Trabajo Final de Carrera - Compiladores

Enrique Marquet Durán

Ingeniería en Informática

Gerard Enrique Manonellas

09/01/2015

Resumen del proyecto

El objetivo de este proyecto final de carrera es el estudio del formato ODF (OpenDocument Format), para permitirnos la extracción de texto con la finalidad de transformarlo en texto hablado y guardar el resultado en un archivo de audio.

De los diferentes tipos de documentos que utilizan este estándar ODF nos centraremos en los de tipo texto, que tienen extensión *odt*. Para poder realizar una pequeña aplicación que cumpliera con el objetivo del proyecto se ha tenido que realizar un estudio de las diferentes tecnologías involucradas en esta cruzada.

La primera de las tecnologías que hemos estudiado ha sido el formato del XML y las herramientas que hay para su validación, centrándonos en las DTD. Hay que tener en cuenta que la arquitectura de los archivos OpenDocument está basada en XML, ya que se trata de un archivo comprimido (tipo ZIP) formado por diversos ficheros con este formato, como se verá en el apartado de estudio correspondiente.

Después también se ha realizado un estudio de los diferentes formatos de audio existentes en el mercado, no ha sido un estudio muy técnico, si no que se ha centrado más en ver qué se podía encontrar actualmente y qué diferencias existen entre ellos, para después poder utilizar un formato en la aplicación que se realizaría.

Por último se ha estudiado la tecnología *Text-To-Speech* (conversión de texto-voz), que está muy ligada a la lingüística. En este apartado se ha visto las diferentes posibilidades que hay para realizar el proceso de conversión, así como las diferentes herramientas que hay en el mercado que lo realizan, también se buscaron librerías que permitieran realizar este proceso para introducirlo en nuestra aplicación resultante del estudio.

La pequeña aplicación realizada en este proyecto final de carrera se ha nutrido de la información obtenida durante la fase de estudio de estos conceptos y tecnologías.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

Capítulo 1: Introducción

1.1 Justificación del PFC y contexto en el que se desarrolla: punto de partida y aportación del PFC

Actualmente, toda la información a nivel de empresa se documenta mediante documentación escrita que se suele almacenar en soporte digital y discos duros de las organizaciones.

La verdad es que a nivel personal también está resultando una tendencia, ya no solemos guardar textos escritos a mano. Por otro lado también es raro encontrar documentos escritos por máquinas de escribir. En la actualidad se ha masificado el uso de procesadores de texto, ya que nos facilitan mucho el trabajo de escribirlos.

Por esta razón en este proyecto nos centraremos en un formato abierto llamado OpenDocument Format (ODF) que permite realizar todo tipo de documentación (se haría utilizando cualquier programa de libre distribución que soporte este formato, como OpenOffice).

Ahora bien, esto nos permitiría acceder a la documentación, sin necesidad de papel, desde cualquier lugar. Pero no siempre estamos en disposición de leer, pero si con la necesidad de saber que es lo que contiene el documento, por esta razón cada vez más se generan audios con el contenido de documentos o libros para facilitar el acceso a esta información sin tener que dedicarse al 100% a esta tarea. La finalidad de este proyecto es realizar una aplicación que dé respuesta a esta necesidad.

Hoy en día la mayoría de aplicaciones se diseñan para tener en cuenta la diversidad funcional, ya que no todo el mundo dispone de las mismas habilidades, es por esta razón que empiezan a haber utilidades para convertir nuestros textos en textos audibles (que podamos leer utilizando nuestros oídos).

Gracias a la realización de este proyecto tendremos la oportunidad de profundizar en el estudio del formato abierto ODF que es un estándar de documentos de datos de OASIS. También habrá de hacer un estudio previo del formato XML[1] en el cuál se basa este formato. También nos dará la oportunidad de estudiar los diferentes formatos de audio libre y herramientas para la síntesis de voz que se pueden encontrar en el mercado.

Por lo tanto, este proyecto será un medio para aprender diferentes tecnologías, que cada vez más

1 Siglas en inglés de eXtensible Markup Language

se relacionan entre sí, y que nos servirá como base para realizar una aplicación que coja texto de un documento y nos genere archivos de audio por medio de la síntesis de voz.

En definitiva, este proyecto pretende enseñar las ventajas que aportan estas tecnologías cuando se aplican en conjunto y que pueda servir como punto de partida de nuevos estudios o desarrollos mediante estas tecnologías.

1.2 Objetivos del TFC

Mucha gente utiliza la suite OpenOffice para realizar todo tipo de documentación, pero lo que no conoce tanta gente es cómo se estructuran estos documentos internamente. Una parte de este trabajo consistirá en el estudio de su arquitectura interna.

También tenemos como objetivo saber como pasar un documento de texto a audio.

Los objetivos generales del presente Trabajo Final de Carrera son:

- El estudio del formato de documento de texto OpenOffice, más concretamente la versión 1.2 de ODF (Open Document Format).
- Analizar los diferentes formatos de audio libre y herramientas para la síntesis de voz (Text-To-Speech), así como las utilidades que hay en el mercado para OpenDocument en este ámbito.
- Definir documento XML 'lenguaje de marcas extensible' con la configuración para controlar el proceso de generación de voz y su correspondiente DTD[2] 'definición de tipo de documento'.
- Diseñar e implementar una herramienta que realice el proceso de síntesis de voz, partiendo de un documento de texto (OpenDocument) y otro de configuración (XML).

Mi objetivo es realizar la aplicación en Java, dada mi experiencia y conocimientos adquiridos durante la carrera, con la ayuda de librerías especificas para el tratamiento de documentos ODF y para el proceso de síntesis de voz y generación de ficheros de audio.

1.3 Enfoque y método seguido

Como se ha dicho en el apartado anterior, tendremos que realizar una aplicación partiendo de un

2 Siglas en inglés de document type definition

documento de texto, con formato ODF, y un documento de configuración permita sintetizar su contenido y generar archivos de audio para que el usuario los pueda escuchar. Para realizar esta tarea se debe partir de unos conocimientos previos de las tecnologías que se aplicaran y, por este motivo, tenemos una tarea importante de búsqueda y estudio.

Como primer paso, empezaremos por realizar un estudio del formato XML y DTD. Seguidamente estudiaremos el formato y la arquitectura de los documentos OpenDocument que, como veremos, está basada en XML.

Posteriormente, habrá que estudiar los diferentes formatos de audio y las herramientas para sintetizar voz (Text-To-Speech), que es un pilar importante para el desarrollo de este proyecto.

Una vez adquiridos estos conocimientos, nos encontraremos en condiciones para empezar el diseño de una aplicación que permita plasmar lo aprendido en esta investigación.

Para hacer este desarrollo seguiremos el ciclo de de vida clásico (o en cascada), es decir, pasaremos por las fases de especificación, análisis, diseño, implementación y pruebas hasta llegar al objetivo propuesto.

1.4 Planificación del proyecto

La planificación de este proyecto ha consistido en dividir en tareas cada uno de los hitos propuestos.

Hay que tener en cuenta que este proyecto tiene dos grandes bloques de trabajo bien diferenciados, por un lado tenemos, el estudio de las diferentes tecnologías y, por otro, el desarrollo de un producto a partir de los conocimientos adquiridos (con su fase de análisis, diseño e implementación).

Con esta división de tareas se podría pensar en usar un software específico para su planificación, pero al tratarse de un proyecto enfocado al estudio de unas determinadas tecnologías he optado por una temporización ajustada a los plazos de entrega, que permita profundizar en el estudio de estas tecnologías y deje un tiempo razonable para el desarrollo de la aplicación.

Debido a mi disponibilidad de tiempo no me resulta posible fijar una jornada laboral de duración fija, ni designar a priori los días laborables de la semana.

Para la estimación de esfuerzos se ha considerado en general una dedicación prevista de unas 10 horas por cada semana natural del plan. No obstante, se han realizado ajustes en las duraciones

de algunas tareas para tener en cuenta que su periodo de realización incluye fines de semana o festivos, ya que son días en los que se pueden concentrar un mayor número de horas de trabajo.

1.4.1 Planificación temporal

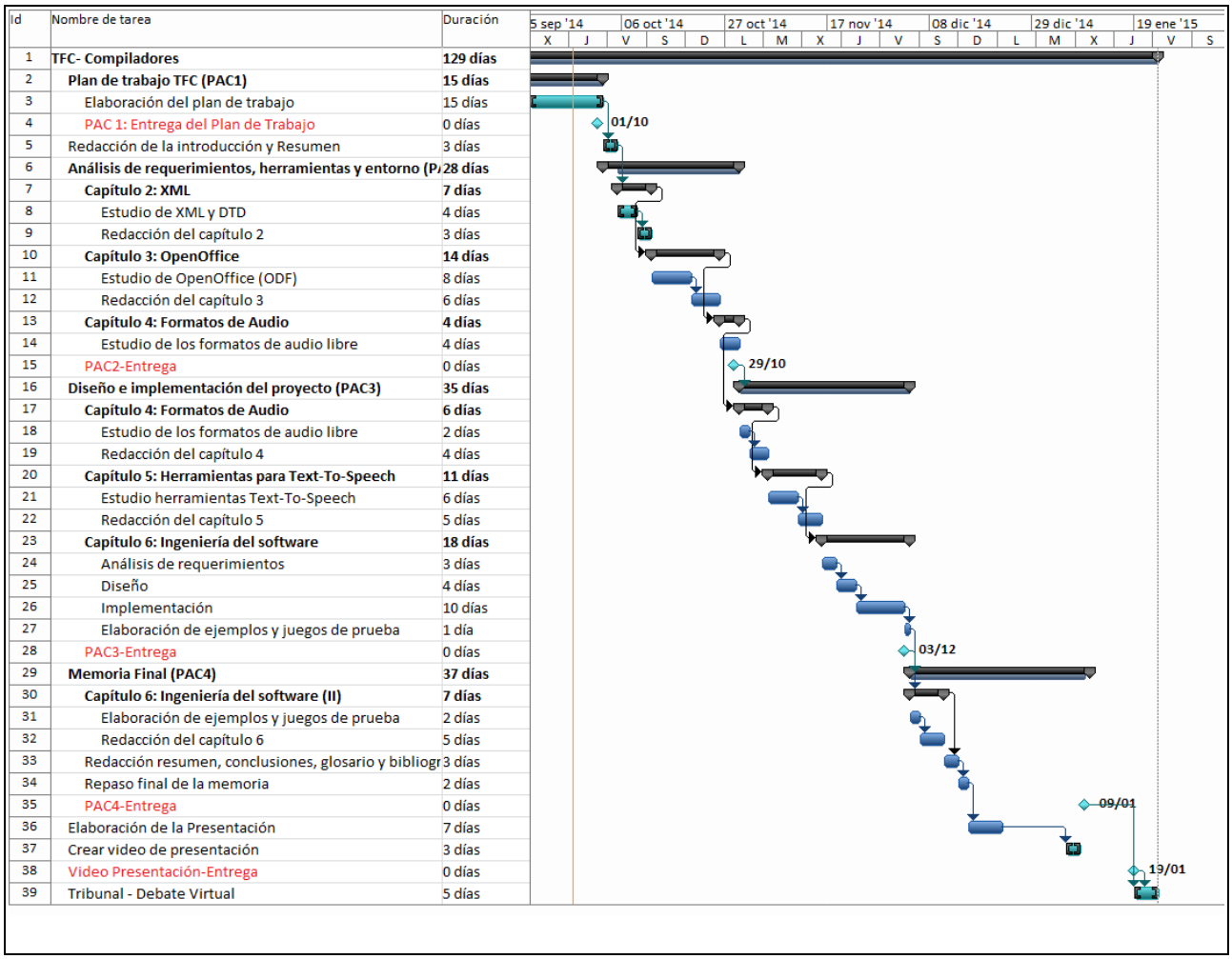
Utilizaré Microsoft Project 2010 para hacer la planificación. He considerado la fecha de inicio del proyecto el 17-09-2014, y he tenido en cuenta cada una de las Pruebas de Evaluación Continuada como hitos del proyecto. Sólo he tenido en cuenta las actividades y subactividades principales para que la tabla no quedase demasiado cargada impidiendo de ver lo relevante del proyecto.

La planificación temporal de cada actividad y subactividad es la siguiente:

Id	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	**TFC- Compiladores**	**129 días**	**mié 17/09/14**	**vie 23/01/15**	
2	**Plan de trabajo TFC (PAC1)**	**15 días**	**mié 17/09/14**	**mié 01/10/14**	
3	Elaboración del plan de trabajo	15 días	mié 17/09/14	mié 01/10/14	
4	PAC 1: Entrega del Plan de Trabajo	0 días	mié 01/10/14	mié 01/10/14	
5	Redacción de la introducción y Resumen	3 días	jue 02/10/14	sáb 04/10/14	3
6	**Análisis de requerimientos, herramientas y entorno (PAC 2)**	**28 días**	**jue 02/10/14**	**mié 29/10/14**	
7	**Capítulo 2: XML**	**7 días**	**dom 05/10/14**	**sáb 11/10/14**	**5**
8	Estudio de XML y DTD	4 días	dom 05/10/14	mié 08/10/14	
9	Redacción del capítulo 2	3 días	jue 09/10/14	sáb 11/10/14	8
10	**Capítulo 3: OpenOffice**	**14 días**	**dom 12/10/14**	**sáb 25/10/14**	**7**
11	Estudio de OpenOffice (ODF)	8 días	dom 12/10/14	dom 19/10/14	
12	Redacción del capítulo 3	6 días	lun 20/10/14	sáb 25/10/14	11
13	**Capítulo 4: Formatos de Audio**	**4 días**	**dom 26/10/14**	**mié 29/10/14**	**10**
14	Estudio de los formatos de audio libre	4 días	dom 26/10/14	mié 29/10/14	
15	PAC2-Entrega	0 días	mié 29/10/14	mié 29/10/14	
16	**Diseño e implementación del proyecto (PAC3)**	**35 días**	**jue 30/10/14**	**mié 03/12/14**	**15**
17	**Capítulo 4: Formatos de Audio**	**6 días**	**jue 30/10/14**	**mar 04/11/14**	**13**
18	Estudio de los formatos de audio libre	2 días	jue 30/10/14	vie 31/10/14	
19	Redacción del capítulo 4	4 días	sáb 01/11/14	mar 04/11/14	18
20	**Capítulo 5: Herramientas para Text-To-Speech**	**11 días**	**mié 05/11/14**	**sáb 15/11/14**	**17**
21	Estudio herramientas Text-To-Speech	6 días	mié 05/11/14	lun 10/11/14	
22	Redacción del capítulo 5	5 días	mar 11/11/14	sáb 15/11/14	21
23	**Capítulo 6: Ingeniería del software**	**18 días**	**dom 16/11/14**	**mié 03/12/14**	**20**
24	Análisis de requerimientos	3 días	dom 16/11/14	mar 18/11/14	
25	Diseño	4 días	mié 19/11/14	sáb 22/11/14	24
26	Implementación	10 días	dom 23/11/14	mar 02/12/14	25
27	Elaboración de ejemplos y juegos de prueba	1 día	mié 03/12/14	mié 03/12/14	26
28	PAC3-Entrega	0 días	mié 03/12/14	mié 03/12/14	
29	**Memoria Final (PAC4)**	**37 días**	**jue 04/12/14**	**vie 09/01/15**	**28**
30	**Capítulo 6: Ingeniería del software (II)**	**7 días**	**jue 04/12/14**	**mié 10/12/14**	**27**
31	Elaboración de ejemplos y juegos de prueba	2 días	jue 04/12/14	vie 05/12/14	
32	Redacción del capítulo 6	5 días	sáb 06/12/14	mié 10/12/14	31
33	Redacción resumen, conclusiones, glosario y bibliografía	3 días	jue 11/12/14	sáb 13/12/14	30
34	Repaso final de la memoria	2 días	dom 14/12/14	lun 15/12/14	33
35	PAC4-Entrega	0 días	vie 09/01/15	vie 09/01/15	
36	Elaboración de la Presentación	7 días	mar 16/12/14	lun 22/12/14	34
37	Crear video de presentación	3 días	lun 05/01/15	mié 07/01/15	36
38	Video Presentación-Entrega	0 días	lun 19/01/15	lun 19/01/15	
39	Tribunal - Debate Virtual	5 días	lun 19/01/15	vie 23/01/15	35;38

