.1 Servicio de configuración de de despertador automático*

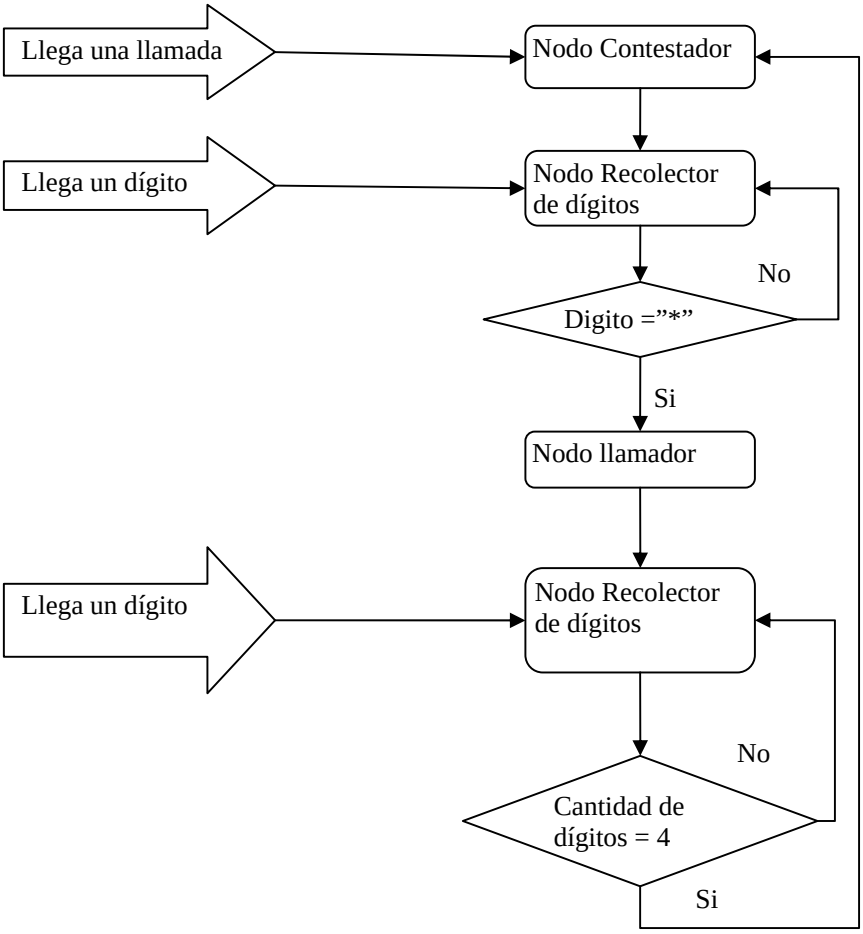
Este servicio tiene como objetivo que el usuario desde su teléfono pueda configurar la hora en la que desee que lo llamen. A partir de que el usuario configura su servicio, será llamado todos los días a la hora indicada. La hora tiene que entrarse en el formato militar. El usuario puede tener más de una vez en el día.

El servicio se brindara de la siguiente forma, una vez que sea accedido el servicio se le pedirá al usuario que entre los números de su teléfono para que sean confirmado por el software, que lo hará llamando al número tecleado, los números se entraran tecleando los dígitos del teléfono terminando siempre con el digito "*", este dígito indicará al software que todos los dígitos fueron entrados. Ya hecha la llamada se pedirá al usuario la hora en la cual en la cual desea que sea llamado, esta hora tiene que entrarse en formato militar. Con la hora ya entrada por el usuario se retornara al nodo principal para poder configura a otros usuarios que accedan al servicio. Tanto para los dígitos como para la hora el software confirmara los dígitos para permitirle al usuario rectificar en caso de equivocarse.

