



## **Notas de Aceptación**

---

---

---

---

---

---

---

Firma Jurado

---

Firma Director de  
Proyecto de grado

---

Firma Jurado

---

Firma Coordinador de Ingeniería  
de Sistemas – Tuluá.

Tuluá – Valle del Cauca.

**SISTEMA DE RECONOCIMIENTO DE COMANDOS DE VOZ EN  
ESPAÑOL**



**Andrés Vélez Pérez**

**Johan Christian Martan**

**Universidad del Valle  
Escuela de Ingeniería de Sistemas y Computación  
Ingeniería de Sistemas  
Tuluá - Valle del Cauca  
2008**

# **SISTEMA DE RECONOCIMIENTO DE COMANDOS DE VOZ EN ESPAÑOL**



**Proyecto de Grado presentado por:**

Andrés Vélez Pérez

Cod: \*\*\*\*\*

Johan Christian Martan

Cod: \*\*\*\*\*

**Director de Proyecto de Grado:**

Jaime López Carvajal

MSc. Sistemas

**Universidad del Valle**

**Escuela de Ingeniería de Sistemas y Computación**

**Ingeniería de Sistemas**

**Tuluá - Valle del Cauca**

**2008**

## *AGRADECIMIENTOS*

*Gracias a Dios por darnos destreza e inteligencia. A nuestros padres que con su apoyo y compresión supieron entender los momentos difíciles de nuestra carrera y a todas aquellas personas que de una u otra manera contribuyeron a que lográramos culminar con éxito esta etapa tan importante y trascendental de nuestras vidas.*

## TABLA DE CONTENIDO

Listas de Fórmulas	v
Listas De Figuras	vi
Listas De Tablas	ix
<b>1. Resumen</b>	1
<b>2. Problema de Investigación</b>	2
<b>3. Objetivos</b>	3
<b>3.1. Objetivo General</b>	3
<b>3.2. Objetivos Específicos</b>	3
<b>3.3. Objetivos Estratégicos</b>	3
<b>4. Marco Teórico</b>	4
<b>4.1. Introducción</b>	4
<b>4.2. Generación de la Señal Sonora</b>	6
<b>4.3. Tratamiento de la Señal Sonora</b>	7
<b>4.4. Transformada de Fourier</b>	18
<b>4.5. Coeficientes Cepstrales de Mel</b>	26
<b>4.5.1. Escala de Mel</b>	26
<b>4.5.2. Transformada discreta del Coseno</b>	27
<b>4.6. Reconocimiento de Patrones</b>	28
<b>4.6.1 Enfoques de Reconocimiento de Patrones</b>	29
<b>4.6.2 Problemas de Reconocimiento de Patrones</b>	29
<b>4.6.3 Redes Neuronales</b>	30
<b>4.7. Interpretador de Comandos</b>	47

<b>5.</b> Estado del Arte	48
<b>6.</b> Diseño	50
<b>6.1.</b> Comandos	51
<b>6.2.</b> Red Neuronal	53
<b>6.3.</b> Análisis de Requerimientos	54
<b>6.4.</b> Casos de Uso	55
<b>6.5.</b> Diagrama de Secuencia	58
<b>6.6.</b> Diagrama de Estados	60
<b>6.7.</b> Diagrama de Clases	61
<b>6.8.</b> Diagrama de Componentes	62
<b>7.</b> Desarrollo (Implementación)	63
<b>7.1.</b> Manejo de hilos en Java	65
<b>7.2.</b> Archivos de Configuración	66
<b>7.3.</b> Captura de la Voz	68
<b>7.4.</b> Detección de bordes	69
<b>7.5.</b> Características Espectrales	74
<b>7.6.</b> Red Neuronal	78
<b>7.7.</b> Ejecución del comando	84
<b>7.8.</b> Configuración	87
<b>8.</b> Pruebas	91
<b>9.</b> Conclusiones	95
<b>10.</b> Proyecciones	96
<b>11.</b> Referencias	97
<b>ANEXO A</b> Métodos Abreviados De Teclado De Windows	
<b>ANEXO B</b> Especificación de Requerimientos	

**ANEXO C** Casos de Uso

**ANEXO D** Diagramas de Secuencia

**ANEXO E** Diagramas de Clase

**ANEXO F** Pruebas

## LISTA DE FÓRMULAS

<b>1.</b> Base del Teorema de Nyquist.	11
<b>2.</b> Cruces por Cero.	15
<b>3.</b> Transformada Discreta de Fourier.	19,72
<b>4.</b> Ventana Rectangular.	21
<b>5.</b> Ventana de Hamming.	21
<b>6.</b> Ventana de Hanning.	21
<b>7.</b> Ventana de Parzen.	21
<b>8.</b> Escala de Mel.	26
<b>9.</b> Coeficientes Ceptrales de Mel	27
<b>10.</b> Cálculo de una Neurona.	32
<b>11.</b> Cálculo de una Capa.	32
<b>12.</b> Función sigmoidal.	33
<b>13.</b> Regla Delta.	36
<b>14.</b> Regla Widrow Hoff.	36
<b>15.</b> Perceptrón simple.	38
<b>16.</b> Perceptrón.	38
<b>17.</b> Error de una neurona de la capa de salida.	43
<b>18.</b> Error de una neurona que no este en la capa de salida.	43
<b>19.</b> Error del algoritmo de propagación hacia atrás.	44
<b>20.</b> Función Manigtud	70

## LISTA DE FIGURAS

<b>1.</b> Diagrama del sistema de reconocimiento de comandos de voz.	<b>5</b>
<b>2.</b> Forma de una onda donde el eje horizontal representa la presión media del aire.	<b>8</b>
<b>3.</b> Espectro audible.	<b>8</b>
<b>4.</b> Onda senoidal.	<b>9</b>
<b>5.</b> Representación de la palabra “Hola”.	<b>9</b>
<b>6.</b> Organización de datos de un archivo WAV.	<b>11</b>
<b>7.</b> Diagrama de flujo de un detector de bordes.	<b>15</b>
<b>8.</b> Propiedades de la Transformada de Fourier discreta.	<b>20</b>
<b>9.</b> Respuesta en frecuencia de los diferentes tipos de ventana.	<b>22</b>
<b>10.</b> Corresponde a una función sinusoidal de 200 Hz de frecuencia, muestreada a 11025 Hz y con un ancho de ventana rectangular de 200 muestras.	<b>23</b>
<b>11.</b> Espectro de una señal sinuosidad de 500 Hz de frecuencia y muestreada a 10000 Hz.	<b>23</b>
<b>12.</b> Espectro de una señal sinuosidad de 500 Hz muestreada a 10000 Hz	<b>24</b>
<b>13.</b> Esquema de cómo se forma el espectro variante en el tiempo.	<b>25</b>
<b>14.</b> Banco de Filtros de Mel.	<b>27</b>
<b>15.</b> Enfoques y problemas de reconocimiento de patrones.	<b>30</b>
<b>16.</b> Modelo de neurona artificial.	<b>31</b>
<b>17.</b> Función escalón.	<b>32</b>
<b>18.</b> Red neuronal artificial.	<b>33</b>
<b>19.</b> Funcionamiento de una red neuronal.	<b>35</b>
<b>20.</b> Regla delta (Capa).	<b>36</b>
<b>21.</b> Regla delta (Red).	<b>37</b>
<b>22.</b> El perceptrón.	<b>37</b>

<b>23.</b> Plano de separación en un perceptrón.	38
<b>24.</b> Separabilidad lineal de las funciones AND, OR y XOR.	39
<b>25.</b> Perceptrón.	39
<b>26.</b> Función XOR resuelta con una RN multicapa.	41
<b>27.</b> Tipo de funciones que se pueden resolver con una RN de tres capas.	41
<b>28.</b> Tipo de funciones que se pueden resolver con una RN multicapa generalizada.	42
<b>29.</b> Valores involucrados en la regla delta generalizada.	44
<b>30.</b> Método de gradiente descendente.	45
<b>31.</b> Problemas presentado por los mínimos locales en la convergencia de Aprendizaje de las RN multicapa.	46
<b>32.</b> Problemas de oscilación cuando se toma un factor de aprendizaje Demasiado grande.	46
<b>33.</b> Funcionamiento del interpretador de comandos.	47
<b>34.</b> Diagrama de Paradigma de Prototipos.	50
<b>35.</b> Estructura de datos de la red neuronal.	53
<b>36.</b> Caso de uso captura de voz.	55
<b>37.</b> Diagrama de Secuencia: captura de voz.	58
<b>38.</b> Diagrama de secuencia captura de voz (continuación).	59
<b>39.</b> Diagrama de Estado: Configuración.	60
<b>40.</b> Diagrama de Clases (IcVoz).	61
<b>41.</b> Diagrama de componentes: Ejecutable.	62
<b>42.</b> Menú e Icono en la barra de tareas.	64
<b>43.</b> Ayuda de la aplicación.	64
<b>44.</b> Ordenamiento de los archivos de configuración.	66
<b>45.</b> Diagrama que muestra el proceso de detección. (En grabación).	72

<b>46.</b> Diagrama que muestra como es el proceso de detección (en vivo).	73
<b>47.</b> Diagrama de obtención de los MFCC.	74
<b>48.</b> Grafico de complejidad de FFT.	75
<b>49.</b> Diagrama de banco de filtros de Mel.	77
<b>50.</b> Configuración de IcVoz (Perfil).	87
<b>51.</b> Configuración de IcVoz (Comando).	88
<b>52.</b> Configuración de IcVoz (Edición).	89
<b>53.</b> Configuración de IcVoz (Espera).	89

## **LISTA DE TABLAS**

<b>1.</b> Relación de Razón de muestreo y su respectiva Frecuencia de Nyquist.	12
<b>2.</b> Aplicaciones destacadas en el reconocimiento de voz.	48
<b>3.</b> Publicaciones sobre Reconocimiento de Voz.	49
<b>4.</b> Métodos abreviados de teclado para Windows.	51
<b>5.</b> Métodos abreviados de teclado para algunas aplicaciones.	51
<b>6.</b> Comandos del interpretador.	52
<b>7.</b> Análisis de Requerimientos.	54
<b>8.</b> Caso de Uso captura de voz.	56
<b>9.</b> Prueba realizada al interpretador. (Usuario de la prueba: Andrés Vélez Pérez).	92
<b>10.</b> Prueba realizada al interpretador. (Usuario de la prueba: Christian Martan).	93
<b>11.</b> Estadística General de la prueba aplicada a 10 Usuarios.	94

## **1. RESUMEN**

Este es un proyecto de reconocimiento de comandos de voz para el idioma español, que consiste en la conversión de voz en texto que representa un comando y su posterior ejecución. Este proyecto vincula varias áreas del conocimiento, como son el reconocimiento de patrones, tratamiento de señales de voz, matemáticas y redes neuronales artificiales. Se inicia abordando el estudio de identificación de los métodos utilizados en el reconocimiento del habla humana, mediante las técnicas de tratamiento de señales de audio, útiles en el procesamiento de los comandos pronunciados por el usuario.

Los comandos de voz humana se obtendrán por medio de un micrófono, el cual entregará una señal análoga que se transforma a señal digital mediante el uso de conversor analógico/digital. Esta señal digital se le realiza un pre-procesamiento, se le pasa un algoritmo de detección de bordes y luego se extrae las características de cada comando usando la transformada de Fourier y los Coeficientes Ceptrales de Mel, que es usado por la red neuronal artificial para entrenamiento y producción según sea el caso. Fundamentalmente se obtuvo un interpretador que permite ejecutar veinte comandos a través de la voz en lugar de usar el teclado con un porcentaje de reconocimiento del 75%.

## **2. PROBLEMA DE INVESTIGACIÓN**

Durante el desarrollo de este proyecto se pretende mejorar el uso de los computadores a las personas con inhabilidades físicas, las cuales, por su discapacidad presentan dificultades a la hora manejar un ordenador. Es por esto, que mediante comandos de voz se les brindará la oportunidad de acceder de una forma más fácil. Este interpretador también puede ser utilizado por todas aquellas personas que no sufren discapacidad, permitiéndoles contar con una herramienta útil para ampliar la interacción hombre-máquina.

Un sistema de reconocimiento de voz implica fusionar información que procede de diversas fuentes (acústica, fonética y fonológica), en presencia de ambigüedades, incertidumbres y errores implícitos para obtener una interpretación aceptable del mensaje acústico recibido.

Un sistema (interpretador de comandos) de éste tipo requiere entrenamiento antes de empezar a usarlo, dado que cada usuario (hablante) posee su propia pronunciación, por lo tanto se debe configurar con anterioridad con las palabras del hablante ya establecidas. En la ejecución del interpretador se busca que este sea capaz de distinguir palabras completas, es decir, que no se de un corte incorrecto de la palabra, también se debe tener en cuenta la robustez del interpretador, ya que necesita de un reconocimiento claro de la voz humana en presencia del ruido ambiental, el cual puede generar resultados erróneos en el interpretador (es decir, no reconocimiento de los patrones), además se debe recordar la capacidad del interprete de reconocer una gran cantidad de comandos, por lo que usualmente puede disminuir el rendimiento del interpretador.

### **3. OBJETIVOS**

#### **3.1 Objetivo General**

Desarrollar un interpretador de comandos de voz utilizando redes neuronales artificiales, dando acceso a personas con inhabilidades físicas mejorando la interacción hombre-máquina.

#### **3.2 Objetivos Específicos**

- Seleccionar los comandos de voz que con más frecuencia son ejecutados por el usuario a través del uso del teclado.
- Aplicar las técnicas de procesamiento de señales de voz para la obtención de los patrones.
- Diseñar y entrenar una red neuronal como la técnica de reconocimiento de patrones.
- Implementar un interpretador que reconozca los comandos de voz.
- Realizar pruebas al interpretador de comandos de voz.

#### **3.3 Objetivos Estratégicos**

- Reconocer con alta precisión (mayor del 70%) los comandos de voz pronunciados por el usuario.
- Permitir un mínimo de un usuario en el interpretador.
- Reconocer el comando de voz del usuario con una baja relación señal ruido.

## **4. MARCO TEORICO**

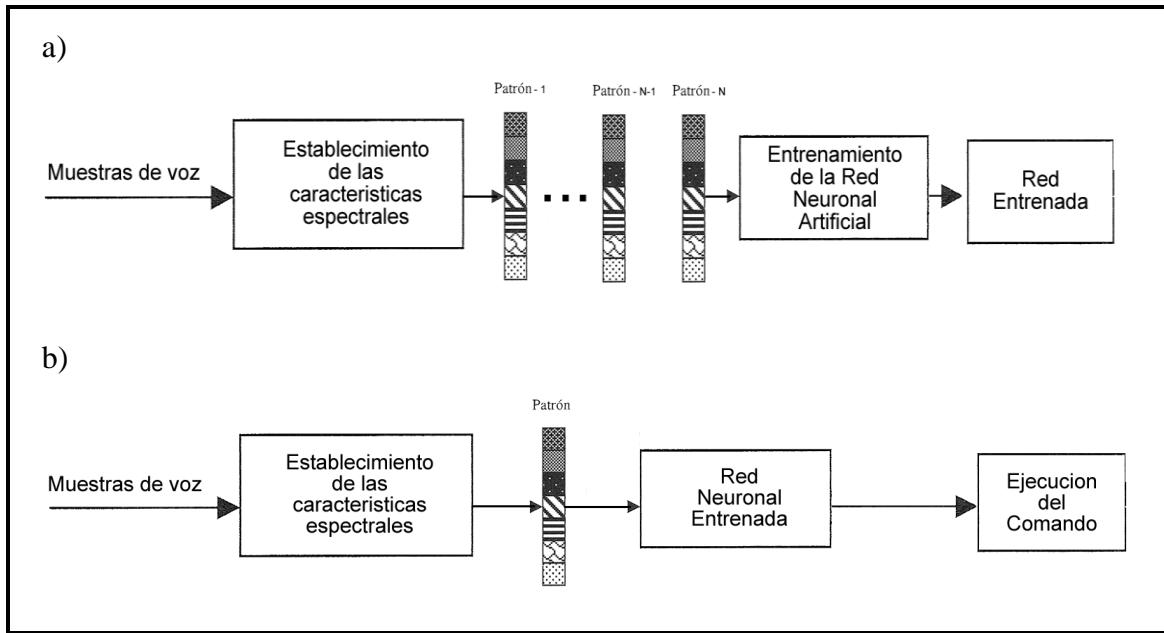
### **4.1. Introducción**

En este documento se presenta un sistema de reconocimiento de comandos de voz en español usando redes neuronales artificiales; En este capítulo se presenta un marco teórico de los aspectos más importantes que componen este sistema como son la generación de la señal, tratamiento de la señal y reconocimiento de patrones.

Para el sistema de reconocimiento de comandos de voz, los patrones se obtienen seleccionando grupos significativos del lenguaje que se pretenden reconocer. Estos grupos pueden estar constituidos por unidades tales como sonidos básicos (correspondientes a fonemas y alófonos), difonemas, demialófonos, palabras, etc. El grupo seleccionado para este proyecto es el de las “palabras”, se realizarán grabaciones de los comandos de voz (con un contexto asociado) y se obtendrán sus características espectrales (valores obtenidos aplicando la Transformada de Fourier y los Coeficientes Cepstrales de Mel) para realizar la fase de entrenamiento en la red neuronal artificial y de esta forma permitir asociar al patrón reconocido cuando ejecute el comando pronunciado por el usuario.

En el reconocimiento de patrones se utilizará las redes neuronales artificiales, las cuales necesitan un proceso de entrenamiento para que reconozcan los comandos pronunciados por el (o los) usuario(s) del intérprete.

En la figura 1a. se muestra un diagrama de entrenamiento de una red neuronal para el sistema de reconocimiento de comandos de voz en español, el cual indica que se deben obtener unas muestras de voz (grabaciones de audio), correspondientes a los comandos de voz, extrayéndole así las características espectrales a cada comando, para con ellas entrenar la red, la cual va hacer utilizada después por el sistema de reconocimiento.



**Figura 1.** Diagrama del sistema de reconocimiento de comandos de voz.

**a)** Sistema de Entrenamiento. **b)** Sistema de Producción.

En la figura 1b. se da a conocer un diagrama en el que se muestra las etapas involucradas en el reconocimiento de comandos de voz utilizando redes neuronales, en la cual los comandos de voz ya están establecidos y la red ha sido entrenada.

## **4.2. Generación de la señal sonora**

El proceso de producción de la señal sonora por el ser humano consiste en mecanismos biológico-articulatorios que son: respiración, fonación y articulación (Álvarez, 1991).

**4.2.1. Respiración:** Durante este proceso se llevan a cabo dos subprocessos importantes denominados inspiración y espiración del aire, siendo este último la materia prima de la fonación. Para que este mecanismo se cumpla, existe un conjunto de órganos encargados de ejercer tales funciones, como es el aparato respiratorio, compuesto de: fosas nasales, laringe, tráquea, bronquios, bronquiolos y pulmones.

Los sonidos que se dan en el momento de hablar son de carácter espiratorio, exceptuando algunos sonidos inspirados de algunas lenguas, muy diferentes a los del idioma español que se producen cuando el aire inicia su proceso de salida.

**4.2.2. Fonación:** Consiste en el hecho mismo de producir la voz y es una de las formas de alterar el rítmico proceso respiratorio. La voz se debe a la vibración de las cuerdas vocales, que se ve modificada por la forma y el volumen de las cavidades, faríngea y bucal. Como las cuerdas vocales se hallan en la laringe a ésta se la ha denominado “Órgano de la fonación”.

**4.2.3 Articulación:** Constituye el tercer factor del proceso completo y consiste en la forma como se organizan los órganos activos y pasivos en el momento del paso de la columna de aire. Los articuladores activos son: maxilar inferior, labios, lengua, velo del paladar y cuerdas vocales. Los articuladores pasivos son: dientes superiores, alvéolos, paladar duro y cavidad nasal. El contacto entre dos activos o entre un activo y un pasivo, determina un punto de articulación que diferencia un sonido de otros.

La voz son ondas sonoras producidas en la laringe por la salida del aire (espiración) que, al atravesar las cuerdas vocales, las hace vibrar. La voz se define en cuanto a su tono, calidad e intensidad o fuerza. El tono óptimo o más adecuado para el habla, al igual que su rango de variación, depende de cada individuo y está determinado por la longitud y masa de las cuerdas vocales. Por tanto, el tono puede alterarse, variando la presión del

aire exhalado y la tensión sobre las cuerdas vocales. Esta combinación determina la frecuencia a la que vibran las cuerdas: a mayor frecuencia de vibración, más alto es el tono.

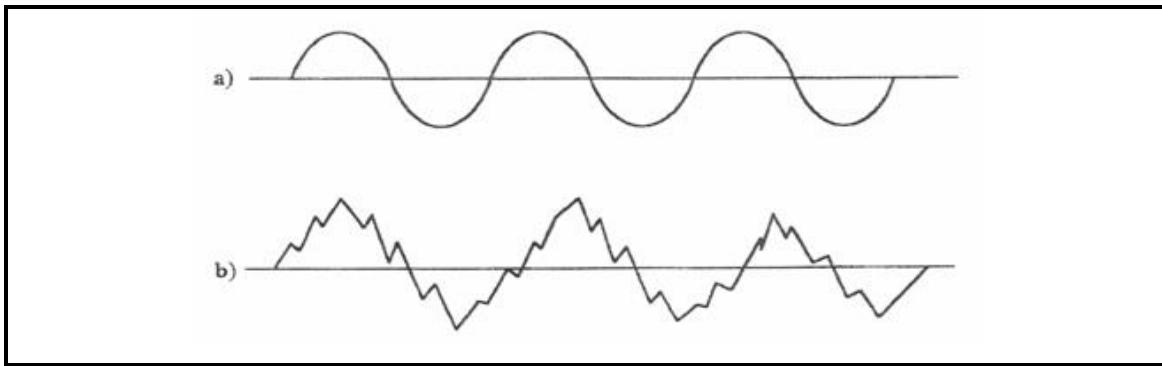
Otro aspecto de la voz es la resonancia. Una vez que ésta se origina, resuena en el pecho, garganta y cavidad bucal. La calidad de la voz depende de la resonancia y de la manera en que vibran las cuerdas vocales, mientras que la intensidad depende de la resonancia y de la fuerza de vibración de las cuerdas.

#### **4.3. Tratamiento de la señal sonora**

Como se mencionó anteriormente, el habla es producida por el órgano fonador humano, ya que la voz son ondas de sonido que representan variaciones de presión del aire por encima y por debajo de un nivel medio y a una velocidad determinada.

Una onda de sonido con una frecuencia específica está representada por una senoide, como la que aparece en la figura 2a, en la cual una alternancia positiva y una alternancia negativa constituyen un ciclo completo. Tales ondas se propagan con velocidad lineal uniforme, que depende del medio a través del cual se propaga y de la temperatura del medio.

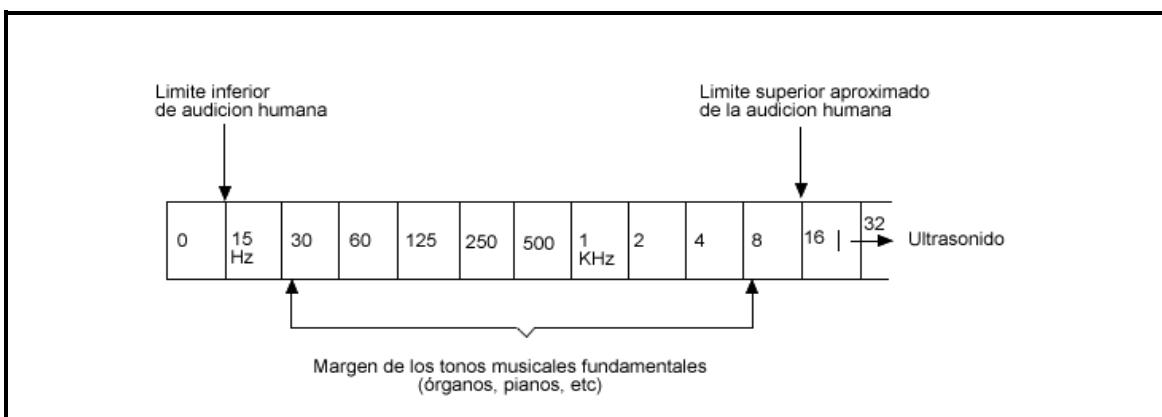
El timbre de un sonido depende en general de su frecuencia de vibración. El sonido es una mezcla muy compleja de frecuencias como se muestra en la figura 2b, a diferencia de un tono puro de una sola frecuencia como se muestra en la figura 2a. El espectro de frecuencia audible (figura 3) se extiende desde aproximadamente 15Hz hasta los 16KHz. Por debajo de 15Hz las vibraciones se perciben por el tacto en vez del oído y por encima de 16KHz el grado de audición depende de la edad del oyente.



**Figura 2.** Forma de una onda donde el eje horizontal representa la presión media del aire.

a) Tono puro. b) Onda típica de voz.

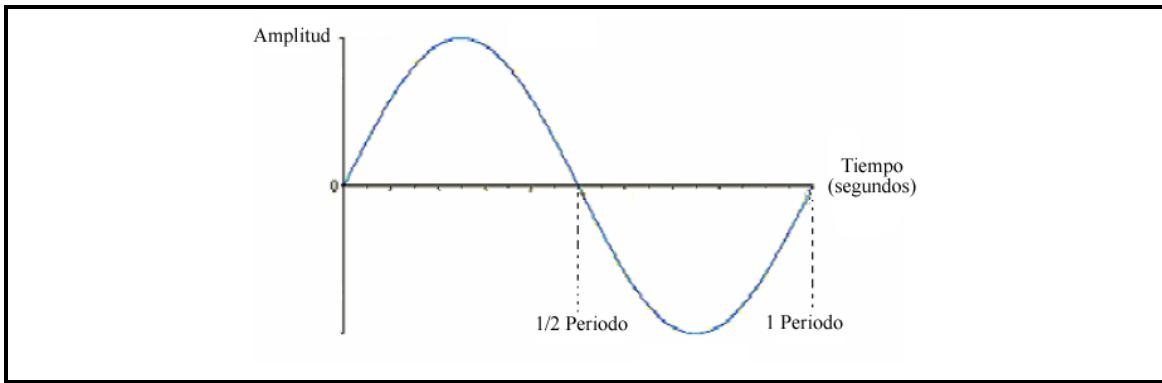
(Tomasi, 1996)



**Figura 3.** Espectro audible.

(Mandl, 1982)

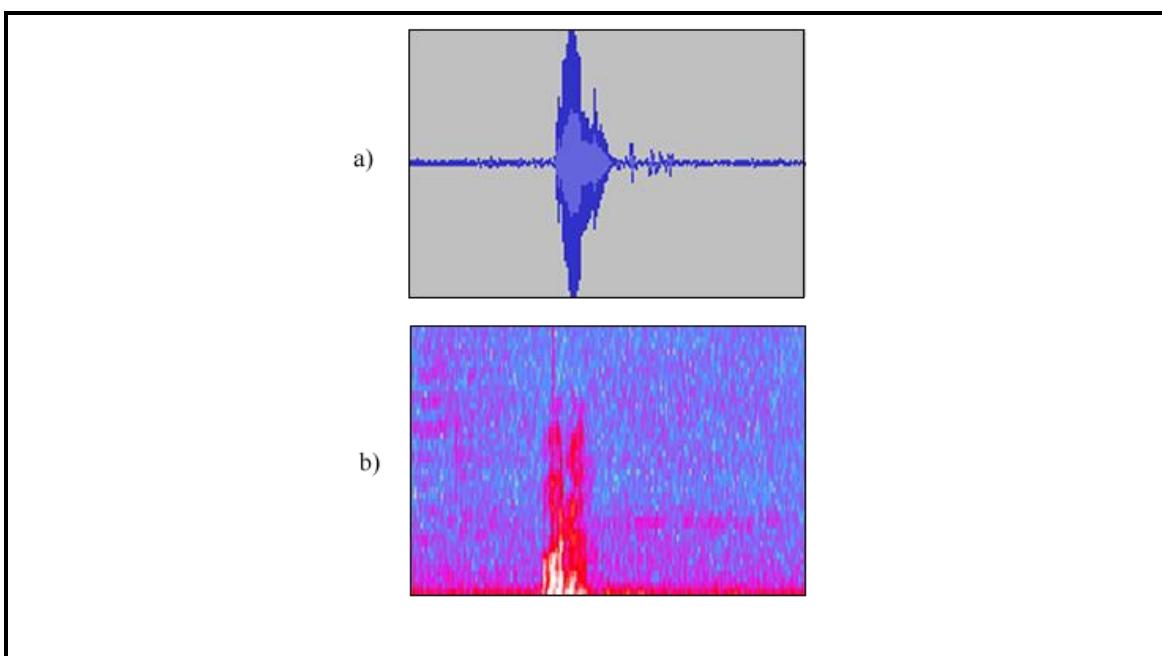
En esencia, el análisis de señales consiste en un análisis matemático de la frecuencia, amplitud, y fase de una señal, donde la frecuencia es el número de veces que se repite en un segundo cualquier fenómeno periódico; la amplitud de una onda representa el valor máximo que alcanza la perturbación en un punto y la fase es una medida de la posición relativa de la señal dentro de un período de la misma. Estas características son representadas por ondas seno o coseno. Las ondas periódicas pueden analizarse ya sea en el dominio del tiempo o en el dominio de la frecuencia. La onda es periódica si se repite en intervalos de tiempo fijos llamados períodos (Figura 4).



**Figura 4.** Onda senoidal.

El dominio del tiempo: La forma de onda de una señal muestra la forma y la magnitud instantánea de la señal con respecto al tiempo pero no necesariamente indica su contenido de frecuencia (Figura 5a).

Dominio de la frecuencia: Esta no se muestra como una forma de onda, en vez de esto se muestra una gráfica de la amplitud contra la frecuencia pero no necesariamente indica la forma de onda o la amplitud combinada de todos los componentes de entrada de información en un tiempo específico (Figura 5b).



**Figura 5.** Representación de la palabra “Hola”.  
**a)** En el dominio del tiempo. **b)** En el dominio de la frecuencia.

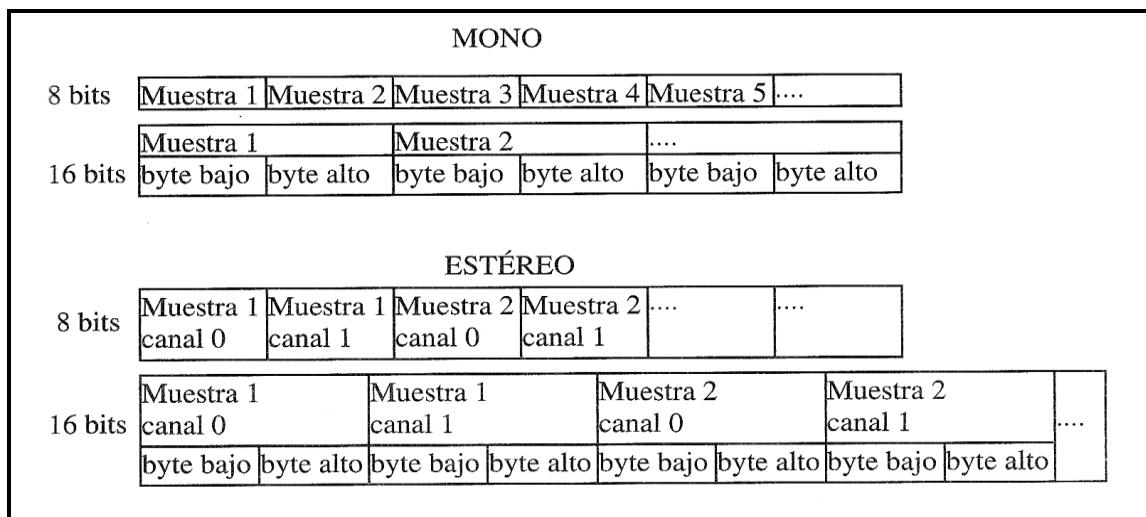
Para la obtención de la voz humana se requiere llevar a cabo un muestreo a través de un convertidor analógico/digital (micrófono y tarjeta de sonido del computador), mientras que los valores numéricos de cada muestra (que indican amplitudes de la señal) se almacenan en la memoria (RAM) del computador para ser procesadas por un software en tiempo real (como el sistema de reconocimiento de comandos de voz). En caso de ser almacenadas en un archivo, resulta conveniente que se utilice algún estándar de almacenamiento de archivos de voz (formato WAV), con el fin de poder utilizarlos con diferentes programas.

A continuación se describe el formato de un archivo WAV, propiedad de Microsoft. El archivo está formado por dos bloques: la cabecera y las muestras digitalizadas del sonido. La cabecera tiene un tamaño de 44 bytes y contiene el tipo y organización de las muestras. Su contenido es el siguiente (Bernal, Bobadilla y Gómez, 2000):

- Campo 1: bytes 0 .. 3. Contiene la palabra "RIFF" en código ASCII.
- Campo 2: bytes 4 .. 7. Tamaño total del archivo en bytes menos 8 (no incluye los dos primeros campos).
- Campo 3: bytes 8 .. 15. Contiene la palabra "WAVEfmt " en código ASCII (Fijarse que hay un blanco después de la t).
- Campo 4: bytes 16 .. 19. Formato: para PCM vale 16.
- Campo 5: bytes 20 .. 21. Formato: para PCM vale 1.
- Campo 6: bytes 22 .. 23. Se indica si es mono (1) o estéreo (2).
- Campo 7: bytes 24 .. 27. Frecuencia de muestreo, puede valer: 11.025,22.050 o 44.100.
- Campo 8: bytes 28 .. 31. Indica el número de bytes por segundo que se debe intercambiar con la tarjeta de sonido para una grabación o reproducción.
- Campo 9: bytes 32 .. 33. Número de bytes por captura, pueden ser 1, 2 o 4.
- Campo 10: bytes 34 .. 35. Número de bits por muestra, pueden ser 8 ó 16.
- Campo 11: bytes 36 .. 39. Contienen la palabra "data" en código ASCII.
- Campo 12: bytes 40 .. 43: Número total de bytes que ocupan las muestras.

Algunos campos contienen información redundante. El campo 8 se puede calcular mediante: (campo 7)\*(campo 9) o (campo 6) \* (campo 7)\*(campo 10)/8, ... Si la captura es mono y de 16 bits por muestra el campo 9 debe valer 2 y así para los demás campos.

Cuando la grabación es en estéreo se almacena de forma alternada, una muestra de cada canal.



**Figura 6.** Organización de datos de un archivo WAV.  
( Bernal, Bobadilla y Gómez, 2000)

Cuando se captura en 8 bits por muestra el tipo de dato leído es byte, es un tipo sin signo con un rango de 0 a 255; en este caso el silencio acústico se encuentra en el valor 128. Éste no es un tipo adecuado para operar matemáticamente, por ello se debe realizar un desplazamiento para que el silencio se localice en el valor 0.

Antes de realizar la grabación o la obtención de muestras se debe definir la tasa de muestreo para grabar el sonido, de acuerdo al teorema de Nyquist (Teorema de muestreo), este establece que "una señal analógica puede ser reconstruida, sin error, de muestras tomadas en iguales intervalos de tiempo. La frecuencia de muestreo debe ser igual, o mayor, al doble del ancho de banda de la señal analógica" (Martínez, 2006). La teoría del muestreo define que para una señal de ancho de banda limitado, la frecuencia de muestreo ( $f_m$ ), debe ser mayor que dos veces su ancho de banda [B] medida en Hertz [Hz].

$$f_m > 2 \cdot B \quad (1)$$

Suponiendo que la señal a ser digitalizada es la voz, el ancho de banda de la voz es de 4000 Hz aproximadamente, por lo cual, su frecuencia de muestreo será  $2*B = 2*(4000 \text{ Hz})$ , es igual a 8000 Hz, equivalente a 8000 muestras por segundo. Entonces la frecuencia de muestreo de la voz debe ser de al menos 8000 Hz, para que pueda regenerarse sin error.

La frecuencia  $2*B$  es llamada la frecuencia de muestreo de Nyquist. El teorema de muestreo fue desarrollado en 1928 por Nyquist y probado matemáticamente por Shannon en 1949.

La frecuencia de muestreo determina la calidad del sonido grabado. Algunas de las frecuencias de muestreo utilizadas para grabar música digital son las siguientes:

Razón de muestreo	Frecuencia de Nyquist
22,050 KHz	11,025 KHz
24,000 KHz	12,000 KHz
30,000 KHz	15,000 KHz
44,100 KHz	22,050 KHz
48,000 KHz	24,000 KHz

**Tabla 1:** Relación de Razón de muestreo y su respectiva Frecuencia de Nyquist.  
( Martínez, 2006)

Antes de utilizar la transformada de Fourier u otra técnica diferente para la extracción de un patrón, se requiere determinar el inicio y fin de cada palabra, para eso se usan técnicas de detección de bordes, que resultan útiles para reconocer palabras aisladas en un sistema de reconocimiento de comandos de voz.

En el reconocimiento automático del habla, la dificultad depende del tipo de objetos acústicos que se trate de aislar. Idealmente, sería deseable utilizar objetos relacionados con categorías lingüísticas correspondientes a percepciones del mensaje oral como, por ejemplo, los fonemas, sílabas, palabras o frases. No obstante, la manifestación física de estos objetos en la señal vocal suele ser muy compleja y siempre relacionada con gran cantidad de conocimientos a priori sobre la fonología, léxico y gramática de la lengua. Por esta razón se suelen introducir otros objetos, más o menos relacionados con los anteriormente mencionados, pero cuya manifestación en la señal vocal es más directa y

menos relacionada con los conocimientos a priori sobre la lengua. Algunos de estos objetos son los siguientes:

- Segmentos de señal de vocal o de silencio: separación de palabras o frase de fondo que las rodea (detección de bordes)
- Segmentos sonoros o sordos: segmentos de señal en los que existe periodicidad o en los que no existe.
- Segmentos vocálicos o consonánticos: relacionados con picos o valles de la energía dependiente del tiempo de la señal vocal.
- Microfonemas: elementos acústicos obtenidos directamente del submuestreo temporal de alguna representación paramétrica del habla (o definidos por la sucesivas posiciones de la ventana de análisis dependiente del tiempo)
- Difonemas: intervalo comprendido entre los puntos centrales de dos segmentos estacionarios de señal. Idealmente se asimilan a parejas de “semifonemas” (mitades de fonemas).
- Pseudofonemas: elementos asimilables mas o menos directamente con los fonemas de una lengua

#### **4.3.1. Detección de bordes**

La detección de bordes suele afrontarse en dos etapas diferenciadas: “detección gruesa” y “detección fina”. La detección gruesa conviene llevarla a cabo en tiempo real, en combinación con el proceso de adquisición de la señal vocal.

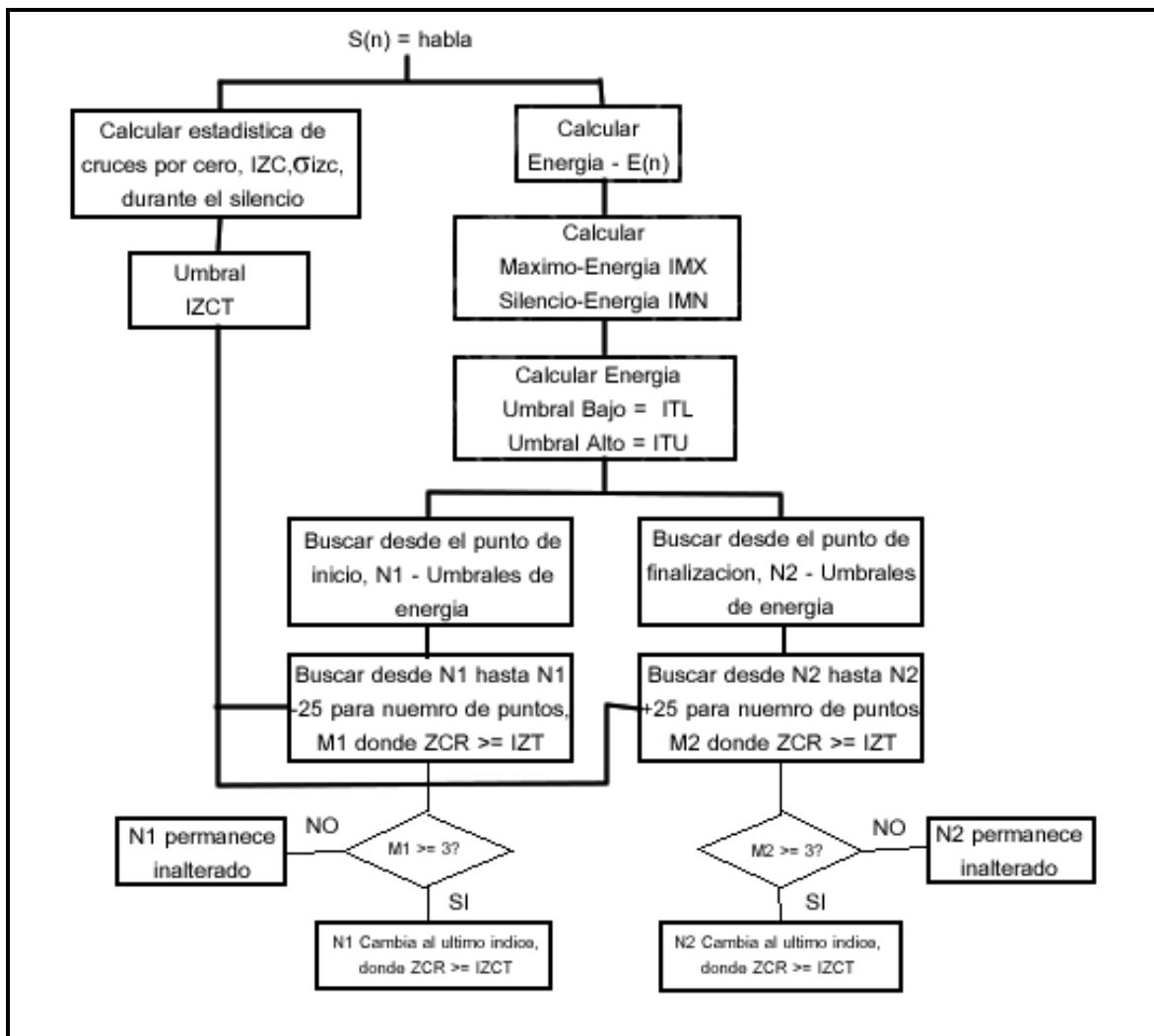
Un algoritmo simple y eficaz para la detección gruesa de bordes, válido si el ruido no es excesivo, utiliza umbrales de tiempo y energía (o amplitud) de la señal: la superación de estos umbrales indica la presencia de palabras. El “umbral de tiempo” resulta necesario debido a que no es suficiente afirmar que hay palabra cuando hay energía: ello llevaría a tomar como palabra un ruido espurio (ruido ambiental). Una palabra presenta siempre un mínimo de energía durante un mínimo de tiempo; y viceversa, la ausencia de energía no implica fin de palabra, puesto que se terminaría al encontrar el silencio que forma parte de un fonema explosivo (/p/, /t/, /k/, ...). Una zona de silencio real tiene un mínimo de duración por otra parte, es necesario considerar también como parte de la palabra el

tramo de la señal inmediatamente anterior y posterior al momento del paso por el umbral de energía, pues, en caso contrario, podría perderse los principios y/o finales de la palabra constituidos por señal de energía inferior al umbral (nasales, fricativas, débiles, iniciales y finales etc.). Para la realizar la detección gruesa en el tiempo real, se requiere una memoria tapón (buffer) cíclica que permita trabajar a “micrófono abierto”, es decir con tiempos de silencio iniciales indefinidamente largos. En este buffer se va almacenando cíclicamente la señal (parametrizada o no), con lo que en cada momento t se dispone de la señal adquirida desde el instante  $t - Tb$  hasta t, dependiendo Tb del tamaño de la memoria cíclica. De esta manera en el momento en que se detecta el “principio eficaz” de palabra, se dispondrá aun del segmento de señal inmediatamente anterior en el que esta contenido el principio real de la palabra. El tamaño de la memoria cíclica dependerá de la longitud temporal del segmento inicial anterior al “principio eficaz” que sea necesario conservar.