Fig. 1: Tareas del proyecto

La planificación temporal de cada una de las tareas según el método de Gantt es la siguiente:



Id	Nombre de tarea	Duración
1	**TFC- Compiladores**	**129 días**
2	**Plan de trabajo TFC (PAC1)**	**15 días**
3	Elaboración del plan de trabajo	15 días
4	PAC 1: Entrega del Plan de Trabajo	0 días
5	Redacción de la introducción y Resumen	3 días
6	**Análisis de requerimientos, herramientas y entorno (P**	**28 días**
7	**Capítulo 2: XML**	**7 días**
8	Estudio de XML y DTD	4 días
9	Redacción del capítulo 2	3 días
10	**Capítulo 3: OpenOffice**	**14 días**
11	Estudio de OpenOffice (ODF)	8 días
12	Redacción del capítulo 3	6 días
13	**Capítulo 4: Formatos de Audio**	**4 días**
14	Estudio de los formatos de audio libre	4 días
15	PAC2-Entrega	0 días
16	**Diseño e implementación del proyecto (PAC3)**	**35 días**
17	**Capítulo 4: Formatos de Audio**	**6 días**
18	Estudio de los formatos de audio libre	2 días
19	Redacción del capítulo 4	4 días
20	**Capítulo 5: Herramientas para Text-To-Speech**	**11 días**
21	Estudio herramientas Text-To-Speech	6 días
22	Redacción del capítulo 5	5 días
23	**Capítulo 6: Ingeniería del software**	**18 días**
24	Análisis de requerimientos	3 días
25	Diseño	4 días
26	Implementación	10 días
27	Elaboración de ejemplos y juegos de prueba	1 día
28	PAC3-Entrega	0 días
29	**Memoria Final (PAC4)**	**37 días**
30	**Capítulo 6: Ingeniería del software (II)**	**7 días**
31	Elaboración de ejemplos y juegos de prueba	2 días
32	Redacción del capítulo 6	5 días
33	Redacción resumen, conclusiones, glosario y bibliogr	3 días
34	Repaso final de la memoria	2 días
35	PAC4-Entrega	0 días
36	Elaboración de la Presentación	7 días
37	Crear video de presentación	3 días
38	Video Presentación-Entrega	0 días
39	Tribunal - Debate Virtual	5 días

Fig. 2: Diagrama de Gantt

1.4.2 Descripción de las actividades

Después de hacer un análisis a priori del proyecto he determinado unas tareas principales que se corresponden con la elaboración del plan de trabajo, y como he dicho antes, en el estudio de las tecnologías implicadas en el proyecto, el desarrollo de una aplicación y la preparación de una presentación del trabajo realizado. Paso a describir brevemente en que consistirán estas tareas principales:

Plan de trabajo

Se trata de describir cómo se realizará el proyecto y en qué consistirá. Se deben identificar las tareas más importantes que tendremos, para poder pasar a realizar una estimación de coste temporal para las diferentes partes.

Estudio del lenguaje XML

Este es el punto inicial en el que se basa gran parte del proyecto, hay que tener en cuenta que los documentos ODF están basados en este lenguaje. Primero se hará una descripción de este lenguaje, explicando cuáles son sus características más importantes y sintaxis para la elaboración de documentos XML. También veremos que para su validación se puede utilizar DTDs, de estos veremos su sintaxis y cómo se utiliza para validar documentos XML a través de su descripción.

Estudio del formato OpenOffice

Es un punto importante, y básico, de estudio para este proyecto. Nos centraremos principalmente en el estudio del formato de los documentos de texto, para después poder realizar la aplicación de convertir el texto en audio.

Haré una introducción al formato OpenDocument, cuáles son sus ventaja, también comentaré los diferentes formatos de documentos que soporta.

En un aspecto más técnico habrá que explicar cuál es su arquitectura, y se verá cuál es el contenido de un documento de texto.

También habrá que hablar de las diferentes herramientas o librerías que hay para dar soporte a la programación.

Estudio de los formatos de audio

En este apartado intentaremos hacer un estudio de los diferentes formatos de audio libres que existen en el mercado, para posteriormente decidir cuál se utilizará en nuestro aplicativo.

Seguramente la decisión de que formato de audio se utilizará vendrá también condicionada por la elección de la herramienta Text-To-Speech que se estudiará en el siguiente apartado.

Estudio de herramientas Text-To-Speech

En este apartado se hará un estudio de las diferentes herramientas Text-To-Speech que hay en el mercado, así como las herramientas para la síntesis de voz.

Se intentará buscar la más indicada para integrarla en nuestro aplicativo, teniendo en cuenta que se realiza en entorno Java.

Ingeniería del software

Este es un gran bloque que engloba todas las tareas para el desarrollo de una aplicación que

cumpla con las especificaciones.

Este apartado lo hemos divido en tres bloques que son: análisis de requerimientos, diseño, e implementación. También se incluirá un subapartado para los ejemplos y juegos de prueba.

1.5 Análisis de riesgos

Con la información disponible para empezar el proyecto, se identifican dos riesgos importantes a tener en cuenta que podrían provocar no alcanzar el objetivo final. Estos son:

- *La falta de tiempo*

  Debido a la larga jornada laboral del estudiante, le queda pocas horas al día para dedicar al proyecto, aunque en los días festivos puede dedicarle más horas siempre y cuando no ocurran imprevistos. Esto podría suponer un retraso y podría suponer el fracaso del proyecto.

- *Inexperiencia con la tecnología envuelta en el proyecto*

  Se tratan de nuevas tecnologías para el alumno, con lo que es sumamente importante la fase de investigación y documentación de cada una de las tecnologías implicadas en el proyecto. Este estudio es el que se gastará la mayor parte del tiempo. Se espera que en la fase de programación, estas tecnologías no nos causen demasiadas complicaciones porque podría suponer la no consecución de la aplicación representativa de los conocimientos adquiridos de estas tecnologías.

A continuación identifico los diferentes riesgos, que en esta fase se identifican como posibles, con su grado de repercusión en el proyecto y las consecuencias que se podrían derivar.

1.5.1 Riesgo de problemas relacionados con la investigación

Grado de repercusión: Medio.

Consecuencias: repercutiría en un retraso en las posteriores etapas.

1.5.2 Riesgo de problemas relacionados con la tecnología

Grado de repercusión: Medio.

Consecuencias: Si nos encontramos con problemas para juntar todas las tecnologías, nos

retrasaría el desarrollo de la aplicación, ya que habría que buscar otras soluciones (librerías, sistemas de comunicación, interficies, etc) que se pudieran integrar para resolver el problema.

1.5.3 Riesgo de problemas con los juegos de pruebas

Grado de repercusión: Bajo.

Consecuencias: Si nos encontramos con este tipo de problemas, no queda otra solución que volver a realizarlos.

1.5.4 Riesgo derivado del no acceso a Internet

Grado de repercusión: Alto.

Consecuencias: Es un riesgo que hoy en día es difícil que suceda ya que las comunicaciones son muy estables. Pero mi experiencia en el trabajo me dice que pueden haber cortes provocados por inclemencias meteorológicas o por operaciones realizadas por los operadores de la red, y a veces tardan a normalizar el servicio. Esto repercutiría principalmente en la fase de investigación, ya que nos imposibilitaría de realizar esta tarea, en la fase de desarrollo en principio ya no afectaría tanto.

1.6 Productos obtenidos

Los productos obtenidos de este Trabajo Final de Carrera son los siguientes:

- El Plan de Trabajo, que recoge la planificación y estimación de las tareas necesarias para llevar a cabo los objetivos previstos.
- El Producto, que consiste en una aplicación que a partir de un documento de texto lo procesa generando ficheros de audio con voz sintetizada, y su correspondiente documentación técnica asociada.
- La presente Memoria, que es el documento que sintetiza el trabajo realizado y muestra que se han alcanzado los objetivos propuestos. Incluye toda la información relevante del estudio realizado sobre las diferentes tecnologías utilizadas en el proyecto y detalla la solución elaborada.
- La Presentación, que explicará los conceptos principales del proyecto y resumirá de forma clara y concisa el trabajo realizado así como los resultados obtenidos.

1.7 Breve descripción de los otros capítulos de la memoria

Como hemos dicho en puntos anteriores, la realización de este proyecto está dividido en dos grandes bloques bien diferenciados. Por un lado, tenemos el estudio de las diferentes tecnologías necesarias para elaborar este proyecto y, por otro, la realización de una aplicación en la que se utilice los conocimientos adquiridos.

Los capítulos quedaran estructurados de la siguiente manera:

1) Estudio del formato XML, los esquemas DTD asociados y el proceso de validación de los documentos que siguen este formato.
2) Estudio del formato de documento ODF a partir de la presentación de su arquitectura, y sus características principales.
3) Un estudio de los formatos de audio que se puede encontrar en el mercado, centrándonos en los que nos pueden ser útiles para la realización de la aplicación.
4) Miraremos qué herramientas podemos encontrar para la realización de Text-To-Speech.
5) Tendremos un capítulo dedicado al proceso de la Ingeniería del Software utilizado para la generación de la aplicación.
6) Por último habrá un capitulo para las conclusiones, valoraciones y líneas futuras de trabajo que podrá seguir este PFC.

Capítulo 2: Estudio del formato XML

2.1 Definición y objetivos

XML (Extensible Markup Language) es un lenguaje de marcas desarrollado por el World Wide Web Consortium (W3C) que deriva del lenguaje SGML y es utilizado para almacenar datos en forma legible, o para estructurar información en un documento o en cualquier fichero de texto, como por ejemplo ficheros de configuración de un programa o una tabla de datos.

El formato XML se ha convertido en un estándar para el intercambio de datos entre diferentes plataformas. Se puede decir que es una tecnología sencilla con un gran potencial en muchos ámbitos.

2.2 Breve historia

Todo comenzó en los años setenta cuando IBM diseñó Generalized Markup Language (GML) para solucionar los problemas que tenían para almacenar, de forma estructurada, grandes cantidades de información. Siendo normalizado por ISO en 1986 bajo la norma Standard Generalized Markup Language (SGML). Esta norma permitió definir nuevos lenguajes basados en etiquetas.

En 1989 Tim Berners Lee creó la web y el lenguaje HTML bajo el marco del lenguaje SGML. Poseía una limitación, cada documento pertenecía a un vocabulario fijo predeterminado por el DTD, no permitiendo la combinación de elementos de diferentes vocabularios.

La aparición de XML en 1996 (y su primera especificación en 1998) como subconjunto del lenguaje SGML permitió hacer más fácil a los programas automáticos la interpretación de documentos.

2.3 Características de un documento XML

Las características más importantes de este formato son:

- Es un método que permite introducir datos estructurados en un fichero de texto. Es una característica muy importante ya que permite, a través de etiquetas, facilitar la legibilidad de estos ficheros.
- Se trata de un lenguaje extensible. Esto significa que una vez diseñado permite añadirle

nuevas marcas de manera que se pueda seguir entendiendo el nuevo formato por parte de los antiguos usuarios.

- Permite crear estructuras jerárquicas. Pero sólo permiten un nodo raíz del que todos los demás están incluidos, esto significa que sólo pueden tener un elemento inicial.
- Acepta diferentes codificaciones de caracteres. En el apartado de sintaxis de este formato veremos como se declara el tipo de codificación del documento.

2.3.1 Estructura del formato XML

Como ya hemos dicho se trata de una estructura jerárquica, en la cuál solo hay un elemento raíz para toda la estructura.

Esta estructura está compuesta por diferentes elementos (o marcas) codificados de la siguiente manera:

```
<elemento></elemento>
```

Veamos un ejemplo, queremos guardar la colección de discos que tenemos. Podemos pensar en un disco como un álbum en el que guardaremos información sobre su autor, título, etc. Se supone que tendremos más de un disco, en la figura siguiente vemos como sería la estructura de la información que queremos guardar en el fichero XML:

```
<discos>
   <album>
        <autor>SABINA Y CIA Nos sobran los motivos</autor>
        <titulo>Joaquín Sabina</titulo>
        <formato>MP3</formato>
        <localizacion>Varios CD5 </localizacion>
   </album>
</discos>
```

Fig. 3: Ejemplo documento XML

Como se puede ver en este ejemplo, el formato XML contiene una serie de elementos que se estructuran de forma jerárquica, como habíamos comentado anteriormente. Estos elementos a su vez pueden tener contener unos atributos que se definirán dentro del mismo elemento:

```
<elemento atributo='valor'></elemento>
```

En este caso el elemento 'elemento' contiene un atributo que se llama 'atributo' y su valor es

'valor'.

Volviendo al ejemplo de los discos podríamos añadirle un identificador al álbum para que se pueda identificar fácilmente dentro del sistema de discos:

```
<discos>
   <album id=001>
        <autor>SABINA Y CIA Nos sobran los motivos</autor>
        <titulo>Joaquín Sabina</titulo>
        <formato>MP3</formato>
        <localizacion>Varios CD5 </localizacion>
   </album>
</discos>
```

Fig. 4: Ejemplo documento XML con atributo

2.3.2 Sintaxis de un documento XML

Todo documento XML se divide en un prólogo y un cuerpo del documento.

El prólogo del documento, aunque no es obligatorio, siempre se encuentra en la primera línea, y podrá contener:

- Una declaración XML. Es la sentencia que declara al documento como un documento XML.
- Una declaración de tipo de documento. Enlaza el documento con su DTD (definición de tipo de documento), o la DTD puede estar incluido en la propia declaración o ambas cosas al mismo tiempo.
- Algún comentario e instrucciones de procesamiento.