#### 3.8.1.1 Programación del servicio

Para la programación este servicio primeramente cuenta con un nodo reproductor que indique que se ha accedido al servicio, este tiene como hijo por defecto un nodo recolector de dígitos para obtener el número del teléfono del usuario. Este nodo contará con un nodo llamador como hijo por defecto con el objetivo de llamar al número telefónico entrado con anterioridad, este proceso tiene como desventaja que solo se pondrá configurar el servicio desde el propio teléfono del usuario, pero así aseguramos que el teléfono este correcto y evitamos bromas que pudieran molestar a otros propietarios de teléfonos . El nodo llamador tiene como hijo por defecto un nodo del tipo recolector de hora para obtener la hora deseada por el usuario. Y como un nodo retorno como hijo por defecto del recolector de hora volvemos al principio para esperar por otro usuario.

#### 3.8.1.2 Diagrama lógico del servicio




### *3.8.2 Servicio de despertador automático*

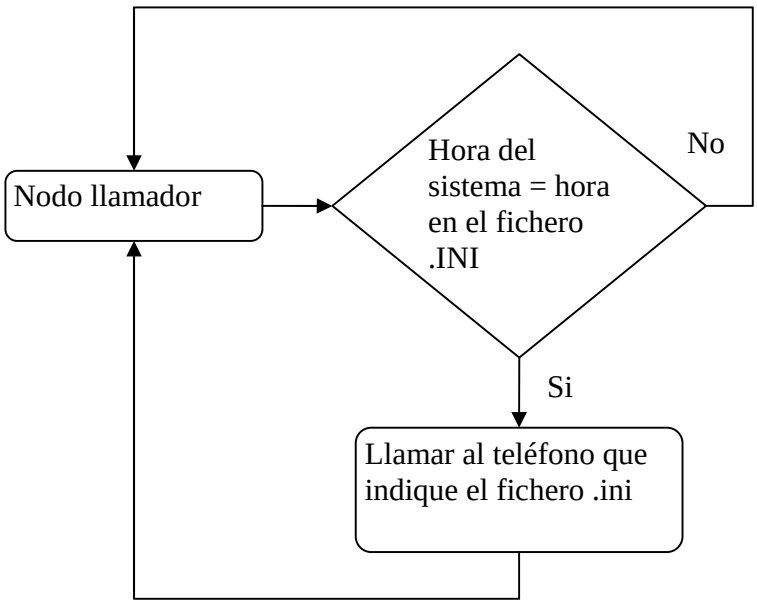
El objetivo de este servicio es que el usuario sea llamado a la hora que se configuró previamente, esta llamada ocurrirá diariamente a la hora indicada. Si cuando se genera la llamada al usuario el teléfono del mismo estuviera en estado de ocupado se asumirá que la llamada fue realizada. Al realizarse la llamada, una vez que el usuario descuelgue para responderla se reproducirá un fichero de audio indicando el servicio, al finalizar este fichero el software colgara y habrá terminado el servicio para el usuario. La ejecución de este servicio consistirá en encuestar constantemente al fichero .ini en donde estarán guardadas las configuraciones de hora y teléfono a quien llamar, cuando la hora de los

usuarios coincida con la hora de la PC se realizara la llamada al teléfono indicado. De tener varios usuarios a la misma hora estos se estarán ejecutando de uno en uno hasta que se agoten.

#### 3.8.2.1 Programación del servicio

Este servicio contara con un nodo llamador de lista que siempre estará en ejecución. El nodo tiene que tener asignado la dirección del fichero .INI en donde se encuentran las configuraciones de hora y teléfono de los usuarios.

#### 3.8.2.2 Diagrama lógico del servicio




### *3.8.3 Contestadora Telefónica*

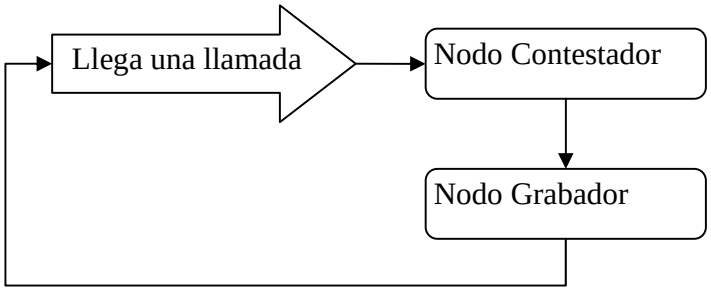
El objetivo de este servicio es atender a todos los usuarios a pesar de no encontrarse ninguna persona para atender la llamada. El software reproducirá un fichero de audio para indicar el servicio y el momento en el que el usuario comenzara a dejar su mensaje. Los mensajes de los usuarios serán grabado y almacenados en algún lugar del disco que se haya escogido previamente. El operador puede revisar en cualquier momento los