La detección gruesa asegura que la palabra o frase pronunciada quedará contenida en su totalidad entre las fronteras detectadas, lo que resulta eficiente en muchos casos. No obstante, un procedimiento mucho más fino conduciría no solo a una mejora de los resultados de una eventual etapa subsiguiente de reconocimiento, sino a un indudable ahorro de memoria utilizada para el almacenamiento de la señal y de tiempo a emplear en los tratamientos posteriores. La detección fina se suele basar en los parámetros extraídos para los tratamientos posteriores (reconocimiento). A partir del resultado proporcionando por la detección gruesa, el algoritmo de detección fina debe buscar de atrás hacia adelante, a partir del momento en que se cruzó inicialmente el umbral de amplitud, el punto donde los parámetros extraídos indican la existencia (fina) de silencio. Para el fin de palabra, también de atrás hacia adelante a partir del final “gruesa” se procede de forma análoga hasta encontrar el momento en que deja de haber silencio. La existencia o no de silencio se determina a partir de los parámetros utilizados, y la manera de decidir dicha existencia depende obviamente de la naturaleza de esta. Un método simple consiste en utilizar alguna medida de similitud disponible para los vectores de parámetros empleados, y establecer (mediante aprendizaje) un patrón de silencio con su correspondiente umbral de similitud (esto equivale a una función discriminante lineal). Con estas premisas la decisión de existencia (o no) de silencio corresponderá a valores de similitud superiores (o inferiores) al umbral de similitud (Casacuberta y Vidal, 1987).

#### 4.3.1.1. Algoritmo Propuesto

Un algoritmo de detección de bordes sencillo de realizar (Rabiner y Sambur, 1975). Se ilustra en la figura 7.



**Figura 7.** Diagrama de flujo de un detector de bordes.  
(Rabiner y Sambur, 1975)

Este algoritmo basa sus cómputos en promedios temporales (mediante una ventana de tiempo rectangular) del valor absoluto de la señal y de los cruces por cero (es decir, las veces en que cambia de signo la señal con respecto a su media). Debe tenerse en cuenta que el promedio de cruces por cero es un estimador de la frecuencia fundamental (aunque algo rudimentario) que basta para el análisis de comienzo y fin de la palabra. Este estimador es el que ayudará a determinar el comienzo y fin de la palabra cuando la misma empieza o termina con sonidos de baja energía y alto valor de frecuencia, como

es el caso de las fricativas 'f', 's', 'y', 'z', etc. En esos casos, un análisis único de la función magnitud truncaría en forma incorrecta el sonido; lo que se hace es examinar la función de cruces por cero a partir de los instantes donde sería truncado por la función magnitud, hacia el comienzo (en el caso de la determinación del principio de la palabra) y hacia el final (en el caso de determinación del final de la palabra) del vector de sonido. Si se observa un cambio radical en los cruces por cero (aun en la zona de baja energía) esta será propuesta como nuevo comienzo (o fin) de la palabra.

Al vector sonoro se le debe extraer la componente continua (restarle la media estadística). Cuando se extrae la componente continua, se le calcula el módulo y los cruces por cero. Dado que un cruce por cero ocurre cuando el signo de dos muestras sucesivas es distinto, podemos calcular rápidamente los cruces por cero tomando el signo del vector sonido, posición a posición y efectuando la siguiente operación:

$$Z_n = \sum_{m=-\infty}^{\infty} |sgn[x(m)] - sgn[x(m-1)]| \cdot w(n-m) \quad (2)$$

Este resultado es guardado en el vector cruces. Para esta operación se asume que el signo en  $t=-1$  es 1. Cruces contiene de esta manera 1 donde se produjo un cambio de signo y 0 donde no lo hubo. Luego se efectúa un promediado con ventanas temporales unitarias adyacentes de 10 msec que no se solapan (frames). Este promediado equivale a hacer un filtrado con un filtro pasabajos. Es así como de cada ventana se toma un solo valor que representa todo ese intervalo promediado, disminuyendo notoriamente la cantidad de valores con los que se trabaja: estos resultados así obtenidos se guardan en un vector de dimensión mucho menor. Se tendrán así dos nuevos vectores de trabajo que pueden ser interpretados como dos funciones del tiempo que representan al archivo grabado: un vector que representa la función magnitud promedio, y un vector que representa la tasa promedio de cruces por cero.

En el algoritmo original de Rabiner y Sambur de 1975 se utiliza la función energía y no la función magnitud; la propuesta que realiza (Merlo, Caram y García, 1997) de utilizar la función magnitud se basa en que por ser la energía el cuadrado de la función magnitud marca más notoriamente la diferencia entre zonas adyacentes de alta y baja energía, pudiéndose producir un corte incorrecto de la palabra

Al utilizar la función magnitud (o módulo de la señal) el contorno de amplitud es más suave y con cortes menos abruptos: los "pozos" de la función magnitud son menos

profundos que los "pozos" de la función energía. Se asume a priori en el algoritmo que los 100 mseg. iniciales del vector de sonido no contienen habla, por lo que se computan la media y la desviación estándar de la función magnitud promedio y tasa de cruces por cero durante este intervalo de tiempo, para dar una caracterización estadística al ruido de fondo.

A continuación se procede a analizar la forma de la función magnitud promedio para hallar un intervalo en el cual siempre se exceda un valor de umbral muy conservativo ITU (Interval Threshold Upper ó umbral de intervalo superior). A tal fin se toma como ITU un determinado porcentaje del máximo valor que toma la función magnitud promedio. Puede ocurrir que el sonido hablado fuera tan poco energético que quedara enmascarado por el ruido. Para evitar este inconveniente, y considerando que los valores de magnitud del ruido se hallan en un 99% en la banda comprendida entre su media +/- 3 desviaciones estándar, tomamos como nuevo ITU al máximo entre el porcentaje antes especificado y la media del ruido mas tres desviaciones estándar. En este sentido se procede a comenzar con un porcentaje bajo (es decir, tratando de estar lo más cerca posible de los límites de la palabra) e ir aumentándolo en 10% sucesivamente si no se cumple la condición de que dentro del intervalo siempre se supere el ITU. De esta manera se llega a establecer el ITU, y con él, los valores iniciales de los punteros de principio y fin de la palabra dentro del vector magnitud promedio. Luego comenzando a trabajar hacia atrás en el vector desde el punto donde fue excedido ITU por primera vez, se halla punto donde la función magnitud promedio cae por primera vez por debajo de un cierto valor de umbral inferior ITL (Interval Threshold Lower ó umbral de intervalo inferior, cuyo valor adoptado en el presente trabajo es igual a la media estadística del ruido mas dos desviaciones estándar). Dicho punto es seleccionado tentativamente como el origen del habla en el vector magnitud promedio. Un procedimiento totalmente similar es seguido para determinar el punto final (obviamente con los mismos valores de ITU e ITL). Este procedimiento de doble umbral asegura un suave acercamiento a los extremos inicial y final de la parte hablada del vector sonoro.

Una vez parados sobre los índices hallados por el ITL (principio y fin), es razonable pensar que los verdaderos límites de la palabra se hallan fuera de este intervalo. A partir de aquí nos moveremos hacia adelante y hacia atrás con los punteros PRINCIPIO y FIN comparando la tasa de cruces por cero con un límite o umbral IZCT (Interval Zero

Crossing Threshold ó umbral del intervalo de cruces por cero) determinado a partir de las estadísticas de la tasa de cruces por cero del ruido de fondo. Este análisis se limita hasta 25 marcos (frames) o posiciones anteriores a principio (siguientes a fin). Si la tasa de cruces por cero supera el umbral 3 o mas veces, el punto inicial principio es movido hasta la primera posición en que fue superado el límite IZCT; en caso contrario el punto inicial (principio) queda definido como lo estaba al comienzo. Un procedimiento similar se sigue para determinar el punto final.

Una vez establecidos los puntos extremos de la parte hablada, simplemente se realiza una transferencia de esa parte intermedia del vector de entrada a otro vector de salida, que en muchos casos llega a ser notablemente menor en tamaño (40 % en algunos casos), lo cual ahorra memoria (factor limitante muy importante al momento de procesar señales) y disminuye mucho el tiempo de computo gracias a la disminución o descarte de información innecesaria y ruidosa (Merlo, Caram y García, 1997).

A partir de aquí queda eliminado el ruido de fondo inicial y final del archivo sonoro (en el caso de ser un archivo), y se procede al mapeo del intervalo hablado en un patrón representativo. Evidentemente, para un correcto funcionamiento del intérprete, el patrón o vector obtenido debe ser representativo de la frase dicha (palabra): es decir que para una misma palabra varias veces pronunciada por distintas personas (con duraciones, frecuencia fundamental -"pitch"- y energías instantáneas distintas) los vectores obtenidos deben ser similares (Merlo, Caram y García, 1997). Este proceso implica también independizarse del tiempo de duración de la palabra. Para este proceso se opta por aplicar la Transformada de Fourier que se describe a continuación.

#### **4.4 Transformada de Fourier**

La señal producida por la voz humana y obtenida a través de un micrófono se encuentra en el dominio del tiempo. Para su procesamiento en el dominio de la frecuencia se usan algoritmos que utilizan la Transformada de Fourier.

La Transformada de Fourier es una herramienta de análisis muy utilizada en el campo científico (acústica, ingeniería biomédica, electromagnetismo, comunicaciones, etc.). Esta transforma una señal representada en el dominio del tiempo al dominio de la

frecuencia sin alterar su contenido de información, es decir, sólo es una forma diferente de representarla.

La Transformada de Fourier es una herramienta muy útil cuando se trabaja con modelos matemáticos, pero si queremos trabajar con señales físicas reales y operando mediante computador debemos trabajar con modelos finitos y discretos.

Una vez muestreada debemos convertirla en finita. Para ello limitamos el número de puntos que se toman. Matemáticamente es multiplicar la señal por una ventana temporal; el efecto que provocamos es convolucionar el espectro de la señal muestreada con el espectro de la ventana, y se produce una distorsión de la transformada de la señal original. Por ello conviene elegir un tipo de ventana que produzca una menor distorsión.

Aunque en el modelo se haya aplicado una ventana, se sigue teniendo infinitas muestras que valen cero, obteniendo un espectro continuo; como última decisión se debe limitar el número de ceros que se toman, lo que provocará un muestreo del espectro continuo.

La Transformada de Fourier discreta se define en la ecuación 3:

$$G\left(\frac{h}{NT}\right) = \sum_{n=0}^{N-1} g(nT) e^{\frac{-j2\pi hn}{N}}, h = 0, 1, \dots, N-1 \quad (3)$$

$$g(nT) = \frac{1}{N} \sum_{n=0}^{N-1} G\left(\frac{h}{NT}\right) e^{\frac{j2\pi hn}{N}}, h = 0, 1, \dots, N-1$$

Donde  $T$  es el período de muestreo de la señal original,  $N$  es el número de puntos que se toman (incluyendo los ceros),  $h$  la variable índice de frecuencias y  $n$  la variable índice de la muestras.

Las propiedades de la Transformada de Fourier discreta se pueden enunciar en los siguientes puntos:

Linealidad:  $x(n) + y(n) \Leftrightarrow X(h) + Y(h)$ .

Simetría:  $\frac{1}{N}X(n) \Leftrightarrow x(-h)$ .

Desplazamiento temporal:  $x(n-i) \Leftrightarrow X(h)e^{\frac{-j2\pi hi}{N}}$ .

Desplazamiento en frecuencias:  $x(n)e^{\frac{-j2\pi hi}{N}} \Leftrightarrow X(h-i)$ .

Funciones pares:  $X(h) = R(h)$ .

Funciones impares:  $X(h) = jI(h)$ .

**Figura 8.** Propiedades de la Transformada de Fourier discreta.

(Bernal, Bobadilla y Gómez, 2000)

Según la propiedad del desplazamiento temporal, se observa que cuando se produce éste cambia la fase en el dominio de la frecuencia, pero el módulo permanece constante. Por ello, se calcula exclusivamente el módulo, para que los cálculos sean independientes de la posición de la ventana respecto a la señal a analizar.

Al ser la voz una señal no estacionaria, debemos tomar ventanas relativamente pequeñas para que dentro de cada ventana se pueda considerar “cuasiestacionaria”. Por otra parte, si tomamos ventanas pequeñas la resolución en frecuencias resulta pobre; ya que en la transformada discreta se calcula la amplitud en las frecuencias  $h/NT$ , siendo el rango  $h = 0, 1, \dots, N-1$ , por lo tanto, se estudian tantas frecuencias como indique el valor de  $N$ .

Suponiendo que la onda  $g(t)$  tiene una frecuencia máxima de  $f_{\max}$ , y se muestrea al doble de su frecuencia máxima,  $T = 1 / 2f_{\max}$ . El espectro de la onda con el simétrico de su copia; si calculamos las frecuencias desde  $h = 0, 1, \dots, N-1$ , se obtiene el espectro que buscamos y el simétrico. Por ello, debe calcularse hasta  $h = (N / 2) - 1$  si  $N$  es par, y hasta  $h = \text{parte entera} (N / 2)$  si  $N$  es impar. El cálculo de la siguiente mitad sería redundante.

La frecuencia máxima de estudio sería  $((N/2) - 1) / NT$  (para  $N$  par). El efecto de  $NT$  aumentar de tamaño la ventana hace que el número de frecuencias que se estudian sea mayor, a costa de disminuir la distancia entre ellas y obteniendo una mejor precisión, pero no cambia el límite de la frecuencia máxima: para un tamaño de ventana infinita sería de  $1 / 2T$ .

#### 4.4.1 Tipos de ventana

Hasta ahora se ha hablado de ventanas rectangulares. Se definen como:

$$h(t) = \begin{cases} 1 & \rightarrow |t| \leq \frac{T_0}{2} \\ 0 & \rightarrow |t| > \frac{T_0}{2} \end{cases}, \text{ siendo } T_0 \text{ el ancho de la ventana.} \quad (4)$$

Se recuerda que el hecho de utilizar una ventana hace que se convolucione la transformada de la señal con la transformada de la ventana. Por lo tanto, se debe elegir aquella ventana que produzca menor distorsión.

Existen otros tipos de ventana cuyas transformadas pueden producir menos distorsión, entre ellas las de Hamming, Hanning y Parzen, que se definen como:

Hamming:

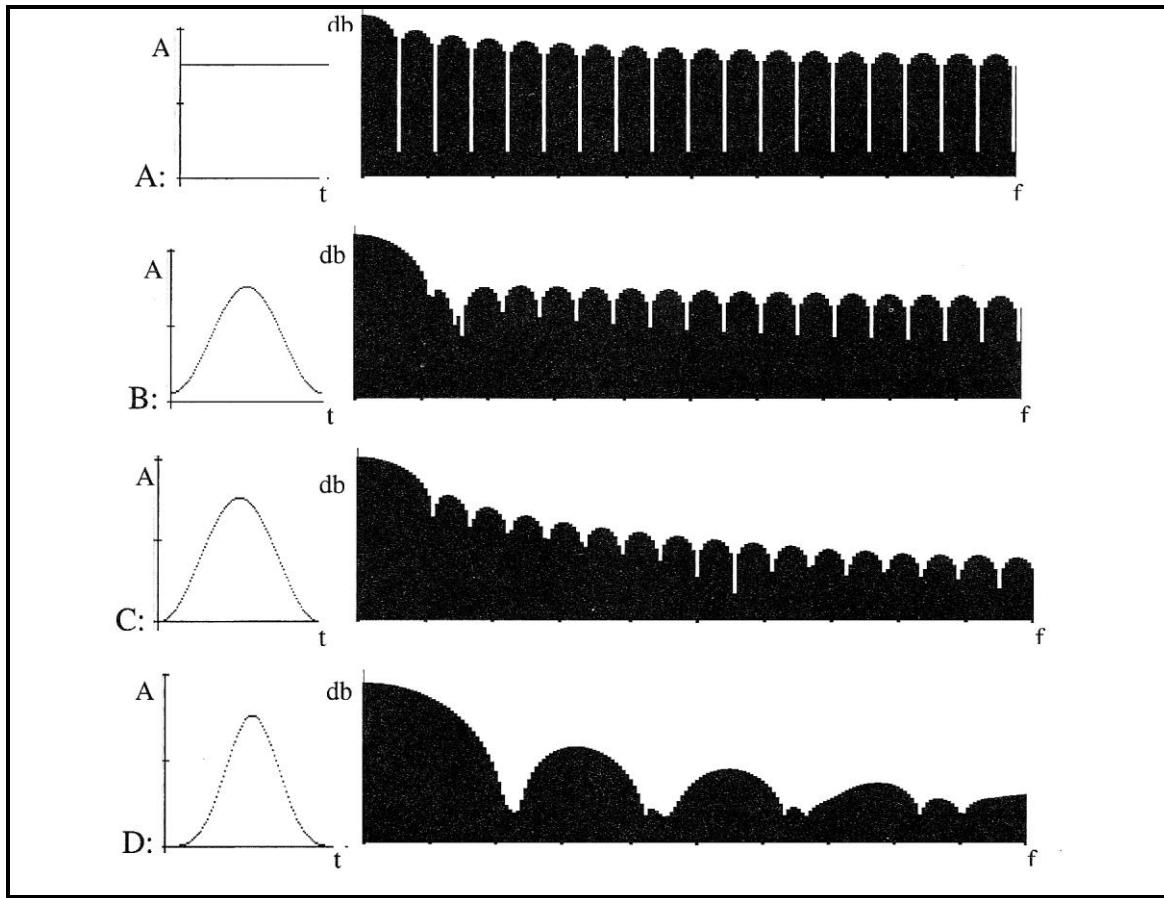
$$h(t) = \begin{cases} 0'54 + 0'46 \cos\left(\frac{2\pi t}{T_0}\right) & \rightarrow |t| \leq \frac{T_0}{2} \\ 0 & \rightarrow |t| > \frac{T_0}{2} \end{cases} \quad (5)$$

Hanning:

$$h(t) = \begin{cases} \frac{1}{2} \left[ 1 + \cos\left(\frac{2\pi t}{T_0}\right) \right] & \rightarrow |t| \leq \frac{T_0}{2} \\ 0 & \rightarrow |t| > \frac{T_0}{2} \end{cases} \quad (6)$$

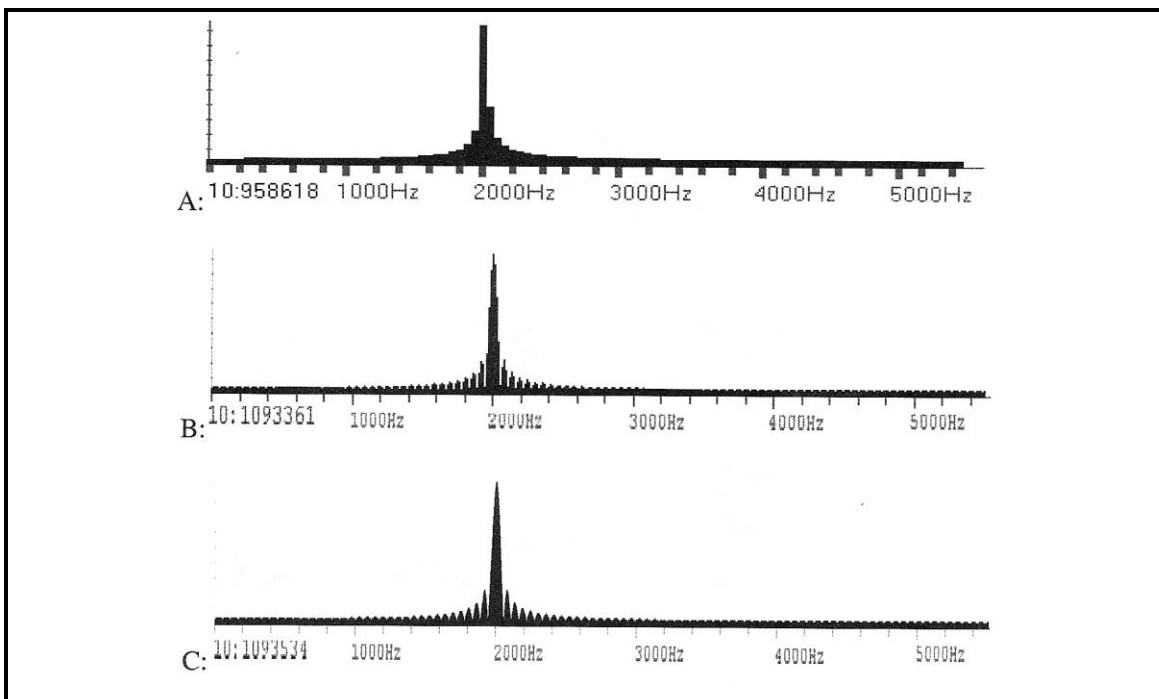
Parzen:

$$h(t) = \begin{cases} 1 - 24\left(\frac{t}{T_0}\right)^2 + 48\left|\frac{t}{T_0}\right|^3 & \rightarrow |t| < \frac{T_0}{4} \\ 2\left[1 - \frac{2|t|}{T_0}\right]^3 & \rightarrow \frac{T_0}{4} < |t| < \frac{T_0}{2} \\ 0 & \rightarrow |t| \geq \frac{T_0}{2} \end{cases} \quad (7)$$



**Figura 9.** Respuesta en frecuencia de los diferentes tipos de ventana.  
**A.** Ventana rectangular. **B.** Ventana Hamming. **C.** Ventana Hanning. **D.** Ventana Parzen.  
 (Bernal, Bobadilla y Gómez, 2000)

Otro aspecto, interesante a tener en cuenta es el número de ceros que se toman para el cálculo de la Transformada de Fourier. Ya se comentó que, por la no estacionalidad de la voz, se debía tomar un número reducido de muestras que se pudieran considerar cuasiestacionarias; se llama a este número  $M$ . Por cuestiones de cálculo, se trabaja con la expresión  $g(nT)h(nT)$ , siendo  $h(nT)$  la ventana utilizada. Ahora falta por definir la variable  $N$ , que se el número de puntos con que se aplica la Transformada de Fourier discreta. Como mínimo será  $M$ , que es el número de muestras de la ventana, pero normalmente se toma  $N > M$ , añadiendo ceros a las muestras, para dar una mejor visión del espectro continuo, con una distancia entre muestras de  $1/NT$ . En la figura 10 se tiene un ejemplo del efecto de añadir ceros al cálculo de la Transformada de Fourier.



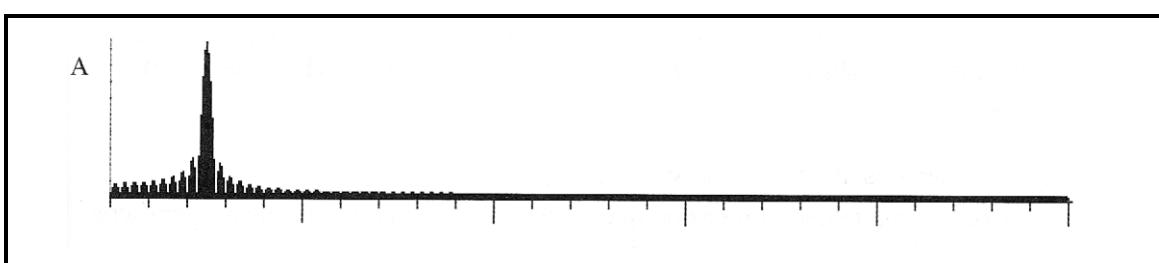
**Figura 10.** Corresponde a una función sinusoidal de 200 Hz de frecuencia, muestreada a 11025 Hz y con un ancho de ventana rectangular de 200 muestras. Todas las figuras se normalizan en altura, el número que aparece a la izquierda es el factor de escala aplicado para la normalización.

**A.**  $N = 200$ . **B.**  $N = 800$ . **C.**  $N = 3200$ .

(Bernal, Bobadilla y Gómez, 2000)

Un caso particular es cuando se toma un tamaño de ventana que sea un múltiplo del período de la señal original. En tal caso obtenemos un espectro ideal dado que se muestra el espectro continuo en los puntos donde la función resultante de la ventana vale cero, con lo que la distorsión queda oculta.

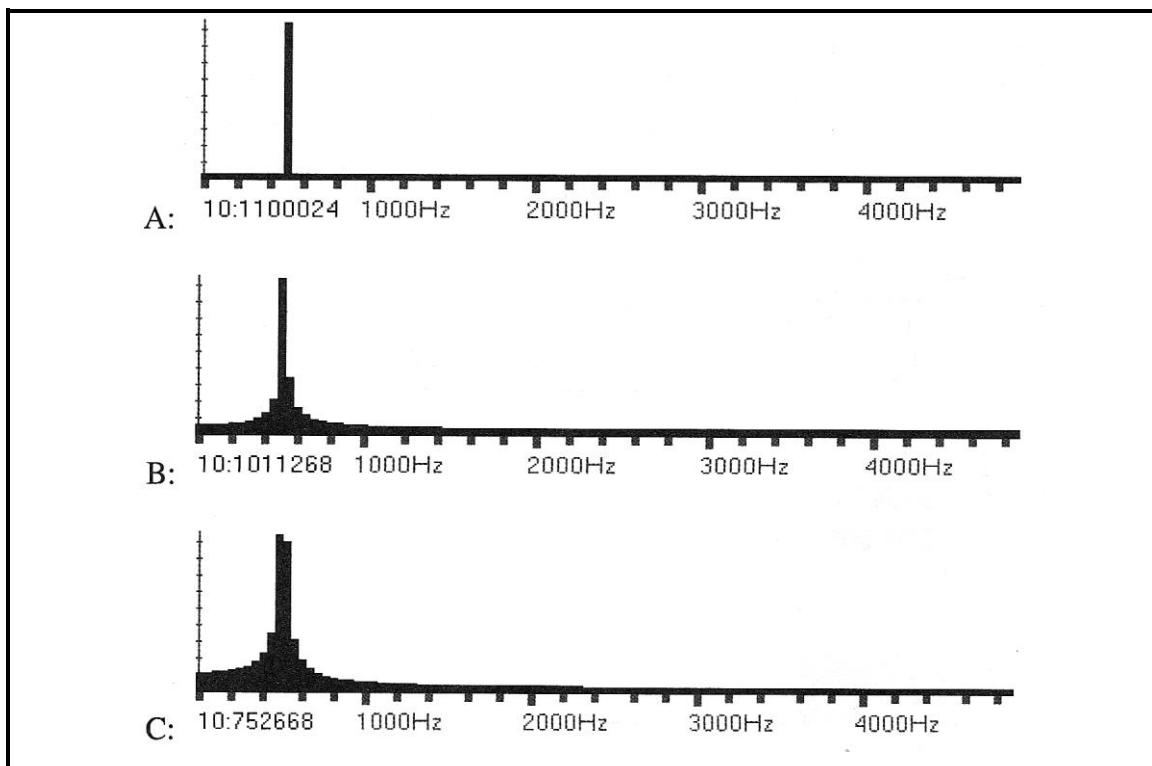
En la figura 11 se toma un número de ceros suficiente para apreciar el espectro completo.



**Figura 11.** Espectro de una señal sinuosidad de 500 Hz de frecuencia y muestreada a 10000 Hz. Representa un espectro bien definido de la señal original. Un período completo contiene 20 muestras.

(Bernal, Bobadilla y Gómez, 2000)

En la figura 12 se han tomado tres casos; en la gráfica superior coincide el número de muestras con el período de la señal original; en la intermedia se desfase en un cuarto de período, y en la inferior en medio período.



**Figura 12.** Espectro de una señal sinuosidad de 500 Hz muestreada a 10000 Hz.

- A. se ha tomado una ventana múltiplo del período de la señal original (200 muestras).
- B. se ha tomado un cuarto de período más (205 muestras).
- C. Medio período de más (210 muestras), este ultimo sería el peor de los casos.

(Bernal, Bobadilla y Gómez, 2000)

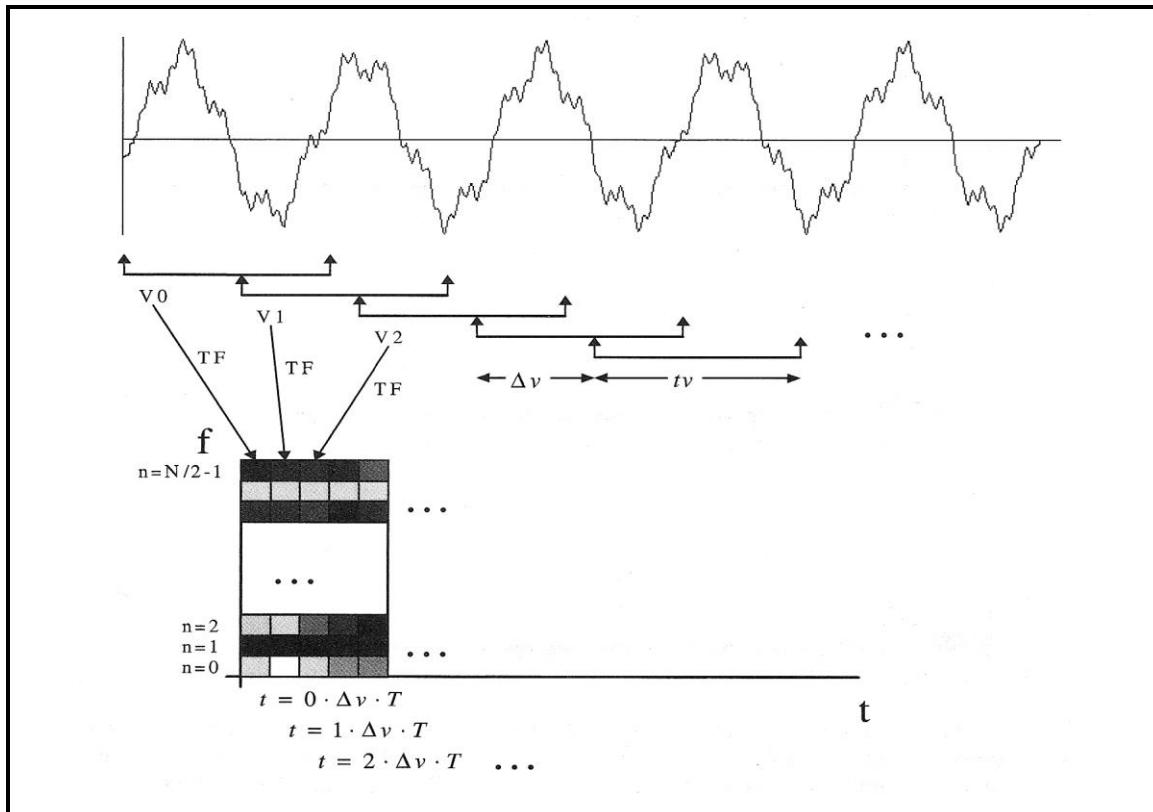
#### 4.4.1.1. Espectros variantes en el tiempo

Hasta ahora se ha visto el espectro de una sola ventana. Pero para observar la evolución de la señal de voz se debe visualizar una secuencia de espectros de cada ventana.

Existen dos alternativas básicas:

1. Utilizar una tercera dimensión que represente el tiempo, presentando gráficos en tres dimensiones (la tercera dimensión sería la secuencia de ventanas).
2. Utilizar el color o una escala de grises como tercera dimensión y representando la amplitud o energía.

Se considera que la segunda solución es más fácil de interpretar y será la que se toma. En la figura 13 se tiene un ejemplo general de cómo se va formando el espectro variante en el tiempo.



**Figura 13.** Esquema de cómo se forma el espectro variante en el tiempo.  
(Bernal, Bobadilla y Gómez, 2000)

Al aplicar la Transformada de Fourier, da un conjunto de resultados correspondientes a las amplitudes de las distintas frecuencias, es el módulo de esta amplitud el que se codifica en colores.

Como el espacio de las amplitudes es muy diferente, normalmente se aplica una escala logarítmica para la codificación; además se establece un nivel mínimo de señal para eliminar el ruido que aparece. Tanto si se elige una paleta de colores o de grises, no existe ninguna predeterminada, queda a gusto del programador.

También se habla de ancho de banda al aplicar la Transformada de Fourier. Se entiende por ello la frecuencia de muestreo dividida por el tamaño de la ventana de muestras. Sí,

cuando hablamos de la Transformada de Fourier en banda ancha (300 Hz) se refiere a que si muestreamos a 11.025 Hz se toma un tamaño de ventana de 37 puntos.

#### 4.5. Coeficientes Cepstrales de Mel (MFCC)

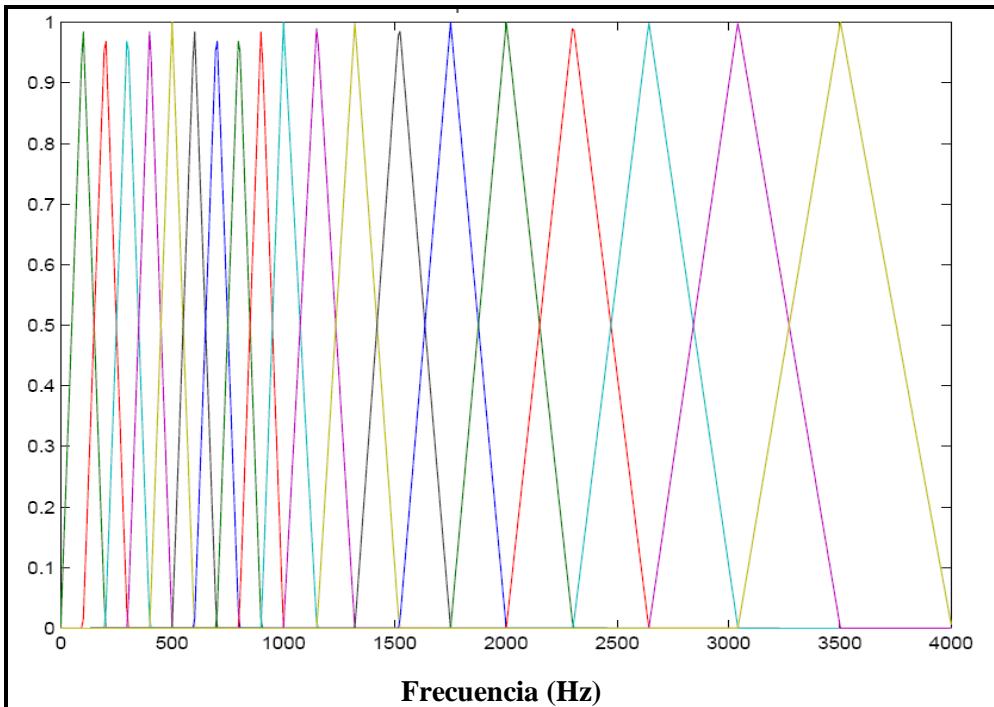
Los coeficientes cepstrales de Mel o MFCC (Mel-Frequency Cepral Coeficients) por su sigla en inglés, es un método usado y comprobado. Los MFCC se basan en la manera en la que funciona el oído humano, con filtros lineales en bajas frecuencias y filtros logarítmicos en altas frecuencias. Estas características están expresadas en la escala frecuencial de Mel, la frecuencia lineal trabaja por debajo de los 1000 Hz y la logarítmica por encima de los 1000 Hz. Para extraer los coeficientes se divide en dos secciones, la escala de Mel y la transformada discreta del coseno.

##### 4.5.1 Escala de Mel

Como se menciona arriba, los estudios sobre la percepción de sonidos que realiza el oído humano es de forma logarítmica y no en escala lineal. Así para cada frecuencia  $f$  hay una relación en escala de Mel. Como punto de referencia el “pitch” que equivale a 1 KHz, 40dB esta definido como 1000 mels. En la formula 8 se muestra una aproximación del cálculo.

$$mel(f) = 2595 \times \log_{10} \left( 1 + \frac{f}{700} \right) \quad (8)$$

Para simular la extracción de las frecuencias de Mel se aplica un banco de filtros a los datos arrojados por la transformada de Fourier. Este banco de filtros es uniformemente espaciado en la escala de Mel, tiene una forma de filtro paso banda triangular, y los espacios triangulares son equiespaciados en la escala de Mel. El número de filtros pasa banda de Mel es típicamente 20, pero este puede variar dependiendo de la frecuencia de muestreo.



**Figura 14:** Banco de Filtros de Mel.  
(Nilsson, 2001)

#### 4.5.2 Transformada Discreta del Coseno

Después de calcular el filtro triangular, y de haber calculado el logaritmo de sus sumas, el siguiente paso es convertir el logaritmo del espectro de Mel al dominio del tiempo. El resultado es llamado coeficientes cepstrales de Mel. La representación cepstral del habla provee una buena representación de la señal dada una ventana para ser analizada, porque los coeficientes espectrales de Mel son números reales, y pueden ser convertidos al dominio temporal usando la Transformada discreta del Coseno (fórmula 9).

$$mfcc(i) = \frac{1}{Nfilters} \sum_{l=1}^{Nfilters} mfb(l) \cos\left(i\left(l - \frac{1}{2}\right) \times \frac{\pi}{Nfilters}\right) \quad (9)$$

$$i = 1, \dots, Nfilters$$

La función “ $mfb(l)$ ” se refiere al cálculo del banco de filtros de Mel; Se descarta el primer componente de la Transformada Discreta del Coseno, ya que el primer componente solo representa y refleja el logaritmo de la energía de la ventana. (Nilsson, 2001).

#### **4.6. Reconocimiento de Patrones**

La fase final en el proceso de reconocimiento está determinada por la utilización de algún tipo de clasificador de patrones. Puesto que en las etapas previas se obtenían una serie de parámetros que intentaban representar a cada una de las unidades individuales de reconocimiento (palabras), ahora es necesario aplicar un método de clasificación que determine a qué patrón se ajusta mejor cada grupo de parámetros que representa una unidad.

En el área de reconocimiento de patrones, se ha desarrollado una variedad de clasificadores que tratan de hacer uso de diversos métodos y técnicas, para separar (clasificar) las clases de objetos determinados por los parámetros de entrada al clasificador.

Los clasificadores que más auge han experimentado en los últimos años, son las redes neuronales, aunque también se ha utilizado las cadenas de Markov, árboles de decisión, funciones discriminantes lineales, clasificadores estadísticos, etc. (Bernal, Bobadilla y Gómez, 2000).

Una gran parte de los clasificadores usados para el diseño de reconocedores de voz se basan en la utilización de Redes Neuronales (RN). Las RN proporcionan un método de aprendizaje automático paramétrico. De esta manera, se puede conseguir hacer separación de clases a partir de un conjunto finito de muestras (patrones) proporcionados al sistema.

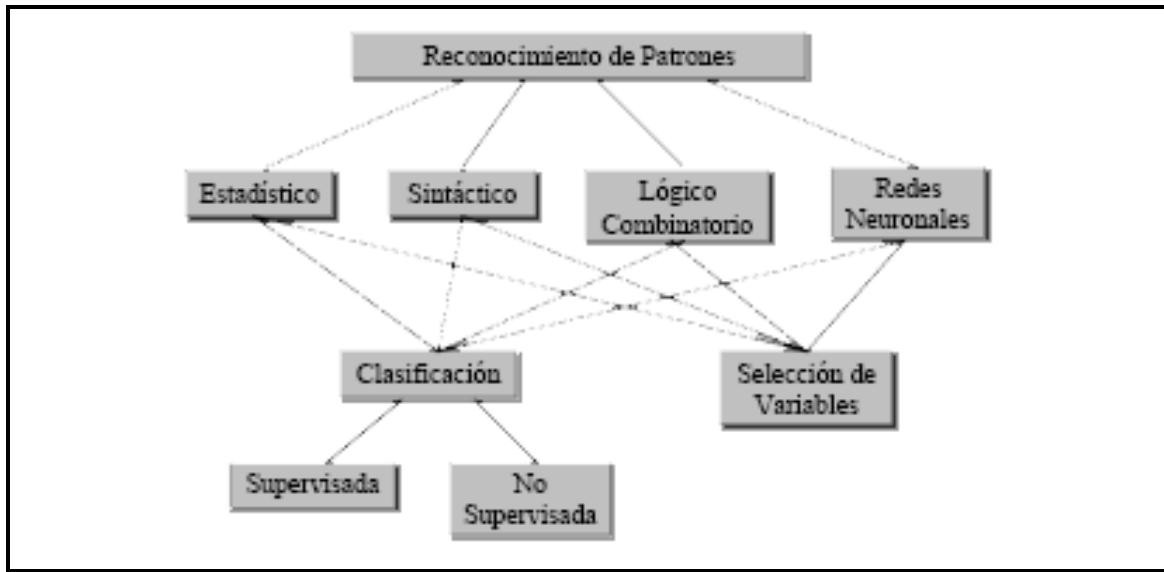
El tipo de RN que ha sido utilizado con mayor asiduidad en la creación de sistemas de reconocimiento, es el “Multicapa con aprendizaje supervisado, basado en el algoritmo de propagación hacia atrás (Backpropagation)”. (Bernal, Bobadilla y Gómez, 2000).