Veamos un ejemplo de la declaración XML:

```
<?xml version="1.0" encoding="iso-8859-1" standardalone=”yes”?>
```

De esta declaración se obtiene la siguiente información:

- El formato del documento mediante XML.
- La especificación de la versión de XML utilizada. Hasta el momento sólo existe la “1.0”.

- La codificación de caracteres utilizado mediante el atributo encoding (opcional). Hace referencia al modo en que se representan los caracteres internamente (los más normales UTF-8 y UTF-16), en este caso representa el alfabeto latin-1.
- El tipo de documento mediante el atributo standalone (opcional). En el caso de que su valor sea “no”, que es el valor por defecto, nos indica que el documento es un documento de definición de tipo; el otro valor posible, que es el que hemos puesto en este ejemplo significa que el documento es independiente.

En el cuerpo del documento estarán definidos los elementos, como hemos explicado en el apartado anterior, representados de forma estructurada y jerárquica teniendo un solo elemento raíz.

En la parte del cuerpo del documento también encontramos las entidades predefinidas, las secciones CDATA, y los comentarios.

Tanto las entidades predefinidas como las secciones CDATA permiten representar valores especiales que no serán interpretados como marcado por el procesador. En XML 1.0 se definen cinco entidades para representar caracteres especiales como se muestra en la siguiente figura[3]:

ENTIDAD	CARACTER
&amp;	&
&lt;	<
&gt;	>
&apos;	'
&quot;	"

Fig. 5: Entidades predefinidas

Las secciones CDATA empiezan por la cadena "<![CDATA[" y terminan con la cadena "]]>" y sólo ésta última se reconoce como marca.

Por último los comentarios, que se encuentran en el cuerpo, son puramente informativos y el procesador XML no los tiene en cuenta, Tienen el mismo formato que los comentarios de HTML, por lo que comienzan con "<!--" y terminan con "-->". La cadena "--" no puede aparecer dentro de un comentario.

Hasta ahora hemos descrito como está organizado un documento XML. Ahora bien, todos los

3 Figura obtenida de http://flanagan.ugr.es/xml/xml.htm

documentos XML deben estar “bien formado” esto se refiere a su estructura básica es decir, que se componga de elementos, atributos y comentarios como XML especifica que se escriban.

Además de lo que se ha comentado hasta ahora del formato XML, hay que decir que éste es "case sensitive", es decir, las mayúsculas y las minúsculas son tratados como caracteres diferentes. Siguiendo con el ejemplo del apartado anterior, tenemos:

<elemento></elemento> no es el mismo elemento que <Elemento></Elemento>

Todos los elementos y atributos del documento deben tener una sintaxis correcta. Los elementos XML pueden tener contenido (más elementos, caracteres o ambos a la vez), o bien ser elementos vacíos. El caso anterior tenemos una etiqueta de inicio y una de fin, para el caso de elementos vacíos se puede usar una marca de inicio seguida del carácter de fin de marca <elemento/>.

Los valores de los atributos de un elemento deben estar delimitados por comillas dobles (") o comillas simple ('). Cuando se usa uno para delimitar el valor del atributo, el otro se puede usar dentro

<elemento atributo1='valor1' atributo2=”valor2”></elemento>

o bien <elemento atributo1='valor1' atributo2=”valor2”/>

Finalmente, si nuestro documento cumple con las reglas de sintaxis explicadas en este apartado diremos que nuestro documento XML está “bien formado” como hemos comentado antes.

2.4 Validación de un documento XML

En el apartado anterior hemos visto que todo documento XML debe cumplir con una sintaxis para considerarlo “bien formado”. Ahora damos un paso más y tendremos que realizar un análisis más profundo para ver si utiliza el vocabulario correcto y, que la estructura y relaciones entre los diferentes elementos es la deseada. Si lo conseguimos podremos decir que nuestro documento es “válido”.

Existen varios métodos para validar un documento XML, entre los que destacan los *Esquemas XML*[4] y las *Definiciones del Tipo de Documento*[5].

En los subcapítulos siguientes se explicará con más profundidad que es la DTD y cómo se utiliza para validar los documentos XML, ya que se trata del método que utilizaremos para desarrollar la

4 Traducción del Inglés XML Schema
5 DTD (acrónimo del Inglés Document Type Definition)

aplicación de este proyecto.

2.4.1 Características de una DTD

Una DTD es un documento que nos permite definir la estructura de un documento XML, definiendo un conjunto de reglas sintácticas para definir las etiquetas.

La creación de una DTD es como crear nuestro propio lenguaje de marcado, para una aplicación específica. Todo documento que se ajusta a una DTD se dice que “válido”. Hay que remarcar que no todos los documentos XML deben tener necesariamente una DTD asociada.

Podemos decir que una DTD describe:

- **Elementos**: cuáles son las etiquetas permitidas y cuál es el contenido de cada etiqueta.
- **Estructura**: nos dice en qué orden van las etiquetas en el documento.
- **Anidamiento**: nos dice qué etiquetas van dentro de cuáles.

La especificación de una DTD se puede hacer de dos maneras diferentes:

- En un fichero externo: podrá ser compartido por varios (puede que miles) de documentos..
- Contenidos en el propio documento XML: formando parte de su declaración de tipo de documento.

Veamos un ejemplo, empezando por el primer caso de declaración externa:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT etiqueta (nombre, calle, ciudad, pais, codigo)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT calle (#PCDATA)>
<!ELEMENT ciudad (#PCDATA)>
<!ELEMENT pais (#PCDATA)>
<!ELEMENT codigo (#PCDATA)>
```

Fig. 6: Definición externa DTD

```
<?xml version=”1.0” encoding=”UTF-8”?>

<!DOCTYPE etiqueta SYSTEM "prueba.dtd">

<etiqueta>
      <nombre>Pepe García</nombre>
      <calle>C/Ronda, 3</calle>
      <ciudad>Armilla</ciudad>
      <pais>España</pais>
      <codigo>18465</codigo>
</etiqueta>
```

Fig. 7: Documento XML con DTD asociado

Si el ejemplo anterior queremos definir la DTD dentro del XML quedaría de la manera que muestra la siguiente figura

```
<?xml version=”1.0” encoding=”UTF-8”?>

<!DOCTYPE etiqueta[
<!ELEMENT etiqueta (nombre, calle, ciudad, pais, codigo)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT calle (#PCDATA)>
<!ELEMENT ciudad (#PCDATA)>
<!ELEMENT pais (#PCDATA)>
<!ELEMENT codigo (#PCDATA)>
 ]>

 <etiqueta>
      <nombre>Pepe García</nombre>
      <calle>C/Ronda, 3</calle>
      <ciudad>Armilla</ciudad>
      <pais>España</pais>
      <codigo>18465</codigo>
 </etiqueta>
```

Fig. 8: Documento XML con DTD interno

Las DTDs describen la estructura de los documentos XML mediante declaraciones. Los tipos de declaraciones que tenemos son:

- Declaraciones de entidades
- Declaraciones de notaciones
- Declaraciones de tipo de elemento
- Declaraciones de lista de atributos

Ahora pasamos a describir brevemente como serían estas declaraciones.

Declaración de entidades:

Una entidad consiste en un nombre y su valor (son similares a las constantes en los lenguajes de programación). El procesador XML sustituirá las referencias a entidades por sus valores antes de procesar el documento. En el documento utilizaremos la referencia de la siguiente manera, primero se pone el carácter “&” seguido del nombre de la entidad y terminado por “;”. (es decir, &nombreEntidad;).

Su declaración podría ser: <!ENTITY nombreEntidad "valorEntidad">

Las entidades pueden ser: Internas o Externas; Analizadas o No analizadas; Generales o Parámetro.

Declaración de notaciones:

Las notaciones se usan para definir las entidades externas que no serán analizadas por el procesador XML. Para referenciar estas entidades no se utiliza la notación estándar, definida antes, sino que se utiliza el nombre de la entidad directamente.

Declaración de tipo de elemento:

Los elementos son la base de las marcas XML. Estas declaraciones deben empezar con "<! ELEMENT" seguidas por el identificador genérico del elemento que se declara y una especificación de contenido.

<!ELEMENT nombreElemento (contenido)>

La especificación de contenido puede ser de cuatro tipos: EMPTY, (#PCDATA) o ANY o escribir expresiones más complejas.

La definición EMPTY significa que el elemento es vacío, esto es, no puede tener contenido

<!ELEMENT salto-de-pagina EMPTY>

La definición ANY significa que el elemento puede contener cualquier cosa (texto y otros elementos). No se suele utilizar ya que debemos estructurar adecuadamente los documentos XML.

Los (#PCDATA) son elementos que pueden contener texto.

<!ELEMENT ejemplo (#PCDATA)>

Las expresiones más complejas son las que el elemento contendrá otros elementos, estos se deben indicar utilizando conectores y modificadores. Ejemplos:

<!ELEMENT aviso (titulo?, (parrafo+, grafico)*)>

<!ELEMENT ejemplo ((a, b)|(b, a))>

Declaración de lista de atributos:

Los atributos permiten añadir información adicional a los elementos de un documento. Los atributos no pueden contener sub-atributos, además sólo se pueden especificar una vez y en cualquier orden.

Se definen de la siguiente manera:

<!ATTLIST nombreElemento nombreAtributo tipoAtributo valorInicialAtributo >

Los tipos de atributos son los siguientes:

- **CDATA**: texto sin restricciones
- **NMTOKEN**: “abc...z0123..9-_:.” (tipo de lista)
- **NMTOKENS**: NMTOKEN + espacios
- **ID**: empezar con letra
- **IDREF**: el valor del atributo debe coincidir con el valor del atributo ID de otro elemento
- **valores**: el atributo sólo puede contener uno de los términos de una lista. Esta lista va entre paréntesis y cada término separado por una barra vertical “|”.

Los valores iniciales pueden ser:

- **#REQUIRED**: el atributo es obligatorio, sin valor predeterminado
- **#IMPLIED**: el atributo no es obligatorio y sin valor predeterminado
- **#FIXED valor**: el atributo tiene un valor fijo
- **valor**: el atributo tiene un valor predeterminado

2.4.2 Ventajas e inconvenientes de las DTDs

Originalmente las DTD se desarrollaron para ser utilizados con SGML, fueron la primera forma de validación del formato XML mediante un esquema. Sus principales ventajas son:

- Permite que todos los XML tengan una descripción de su propio formato.
- Distintas personas se pueden poner de acuerdo para utilizar una DTD común para intercambiar datos.
- Un programa puede usar una DTD para verificar que los datos que se reciben del mundo externo son válidos,
- también puede usar la DTD para verificar sus datos propios.

Por otro lado las DTDs presentan unas limitaciones importantes:

- Posee un lenguaje propio, con lo que además de aprender XML hay que estudiar el lenguaje de las DTDs.
- No permite definir elementos locales que sólo sean válidos dentro de otros elementos. Para resolverlo habría que utilizar prefijos en el nombre del elemento para diferenciarlos.
- Es poco flexible en la definición de los tipos de datos de los elementos, no permite datos numéricos, fechas, monedas, etc. sino que sólo presenta variaciones limitadas sobre cadenas.

Capítulo 3: Estudio formato OpenOffice (ODF)

Apache OpenOffice es una suite ofimática libre (código abierto y distribución gratuita) que incluye herramientas como editor de textos, hoja de cálculo, presentaciones, base de datos, y herramientas para dibujo. Está disponible en varias plataformas. Soporta numerosos formatos de archivo, incluyendo como predeterminado el formato estándar ISO/IEC OpenDocument (ODF), este es el formato que estudiaremos con más detenimiento en este proyecto por ser un estándar abierto. Soporta también más de 110 idiomas.

3.1 Breve historia de OpenOffice

La historia de Apache OpenOffice se remonta a 1994, año en que comenzó el desarrollo de la suite ofimática propietaria StarOffice.

El 19 de julio de 2000, Sun Microsystems anunció que dejaba disponible el código fuente de StarOffice para descarga bajo la Licencia pública general limitada de GNU (LGPL) con la intención de construir una comunidad de desarrollo de código abierto alrededor de este programa (proyecto OpenOffice.org).

El 20 de octubre de 2005, OpenOffice.org 2.0 fue lanzado de forma oficial. Teniendo que sacar una versión con los parches para los errores menores al cabo de ocho semanas, aprovecharon para añadirle nuevas características.

Antes de que el código base fuera donado a Apache, OpenOffice.org se encontraba en la fase beta de la versión 3.4.

El 8 de mayo de 2012 se lanzó Apache OpenOffice 3.4, primera versión después de la donación, esta versión incluyó una nueva opción de cifrado de ODF 1.2.

Apache OpenOffice 4 salió en Julio de 2013, y ahora estamos en la versión 4.1.1 (del 21/08/2014).

OpenOffice.org permite importar y exportar documentos en diferentes formatos de archivo.

Las aplicaciones incluidas en la suite ofimática Apache OpenOffice son las siguientes: Writer (procesador de textos), Calc (hoja de cálculo), Impress (programa de presentación similar a Microsoft PowerPoint), Base (es un programa de base de datos similar a Microsoft Access), Draw (editor de gráficos vectoriales y herramienta de diagramación, similar a Microsoft Visio) y Math

(aplicación diseñada para la creación y edición de fórmulas matemáticas).

Como he dicho antes utilizaremos el formato OpenDocument (ODF) para el proyecto, porque es un formato abierto y lo utiliza OpenOffice. En el capítulo siguiente profundizaremos más en este formato.

3.2 Introducción

El *Formato de Documento Abierto para Aplicaciones Ofimáticas* de OASIS[6] es un formato de archivo abierto y estándar especialmente enfocado a las aplicaciones ofimáticas cuyo objetivo es permitir que los documentos creados con cualquier aplicación ofimática puedan ser interpretados correctamente por otra. Para conseguirlo y maximizar la interoperatividad entre aplicaciones, reutiliza estándares ya establecidos como XML, XHTML, SVG, XSL, Xlink, Xforms, MathML.

OpenDocument fue publicado como estándar OASIS el 1 de mayo de 2005 y desde entonces ha ido superando diversas fases en su proceso de estandarización.

ODF es un formato libre que pretende establecerse como estándar de manera que no se produzca ningún tipo de pérdida a la hora de trabajar con un mismo fichero empleando distintas aplicaciones ofimáticas. Para que no se tenga de depender siempre de un mismo producto ofimático de pago.

3.3 Arquitectura OpenDocument

Cuando creamos un documento de texto nuevo, incluso sin ponerle texto dentro, y lo grabamos, cuando vamos a ver el fichero generado nos puede sorprender que no ocupe 0 kb. Esto es debido a que todos los archivos llevan incorporada una información sobre su autor, fecha de creación, plantillas empleadas, etc. Son datos que se unirían a lo que sería el texto propiamente dicho.

Hay que decir que un fichero ODF tiene una estructura determinada para todos los documentos. Este fichero integra un conjunto de archivos en una carpeta comprimida (en formato ZIP).

Ahora pasamos a ver cuál es su estructura interna, al ser un archivo comprimido, cuando lo descomprimimos nos encontramos con:

- **Archivos XML**: content.xml, meta.xml, settings.xml y styles.xml

6 en inglés, OASIS Open Document Format for Office Applications, también referido como formato OpenDocument (ODF)

- **Otros archivos**: mimetype y layout-cache
- **Directorios**: META-INF/ ,Thumbnails/ ,Pictures/ y Configurations2/

El formato OpenDocument ofrece una clara separación entre el contenido, la disposición de éste en el documento y los metadatos.

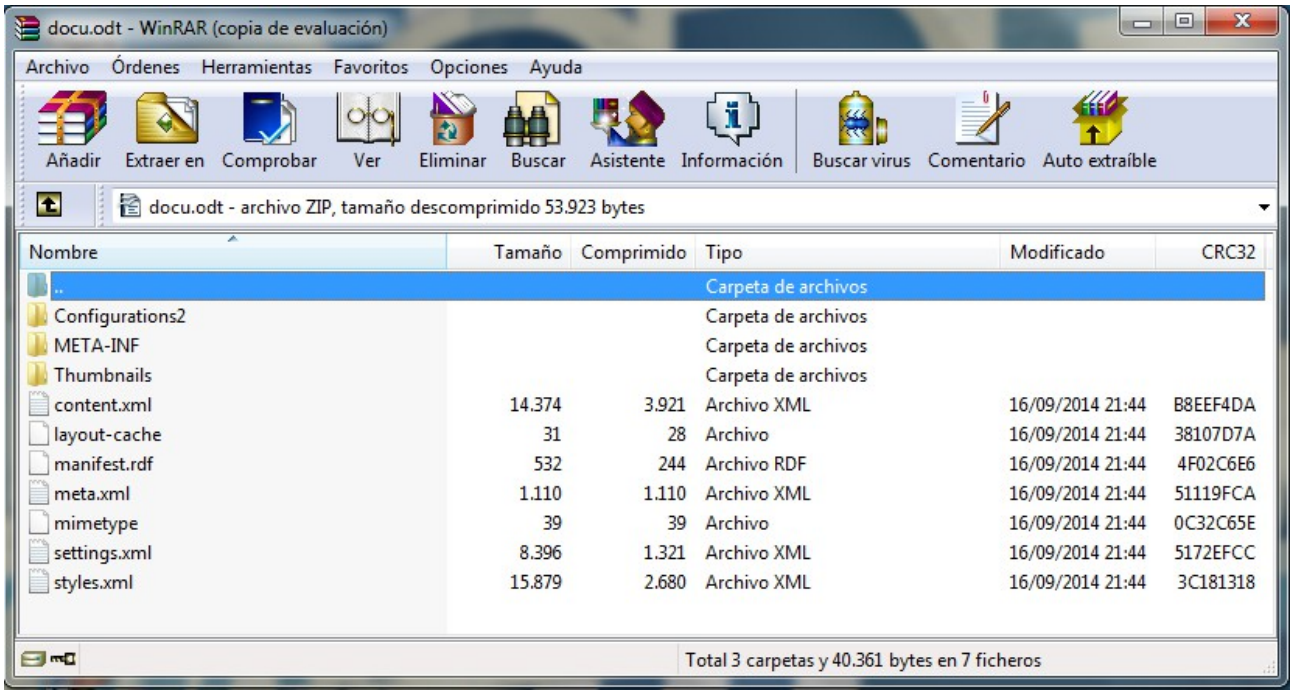


Fig. 9: Estructura documento ODF descomprimido

Ahora pasamos a explicar lo más importante del contenido de este archivo comprimido.

3.3.1 Fichero content.xml

Este es el archivo más importante porque almacena el contenido real del documento (excepto los datos binarios como las imágenes). El formato está inspirado en el lenguaje HTML, aunque es bastante más complejo. Si quisiéramos modificar un archivo de texto podríamos efectuar los cambios en este documento y comprobaríamos, una vez comprimida la carpeta, cómo se habían realizado los cambios.

La parte de inicio del fichero hace referencia a la definición de los esquemas que se utilizan en el documento, después nos encontramos con la declaración de las fuentes utilizadas en el documento.

