Capítulo 3: Diseño y Construcción de la Aplicación

3.1 Diseño del sistema

En este capítulo se presenta el proceso de diseño y construcción que se realizó para crear la aplicación Conquiro-Vox. El cual se desarrolló teniendo en cuenta que personas invidentes, personas con visión reducida o sin discapacidades harán uso de esta aplicación.

Para ayudar a usuarios invidentes Conquiro-Vox cuenta con una interfaz capaz de reproducir sonidos y leer en voz alta cualquier menú que se le presente al usuario, esto gracias a los paquetes del CSLU Toolkit. Los sonidos son reproducidos para avisar al usuario sobre las diferentes acciones que está realizando la aplicación, por ejemplo: al abrirse una nueva ventana, cuando sea el momento de dictarle la consulta o para avisar que ha ocurrido un error en la ejecución. La función principal de esta ayuda auditiva es informar al usuario invidente que es lo que está pasando en todo momento referente a la aplicación.

Los usuarios con videntes o visión reducida, cuentan con una interfaz grafica que contiene botones, menús emergentes y ventanas, los cuales pueden ser utilizados por medio del "ratón" a través de la pantalla, si el usuario así lo desea. Así la interfaz gráfica permite que la aplicación sea utilizada, sin importar la existencia o no de capacidades diferentes físicas del usuario.

3.2 Descripción del software

Una vez terminado el análisis de la aplicación el siguiente paso fue codificar la interfaz del software paneles, botones, menús y un selector de archivos, consciente en todo momento que esta aplicación seria usada por invidentes como personas con visión reducida. Debido a esto la interfaz se construyó de tal manera que todos los botones y

menús que se presentan sean fácilmente utilizados por medio del teclado haciendo uso del método “setMnemonic()” que permite asignar a cada elemento de la interfaz un acceso directo con tan solo presionar la tecla “Alt” y la primer letra con la que comienza el nombre del botón o menú. De esta forma se facilita a los usuarios invidentes la interacción con la interfaz. Este proceso dio como resultado la clase Interfaz la cual como su nombre lo dice presenta la interfaz grafica de la aplicación por medio de la cual el usuario interactúa con todos los elementos de manera transparente. Cabe mencionar que esta clase es la principal pues además de realizar las tareas antes mencionadas es la que contiene el método main().

El siguiente elemento que se codificó de esta aplicación fue la parte referente con la grabación de sonido lo que dio como resultado la clase Record. Un aspecto importante de esta clase es que a diferencia de las demás fue tomada de la página web titulada Java Sound API de Sun Microsystems, ya que es parte de los ejemplos base del API Sound de Java, su nombre original es CapturePlayback.java y fue desarrollada por Sun Microsystems en 1999. [Sun Microsystems, 2007]. Como se menciono anteriormente, fue creada a partir de la clase CapturePlayback, la cual fue modificada para obtener solamente las funciones de grabación de sonido, entre las modificaciones que sufrió se encuentran, la eliminación de la interfaz grafica, se retiró la clase PlayBack que estaba contenida dentro de la misma la cual permitía reproducir lo que se había grabado, de igual manera se quitó la clase SamplingGraph la cual se encargaba de dibujar en la interfaz grafica las ondas de sonido, se modificó y se extrajo la clase FormatControls esto con el objetivo de hacerla más simple e identificar fácilmente el lugar donde se modifican las características de la grabación, para hacer esto fue necesario entender completamente el código para que después de eliminar dichas funciones la clase siguiera funcionando de manera normal. Además se agregaron nuevas opciones para ajustarla a las necesidades de Conquiro-Vox, como una nueva interfaz grafica que permita que la clase Interfaz controle la función de grabado, ahora se escucha un sonido el cual indica que el sistema está grabando la consulta del usuario, lo que permite que el usuario este enterado sin necesidad de ver la pantalla, una opción que permite repetir el proceso de grabación, en el caso de que la grabación anterior este contaminada con ruido excesivo o el usuario no esté convencido de la claridad de su consulta.

El sintetizador de voz que se seleccionó al igual que el reconocedor de voz están programados en Tcl. Fue necesario tomar varios tutoriales de Tcl para crear el código que llama al sintetizador y el reconocedor. Para correr el sintetizador se creó un nuevo script en Tcl llamado tts.tcl, (Figura 11), el cual convierte a sonido cualquier texto que le es enviado, para esto hace uso de los paquetes Audio y TTS del CSLU Toolkit.

Para llamar al script tts.tcl, se construyó la clase TTSJ la cual recibe como argumentos el String, que contiene el texto que se desea leer, y el archivo que se quiere modificar, en este caso es el archivo tts.tcl. Lo primero que hace es editar de manera dinámica el archivo tts.tcl para que se actualice constantemente la información que será leída en voz alta, cada vez que se actualiza el script se edita la línea numero 10, dicha información está contenida en el String que recibe la clase. De esta forma se puede leer el contenido de cualquier ventana que sea mostrada. La segunda función se encarga de invocar al script tts.tcl, esto lo realiza haciendo uso de las librerías del Jacl 1.4.0 las cuales permiten introducir comandos de Tcl en el código de Java haciendo uso del método "eval". El comando de Tcl usado en toda la aplicación para invocar scripts es el "exec" el cual nos permite correr el script tts.tcl en la terminal tclsh80.exe dando como resultado la lectura en voz alta de el texto deseado. De la misma manera esta clase está diseñada para llamar a otro script que lleva por nombre sound.tcl, este se encarga de reproducir los sonidos de advertencia que posee la interfaz, los cuales son tres ding.wav, que indica al usuario que el sistema está listo para grabar su consulta, win.wav, el cual señala que una ventana se ha mostrado, y error.wav que informa al usuario que un error ha ocurrido durante la ejecución del programa. Para esto la clase al ser llamada debe de tener en sus argumentos el nombre del archivo que se va a reproducir y el script que se va a llamar en este caso el sound.tcl.

```
1  package require Audio
2  package require TTS
3
4  Audio create sound
5  sound attach "Device"
6
7  TTS create fest
8  fest setVoice abc
9  fest attach "Device"
10 set text {Menu principal...}
11 foreach {w t} [fest tts $text] break
12 sound play $w
13 nuke $w
```