#### **4.6.1 Enfoques de Reconocimiento de Patrones**

- a) Estadístico:** Este enfoque se basa en la teoría de probabilidad y estadística y supone que se tiene un conjunto de medidas numéricas con distribuciones de probabilidad conocidas y a partir de ellas se hace el reconocimiento.
- b) Sintáctico:** Este enfoque se basa en encontrar las relaciones estructurales que guardan los objetos de estudio, utilizando la teoría de lenguajes formales. El objetivo es construir una gramática que describa la estructura del universo de objetos.
- c) Redes Neuronales:** Este enfoque supone que tiene una estructura de neuronas interconectadas que se estimulan unas a otras, las cuales pueden ser “entrenadas” para dar una cierta respuesta cuando se le presentan determinados valores.
- d) Lógico Combinatorio:** Este enfoque se basa en la idea de que la modelación del problema debe ser lo más cercana posible a la realidad del mismo, sin hacer suposiciones que no estén fundamentadas. Uno de sus aspectos esenciales de las características utilizadas para describir a los objetos de estudio debe ser tratado cuidadosamente.

#### **4.6.2 Problemas de Reconocimiento de Patrones**

- a) Selección de variables:** Consiste en determinar cual conjunto de características es el más adecuado para describir los objetos.
- b) Clasificación Supervisada:** Consiste en clasificar nuevos objetos basándose en la información de una muestra ya clasificada.
- c) Clasificación no supervisada:** Consiste en dada una muestra no clasificada encontrar la clasificación de la misma.



**Figura 15:** Enfoques y problemas de reconocimiento de patrones.  
(Carrasco, 2006)

#### 4.6.3 Redes Neuronales

Para comenzar, se presentan tres definiciones de lo que es una RN (Bernal, Bobadilla y Gómez, 2000). Estas definiciones, como se podrá observar, parten de tres visiones muy diferentes (pero compatibles) de la misma realidad.

- "Una nueva forma de computación inspirada en modelos biológicos".
- "Un modelo matemático compuesto por un gran número de elementos procesales organizados en niveles".
- "Son redes interconectadas masivamente en paralelo, de elementos simples, los cuales intentan interactuar con los objetos del mundo real del mismo modo que lo hace el sistema nervioso biológico".

##### 4.6.3.1 Características de una neurona artificial simplificada

Las neuronas artificiales más comúnmente usadas en RN presentan una estructura y siguen unas pautas de funcionamiento muy similares a sus correspondientes al modelo natural.

En adelante, se utiliza la siguiente nomenclatura básica para describir y analizar las neuronas artificiales:

- $U_i$  para representar la neurona  $i$ -ésima.
- $Y_i$  para representar el resultado que genera la neurona  $Y_i$  (equivalente a la salida que se produce en el axón de la neurona biológica).
- $W_{ji}$  para representar el valor de inhibición/excitación entre las neuronas  $U_i$  y  $U_j$  (equivalente al efecto de los neurotransmisores sobre la sinapsis que une  $U_i$  con  $U_j$ ):

$W_{ji} > 0 \rightarrow$  sinapsis excitadora

$W_{ji} = 0 \rightarrow$  no existe conexión

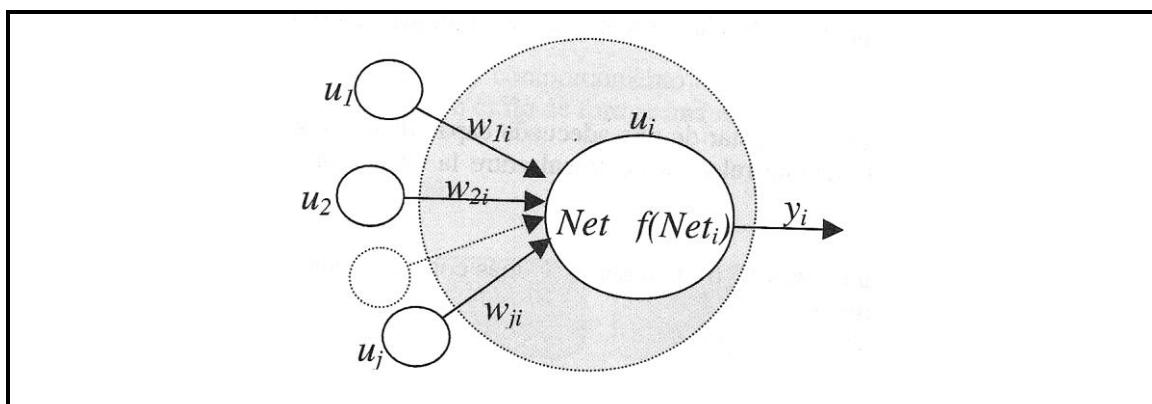
$W_{ji} < 0 \rightarrow$  sinapsis inhibidora

- $Net_i$  para representar el valor conjunto de todas las señales que le llegan a la

$$Net_i = \sum_j y_j * w_{ji} .$$

célula  $U_i$ . De esta manera, podemos establecer que

- $f(Net_i)$  para representar la función de salida o transferencia (equivalente al proceso que realiza la célula biológica, en función de la acumulación de los voltajes que le llegan:  $Net_i$ ).



**Figura 16.** Modelo de neurona artificial.  
(Bernal, Bobadilla y Gómez, 2000)

#### 4.6.3.2. Función de salida (función de transferencia)

En principio, se puede establecer por simplicidad, que la salida de la neurona  $U_i$  sea igual a la entrada que se le presenta ( $Net_i$ ). Este enfoque presenta el grave inconveniente de que se está realizando una transformación lineal. En consecuencia, el proceso entre unas neuronas y otras se limita a unas multiplicaciones/adiciones anidadas, empleadas para calcular los distintos  $Net_i$  involucrados.

De esta manera, si analizamos los cómputos involucrados en tres neuronas U, U' y U'', obtenemos:

$$\sum_l \left[ \sum_k \left( \sum_j y_j * w_{ji} \right) * w'_{ki} \right] * w''_{li} = Net_i \quad (10)$$

l, k y j se han establecido como índices diferentes, para resaltar el hecho de que cada neurona puede tener conectadas un número arbitrario de neuronas a su entrada. Este número en general no coincidirá con el de otras neuronas situadas en posiciones (capas) diferentes. Esta operación puede ser simplificada como:

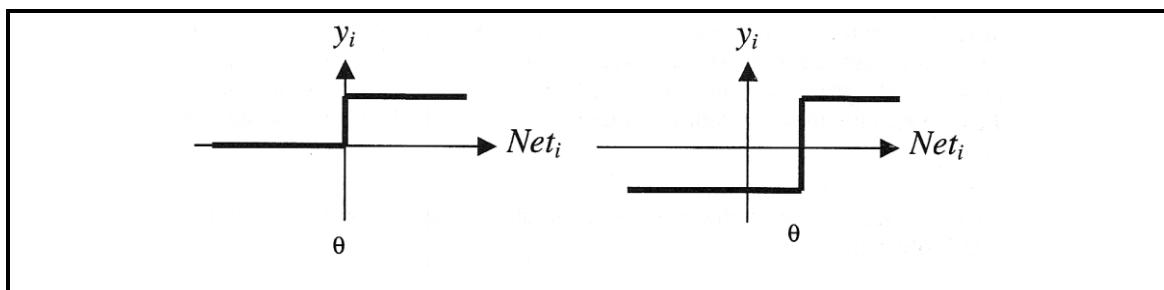
$$\sum_l y_l * (w_{li} * w'_{ki} * w''_{ji}) = \sum_l y_l * w_{li} \quad (11)$$

En definitiva, para dotar de una adecuada capacidad de cómputo a una RN, es necesario establecer una relación no lineal entre la entrada a una neurona y su salida.

Entre las funciones de transferencia (f) más corrientes que se utilizan en los sistemas de RN, tenemos:

#### 4.6.3.2.1. Función escalón

Las funciones anteriores son ejemplos del tipo "Función escalón", que se caracteriza por ofrecer únicamente dos valores de salida. Estos valores se determinan según el factor umbral  $\Theta$ . Cuando  $Net_i > \Theta$ ,  $Y_i$  toma un valor. Cuando  $Net_i < \Theta$ ,  $Y_i$  toma el otro valor establecido.



**Figura 17.** Función escalón.  
(Bernal, Bobadilla y Gómez, 2000)

#### 4.6.3.2.2. Función sigmoidal

Para permitir que la función de salida sea derivable en cualquier intervalo (característica importante en los métodos de aprendizaje), se utiliza la función sigmoidal cuya fórmula es:

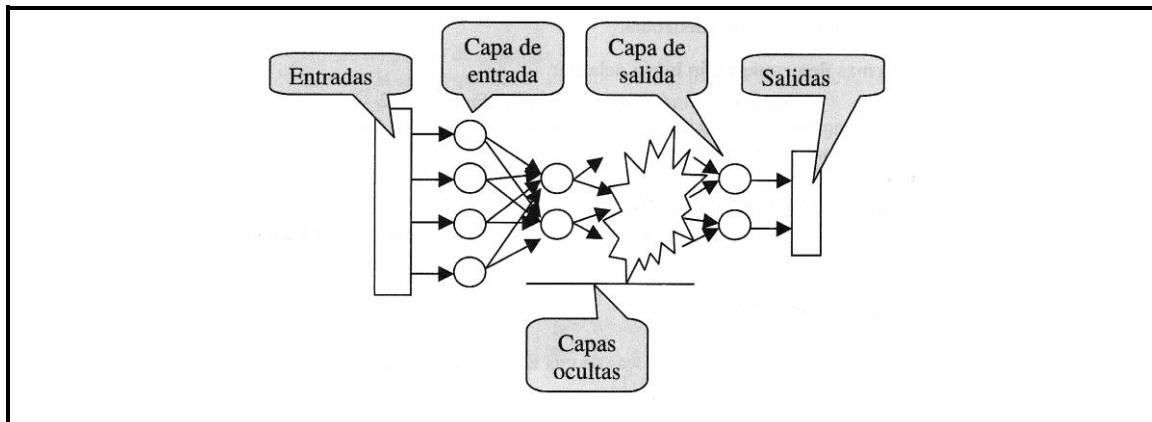
$$f(x) = \frac{1}{1 + e^{-ax}} \quad (12)$$

Esta función sigmoidal es parecida a la función escalón, pero evita la discontinuidad en el punto en el que  $Net_i$  toma el valor del umbral.

#### 4.6.3.3. Estructura de una red neuronal artificial

Una vez establecido el comportamiento de una neurona artificial, se va a conectar neuronas entre sí con el fin de formar una red de computación.

##### Niveles o capas de neuronas



**Figura 18.** Red neuronal artificial.  
(Bernal, Bobadilla y Gómez, 2000)

**Entradas:** Valores que alimentan a la RN, por ejemplo píxeles en un sistema de reconocimiento de patrones visuales, valores proporcionados por una transformada de Fourier en el caso de reconocimiento de voz, etc.

**Salidas:** Clases reconocidas con la RN, por ejemplo un carácter numérico en un sistema de reconocimiento óptico de códigos postales, un tipo de sonido del español en una aplicación de reconocimiento de voz, etc.

**Capa de entrada:** Conjunto de neuronas que recogen directamente los valores de entrada a la RN.

**Capa de salida:** Conjunto de neuronas que proporcionan los valores de salida de la RN.

**Capas ocultas:** Capas de la RN que procesan la información de tal manera que permiten una adecuada separabilidad de las clases que se pretenden reconocer.

#### 4.6.3.4. Formas de conexión entre neuronas

##### Sistemas de propagación hacia delante

Siguen esquemas como el de la figura 18. En las RN con propagación hacia adelante, cada neurona se conecta únicamente con las neuronas de la capa siguiente.

##### Sistemas de propagación hacia atrás

Estos sistemas recurrentes permiten la conexión sin restricciones de cada neurona.

##### Características generales

Entre las características básicas de una RN se encuentran

- Número de niveles o capas
- Número de neuronas por cada capa
- Patrones de conexión (topología)

#### 4.6.3.5. Aprendizaje

Al igual que las conexiones entre neuronas y sus valores sinápticos constituyen la clave para la codificación de información en el cerebro, el conjunto de valores  $W_{ij}$  nos proporciona la información que alberga una RN artificial.

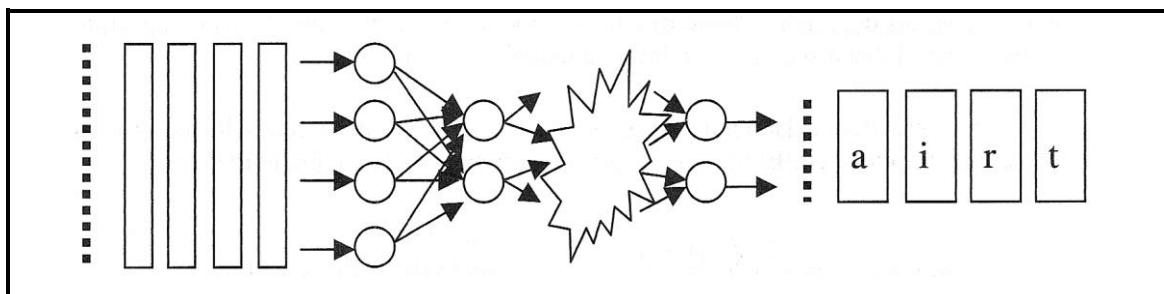
El proceso de aprendizaje consiste en variar los valores sinápticos  $W_{ij}$  siguiendo unas pautas establecidas. En los sistemas biológicos se produce una continua creación y destrucción de conexiones entre las neuronas, en los sistemas que los simulan, la

destrucción de una conexión se consigue haciendo que su peso asociado  $W_{ij}$  tome el valor cero.

La RN ha aprendido cuándo se han modificado los pesos de tal manera que por cada entrada que se le presenta nos proporciona el resultado esperado. Esta finalización en la modificación de pesos se puede expresar como:  $dW_{ij} / dt = 0$ , cada peso de la RN. En una primera aproximación, el aprendizaje se puede clasificar como:

#### 4.6.3.5.1. Supervisado

1. Se aplica un patrón de entrada.
2. Se obtiene la salida que la RN calcula sobre el patrón de entrada introducido.
3. Se compara la salida obtenida con la esperada.
4. Si existe error, significa que hay que variar los pesos de alguna manera para adaptar la RN al problema concreto de reconocimiento.
5. El proceso anterior se repite hasta que se considera aceptable la diferencia entre las salidas que se obtienen y las que se esperan.



**Figura 19.** Funcionamiento de una red neuronal.  
(Bernal, Bobadilla y Gómez, 2000)

#### 4.6.3.5.2. No supervisado

Emplea las características que las redes recurrentes ofrecen para estabilizar su aprendizaje sin necesidad de ir introduciendo los pares [patrón de entrada, salida esperada] que las redes supervisadas necesitan.

En el “aprendizaje supervisado por corrección de error”, se ajustan los pesos en función de la diferencia entre los valores deseados y los obtenidos en la salida de la red; es decir, en función del error producido en la salida.

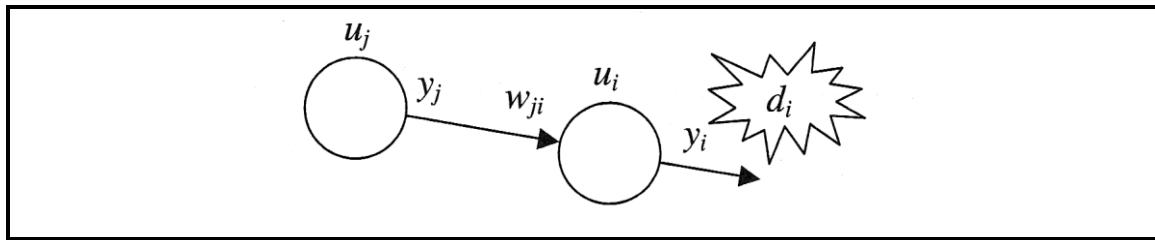
$$\Delta w_{ji} = \alpha * y_j * (d_i - y_i) \quad (\text{regla delta}) \quad (13)$$

$d_i$ : valor de salida deseado para la neurona  $u_i$

$\alpha$ : factor de aprendizaje (regula la velocidad de aprendizaje)

$\Delta w_{ji} + w_{ji}^{\text{actual}} - w_{ji}^{\text{anterior}}$ : modificación del peso  $w_{ji}$

$d_i - y_i$ : error que se produce en la neurona  $u_i$



**Figura 20.** Regla delta.  
(Bernal, Bobadilla y Gómez, 2000)

A este tipo de aprendizaje, se le denomina Hebbiano, dada la suposición de Hebb (1949): "Cuando un axón de la célula j toma parte en el disparo de la célula i de forma persistente, tiene lugar algún proceso de cambio metabólico en una de las células, o en las dos, de tal modo que la eficiencia de j, como una de las células que desencadena el disparo de i, se ve incrementada"

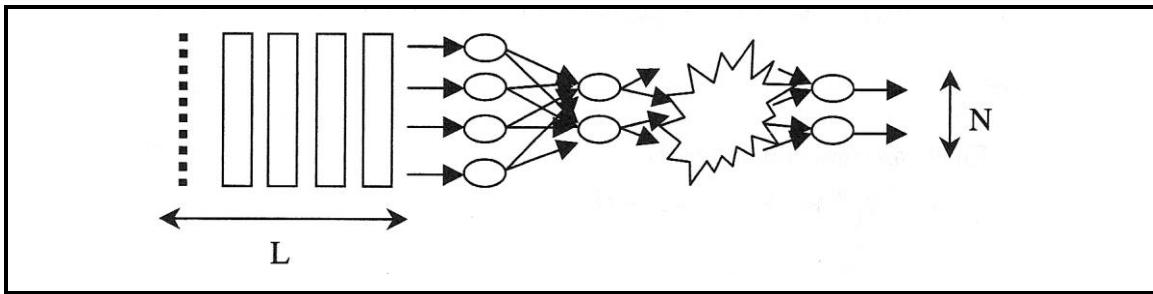
El aprendizaje Hebbiano sólo tiene en cuenta informaciones locales (a las dos células implicadas). El modelo se puede aumentar de la siguiente manera:

$$\text{Error global} = \frac{1}{2L} \sum_{k=1}^L \sum_{j=1}^N (y_j^k - d_j^k)^2 \quad (\text{regla de Widrow y Hoff}) \quad (14)$$

La fórmula anterior proporciona una medida del mínimo error cuadrático medio (LMS).

N: número de neuronas de salida

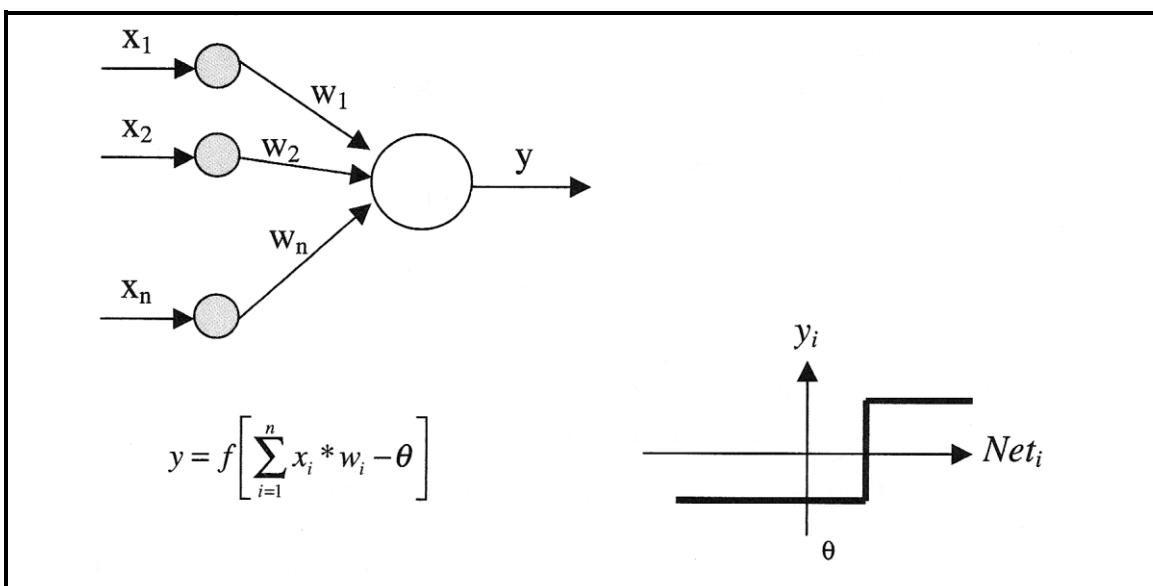
L: número de patrones



**Figura 21.** Regla delta.  
(Bernal, Bobadilla y Gómez, 2000)

#### 4.6.3.6. El Perceptrón

El caso más sencillo de RN es el que presenta una sola neurona de cómputo. A esta estructura se le denomina perceptrón y su estudio resulta obligado antes de profundizar en redes neuronales más complejas.



**Figura 22.** El perceptrón.  
(Bernal, Bobadilla y Gómez, 2000)

En este caso se ha omitido el segundo subíndice (correspondiente a la neurona destino), puesto que solamente tenemos una neurona de cómputo.

Las neuronas representadas con sombreado pertenecen a la capa de entrada. Estas neuronas son parte del formalismo de la red. Se encargan de recibir y distribuir los datos del exterior, sin realizar cálculos sobre los mismos.

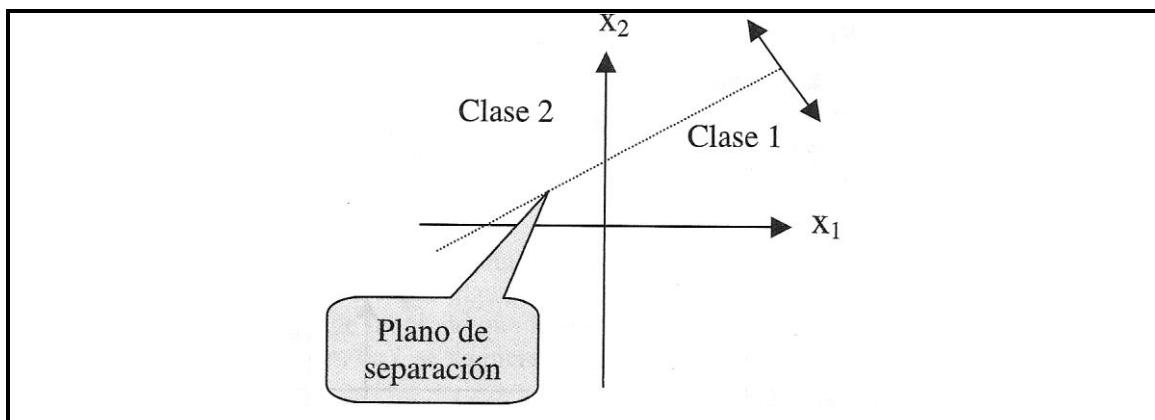
#### 4.6.3.6.1. Análisis:

En el caso más simple ( $N=2$ ):  $Y = f(X_1 * W_1 + X_2 * W_2 - \Theta)$  (15)

Esto significa que el resultado de la neurona va a tomar uno de los dos valores previstos en la función escalón (p.e. -1 y 1). El valor de salida dependerá de si  $(X_1 * W_1 + X_2 * W_2)$  es mayor o menor que el umbral.

Por lo tanto  $X_1 * W_1 + X_2 * W_2 - \Theta = 0$  proporciona la base para calcular la salida. Puesto que la fórmula anterior se corresponde con la de una recta en la que pretendemos asignar valores a los pesos, despejando:

$$x_2 = -\frac{w_1}{w_2} x_1 + \frac{\theta}{w_2} \quad (16)$$



**Figura 23.** Plano de separación en un perceptrón.  
(Bernal, Bobadilla y Gómez, 2000)

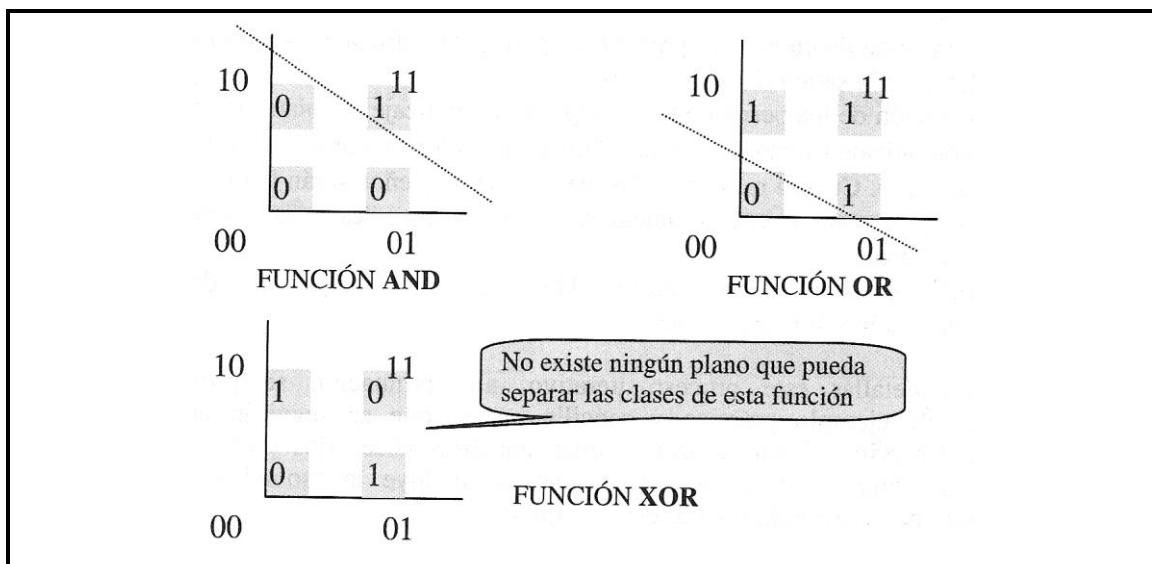
Como se puede observar, según varian los valores de los pesos y el umbral, obtendremos diversas inclinaciones ( $-W_j/W_2$ ) y desplazamientos ( $\Theta /W_2$ ) respecto al origen.

En el caso (habitual) de trabajar con múltiples dimensiones, se encontraría con un hiperplano de separación, que clasifica los patrones de entrada en un espacio multidimensional.

Puesto que un perceptrón solamente puede realizar separación lineal mediante su

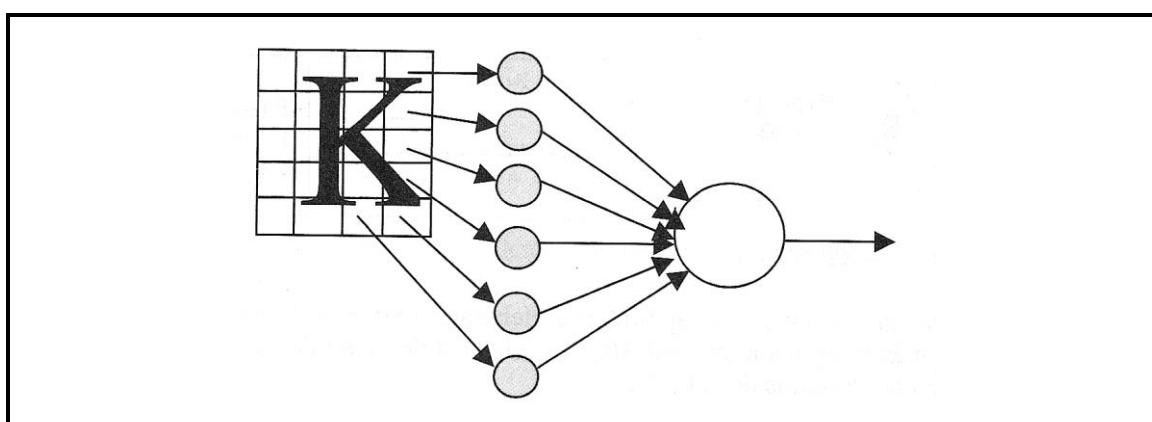
hiperplano asociado, este modelo sólo resuelve problemas de clasificación en los que las clases sean separables geométricamente.

Las funciones lógicas AND y OR son separables linealmente, por lo tanto pueden implementarse con un perceptrón. La función XOR no presenta esta característica.



**Figura 24.** Separabilidad lineal de las funciones AND, OR y XOR.  
(Bernal, Bobadilla y Gómez, 2000)

Se puede tratar de confiar labores complejas a un perceptrón, pero en general se necesitan redes neuronales con varias capas y varias neuronas por capa para resolver problemas reales. Hay que tener en cuenta que, habitualmente, no se presentan situaciones que admitan separabilidad lineal.



**Figura 25.** Perceptrón.  
(Bernal, Bobadilla y Gómez, 2000)

#### **4.6.3.6.2. Aprendizaje:**

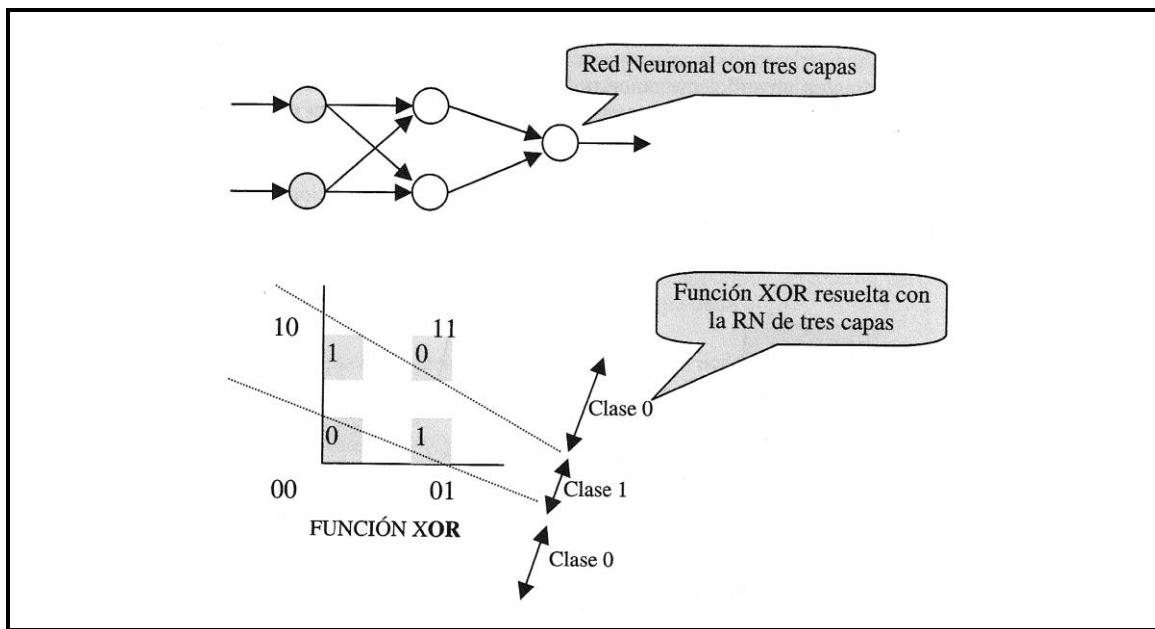
El perceptrón utiliza un aprendizaje supervisado hebbiano. Las etapas principales que conlleva este proceso son:

1. Inicialización de los pesos y del umbral. Habitualmente se asignan valores aleatorios que se encuentren en un rango equivalente al de las entradas y salidas.
2. Presentación de un par de aprendizaje [patrón de entrada, salida esperada].
3. Cálculo de la salida del perceptrón.
4. Adaptación de los pesos según la regla de aprendizaje hebbiano. Como factor de aprendizaje (o ganancia)  $\alpha$  se establece un valor mayor que cero y menor o igual a uno. Cuanto más pequeño sea  $\alpha$ , más pequeños serán los incrementos de los pesos, por lo que se llegará a la solución más despacio. ¡Pero también más seguro!
5. Repetición de los pasos anteriores hasta que todos los patrones de entrada produzcan la salida esperada (excepto el paso número 1).

#### **4.6.3.7. Redes multicapa generalizadas**

Puesto que el perceptrón solamente permite realizar separaciones lineales, es preciso diseñar redes neuronales que consigan clasificar elementos que no cumplan la condición de separabilidad lineal.

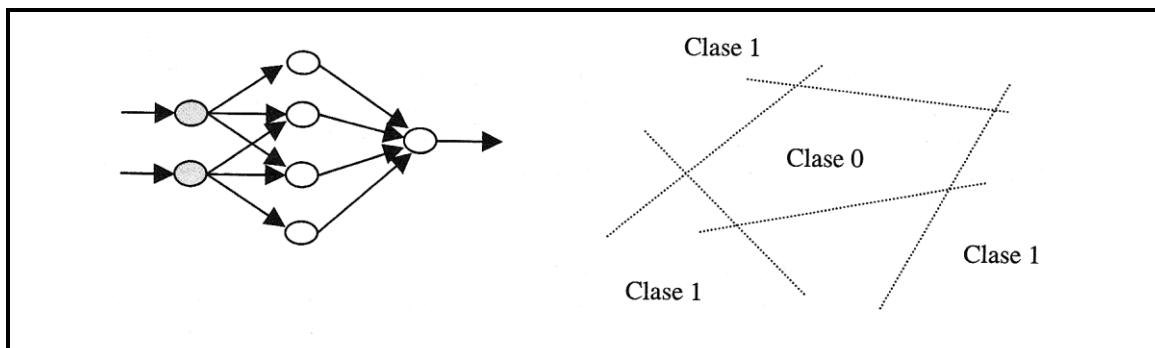
Como ya se mostró (Figura 24), un perceptrón es capaz de implementar una función OR o AND. De esta manera, añadiendo una capa al perceptrón tradicional, podemos combinar de forma lógica distintos campos de separación lineal. Este concepto se puede entender más adecuadamente en la figura 26:



**Figura 26.** Función XOR resuelta con una RN multicapa.  
(Bernal, Bobadilla y Gómez, 2000)

En este caso, cada neurona de la capa oculta soporta un hiperplano de separación lineal (representados con líneas discontinuas). La neurona de salida combina, con una función lógica, los espacios que ocupa cada clase que se pretende reconocer.

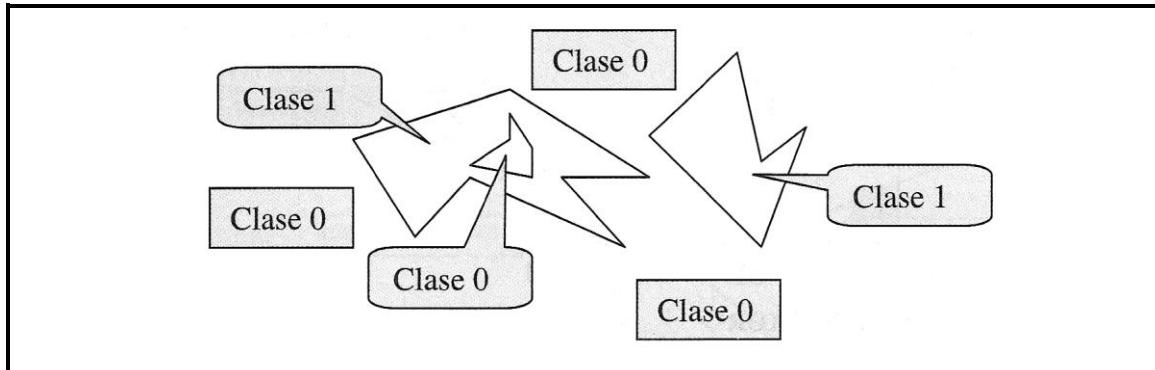
Con este mismo esquema de RN de tres capas se pueden conseguir separaciones de clases más complejas sin más que añadir nuevas neuronas en la capa oculta:



**Figura 27.** Tipo de funciones que se pueden resolver con una RN de tres capas.  
(Bernal, Bobadilla y Gómez, 2000)

En la mayor parte de los casos resulta necesario trabajar con redes de varias capas que presenten un suficiente número de neuronas en cada una de estas capas. Al aumentar el número de capas se consigue crear un mayor número de zonas de separación entre las clases.

Como ejemplo de las zonas de separación (en dos dimensiones) que se puede conseguir con una RN de cuatro capas, se presenta en la figura 28:



**Figura 28.** Tipo de funciones que se pueden resolver con una RN multicapa generalizada.  
(Bernal, Bobadilla y Gómez, 2000)

La utilización de redes multicapa es necesaria para poder soportar la mayor parte de las aplicaciones de reconocimiento que se utilizan actualmente. Sin embargo, la regla de aprendizaje Hebbiano, por sí sola, no es suficiente como para proporcionar un método de aprendizaje automático aplicable a este tipo de RN (Bernal, Bobadilla y Gómez, 2000)..

#### 4.6.3.7. Algoritmo Back Propagation

En 1986, Rumelhart, Hinton Y Williams desarrollaron un método para que una red neuronal "aprendiera" la asociación que existe entre los patrones de entrada a la misma y las clases correspondientes, utilizando más niveles de neuronas que los dos que empleó Rosenblatt (1961) para desarrollar el perceptrón. A este método se le denominó como "propagación del error hacia atrás" o retropropagación de gradiente (Back Propagation).

Como se mostró en el apartado anterior existen algunas funciones que el perceptrón no puede resolver. Para solucionar este problema se emplean las redes multicapa, donde el conjunto de neuronas de las capas ocultas permiten formar lo que se conoce como "representación interna del conocimiento".

El método de aprendizaje de propagación del error hacia atrás utiliza la siguiente técnica: dado un patrón de entrada, compara el resultado obtenido en las unidades de salida con la respuesta que se desea obtener. A continuación reajusta los pesos de la red de manera que, la siguiente vez que se presente el mismo patrón de entrada, la red produzca un resultado más cercano al deseado, es decir, que el error disminuya.

A este método también se le denomina "regla delta generalizada", debido a que es una generalización del procedimiento utilizado en el perceptrón para ajustar los pesos de la red, al que "Rumelhart" llamó "regla delta".

La regla delta generalizada se desarrolla dentro del contexto de redes multicapa con conexiones de tipo de propagación hacia delante y con unidades que poseen funciones de activación no lineales. Este tipo de funciones se definen como funciones no decrecientes, y derivada. La función sigmoidal pertenece a este tipo de funciones. El funcionamiento de la regla delta generalizada parte de la regla delta empleada en el perceptrón:

$$\Delta w_{ji} = \alpha * y_j * (d_i - y_i) \quad (\text{regla delta})$$

$d_i - y_i = \delta_i \rightarrow$  error que se produce en la neurona  $u_i$

Regla delta generalizada,  $\delta_i = (d_i - y_i) * f'(Net_i)$

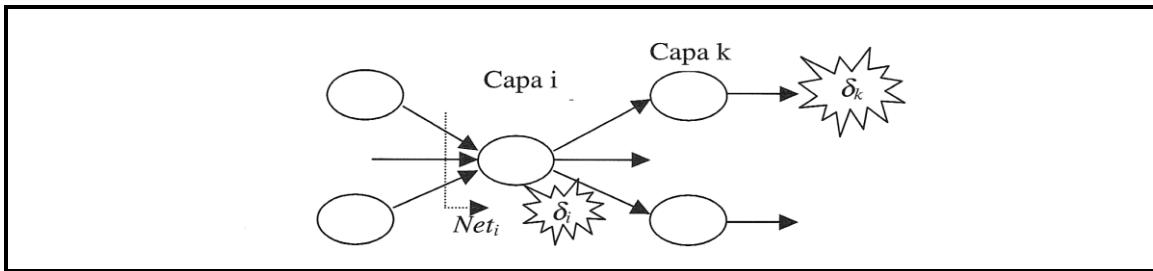
Esta fórmula es como la de la regla delta, excepto en el término de la derivada de la función de salida. Este término representa la modificación que hay que realizar en la entrada que recibe la neurona  $u_i$ . En el caso de que  $u_i$  no sea una unidad de salida, el error que se produce está en función del error que se comete en las unidades que reciben como entrada la salida de  $u_i$ . Por ello, a este procedimiento se le llama propagación del error hacia atrás .

- En el caso de que una neurona pertenezca a la capa de salida, se emplea:

$$\delta_i = f'(Net_i) * (d_i - y_i) \quad (17)$$

- En el caso de que una neurona no pertenezca a la capa de salida, se emplea:

$$\delta_i = f'(Net_i) \sum_k w_{ik} * \delta_k \quad (18)$$



**Figura 29.** Valores involucrados en la regla delta generalizada.  
(Bernal, Bobadilla y Gómez, 2000)

La principal idea que se debe obtener en este momento es que el algoritmo de aprendizaje Back Propagation se basa en la obtención de errores empezando en la capa de salida y retrocediendo sucesivamente hasta la capa de entrada. Una vez obtenido el error D de una neurona, se puede calcular la variación que hay que ir dando a los pesos W de entrada a esa neurona.