```
<?xml version="1.0" encoding="UTF-8"?>
- <office:document-content office:version="1.2" xmlns:field="urn:openoffice:names:experimental:ooo-ms-interop:xmlns:field:1.0"
  xmlns:textooo="http://openoffice.org/2013/office" xmlns:tableooo="http://openoffice.org/2009/table"
  xmlns:grddl="http://www.w3.org/2003/g/data-view#" xmlns:xhtml="http://www.w3.org/1999/xhtml"
  xmlns:of="urn:oasis:names:tc:opendocument:xmlns:of:1.2" xmlns:rpt="http://openoffice.org/2005/report"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xforms="http://www.w3.org/2002/xforms" xmlns:dom="http://www.w3.org/2001/xml-events"
  xmlns:oooc="http://openoffice.org/2004/calc" xmlns:ooow="http://openoffice.org/2004/writer"
  xmlns:ooo="http://openoffice.org/2004/office" xmlns:script="urn:oasis:names:tc:opendocument:xmlns:script:1.0"
  xmlns:form="urn:oasis:names:tc:opendocument:xmlns:form:1.0" xmlns:math="http://www.w3.org/1998/Math/MathML"
  xmlns:dr3d="urn:oasis:names:tc:opendocument:xmlns:dr3d:1.0" xmlns:chart="urn:oasis:names:tc:opendocument:xmlns:chart:1.0"
  xmlns:svg="urn:oasis:names:tc:opendocument:xmlns:svg-compatible:1.0"
  xmlns:number="urn:oasis:names:tc:opendocument:xmlns:datastyle:1.0" xmlns:meta="urn:oasis:names:tc:opendocument:xmlns:meta:1.0"
  xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:fo="urn:oasis:names:tc:opendocument:xmlns:xsl-fo-compatible:1.0"
  xmlns:draw="urn:oasis:names:tc:opendocument:xmlns:drawing:1.0" xmlns:table="urn:oasis:names:tc:opendocument:xmlns:table:1.0"
  xmlns:text="urn:oasis:names:tc:opendocument:xmlns:text:1.0" xmlns:style="urn:oasis:names:tc:opendocument:xmlns:style:1.0"
  xmlns:office="urn:oasis:names:tc:opendocument:xmlns:office:1.0">
    <office:scripts/>
  - <office:font-face-decls>
      <style:font-face svg:font-family="Mangal" style:name="Mangal1"/>
      <style:font-face svg:font-family="'Times New Roman'" style:name="Times New Roman" style:font-pitch="variable" style:font-family-
        generic="roman"/>
      <style:font-face svg:font-family="Arial" style:name="Arial" style:font-pitch="variable" style:font-family-generic="swiss"/>
      <style:font-face svg:font-family="Mangal" style:name="Mangal" style:font-pitch="variable" style:font-family-generic="system"/>
      <style:font-face svg:font-family="'Microsoft YaHei'" style:name="Microsoft YaHei" style:font-pitch="variable" style:font-family-
        generic="system"/>
      <style:font-face svg:font-family="SimSun" style:name="SimSun" style:font-pitch="variable" style:font-family-generic="system"/>
    </office:font-face-decls>
```

Fig. 10: Archivo contenido content.xml

Después en office:automatic-styles están declarados los estilos particulares del documento. Si seguimos mirando el contenido del fichero nos encontramos con el cuerpo del documento que sería *<office:body>* donde está descrita la estructura del documento, así como su contenido escrito. En la figura siguiente se muestra la continuación del fichero enseñado en la figura anterior, en la que se puede ver un ejemplo en el que hay sólo tres líneas escritas a modo de ejemplo, junto con un estilo particular de este documento.