Figura 11. Código del script tts.tcl

Sin duda alguna lo más difícil al momento de construir esta aplicación fue controlar el reconocedor desde la clase Interfaz, porque el reconocedor está programado en Tcl. Después de un prolongado tiempo de investigación y realización de pruebas, se logró hacer funcionar esta herramienta desde Java, gracias a esto se creó el script llamado recog.tcl (Figura 12). Para correr este script desde Java se creó la clase Recog esta realiza las funciones de llamar al reconocedor y procesar los resultados arrojados por el reconocedor para convertirlos a un formato entendible. Una vez que se corre el script este utiliza el archivo word2pho.vocab y le da formato para que pueda ser utilizado por el reconocedor enseguida procesa la información contenida en el archivo de audio Res1.wav, que contiene la consulta del usuario, para compararla con las palabras del vocabulario, de esta forma obtiene las cuatro mejores respuestas. Para obtener las respuestas se usa el comando "catch" el cual guarda los resultados en la variable "squa", en este punto la variable "squa" contiene los resultados arrojados por el reconocedor pero estos aun están en Tcl. Para que Java los obtenga es necesario ejecutar la siguiente instrucción "Squalo = i.getResult().toString()", "i.getResult()" es un método de la librería de Jacl el cual obtiene la última acción realizada dentro de la terminal tclsh80.exe, el resultado obtenido tiene que ser convertido a "String" y es asignado a "Squalo" que es un "String" previamente declarado en el código de la clase Recog. En este punto Recog ya posee los resultados, pero estos están en un formato no entendible es por eso que se les aplican varias operaciones para obtener solo las cuatro palabras reconocidas, estas son guardadas en un arreglo de "Strings" que lleva el nombre de "datf" por ultimo este arreglo es enviado a Interfaz para que muestre los resultados al usuario.

```
set c1 "{"
set c2 "}"
set c3 {"}
set vocab_file [open word2pho.vocab r]
array set vocab [read $vocab_file]
foreach {vo cab} [array get vocab] {
   set v "{$c3$vo$c3 $c1$cab$c2 }"
   append list " " $v
}
package require Genrecog
genrecog initialize gen1
set w [wave read Res1.wav]
genrecog tree gen1 search $list
genrecog pipe gen1 search $w
puts [genrecog result gen1 search]
genrecog nuke gen1
nuke $w
```

Figura 12. Código del script genrecong.tcl.

Los problemas que se presentaron en el desarrollo de esta clase fueron: lograr que desde java corra el código en Tcl, entender el funcionamiento del reconocedor, conocer el formato que debería de llevar el vocabulario y hacer que todo corriera de manera coordinada. Lo que dificultó enormemente esta tarea fue la falta de información sobre la construcción, funcionamiento y uso del reconocedor.

Para la administración del vocabulario se creó la clase Trans que realiza la función de agregar la pronunciación a cualquier palabra que se encuentre en el formato debido, de esta forma a partir de una lista de palabras es como se crea un vocabulario. Para realizar este proceso, se diseñó un algoritmo recursivo el cual primero obtiene la información del archivo abcdario.dat, este contiene la pronunciación de cada letra del abecedario así como algunas otras, una vez que esta información es obtenida, cada "String" que es enviado a la clase Trans es dividido hasta que se obtiene cada letra que conforma la palabra, como ejemplo, si entra un "String" con la palabra historia este proceso nos daría como resultado h i s t o r i a, una vez que se tiene la palabra dividida en letras se le asigna la pronunciación a cada una en base al archivo abcdario.dat. De esta forma es como se obtiene la pronunciación de cada palabra, la pronunciación de historia es (i s t\[c t\[o r i a), los resultados son guardados en el archivo word2pho.vocab el cual puede ser usado en cualquier momento por el reconocedor. Esta

clase ofrece la posibilidad de crear un nuevo vocabulario, para esto hace un respaldo del archivo existente word2pho.vocab creando una lista de palabras en el archivo oldwors.txt, después crea un nuevo vocabulario pero ahora con la información contenida en la nueva lista escogida por el usuario. La otra opción es agregar al vocabulario, esta como su nombre lo indica abre el archivo word2pho.vocab existente y le agrega la información de la nueva lista de palabras. En resumen esta clase se encarga de administrar todo lo referente al vocabulario de la aplicación. El problema que se genera en esta clase es que el uso de algoritmos recursivos vuelve lento el proceso cuando se trabaja con listas de palabras que contiene más de 10’000 términos, debido a que son varias las operaciones que se utilizan, pues se tiene que comparar letra por letra para asignar la pronunciación, además de revisar que se encuentren en el formato correcto así como realizar un archivo de respaldo del vocabulario remplazado.

Por último se realizaron modificaciones a la aplicación Linter-Vox para integrarla a Conquiro-Vox. Estas modificaciones en un principio causaron muchos problemas debido a que Linter-Vox tenía su propio método main() y al eliminarlo dejaba de funcionar la aplicación. Se tuvo que desarrollar una estrategia que permita que la clase Interfaz controle y arranque esta aplicación. Linter-Vox se utiliza como lector de resultados, para esto se le eliminó la función de recibir la consulta desde el teclado, pues ahora la recibe directamente de los resultados obtenidos por el reconocedor de voz. Una vez que tiene la consulta, se conecta al motor de búsqueda Google. Para llevar a cabo esta conexión es necesario, agregar el archivo googleapi.jar a la librería de Java y registrarse para obtener una autentificación la cual permite realizar hasta 1000 búsquedas por día. Dentro del código debe de contener como variable el número de autentificación, numero de resultados deseados, idioma en que se desean los resultados y la consulta. Se crea un objeto de la clase GoogleSearch y se le asignan estas variables. Usando la clase GoogleSearchResult se obtiene los resultados, los cuales son leídos en voz alta gracias al uso de los agentes de Microsoft. Otra de las características principales de Linter-Vox es que permite navegar entre las ligas disponibles de los resultados.

Una vez que todas estas funciones trabajan en conjunto dan como resultado la aplicación Conquiro-Vox. La cual tiene como características principales:

- Uso de herramientas de reconocimiento de voz.
- Utilización de funciones de síntesis de voz.
- Uso de los agentes de Microsoft para la lectura de resultados.
- Conexión al motor de búsqueda Google.
- Uso de algoritmos recursivos para la administración y transformación de vocabularios.
- Interfaz enfocada a interactuar con usuarios invidentes o con visión reducida.
- Aplicación gratuita y basada en el idioma español mexicano.