Este proceso se repite con sucesivos pares (patrón de entrada, valor de salida esperado), actuando sobre todas las neuronas, hasta que se considere que la red ha aprendido y proporciona valores de salida adecuados. Una vez establecida la forma de calcular los errores, la variación de los pesos que exige el aprendizaje se realiza de la forma habitual:

$$\Delta w_{ji} = \alpha * \delta_i * y_j \quad (19)$$

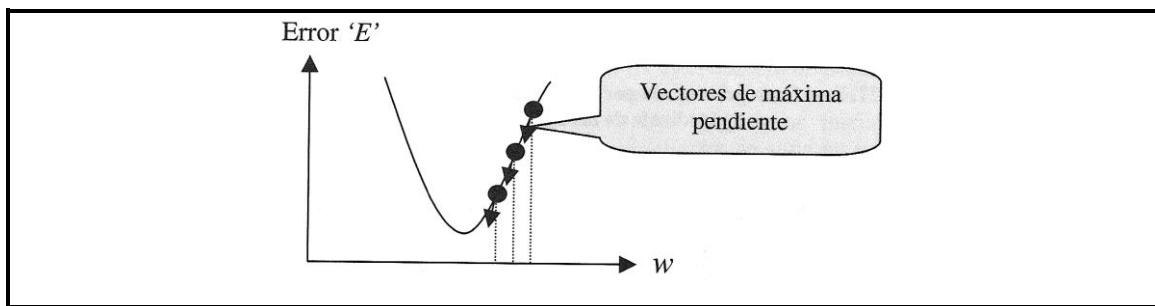
#### Algoritmo De Propagación Hacia Atrás

1. Inicializar los pesos con valores aleatorios pequeños
2. Tomar un patrón de entrada x escogido aleatoriamente
3. Propagar la señal hacia delante a través de toda la RN
4. Calcular los errores que se producen en la capa de salida, empleando  

$$\delta_i = f'(Net_i) * (d_i - y_i)$$
5. Calcular los errores pertenecientes a la capa anterior, empleando  

$$\delta_i = f'(Net_i) \sum_k w_{ik} * \delta_k$$
6. Volver al paso 5 hasta que se alcance la capa de entrada
7. Actualizar los pesos, empleando la formula 19.
8. Volver al paso 2 hasta que el error en la capa de salida sea menor que un umbral establecido o hasta que se haya alcanzado un número fijado de iteraciones

En el caso del perceptrón, si se tienen 'n' pesos que conectan las unidades de entrada con la unidad de salida, se puede obtener una superficie representada en un espacio de 'n+ 1' dimensiones, donde una de ellas es el error y el resto son los pesos. La regla delta encuentra un valor mínimo (local o global) en esa superficie mediante la aplicación de pasos descendentes por la misma. Este proceso se denomina gradiente descendiente. El párrafo anterior se expresa en la figura 30:



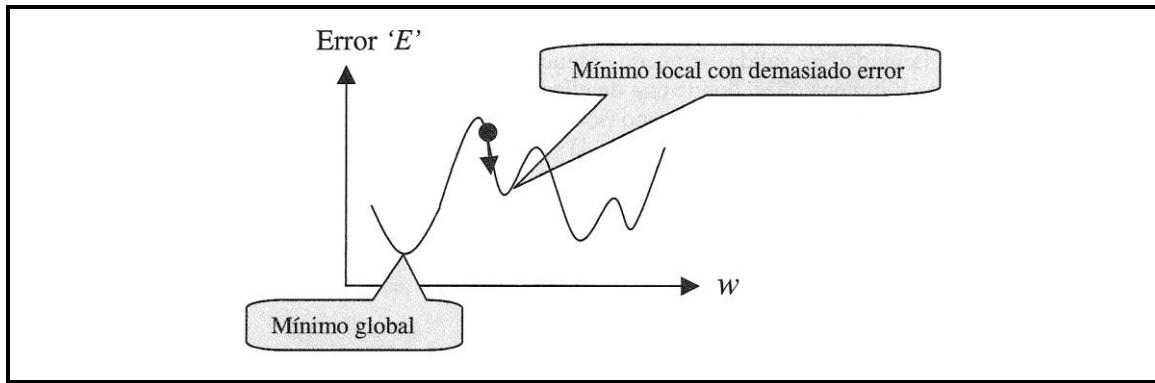
**Figura 30.** Método de gradiente descendente.  
(Bernal, Bobadilla y Gómez, 2000)

La hipérbola representa la superficie de error correspondiente a un perceptrón con función sigmoidal. El error mínimo corresponde con el mínimo de la hipérbola.

En este caso, solamente existe una dimensión, correspondiente a la variable 'peso', es decir, solamente disponemos de una neurona en la capa de entrada. En el caso de una red con dos neuronas en la capa de entrada, (p.e. la función AND) la hipérbola se dibujaría en el espacio tridimensional, y existirían dos dimensiones de pesos. Para más de dos neuronas en la capa de entrada se pasaría a hiperespacios de 4,5, etc. dimensiones.

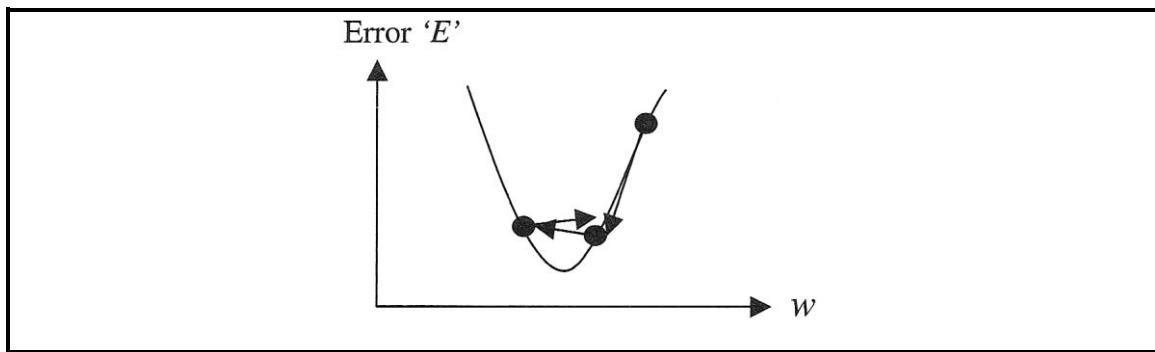
Cada símbolo dibujado en la hipérbola, representa un avance de gradiente negativo que nos acerca a la solución (el mínimo de la función). Estos avances se deben a las variaciones que producimos en los pesos al aplicar el mecanismo de aprendizaje.

Cuando se emplea una RN multicapa generalizadas la superficie de error es mucho más compleja. En este caso pueden existir multitud de mínimos y máximos locales que hacen imposible asegurar la convergencia del algoritmo en la mayor parte de las aplicaciones de reconocimiento.



**Figura 31.** Problemas presentados por los mínimos locales en la convergencia de Aprendizaje de las RN multicapa.  
(Bernal, Bobadilla y Gómez, 2000)

El factor de aprendizaje  $\alpha$  nos marca una proporción de magnitud de los pasos (variación de los pesos). Si  $\alpha$  es grande (cercano a uno) los pasos son grandes (las flechas más largas), por lo que se avanza más deprisa hacia la solución, pero puede darse un efecto de 'oscilación' alrededor del mínimo.

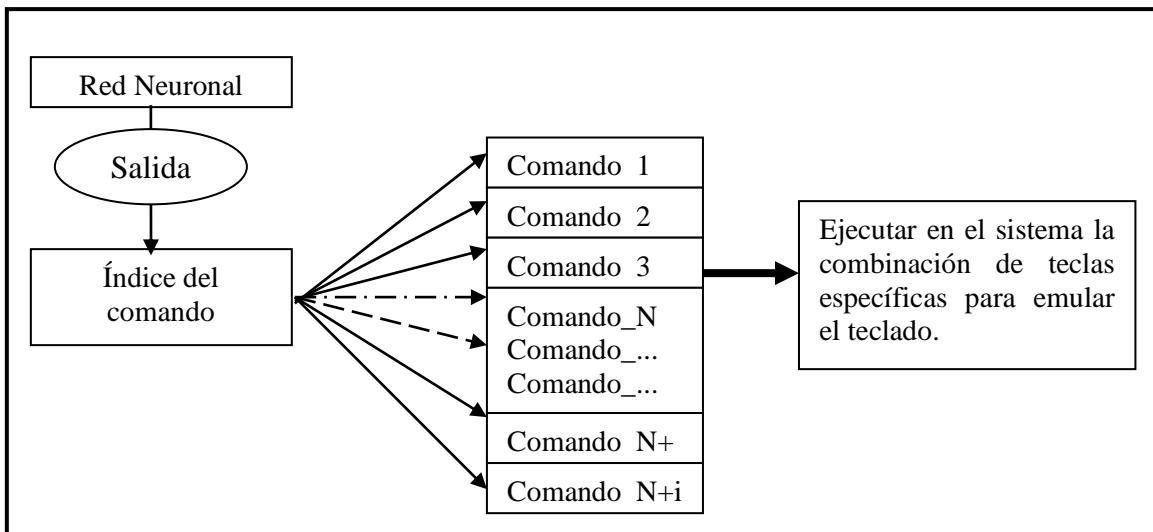


**Figura 32.** Problemas de oscilación cuando se toma un factor de aprendizaje Demasiado grande.  
(Bernal, Bobadilla y Gómez, 2000)

Cada punto en la superficie del error corresponde a un conjunto de valores de los pesos de la red. Con el gradiente descendente, siempre que se realiza un cambio en los pesos de la red, se asegura el descenso por la superficie del error. Así, el algoritmo sólo encuentra el valle más cercano, lo que puede hacer que caiga en un mínimo local. Sin embargo, ha de tenerse en cuenta que no tiene por qué alcanzarse el mínimo global de la superficie del error, sino que puede bastar con un error mínimo preestablecido.

#### 4.7. Interpretador de Comandos

En la etapa final del intérprete, permitirá ejecutar el comando que el usuario pronunció y que la red neuronal reconoció como un comando válido para su ejecución. Como se muestra en la figura 32, la cual describe el funcionamiento en un estado ideal, es decir, el comando se reconoció sin ningún tipo de error.



**Figura 33.** Funcionamiento del interpretador de comandos.

Cada comando representa una secuencia de teclas a ejecutar por el software, la implementación depende del lenguaje de programación a usar, el cual permitirá emular el teclado a través del software. Uno de los lenguajes que posee una interfaz que se puede usar es Java, tiene una clase llamada Robot (paquete java.awt.Robot) la cual permite realizar la emulación del teclado.

La clase Robot de Java proporciona eventos de entrada al sistema nativo con el propósito de automatización de pruebas, demos y otras aplicaciones que requieran el control del teclado y Mouse.

El sistema de reconocimiento de comandos de voz se propuso para ser implementado en Java dado que este tiene una interfaz que permite el control del teclado como ya se mencionó, y además es el lenguaje que se maneja con mayor propiedad entre los desarrolladores de este proyecto, siguiendo las técnicas de modelación de desarrollo de software (UML).

## 5. ESTADO DEL ARTE

La aplicación actual de reconocimiento de voz más desarrollada en estos momentos es Dragón Naturally Speaking con módulos de lenguaje técnico en inglés. Le siguen aplicaciones de GNU Linux Sphinx que es software de reconocimiento de voz que trae módulos incorporados de diferentes idiomas los cuales implementan diferentes aplicaciones para ejecutar comandos de voz.

En el mercado se encuentran actualmente muchas aplicaciones comerciales y demostrativas que solamente ejecutan una mínima cantidad de comandos en el sistema. Actualmente Microsoft posee un sistema de reconocimiento de voz para Windows en el idioma inglés y aplicaciones Web que utilizan comandos de voz, como VoiceXML. y existen exploradores como Opera que pueden ejecutar comandos de voz para navegar en Internet.

En la tabla 2 se describen algunas aplicaciones de reconocimiento de voz:

Producto	Características	Requerimientos
Dragon Naturally Speaking (Nuance Communications)	160 palabras por minuto. Mensajes de correo electrónico, mensajes instantáneos, documentos y hojas de cálculo. Controlar todas las funciones del ordenador. Idioma en español e Ingles.	Microsoft® Windows® XP (SP1 o superior) Home y Professional, Me, 2000 (SP4 o superior). Tarjeta de sonido Creative® Labs Sound Blaster® 16 o equivalente, compatible con grabación de 16 bits. Altavoces. Casco con micrófono con anulación de ruidos aprobado por ScanSoft.
Voice Xpress (Lernout & Hauspie)	Vocabulario de más de 530,000 palabras. Comandos naturales para Microsoft Windows y XpressPad. Español	Windows 98 o Windows NT Micrófono con auriculares de alta calidad.
IBM Via Voice Advanced Edition (ScanSoft)	Advanced acoustic model. Speakpad –el procesador de textos que incorpora el programa. Proceso de aprendizaje. La eficacia en el reconocimiento de la voz viene a ser de un 90%. Español.	Microsoft Word 97, 2000 y XP. Micrófono con cancelación de ruido.
Sphinx-4 (The CMU Sphinx Group Open Source Speech Recognition Engines)	Hidden Markov Models (HMM) Vocabulario extendido 64.000 palabras.	Java 2 SDK, Standard Edition 5.0 Licencia BSD, GPL.

**Tabla 2:** Aplicaciones destacadas en el reconocimiento de voz.

En la tabla 3 se muestran algunas publicaciones destacadas en el reconocimiento de voz:

Nombre	Descripción
Modelo de un Reconocedor de Palabras Aisladas e Independiente del Locutor. (Mejía, Vanegas, 2006)	Este desarrollo describe un reconocedor independiente del hablante el cual usa LPC y DTW el cual esta implementado en un dispositivo lógico programable (FPGA)
Reconocimiento de la Voz Mediante una Red Neuronal de Kohonen.  (Merlo, Fernández, Caram, Priegue, y García Martínez, 1997)	El reconocimiento de la voz mediante diversas técnicas tales como cadenas ocultas de Markov y Redes Neuronales es tema de investigación constante, obteniendo resultados de distinta desarrollo según el método elegido. En el presente trabajo se exhiben los resultados de una experiencia en reconocimiento de voz de un individuo, tomando como patrones a ser reconocidos las cifras decimales (0-9), y utilizando como método una red neuronal de Kohonen. Luego de una fase de entrenamiento y sintonización, produce con solo cien neuronas un aceptable resultado de reconocimiento (65%).
Reconocimiento del Habla Basado en Grafemas. Grapheme based Speech Recognition. (Killer, Stüker, Schultz, 2006)	<p>Los sistemas de reconocimiento del habla de vocabulario grandes representan las palabras tradicionalmente en relación con unidades de subpalabra, generalmente fonemas.</p> <p>Este trabajo investiga el potencial del desempeño de grafemas como subunidades.</p> <p>Para desarrollar reconocedores dependiente del hablante basado en grafemas, algunos procedimientos basados en árboles de decisión son llevados a cabo y comparados con otros componentes del mismo árbol.</p> <p>Los reconocedores del habla basados en grafemas para las tres lenguas - inglés, alemán, y español - son entrenados y comparado con sus homólogos.</p> <p>Los resultados muestran los lenguajes con una cerrada relación de grafemas a fonema, el modelamiento basado grafemas en es tan bueno como el basado en fonemas. Además los reconocedores multilenguas basados en grafemas son diseñados investigar si la información basada en grafemas puede ser compartido con éxito entre varias lenguas.</p>
Reconocimiento del habla utilizando redes neuronales. Speech Recognition using Neural Networks (Tebelskis, 1995)	<p>Esta tesis revisa cómo las redes neuronales artificiales pueden beneficiar a un gran vocabulario, independiente del hablante en un Sistema continuo de reconocimiento del habla.</p> <p>La mayoría de los sistemas de reconocimiento del habla están basados en los modelos de Markov ocultos (HMMs), un marco estadístico que soporta el modelado tanto acústico como temporal.</p> <p>A pesar de su rendimiento de punta, HMMs hace varias suposiciones al modelado que limitan su eficacia potencial.</p> <p>Las redes neuronales evitan muchas de estas suposiciones, mientras que también pueden aprender las funciones complicadas, generalizar eficazmente, tolerar el ruido, y respaldar el paralelismo.</p> <p>Mientras las redes neuronales pueden ser aplicadas al modelado acústico fácilmente, no está todavía claro cómo pueden ser usados para el modelado temporal.</p> <p>Por lo tanto, se analiza una clase de sistemas llamados híbridos de NN - HMM, en los que las redes neuronales llevan a cabo el modelado acústico, y HMMs llevan a cabo el modelado temporal.</p>

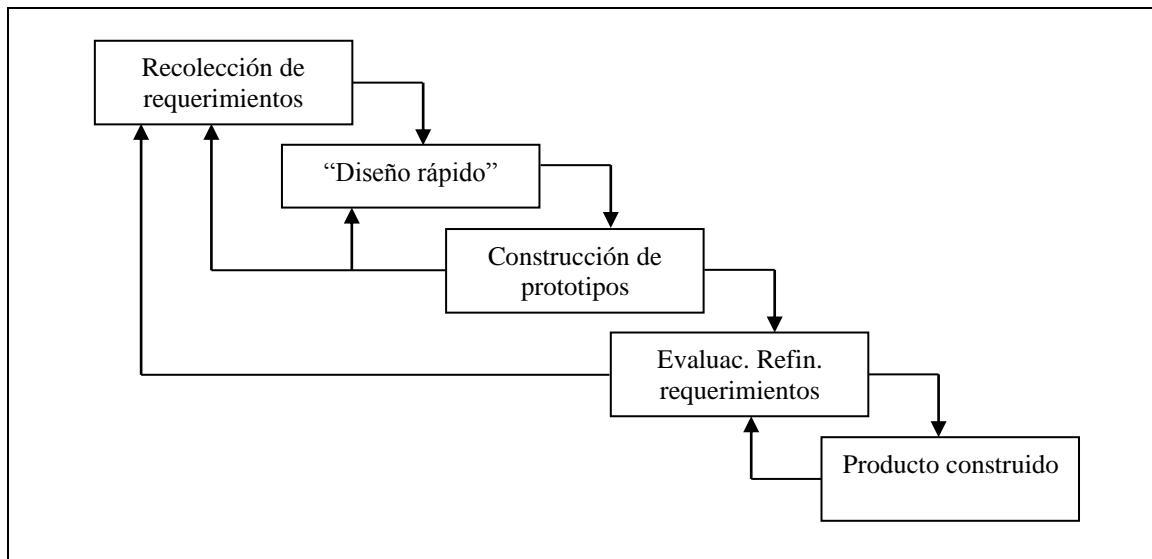
**Tabla 3:** Publicaciones sobre Reconocimiento de Voz.

## 6. DISEÑO

En el diseño se lleva a cabo el proceso del desarrollo de software, el que se uso en este proyecto lo componen tres partes que son:

- a. Definición de los requerimientos.
- b. Diseño e implementación.
- c. Pruebas.

El paradigma que se utilizo al desarrollar el sistema de reconocimiento de comandos de voz en español, es el de Orientado a Objetos, bajo el ciclo de vida de prototipos, figura 34, permite asegurar que el trabajo se esta realizando bien y que se esta cumpliendo con los requerimientos y metas establecidas en el proyecto.



**Figura 34.** Diagrama de Paradigma de Prototipos.

En esta sección se incluyen los comandos seleccionados para el interpretador de comandos de voz, un diseño sencillo de la estructura de datos de la red neuronal artificial.

## 6.1. Comandos Del Teclado

Para la selección de los comandos más utilizados se uso el sistema de método abreviado del teclado de Windows, el cual los divide en cinco secciones que se muestran en la tabla 4.

Métodos abreviados de teclado para Windows
Métodos abreviados de teclado generales
Métodos abreviados de teclado en los cuadros de diálogo
Métodos abreviados de Natural Keyboard
Métodos abreviados de teclado de accesibilidad
Métodos abreviados de teclado del Explorador de Windows

**Tabla 4:** Métodos abreviados de teclado para Windows.

Para observar los métodos abreviados de teclado de Windows ver Anexo A.

Aunque cada aplicación tiene sus propios métodos abreviados para el teclado como los que se encuentran en la tabla 5, pero con la diferencia de que si la aplicación es en inglés no coincide cuando sea en español.

Comando	Descripción	Idioma
Crtl + A	Abrir un archivo	Español
Crtl + O	Abrir un archivo	Inglés
Crtl + N	Nuevo archivo	Inglés
Crtl + U	Nuevo archivo	Español
Crtl + G	Guardar archivo	Español
Crtl + S	Guardar archivo	Inglés
Crtl + P	Imprimir documento	Inglés, Español
Crtl + E	Seleccionar todo	Español
Crtl + A	Seleccionar todo	Inglés

**Tabla 5:** Métodos abreviados de teclado para algunas aplicaciones.

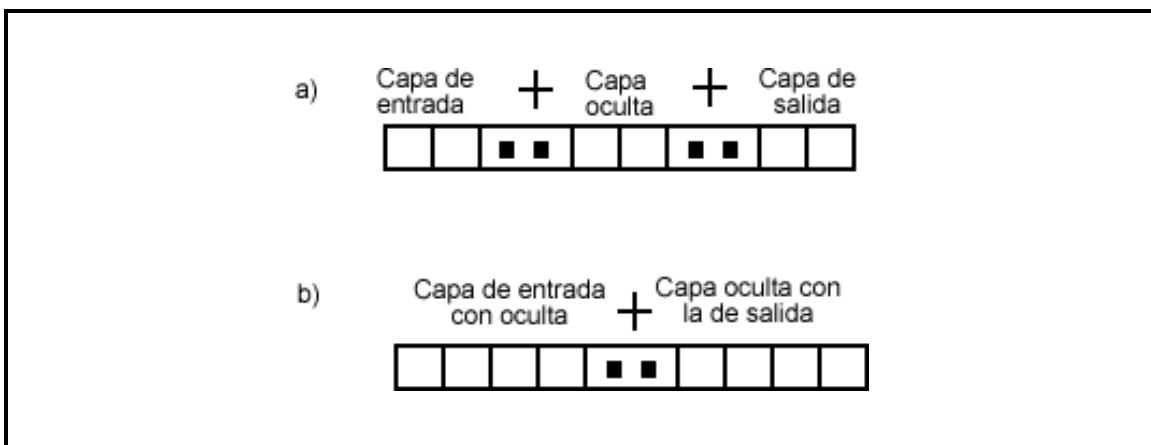
En la tabla 6 se muestran los veinte comandos escogidos para la realización del interpretador de comandos.

<b>Comando</b>	<b>Tecla (s)</b>
Enter	Enter
Arriba	FlechaArriba
Abajo	FlechaAbajo
Izquierda	Flecha Izq.
Derecha	Flecha Der.
Copiar.	Ctrl+C
Cortar.	Ctrl+X
Pegar.	Ctrl+V
Deshacer.	Ctrl+Z
Menú	Alt
Salir	Esc
Inicio	Win
Alternar	Alt+Tab
Explorador	Win+E
Cerrar	Alt+F4
Guardar	Ctrl+G
Borrar	Supr
Seleccionar	Ctrl+A
Escritorio	Win+D
Ayuda	F1

**Tabla 6:** Comandos del interpretador.

## 6.2. Red Neuronal Artificial

Para la realización del intérprete de comandos de voz se decidió utilizar una red neuronal multicapa con propagación hacia adelante y un algoritmo de entrenamiento llamado BackPropagation. La red se conforma de tres capas (capa de entrada, capa oculta, capa de salida), la cual se manejará en un solo vector (ver figura 35a), la matriz de pesos se manipulará en solo vector (ver Figura 35b) y se utilizará un vector de manera similar para los umbrales.



**Figura 35.** Estructura de datos de la red neuronal.

a) Vector de Neuronas. b) Vector de pesos.

El vector que guarda los pesos de las neuronas se almacena de forma contigua, es decir, la primera neurona de la capa de entrada (comenzando por la derecha) con cada una de la capa oculta, la segunda neurona de la capa de entrada con cada una de la capa oculta, etc. y así sucesivamente para la capa oculta contra la de salida. Para realizar otros cálculos se requiere manejar la taza de aprendizaje, el momentum el cual es usado para el entrenamiento, los umbrales, los deltas para la actualización de la red neuronal, cálculo del error, y un vector de pesos adicionales como variable auxiliar cuando la red se esta entrenando.

Para determinar la cantidad de neuronas en cada capa se realiza una aproximación de la capa de entrada [128, 260]. La capa oculta contiene dos tercios de la capa de entrada y la capa de salida ya se encuentra pre-establecida con veinte neuronas (veinte comandos).

### 6.3. Análisis de Requerimientos

Los siguientes requerimientos son los más relevantes, para ver los otros requerimientos que complementan el desarrollo ver Anexo B.

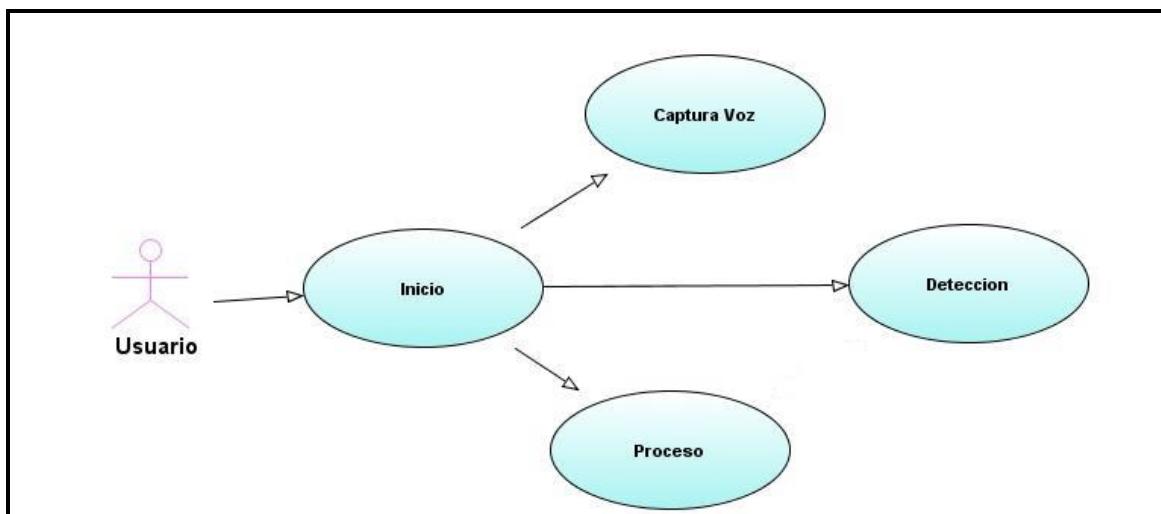
Requerimientos		
Número	Funciones	Categoría
1.	El interpretador de comandos detectará la presencia en la señal de audio del usuario, la cual se extraerá para su procesamiento.	Oculto
2.	El interpretador implementará un algoritmo de detección de bordes para analizar si en la señal hay presencia de la voz del usuario	Oculto
3.	La parte de la señal extraída como un comando de voz pronunciado por el usuario, el interpretador deberá extraerle sus componentes frecuenciales para construir el patrón que se analizara	Oculto
4.	El interpretador implementara una red neuronal artificial con manejo de memoria, como herramienta para identificación de patrones.	Oculto
5.	La red neuronal debe tener un entrenamiento previo para que la voz de los usuarios sea identificada por el interpretador de comandos de voz.	Oculto
6.	El interpretador de comandos de voz deberá poseer una interfaz gráfica que el permita configurar al usuario, la configuración del perfil, edición y/o modificación de cada comando de voz, a la interfaz de configuración se le denominara asistente de configuración.	Visible
7.	El sistema de producción o ejecución del comando de voz no debe contar con interfaz de usuario, solo se mostrara mensajes salientes del icono de la barra del sistema describiendo el comando que ha pronunciado.	Visible
8.	En el proceso de edición, cuando el usuario detenga la grabación, el sistema realizara el proceso de detección de	Oculto

	bordes y extracción de características espectrales en el caso de que se haya encontrado información en la grabación, y los datos permanecerán en memoria hasta que el usuario guarde o cancele los cambios.	
9.	En el interpretador de comandos, cuando el usuario pronuncie un comando que sea válido o esté habilitado, el sistema debe mostrar un mensaje en el ícono de la barra del sistema diciendo que comando acaba de decir.	Visible
10.	El sistema cuando no este activado debe mostrar un ícono en la barra del sistema en escala de grises, en caso contrario a color.	Visible

**Tabla 7:** Análisis de Requerimientos.

#### 6.4. Casos de Uso

El caso de uso de la figura 36 visualiza el comportamiento de la captura de voz, al momento de ejecutar un comando de voz. Para visualizar los otros casos de usos ver Anexo C.



**Figura 36.** Caso de uso captura de voz.

INFORMACIÓN GENERAL	
<b>Nombre:</b>	Captura de Voz
<b>Actores:</b>	Usuario
<b>Propósito:</b>	Capturar la voz humana por medio del micrófono
<b>Resumen:</b>	La aplicación detecta la voz humana y la captura para después ser interpretada y ejecutada
<b>Referencias Cruzadas:</b>	Requerimiento N°: 36

<i>i.Curso Normal de los Eventos</i>	
<i>Acción de los Actores</i>	<i>Respuesta del Sistema</i>
<p>1. El usuario inicia la aplicación.</p> <p>2. El usuario habla</p>	<p>3. El sistema detectará la presencia de señal de audio del usuario.</p> <p>4. El sistema captura la voz</p>

INFORMACIÓN GENERAL	
<b>Nombre:</b>	Detección
<b>Actores:</b>	sistema
<b>Propósito:</b>	Detectar el comando pronunciado por el usuario
<b>Resumen:</b>	La aplicación cuando detecta la voz humana pasa a reconocer el comando pronunciado.
<b>Referencias Cruzadas:</b>	Requerimiento N°: 7-8-32

<i>ii.Curso Normal de los Eventos</i>	
<i>Acción de los Actores</i>	<i>Respuesta del Sistema</i>
	<p>1. El sistema utiliza un algoritmo de detección de bordes.</p> <p>2. Guarda la información en un paquete.</p>

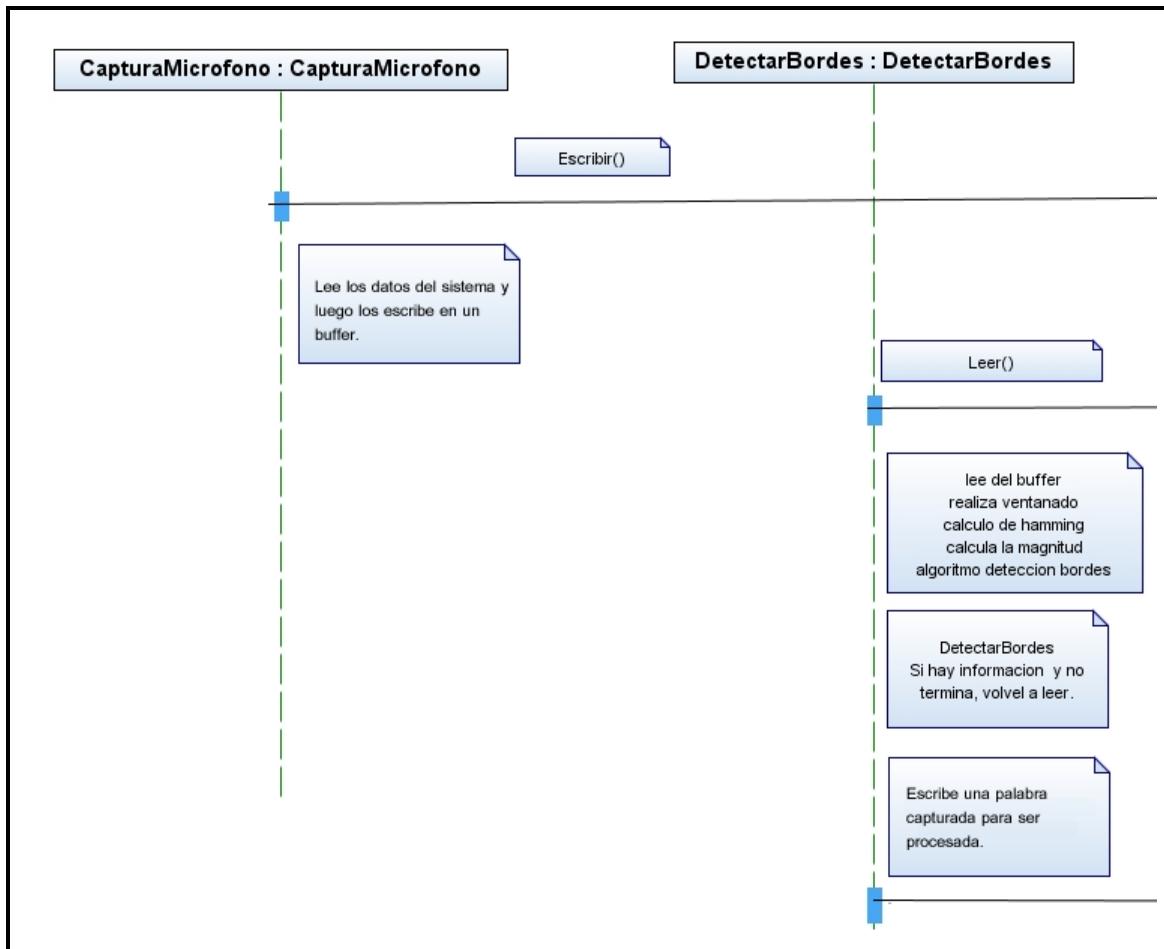
INFORMACIÓN GENERAL	
<i>Nombre:</i>	Proceso
<i>Actores:</i>	Usuario
<i>Propósito:</i>	Procesa el comando pronunciado a partir una serie de cálculos como lo son: calculo del espectro, proceso de la red neuronal y envío de los valores calculados al dispositivo.
<i>Resumen:</i>	Procesa los comandos pronunciados por el usuario a través de una serie de cálculos y los ejecuta.
<i>Referencias Cruzadas:</i>	Requerimiento N°: 9-10-35-37-45

<i>iii.Curso Normal de los Eventos</i>	
<i>Acción de los Actores</i>	<i>Respuesta del Sistema</i>
	<ol style="list-style-type: none"> <li>1. Lee buffer el comando guardado.</li> <li>2. Calcula del espectro de la señal.</li> <li>3. La red neuronal procesa el comando y calcula su salida.</li> <li>4. Se envía la orden para que se ejecute el comando pronunciado en el dispositivo.</li> </ol>

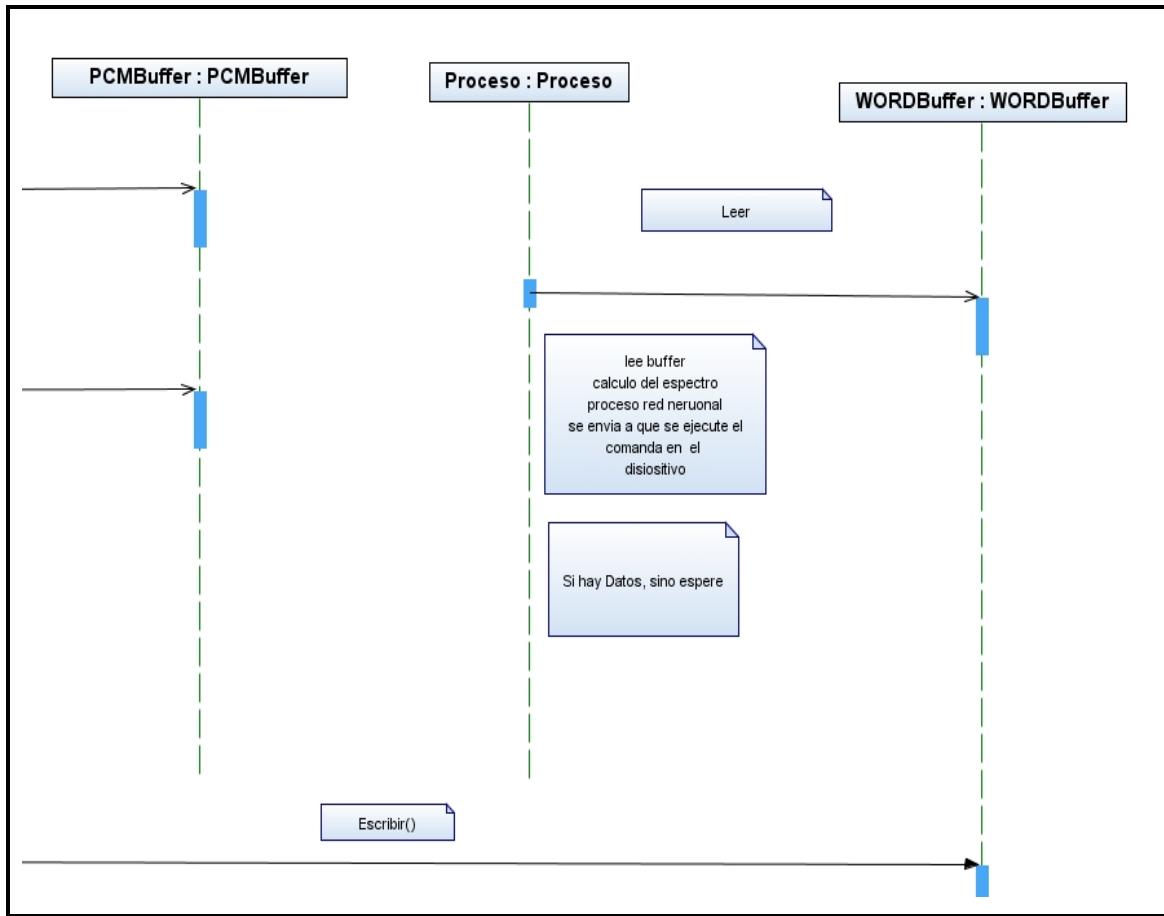
**Tabla 8:** Caso de Uso captura de voz.

## 6.5. Diagrama de Secuencia.

El diagrama de la figura 37 muestra la secuencia de captura de la voz hasta la ejecución del comando. Para ver los otros diagramas ver Anexo D.

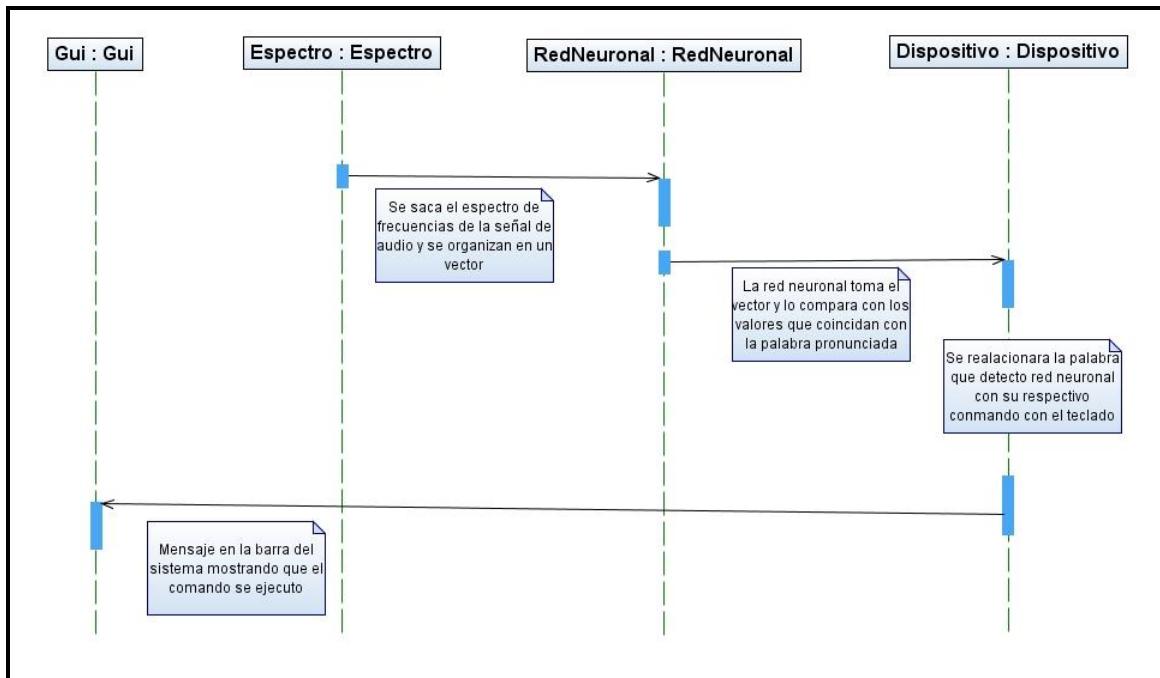


**Figura 37.** Diagrama de Secuencia: captura de voz. Parte a.



**Figura 37.** Diagrama de Secuencia: captura de voz. Parte b.

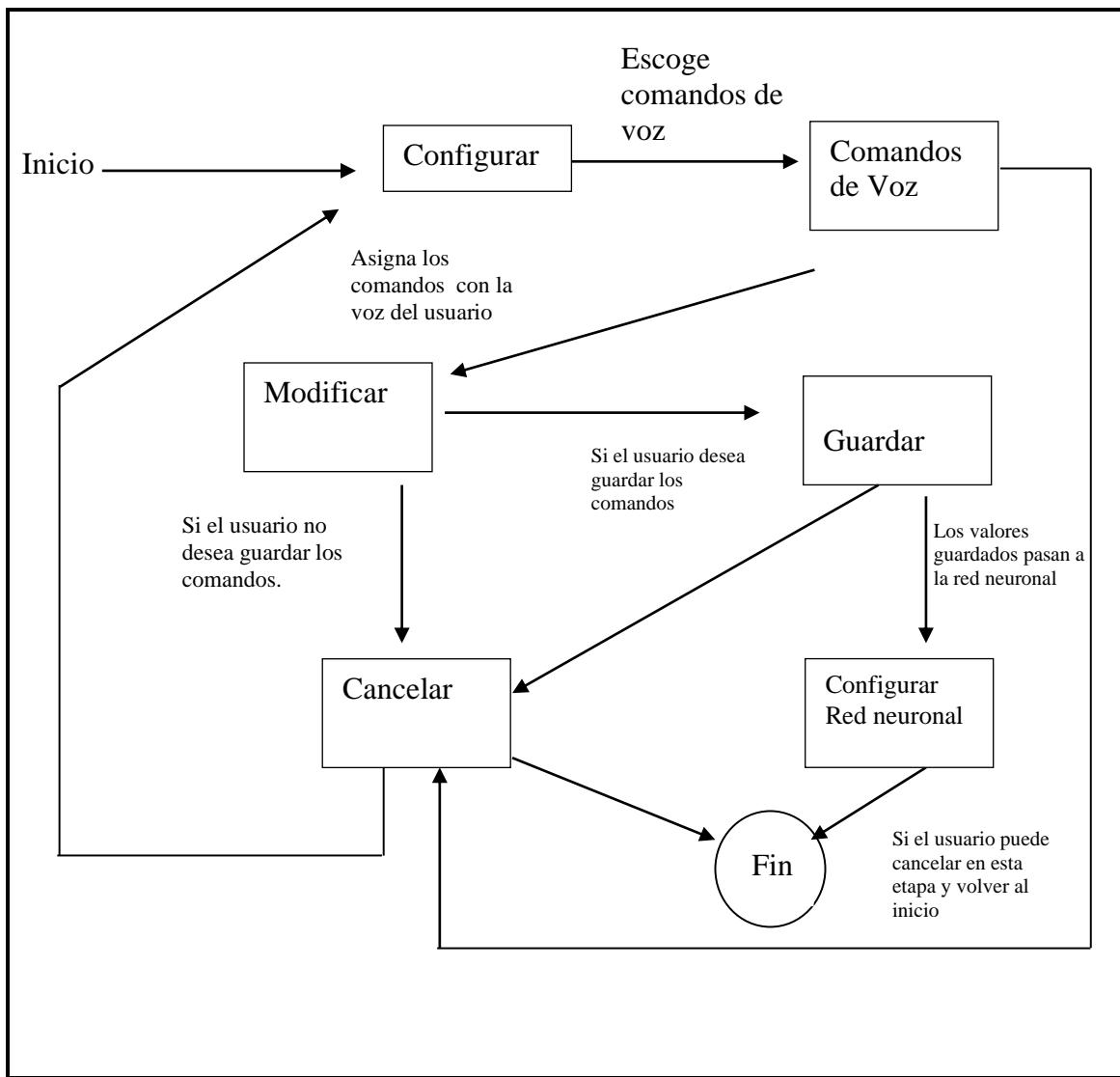
El diagrama de la figura 38 muestra un sub-proceso interno para llegar a ejecutar el comando de voz.