mensajes llegados reproduciendo los ficheros de audio que se encuentran en el lugar escogido previamente.

#### 3.8.3.1 Programación del servicio

Cuando se acceda al servicio se accederá a un nodo reproductor indicando que se accedió al servicio este nodo tiene como hijo por defecto un nodo grabador que tiene especificado el lugar de grabación además de los segundos de grabación permitidos, si el mensaje que el usuario esta dejando se pasa del tiempo permitido se parara automáticamente la grabación en ese tiempo, También si existe un tiempo igual a xxx. de silencio se parará la grabación igualmente.

#### 3.8.3.2 Diagrama lógico del servicio




### *3.8.4 Servicio de portales telefónicos*

El objetivo de este servicio es que el usuario pueda navegar desde su teléfono por diferentes opciones puestas por el operador. Cada una de estas opciones puede ser accedida por el usuario tecleando dígitos en su teléfono. Con el uso de nodos retorno el usuario pudiera retornar a puntos anteriores si así lo deseara, también en algún momento pudiera acabar su trayecto y dejar el servicio.

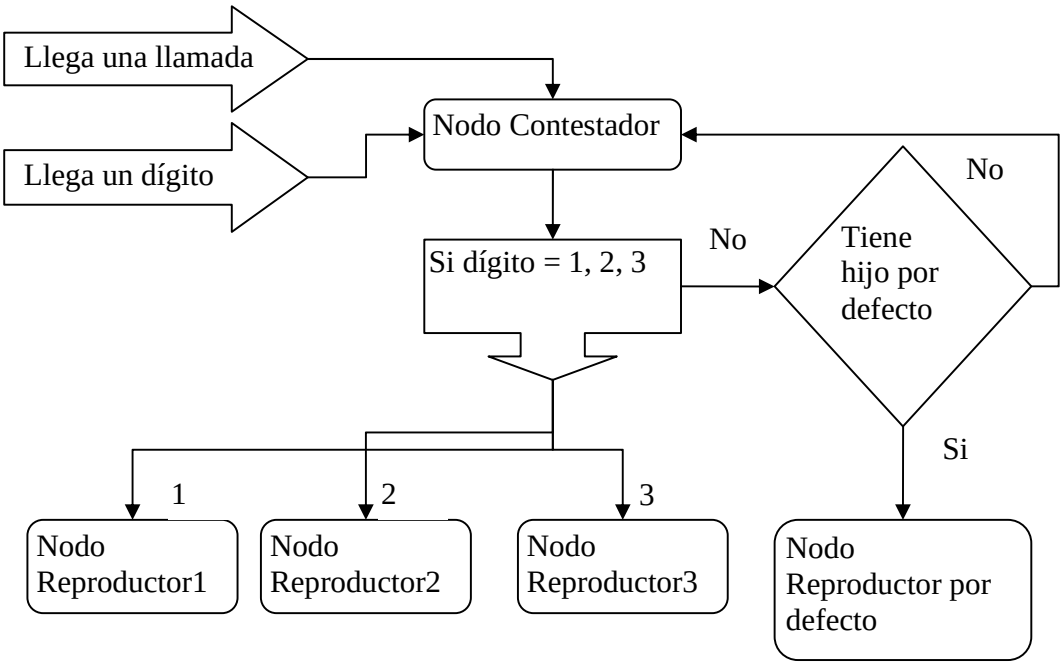
Este servicio es muy abierto, él puede contar con cualquier tipo de nodos, puede tener incluidos otros servicios explicados anteriormente como unas de sus opciones. Nuestro

ejemplo a favor de que sea mas comprensible será bien sencillo, cada vez que entre una llamada se pondrá una mensaje de audio indicando que el servicio esta en ejecución luego el usuario tiene varias opciones que consistirán en otros mensajes de audio. Si el usuario no oprime un dígito de los que se correspondan con las opciones dadas se pasa a un mensaje de audio por defecto.