3.3 Definición del vocabulario para el sistema

El vocabulario es la base sobre la cual el reconocedor de voz obtiene los resultados. Por eso es tan importante. Si una palabra no existe dentro del vocabulario esta jamás podrá ser reconocida. Aunque esto no significa que lo mejor para un vocabulario es que contenga el mayor numero de palabras posibles. Al aumentar el número de palabras dentro del vocabulario el reconocedor de voz pierde precisión, aumenta la posibilidad de encontrar palabras que suene de manera similar, lo cual provoca que el reconocedor se confunda entre palabras que tengan una pronunciación similar.

Debido a eso la aplicación permite al usuario, en el caso que la consulta dictada no sea encontrada, agregar palabras al vocabulario por medio del teclado. Sin embargo, es posible que con el uso repetido de la aplicación el vocabulario crezca tanto hasta llegar al punto donde el reconocedor pierda precisión en sus resultados.

La estrategia que se ha seguido para contener este problema, se basa en la creación de diferentes vocabularios, cada uno de ellos relacionado a un contexto diferente. De esta forma al tener varios vocabularios crecen a un ritmo menor que al tener uno solo. La aplicación permite que en cualquier momento sea cargada una nueva lista de palabras la cual será convertida en un nuevo vocabulario de forma automática, de esta manera si el usuario decide hacer búsquedas basadas en algún contexto en especifico puede cambiar al vocabulario que más le convenga, así se mantiene la

precisión del reconocedor. Es importante mencionar que el uso de la función para el cambio de vocabulario debe de ser ejecutado por el administrador del sistema. Además, este debe de darle un mantenimiento constante, para evitar que un vocabulario contenga palabras que estén fuera del contexto, o sean indeseadas. Así como la creación de nuevas listas de palabras basadas en las necesidades de los usuarios.

Las listas de palabras deben de seguir el siguiente formato, por ningún motivo las palabras deben de contener números, signos o acentos, cada término debe de estar en una línea, si un término consta de dos o más palabras están deben de estar separadas por un espacio, es permitido el uso de letras mayúsculas y minúsculas, así como la letra ñ y ü. Ver ejemplo en la figura 13.

Preclasico
olmeca
zapotecas
Teotihuacan
mayas
posclasico
toltecas
TENOCHTITLAN
Cristobal Colon
CONQUISTA

Figura 13. Fragmento de una lista de palabras referente al contexto "Historia de México".

Una vez procesada la lista de palabras el sistema la convierte a un vocabulario el cual será utilizado por el reconocedor de voz. La lista de la figura 13 queda de la siguiente manera una vez procesada (Figura 14).

preclasico {pc p r e kc k l a s i kc k o}
olmeca {o l m e kc k a}
zapotecas {s a pc p o t\[c t\[e kc k a s}
teotihuacan {t\[c t\[e o t\[c t\[i u a kc k a n}
mayas {m a j a s}
posclasico {pc p o s kc k l a s i kc k o}
toltecas {t\[c t\[o l t\[c t\[e kc k a s}
tenochtitlan {t\[c t\[e n o tS t\[c t\[i t\[c t\[l a n}
cristobal_colon {kc k r i s t\[c t\[o bc b a l (.pau) kc k o l o n}
conquista {kc k o n kc k i s t\[c t\[a}

Figura 14. Vocabulario basado en la lista de palabras de la figura 13.

El sistema cuenta con tres listas de palabras enfocadas a contextos diferentes como son historia de México, finanzas y deportes. La lista de deportes es la más extensa hasta este momento cuenta con 246 términos, seguido de la de historia de México con 158 y por último la lista de deportes con 122. Estas listas aun son pequeñas pero con el uso incrementaran de manera notable.

3.4 Diagrama de clases

El diagrama de clase de esta aplicación muestra información acerca de los tipos de variables y métodos que fueron implementados durante el desarrollo y construcción del sistema. En la figura 15 se muestra el diagrama de clases de la aplicación Conquiro-Vox, el cual está conformado por 8 clases, todas estas coordinadas por la clase Interfaz. Este diagrama no muestra a detalle cada clase pues su finalidad es mostrar la relación que existe entre clases, más adelante se mostrará y explicará cada clase a detalle.