**Figura 38.** Diagrama de secuencia captura de voz (continuación).

## 6.6. Diagrama de Estados

La figura 39 muestra el diagrama de estado con respecto a la configuración de la aplicación.



**Figura 39.** Diagrama de Estado: Configuración.

## 6.7. Diagrama de Clases

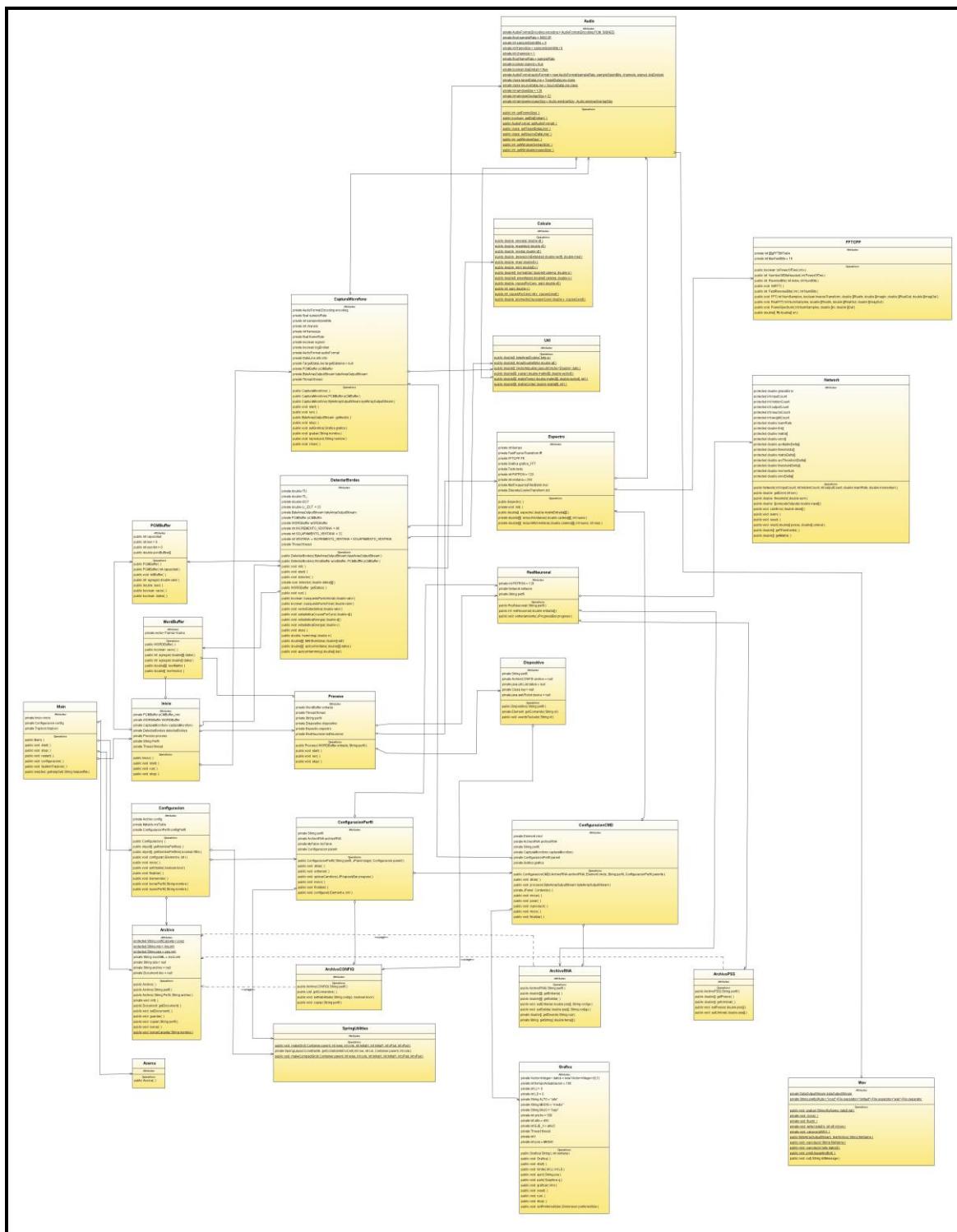
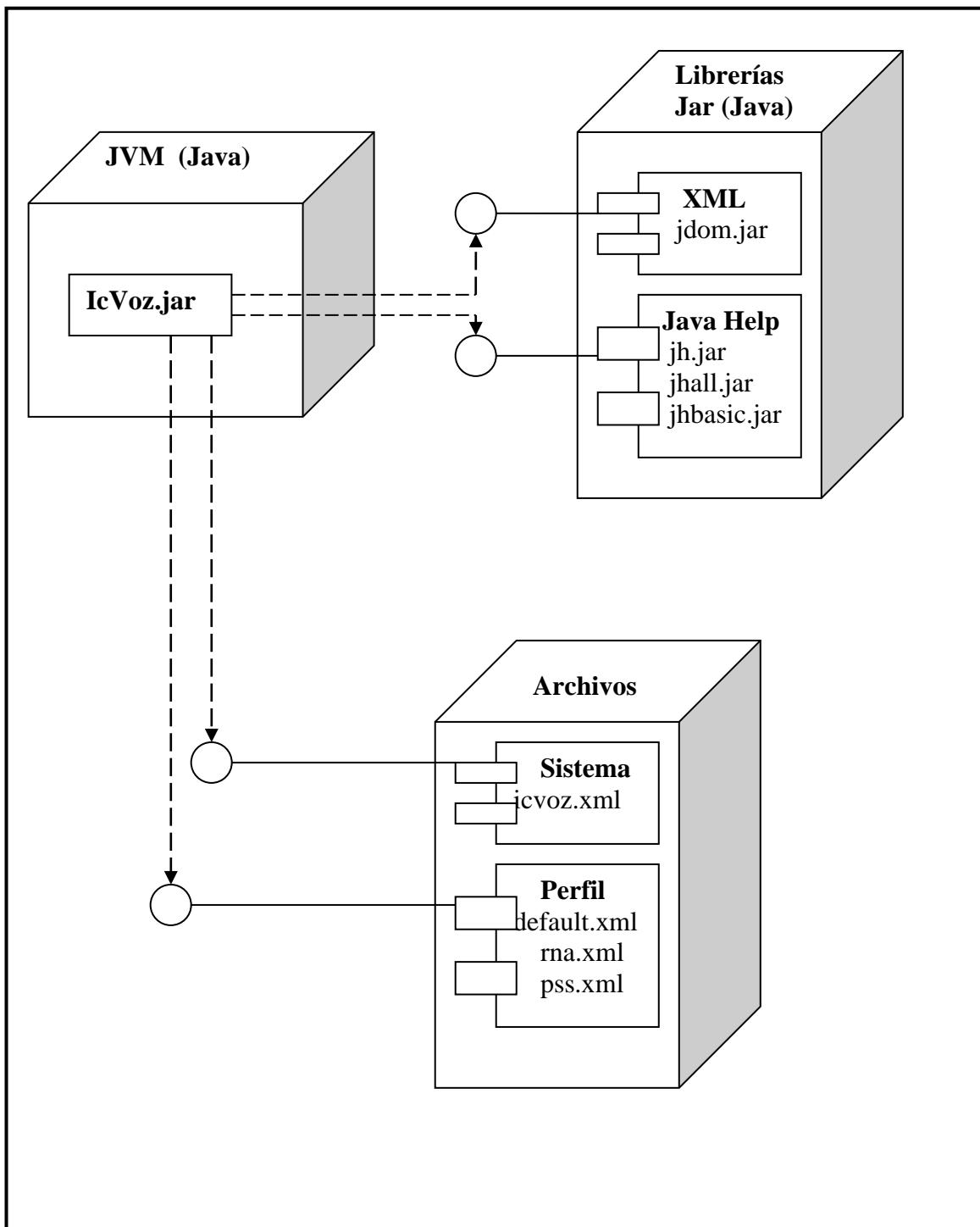


Figura 40. Diagrama de Clases (IcVoz).

Ver Anexo E para ver el diagrama más detallado.

## 6.8. Diagrama de Componentes

El diagrama de la figura 41 describe la interacción de la aplicación con su entorno. (Archivos y librerías).



**Figura 41.** Diagrama de componentes: Ejecutable.

## **7. DESARROLLO (IMPLEMENTACIÓN)**

El producto de éste trabajo dio como resultado un Interpretador de Comandos de Voz o su acrónimo IcVoz, con veinte comandos y mono-usuario. Tiene la capacidad de crear o borrar usuarios (perfiles), escoger cual es el perfil a usar y activar/desactivar los comandos a utilizar. Implementado totalmente en Java, versión 1.6.0 bajo el sistema operativo Windows XP SP2.

La implementación del sistema de comandos de voz en español se divide en dos secciones:

### **Sección Uno:**

Sistema de Entrenamiento:

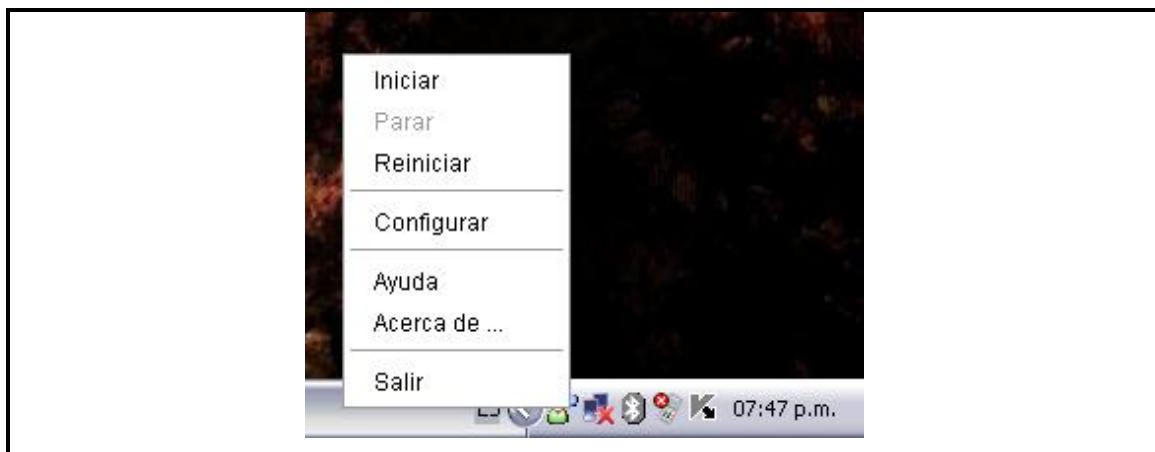
- Realizar la grabación de la voz
- Detector de bordes
- Extracción de características de la voz
- Entrenamiento de la red Neuronal.

### **Sección Dos:**

Sistema de Producción:

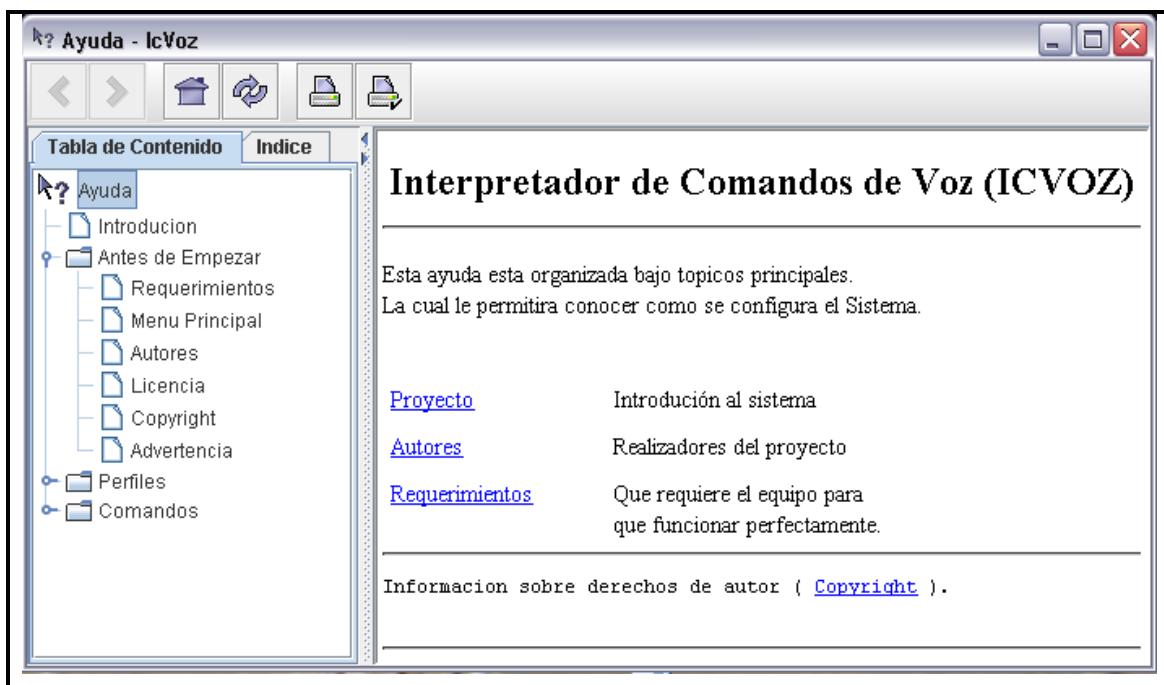
- Capturar del medio la voz (a través del micrófono)
- Detectar la voz
- Extracción de Características de la voz
- Ejecución de la red neuronal
- Ejecución del comando en el sistema.

Inicialmente la aplicación cuenta con un ícono en la barra del sistema en la cual el usuario maneja IcVoz, esta opción de poner un ícono, la presta la versión de Java 1.6.x además de un menú para el usuario como se observa en la figura 42.



**Figura 42.** Menú e Icono en la barra de tareas.

La ayuda se construyó con una API de java (JavaHelp 2.0) que proporcionan una herramienta útil, la cual solo hay que configurarla para su funcionamiento, y proporcionarle los archivos necesarios de la ayuda de IcVoz.



**Figura 43.** Ayuda de la aplicación.

Los puntos siguientes muestran porciones de código y diagramas que muestran la implementación que se obtuvo.

## 7.1. Manejo de hilos en Java

Principalmente como el IcVoz es un sistema de procesamiento continuo, todos los hilos se manejan de una forma similar, a continuación se muestra una clase llamada prueba, con la estructura básica que siguen todas las clases que son hilos.

```
public class Prueba implements Runnable
{
    private Thread thread;

    public Prueba()
    {
        return;
    }

    public void start()
    {
        thread = new Thread(this,"Mi Hilo");
        thread.start();
    }

    public void run()
    {
        while(thread!=null)// while(!thread.interrupted())
        {
            //Trabajo continuo
        }
    }

    public void stop()
    {
        if (thread != null)
        {
            thread.interrupt();
        }
        thread = null;
    }
}
```

Para su llamado inicializamos un objeto y lo llamamos “prueba.start()” para iniciar o comenzar a correr el hilo, para detenerlo se implementó un método llamado “stop”. Debido a que se tiene que realizar procesamiento en paralelo las siguientes clases son hilos: Inicio, CapturaMicrofono, DetectarBordes y Proceso

## 7.2. Archivos de Configuración

Para que IcVoz almacene la información correspondiente se decidió manejar archivos planos con formato en XML, la API que se utilizo para procesar el archivo XML es de JDOM Project, el formato XML presta una organización de datos la cual pude ser fácilmente modificable, además como no se requieren grandes cantidades de datos no se realizó con un motor de Bases de Datos.

Los archivos se organizan por nombres de perfiles y un archivo principal único que permite saber los perfiles existentes y el perfil principal actual. La organización se muestra de la siguiente forma:

S.O. - Carpeta donde se instale la aplicación.

```
| - icvoz
  | - icvoz.xml
  | - default
  |   | - default.xml
  |   | - rna.xml
  |   | - pss.xml
  |   | - wav
  |   |   | - * . wav
  |
  | - andres
  |   | - andres.xml
  |   | - rna.xml
  |   | - pss.xml
  |   | - wav
  |   |   | - * . wav
  |
  | - christian
  |   | - christian.xml
  |   | - pss.xml
  |   | - rna.xml
  |   | - wav
  |   |   | - * . wav
```

**Figura 44.** Ordenamiento de los archivos de configuración.

El nombre de la carpeta raíz es “icvoz”, el archivo “icvoz.xml” es el archivo de configuración de perfiles y una carpeta llamada “default” la cual lleva el nombre que el

usuario le ha dado, pero “default” viene por defecto como el perfil base del cual se desprenden los demás perfiles, dentro esta carpeta se ubican el archivo “default.xml” el cual lleva el nombre del perfil y el nombre de la carpeta, esta contiene la lista de comandos, el archivo “rna.xml” contiene la entrada y correspondiente salida de la red neuronal el cual es utilizado para realizar el entrenamiento, el archivo “pss.xml” contiene los pesos y umbrales pertenecientes a la rede neuronal que se guardan después de cada entrenamiento y se carga al poner a funcionar la red neuronal y por ultimo se encuentra la carpeta “wav” la cual contiene las grabaciones realizadas por el usuario.

Para manejar los diferentes tipos o estructuras de los archivos se construyo una clase padre llamada “Archivo”, esta clase es la encargada de manejar el archivo icvox.xml, la cual se usa como parente para los demás archivos por operaciones que son comunes para todos, como son: abrir, obtener la información del documento, guardar los cambios y copiar el archivo hacia otro perfil.

La clase encargada de realizar manejar el archivo propio del perfil del usuario ósea “deafult.xml” es “ArchivoCONFIG”, la cual implementa un método adicional que es la obtención de la lista de comando y sobrescribe el método copiar.

La clase encargada de almacenar la entrada y salida para la red neuronal es “ArchivoRNA” la cual es utilizada para realizar el entrenamiento.

Y la última clase la cual es la encargada de guardar los pesos y umbrales de la red neuronal es “ ArchivoPSS”.

Los archivos en formato WaveForm son grabados por la clase “Wav”, esta clase es la encargada de guardar y reproducir el audio ya sea directamente desde el archivo que esta ubicando dentro de la carpeta Wav o reproducir el audio realizado en una grabación que se encuentra en un buffer temporal del sistema.

### **7.3. Captura de la Voz**

La clase que se utiliza para capturar el audio se llama CapturaMicrofono la cual se implementa como hilo para capturar solo el audio que proviene del micrófono. Pero se utiliza una clase donde se tiene la configuración en la cual se debe extraer los datos llamada Audio.java, a continuación se muestra la configuración utilizada para capturar el sonido a través del micrófono:

```
this.encoding =           AudioFormat.Encoding.PCM_SIGNED;
this.sampleRate=          11025.0F;
this.sampleSizeInBits =    8;
this.channels =           1;
this.frameSize =          1;
this.frameRate =          11025.0F;
this.signed =              true;
this.bigEndian =           true;
```

La descripción es:

Codificación: PCM con signo.

Muestreo: 11025 Hz

Canales: 1

Marcos: 1

MuestreoMarcos: 11025 Hz

Con signo: Si

Bit significativo: Alto.

Para el proceso de grabación y obtención en tiempo real se procede a guardar en un buffer, para la grabación se utiliza una clase de java java.io.ByteArrayOutputStream y para el tiempo real se almacena en una estructura construida para esto, de tipo cola buffer.PCMBuffer, para almacenar se activa y desactiva por condicional el tipo de buffer a utilizar.

```

public void run()
{
    System.out.println("[Captura de audio iniciada]");
    int off=0; //Inicio
    int len=this.frameSize; //Cantidad de bits ha leer
    byte[] b = new byte[len];// Se almacena los bits leídos
    targetDataLine.start(); //Iniciar la línea de entrada (micrófono)
    if(byteArrayOutputStream!=null)//Buffer para grabación
    {
        while(thread != null)
        {
            targetDataLine.read(b,off,len);
            byteArrayOutputStream.write(b,off,len);
            // Graficar
            for(int i=off; i<len; i++)
            {
                grafica_PCM.graficar((int)b[i]);
            }
        }
    }else{//Buffer para obtención en tiempo real
        while(thread != null)
        {
            targetDataLine.read(b,off,len);
            for(int i=off; i<len; i++)
            {
                //Esperar a que le buffer este vacío
                while(!pCMBuffer.vacio())
                {
                    thread.yield();
                }
                pCMBuffer.agregar((double)b[i]);
                grafica_PCM.graficar((int)b[i]);
            }
        }
    }
}

```

## 7.4. Detección de bordes

Para la detección de bordes se uso un algoritmo presentado por Rabiner y Sambur (Rabiner y Sambur, 1975), y modificado por (Merlo, Caram y García, 1997), al cual nosotros realizamos modificaciones para adaptarlo en función al tiempo real y no a una grabación como originalmente estableció.

La detección de bordes, detección de la palabra pronunciada por el usuario se realizan en dos operaciones diferentes dependiendo de la utilización, es decir, cuando se busca en una grabación a cuando se realiza buscando en la señal de audio en tiempo real, aunque el método en sí es el mismo.

Comenzamos creando ventanas de 10 ms de audio equivalentes a 128 datos en un vector, procedemos en la creación a realizar solapamiento de 25% y aplicar una ventana de Hamming, después de realizar este proceso tenemos una matriz de datos nxm (ventanas x 128).

Con la matriz de ventanas procedemos a crear un vector de longitud n (ventanas) donde se almacenara el cálculo de la magnitud de cada ventana, ver formula 20, a partir de este se calcula los umbrales para de terminar si hay o no principio de una señal y final de una señal, los cálculos son los siguientes:

$$M(n) = \frac{1}{N} \sum_{m=0}^{N-1} |x(m)| \quad (20)$$

IMX = A la mayor magnitud

IMN = Mínima magnitud

I1 = (0.03\*(IMX-IMN)) + IMN;

I2=4.0\*IMN;

ITL= min(I1,I2)

ITU= 4.0\*ITL

Las variables I1 e I2 son auxiliares la ITL y ITU corresponde a ITL = Internal Treshold Low y ITU Internal Treshold Upper , en español umbral de máximo intervalo y umbral de mínimo intervalo, al calcular estas variables se construye una función que indica cuando se comienza o termina una palabra, las siguientes condiciones indican si hay presencia de audio o comienza un mensaje en el audio:

Determina si hay presencia de información en la grabación.

M = magnitud en un punto dado.

Si M > ITL entonces

Si M < ITU entonces

Falso

Sino

Si M > ITU entonces

Verdadero Sino

Falso Sino

Falso

Determina si finalizo la información en la grabación.

M = magnitud en un punto dado

Si M > ITL entonces

Si M > ITU entonces

Falso

Sino

Si M < ITU entonces

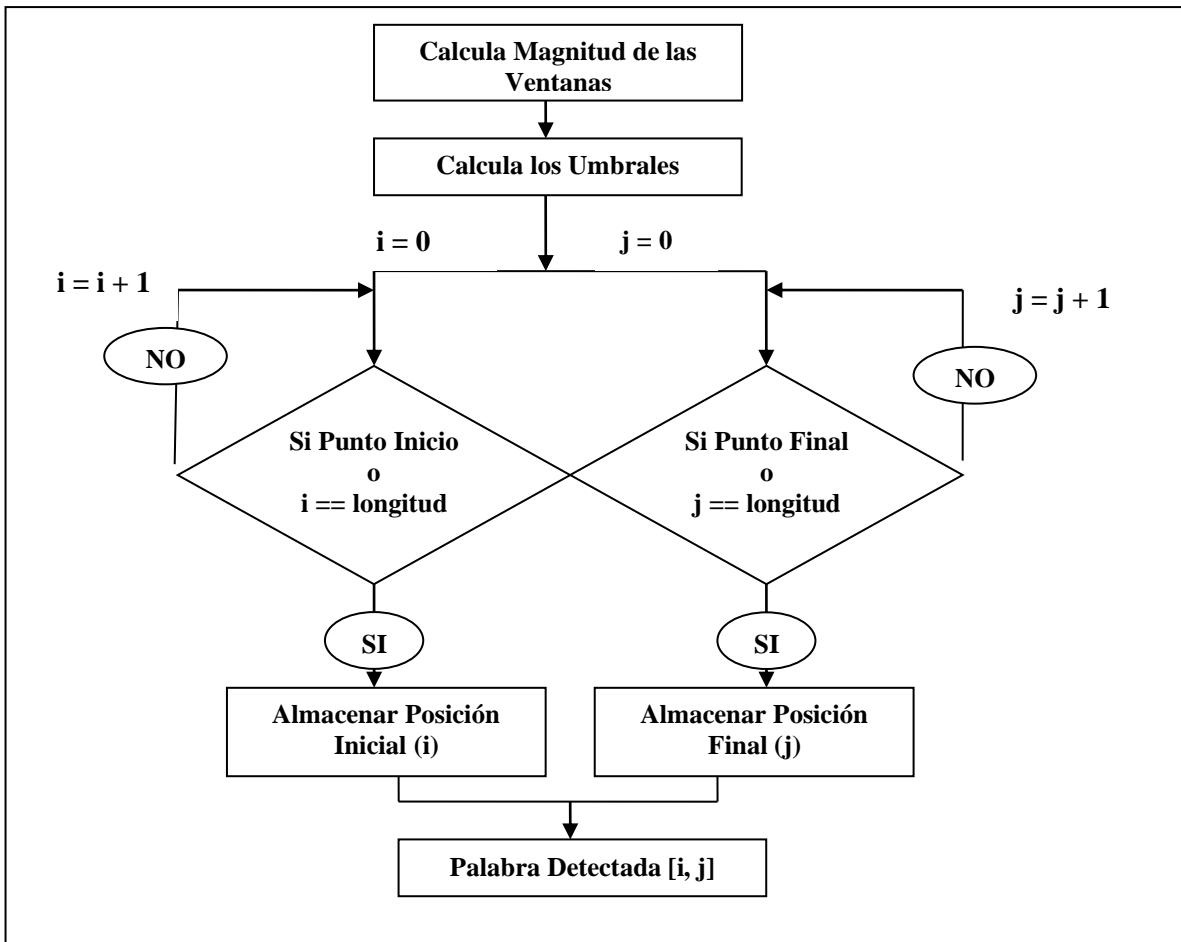
Falso Sino

Falso sino

Verdadero

Para iniciar el proceso se inicia desde la primer ventana, se le calcula la magnitud y si en esta se determina que hay presencia de información se pasa luego a buscar el punto final con la otra condición, en caso contrario se busca en el resto de las ventanas.

Como existe el problema de un corte en la información se agrega una condición que se debe tener 11 intervalos o 110 ms de silencio dentro del rango de ITL para poder dar por finalizada la palabra, esta condición de tiempo se realizó a prueba y error.



**Figura 45.** Diagrama que muestra el proceso de detección. (en grabación).

Para realizar este mismo proceso en tiempo real ya que no se tiene un final como en una grabación, se hizo el mismo proceso pero con una diferencia a la hora de hacer los cálculos. Debido a que no se cuenta con un máximo y un mínimo de magnitud, el máximo equivale al cálculo de magnitud de una primera ventana de 10 ms sin solapamiento y para el valor mínimo aplicamos el logaritmo en base 10 de IMX dividido en 2. Como se muestra a continuación.

$$IMX = M$$

$$IMN = \log_{10}(IMX) / 2.0$$

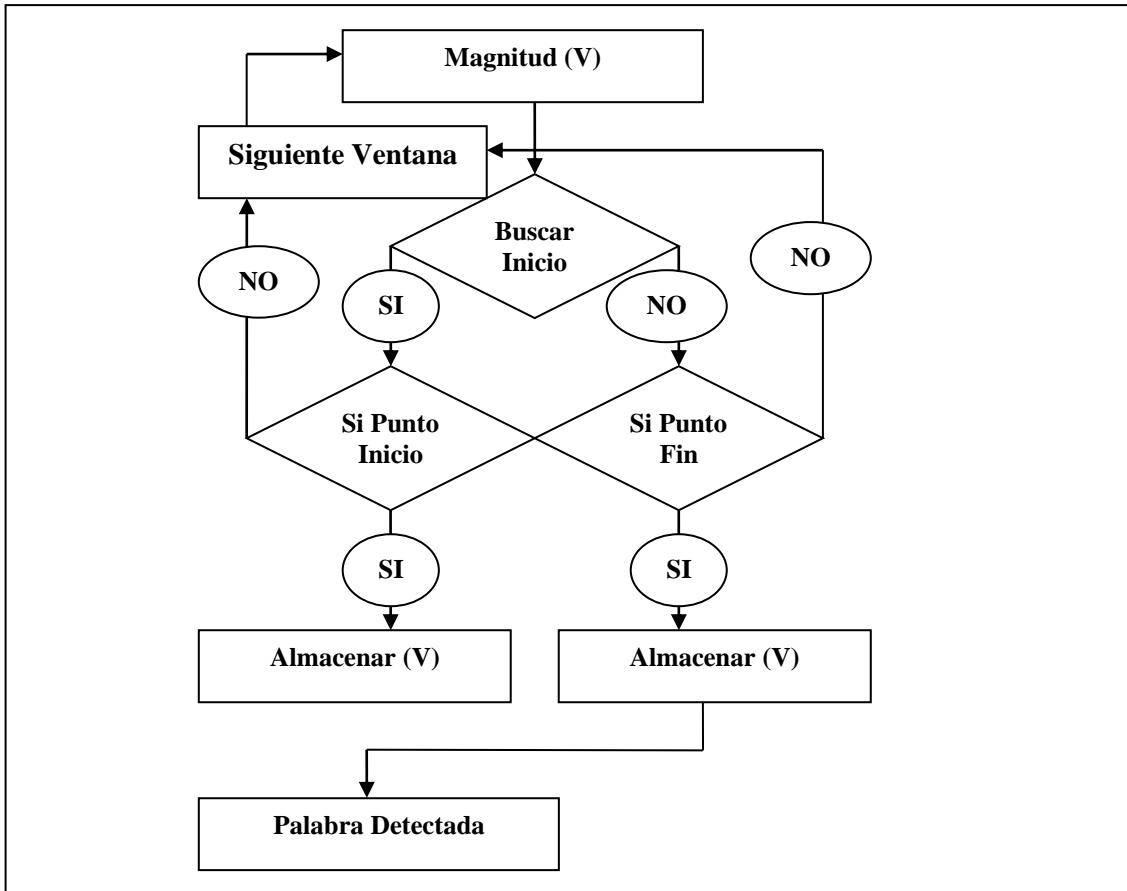
$$I1 = (0.03 * (IMX - IMN)) + IMN;$$

$$I2 = 4.0 * IMN;$$

$$ITL = \min(I1, I2)$$

$$ITU = 4.0 * ITL$$

Y el proceso es el mismo como en una grabación. Al final este proceso y tener esto almacenado, se rectifica con el proceso de grabación para limitar casi exacta de la palabra o comando pronunciado, es decir, se vuelve a pasar lo detectado por otra detección.

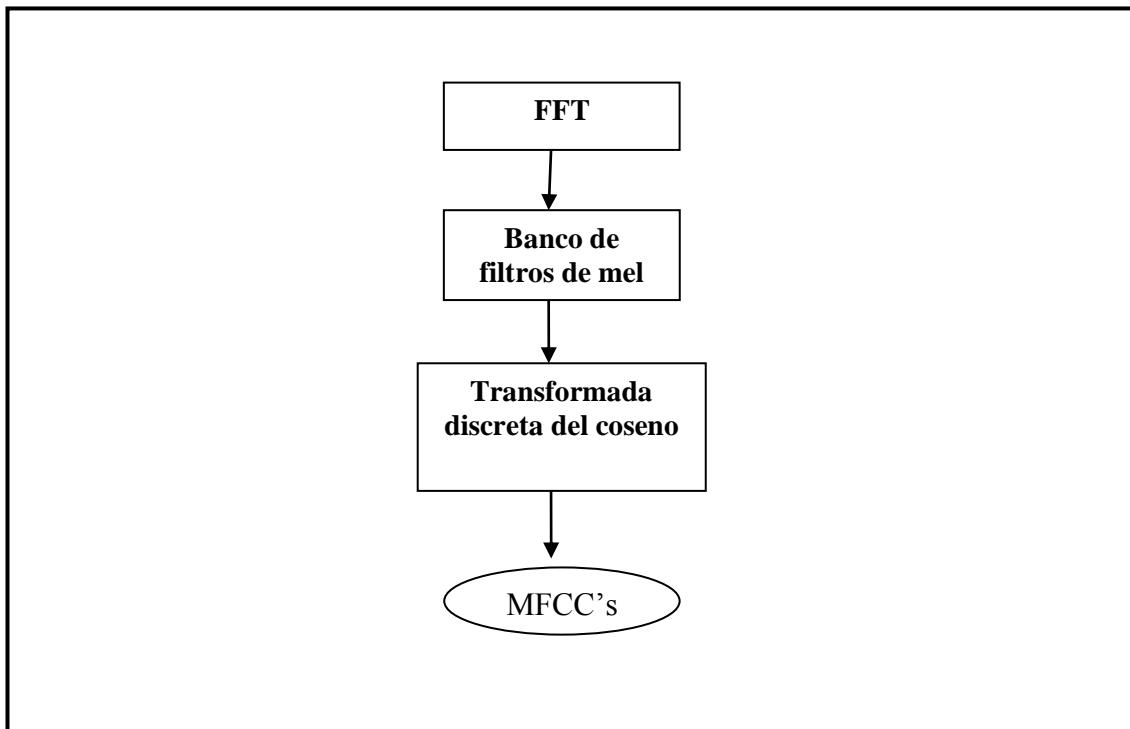


**Figura 46.** Diagrama que muestra como es el proceso de detección (en vivo).

Las clases que realiza este proceso son `Captura.DetectarBordes` y una clase adicional llamada `util.Calcular` para funciones auxiliares de cálculos, aunque se implementaron las dos versiones anteriores en diferentes métodos. La clase `Captura.DetectarBordes` es un hilo, para de esta forma poder procesar en paralelo la entrada de datos, y así ser ejecutado en tiempo real.

## 7.5. Características Espectrales

Para la obtención de los parámetros se optó por obtener los coeficientes cepstrales de Mel o su sigla en inglés MFCC, para esto se lleva a cabo el proceso que se muestra en la figura 47, pero antes de procesar cada ventana se le ha aplicado Hamming.



**Figura 47.** Diagrama de obtención de los MFCC.

### 7.5.1. Transformada rápida de Fourier (FFT)

La transformada de Fourier es utilizada para pasar del dominio del tiempo al dominio de la frecuencia, el algoritmo normalmente utilizado es la Transformada rápida de Fourier Radix-2, el cual es utilizado en esta aplicación.

La FFT es de gran importancia en una amplia variedad de aplicaciones, desde el tratamiento digital de señales y filtrado digital en general a la resolución de ecuaciones diferenciales parciales o los algoritmos de multiplicación rápida de grandes enteros.

$$G\left(\frac{h}{NT}\right) = \sum_{n=0}^{N-1} g(nT) e^{\frac{-j2\pi hn}{N}}, h = 0, 1, \dots, N-1$$

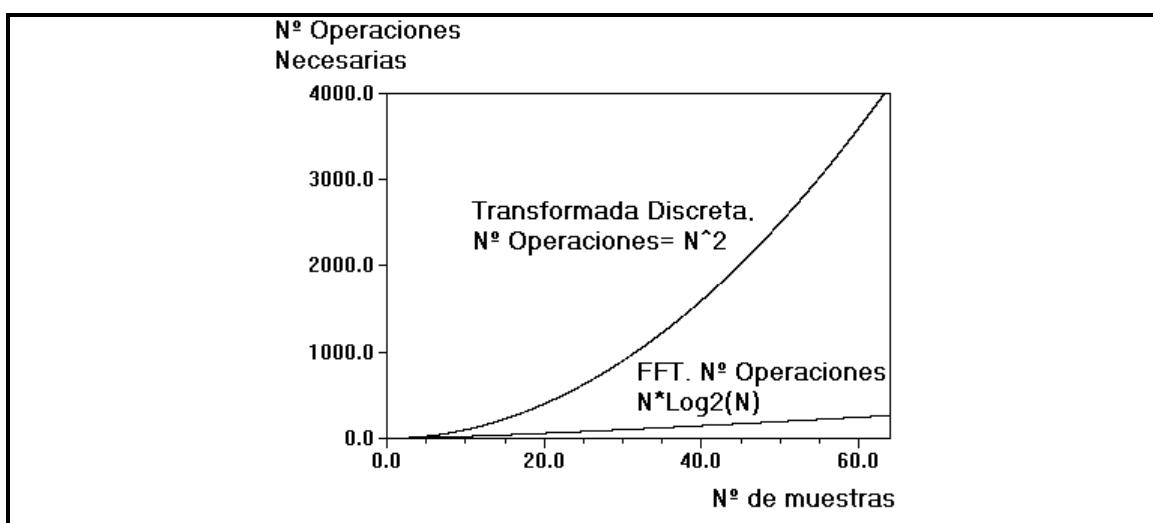
(3)

$$g(nT) = \frac{1}{N} \sum_{n=0}^{N-1} G\left(\frac{h}{NT}\right) e^{\frac{j2\pi hn}{N}}, h = 0, 1, \dots, N-1$$

La evaluación directa de esa fórmula requiere  $O(n^2)$  operaciones aritméticas. Mediante un algoritmo FFT se puede obtener el mismo resultado con sólo  $O(n \log n)$  operaciones.

En general, dichos algoritmos dependen de la factorización de  $n$  pero, al contrario de lo que frecuentemente se cree, existen FFT para cualquier  $n$ , incluso con  $n$  primo.

La diferencia de velocidad de cálculo entre la tradicional transformada discreta y la FFT aumenta según el número de muestras a analizar, según se puede apreciar en la gráfica, ya que mientras una aumenta el número de operaciones necesarias para la resolución de forma exponencial, la otra lo hace de forma prácticamente lineal.



**Figura 48.** Grafico de complejidad de FFT.

A continuación se muestran los nombres de los métodos principales:

Se comprueba que la entrada de la transformada sea potencia de dos.

```
boolean isPowerOfTwo(int x)
```

Método que invierte un valor numérico.

```
int reverseBits(int index, int NumBits)
```

Método que construye una tabla para agilizar la inversión de los índices.

```
void initFFT()
```

El método que calcula o computa la FFT es el siguiente:

```
public void FFT(int NumSamples, boolean InverseTransform, double []RealIn, double []ImagIn, double []RealOut, double []ImagOut)
```

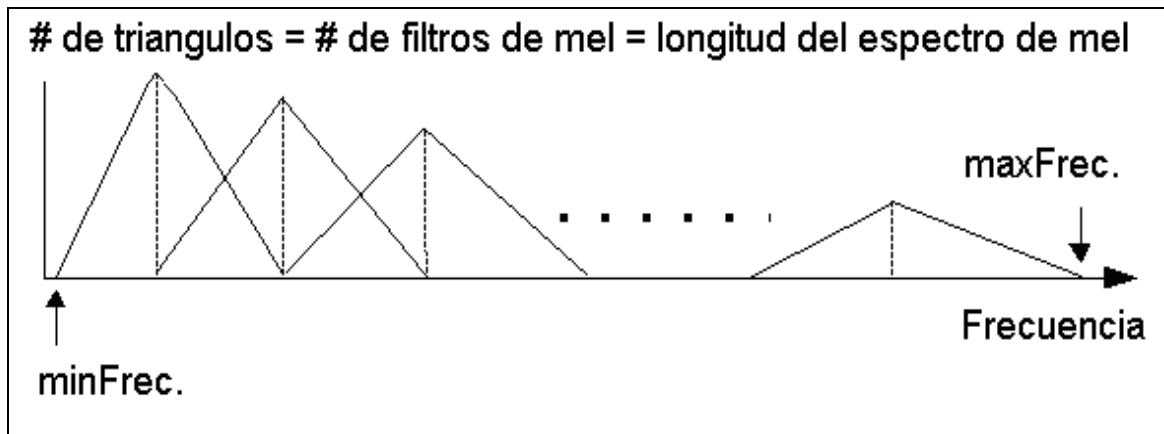
A éste método se le pasa el tamaño de la entrada, si es transformada inversa para normalizar los datos, los vectores de entrada y salida tanto para la parte real o como para la imaginaria.

```
public double[] process(double[] sn)
```

El método “process” recibe una ventana de datos, el cual hace un llamado al método “FFT” para realizar el cálculo, cuando ya se tiene realizado este se obtiene el módulo de la transformada, el módulo es una operación de obtener la mitad de parámetros de la ventana (Ejemplo: 128 da 64), calculando la raíz cuadrada de la suma al cuadrado de cada componente, tanto de la parte real como de la imaginaria.