#### 3.8.4.1 Programación del servicio

El servicio comenzara con un nodo reproductor para atender todas las llamadas entrantes, este nodo tiene varios hijos, el acceso a estos hijos depende del dígito discado por el usuario que esta navegando por el portal en ese momento. Los hijos del nodo contestador son del tipo reproductor y tienen asignado el dígito que le correspondan a la opción deseada por el operador, existe un hijo que no tiene dígito asignado pero en el nodo principal lo tienen asignado como hijo por defecto, para que en caso de discarse una opción mal se ejecute este nodo. Todos los nodos tienen como hijos por defecto un nodo retorno hacia el nodo principal con el objetivo de terminar el servicio.

#### 3.8.4.2 Diagrama lógico del servicio


Llega una llamada
Llega un dígito
Nodo Contestador
Si dígito = 1, 2, 3
No
Tiene hijo por defecto
No
Si
1
2
3
Nodo Reproductor1
Nodo Reproductor2
Nodo Reproductor3
Nodo Reproductor por defecto

# Conclusiones y Recomendaciones

Con la culminación de este trabajo se cuenta con un software de producción nacional que implementa servicios CTI, que puede ser modificado para incluirle mejoras y otras posibilidades. El mismo constituye una opción para las empresas que deseen implementar estos servicios por sí mismas, sin contratar a terceros. Es una primera versión como plataforma principal para brindar el servicio de menús telefónicos, al tiempo que ha permitido dominar la tecnología de las CTI sobre Windows a través de las TAPI.

Se recomienda efectuar modificaciones al programa en cuanto a su interfaz gráfica, como la edición de las propiedades de los nodos en una ventana aparte. Además, se propone separar las aplicaciones de edición de menús y la de ejecución, pasando a esta última el módulo de simulación del servicio.

# Referencias bibliográficas

[1] "For the Telephony API, Press 1; For Unimodem, Press 2; or Stay on the Line" Microsoft Systems Journal, April 1998. http://www.microsoft.com/MSJ/0498/TAPI.htm

[2] Fernando Martín Rodríguez "CTI (Computer Telephony Integration), Integración de Telefonía y Ordenador" visitado 30/3/2004 http://www.gpi.tsc.uvigo.es/~fmartin/CursoCTI/CursoCTI.pdf

[3] Michael C. Amundsen "Introduction to MAPI, SAPI, and TAPI Developer's Guide" visitado 1/4/2004 http://www.soldierx.com/books/Introduction%20to%20MAPI,%20SAPI,%20and%20TAPI%20Developer's%20Guide/index.html

[4] Robert Keith Elias and Alan C. Moore, Ph.D "Extending TAPI Playing and Recording Sounds during Telephony Calls" November 1999 Delphi Informant Magazine

[5] Delphi 6.0 Programmer's Reference, 2001.

[6] KDTeleTools Help File, 2003.

[7] WinIVR Help File, 2001.

[8] Microsoft's Telephoni API (TAPI) Reference, 2001.

# Glosario

| | |
|---|---|
| CTI | Integration Telephony Computer |
| DTMF | Dual-Tone-Multi-Frequency |
| TAPI | Telephony Application Programming Interface |
| CEETI | Centro de Estudio de Electrónica y Tecnologías de la Información |
| WOSA | Windows Open Services Architecture |
| API | Application Programming Interface |
| SPI | Service Provider Interface |
| DLL | Dynamic Library Link |
| PC | Personal Computer |
| PBX | Private Branch Exchange |
| ISDN | Integrated Services Digital Network |
| TSPI | Telephony Service Provider Interface |
| RDSI | Red Digital de Servicios Integrados |