```
  - <office:automatic-styles>
    - <style:style style:name="P1" style:parent-style-name="Standard" style:family="paragraph">
        <style:text-properties style:font-weight-complex="bold" style:font-weight-asian="bold" fo:font-weight="bold"/>
      </style:style>
    </office:automatic-styles>
  - <office:body>
    - <office:text>
      - <text:sequence-decls>
          <text:sequence-decl text:name="Illustration" text:display-outline-level="0"/>
          <text:sequence-decl text:name="Table" text:display-outline-level="0"/>
          <text:sequence-decl text:name="Text" text:display-outline-level="0"/>
          <text:sequence-decl text:name="Drawing" text:display-outline-level="0"/>
        </text:sequence-decls>
        <text:h text:outline-level="1" text:style-name="Heading_20_1">Esto es la cabecera del texto</text:h>
        <text:p text:style-name="P1">Esto es un párrafo en negrita</text:p>
        <text:p text:style-name="Standard">Este es otro párrafo normal.</text:p>
      </office:text>
    </office:body>
</office:document-content>
```

Fig. 11:Archivo contenido content.xml (cont.)

Como podemos ver, para un documento de texto de tres líneas nos ha generado muchas líneas. Por este motivo, creo que es mejor explicar los elementos más representativos de este

documento, centrándonos en los que se necesitan para elaborar el aplicativo resultante de este estudio.

A continuación vemos cuáles son estos elementos:

Párrafos:

En un documento se podría decir que es el elemento más importante (se identifica por <*text:p*>). Este elemento contiene un conjunto de caracteres y/o otros elementos, tanto elementos que contienen texto como estilos.

Junto con éstos podríamos clasificar también las cabeceras (en inglés headings), se identifican con <*text:h*>, que contienen el texto de la cabecera y su nivel entre otros atributos.

En los párrafos nos podemos encontrar con:

- Elementos Span <*text:span*> que representan porciones de texto que poseen unos atributos o estilos diferentes del párrafo en que se encuentra.
- Notas <*text:note*> sirven para insertar una nota al pie de página o al final. El texto que va en la nota está definido en una etiqueta tipo párrafo dentro de nota.
- Marcas y referencias.
- Enlaces <*text:a*>, entre otros atributos tiene la dirección de enlace.
- También nos encontramos con listas <*text:list*>
- Elementos de dibujo e imagen.
- Hay elementos que dejan un espacio en blanco <*text:s/*>, los tabuladores <*text:tab*> , los saltos de línea <*text:line-break/*>, o salto de página <*text:soft-page-break/*>

Listas:

Las listas están formadas por enumeraciones y viñetas, se identifican por <*text:list*>. Dentro de este elemento nos encontramos con dos tipos de elementos que serian los items que contendrían el texto de cada una de las entradas de la lista (pueden ser elementos tipo <*text:h*> y <*text:p*>) y son identificadas por etiqueta <*text:list-item*>. El otro elemento que nos podemos encontrar es un tipo especial que se muestra antes de la lista y se trata de su cabecera, éste puede contener uno o más párrafos y se identifica por la etiqueta <*text:list-header*>.

Tablas:

El elemento que representa a una tabla es *<table:table>*, se pueden encontrar en los documentos de texto y en las hojas de cálculo. Éstas están formadas por filas *<table:table-row>* que a su vez contendrá celdas *<table:table-cell>*. Será dentro de las celdas donde nos encontremos con contenido de texto que puede estar dentro de párrafos u otros elementos.

Tablas de contenido:

Este elemento nos aporta una tabla de contenidos que puede servir como indice y guía del contenido que nos encontramos en el documento. Se identifica por *<text:table-of-content>*, y en cualquiera de estas tablas nos encontramos una especificación de cómo ha estado generada la tabla esto está dentro del elemento *<text:table-of-content-source>* y después tendremos el contenido del índice representado por el elemento *<text:index-body>* (el texto lo encontraremos dentro de párrafos).

3.3.2 Fichero styles.xml

En este fichero se definen los estilos y fuentes predeterminados que se utilizan en un documento. Estos estilos no tienen porque ser utilizados en el documento, son los que serían comunes a todos los documentos que utilizan el formato ODF.

En el fichero content.xml también se declaran estilos y fuentes particulares del documento, y hacen referencia a los que son predeterminados y declarados en styles.xml, heredando sus propiedades e insertandoles algunas particularidades que los diferenciarían de los generales.

Para ver esta diferencia entre estilo particular y estilo predeterminado, podemos encontrar en el ejemplo de la figura 11 presentado en el apartado anterior:

<text:p text:style-name="P1">Esto es un párrafo en negrita</text:p>

<text:p text:style-name="Standard">Este es otro párrafo normal.</text:p>

La primera hace referencia a un estilo particular declarado en el fichero content.xml y el segundo es un estilo predeterminado declarado en styles.xml.

3.3.3 Fichero meta.xml

Este archivo incluye los metadatos del documento. Por ejemplo, el nombre del autor, la fecha de creación, la identificación de la última persona que lo modificó, la fecha de última modificación,

etc. Son los datos asociados al documento.

3.3.4 Fichero settings.xml

Fichero donde se guardan las propiedades del documento, ofreciendo información sobre el estado del documento que estamos realizando con respecto a la aplicación. Es decir, la posición del cursor en el momento de cerrar el archivo, el tamaño de visualización (factor de zoom), la posición de las barras de herramientas, etc. Estas propiedades no son contenido ni afectan a la disposición de éste en el documento.

3.3.5 Fichero mimetype.xml

Se trata de un archivo con una única línea que contiene el tipo MIME del documento. Por ejemplo en el caso de un documento de texto como el utilizado en los ejemplos, contendría la siguiente línea:

application/vnd.oasis.opendocument.text

Es donde realmente se guarda el tipo de documento que se ha generado. Su finalidad es la de identificar internamente el tipo de documento, a pesar de que nos modifiquen la extensión. Realmente la extensión del documento sólo le sirve al usuario para poder facilitarle la identificación del archivo.

3.3.6 Carpeta pictures

Esta carpeta contiene todas las imágenes del documento. Las imágenes realmente no se guardan dentro del documento content.xml, sino que se guarda una referencia a un fichero de esta carpeta. Estas referencias se identifican mediante el uso de la etiqueta <*draw:image*>. La mayoría de las imágenes se guardan en su formato original (GIF, JPEG, PNG), aunque los mapas de bits se convierten a PNG por cuestiones de tamaño.

3.4 Tipos de documentos soportados

El formato ODF incluye todos los tipos de documentos de oficina disponibles: documentos de texto *odt*, hojas de cálculo *ods*, presentaciones *odp*, gráficos *odg* y bases de datos *odb*.

La relación de tipos y extensiones para documentos es la siguiente:

Tipo de formato	Extensión del archivo	Tipo de MIME
Documento de texto	.odt	application/vnd.oasis.opendocument.text
Hoja de cálculo	.ods	application/vnd.oasis.opendocument.spreadsheet
Presentación	.odp	application/vnd.oasis.opendocument.presentation
Dibujo	.odg	application/vnd.oasis.opendocument.graphics
Gráfica	.odc	application/vnd.oasis.opendocument.chart
Fórmula matemática	.odf	application/vnd.oasis.opendocument.formula
Base de datos	.odb	application/vnd.oasis.opendocument.database
Imagen	.odi	application/vnd.oasis.opendocument.image
Documento maestro	.odm	application/vnd.oasis.opendocument.text-master

Fig. 12: Tipos MIME de los documentos OpenOffice

La relación de los tipos de plantillas existentes son:

Tipo de formato	Extensión del archivo	Tipo de MIME
Documento de texto	.ott	application/vnd.oasis.opendocument.text-template
Hoja de cálculo	.ots	application/vnd.oasis.opendocument.spreadsheet-template
Presentación	.otp	application/vnd.oasis.opendocument.presentation-template
Dibujo	.otg	application/vnd.oasis.opendocument.graphics-template

Fig. 13: Tipos MIME de las plantillas OpenOffice

3.5 Librerías de soporte al formato

Hemos podido ver que la arquitectura del formato OpenDocument es muy compleja, con sólo tres líneas de texto nos genera varios ficheros relacionados y comprimidos en un documento con su correspondiente extensión. Cada uno de estos ficheros tiene una estructura XML complicada.

Tal como se ha explicado al principio del documento, el objetivo es desarrollar una aplicación que dado un fichero de configuración y un documento realice la conversión del texto en archivos de audio según las opciones elegidas. Este proceso parece complicado, a priori, ya que para acceder al texto hay que acceder a la estructura XML del documento content.xml, con lo que primero habría que descomprimir el documento para poder realizar este proceso.

Por lo que he visto una necesidad de buscar la existencia de alguna librería que facilite este proceso. Primero he buscado en OpenOffice.org, donde he encontrado Apache OpenOffice SDK (AOO SDK) que sirve para crear extensiones de OpenOffice. De este SDK hay soporte para los lenguajes Java y C. Utiliza UNO (Universal Network Objects) para acceder a las aplicaciones de OpenOffice.

En principio no se pretende crear una extensión de OpenOffice en este proyecto, por lo que hay que encontrar otra librería que nos permita trabajar con estos documentos de manera ágil.

Buscando por librerías de ODF, nos encontramos con la librería AODL (An Open Document Library) que está completamente escrito en C# y que se puede usar en .net. Y nos encontramos también con la librería odf4j que está desarrollada en Java.

Esta última librería de Java se ha dejado abandonada y ha pasado a llamarse ODFDOM, pertenece al proyecto de ODF Toolkit Project de Apache.

Por motivos de diseño e implementación utilizaré la librería del proyecto ODF Toolkit. En el siguiente apartado explicamos de qué se trata esta librería y cuál es su uso.

3.5.1 Apache ODF Toolkit (incubating)

ODFDOM es una librería de libre distribución de OpenDocument Format (ODF). Su objetivo es proporcionar una manera fácil de crear, acceder y manipular archivos ODF sin que se tengan conocimientos detallados de la especificación.

La API ODFDOM sigue un enfoque por capas para acceder a los documentos. Estas capas son:

- **Capa ODF Package**: proporciona acceso a los recursos almacenados dentro del paquete ODF, tales como secuencias XML, imágenes u objetos incrustados.
- **Capa ODF XML**: Proporciona todas las características de un formato de *office*, tales como tablas, imágenes, numeración, etc. Esta capa está formada por dos *APIs* que representan dos visiones diferentes sobre las características:
    - *The low-level DOM API*: Da acceso a XML, las partes elementales de las características de *ODF schema*.
    - *The high-level Document API*: Proporciona una visión diferente de nivel más alto en las características de *ODF schema*. Esta API se preocupa de la usabilidad, escondiendo todos los detalles de la implementación de ODF XML del usuario, cubriendo escenarios frecuentes de usuario.

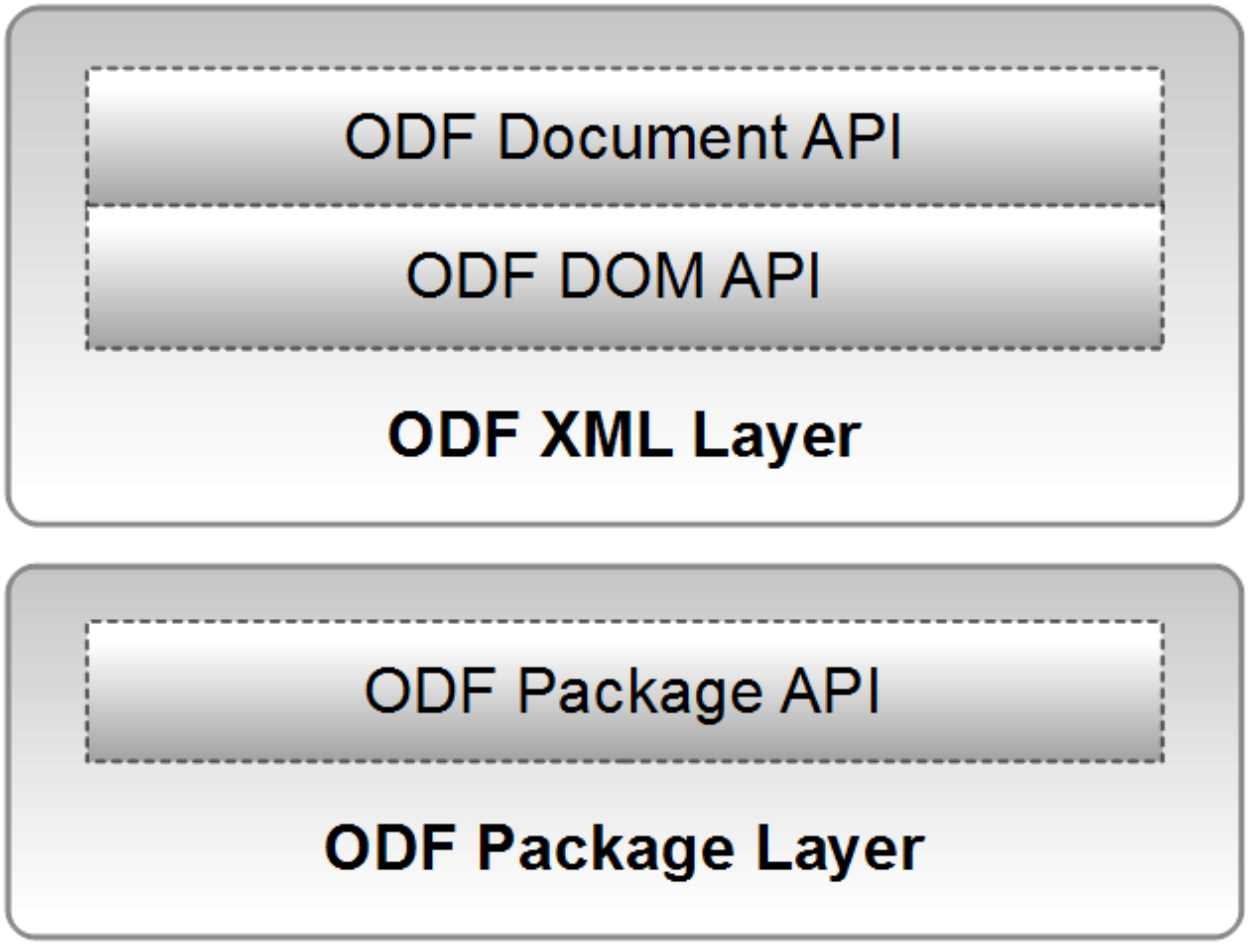


Fig. 14: Capas ODFDOM

La ODF DOM API da acceso a XML, las partes elementales de las características de ODF schema. Con esta API es fácil de manipular los nodos XML. Para cada ODF XML elemento y ODF XML atributo definido por la gramática ODF existe una única clase, proporcionando métodos para sus hijos.

Veamos con un pequeño ejemplo de ODF XML representando una tabla en ODF (sacado de http://incubator.apache.org/odftoolkit/odfdom/Layers.html):