### 7.5.2. Banco de filtros de Mel

Filtre un espectro (FFT) a través de un banco de filtros de mel. La salida es un vector de valores filtrados, típicamente llamado espectro de Mel, a cada resultado de filtro le corresponde el espectro de entrada a través de un filtro individual. Por consiguiente, la longitud del vector de salida es igual al número de filtros creados.



**Figura 49.** Diagrama de banco de filtros de Mel.

Estos son los métodos que realizan el banco de filtros de Mel.

En este método se inicializan la frecuencia mínima, máxima y el número de filtros.

```
public MelFrequencyFilterBank(float minFreq, float maxFreq, int numberFilters)
```

```
private double linToMelFreq(double inputFreq)
{
    return (2595.0 * (Math.log(1.0 + inputFreq / 700.0) / Math.log(10.0)));
}
```

El método “linToMelFreq” retorna la frecuencia en escala de Mel.

```
private void buildFilterbank(int numberFftPoints, int numberFilters,
                            double minFreq, double maxFreq)
```

Construye un banco de filtros de mel con los parámetros dados. Cantidad de puntos de la FFT, cantidad de filtros a realizar es 20, frecuencia mínima (15 Hz) y máxima (4 KHz).

### 7.5.3. Transformada Discreta del coseno

Es una variación de la transformada discreta de Fourier donde los datos se descomponen en sumas de cosenos (y no de senos y cosenos como en la de Fourier). Esta transformada se utiliza para obtener los coeficientes cepstrales de Mel o su sigla en inglés MFCC, este análisis provee una buena representación de la señal espectral dada en una ventana de audio.

El constructor de esta clase se inicializa proporcionándole la cantidad de filtros de mel a procesar (20 filtros de mel), y el cepstrum (13 de MFCC).

```
public DiscreteCosineTransform(int numberMelFilters, int cepstrumSize)
{
    this.numberMelFilters = numberMelFilters;
    this.cepstrumSize = cepstrumSize;
}
```

Este método construye una tabla para procesar la transformada discreta del coseno.

```
private void computeMelCosine()
// Proceso los bancos de filtros del coseno de mel.
```

El método “applyMelCosine” aplica la transformada discreta del coseno a los datos suministrados.

```
private double[] applyMelCosine(double[] melspectrum)

public double[] process(double[] input)
```

Y con este método se procesan los datos donde se crean los coeficientes espectrales de mel a partir de un espectro de entrada.

## 7.6 Red Neuronal

La red neuronal de alimentación hacia delante (feed forward) funciona de la siguiente manera: La capa de entrada debe conectarse con la capa oculta, esta a su vez se conecta entonces a la capa de salida.

Después de tener el tipo de la red neuronal, se debe seleccionar la cantidad de neuronas que tiene que llevar cada capa, para esta paso se decidió la cantidad de veinte comandos, por consiguiente la capa de salida tendría veinte neuronas, la capa de entrada 169 neuronas y la capa oculta se ajustó a prueba y error basándose en la fórmula la cual debe tener dos tercios de la capa de entrada (Heaton, 2005), si se realiza el cálculo da mas o menos 112 neuronas, el proceso de entrenamiento era un proceso muy demorado y por tal motivo se ajusto a 73 neuronas ocultas.

En este proyecto se hace uso de la red neuronal feedforward la cual esta implementada en la clase de Network.java. Hay varios métodos que constituyen la clase Network.

En el constructor de la red se tienen los siguientes parámetros:

```
public Network(int inputCount,      hiddenCount del int,      outputCount  
del int,      learnRate doble,  doble momentum)
```

"inputCount - El número de neuronas que están en la capa de la entrada.

"hiddenCount - El número de neuronas que están en la capa oculta.

"outputCount - El número de neuronas que están en la capa de salida.

"learnRate - La tasa de aprendizaje para el backpropagation que entrena el algoritmo.

"momentum - La velocidad adquirida para el backpropagation que entrena el algoritmo.

El constructor inicializa las estructuras de datos que implementan la red neuronal.

```
// Numero de neuronas de la red.  
neuronCount = inputCount + hiddenCount + outputCount;  
// Numero de pesos de la red.  
weightCount = (inputCount * hiddenCount)+(hiddenCount * outputCount);  
  
fire      = new double[neuronCount];// Salida de cada neurona.  
matrix    = new double[weightCount];// Pesos de la RNA.  
//Cambios que pueden ser aplicados a los pesos.  
matrixDelta = new double[weightCount];  
thresholds = new double[neuronCount];// Umbrales de la RNA.  
errorDelta = new double[neuronCount];// Cambios en el error.  
error      = new double[neuronCount];// Ultimo error calculado  
// Acumuladores de los Delta de los umbrales para entrenamiento.  
accThresholdDelta = new double[neuronCount];  
// Acumuladores de los Delta de los pesos para entrenamiento.  
accMatrixDelta    = new double[weightCount];  
// Deltas de umbralizacion.  
thresholdDelta    = new double[neuronCount];
```

La tasa de velocidad y aprendizaje adquirida es para el algoritmo del backpropagation, que efectúa el proceso de aprendizaje de la red neuronal.

En general ambos valores deben ser mayores de cero, pero también menores que 1, el valor utilizado en el proyecto o es de 0.1 para ambos valores. Una vez la red neuronal ha sido instanciada, usando el constructor, está básicamente listo para utilizarse. A este punto la red neuronal tiene una matriz de peso aleatoria y deberán ser entrenadas o inicializar los pesos antes de realizar cualquier proceso, el paso siguiente es cargar de un archivo de configuración los pesos de la red neuronal para poder de esta manera reconocer los comandos de los usuarios.

El método “computeOutputs” es el encargado de ejecutar o procesar el patrón que representa un comando de voz.

```
p\xf3blico doble [] computeOutputs(double input [])
```

Como se puede observar, es un vector de tipo “double” que se le pasa al método de “computeOutputs”. El tamaño de esta vector debe corresponder al número de entrada de la red neuronal. El vector que retorna muestra que neuronas se han activado con el patrón de entrada. Se obtiene un proceso aparte donde se selecciona la neurona o comando ganador con el valor mas alto que haya arrojado una neurona de la capa de salida.

Por defecto esta implementación de red neuronal usa el método de activación sigmoidal. Este método de activación se muestra a continuación:

```
public double threshold(double sum) {  
    return 1.0 / (1 + Math.exp(-1.0 * sum));  
}
```

Para el proceso de entrenamiento no existe un método que se llame backpropagation, sino, que está distribuido en varios métodos para realizar un trabajo mas eficiente.

El siguiente método tiene dos tareas principales, la primera calcular el error, y la segunda tarea es calcular los deltas del backpropagation. Ahora examinaremos cómo el

error de salida, para un entrenamiento es computado. El código de fuente para el método "calcError" se muestra a continuación:

```

public void calcError(double ideal[]) {
    int i, j;
    final int hiddenIndex = inputCount;
    final int outputIndex = inputCount + hiddenCount;

    // clear hidden layer errors
    for (i = inputCount; i < neuronCount; i++) {
        error[i] = 0;
    }

    // layer errors and deltas for output layer
    for (i = outputIndex; i < neuronCount; i++) {
        error[i] = ideal[i - outputIndex] - fire[i];
        globalError += error[i] * error[i];
        errorDelta[i] = error[i] * fire[i] * (1 - fire[i]);
    }

    // hidden layer errors
    int winx = inputCount * hiddenCount;

    for (i = outputIndex; i < neuronCount; i++) {
        for (j = hiddenIndex; j < outputIndex; j++) {
            accMatrixDelta[winx] += errorDelta[i] * fire[j];
            error[j] += matrix[winx] * errorDelta[i];
            winx++;
        }
        accThresholdDelta[i] += errorDelta[i];
    }

    // hidden layer deltas
    for (i = hiddenIndex; i < outputIndex; i++) {
        errorDelta[i] = error[i] * fire[i] * (1 - fire[i]);
    }

    // input layer errors
    winx = 0; // offset into weight array
    for (i = hiddenIndex; i < outputIndex; i++) {
        for (j = 0; j < hiddenIndex; j++) {
            accMatrixDelta[winx] += errorDelta[i] * fire[j];
            error[j] += matrix[winx] * errorDelta[i];
            winx++;
        }
        accThresholdDelta[i] += errorDelta[i];
    }
}

```

Como se puede observar del código anterior, el error es calculado en el primer bloque de código en el método. El cálculo del error empieza con las líneas siguientes.

```

// layer errors and deltas for output layer
for (i = outputIndex; i < neuronCount; i++) {

```

Primero se realiza un ciclo en cada neurona de la capa de salida. Cada elemento de salida de la red neuronal se compara contra el rendimiento ideal. La diferencia entre estos dos valores es calculada. Este valor se eleva al cuadrado y se guarda. Este valor se agrega al error de salida, guardando el error global.

```
error[i] = ideal[i - outputIndex] - fire[i];
globalError += error[i] * error[i];
errorDelta[i] = error[i] * fire[i] * (1 - fire[i]);
}
```

Al finalizar el proceso, se agrega ahora el error a los elementos de entrenamiento y al acumulador del error global. Este proceso continúa para cada uno de los elementos de entrenamiento. Una vez cada elemento en juego ha sido calculado entonces está listo para calcular el error cuadrático medio.

Se anota que el método anterior calcula el error de cada una de las neuronas en las otras capas. Esto se usará cuando el método de entrenamiento es llamado para ajustar los pesos de la red neuronal.

El error cuadrático medio permite que la red neuronal funcione si el entrenamiento ha tenido lugar. El error puede calcularse en cualquier momento después de que el método "calcErrors" sea llamado. Este cálculo se hace por el método "getError." El método "getError" se muestra aquí.

```
public double getError(int len) {
    double err = Math.sqrt(globalError / (len * outputCount));
    globalError = 0;
    return err;
}
```

Como se puede observar, se le pasa como parámetro la longitud del sistema de entrenamiento. Esto es necesario porque el cuadrado medio es un promedio. Para tomar el error medio a través de todos los elementos determinados del entrenamiento, se debe saber el tamaño del sistema de entrenamiento. El error es entonces calculado dividiendo

el error global por el producto de la longitud determinada de entrenamiento y el número de las neuronas de salida. La raíz cuadrada de este cociente produce el error cuadrático medio. Finalmente, después de que se haya calculado el error, el globalError se fija de nuevo a cero. Se hace esto de modo que pueda comenzar a acumular para un nuevo error.

Para complementar el backpropagation se procede ha realizar el entrenamiento, para esto se llama al método “learn”. El cual modifica los pesos y umbrales de la red neuronal para realizar el proceso de aprendizaje.

```
private void learn()
{
    int i;

    // process the matrix
    for (i = 0; i < matrix.length; i++) {
        matrixDelta[i] = (learnRate * accMatrixDelta[i]) +
(momentum * matrixDelta[i]);
        matrix[i] += matrixDelta[i];
        accMatrixDelta[i] = 0;
    }

    // process the thresholds
    for (i = inputCount; i < neuronCount; i++) {
        thresholdDelta[i] = learnRate * accThresholdDelta[i] +
(momentum * thresholdDelta[i]);
        thresholds[i] += thresholdDelta[i];
        accThresholdDelta[i] = 0;
    }
}
```

El proceso que se lleva a cabo para realizar un solo ciclo de entrenamiento se visualiza en el siguiente método:

```
public void training(double input[][],double output[][])
{
    if(input.length!=output.length)
        throw new java.lang.IndexOutOfBoundsException("Network:
Length input != Length output");

    for (int j=0;j<input.length;j++)
    {
        computeOutputs(input[j]);
        calcError(output[j]);
        learn();
    }
}
```

Primero se prueba que la longitud sea acorde con la cantidad de neuronas de entrada, el segundo paso que se realiza es calcular la salida y después calcular el error, cuando ya se tiene el error se llama al método “learn”.

La clase Network es la implementación de la red neuronal, pero la clase RedNeuronal es la que la usa. Esta clase (RedNeuronal) es la delegada de construir una instancia de Network y configurarla para su uso, también es la encargada de que se dé el proceso de producción y entrenamiento.

Para la realización del entrenamiento un solo ciclo no basta y el límite o porcentaje de error prefijado para la red es de un máximo de 0.5% y un mínimo de 0.35% este resultado se logra realizando 10.000 ciclos o llamados al método “trainning” con un promedio que se mantiene en 0.45%, estos valores se fijaron a ensayo y error.

Para realizar la comprobación de que la red neuronal funciona adecuadamente se le pasaba las veinte grabaciones y se revisaba que solo se activara la neurona correspondiente en la capa de salida, debido que la función de activación es una sigmoidal, la salida de una neurona se encuentra en el intervalo [0,1], por lo que se comprobó que la salida siempre fuera uno o su aproximación y las otras 19 el valor de cero, pero se encontró que la salida de unas neuronas no eran uno, sino 0.96, 0.98, superior a 0.95 que por aproximación es uno y las otras neuronas 19 con valores de 0.04, 0.01, por los resultados en esa prueba se muestra que el error se mantiene al 0.45%.

## 7.7. Ejecución del comando

La clase emulador.Dispositivo es la encargada de ejecutar el comando de voz, en el constructor de esta clase se inicializa una instancia de java.awt.Robot la cual es la encargada de ejecutar o simular un evento del teclado o ratón, que para IcVoz solo es el teclado, también se instancia la clase java.awt.event.KeyEvent de forma estática para obtener los códigos correspondientes a las teclas del teclado.

```

public Dispositivo(String perfil)
{
    this.perfil=perfil;

    try{
        //Inicializa el archivo de configuración
        this.archivo = new ArchivoCONFIG(this.perfil);
        //Se obtienen los comandos
        this.datos = archivo.getComandos();
        //Se inicializa la clase que contiene los códigos del teclado
        key = Class.forName("java.awt.event.KeyEvent");
        //Se inicializa la clase que emulará el teclado
        device = new java.awt.Robot();

    }catch(Exception e)
    {
        System.out.println("Error: Dispositivo:Constructor");
    }
}

```

Cuando se le pasa el código a esta clase se procede a buscar en el archivo de configuración y luego a relacionar este código al nombre de las teclas en caracteres, después se procede a realizar la conversión de caracteres al código que contiene java para ser ejecutado en la clase java.awt.Robot.

```

/*
 * Para ejecutar una tecla se realiza la operación de:
 *     keyPress          (Presion)
 *     keyRelease        (Suelta)
 */

public void eventoTeclado(String id)
{
    //Se busca el comando
    Element cmd = getComando(id);

    //Se realiza la codición de que exista.
    if(cmd==null)
    {
        System.out.println("Error: Comando no Encontrado");
        return;
    }

    //Se obtiene el modificador "Ctrl", "Alt"...
    String modificador = cmd.getAttributeValue("modificador");
    //Se obtiene la tecla "A", "Enter"...
    String tecla = cmd.getAttributeValue("tecla");
    //Se realiza la validación de que se haya inicializado la clase
    //    - java.awt.Robot
    //    - java.awt.event.KeyEvent
    if(this.key==null || this.device == null)
    {
        System.out.println("Error:Controlador      de      teclado      no
encontrado");
        return;
    }
}

```

```

// Se valida de que la tecla exista
if(tecla==null&&modificador==null)
    return;

if(tecla==null)
    return;

try{

    //Se procede a convertir el modificador en entero
    // y ejecutar la tecla
    if(modificador!=null)
        device.keyPress(key.getField(modificador).getInt(key));

    //Se procede a convertir la tecla en entero
    // y ejecutar la tecla
    if(tecla!=null)
        device.keyPress(key.getField(tecla).getInt(key));

    //Se procede a convertir el modificador en entero
    // y ejecutar la tecla
    if(modificador!=null)
        device.keyRelease(key.getField(modificador).getInt(key));

    //Se procede a convertir la tecla en entero
    // y ejecutar la tecla
    if(tecla!=null)

        device.keyRelease(key.getField(tecla).getInt(key));

}catch(Exception e)
{
    System.out.println("Error:Dispositivo:eventoTeclado \n"+e);
}

//Se obtiene el nombre del comando.
String nombre = cmd.getAttributeValue("nombre");
//Se notifica al usuario el comando ejecutado
main.Main.mensajeInfo("( ( Voz ) )","Se ha ejecutado: "+nombre);

}

```

Para pasar los caracteres al código que se va a ejecutar, esta se realiza de forma estática, utilizando el mismo nombre que tiene la variables en Java de la clase "java.awt.event.KeyEvent", ya que Java cuenta con unos métodos que permiten pasar o buscar un atributo de una clase basado en su nombre, si el atributo no se encuentra retorna nulo (null).

## 7.8. Configuración

La configuración es una interfaz gráfica en la que el usuario puede modificar los patrones de voz. Ver figura 50. El usuario cuenta con una interfaz que le permite modificar, crear y borrar perfiles a excepción de un perfil llamado “default” y cualquier otro perfil que se encuentre como predeterminado.



Figura 50. Configuración de IcVoz (Perfil).

En esta pantalla como se observa se tiene un botón de configurar al frente de cada nombre de perfil, este con el motivo de que el usuario modifique el perfil que va a usar, al presionar este botón aparece en la misma interfaz una lista de lo veinte comandos que tiene IcVoz. En la figura siguiente se observa la configuración o el perfil que se está modificando, también aparece la opción al frente de cada perfil de edición de cada comando.

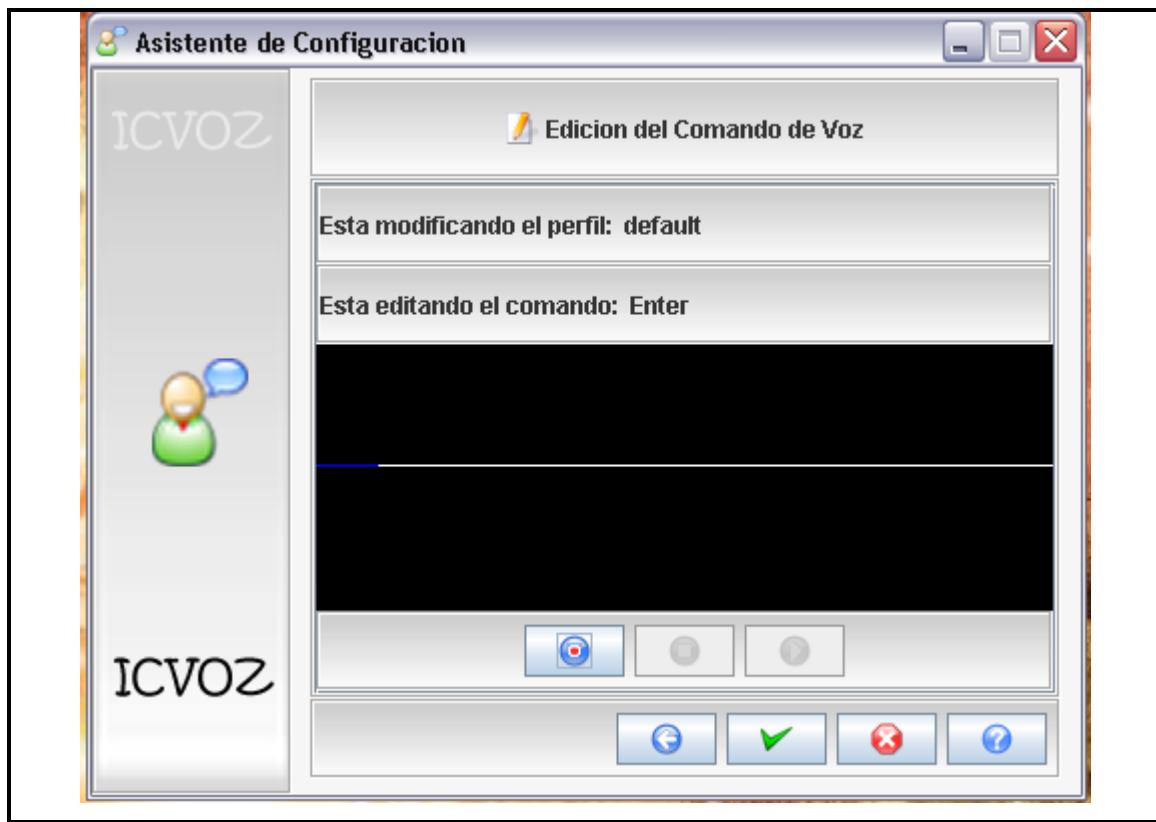


**Figura 51.** Configuración de IcVoz (Comando).

En la siguiente figura se muestra donde el usuario realiza la grabación para realizar el proceso de entrenamiento de la red neuronal, además el usuario puede modificar los veinte comandos al mismo tiempo o solo los que el considere deficientes, sin la necesidad de estar entrenando o aceptando los cambios para realizar el entrenamiento.

En la pantalla donde se muestra la edición se observa que perfil y comando esta editando, al usuario cuenta con tres botones para el manejo del audio, el primero es de grabación, el segundo de detener la grabación y el tercero de reproducir la grabación cuando esta hecha.

En todas las figuras mostradas de este asistente se observa un botón de cancelar, el usuario puede salir de la interfaz cuando este lo decida.



**Figura 52.** Configuración de IcVoz (Edición).

Cuando el usuario haya modificado los comandos, le aparecerá una ventana en la cual se le informa el proceso de configuración (entrenamiento de la red neuronal).



**Figura 53.** Configuración de IcVoz (Espera).

Para llevar a cabo el proceso de entrenamiento se tiene una clase llamada RedNeuronal.java que es la que instancia Network.java para el manejo de la red. En las siguientes líneas de código se muestra donde se realiza el entrenamiento.

```

public void entrenamiento(JProgressBar progreso)
{
    System.out.println("Entrenando");
    ArchivoRNA archivoRNA;
    ArchivoPSS archivoPSS;
    double entrada[][][];
    double salida[][][];
    try{
        archivoRNA = new ArchivoRNA(this.perfil);
        // Se cargan la entrada y salida de la RNA.
        entrada = archivoRNA.getEntrada();
        salida = archivoRNA.getSalida();
        // Se borran los pesos actuales de la red.
        network.reset();
        System.out.println("Entrada : "+entrada.length);
        System.out.println("Salida : "+salida.length);
        // Se pone como limite de entrenamiento una cantidad
        // fija de ciclos.
        int max = 10000;
        int update=0;
        double error;
        for (int i=0;i<max;i++)
        {
            network.training(entrada,salida);
            error = network.getError(entrada.length);

            progreso.setValue(i/100);
            update++;
            if (update==1000)
            {
                // Se actualiza la interfaz grafica cada mil ciclos.
                progreso.setValue(i/100);
                System.out.println("CyclesLeft:"+ (max-i)+",Error:"+error);
                update=0;
            }
        }
        // Se actualiza la interfaz grafica.
        progreso.setValue(99);
        // Se almacena los nuevos pesos y umbrales de la red.
        archivoPSS = new ArchivoPSS(this.perfil);
        archivoPSS.setPesos(network.getMatrix());
        archivoPSS.setUmbral(network.getThresholds());
        archivoPSS.guardar();
        progreso.setValue(100);
    }catch(Exception e){
        System.out.println("Error: RedNeuronal:entrenamiento");
        e.printStackTrace();
    }
    System.out.println("Entrenamiento realizado");
}

```

Con este método de la clase “RedNeuronal” se realiza el entrenamiento de la red neuronal la cual esta implementada en la clase “Network” como ya se menciono.

## **8. PRUEBAS**

El sistema de reconocimiento de comandos de voz en español, es una aplicación monousuario el cual reconoce veinte (20) comandos de voz. Se realizaron dos pruebas, la primera prueba nos indica el porcentaje de reconocimiento de la aplicación, la segunda prueba se realiza para mostrarle al usuario su funcionamiento y determinar la aceptación del usuario. Para ver resultados de esta prueba ver Anexo F.

La primera prueba se realiza siguiendo los siguientes pasos:

- Se procede a realizar la grabación y obtención de patrones para el entrenamiento de la red neuronal (Los veinte comandos).
- Se procede a llenar una tabla (Ver Tabla 9) con el correo, edad y sexo de la persona, esta tabla se utiliza para calcular el porcentaje de reconocimiento global y local. Cuando se habla de global es el promedio de las sumas del reconocimiento local; cuando nos referimos a lo local hablamos de la cantidad de veces (de diez) que el programa reconoció mal o bien un comando.
- Se procede con el usuario a calcular el porcentaje de reconocimiento de cada comando.
- Si al finalizar el usuario obtiene un porcentaje global por debajo del 70% se procede a realizar de nuevo el entrenamiento modificando solo los comandos que de forma local están por debajo del 60%.
- Se procede a calcular de nuevo el porcentaje global, pero esta vez modificando los comandos que quedaron por debajo del 60% la primera vez, este proceso y el punto anterior se realiza un máximo de tres veces.

En las tablas 9 y 10 tienen los resultados obtenidos por los dos desarrolladores siguiendo los pasos descritos anteriormente, los resultados obtenidos por la pruebas de los desarrolladores fueron la medida para determinar si el porcentaje de reconocimiento era aceptable.

Estadística						
Interpretador de Comandos de Voz						
Numero	Comando	Tecla (s)	Bien	Mal	Total	Bien %
0	Enter	Enter.	6	4	10	60
1	Arriba	FlechaArriba.	8	2	10	80
2	Abajo	FlechaAbajo.	8	2	10	80
3	Izquierda	Flecha Izq.	6	4	10	60
4	Derecha	Flecha Der.	9	1	10	90
5	Copiar.	Ctrl.+ C	9	1	10	90
6	Cortar.	Ctrl.+ X	8	2	10	80
7	Pegar.	Ctrl.+ V	7	3	10	70
8	Deshacer.	Ctrl.+ Z	8	2	10	80
9	Menú	Alt.	8	2	10	80
10	Salir	Esc.	7	3	10	70
11	Inicio	Win.	9	1	10	90
12	Alternar	Alt.+ Tab.	8	2	10	80
13	Explorador	Win + E	9	1	10	90
14	Cerrar	Alt.+ F4	6	4	10	60
15	Guardar	Ctrl.+ G	6	4	10	60
16	Borrar	Supr.	7	3	10	70
17	Seleccionar	Ctrl.+ A	7	3	10	70
18	Escritorio	Win+ D	9	1	10	90
19	Ayuda	F1	6	4	10	60
<hr/>						
Total			151	49	200	76
Porcentaje de Reconocimiento						75,50
Porcentaje de Mal Reconocimiento						24,50
Total						100,00

**Tabla 9:** Prueba realizada al interpretador.  
(Usuario de la prueba: Andrés Vélez Pérez)

Estadística						
Interpretador de Comandos de Voz						
Numero	Comando	Tecla (s)	Bien	Mal	Total	Bien %
0	Enter	Enter.	9	1	10	90
1	Arriba	FlechaArriba.	8	2	10	80
2	Abajo	FlechaAbajo.	8	2	10	80
3	Izquierda	Flecha Izq.	8	2	10	80
4	Derecha	Flecha Der.	8	2	10	80
5	Copiar.	Ctrl.+ C	9	1	10	90
6	Cortar.	Ctrl.+ X	8	2	10	80
7	Pegar.	Ctrl.+ V	4	6	10	40
8	Deshacer.	Ctrl.+ Z	7	3	10	70
9	Menú	Alt.	9	1	10	90
10	Salir	Esc.	8	2	10	80
11	Inicio	Win.	6	4	10	60
12	Alternar	Alt.+ Tab.	9	1	10	90
13	Explorador	Win + E	9	1	10	90
14	Cerrar	Alt.+ F4	7	3	10	70
15	Guardar	Ctrl.+ G	9	1	10	90
16	Borrar	Supr.	1	9	10	10
17	Seleccionar	Ctrl.+ A	7	3	10	70
18	Escritorio	Win+ D	8	2	10	80
19	Ayuda	F1	8	2	10	80
Total			75	50	200	75
Porcentaje de Reconocimiento						
75,00						
Porcentaje de Mal Reconocimiento						
25,0						
Total						
100,00						

**Tabla 10:** Prueba realizada al interpretador.  
(Usuario de la prueba: Christian Martan)

Debido a que con las dos pruebas anteriores no se pueden obtener conclusiones que permitan realizar mejoras sobre el proyecto, se decidió realizar la prueba a diez personas. Los resultados obtenidos se observan en la tabla 11. Para ver resultados individuales ver Anexo F.

<b>Estadística General</b>				
<b>Usuario</b>	<b>Sexo</b>	<b>Edad</b>	<b>Bien (%)</b>	<b>Mal (%)</b>
limp	M	21	71,50	28,50
lufe	M	21	88,50	11,50
vicm	M	22	76,50	23,50
cahe	M	22	74,50	25,50
john	M	23	83,50	16,50
pave	M	22	83,50	16,50
alex	M	24	77,50	22,50
dein	F	22	78,50	21,50
yuri	F	23	83,50	16,50
sami	F	29	88,00	12,00
Total:			80,55	19,45

**Tabla 11:** Estadística General de la prueba aplicada a 10 Usuarios.

La tasa promedio de reconocimiento es del 80.55%, se puede observar que el máximo porcentaje de reconocimiento es del 80.5 % y el mínimo de 71.5%, estos resultados se deben a la buena o mala pronunciación de los usuarios, esta conclusión se obtuvo porque algunos usuarios no pronunciaban bien los comandos.

Si una persona utiliza la aplicación y además tiene buena pronunciación de las palabras no tendrá dificultad en utilizar una herramienta como esta (IcVoz), pero sino tiene buena pronunciación se le dificultará utilizarla.

El porcentaje de reconocimiento para la aplicación se deja en 75% por los resultados dados en las pruebas hechas por los propios desarrolladores.

## **9. CONCLUSIONES**

En el diseño y construcción de una red neuronal se tienen en cuenta dos factores fundamentales: la cantidad de datos a clasificar y la capacidad de procesamiento en el momento de realizar el entrenamiento, por tal motivo un patrón de entrada grande y una mayor cantidad de comandos afecta a la red neuronal negativamente en el proceso de entrenamiento ya que requiere mucho tiempo de computo y calculo de los pesos de la red neuronal.

En la realización de las pruebas se encontró que a los usuarios les afectaba el ruido ambiental, a veces se generaba un comando producido por el ruido, para un funcionamiento correcto de la aplicación se requiere que el ruido ambiental sea mínimo.

Este proyecto facilita las actividades realizadas por las personas al momento de interactuar con el computador, también se enmarca el hecho de que permite navegar por los menús de la computadora sin utilizar el teclado o el Mouse. Además este trabajo puede ser adaptado para que a través del computador se pueda manejar otros equipos (como un sistema de control remoto para televisores, abrir/cerrar puertas, etc.). Estos sistemas pueden mejorar la calidad de vida de muchas personas al momento de acceder a ordenadores, que en la vida de hoy son necesarios.

## **10. PROYECCIONES**

Un sistema de reconocimiento de habla actualmente no tiene una solución específica, hasta ahora ha ido evolucionando poco a poco y cada vez se plantean mejoras. Para este sistema en particular un sistema de reconocimiento automático de habla que reconoce comandos de voz se plantean las siguientes pautas para su mejoramiento:

- Se debe mejorar la eficiencia del detector de voz (detectar la voz en el medio).
- El sistema debe trabajar con múltiples usuarios (reconocimiento de voz independiente de locutor).
- Mejorar un 10% el reconocimiento de la voz, mejorando los algoritmos de extracción de características de la voz.
- Poseer una mayor cantidad de comandos de voz.
- Permitir al usuario agregar nuevos comandos.

## **11. REFERENCIAS**

ÁLVAREZ, L.E. Análisis Fonético Universidad del Quindío, Año 1991, 352 Páginas.

BERNAL B. JESÚS, BOBADILLA S. JESÚS, GÓMEZ V. PEDRO Reconocimiento de Voz y Fonética Acústica. ALFAOMEGA Grupo Editor año 2000, 332 Páginas.

CASACUBERTA F. Y VIDAL E., Reconocimiento Automático del Habla, Maracombo S.A. Año 1987, 205 Páginas.

CARRASCO O. JESÚS A., Reconocimiento de Patrones [Consultado Mayo 3 de 2006] Disponible en: <http://ccc.inaoep.mx/~ariel/recpat.pdf>

HEATON JEFF, Introduction to Neural Networks with Java, ISBN: 097732060X, ISBN: 0977320634, Year 2005. 380 Pages.

KILLER MIRJAN, STÜKER SEBASTIAN, SCHULTZ TANJA, Grapheme based Speech Recognition, [Consultado Julio 25 de 2006] Disponible en: <http://www.cs.cmu.edu/~tanja/Papers/Euro03-KillerSchultz.pdf>

MARTÍNEZ EVELIO. Teorema de Nyquist [Actualizado 1997-2003 consultado Mayo 3 de 2006] Disponible en: <http://www.eveliux.com/fundatel/nyquist.html>

MANDL MATTHEW, Principios de las comunicaciones electrónicas, Maracombo S.A, capítulo 1, 1982 Ed.

MARIAN PRUTSCHER, Transformadas de Fourier [Actualizado en Octubre 27 de 1998 Consultado en Abril 19 de 2006] Disponible en: [http://mwt.e-technik.uni-ulm.de/lehre/basic\\_mathematics/fourier\\_es/node1.php3](http://mwt.e-technik.uni-ulm.de/lehre/basic_mathematics/fourier_es/node1.php3)

MEJÍA CARLOS, VANEGAS H. MAURICIO, Modelo de un Reconocedor de Palabras Aisladas Independiente de Locutor. Universidad Pontificia Bolivariana, circular 1 No. 70 – 01. [Consultado Octubre 10 de 2006]. No Disponible en Internet.

MERLO, G; FERNÁNDEZ, V.; CARAM, F.; PRIEGUE, R. Y GARCÍA MARTÍNEZ, R. Reconocimiento De La Voz Mediante Una Red Neuronal De Kohonen [Consultado Marzo 22 de 2006] Disponible en:  
<http://www.fi.uba.ar/laboratorios/lsi/c-reconocimientodevozconkohonen-acic97.pdf>

NILSSON MAGNUS, Speaker Verification in Java, School of Microelectronic Engineering, Griffith University, pp. 1-35, October 18, 2001.

RABINER L. R., SAMBUR M. R., An Algorithm for Determining the Endpoints of Isolated Utterances, Bell System Tech. Jour., Vol. 54, No. 2, pp. 297-315, February 1975.

TEBELSKIS JOE, Speech Recognition using Neural Networks, School of Computer Science, Pittsburgh, 1995. [Consultado Junio 13 de 2006] Disponible en:  
<http://citeseer.ist.psu.edu/article/tebelskis95speech.html>

TOMASI WAYNE 2 ed. Sistemas de comunicaciones electrónicas, Pearson educación, capítulo 15, 1996.

**ANEXO A**  
**MÉTODOS ABREVIADOS DE TECLADO DE WINDOWS**

**Métodos abreviados de teclado generales**

<b>Presione</b>	<b>Para</b>
CTRL+C	Copiar.
CTRL+X	Cortar.
CTRL+V	Pegar.
CTRL+Z	Deshacer.
SUPRIMIR	Eliminar.
MAYÚS+SUPR	Eliminar permanentemente el elemento seleccionado, sin colocarlo en la Papelera de reciclaje.
CTRL al arrastrar un elemento	Copiar el elemento seleccionado.
CTRL+MAYÚS al arrastrar un elemento	Crear un acceso directo al elemento seleccionado.
F2	Cambiar el nombre del elemento seleccionado.
CTRL+FLECHA DERECHA	Mover el punto de inserción al principio de la palabra siguiente.
CTRL+FLECHA IZQUIERDA	Mover el punto de inserción al principio de la palabra anterior.
CTRL+FLECHA ABAJO	Mover el punto de inserción al principio del párrafo siguiente.
CTRL+FLECHA ARRIBA	Mover el punto de inserción al principio del párrafo anterior.
CTRL+MAYÚS con cualquier tecla de dirección	Resaltar un bloque de texto.
MAYÚS con cualquier tecla de dirección	Seleccionar varios elementos en una ventana o en el escritorio o seleccionar texto en un documento.
CTRL+A	Seleccionar todo.
F3	Buscar un archivo o una carpeta.

ALT+ENTRAR	Ver las propiedades del elemento seleccionado.
ALT+F4	Cerrar el elemento activo o salir del programa activo.
ALT+ENTRAR	Mostrar las propiedades del objeto seleccionado.
ALT+BARRA ESPACIADORA	Abrir el menú contextual de la ventana activa.
CTRL+F4	Cerrar el documento activo en los programas que permitan tener abiertos varios documentos simultáneamente.
ALT+TAB	Cambiar de un elemento abierto a otro.
ALT+ESC	Desplazarse por los programas en el orden en que se abrieron.
F6	Desplazarse por los elementos de pantalla de una ventana o del escritorio.
F4	Mostrar la lista de la barra Dirección en Mi PC o en el Explorador de Windows.
MAYÚS+F10	Mostrar el menú contextual del elemento seleccionado.
ALT+BARRA ESPACIADORA	Mostrar el menú de sistema de la ventana activa.
CTRL+ESC	Mostrar el menú <b>Inicio</b> .
ALT+letra subrayada de un nombre de menú	Mostrar el menú correspondiente.
Letra subrayada en un nombre de comando de un menú abierto	Ejecutar el comando correspondiente.
F10	Activar la barra de menús en el programa activo.
FLECHA DERECHA	Abrir el siguiente menú de la derecha o abrir un submenú.
FLECHA IZQUIERDA	Abrir el siguiente menú de la izquierda o cerrar un submenú.
F5	Actualizar la ventana activa.
RETROCESO	Ver la carpeta de un nivel superior en Mi PC o el Explorador de Windows.
ESC	Cancelar la tarea actual.

MAYÚS al insertar un CD en la unidad de CD-ROM	Evitar que el CD se reproduzca automáticamente.
--	---

### Métodos abreviados de teclado en los cuadros de diálogo

Presione	Para
CTRL+TAB	Avanzar a través de las fichas.
CTRL+MAYÚS+TAB	Retroceder a través de las fichas.
TAB	Avanzar a través de las opciones.
MAYÚS+TAB	Retroceder a través de las opciones.
ALT+letra subrayada	Ejecutar el comando correspondiente o seleccionar la opción correspondiente.
ENTRAR	Ejecutar el comando de la opción o el botón activo.
BARRA	Activar o desactivar la casilla de verificación si la opción activa es una casilla de verificación.
ESPACIADORA	
Teclas de dirección	Elegir un botón si la opción activa es un grupo de botones de opción.
F1	Mostrar la Ayuda.
F4	Mostrar los elementos de la lista activa.
RETROCESO	Abrir una carpeta de un nivel superior si hay una carpeta seleccionada en un cuadro de diálogo <b>Guardar como</b> o <b>Abrir</b> .

### Métodos abreviados de Natural Keyboard

Puede utilizar los siguientes métodos abreviados de teclado con Microsoft Natural Keyboard o cualquier otro teclado compatible que incluya la tecla del logotipo de

Microsoft ( ) y la tecla Aplicación ( ).

Presione	Para
	Mostrar u ocultar el menú <b>Inicio</b> .
+ INTER	Mostrar el cuadro de diálogo <b>Propiedades del sistema</b> .
+ D	Mostrar el escritorio.

+ M	Minimizar todas las ventanas.
+ Mayús + M	Restaurar todas las ventanas minimizadas.
+ E	Abrir Mi PC.
+ F	Buscar un archivo o una carpeta.
CTRL+  + F	Buscar equipos.
+ F1	Mostrar la Ayuda de Windows.
+ L	Bloquear su equipo si está conectado a un dominio de red, o cambiar de usuario si no está conectado a un dominio de red.
+ R	Abrir el cuadro de diálogo <b>Ejecutar</b> .
	Mostrar el menú contextual del elemento seleccionado.
+ U	Abrir Administrador de utilidades.

### Métodos abreviados de teclado de accesibilidad

Presione	Para
MAYÚS DER durante ocho segundos	Activar y desactivar FilterKeys.
ALT izq + MAYÚS izq + IMPR PANT	Activar y desactivar Contraste alto.
ALT IZQ + MAYÚS IZQ + BLOQ NUM	Activar y desactivar MouseKeys.
MAYÚS cinco veces	Activar y desactivar StickyKeys.
BLOQ NUM durante cinco segundos	Activar y desactivar ToggleKeys.
+ U	Abrir Administrador de utilidades.

### Métodos abreviados de teclado del Explorador de Windows

Presione	Para
FIN	Mostrar la parte inferior de la ventana activa.
INICIO	Mostrar la parte superior de la ventana activa.
BLOQ NUM+ASTERISCO en el teclado numérico (*)	Mostrar todas las subcarpetas de la carpeta seleccionada.

BLOQ NUM+SIGNO MÁS en el teclado numérico (+)	Mostrar el contenido de la carpeta seleccionada.
BLOQ NUM+SIGNO MENOS en el teclado numérico (-)	Contraer la carpeta seleccionada.
FLECHA IZQUIERDA	Contraer la selección actual si está expandida o seleccionar la carpeta principal.
FLECHA DERECHA	Mostrar la selección actual si está contraída o seleccionar la primera subcarpeta.