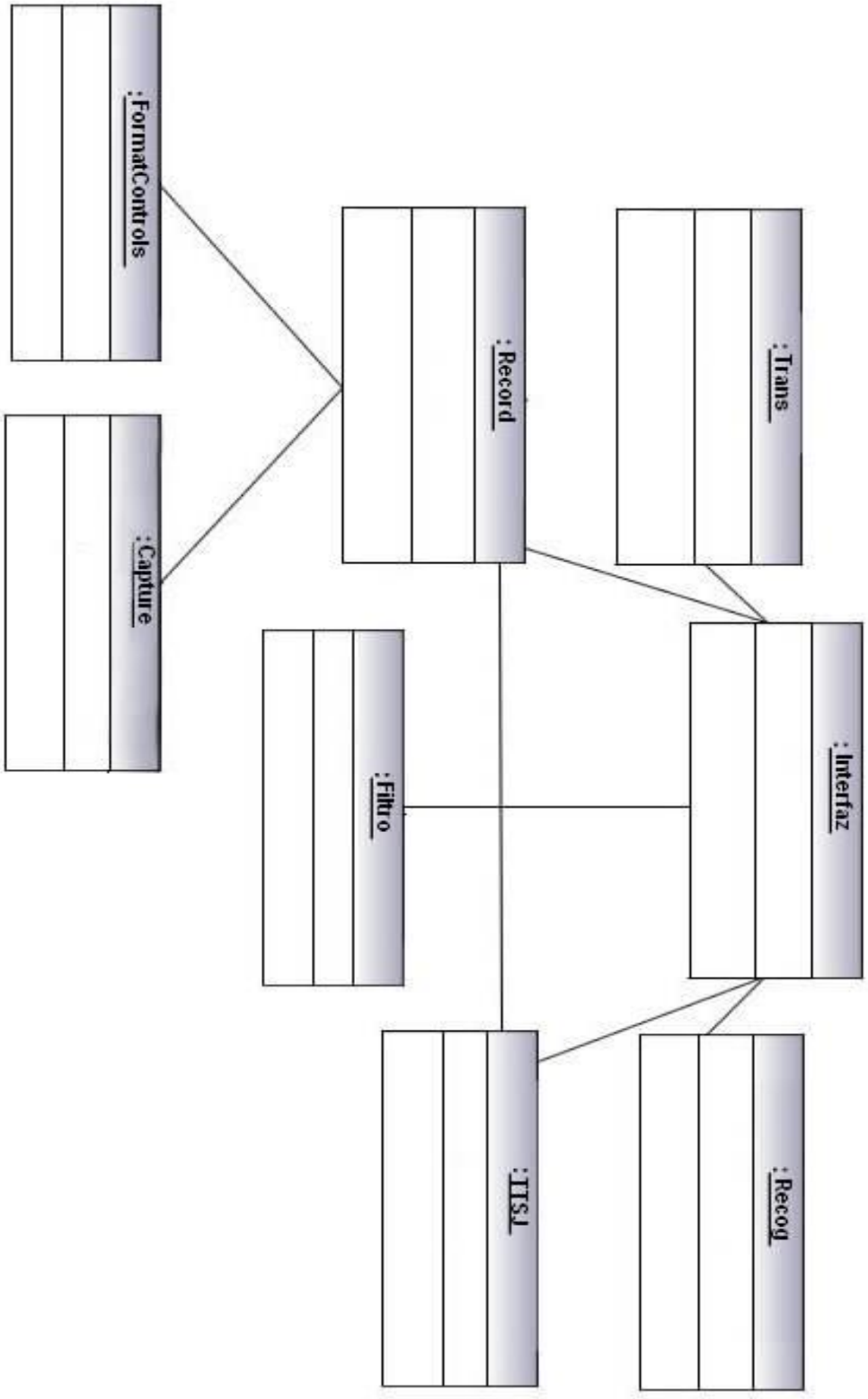

Figura 15. Diagrama de clases.

Las clases TTSJ y Record son las encargadas de controlar todo lo relacionado con el procesamiento del audio para la Interfaz, como su nombre lo dice, la clase Record cumple la función de grabar la consulta dictada por el usuario, por su parte la clase TTSJ (Figura 16), es la encargada de reproducir archivos de audio y leer en voz alta cualquier texto que se le envié, esto gracias al uso de los scripts realizados en Tcl los cuales son: tts.tcl y sound.tcl.

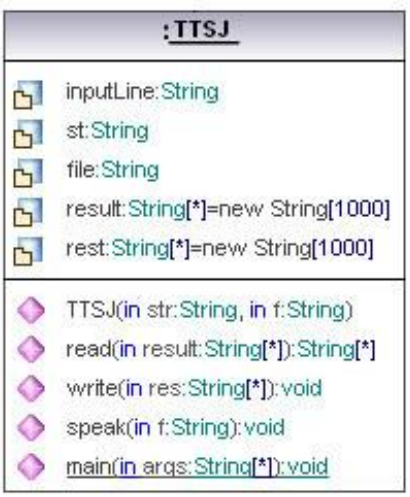

Figura 16. Clase TTSJ.

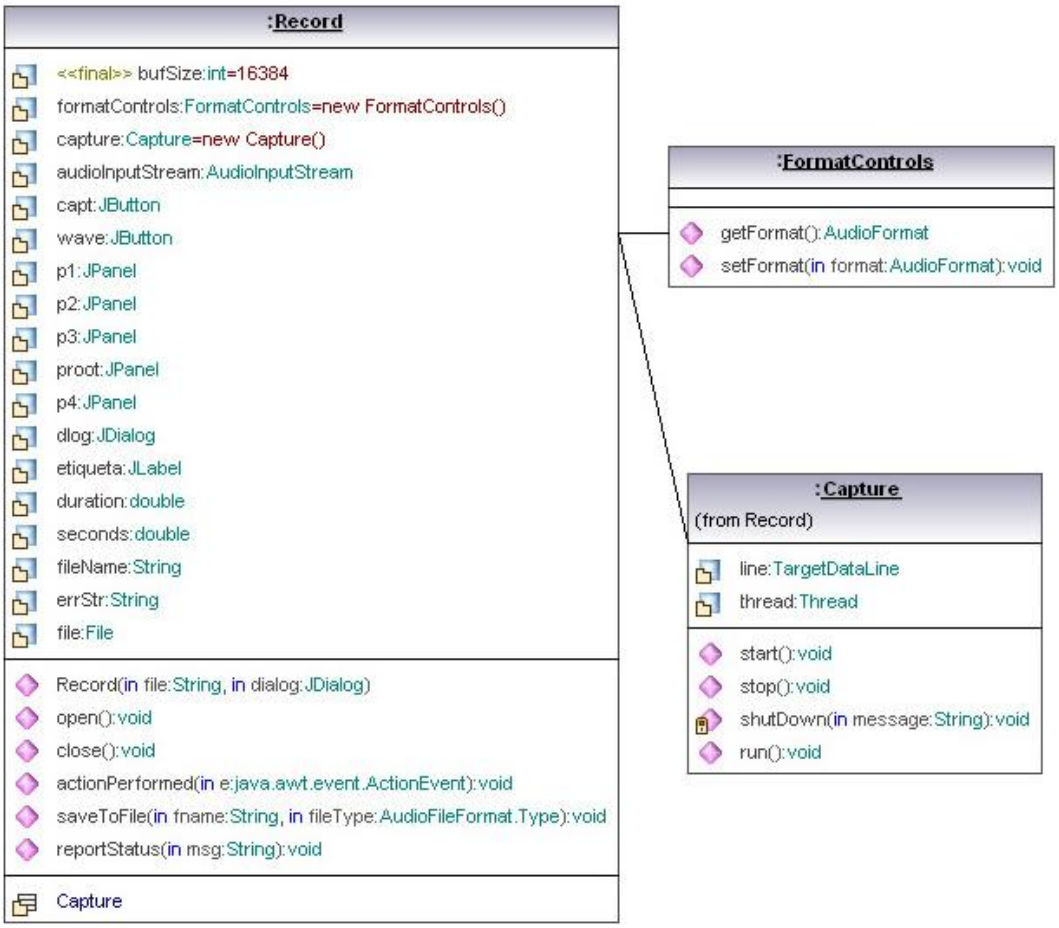

Figura 17. Clase Record.

La figura 17 hace referencia a la clase Record, que es la encargada de grabar la consulta del usuario. Esta es llamada desde la clase Interfaz, permite que el usuario dicte su consulta y esta sea grabada en un archivo WAV. Esta clase a su vez crea un objeto de la clase FormatControls, el cual contiene todas las características de la grabación, y otro objeto de la clase Capture, la cual se encarga de capturar el sonido y grabarlo como archivo de sonido.