```
<table:table table:name="Table - fruits vs. vegies">.
    <table:table-column table:number-columns-repeated="2"/>.
    <table:table-row>.
        <table:table-cell>.
            <text:p>Lemon</text:p>.
        </table:table-cell>.
        <table:table-cell>.
            <text:p>Orange</text:p>.
        </table:table-cell>.
    </table:table-row>.
    <table:table-row>.
        <table:table-cell>.
            <text:p>Carrot</text:p>.
        </table:table-cell>.
        <table:table-cell>.
            <text:p>Potato</text:p>.
        </table:table-cell>.
    </table:table-row>.
</table:table>.
```

Fig. 15: ***Representación tabla ODF***

Esta tabla en XML sería usada tanto para un documento de texto como para una hoja de cálculo.

La estructura de este XML será pasado a la estructura de la clase ODF DOM como:

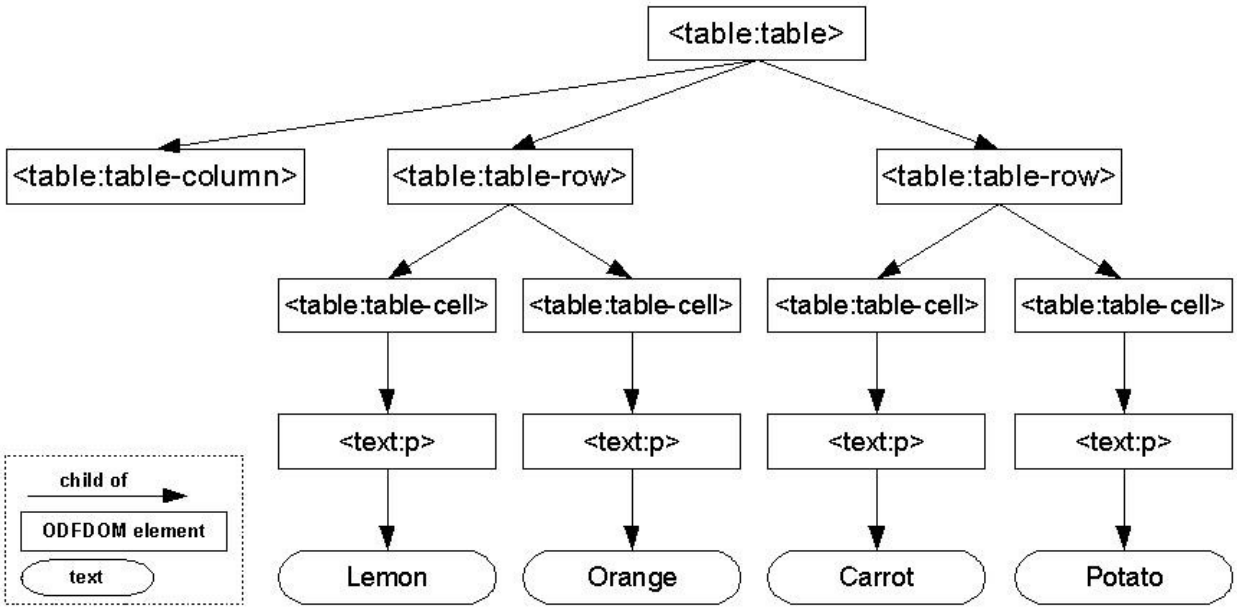


Fig. 16: Representación estructura ODF DOM

En Agosto de 2011 el proyecto ODF Toolkit pasó a ser Apache ODF Toolkit(incubating).

Aparece un subproyecto llamado *Simple API* (o también *Simple Java API for ODF*). Se trata de una herramienta fácil de usar, de alto nivel de Java API para crear, modificar y extraer información

de documentos ODF. Está también escrito en Java y se trata de una abstracción de alto nivel del nivel bajo ODFDOM API.

Se utilizará en el proyecto estas librerías de Apache ODF Toolkit para que nos facilite el acceso al texto del documento en el fichero content.xml, sin tener que realizar a mano la extracción de los ficheros contenidos en el zip.

Capítulo 4: Formatos de audio libre

4.1 Introducción

Un formato de archivo de audio es un contenedor multimedia que guarda una grabación de audio.

Lo que diferencia un archivo de otro son, entre otros, sus propiedades, si se trata de un formato libre (o abierto) o es propietario (pertenece a una empresa u organismo), cómo se almacenan los datos, sus capacidades de reproducción, y cómo puede utilizarse un archivo en un sistema de administración de archivos (etiquetado).

La ventaja de un formato libre es que puede ser utilizado por cualquiera sin tener que pagar una licencia o dinero para su uso, ya que es de libre uso. Además permite que se puedan modificar o estudiarlos, ya que su código fuente es de libre acceso.

Por contra los formatos cerrados o propietarios son creados con fines comerciales y se caracterizan porque su código fuente no es de libre acceso.

La manera de almacenar audio digital puede ser sin comprimir o comprimido para reducir el tamaño del formato.

4.2 Tipos de formatos

Existen diferentes tipos de formato según la compresión del audio.

Hay que diferenciar lo que es un formato de archivo y lo que es un códec[7]. El códec codifica y decodifica los datos del audio que son guardados en una archivo con un formato de audio específico. La mayoría de formatos de archivo de audio solo soportan un tipo de datos.

Hay tres grupos principales de formatos de archivo de audio:

- Formatos de audio sin comprimir: como WAV, AIFF o AU.
- Formatos sin pérdida (formato de audio comprimido sin perdida): como FLAC, MPEG-4 SLS, MPEG-4 ALS, MPEG-4 DST, Apple Lossless y WMA Lossless.
- Formatos con pérdida (algoritmo de compresión con perdida): como MP3, Vorbis y Musepack.

7 Es la abreviatura de codificador-decodificador.

Formatos de audio sin comprimir:

Normalmente están archivados como *.wav* en Windows y como *.aiff* en MAC. Son formatos flexibles para almacenar varias combinaciones de frecuencia de muestreo o tasa de bits, siendo adecuados para archivar grabaciones originales.

Existe un tipo llamado CDA (audio CD Track) que es un archivo pequeño que sirve como acceso directo a parte de los datos de un CD. BWF (Broadcast Wav Format) es el formato de audio estándar creado por la Unión Europea de Radiodifusión como sucesor de WAV, y permite el almacenamiento de meta-datos en el archivo que es usado por muchos programas profesionales de edición de audio en televisión y cine.

Formatos de audio comprimido sin pérdida (Lossless):

Requiere más tiempo de procesamiento que los formatos sin comprimir, pero son más eficientes ya que ocupan menos espacio. Los formatos sin comprimir codifican tanto audio como silencio con el mismo número de bits por unidad de tiempo, ocupando el mismo espacio que si tuviera voz. Pero en un formato comprimido el silencio no ocupa prácticamente nada. Los ratios de compresión son de más o menos 2:1. Estos formatos intentan reducir el tiempo de procesamiento manteniendo un buen ratio de compresión.

Formatos de audio comprimido con pérdida:

Estos formatos comprimen los datos descartando algunas partes. El proceso intenta minimizar la cantidad de datos que mantiene el archivo reduciendo su peso y por lo tanto su calidad. Para conservar gran parte de su calidad, los canales de audio que pierden son los que no captan el oído humano.

Ahora pasaremos a ver con más detalle algunos de estos formatos de audio libre de más uso en el mercado en los siguientes apartados.

4.2.1 Matroska

Es un formato contenedor de estándar abierto que puede contener una cantidad ilimitada de vídeo, audio, imagen o pistas de subtítulos dentro de un solo archivo. Sirve como formato universal para el almacenamiento de contenidos audiovisuales y multimedia. Permite reproducir el archivo tanto en ordenadores como en otros dispositivos con la suficiente potencia de procesamiento.

Tiene características similares a AVI, MP4 o ASF pero no se trata de un formato propietario por lo que no se pagan derechos de autor ni está sometido a limitaciones por copyright. Permite que los usuarios puedan realizar sus propios archivos gratuitamente.

Las extensiones para este tipo de archivos Matroska son .MKV para vídeo (con subtítulos y audio), .MKA para archivos solamente de audio, .MKS sólo para subtítulos y .MK3D para vídeo estereoscópico.

Aun siendo un software libre, Matroska no se limita a sistemas operativos de código abierto, como GNU/Linux, está pensado para poder ser usado en todos los sistemas operativos aunque sean propietarios, es decir, se quiere que se pueda usar en Windows, Mac OS X, Android o cualquier otro.

Como otros formatos de contenedores, hay que decir que por sí solo no especifica un formato de compresión. Por lo que diversas tecnologías de compresión de audio o vídeo (códecs) pueden ser utilizadas para codificar el audio o vídeo para luego ser almacenado en un solo archivo contenedor Matroska.

El proyecto fue anunciado el 7 de diciembre de 2002, como una bifurcación del Contenedor de Formato Audiovisual (MCF, en sus siglas en inglés). Los desarrolladores del proyecto Matroska creen que el uso del Meta Lenguaje Binario Extendible (EBML por sus siglas en inglés) les aporta una serie de ventajas, que incluye la posibilidad de ampliarlo en un futuro.

El formato ha sido diseñado desde cero para ser extendido y duradero. Los fundadores de Matroska tienen las siguientes metas (según comentan):

- Crear y documentar un moderno, flexible, y multi-plataforma contenedor multimedia.
- Establecer Matroska como la alternativa de código abierto a los contenedores existentes como AVI, ASF, MOV, RM, MP4, MPG (comparable a Ogg).
- Desarrollar un conjunto de herramientas para la creación, edición e implementación de los ficheros Matroska.
- Desarrollar librerías y herramientas para que las aplicaciones de software puedan usar Matroska.
- Trabajar con fabricantes de hardware para incluir apoyo Matroska en dispositivos audiovisuales y multimedia.

- Poner en marcha una serie de filtros DirectShow para la reproducción y creación de ficheros Matroska en sistemas Windows.

Las principales características de Matroska son:

- Una o varias pistas de audio alternativas.
- Posibilidad de Audio y Vídeo en VFR (Framerate Variable).
- Varios subtítulos, incluyendo SSA/ASS[8] con funciones avanzadas, y recientemente PGS.
- Varios capítulos en un mismo fichero.
- Streaming o vídeo visualizado en tiempo real mientras se descarga.
- Mejor tolerancia a errores respecto a AVI o MP4.
- Posibilidad de añadir archivos de cualquier tipo.
- Posibilidad de empotrar una imagen en el contenedor como muestra del contenido.
- Soporte para *tags*.

4.2.2 Ogg

Es un formato libre y abierto de contenedor, desarrollado y mantenido por la Fundación Xiph.Org, y está diseñado para proporcionar una difusión de flujo eficiente y manipulación de medios digitales de alta calidad.

Este formato puede multiplexar varios flujos independientes para audio, vídeo, texto y metadatos. La capa de audio es más comúnmente proporcionada por el formato Vorbis orientado a la música, pero otras opciones incluyen los códecs de compresión Opus, FLAC para audio sin pérdidas y OggPCM.

Desde 2007 la Fundación Xiph.Org recomienda que la extensión .ogg sea utilizada sólo para archivos de audio Ogg Vorbis, así como decidió crear un nuevo conjunto de extensiones de archivo y tipos de medios para describir los diferentes tipos de contenido:

- **.oga** : archivos de audio.
- **.ogv** : para vídeo con o sin sonido (incluyendo Theora).

8 Substation Alpha o Sub Station Alpha (abreviación SSA) y Advanced SubStation Alpha (abreviación ASS).

- **.ogx** : para multiplexado Ogg.
- **.ogg** : para archivos de audio Ogg Vorbis.

Debido a que es un formato libre, varios códecs de Ogg se han incorporado a reproductores de medios portátiles y receptores GPS de diferentes fabricantes.

Ogg no es solo un códec de vídeo o de audio, sino que es un contenedor que comprende audio, vídeo y subtítulos, y que permite reproducir el archivo tanto en ordenadores como otros dispositivos con la suficiente potencia de procesamiento.

Una ventaja técnica es que sus códecs han evolucionado técnicamente en su medio. Su calidad está relacionada a la compresión que se considere, utiliza un sistema de compresión inteligente que elimina las partes no audibles para el oído humano y parte del ruido ambiente, esto permite eliminar las distorsiones en las grabaciones antiguas. Conserva mejor el sonido cuando no hay diferencia de tasas de bits.

Ogg encapsula datos no comprimidos y permite la interpolación de los datos de audio y de vídeo dentro de un solo formato. Además es un contenedor orientado a la difusión de flujo multimedia, lo que significa que puede ser escrito y leído en un solo paso. Las características del flujo de bits de Ogg según Wikipedia son:

- Orientado a la difusión de flujo multimedia.
- Usa aproximadamente de 1 a 2% del ancho de banda del flujo de bits, para la marca del límite del paquete, entramado de alto-nivel, sincronización y búsqueda.
- Especificación de la posición absoluta dentro de la muestra del flujo original.
- Mecanismo simple para una fácil corrección limitada.
- Detección de corrupción, acceso aleatorio a los datos en posiciones arbitrarias en el flujo de bits.

Breve historia:

El proyecto Ogg Vorbis fue iniciado en 1993 por el programador Chris Montgomery, con unos intentos de programar un paquete de compresión de audio simple durante un fin de semana.

El ogg original fue desarrollado como prácticas en lenguajes informáticos relativamente poco conocidos por un programador durante su periodo de aprendizaje. Christopher Montgomery no era

remunerado por la empresa privada, se inició trabajando de forma autodidacta como parte de un proyecto personal que se hizo más grande en 1993 y que, originalmente, fue nombrado "Squish". Debido a que este nombre era una marca registrada, le sugirieron cambiarlo por "OggSquish". El nombre Ogg, fue usado después para designar al contenedor, y proviene de una maniobra táctica del juego online Netrek.

Actualmente Ogg es parte de un proyecto multimedia más grande de la Fundación Xiph.Org.

4.2.3 Vorbis

Códec de audio digital general con pérdidas, desarrollado por la Fundación Xiph.Org, utiliza el formato de archivo Ogg. Es un códec de audio perceptivo de fines generales previsto para permitir flexibilidad máxima del codificador. También conocido por Ogg Vorbis por ser el códec más comúnmente encontrado en el contenedor Ogg.

El algoritmo de compresión con pérdida produce una menor cantidad de información, ya que se trata de un procedimiento de codificación que tiene como objetivo eliminar una cantidad de información considerada como irrelevante para disminuir el volumen de datos. Es comparable con AAC en la mayoría de *bitrates*[9].

Vorbis también está pensado para frecuencias de muestreo bajas desde telefonía y hasta alta definición, y una gama de representaciones de canales.

Aplicaciones de *streaming* audio como Spotify utilizan el formato Ogg en versión premium.

Breve historia:

Es el primer códec desarrollado como parte de los proyectos multimedia de la Fundación Xiph.Org.

Ogg Vorbis y la Fundación Xiph.Org fueron creadas para proteger la multimedia en Internet del control de intereses privados (como MP3 que tienen derecho a cobrar regalías).

El formato del bitstream para Vorbis I fue congelado el 8 de mayo de 2000; todos los archivos creados desde esa fecha seguirán siendo compatibles con futuros lanzamientos de Vorbis.

En Julio de 2002 la versión 1.0 fue anunciada, adjuntando un agradecimiento por el apoyo recibido y explicando por qué es necesario el desarrollo de códecs libres.

9 Tasa de bits por segundo que tiene un archivo multimedia (Audio o Video).

4.2.4 FLAC (Free Lossless Audio Codec)

Es un códec de audio que permite comprimir el audio sin pérdidas de tal manera que el archivo se ve reducido sin que se pierda ningún tipo de información. Un audio reducido con este algoritmo puede ver reducido su tamaño entre un 50 a 60%. Estos archivos tienen la desventaja que su tamaño es superior al que se obtendría con un algoritmo de compresión con pérdida.

La extensión de estos ficheros es *.flac* aunque se pueden encontrar como *.fla*. Este formato cuenta con soporte para etiquetado de metadatos, inclusión de la portada del álbum, y la búsqueda rápida.

FLAC usa la predicción lineal para convertir las muestras en series de pequeños números no correlativos (se conoce también como "residuos"), que eficientemente se almacenan utilizando la codificación Golomb-Rice. Para los silencios usa la codificación por *longitud de pista* (RLE "Run-Length Encoding") para muestras idénticas.

Es el formato ideal para realizar copias de seguridad de CD, ya que nos permite reproducir la información del original y podremos recuperarla en caso de tener problemas.

FLAC no permite muestras en coma flotante, deben ser en coma fija. Admite cualquier resolución PCM de 4 a 32 bits por muestra, y cualquier tasa de muestreo desde 1 hasta 655350 kHz, y canales de audio de 1 hasta 8.

Este formato utiliza sumas de comprobación de redundancia cíclica para identificar tramas de datos corruptas cuando es usado en un protocolo de flujo de audio, además se guarda en la cabecera de metadatos STREAMINFO un cálculo hash de MD5 del audio *raw* PCM.

Para el etiquetado de los archivos FLAC usa el mismo sistema Vorbis comments.

El proyecto FLAC incluye:

- El códec para la codificación de la información de audio.
- El formato del contenedor.
- Editor de metadatos para ficheros flac por medio de línea de comandos.
- Dispone de una biblioteca que permite programar compresores y reproductores, además de incluir los metadatos de los archivos.
- LibFLAC++ ofrece funcionalidades adicionales respecto a los metadatos, a los ofrecidos

por la libFLAC.

- Mediante el uso de la librería libFLAC permite codificar y decodificar los FLAC streams mediante la línea de comandos.
- Plugins para diferentes reproductores de audio.

Breve historia:

El proyecto fue iniciado y desarrollado por el programador Josh Colson. En Enero de 2001 se lanza la versión 0.5 de la implementación de referencia, y en Julio de 2001 la versión 1.0.

La Fundación Xiph.Org y el proyecto FLAC anunciaron en Enero de 2003 la incorporación de este códec bajo la bandera de Xiph.org.

En Mayo de 2013 lanzan la versión 1.3.0. El desarrollo ha sido trasladado al repositorio de Xiph.org.

4.2.5 Formato AU (.au)

Uno de los primeros sistemas para reproducir y almacenar audio con calidad aceptable fue el formato AU (significa Audio for Unix). Fue desarrollado por la empresa Next Machine Sound System. Se utiliza como formato de audio del sistema Unix de Sun Microsystems, también es el estándar acústico para el lenguaje de programación Java.

Se codifica con 8 bits a una frecuencia de muestreo de 8000 Hz.

Es poco conocido fuera del ambiente UNIX y casi desconocido por el público en general, por ser un formato nativo de las estaciones de trabajo Sun y similares.

4.2.6 Formato AIFF

Es un estándar de formato de audio usado para almacenar datos de sonido en computadoras personales.