	IcVoz	Documento : <b>ANEXO B</b>	Rev.: <b>1.0</b>
	<b>ESPECIFICACIÓN DE REQUERIMIENTOS</b>	Página : <b>1 de 7</b>	

<b>REQUERIMIENTOS FUNCIONALES</b>		
<b>REF N°</b>	<b>FUNCIONES</b>	<b>CATEGORÍA</b>
1.	El interpretador al emular el funcionamiento del teclado por ende el comando ejecutado debe estar asociado con el nombre del comando de voz.	Visible
2.	El inicio o arranque del interpretador estará a cargo del sistema operativo, es decir, se encontrará un script o ejecutable el cual el sistema operativo podrá ejecutar al iniciarse.	Oculto
3.	El usuario debe tener la opción de poder iniciar o arrancar el interpretador de forma manual dependiendo si el usuario lo elige de esta forma.	Visible
4.	El interpretador de comandos debe poseer un ícono en la barra del sistema, el cual tendrá un menú con las opciones de salir, ayuda, configurar, reiniciar, parar e iniciar.	Visible
5.	El interpretador debe reconocer solo palabras aisladas como comandos que se encuentran almacenadas en sus archivos de configuración.	Oculto
6.	La señal de audio que recibe el interpretador de comandos de voz debe ser sometida a un pre-énfasis de señal y después pasa por una serie de ventanas, para su correcto funcionamiento.	Oculto
7.	El interpretador de comandos detectará la presencia en la señal de audio del usuario, la cual se extraerá para su procesamiento.	Oculto
8.	El interpretador implementará un algoritmo de detección	Oculto

	de bordes para analizar si en la señal hay presencia de la voz del usuario.	
9.	La parte de la señal extraída como un comando de voz pronunciado por el usuario, el interpretador deberá extraerle sus componentes frecuenciales para construir el patrón que se analizara.	Oculto
10.	El interpretador implementará una red neuronal artificial con manejo de memoria, como herramienta para identificación de patrones.	Oculto
11.	La red neuronal debe tener un entrenamiento previo para que la voz de los usuarios sea identificada por el interpretador de comandos de voz.	Oculto
12.	El interpretador de comandos debe guardar la configuración o perfil en archivos XML, tanto para la configuración como para almacenar los datos de la Red Neuronal Artificial.	Oculto
13.	El interpretador de comandos de voz tendrá por cada usuario un grupo de archivos de configuración al cual se le denomina “perfil de usuario”, en los cuales se almacenara la información de la red neuronal y de los comandos de voz en archivos con formato XML.	Visible
14.	El interpretador de comandos de voz guardará en una carpeta con el nombre del perfil los archivos de configuración del perfil.	Visible
15.	El interpretador de comandos de voz debe tener un único archivo de configuración, en el cual se almacenará la información en formato XML de los perfiles existentes en el sistema y el perfil que se encuentre como predeterminado.	Visible
16.	El interpretador deberá grabar los archivos de audio en formato WaveForm, para ser reproducidos cuando el usuario realice una configuración.	Oculto
17.	El interpretador de comandos de voz deberá poseer una	Visible

	interfaz gráfica que el permita configurar al usuario, la configuración del perfil, edición y/o modificación de cada comando de voz, a la interfaz de configuración se le denominará asistente de configuración.	
18.	El interpretador al ejecutar la opción de configuración del menú, deberá desplegar el asistente de configuración con una interfaz que da un mensaje de bienvenida y en el cual el usuario podrá pulsar un botón para continuar.	Visible
19.	En el asistente de configuración el usuario podrá crear perfiles de usuario, edición y borrado de perfiles a excepción de un perfil base llamado “default” el cual no podrá ser editado ni borrado. Cuando el usuario haya presionado el botón continuar en el mensaje de bienvenida.	Visible
20.	Cuando el usuario desee crear un nuevo perfil, el asistente debe realizar la validación de que no se encuentre otro perfil con el mismo nombre.	Oculto
21.	El asistente de configuración deberá darle la opción al usuario de escoger el perfil con el cual el interpretador va a funcionar.	Visible
22.	Cuando el usuario seleccione el perfil que va a configurar, el asistente debe desplegar una interfaz mostrando todos los comandos con la opción de edición y de activación individual de cada comando.	Visible
23.	Los comandos de voz correspondientes a cada perfil se podrán activar o desactivar individualmente para su uso.	Visible
24.	Cuando el usuario presione el botón de edición del comando, el asistente mostrara una interfaz en la cual el usuario podrá grabar, parar y reproducir el comando que se encuentra modificando.	Visible
25.	En el asistente de configuración, cuando el usuario detiene la grabación, inmediatamente el sistema debe notificar si hubo o no presencia de una señal en la grabación, es decir si identificó audio que sobresaliera del ruido de fondo.	Visible

26.	Cuando el usuario acepte la modificación del comando el interpretador le dará la opción de editar otro comando, al aceptar esta opción, el asistente regresará a la interfaz anterior. En caso contrario el interpretador guardará los cambios y activará el entrenamiento de la red neuronal con los nuevos parámetros.	Visible
27.	El asistente de configuración debe mostrar un mensaje y una barra de progreso que le permite a usuario conocer el estado de la configuración, en el cual internamente se estará llevando a cabo el proceso de entrenamiento de la red neuronal artificial, cuando el usuario haya modificado y aceptado los cambios a un comando o grupo de ellos.	Visible
28.	En el seguimiento del asistente de configuración, el interpretador deberá poseer un botón de ayuda que le permitirá al usuario pulsarlo para conocer cómo se debe realizar la tarea de configuración en la interfaz que posea en ese instante.	Visible
29.	En cualquier momento el asistente deberá tener un botón que le permita al usuario cancelar la operación que se encuentra en ese instante sin afectar la configuración actual que estaba intentando modificar.	Visible
30.	En el proceso de edición de un comando el asistente deberá mostrar las ondas de voz mientras se realiza la grabación de un comando.	Visible
31.	El usuario al presionar salir del menú, el interpretador validara que el asistente no este abierto para cerrar la aplicación.	Oculto
32.	Al seleccionar “iniciar” en el menú del interpretador ubicado en la barra del sistema, este debe iniciar los procesos de capturar, detección y ejecución del comando de voz.	Oculto
33.	El usuario al seleccionar “reiniciar” en el menú del interpretador, debe parar los procesos e iniciarlos	Oculto

	nuevamente.	
34.	El interpretador detendrá los procesos de captura, detección y ejecución del comando, cuando el usuario seleccione “parar” del menú del interpretador.	Oculto
35.	El sistema de producción o ejecución del comando de voz no debe contar con interfaz de usuario, solo se mostrara mensajes salientes del ícono de la barra del sistema describiendo el comando que ha pronunciado.	Visible
36.	El interpretador en la sección de ejecución deberá estar compuesto de tres sub-procesos principales que son: captura de audio, detección de bordes (captura el comando pronunciado por el usuario) y ejecución del comando.	Visible
37.	El interpretador en el sub-proceso ejecución del comando, contara con la extracción de características del habla, ejecución de la red neuronal y posteriormente se procederá a ejecutar el comando que la red neuronal haya arrojado.	Visible
38.	El interpretador en el sub-proceso de ejecución del comando, se debe implementar el algoritmo de la Transformada rápida de Fourier.	Oculto
39.	El interpretador deberá poseer una ayuda organizada por tópicos principales que son, perfiles, comandos, temas relacionados con el funcionamiento de la aplicación e información de los autores y licencias de librerías y códigos que el interpretador posea.	Visible
40.	El interpretador de comandos debe cargar el perfil que se encuentre como predeterminado para ser usado, el cual se encuentra en el archivo de configuración, cuando se inicie la aplicación y cuando el usuario seleccione la opción de iniciar o reiniciar del menú de la aplicación.	Oculto
41.	El interpretador debe mostrar un mensaje de “Palabra demasiado larga” en el ícono de la barra del sistema, cuando el usuario pronuncie una palabra que exalimite un tiempo prefijado de 3 segundos para la palabra más larga.	Visible

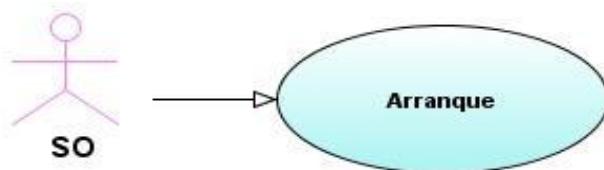
42.	Al instanciar la red neuronal esta debe cargar del archivo correspondiente al perfil predeterminado los pesos y umbrales para poder que la red trabaje adecuadamente, cuando el proceso interno lo indique.	Oculto
43.	Cuando el proceso de entrenamiento de la red neuronal finalice, el interpretador debe guardar inmediatamente los pesos y umbrales que la red entreno en el archivo correspondiente al perfil modificado.	Oculto
44.	En el proceso de edición, cuando el usuario detenga la grabación, el sistema realizara el proceso de detección de bordes y extracción de características espectrales en el caso de que se haya encontrado información en la grabación, y los datos permanecerán en memoria hasta que el usuario guarde o cancele los cambios.	Oculto
45.	En el interpretador de comandos, cuando el usuario pronuncie un comando que sea válido o esté habilitado, el sistema debe mostrar un mensaje en el icono de la barra del sistema diciendo que comando acaba de decir.	Visible
46.	Cuando el usuario seleccione un perfil como predeterminado para ser usado, el interpretador de comandos debe mostrar un mensaje de confirmación para poder realizar el cambio.	Visible
47.	El interpretador al momento que un usuario haya escogido otro perfil como predeterminado y lo ha aceptado, el sistema debe guardar los cambios en el archivo principal inmediatamente.	Oculto
48.	Cuando el usuario seleccione o deseccione un comando de voz, el interpretador debe mostrar un dialogo de confirmación para realizar los cambios.	Visible
49.	Cuando un usuario haya seleccionado o deseleccionado un comando de voz y ha aceptado el cambio, el sistema debe inmediatamente guardar el cambio en su archivo de configuración respectivo al perfil de modificación.	Oculto

50.	Cuando el usuario borre un perfil, el sistema debe borrar el perfil del archivo de configuración principal y los directorios y archivos que han sido generados por el interpretador los cuales están relacionados con este perfil.	Oculto
51.	El sistema cuando no este activado debe mostrar un icono en la barra del sistema en escala de grises, en caso contrario a color.	Visible

<b>REQUERIMIENTOS NO FUNCIONALES</b>		
<b>REF N°</b>	<b>FUNCIONES</b>	<b>CATEGORÍA</b>
52.	En el interpretador de comandos de voz; desde el momento que el usuario pronuncie el comando hasta su ejecución debe demorase un tiempo máximo de cinco segundos.	Visible
53.	Se requiere que el usuario que ejecute la aplicación tenga JVM (Maquina Virtual de JAVA) instalado (JDK 1.6.0).	Visible
54.	Se requiere que el Sistema Operativo soporte la JVM (JDK 1.6.0).	Oculto
55.	Se requiere que el usuario tenga los conocimientos para el manejo de Windows (o el sistema operativo en el que sea instalado).	Visible

	<b>IcVoz</b>	Documento : <b>ANEXO C</b>	Rev.: <b>1.0</b>
<b>CASOS DE USO</b>			Página : <b>1 de 8</b>

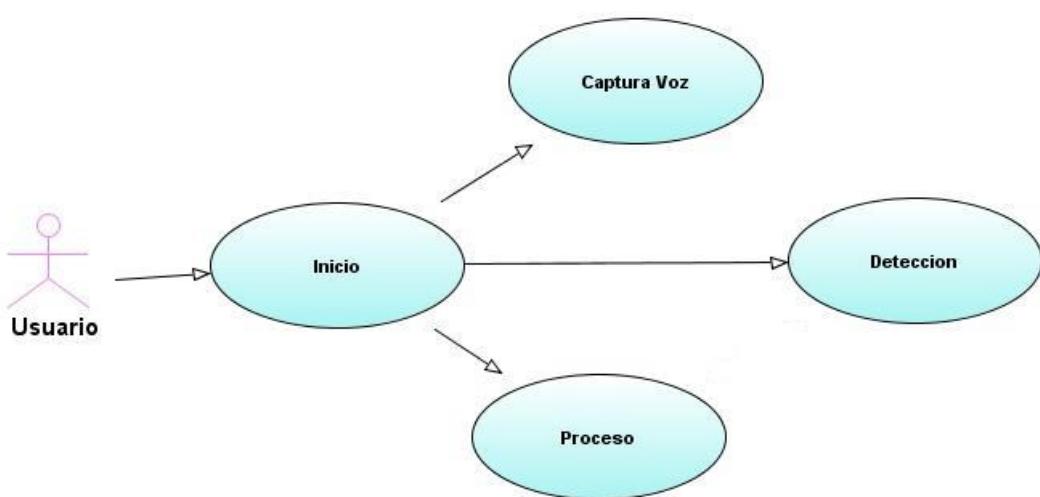
### Caso de Uso: Inicio



INFORMACIÓN GENERAL	
<b>Nombre:</b>	Arranque
<b>Actores:</b>	Sistema Operativo
<b>Propósito:</b>	Arranque de la aplicación
<b>Resumen:</b>	El sistema operativo al iniciar también arranca la aplicación junto con el sistema
<b>Referencias Cruzadas:</b>	Requerimiento N°: 2-3-32-51

iv. Curso Normal de los Eventos	
Acción de los Actores	Respuesta del Sistema
1. El usuario enciende el computador	2. El sistema carga todos los procesos y la aplicación. 3. El sistema muestra un ícono en la barra del sistema el cual contendrá el menú de la aplicación. Este ícono inicialmente aparecerá en escala de grises ya que esta desactivado.

## Caso de Uso: Captura de Voz



INFORMACIÓN GENERAL	
<b>Nombre:</b>	Captura de Voz
<b>Actores:</b>	Usuario
<b>Propósito:</b>	Capturar la voz humana por medio del micrófono
<b>Resumen:</b>	La aplicación captura la voz humana y la detecta para después ser interpretada y ejecutada
<b>Referencias Cruzadas:</b>	Requerimiento N°: 36

<i>v.Curso Normal de los Eventos</i>	
<i>Acción de los Actores</i>	<i>Respuesta del Sistema</i>
1. El usuario inicia la aplicación. 2. El usuario habla	3. El sistema detectará la presencia de señal de audio del usuario. 4. El sistema captura la voz

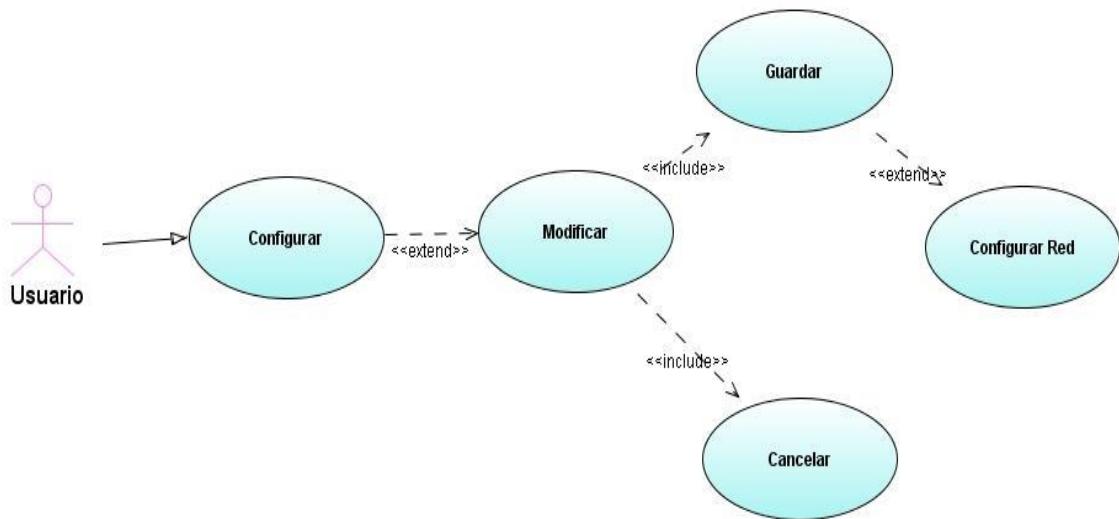
INFORMACIÓN GENERAL	
<b>Nombre:</b>	Detección
<b>Actores:</b>	sistema
<b>Propósito:</b>	Detectar el comando pronunciado por el usuario
<b>Resumen:</b>	La aplicación después de capturar la voz humana pasa a detectar el comando pronunciado.
<b>Referencias Cruzadas:</b>	Requerimiento N°: 7-8-32

<i>vi.Curso Normal de los Eventos</i>	
<i>Acción de los Actores</i>	<i>Respuesta del Sistema</i>
	<ol style="list-style-type: none"> <li>1. El sistema detectará la presencia de señal de audio del usuario.</li> <li>2. El sistema capture la voz</li> <li>3. El sistema utiliza un algoritmo de detección de bordes.</li> <li>4. Guarda la información en un paquete.</li> </ol>

<b>INFORMACIÓN GENERAL</b>	
<b>Nombre:</b>	Proceso
<b>Actores:</b>	Usuario
<b>Propósito:</b>	Procesa el comando pronunciado a partir una serie de cálculos como lo son: calculo del espectro, proceso de la red neuronal y envío de los valores calculados al dispositivo.
<b>Resumen:</b>	Procesa los comandos pronunciados por el usuario a través de una serie de cálculos y los ejecuta.
<b>Referencias Cruzadas:</b>	Requerimiento N°: 9-10-35-37-45

<i>vii.Curso Normal de los Eventos</i>	
<i>Acción de los Actores</i>	<i>Respuesta del Sistema</i>
	<ol style="list-style-type: none"> <li>1. Lee buffer el comando guardado.</li> <li>2. Calcula del espectro de la señal.</li> <li>3. La red neuronal procesa el comando y calcula sus pesos.</li> <li>4. Se envía la orden para que se ejecute el comando pronunciado en el dispositivo.</li> </ol>

## Caso de Uso: Configuración



INFORMACIÓN GENERAL	
<b>Nombre:</b>	Configurar
<b>Actores:</b>	Usuario
<b>Propósito:</b>	Configurar el perfil del usuario y los comandos a utilizar
<b>Resumen:</b>	La aplicación ofrece al usuario un previo entrenamiento para poder utilizar la aplicación.
<b>Referencias Cruzadas:</b>	Requerimiento N°: 4-13-17-19-21-22-40

<i>viii. Curso Normal de los Eventos</i>	
<i>Acción de los Actores</i>	<i>Respuesta del Sistema</i>
1. El usuario hace clic derecho en el icono de la barra del sistema. 2. Escoge la opción configurar 5. El usuario escoge la opción configurar	3. La aplicación muestra una ventana con un mensaje de bienvenido y con la opción del botón siguiente. 4. El sistema muestra una lista de perfiles con las opciones de configurar un perfil o crear uno nuevo. 6. El sistema muestra la lista de los 20 comandos que podrá configurar

INFORMACIÓN GENERAL	
<b>Nombre:</b>	Comandos de voz
<b>Actores:</b>	usuario
<b>Propósito:</b>	Elegir los comandos de voz a utilizar
<b>Resumen:</b>	La aplicación ofrece al usuario una lista de comando para que el usuario pueda utilizar.
<b>Referencias Cruzadas:</b>	Requerimiento N°:17-22-23-24

<i>ix.Curso Normal de los Eventos</i>	
<i>Acción de los Actores</i>	<i>Respuesta del Sistema</i>
1. El usuario elige los comandos que va a utilizar haciendo clic en el botón editar.	2. El sistema muestra una ventana en donde se procederá a configurar cada comando que el usuario selecciono.

INFORMACIÓN GENERAL	
<b>Nombre:</b>	Modificar
<b>Actores:</b>	usuario
<b>Propósito:</b>	Asignar de acuerdo a la voz del usuario los comandos que va a utilizar
<b>Resumen:</b>	El sistema toma los valores capturados por la voz del usuario para conservar o rechazar el comando pronunciado.
<b>Referencias Cruzadas:</b>	Requerimiento N°:16-24-29

<i>x.Curso Normal de los Eventos</i>	
<i>Acción de los Actores</i>	<i>Respuesta del Sistema</i>
2. El usuario hace clic en el botón grabar. 3. Pronuncia el comando. 4. Hace clic en el botón parar y detiene la grabación. 5. Si desea reproduce la grabación y si el usuario acepta la grabación hace clic en aceptar o sino en cancelar.	1. El sistema muestra una ventana con las opciones de grabar, parar y reproducir.

INFORMACIÓN GENERAL	
<b>Nombre:</b>	Guardar
<b>Actores:</b>	usuario
<b>Propósito:</b>	Guardar los valores que se detectaron por cada comando pronunciado
<b>Resumen:</b>	La aplicación guarda los valores de los comandos de voz para después pasarlo a la red neuronal
<b>Referencias Cruzadas:</b>	Requerimiento N°:12-26-49

<i>xi.Curso Normal de los Eventos</i>	
<i>Acción de los Actores</i>	<i>Respuesta del Sistema</i>
1. Usuario hace clic en aceptar	2. El sistema guarda el comando pronunciado. 3. El sistema le pregunta al usuario si desea configurar más comandos.

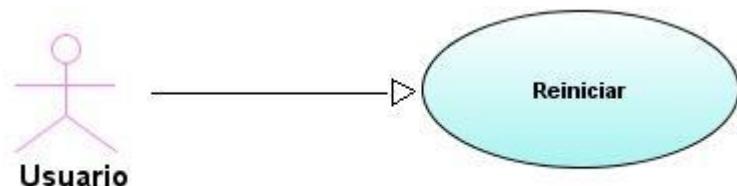
INFORMACIÓN GENERAL	
<b>Nombre:</b>	Cancelar
<b>Actores:</b>	usuario
<b>Propósito:</b>	Cancela los valores que se detectaron por cada comando pronunciado
<b>Resumen:</b>	El sistema toma los valores capturados por la voz del usuario y rechaza el comando pronunciado.
<b>Referencias Cruzadas:</b>	Requerimiento N°:29

<i>xii.Curso Normal de los Eventos</i>	
<i>Acción de los Actores</i>	<i>Respuesta del Sistema</i>
1. Usuario hace clic en cancelar	2. El sistema muestra que ha finalizado el asistente. 3. El sistema no guarda nada en el archivo y finaliza la acción.

INFORMACIÓN GENERAL	
<b>Nombre:</b>	Configurar Red
<b>Actores:</b>	Sistema
<b>Propósito:</b>	Pasar los valores previamente guardados a la red neuronal
<b>Resumen:</b>	La aplicación asigna los valores de los comandos de voz pronunciados por el usuario y estos son pasados a la red neuronal.
<b>Referencias Cruzadas:</b>	Requerimiento N°:10-11-12-13-26-27-43

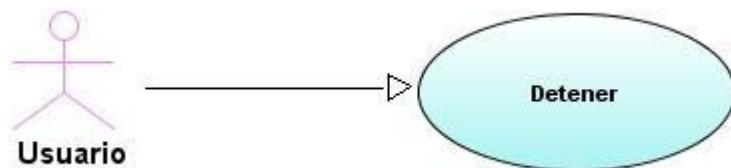
<i>xiii. Curso Normal de los Eventos</i>	
<i>Acción de los Actores</i>	<i>Respuesta del Sistema</i>
	1. El sistema por medio de la red neuronal guarda los pesos del comando pronunciado en un archivo xml.

Caso de Uso: Reiniciar
------------------------



INFORMACIÓN GENERAL	
<b>Nombre:</b>	Re却iciar
<b>Actores:</b>	Usuario
<b>Propósito:</b>	Volver a inicializar el sistema
<b>Resumen:</b>	La aplicació却 arranca de nuevo
<b>Referencias Cruzadas:</b>	Requerimiento N°:4-33-40

<i>xiv.Curso Normal de los Eventos</i>	
<i>Acción de los Actores</i>	<i>Respuesta del Sistema</i>
1. El usuario hace clic en la opción reiniciar	2. La aplicación inicializa los hilos de la aplicación. 3. El color del ícono de la aplicación en la barra de del sistema se coloca azul.



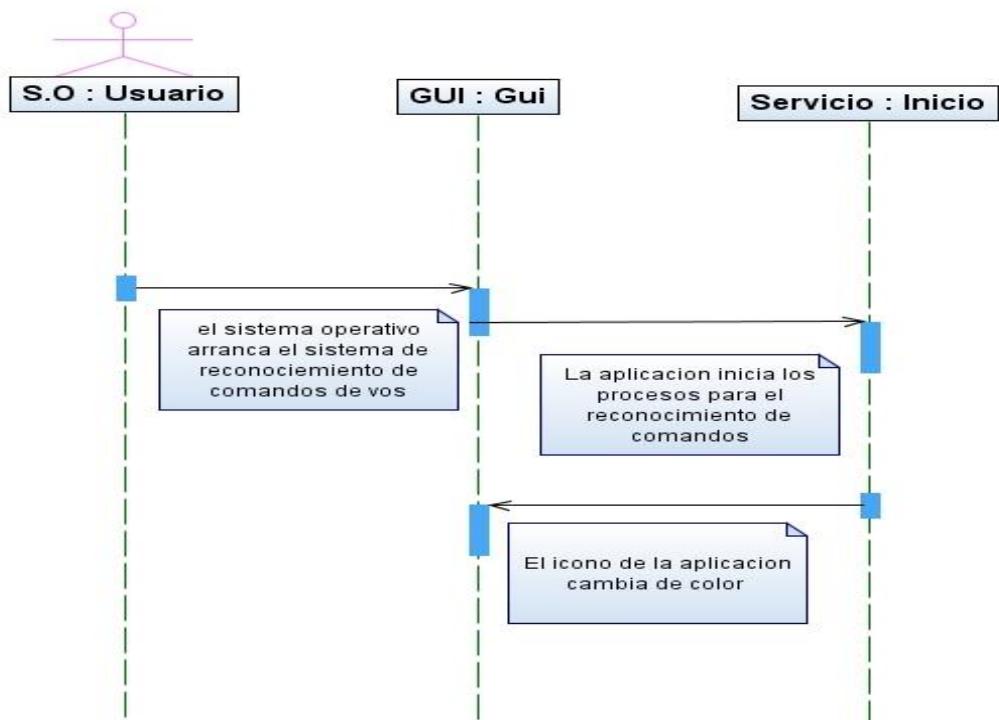
INFORMACIÓN GENERAL	
<i>Nombre:</i>	Detener
<i>Actores:</i>	Usuario
<i>Propósito:</i>	Detiene el sistema
<i>Resumen:</i>	La aplicación para todos los procesos
<i>Referencias Cruzadas:</i>	Rqto nro. 4-34

<i>xv.Curso Normal de los Eventos</i>	
<i>Acción de los Actores</i>	<i>Respuesta del Sistema</i>
1. El usuario hace clic en la opción parar	2. La aplicación detiene los hilos de la aplicación. 3. El ícono de la barra del sistema se coloca de color gris

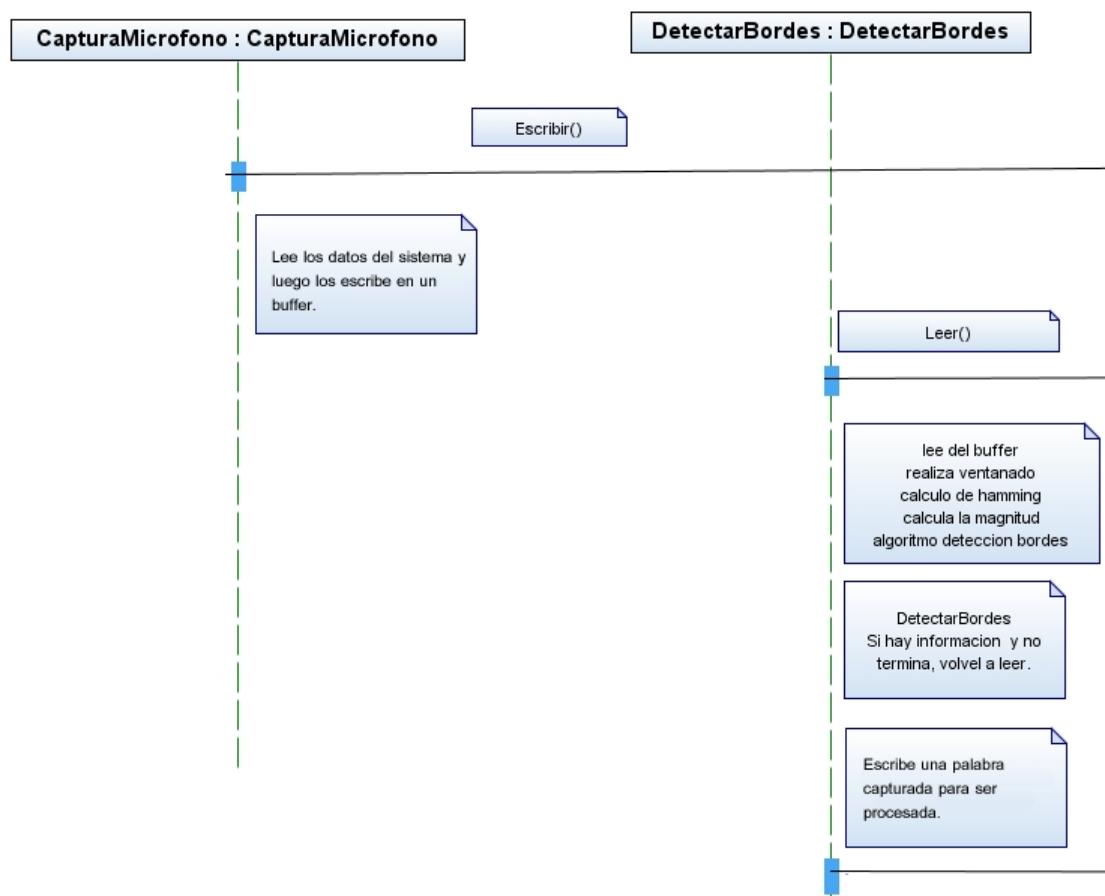
	IcVoz	Documento : ANEXO D	Rev.: 1.0
<b>DIAGRAMAS DE SECUENCIA</b>			Página : <b>1 de 6</b>

### Secuencia: Iniciar

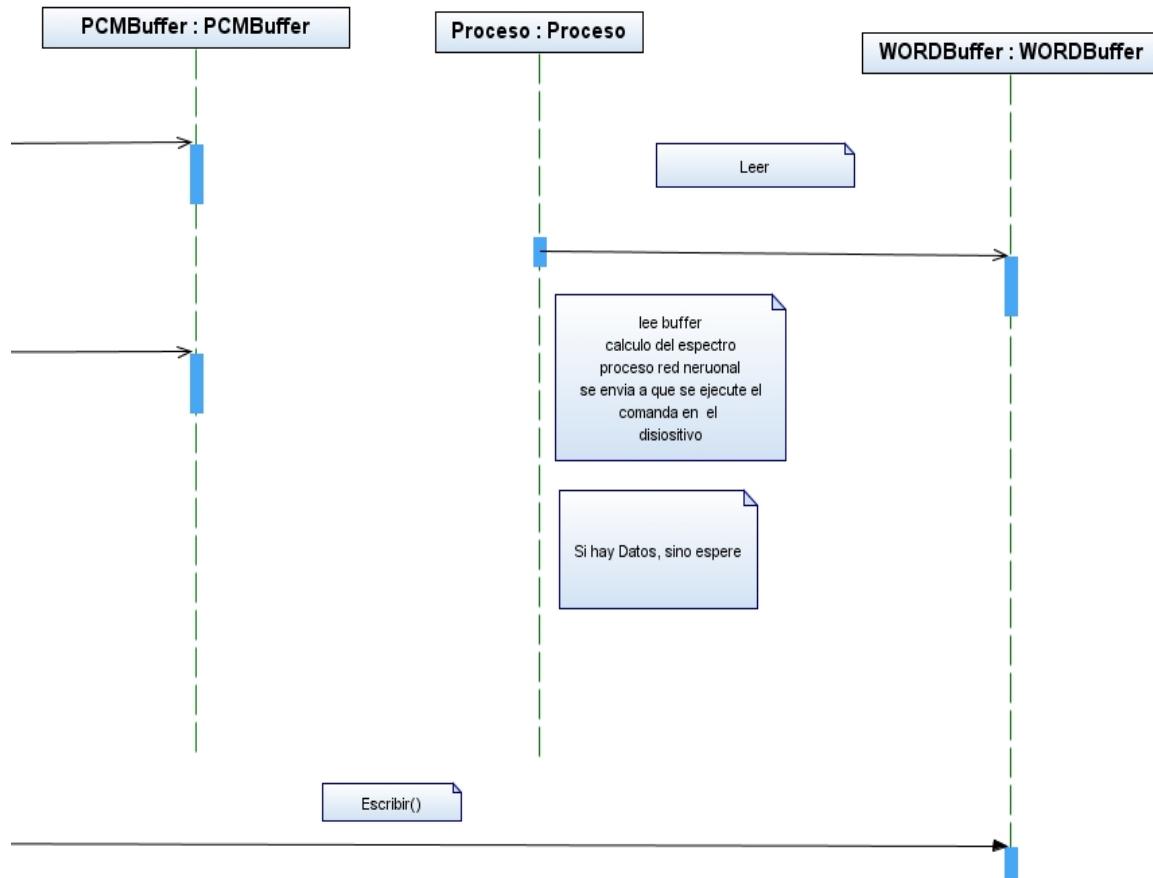
#### Secuencia Iniciar



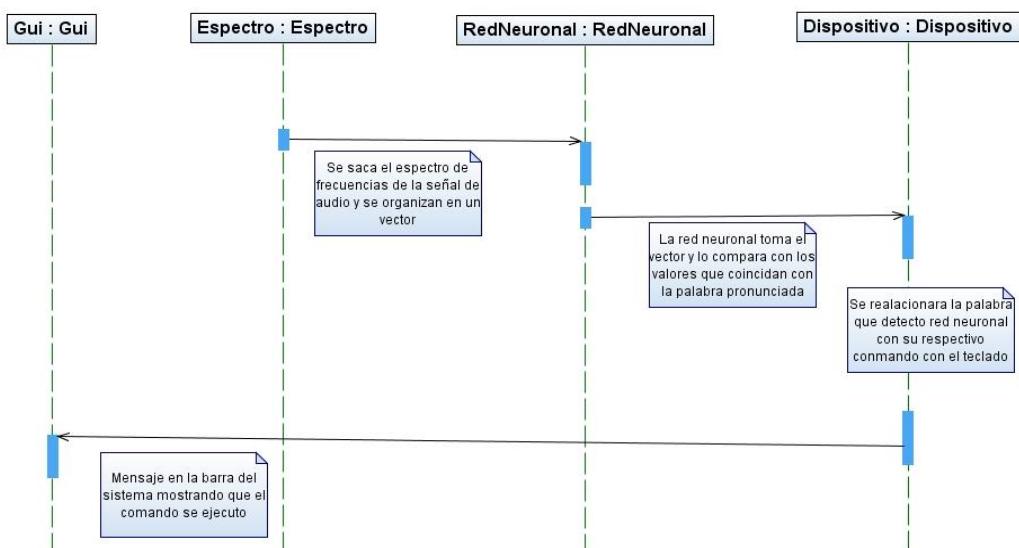
## Secuencia: Capturar Voz



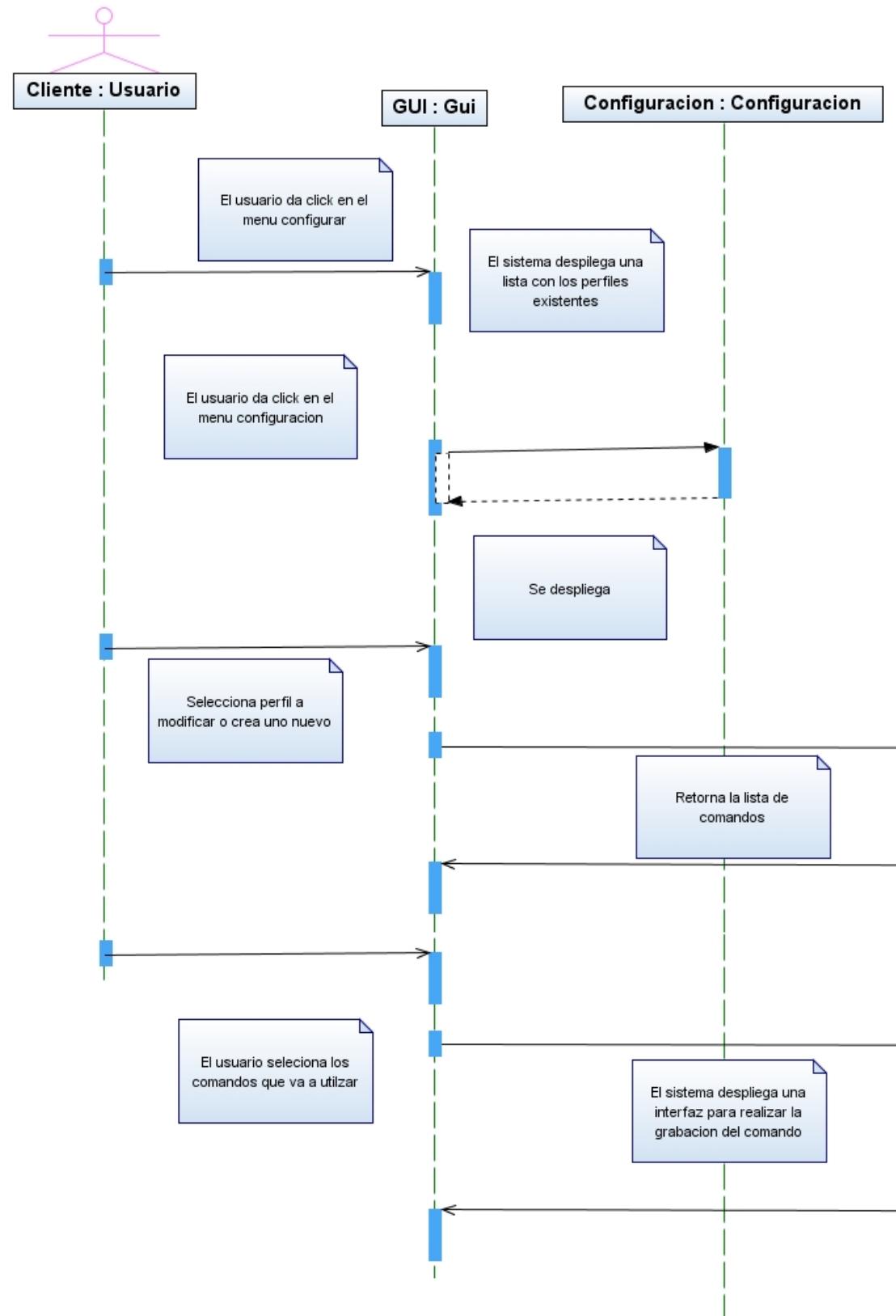
## Secuencia: Capturar Voz



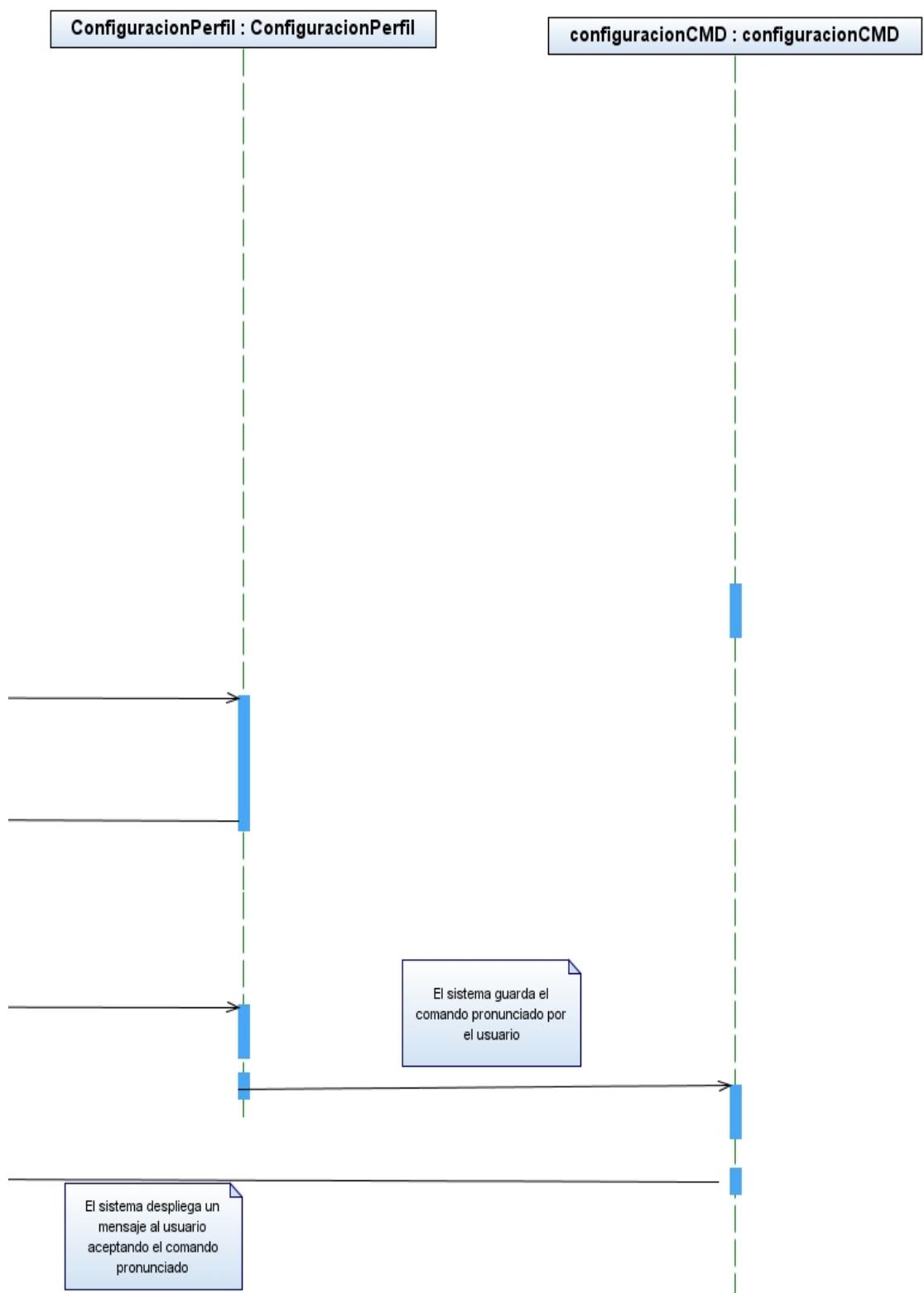
## Secuencia: Proceso (ampliación)