Figura 18. Clase Filtro.

La clase Filtro se encarga de filtrar todos los tipos de archivos que se muestran en un selector de documentos, en este caso el filtro solo permite ver documentos con la extensión txt.

Figura 19. Clase Trans.

La clase Trans es la encargada de agregarle la pronunciación a cada palabra de esta forma es como se crea un vocabulario. Esta clase se ocupa de crear y administrar el vocabulario, gracias a su funcionamiento es posible cambiar el contexto de las consultas.

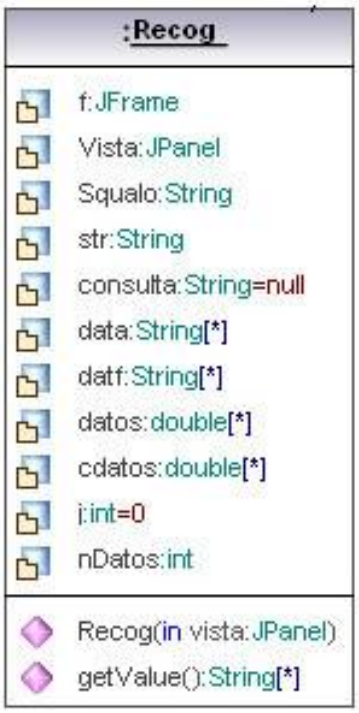

Figura 20. Clase Recog.

La clase Recog es la encargada de llamar al script recog.tcl y dar formato a los resultados obtenidos por el reconocedor. Controla el funcionamiento del reconocedor y procesa los resultados.

Figura 21 Clase Interfaz.

La clase Interfaz, que se muestra completa en la figura 21, es la que se encarga de coordinar todos los procesos de este sistema. Permite la interacción, de todos los elementos que conforman esta aplicación con el usuario de forma transparente. Esta clase es la que posee el método main(), que se encarga de ejecutar la aplicación,

contiene la interfaz grafica así como su especificación y funcionalidad de la misma. Mantiene una comunicación constante con el usuario durante toda la ejecución del sistema hasta su final.

3.5 Diagrama de secuencias

A continuación se muestra una serie de diagramas de secuencia que muestran el comportamiento de Conquiro-Vox durante su funcionamiento.

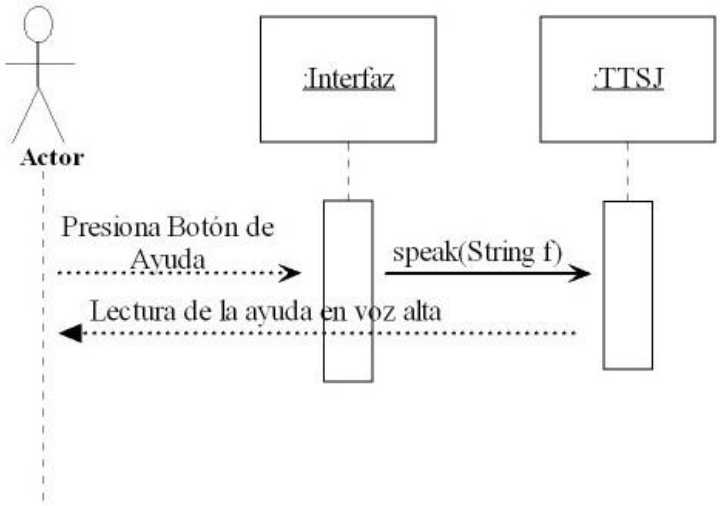

Figura 22. Diagrama de secuencias para la lectura de la ayuda del sistema.

La figura 22 muestra la secuencia para la lectura de la ayuda del sistema. Inicia cuando el usuario presiona el menú Ayuda del sistema lo que indica a la Interfaz que debe de llamar a la clase TTSJ la cual llama a un script desarrollado en Tcl, nombrado tts.tcl, que se encarga de reproducir en voz alta el texto que envía la Interfaz, como resultado el usuario escuchara todas las instrucciones para utilizar la interfaz, la forma de manipularla y el significado de cada sonido. Al terminar, la Interfaz regresa al punto de inicio (Figura 23).

Figura 23. Punto de inicio o menú principal de la aplicación.

El punto de inicio o menú principal de la Interfaz se muestra en la figura 23 desde aquí se pueden realizar todas las funciones permitidas por la interfaz, como son: cargar vocabulario, realizar una búsqueda en la Web, escuchar la ayuda del sistema, salir del sistema y escuchar todas las funciones mencionadas anteriormente.