Viene de la evolución del viejo formato IFF (siglas en inglés de Interchange File Format), de Electronic Arts, usado en las antiguas computadoras Amiga y actualmente es muy utilizado en los equipos comercializados por Apple Macintosh.

Los archivos aiff no se comprimen, almacenándose los datos en big-endian y emplea una *modulación por impulsos codificados* (PCM), lo que hace que el tamaño de los mismos sea

realmente grande, por lo que requieren de gran espacio de almacenaje, hay que tener en cuenta que necesita de aproximadamente 10 Mb para un minuto de audio.

Las extensiones de este formato son *.aiff* o *.aif*. Pero además existe una variante del formato que incluye compresión, denominada como AIFF-C, cuya extensión es *.aifc*.

4.3 Librerías de desarrollo

Ahora veremos una muestra de cuales son las librerías que hemos encontrado para el uso de audio en una aplicación.

He encontrado librerías muy enfocadas al formato de audio con que se quiere trabajar, y no a un nivel más general, aunque hay alguna librería que sí que permite el uso de diferentes formatos.

La primera librería encontrada que voy a comentar es *VorbisSPI* que se trata de una *interface de proveedor de servicio* en Java (del inglés, Java Service Provider Interface), que nos da soporte en plataformas Java para el formato de audio OGG Vorbis. Está basado en la librería Java JOrbis (es un decodificador en Java de Ogg Vorbis).

También existe una librería para el formato FLAC para Java, que se llama jflac. Permite a los desarrolladores de Java escribir programas que utilizan los archivos y los algoritmos FLAC.

Existe una librería que se llama Java Layer que mejora el soporte VBRI VBR. Decodifica/reproduce/convierte MPEG 1/2/2.5 Layer 1/2/3 en tiempo real para plataforma Java. Tiene licencia LGPL, usa menos de 1% de CPU, se trata de un proyecto de código abierto.

La mayoría de librerías están destinadas al uso de un formato, orientadas sobretodo a la reproducción de archivos de esos formatos, tanto a nivel local como a través de la red, pudiendo reproducir también datos que vengan de una web (como el streaming de una radio), o a la conversión de formatos.

Existe una librería que viene dentro del SDK de Java, o sea que no hay que descargarla como un extra, que se llama Java Sound API. Proporciona soporte de bajo nivel a operaciones de audio tales como la reproducción de audio y captura (grabación), mezcla, secuenciación MIDI y síntesis MIDI en un framework extensible y flexible. El paquete se llama 'javax.sound.sampled'. Esta API no asume una configuración de hardware de audio especificas, está diseñado para permitir que diferentes componentes de audio sean instalados en el sistema y accedidos por la API. Soporta funcionalidades comunes como la entrada y salida de una tarjeta de sonido, así como la mezcla de múltiples fuentes de audio.

En la siguiente imagen[10] se puede ver una arquitectura de audio típica:

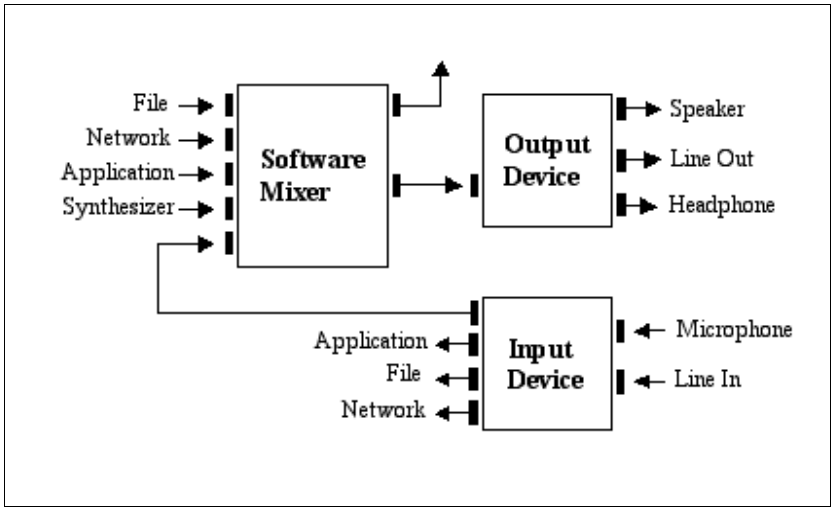


Fig. 17: Una típica arquitectura de audio

Esta API soporta las siguientes características:

- Formatos de audio: AIFF, AU y WAV
- Formatos de archivo de música: tipo MIDI 0, tipo MIDI 1, y *Formato de Música Enriquecida* (RMF siglas del inglés, Rich Music Format)
- Formatos de sonido: de 8 y 16 bits, tanto en mono como estéreo, con velocidades de muestreo de 8 hasta 48 kHz.
- MIDI tabla de ondas síntesis y secuenciamiento en software, y acceso a los dispositivos de hardware MIDI.
- Un mezclador en software que puede mezclar y ofrece hasta 64 canales de audio digital y música MIDI sintetizada.

La elección de la API a utilizar en la aplicación a desarrollar dependerá en gran medida de la herramienta utilizada para Text-To-Speech, pero las mejores opciones son Java Sound API y Java Layer.

En el siguiente capítulo haremos un estudio de las herramientas Text-To-Speech encontradas.

10 Imagen obtenida de http://docs.oracle.com/javase/tutorial/sound/

Capítulo 5: Estudio de Text-To-Speech

5.1 Introducción

La conversión de texto-voz es la generación por medios automáticos de una voz artificial que genera el sonido que produciría una persona al leer un texto cualquiera en voz alta o una voz artificial. Es decir que son sistemas capaces de convertir textos en voz sintética. Los conversores de texto-voz son conocidos también con las siglas CTV o por las siglas en inglés TTS (Text To Speech).

Qué requisitos deben cumplir los conversores TTS:

- La voz sintética que producen debe resultar natural e inteligible.
- Se debe realizar completamente de forma automática la síntesis del habla, sin que se tenga que realizar ningún tipo de reajuste manual en ninguna parte del proceso.
- El texto que se pasa al sistema ha de ser un texto cualquiera, no puede ser que esté *amañado*.

Las fases de conversión de texto a voz son:

- En una primera fase se realiza una representación lingüística simbólica, que sigue los siguientes pasos:
    1. Normalización del texto: Se convierte todo el texto a una forma textual convencional (*tokenización*).
    2. Conversión fonética: asignación de transcripciones fonéticas a cada palabra (a esto se le llama conversión *texto-fonema*[11]).
    3. División prosódica: se divide el texto en unidades sintagmáticas, proposiciones y frases.
- Después el sintetizador coge la representación lingüística simbólica de entrada y la convierte en voz sintética.

11 En inglés es text-to-phoneme cuyas siglas son TTP.

5.2 Voz sintética (Síntesis de habla)

Se trata de una voz artificial (no grabada) que está generada por un proceso de sintetización del habla, esto es la producción artificial de habla humana.

Un sistema usado con este propósito recibe el nombre de sintetizador de habla y puede llevarse a cabo en software o en hardware.

Hay sistemas que en lugar de producir voz a partir de texto lo hacen a partir de representación lingüística simbólica en habla.

La calidad que podemos obtener en una voz vendrá dada por su inteligibilidad y su naturalidad.

Un sintetizador de texto a voz (TTS) es un sistema computarizado que debe ser capaz de leer cualquier texto en voz alta, ya haya sido introducido en el ordenador por un operador o escaneado y enviado a un sistema de reconocimiento óptico de caracteres (OCR).

5.2.1 Problemas de la voz sintética

Los problemas con que nos encontramos con la voz sintética son[12]:

- Los usuarios ofrecen rechazo por la falta de naturalidad y su parecido a un robot.
- Los conversores de texto-voz (CTV) normalmente generan voz de hombre, el motivo de esto se puede ver de la siguiente manera:
    - Hasta hace poco las personas que trabajaban en los laboratorios eran hombres y para sus experimentos utilizaban su propia voz.
    - La voz masculina ofrece mejor calidad de sonido que la femenina.
    - La forma de onda en la voz de mujer tiene un componente de oscilación no periódica que resulta más notable que la del hombre, y resulta difícil de modelar adecuadamente.

5.3 ¿Cómo funciona esta tecnología?

Está basado en un sistema formado por dos partes, por un lado el front-end y por otro el back-end.

¿Qué quiere decir esto? Pues la parte del front-end toma como entrada el texto y produce una representación lingüística fonética. Entonces la parte del back-end toma esta representación

12 Información obtenida de Wikipedia, http://es.wikipedia.org/wiki/S%C3%ADntesis_de_habla

lingüística fonética y produce una forma de onda sintetizada.

Las tareas principales del front-end son dos. La primera, tomar el texto y convertir las partes problemáticas como números y abreviaturas en palabras equivalentes, a esto se le llama normalmente *normalización de texto o procesado*. Asignando una transcripción fonética a cada palabra, dividiendo y marcando el texto en varias unidades prosódicas, como frases y oraciones. A este proceso se le conoce, como hemos dicho antes, con el nombre de conversión de texto a fonema.

La tarea de la parte del back-end es la de coger la representación lingüística simbólica y convertirla en sonido, a esta parte se le llama normalmente *sintetizador*.

5.3.1 Desafíos del cliente (front-end)

Uno de los principales desafíos lo encontramos en el proceso de normalizar textos que no suele ser simple, debido a que están llenos de homográfos, números y abreviaturas que se deben transformar en representación fonética.

Muchos sistemas de texto a voz no generan representaciones semánticas de los textos, usan varias técnicas heurísticas para estimar la manera correcta de hacer desaparecer las ambigüedades, buscando palabras vecinas y usando estadísticas de la frecuencia de aparición de las palabras.

Otro gran problema es la conversión en palabras de los números, ya que no se lee igual el número del DNI que el número de un teléfono.

Lo mismo pasa con las abreviaturas que no siempre significan lo mismo, con lo que no hay una traducción directa. Los sistemas con front-end inteligentes pueden hacer estimaciones acertadas sobre como tratar abreviaturas ambiguas, mientras que otros pueden hacer lo mismo en los restantes casos, por lo que en ocasiones pueden dar lugar a resultados que parecen cómicos.

Otro desafío lo encontramos en el proceso de conversión texto a fonema o grafema a fonema[13], en otras palabras sería determinar la pronunciación de una palabra basándose en su pronunciación.

Una aproximación es la basada en diccionario, en este caso el programa almacena un diccionario que contiene todas las palabras de la lengua y su pronunciación. La determinación de la correcta pronunciación de cada palabra consiste en buscar la palabra en el diccionario y reemplazarla por

13 Es el término usado por los lingüistas para describir sonidos distintivos en una lengua.

la pronunciación especificada en el diccionario. La ventaja que tiene esta técnica es que es rápida y precisa, pero por contra si una palabra dada no aparece en el diccionario falla, y la dimensión del diccionario se va comiendo los recursos del sistema.

Otra aproximación para convertir texto en fonemas es la aproximación basada en reglas, en las que estas reglas de pronunciación de las palabras se aplican a palabras para extraer sus pronunciaciones basadas en su forma escrita. La ventaja de esta técnica es que funciona con cualquier entrada, pero su complejidad de reglas crece sustancialmente a medida que se van teniendo en cuenta ortografías y pronunciaciones irregulares.

Es por eso que cualquier sintetizador de voz usa una combinación de estas dos técnicas.

5.4 Breve historia

Ya mucho antes del desarrollo de procesado de señal, que se intentan crear máquinas que produjesen la voz humana. Ya viene del siglo XI que se empezaron a crear ejemplos tempranos de "cabezas parlantes".

En 1779, el científico Christian Gottlieb Kratzenstein construyó modelos del tracto vocal que podría producir las cinco vocales largas.

En 1837 Charles Wheatstone creó una "máquina parlante" basada en el diseño de von Kempelen, y en 1857 M. Faber construyó la máquina “Euphonia”. En 1923 Paget retomó el diseño que hizo Wheatstone.

VOCODER fue desarrollado en los laboratorios Bell Labs, en los años 30, se trataba de un analizador y sintetizador de voz, y fue desarrollado como un codificador de voz para telecomunicaciones.

Parece ser que los primeros sintetizadores de voz sonaban como si fueran robots y muchas veces inteligibles. Pero la calidad obtenida de la voz sintetizada ha mejorado muchísimo gracias a los sistemas de síntesis contemporáneos, y que en muchas veces no se distingue del habla humana. Incluso el mejor sintetizador electrónico está limitado por la calidad del transductor que produce el sonido.

No fue hasta el final de la década de 1950 que se creó el primer sistema de síntesis computarizado, y no fue hasta 1968 que se finalizó el primer sistema completo de texto a voz, y desde entonces no se ha parado de hacer avances en estas tecnologías.

5.5 Tecnologías de síntesis

Como ya hemos visto, las características de la calidad de un sintetizador de voz son la naturalidad e inteligibilidad. La naturalidad se refiere hasta que punto se parece a la voz de una persona real, y la inteligibilidad se refiere a la facilidad de poder entender la salida producida. Un buen sintetizador debe ser bueno en ambas características.

Hay dos tecnologías utilizadas para generar el habla sintética, que son, la síntesis concatenativa y la síntesis de formantes.

La síntesis concatenativa se basa en la concatenación de segmentos de voz grabados.

La síntesis de formantes no usa muestras de voz humana, sino que la salida se crea usando un modelo acústico. Muchos de estos sistemas generan habla robótica y de apariencia artificial, no pudiendo confundirse con la voz humana. No siempre la naturalidad máxima es el objetivo de un sintetizador de voz.

La síntesis de formantes puede ser muy inteligible, incluso a altas velocidades. La alta velocidad es la que usan los discapacitados visuales para utilizar con mayor fluidez los ordenadores. Al tratarse de programas más pequeños que no necesitan bases de datos de muestras de voz, pueden usarse en sistemas embebidos.

Cabe decir que cada sintetizador es el resultado de una particular y original imitación de la capacidad de lectura humana, sometida a las limitaciones tecnológicas e imaginativas que son característicos de la época de su creación.

5.6 Posibles aplicaciones de los sistemas TTS

Las aplicaciones en las que encontramos la síntesis de voz se han ido incrementando con el paso del tiempo y convirtiendo en una herramienta de tecnología asistiva, permite retirar barreras ambientales para personas con diferentes tipos de discapacidades. Una de las mayores aplicaciones ha sido en el uso de lectores de pantalla para personas con discapacidad visual. Pero estos sistemas de texto a voz lo utilizan personas con otros problemas como la dislexia o con otras dificultades para la lectura (en estos casos nos permiten seleccionar lecturas de letras, silabas, palabras y frases a velocidad lenta).

Las aplicaciones de texto a voz actualmente están también desarrollándose fuera del mercado de la discapacidad como por ejemplo la interacción con los dispositivos móviles a través de interfaces de procesamiento del lenguaje natural (combinan la síntesis de voz con el reconocimiento de voz),

estas interfaces vienen incluidas por defecto en los sistemas Android.

Las posibles aplicaciones de los sistemas TTS de alta calidad son realmente numerosas, podemos poner los siguientes ejemplos:

- En los servicios de telecomunicaciones: Los servicios TTS hacen posible acceder a información textual por teléfono. Sabiendo que la mayoría de las llamadas en realidad requieren muy poca interactividad, esta perspectiva es digna de ser considerada. Los textos podrían ir desde simples mensajes, tales como los eventos culturales locales que no nos podemos perder (cine, teatro, exposiciones, etc.), hasta bases de datos enormes que difícilmente pueden ser leídos y almacenados con voz digitalizada.

- Enseñanza de idiomas: en combinación con un sistema de aprendizaje asistido por ordenador, si juntamos la síntesis de alta calidad ofrecería una ayuda muy útil para aprender nuevos idiomas. Pero debido a la calidad que estos sistemas comerciales ofrecen no se utilizan habitualmente.

- Ayuda a personas con discapacidad: discapacidades en la voz originadas en trastornos motores, pueden tener una ayuda inestimable de estos sistemas. Con la ayuda de un teclado con un diseño especial y un programa de ensamblaje rápido de sentencias, la voz sintética se puede reproducir en pocos segundos remediando estos impedimentos, un ejemplo muy conocido de la utilización de este sistema es el de Stephen Hawking. En el mercado también hay muchas aplicaciones para las personas ciegas, como programas que transmiten mediante voz toda la información que aparece en un monitor.

- Libros hablados y juguetes: hay juguetes para niños que generan voz, de forma que les permite aprender debido a que reaccionan a los estímulos de estos. También en los libros para niños que van contando la historia a partir de los dibujos que el niño ve en el libro.

Una vez visto lo que sería la conversión de texto a voz, veamos cuál es la situación actual del mercado de aplicaciones que nos ofrecen esta utilidad de conversión.

5.7 Aplicaciones TTS

Hoy en día nos encontramos con páginas Web que nos permiten introducir un texto y nos lo convierten a voz (http://www.oddcast.com/home/demos/tts/tts_example.php?sitepal), también existen aplicaciones de escritorio que permiten a las personas con discapacidad visual a que puedan utilizar el ordenador facilitándoles la información que aparece en el escritorio de forma

verbal (un ejemplo de esto sería JAWS[14]).

También existen aplicaciones que te leen los documentos con apretar un botón (como por ejemplo ReadSpeaker[15], es de pago).

Ahora veremos algunas aplicaciones del mercado que nos hacen la conversión a voz de un documento.