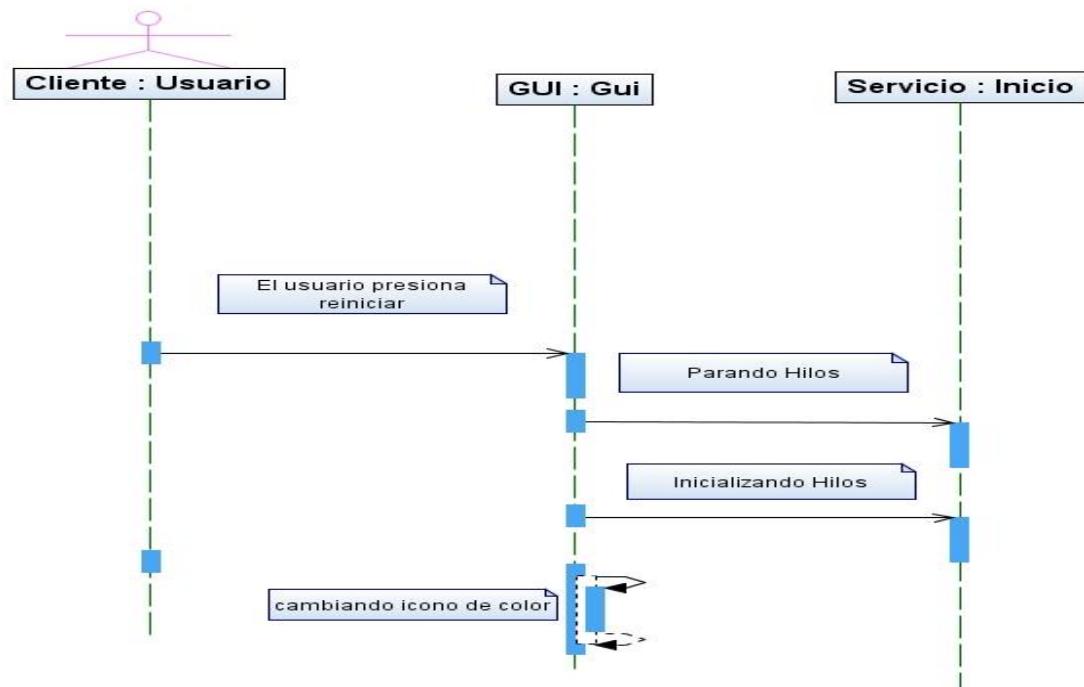
## Secuencia: Configurar



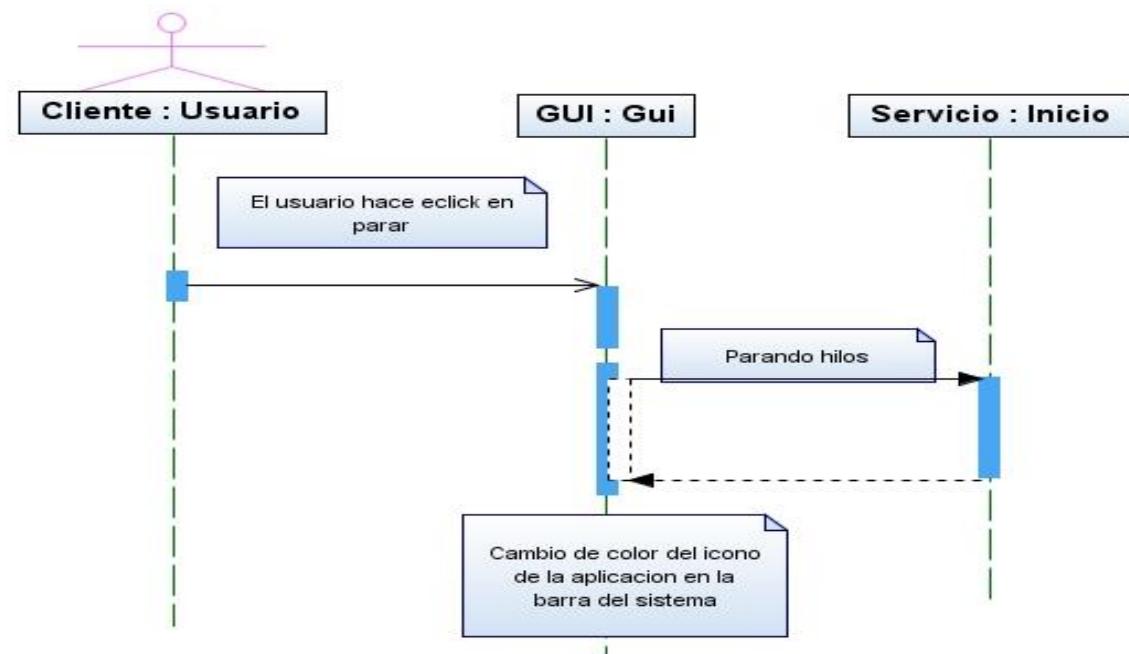
## Secuencia: Configurar



## Secuencia: Reiniciar

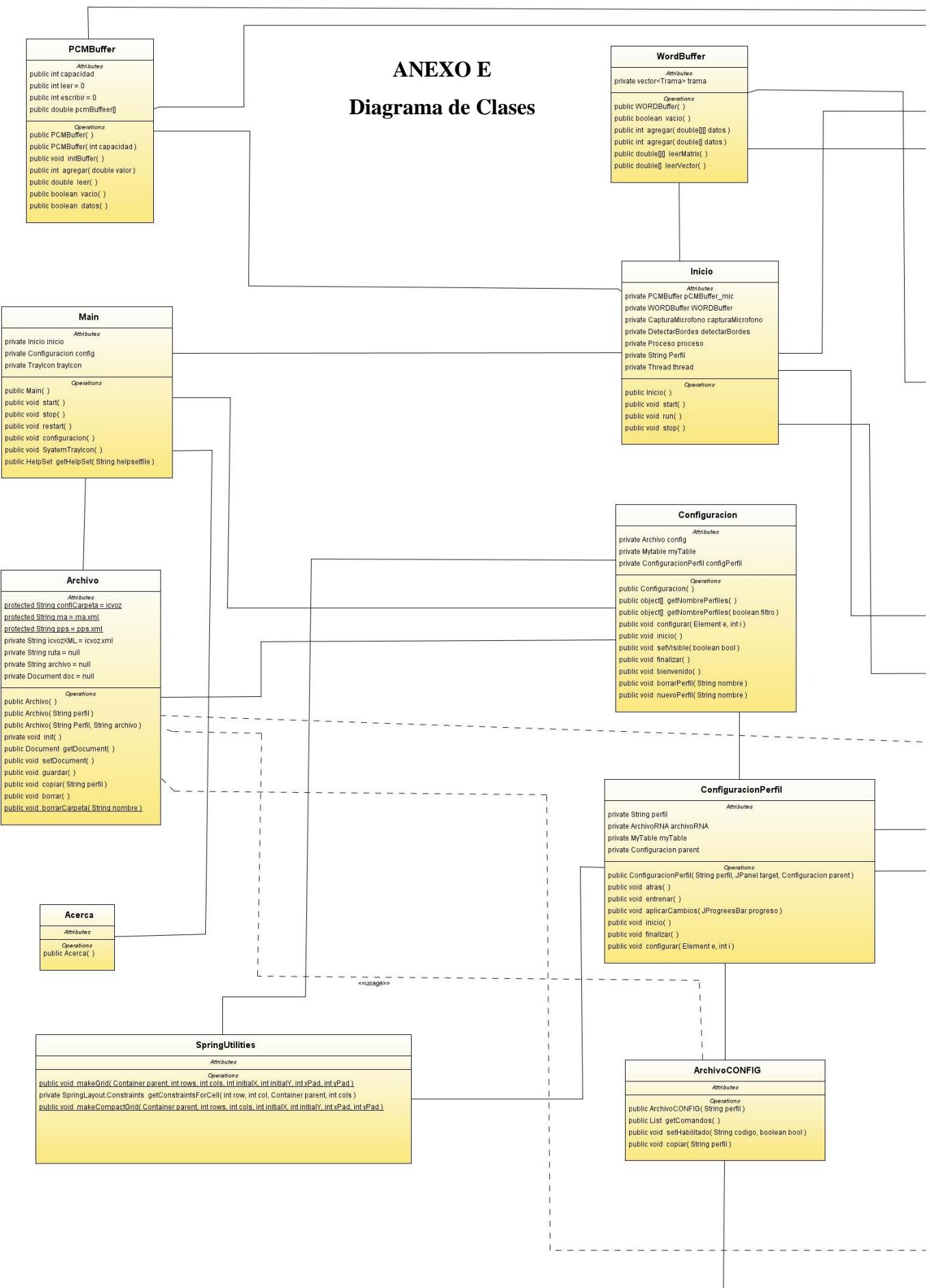


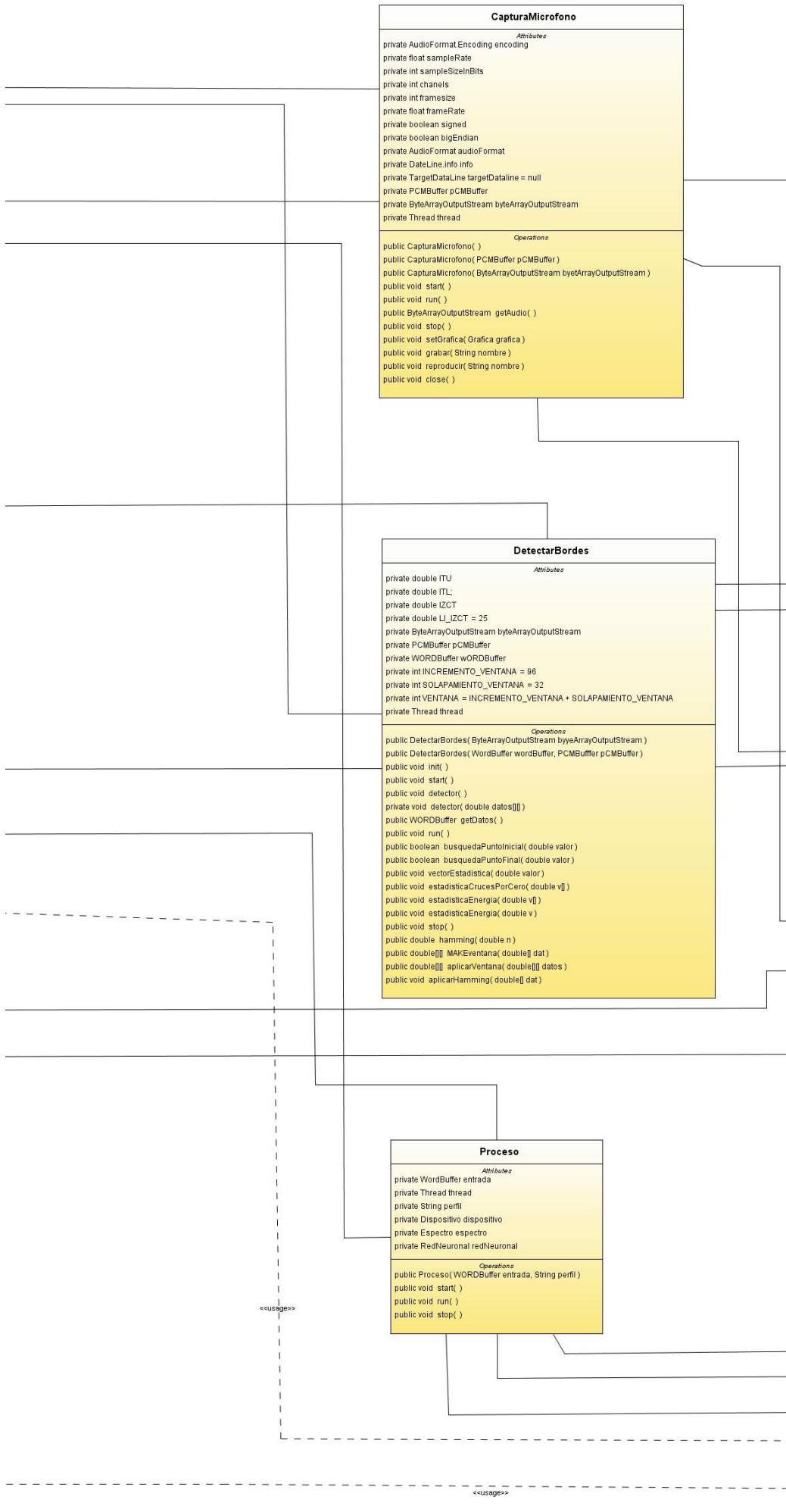
## Secuencia: Parar

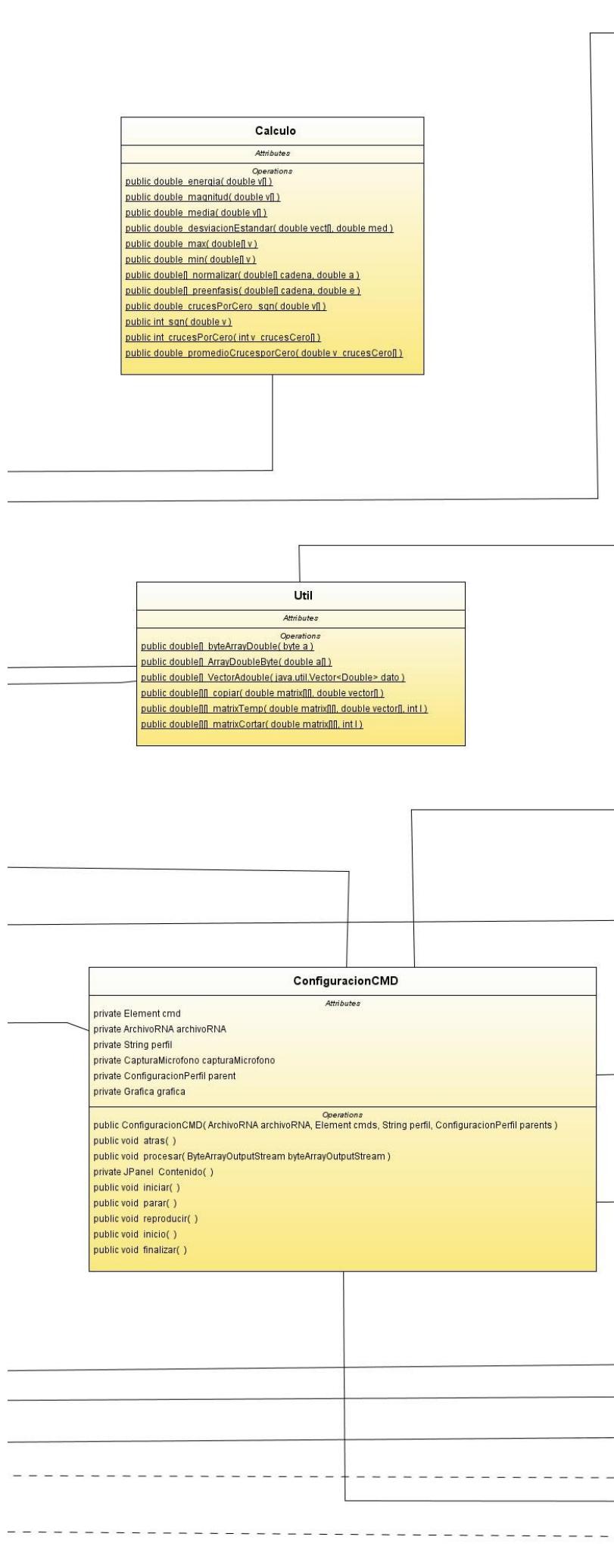


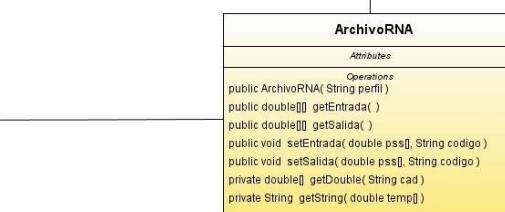
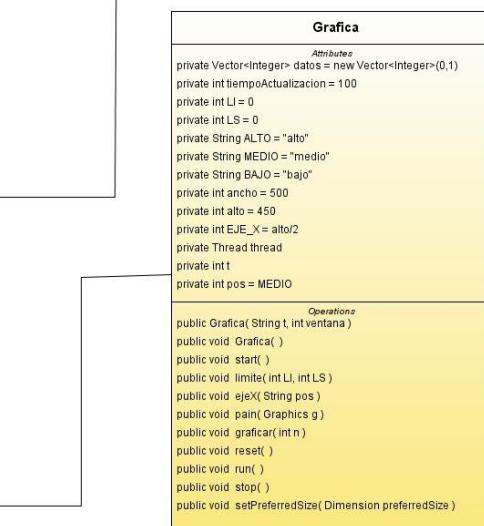
## ANEXO E

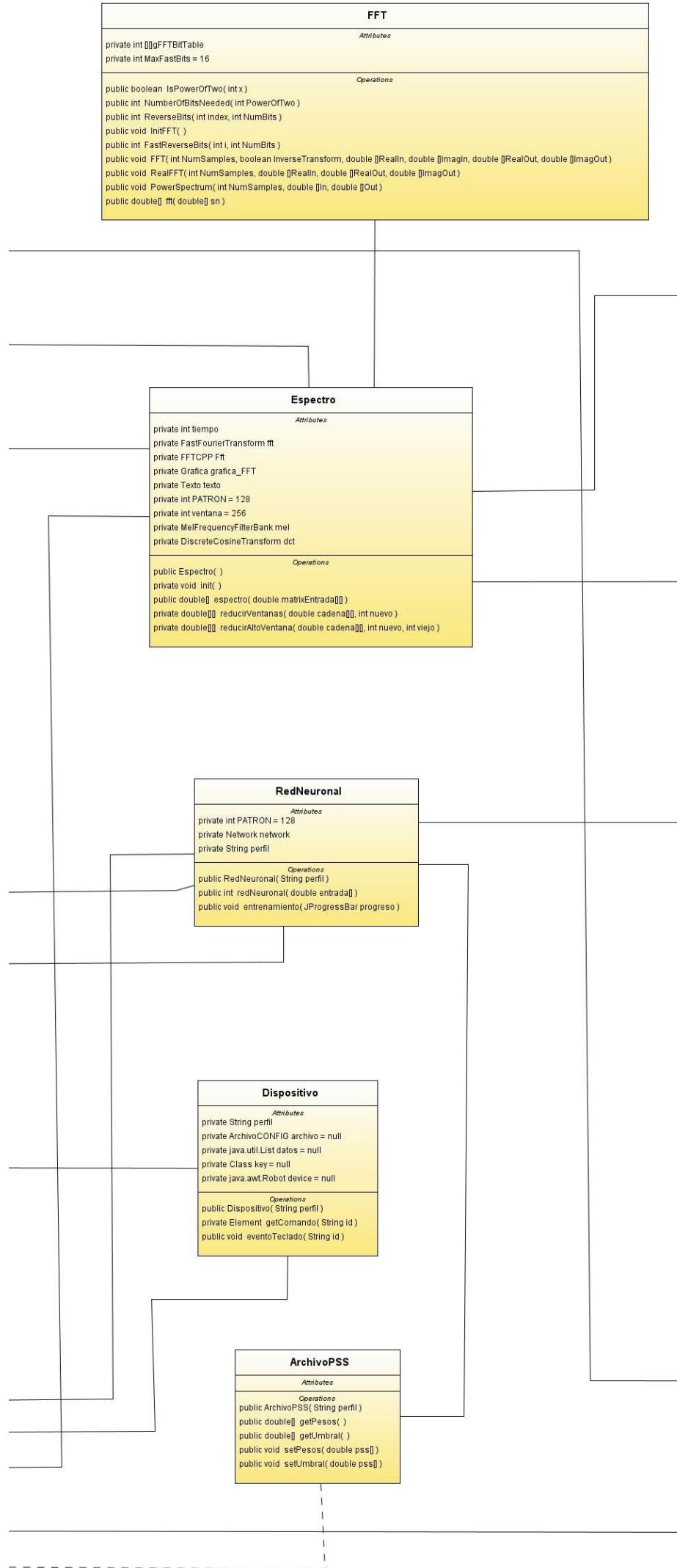
### Diagrama de Clases

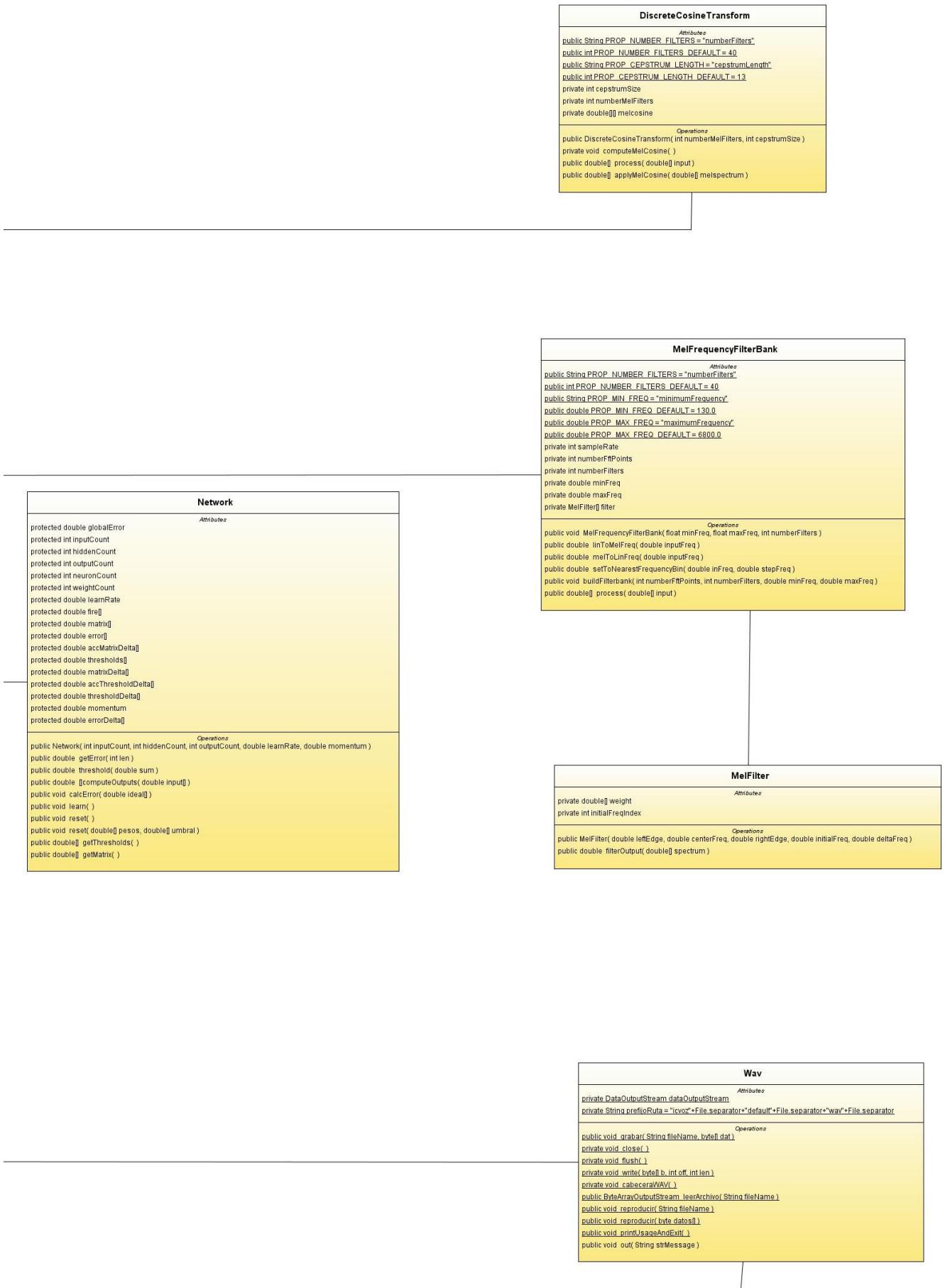












## **ANEXO F**

### **Pruebas Realizadas a diez usuarios**

Este anexo muestra los resultados individuales de la prueba uno y dos.

Se dispuso una segunda prueba donde los usuarios se le indicada a través de un texto unas tareas a realizar utilizando comandos de voz.

El programa que se utilizo para visualizar y aplicar la prueba fue Microsoft Office Word 2003.

#### **Texto para probar el IcVoz**

Por favor siga el texto detenidamente, si tiene alguna pregunta mientras realiza la prueba por favor digala de inmediato.

- 1.** Seleccione el texto que se encuentra en comillas utilizando el teclado y/o el Mouse, cuando tenga seleccionado el texto diga el comando de voz "copiar".

"Un programa de computador siempre hará lo que le ordenes que haga, no lo que quieras que haga."

- 2.** Ponga el cursor en el asterisco y ejecute el comando de voz "pegar".

\*

- 3.** Por favor ejecute el comando de voz "deshacer".

- 4.** Lea antes de realizar cualquier operación: Del siguiente texto cambie la palabra "archive" por el

sinónimo "guardé" de la siguiente manera: Ubique el cursor en la palabra "archive", cuando este ubicado en ese punto ejecute el comando de voz "menú" y navegue en el menú con los comandos "abajo", "izquierda", "derecha" y "arriba", y para seleccionar el sinónimo ejecute el comando "enter". Ahora puede realizar lo indicado.

"Cuando archive algo en la memoria, acuérdese de donde lo guardo."

- 5.** Lea con cuidado antes de realizar cualquier operación: Después de haber realizado los puntos anteriores tiene que ir al escritorio y volver a este documento de la siguiente manera: Ejecute el comando "escritorio" para salir, para volver a entrar ejecute el comando "alternar" hasta que encuentre la ventana que contiene este documento. Ahora puede realizar lo indicado.
  
- 6.** Como última prueba debe cerrar este documento sin guardar los cambios, utilizando solo comandos de voz de la siguiente forma: Para cerrar la venta ejecute "cerrar", para cambiar de opción navegue con los comandos "izquierda" o "derecha" y para presionar el botón correspondiente utilicé el comando "enter". Ahora puede realizar lo indicado.

Gracias por su colaboración.

Los resultados de la segunda prueba son los siguientes:

<b>Usuario \ Ítem</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>
limp	SI	NO	SI	S/N	SI	SI
lufe	SI	SI	SI	SI	SI	SI
vicm	SI	SI	SI	NO	SI	SI
cahe	SI	SI	NO	NO	SI	S/N
john	SI	NO	SI	SI	SI	SI
pave	SI	SI	SI	SI	SI	SI
alex	NO	SI	SI	SI	SI	SI
dein	SI	SI	SI	NO	SI	SI
yuri	NO	SI	SI	SI	SI	SI
sami	SI	SI	SI	SI	SI	SI

Tabla de datos segunda prueba.

Descripción:

SI: punto realizado con éxito, algunos usuarios lo lograron después de un máximo de tres intentos.

NO: Despues de varios intentos el comando de voz no fue reconocido con éxito.

S/N: Realizaron la mitad del punto establecido.

Resultados:

Si se observa la siguiente tabla:

<b>Estadística General</b>				
<b>Usuario</b>	<b>Sexo</b>	<b>Edad</b>	<b>Bien (%)</b>	<b>Mal (%)</b>
limp	M	21	71,50	28,50
lufe	M	21	88,50	11,50
vicm	M	22	76,50	23,50
cahe	M	22	74,50	25,50
john	M	23	83,50	16,50
pave	M	22	83,50	16,50
alex	M	24	77,50	22,50
dein	F	22	78,50	21,50
yuri	F	23	83,50	16,50
sami	F	29	88,00	12,00
Total:			80,55	19,45

Tabla de datos primera prueba.

Las tablas anteriores muestran los resultados de las dos pruebas aplicadas, como se puede observar, los usuarios que alcanzaron un porcentaje superior 80.5% de la primera prueba tuvieron éxito en completar 5 de 6 puntos (83.3%), y solo tres usuarios lograron el 100% de la prueba con un 83.5% como el mínimo y un máximo de 88.5%.

Para medir la aceptación del usuario sobre IcVoz se tomo en cuenta la experiencia al manejar esta aplicación, los usuarios que tuvieron un rendimiento bajo en la primera prueba mostraron una actitud negativa hacia la aplicación, lo contrario paso con los que obtuvieron un porcentaje alto en la primera prueba.

A continuación se muestran los resultados individuales de la primera prueba.

Estadísticas					
Usuario:	limp	Edad:	21	Sexo: M	
Comando	Tecla (s)	Bien	Mal	Total	Bien %
Enter	Enter	9	1	10	90
Arriba	FlechaArriba	6	4	10	60
Abajo	FlechaAbajo	9	1	10	90
Izquierda	Flecha Izq.	10	0	10	100
Derecha	Flecha Der.	4	6	10	40
Copiar.	Ctrl+C	2	8	10	20
Cortar.	Ctrl+X	8	2	10	80
Pegar.	Ctrl+V	5	5	10	50
Deshacer.	Ctrl+Z	9	1	10	90
Menú	Alt	10	0	10	100
Salir	Esc	2	8	10	20
Inicio	Win	10	0	10	100
Alternar	Alt+Tab	9	1	10	90
Explorador	Win+E	9	1	10	90
Cerrar	Alt+F4	3	7	10	30
Guardar	Crtl+G	8	2	10	80
Borrar	Supr	5	5	10	50
Seleccionar	Crtl+A	9	1	10	90
Escritorio	Win+D	6	4	10	60
Ayuda	F1	10	0	10	100
<hr/>					
Total		143	57	200	72
<hr/>					
Porcentaje de Reconocimiento					71,50
<hr/>					
Porcentaje de Mal Reconocimiento					28,50
<hr/>					
Total					100,00

<b>Estadísticas</b>					
<b>Usuario:</b>	<b>lufe</b>	<b>Edad:</b>	<b>21</b>	<b>Sexo: M</b>	
Comando	Tecla (s)	Bien	Mal	Total	Bien %
Enter	Enter	9	1	10	90
Arriba	FlechaArriba	10	0	10	100
Abajo	FlechaAbajo	10	0	10	100
Izquierda	Flecha Izq.	10	0	10	100
Derecha	Flecha Der.	8	2	10	80
Copiar.	Ctrl+C	10	0	10	100
Cortar.	Ctrl+X	10	0	10	100
Pegar.	Ctrl+V	8	2	10	80
Deshacer.	Ctrl+Z	7	3	10	70
Menú	Alt	10	0	10	100
Salir	Esc	7	3	10	70
Inicio	Win	10	0	10	100
Alternar	Alt+Tab	9	1	10	90
Explorador	Win+E	4	6	10	40
Cerrar	Alt+F4	9	1	10	90
Guardar	Ctrl+G	8	2	10	80
Borrar	Supr	9	1	10	90
Seleccionar	Ctrl+A	9	1	10	90
Escritorio	Win+D	10	0	10	100
Ayuda	F1	10	0	10	100
Total		177	23	200	89
Porcentaje de Reconocimiento					88,50
Porcentaje de Mal Reconocimiento					11,50
Total					100,00

<b>Estadísticas</b>					
<b>Usuario:</b>	<b>vicm</b>	<b>Edad:</b> 22		<b>Sexo: M</b>	
Comando	Tecla (s)	Bien	Mal	Total	Bien %
Enter	Enter	7	3	10	70
Arriba	FlechaArriba	7	3	10	70
Abajo	FlechaAbajo	7	3	10	70
Izquierda	Flecha Izq.	10	0	10	100
Derecha	Flecha Der.	6	4	10	60
Copiar.	Ctrl+C	9	1	10	90
Cortar.	Ctrl+X	7	3	10	70
Pegar.	Ctrl+V	7	3	10	70
Deshacer.	Ctrl+Z	8	2	10	80
Menú	Alt	8	2	10	80
Salir	Esc	6	4	10	60
Inicio	Win	9	1	10	90
Alternar	Alt+Tab	9	1	10	90
Explorador	Win+E	7	3	10	70
Cerrar	Alt+F4	5	5	10	50
Guardar	Ctrl+G	5	5	10	50
Borrar	Supr	7	3	10	70
Seleccionar	Ctrl+A	10	0	10	100
Escritorio	Win+D	9	1	10	90
Ayuda	F1	10	0	10	100
Total		153	47	200	77
Porcentaje de Reconocimiento					76,50
Porcentaje de Mal Reconocimiento					23,50
Total					100,00

<b>Estadísticas</b>					
<b>Usuario:</b>	<b>cahe</b>	<b>Edad:</b>	<b>22</b>	<b>Sexo: M</b>	
Comando	Tecla (s)	Bien	Mal	Total	Bien %
Enter	Enter	7	3	10	70
Arriba	FlechaArriba	7	3	10	70
Abajo	FlechaAbajo	8	2	10	80
Izquierda	Flecha Izq.	6	4	10	60
Derecha	Flecha Der.	6	4	10	60
Copiar.	Ctrl+C	9	1	10	90
Cortar.	Ctrl+X	6	4	10	60
Pegar.	Ctrl+V	8	2	10	80
Deshacer.	Ctrl+Z	1	9	10	10
Menú	Alt	8	2	10	80
Salir	Esc	6	4	10	60
Inicio	Win	9	1	10	90
Alternar	Alt+Tab	7	3	10	70
Explorador	Win+E	9	1	10	90
Cerrar	Alt+F4	6	4	10	60
Guardar	Ctrl+G	10	0	10	100
Borrar	Supr	9	1	10	90
Seleccionar	Ctrl+A	9	1	10	90
Escritorio	Win+D	8	2	10	80
Ayuda	F1	10	0	10	100
Total		149	51	200	75
Porcentaje de Reconocimiento					74,50
Porcentaje de Mal Reconocimiento					25,50
Total					100,00

<b>Estadísticas</b>					
<b>Usuario:</b>	john	<b>Edad:</b>	23	<b>Sexo: M</b>	
Comando	Tecla (s)	Bien	Mal	Total	Bien %
Enter	Enter	9	1	10	90
Arriba	FlechaArriba	9	1	10	90
Abajo	FlechaAbajo	8	2	10	80
Izquierda	Flecha Izq.	10	0	10	100
Derecha	Flecha Der.	9	1	10	90
Copiar.	Ctrl+C	7	3	10	70
Cortar.	Ctrl+X	7	3	10	70
Pegar.	Ctrl+V	5	5	10	50
Deshacer.	Ctrl+Z	9	1	10	90
Menú	Alt	10	0	10	100
Salir	Esc	3	7	10	30
Inicio	Win	8	2	10	80
Alternar	Alt+Tab	10	0	10	100
Explorador	Win+E	9	1	10	90
Cerrar	Alt+F4	9	1	10	90
Guardar	Ctrl+G	6	4	10	60
Borrar	Supr	10	0	10	100
Seleccionar	Ctrl+A	9	1	10	90
Escritorio	Win+D	10	0	10	100
Ayuda	F1	10	0	10	100
Total		167	33	200	84
Porcentaje de Reconocimiento					83,50
Porcentaje de Mal Reconocimiento					16,50
Total					100,00

<b>Estadísticas</b>					
<b>Usuario:</b>	pave	<b>Edad:</b>	<b>22</b>	<b>Sexo: M</b>	
Comando	Tecla (s)	Bien	Mal	Total	Bien %
Enter	Enter	8	2	10	80
Arriba	FlechaArriba	7	3	10	70
Abajo	FlechaAbajo	10	0	10	100
Izquierda	Flecha Izq.	10	0	10	100
Derecha	Flecha Der.	7	3	10	70
Copiar.	Ctrl+C	9	1	10	90
Cortar.	Ctrl+X	10	0	10	100
Pegar.	Ctrl+V	6	4	10	60
Deshacer.	Ctrl+Z	6	4	10	60
Menú	Alt	8	2	10	80
Salir	Esc	5	5	10	50
Inicio	Win	10	0	10	100
Alternar	Alt+Tab	9	1	10	90
Explorador	Win+E	6	4	10	60
Cerrar	Alt+F4	10	0	10	100
Guardar	Ctrl+G	10	0	10	100
Borrar	Supr	10	0	10	100
Seleccionar	Ctrl+A	10	0	10	100
Escritorio	Win+D	10	0	10	100
Ayuda	F1	6	4	10	60
Total		167	33	200	84
Porcentaje de Reconocimiento					83,50
Porcentaje de Mal Reconocimiento					16,50
Total					100,00

Estadísticas					
Usuario:	alex	Edad:	24	Sexo: M	
Comando	Tecla (s)	Bien	Mal	Total	Bien %
Enter	Enter	8	2	10	80
Arriba	FlechaArriba	9	1	10	90
Abajo	FlechaAbajo	10	0	10	100
Izquierda	Flecha Izq.	10	0	10	100
Derecha	Flecha Der.	8	2	10	80
Copiar.	Ctrl+C	9	1	10	90
Cortar.	Ctrl+X	7	3	10	70
Pegar.	Ctrl+V	10	0	10	100
Deshacer.	Ctrl+Z	7	3	10	70
Menú	Alt	9	1	10	90
Salir	Esc	6	4	10	60
Inicio	Win	10	0	10	100
Alternar	Alt+Tab	6	4	10	60
Explorador	Win+E	3	7	10	30
Cerrar	Alt+F4	6	4	10	60
Guardar	Ctrl+G	1	9	10	10
Borrar	Supr	10	0	10	100
Seleccionar	Ctrl+A	7	3	10	70
Escritorio	Win+D	9	1	10	90
Ayuda	F1	10	0	10	100
Total		155	45	200	78
Porcentaje de Reconocimiento					77,50
Porcentaje de Mal Reconocimiento					22,50
Total					100,00

<b>Estadísticas</b>					
<b>Usuario:</b>	<b>dein</b>	<b>Edad:</b>	<b>22</b>	<b>Sexo: F</b>	
Comando	Tecla (s)	Bien	Mal	Total	Bien %
Enter	Enter	4	6	10	40
Arriba	FlechaArriba	9	1	10	90
Abajo	FlechaAbajo	7	3	10	70
Izquierda	Flecha Izq.	5	5	10	50
Derecha	Flecha Der.	6	4	10	60
Copiar.	Ctrl+C	7	3	10	70
Cortar.	Ctrl+X	7	3	10	70
Pegar.	Ctrl+V	9	1	10	90
Deshacer.	Ctrl+Z	8	2	10	80
Menú	Alt	10	0	10	100
Salir	Esc	8	2	10	80
Inicio	Win	7	3	10	70
Alternar	Alt+Tab	7	3	10	70
Explorador	Win+E	9	1	10	90
Cerrar	Alt+F4	9	1	10	90
Guardar	Ctrl+G	8	2	10	80
Borrar	Supr	9	1	10	90
Seleccionar	Ctrl+A	9	1	10	90
Escritorio	Win+D	9	1	10	90
Ayuda	F1	10	0	10	100
Total		157	43	200	79
Porcentaje de Reconocimiento					78,50
Porcentaje de Mal Reconocimiento					21,50
Total					100,00

Estadísticas					
Usuario:	yuri	Edad:	23	Sexo: F	
Comando	Tecla (s)	Bien	Mal	Total	Bien %
Enter	Enter	8	2	10	80
Arriba	FlechaArriba	6	4	10	60
Abajo	FlechaAbajo	9	1	10	90
Izquierda	Flecha Izq.	10	0	10	100
Derecha	Flecha Der.	6	4	10	60
Copiar.	Ctrl+C	8	2	10	80
Cortar.	Ctrl+X	10	0	10	100
Pegar.	Ctrl+V	7	3	10	70
Deshacer.	Ctrl+Z	9	1	10	90
Menú	Alt	10	0	10	100
Salir	Esc	10	0	10	100
Inicio	Win	10	0	10	100
Alternar	Alt+Tab	9	1	10	90
Explorador	Win+E	5	5	10	50
Cerrar	Alt+F4	7	3	10	70
Guardar	Ctrl+G	5	5	10	50
Borrar	Supr	10	0	10	100
Seleccionar	Ctrl+A	9	1	10	90
Escritorio	Win+D	10	0	10	100
Ayuda	F1	9	1	10	90
Total		167	33	200	84
Porcentaje de Reconocimiento					83,50
Porcentaje de Mal Reconocimiento					16,50
Total					100,00

<b>Estadísticas</b>					
<b>Usuario:</b>	sami	<b>Edad:</b>	<b>29</b>	<b>Sexo: F</b>	
Comando	Tecla (s)	Bien	Mal	Total	Bien %
Enter	Enter	10	0	10	100
Arriba	FlechaArriba	7	3	10	70
Abajo	FlechaAbajo	7	3	10	70
Izquierda	Flecha Izq.	10	0	10	100
Derecha	Flecha Der.	10	0	10	100
Copiar.	Ctrl+C	9	1	10	90
Cortar.	Ctrl+X	10	0	10	100
Pegar.	Ctrl+V	9	1	10	90
Deshacer.	Ctrl+Z	7	3	10	70
Menú	Alt	10	0	10	100
Salir	Esc	10	0	10	100
Inicio	Win	10	0	10	100
Alternar	Alt+Tab	8	2	10	80
Explorador	Win+E	9	1	10	90
Cerrar	Alt+F4	7	3	10	70
Guardar	Ctrl+G	8	2	10	80
Borrar	Supr	7	3	10	70
Seleccionar	Ctrl+A	8	2	10	80
Escritorio	Win+D	10	0	10	100
Ayuda	F1	10	0	10	100
Total		176	24	200	88
Porcentaje de Reconocimiento					88,00
Porcentaje de Mal Reconocimiento					12,00
Total					100,00