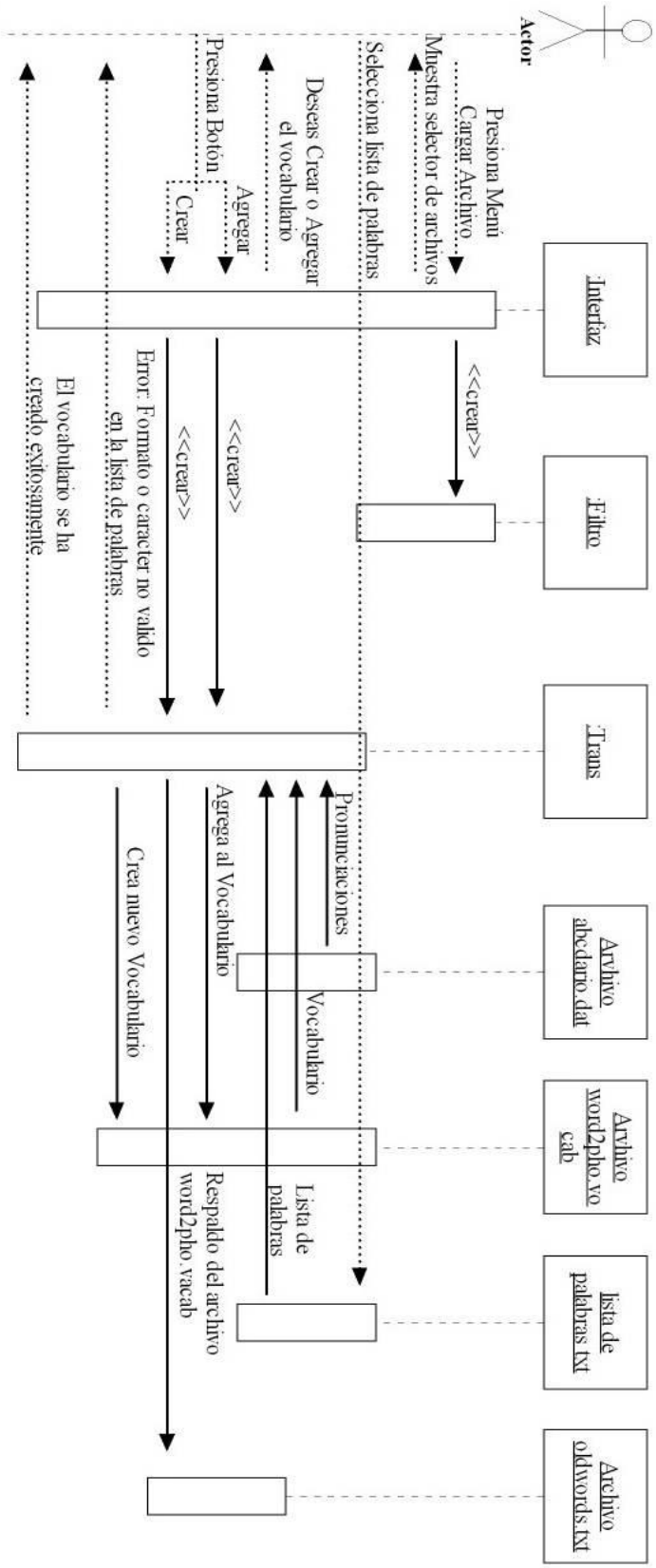

Figura 24. Diagrama de secuencias para la opción cargar archivo.

El diagrama de secuencias para la opción cargar archivo se muestra en la figura 24. Este comienza cuando el usuario presiona el menú Cargar archivo lo cual le indica a la interfaz que debe de mostrar un selector de archivos (Figura 25), para esto la interfaz llama la clase Filtro la cual crea un filtro encargado de mostrar solo los archivos con la extensión .txt contenidos en la carpeta vocabulario. El usuario selecciona el archivo deseado y la aplicación lo procesa para convertirlo en un vocabulario, después muestra una ventana donde pregunta al usuario si desea crear un nuevo vocabulario o agregar las nuevas palabras al vocabulario existente. Esto lo hace llamando a la clase Trans la cual para su funcionamiento necesita el archivo abcdario.dat que contiene la pronunciación de cada letra, este archivo es la base para obtener las pronunciaciones de cada palabra, lo cual da como resultado el archivo word2pho.vocab. Si el usuario escoge la opción crear nuevo vocabulario, la clase Trans crea una lista de palabras basándose en el archivo word2pho.vocab la cual es guardada en el archivo oldwords.txt, hace un respaldo del vocabulario, enseguida procesa la lista de palabras y crea un nuevo vocabulario remplazando el existente. En cambio si el usuario presiona la opción agregar, la clase Trans procesa la lista de palabras y los resultados los agrega al vocabulario. De estas dos formas es creado el archivo word2pho.vocab el cual es vital para que el reconocedor pueda funcionar, una vez terminado el proceso la Interfaz muestra al usuario una ventana informativa confirmando que el nuevo vocabulario ha sido procesado exitosamente y la Interfaz vuelve al punto de inicio mostrado en la figura 23. Si existiera algún problema en el procesamiento de la lista de palabras, la Interfaz muestra una ventana de error indicando que la lista contiene números, acentos, caracteres o símbolos inválidos lo que provoca que termine el proceso.

Figura 25. Selector de archivos.

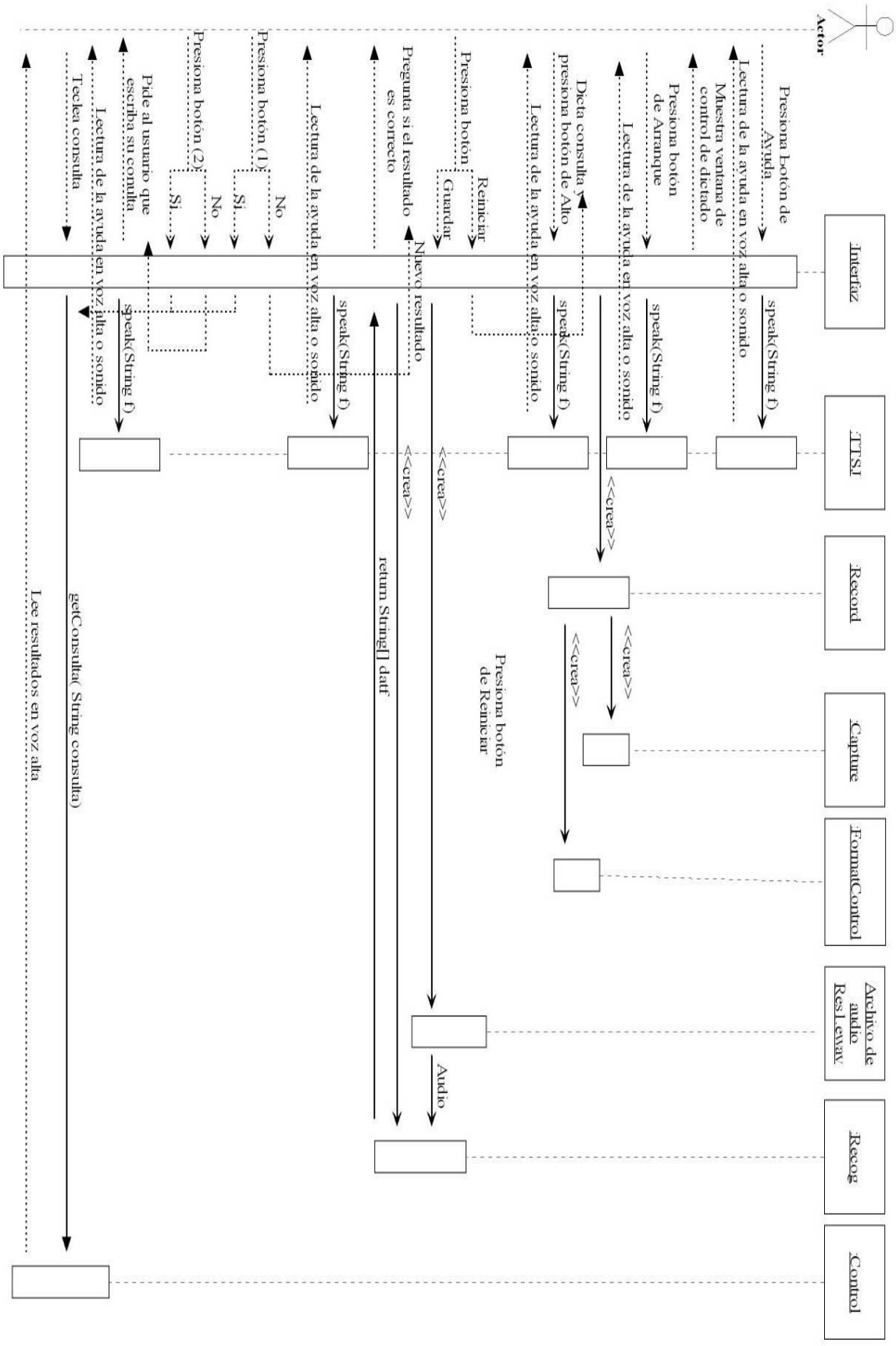

Figura 26. Diagrama de secuencias para la operación de búsqueda.

Por último se describe el diagrama de secuencia de la figura 26 el cual se refiere a la búsqueda. Una vez que el usuario presiona el botón de inicio la Interfaz hace una llamada a la clase TTSJ, para leer en voz alta toda la información y los sonidos de las ventanas que se presenten a lo largo de este proceso. Después la Interfaz muestra al usuario una nueva ventana (Figura 27, la cual llama a la clase Record que se encarga de controlar todo el proceso de dictado. El usuario presiona el botón arranque, nuevamente se llama a la clase TTSJ para que dicte las instrucciones y reproduce un sonido el cual le indica al usuario que es momento de dictar la consulta. Una vez que el usuario dicta su consulta, se le pregunta por medio de otra ventana si desea guardarla o reiniciar el proceso de dictado, si el usuario presiona este último el proceso de dictado se repetirá nuevamente, en cambio si presiona el botón de guardar, la consulta se almacena, lo que significa que se ha creado el archivo Res1.wav.

Figura 27. Ventana para el control del dictado.

El siguiente paso que realiza la Interfaz es llamar la clase "Recog" que a su vez llama a un script desarrollado en Tcl que lleva por nombre recog.tcl, la función de este script es habilitar el reconocedor el cual toma el archivo WAV y el vocabulario para procesarlos y obtener los resultados los cuales son enviados a la Interfaz. Una vez que llegan a Interfaz esta lanza una nueva ventana de confirmación (Figura 28), donde le pregunta al usuario si la palabra que se obtuvo como resultado es la que dictó, para esto se llama nuevamente a las clase TTSJ, si el usuario rechaza la primer confirmación la Interfaz toma el siguiente mejor resultado y se vuelve a mostrar una segunda ventana de confirmación al usuario, si este nuevamente la rechaza, la Interfaz le muestra una nueva ventana (Figura 29), la cual le pide que ingrese su consulta utilizando el teclado, para esto se necesita la clase TTSJ. Una vez aceptada cualquiera de las dos confirmaciones o ingresada la consulta por medio del teclado, Interfaz le envía la consulta a la clase

Control, que es parte de Linter-Vox, este se encarga de hacer la búsqueda en la Web y leer los resultados en voz alta, al terminar la clase Control con su funcionamiento regresa Conquiro-Vox a la posición de inicio.

Figura 28. Ventana para la confirmación de los resultados.

Figura 29. Ventana para el ingreso de consulta usando el teclado.

3.6 Manejo de errores y excepciones

El manejo de errores y excepciones fue un aspecto importante durante el desarrollo de esta aplicación, pues este proceso debería estar pensado tanto para personas videntes como invidentes. Comúnmente se muestra una ventana, con un símbolo llamativo generalmente en color rojo y una "X", cuando algún error ha sucedido en la aplicación, con esto se le informa al usuario el suceso que ha sucedido. Esto funciona para personas videntes, pero al trabajar con personas invidentes se tiene que cambiar la forma de hacer las cosas. Para esto se recurrió al uso de sonidos haciendo énfasis en un sonido de error, el cual se caracteriza por ser fuerte y que llama la atención, de esta forma cada vez que sucede un error este sonido es reproducido y después se explica en voz alta cual es el error que se cometió y como solucionarlo. A continuación se muestran los diferentes errores que pueden suceder durante la ejecución del sistema.

Al iniciar la aplicación la primera acción que realiza la Interfaz es comprobar que todos los archivos necesarios para su funcionamiento se encuentren en el directorio entre los cuales se encuentran los de audio, los scripts en Tcl y la carpeta que contiene las listas de palabras llamada vocabulario. Si alguno de ellos no se encuentra el sistema muestra la siguiente ventana (Figura 30).

Figura 30 Error inicio.

Al realizar la acción de cargar el vocabulario el usuario puede cometer dos errores, uno es que no exista la carpeta que lo contiene y muestre una ventana de error, ver figura 31, o al momento de procesar una lista de palabras para convertirla a vocabulario se pueden generar errores si esta contiene términos en un formato diferente al especificado, si esto sucede la Interfaz muestra el siguiente mensaje (Figura 32).

Figura 31 Error no existe carpeta vocabulario.

Figura 32 Error Vocabulario.

Otro de los errores a los cuales el usuario se puede enfrentar, se presenta al momento de realizar el proceso de búsqueda Conquiro-Vox verifica la existencia del archivo word2phone.vocab, el cual es necesario para el correcto funcionamiento del reconocedor, si este no existe lanza la siguiente ventana de error (Figura 33).

Figura 33 Error al iniciar proceso de búsqueda.

Por ultimo los siguientes errores pueden aparecer al momento en que el usuario escribe su consulta. El primero sucede si el usuario no escribe nada, trata de cerrar o cancelar la operación (Figura 34). El segundo ocurre cuando el usuario ingresa una consulta que contenga, números, acentos, símbolos o caracteres no validos, lo que provoca la aparición del siguiente mensaje ver figura 35. Cual quiera de estos errores regresan al usuario a la ventana teclear consulta (Figura 29).

Figura 34 Error consulta vacía o nula.

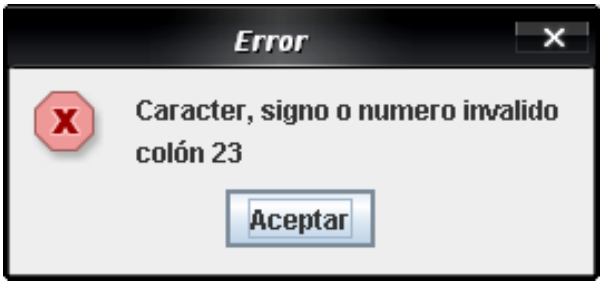

Figura 35 Error carácter no valido.

De esta forma la aplicación Conquiro-Vox recolecta todos los posibles errores durante su ejecución, los cuales son mostrados al usuario para que esté consciente del error que se ha provocado y como lo puede solucionar.

3.7 Comparación del sistema

Es necesario establecer el grado de calidad que ofrece el sistema Conquiro-Vox. Para esto es necesario compararlo con aplicaciones existentes. Aunque en la actualidad no se cuenta con una aplicación que sea equivalente a Conquiro-Vox para realizar comparaciones. Debido a esto lo que se comparó fueron los módulos de la aplicación, específicamente la interfaz, el sintetizador de voz y el lector de resultados. Jaws 9.0 fue la aplicación elegida para realizar las comparaciones puesto que también cuenta con un módulo de síntesis de voz, además permite una navegación en la Web parecida a la que se realiza con el módulo de Linter-Vox.

Jaws es una herramienta diseñada específicamente para ser usada por personas invidentes o con visión reducida entre sus características destacan las siguientes:

- Lee en voz alta todas las ventanas que aparecen en la pantalla.
- Permite cambiar el sintetizador de voz, al igual que la velocidad con que se realiza la lectura.
- Puede trabajar en diferentes idiomas como son español, español latino, inglés, francés, italiano entre otros.
- La aplicación queda residente en el sistema operativo y se puede activar en el momento que sea necesario, presionando una combinación de teclas.

- Al momento de escribir la aplicación nombra cada letra o signo que se es presionado.

En cuanto a la interfaz Conquiro-Vox permite las mismas facilidades a los usuarios al igual que Jaws 9.0. Pues la interacción con la aplicación se realiza por medio de combinaciones de teclas, aunque Jaws carece de sonidos específicos que informen al usuario que eventos están ocurriendo. En cambio Jaws posee la característica de pronunciar cada letra o signo que es tecleado, además una vez terminada la palabra la lee completa, esto con el fin de que el usuario este informado de las palabras que está escribiendo.

En cuanto al módulo de síntesis de voz Jaws es superior debido a que permite utilizar diferentes sintetizadores, algunos de ellos con diferentes voces. Es posible cambiar el idioma del sintetizador y la velocidad con la que se leen los resultados. Es importante destacar que la voz suena más humana que la utilizada por Conquiro-Vox. Jaws a diferencia de Conquiro-Vox es capaz de leer palabras acentuadas, asimismo es capaz de detener o reiniciar la lectura con solo presionar una tecla.

La aplicación Jaws no contiene en si un lector de resultados como Conquiro-Vox, en cambio lee todas las ventanas que aparecen, usando esta función se abrió la página electrónica de Google y se realizó una búsqueda, esto con el fin de realizar el mismo proceso que efectúa el lector de resultados de Conquiro-Vox. Jaws antes de que se cargue completamente la página avisa al usuario que por ciento lleva, una vez cargada la página comienza con la lectura de la misma, esta se va desplazando de manera automática ajustándose al punto que se está leyendo. Algo importante que realiza Jaws es que avisa si lo que está leyendo es un botón, una liga, una tabla etcétera, de esta manera el usuario esta informado en todo momento de los elementos que contiene la pagina. Otra de las características es que permite moverse entre párrafos sin que se detenga la lectura, además solo lee el texto y las ligas evitando leer el código HTML o algún otro. El problema que tiene Jaws es que algunas instrucciones solo están en inglés, entonces causa mucha confusión cuando se escucha que el sintetizador que está en español le las instrucciones escritas en inglés, esto por lo general sucede cuando aparece una nueva ventana. Por su parte Conquiro-Vox una vez que se ha realizado la

búsqueda se centra en leer solo las 10 primeras ligas y estas son enumeradas, de esta forma se evita mencionar información poco importante y facilita el seleccionar la liga deseada.

Un elemento importante que no integra la aplicación Jaws es el uso de una herramienta de reconocimiento de voz, en cambio el funcionamiento de Conquiro-Vox está basado en esta herramienta ya que al usar funciones de reconocimiento de voz se facilita significativamente el uso de la aplicación para personas invidentes o con visión reducida.

Como conclusión Conquiro-Vox es una aplicación de buena calidad, cumple con su objetivo principal. Aunque al ser comparada con Jaws es evidente que aun carece de características y funciones como son mejorar la calidad y control del sintetizador, permitir mayor control en cuanto a la lectura de resultados, hacer que la aplicación quede residente en el sistema operativo, entre otras, que no le permiten tener la misma calidad que ofrece Jaws. Aunque estas carencias no impiden que Conquiro-Vox cumpla la tarea para la que fue creada, es importante pensar en agregar estas características en un futuro cercano con el fin de hacer más atractiva esta aplicación. Esto hasta cierto punto es justificable pues el precio de una licencia de Jaws está alrededor de los $1095 dólares en cambio Conquiro-Vox es totalmente gratuita. La siguiente tabla resume las ideas presentadas (ver figura 36).

	Conquiro-Voz	**Jaws**
Calidad síntesis de voz	Suficiente	Bueno
Reconocedor de voz	Bueno	-No tiene-
Uso de la interfaz	Bueno	Bueno
Precio	Gratuito	$1098 Dólares

Figura 36. Resultados de la comparación de sistemas.

A lo largo de este capítulo se presentó el diseño y construcción de la aplicación. Se describió técnicamente el software desarrollado, haciendo especial énfasis en cómo se controla el reconocedor y el sintetizador de voz, se mostraron y describieron los

diagramas de clase y de secuencia. Por último se señaló como esta aplicación maneja los errores y excepciones que pueden ocurrir y se realizó una comparación del sistema. El siguiente capítulo habla sobre el desarrollo de las pruebas de usabilidad que se le realizaron al software así como el análisis de resultados e identificación de problemas.