5.7.1 Documentos

Hay diversos programas en el mercado que cogen un documento o fichero de texto y lo convierten en audio permitiendo su almacenamiento en un fichero o simplemente su reproducción instantánea.

Zamzar[16]

Nos convierte documentos (doc, pdf, txt y más) a mp3, para ello hay que acceder a su página web, han creado esta facilidad para las personas con dificultades para leer y también para que las personas puedan oír los documentos, *emails*, ficheros pdf cuando están fuera de camino a algún lado. Para realizar la conversión hay que entrar en su página enviar el fichero o URL, introducir el formato al que queremos convertir, ingresar nuestro *e-mail* y pulsar el botón de convertir.

Txt2mp3mac[17]

Convierte ficheros de texto en mp3 en los sistemas Mac OS. Es útil para crear "libros de audio" (audiobook) de los *ebooks*, entre otros. Utiliza Mac OS X speech synthesizer por defecto para realizar el text-to-speech y el paquete LAME v3.98 para la conversión a mp3. La manera de trabajar que tiene es párrafo a párrafo creando ficheros temporales mp3 y al final del proceso los junta en el mp3 final y borra todos los temporales creados.

SodelsCot Profesional[18]

Se trata de una aplicación TTS de pago con un periodo de prueba de 7 días. El sintetizador de voz puede procesar y leer textos desde editores, navegadores web, libros digitales, correos electrónicos y, por supuesto, cualquier documento de un procesador de texto. Ofrece dos voces en castellano. Permite personalizar la pronunciación de palabras y abreviaturas que no están

14 La aplicación se puede descargar en página http://www.freedomsci.de/serv01esn.htm
15 Se puede obtener más información en http://www.readspeaker.com/es/readspeaker-docreader/
16 http://www.zamzar.com/
17 http://code.google.com/p/txt2mp3mac/
18 http://www.sodels.com/sodelscot.htm

preestablecidas por defecto. Según comentan es sencillo de usar, en entorno Windows. Los textos que son procesados pueden ser exportados como archivo de audio y codificados en formato MP3 o WAV. Es muy útil para los estudiantes de idiomas.

Pdftospeech[19]

Aplicación para teléfonos móviles, permite al usuario coger un pdf de la tarjeta de memoria para poderlo oír por el altavoz del móvil utilizando auriculares. La aplicación está a la venta en Android Market.

5.7.2 Navegadores

Mediante extensiones que se instalan en el navegador, permiten convertir una página o trozo de texto marcado a voz.

SpeakIt!

Extensión para el navegador Chrome. Lee el texto seleccionado mediante la tecnología TTS, con detección automática de idioma. Dispone de más de 50 idiomas. Pero no funciona con las páginas encriptadas (https://) .

Texto a Voz 1.10

Es uno de los mejores *add-on* para navegadores Mozilla Firefox. Es suficiente con seleccionar el texto que queremos oír y hacer clic en el botón que se encuentra en la parte inferior derecha, y podremos escuchar el texto.

Chrome Speak

Se trata de otra extensión para el navegador Chrome. Esta extensión lee el texto seleccionado, también permite detener la locución cuando queramos, útil en textos largos y deseamos dejar de oír la voz. Dispone de una pantalla de configuración en la que podremos, entre otras cosas, seleccionar la voz y el idioma.

Evernote Clearly

Se trata de una extensión para el navegador Chrome de pago. Se puede obtener más información en la dirección http://thenextweb.com/apps/2012/11/27/evernote-partners-with-ispeech-adds-text-to-speech-reading-to-its-clearly-extension-for-chrome/

19 http://www.findbestopensource.com/product/pdftospeech

5.7.3 Páginas Web

Disponen de cuadros para introducir texto y convertirlo en audio, algunas permiten seleccionar el idioma y también permiten seleccionar el tipo de salida (a fichero con su formato, o a voz directamente), otras sirven de ayuda en los estudios de idiomas.

VozMe[20]

Hay un textBox en la pantalla donde se inserta el texto que queremos traducir a voz, dispone de los idiomas Catalán, Castellano, Italiano, Hindi, Portugués e Inglés. Permite la generación de un fichero mp3 o simplemente la reproducción por los altavoces.

Permite descargar un plugin para Wordpress, también nos explica como insertar esta facilidad en nuestras propias páginas Web con diferentes ejemplos.

QR voice[21]

Es una Web interesante a pesar de que no nos permite reproducir directamente el texto introducido. Lo llamativo de esta página es que te genera un código QR que lo podemos escanear y acceder a la dirección que nos da el código para oír el texto. Permite la selección de diversos idiomas para la reproducción del mensaje. Nos permite dejar mensajes verbales al público (para anuncios publicitarios), o a alguien en particular.

Howjsay[22]

Se trata de un diccionario fonético online de pronunciación inglesa. Actualmente cuenta con casi 13.000 entradas. Sirve para palabras o frases, no textos.

ImTranslator[23]

Permite teclear un texto y te lo convierte en voz. En su versión gratuita, solamente tiene capacidad para convertir 1.000 caracteres de una sola vez. La voz utilizada para “leer” el texto es de buena calidad y ofrece la opción de traducir el texto. La traducción se hace directamente de texto a texto para que el programa le dé voz, posteriormente, en el idioma que se eligió.

Muy útil para practicar la pronunciación.

20 http://vozme.com/index.php?lang=es
21 http://qrvoice.net/
22 http://www.howjsay.com/
23 http://text-to-speech.imtranslator.net/speech.asp?dir=es

5.7.4 Editores de texto

Actualmente muchos editores de texto permiten añadir unas *extensiones* o complementos que permiten convertir el texto en una locución hablada. Cada programa puede funcionar de una manera diferente, algunos necesitan que tengamos instalados otras aplicaciones con sus bases de datos (que hacen de back-end) para que produzcan la voz a partir de la información que generan (en estos casos el programa funcionaría como un front-end).

Lecturer[24]

Se trata de un *eBook reader* para dispositivos portátiles, a parte de enseñar el texto de una manera fácil de leer con margenes y espaciados ajustables, también dispone de la habilidad de leerlo en voz alta. Para la voz utiliza el sistema de Loquendo TTS (que se usa en los navegadores GPS TomTom Go).

Read Text[25]

Es una extensión para OpenOffice que permite que un programa o aplicación Web lea texto de Writer, Calc, Draw, Impress, Web Writer para convertirlo en voz.

WordTalk[26]

Se trata de un plugin para Microsoft Word, está diseñado para personas con dificultades para leer y escribir, es bueno tener la ayuda de la lectura en voz alta. Está desarrollado para las versiones de Microsoft Word (desde Word 97 hasta Word 2013).

Sus principales características son: habla el texto del documento; Resalta el texto a medida que lo lee; Se puede elegir entre todo el documento, párrafo, frase, o sólo una palabra a leer; Cambiar la velocidad y la voz del habla; Permite convertir el texto y grabarlo como fichero .wav o .mp3.

Speak Text[27]

Lee textos en alto y los convierte a ficheros mp3. Se puede integrar con MS Word, WordPerfect, OpenOffice, con Explorer, permitiendo oír los textos directamente. Útil para los que estudian idiomas ya que permite cambiar la velocidad de la lectura para poder practicar.

OO Text To Speech 0.1[28]

24 http://code.google.com/p/lecturer/
25 http://aoo-extensions.sourceforge.net/en/project/Read_Text
26 http://www.wordtalk.org.uk/Home/
27 http://www.speaktext.com/index.htm
28 http://linux.softpedia.com/get/Text-Editing-Processing/Word-Processors/OO-Text-To-Speech-10204.shtml

Se trata de una macro para OpenOffice de text-to-speech. Es un analizador de sílaba, que mediante un motor de lectura puede leer un documento y traducirlo a un mensaje vocal.

5.8 Librerías para TTS

Ahora expondré unas cuantas librerías de las que he encontrado, que son de utilidad para realizar una aplicación TTS. Hay algunas librerías que son de pago, por lo que no las explicaré, en algunos de los casos lo que se tiene que pagar es el acceso a sus servidores de TTS donde realizan la conversión del texto a audio.

Lo ideal es disponer de un servidor donde obtener la conversión a voz, en estos casos podemos disponer de más idiomas. Como hemos visto al principio de este estudio de TTS, esta tecnología se compone de dos partes diferenciadas, el front-end y el back-end, las bases de datos con las voces (pronunciaciones) pueden ocupar mucho espacio. Para cada uno de los idiomas que queramos ofrecer traducción tendremos que disponer de la equivalencia de las palabras con sus fonemas y después de la base de datos que nos hace la traducción de los fonemas a su voz.

5.8.1 Java Speech API

Es una implementación independiente del estándar JSAPI 2. Nos ofrece un framework básico que puede ser usado por un JSAPI 2 compatible para acceder a motores de habla.

Se trata de una API hecha en Java que permite a las aplicaciones Java incorporar tecnología de voz en sus interfaces de usuario. Soporta dos tecnologías de voz: síntesis de voz y reconocimiento de voz[29].

5.8.2 FreeTTS

Es un sintetizador de voz digital escrito totalmente en Java. Se basa en Flite, y es una implementación de Java Speech API de Sun.

Esta librería es muy práctica para la síntesis de voz pero sólo tiene disponible voces en Inglés y no hay manera de incorporar nuevas voces en otros idiomas, por esta razón no es factible su uso en la aplicación que debemos desarrollar.

29 En Inglés se conoce también por “automatic speech recognition”, sus siglas ASR

5.8.3 eSpeak

Es un software sintetizador de voz de código abierto compacto para Linux, Windows y otras plataformas. Utiliza el método de síntesis de formantes (explicado al principio del capítulo), proporcionando muchos idiomas en poco espacio.

eSpeak proporciona dos métodos de síntesis: el original sintetizador eSpeak y el sintetizador de Klatt. Además, eSpeak puede utilizarse como front-end, proporcionando la traducción del texto-fonema y prosodia, voces MBROLA diphone.

Está disponible como:

- Un programa que se ejecuta desde la linea de comandos (Linux y Windows) para dar voz al texto desde un fichero o desde *stdin*.
- Una versión de librería compartida para que sea usado por otros programas (en Windows a través de una DLL).
- Una versión SAPI5 para Windows.
- eSpeak se ha llevado a otras plataformas que incluyen Android, Mac OSX y Solaris.

Características:

- Incluye diferentes tipos de voces, a las que se les puede modificar sus características.
- Puede producir salida de voz como un fichero WAV.
- Puede soportar SSML (Speech Synthesis Markup Language) y también HTML.
- Tan sólo dos Mbytes ocupa el programa y su información, incluyendo diferentes idiomas.
- Como hemos dicho antes se puede usar como front-end para las voces de MBROLA diphone. Convierte texto a fonemas con información del tono y longitud.
- Puede traducir texto en códigos de fonemas, por lo que se podría adaptar como front-end de otro motor de síntesis de voz.
- Potencial para otros idiomas.
- Las herramientas de desarrollo están disponibles para la producción y adaptación de datos de fonemas.

- Desarrollado en C.

Para poder utilizar este programa y librería, en Windows, hay que descargarse setup_espeak-1.48.04.exe y setup_espeakedit-1.48.03.exe, y ejecutarlos para realizar la instalación. Una vez realizada la instalación ya se podrá utilizar, es conveniente poner el acceso directo a su ubicación en el path.

5.8.4 iSpeech

Se trata de una plataforma comercial de pago. Hay disponibles SDK para diversas plataformas como Java, .NET, PHP, Android, iPhone, entre otros. La que nos resultaría interesante es el Java SDK.

Nos ofrecen una documentación extensa de su API versión 2.2, en donde salen ejemplos de utilización de esta API, que nos permite implementar text-to-speech (TTS) y *reconocimiento de voz automatizado* (ASR) en cualquier aplicación con capacidad para Internet.

Esta API se puede usar con o sin el SDK.

Se puede sintetizar el audio hablado a través iSpeech TTS en una variedad de voces, formatos, *bitrates*, frecuencias, y velocidades de reproducción.

Como he dicho se trata de un servicio de pago para acceder a sus servidores para conseguir las traducciones, en el que han puesto un precio por palabra (en el caso de TTS) o por transacción (en el caso de ASR), pero en el caso de que el servicio sea para las plataformas iPhone, Android o BlackBerry es gratuito con un uso razonable mediante iSpeech SDK para generar apps gratuitas.

Desde su Web nos podemos bajar una demostración del uso de la SDK, pero después de la prueba hay que introducir la clave para poder seguir haciendo pruebas y además debe estar con saldo nuestra cuenta para obtener la traducción a audio desde el servidor, en caso de no tener saldo nos devuelve un error de acceso al servidor donde dice que no tenemos crédito.

5.8.5 Java-google-translate-text-to-speech

Existe una API que permitía el acceso al servicio de Google de TTS. En este caso, hay que hablar en pasado, dejó de funcionar a finales del año 2011 para pasar a ser un servicio de Google de pago debido a que detectaron un abuso por parte de ciertas Web's que utilizaban sus servicios como si fueran propios.

Esta API nos permitía traducir un texto de un idioma a otro, detectar el idioma de un texto, obtener el nombre del idioma del texto, obtener el nombre del idioma del texto pero traducido a otro idioma, y por último permitía obtener el audio, en el idioma, del texto solicitado.

Actualmente se puede utilizar la Web de Google para traducir textos, se puede realizar traducciones entre los 80 idiomas de que dispone la Web.

En la página[30] que nos ofrece la posibilidad de traducir aparecen unos altavoces pequeños, uno en el lado del texto a traducir y otro del lado del texto traducido, si pinchamos sobre alguno de estos altavoces oiremos una reproducción del texto escrito en su correspondiente idioma.

Google no cuenta con una API oficial para poder obtener estos audios, pero sin embargo si que los consigue y esto es porque pasa los datos a otra página que realiza este trabajo. Hay gente que lo observó, y llegó a la conclusión que los audios se cargaban a través de una URL y que se le pasaban dos variables una era el idioma y la otra el texto a traducir/reproducir por el método GET. La URL es la siguiente:

http://translate.google.com/translate_tts?tl=en&q=text

El valor de la variable "tl" debe pertenecer al idioma que debe utilizar para la traducción y la variable “q” equivale al texto de la consulta (*query*) que será el que oiremos.

Se puede utilizar cualquiera de los idiomas que reconoce Google (que en estos momentos son 80), pero tendremos una restricción en el número de caracteres que podremos pasar en la variable texto que no deben ser superior a 100 caracteres. Además tendremos otra restricción, debemos simular que la llamada se realiza desde un navegador (*browser*) porque si no lo hacemos Google no nos dejará obtener el audio.

Lo bueno de este sistema es su simplicidad, si queremos traducir más de 100 caracteres sólo hay que hacer un procedimiento que nos divida el texto en *subtextos* con dimensión máxima de cien caracteres y llamar a la URL tantas veces como *subtextos* tengamos e ir añadiendo el sonido recibido a un fichero de audio, al final del proceso obtendremos un fichero de audio con todo el texto traducido.

En el próximo capítulo se explicarán las diferentes herramientas que se utilizarán para el desarrollo de la aplicación, a partir del análisis de requerimientos y del diseño. Pero está claro que se utilizará esta página de Google para la traducción TTS por su simplicidad y que no requiere de instalaciones de librerías.

30 https://translate.google.es/

Capítulo 6: Ingeniería del software

Todo proyecto debe empezar por un proceso de análisis y especificación que nos detalle lo que se tiene que hacer, a partir de unas necesidades de las personas que nos encargan el proyecto, podríamos decir que sería nuestro punto de partida.

Nosotros a partir de estas necesidades debemos ser capaces de obtener toda la información necesaria para realizar un proyecto que cumpla con los objetivos establecidos con los destinatarios del proyecto.

Una vez tengamos todos los requisitos y especificaciones podremos realizar el diseño que deberá representar y modelar el sistema que tendremos que implementar después.

Por último tendríamos la implementación del diseño, que sería la parte más técnica del proyecto, y que dará lugar a la creación del producto que cumpla con las especificaciones. Habrá que realizar las pertinentes pruebas para demostrar que el producto obtenido funciona cumpliendo los requerimientos de las personas que nos han solicitado el proyecto.

En nuestro caso el punto de partida es el enunciado que ha motivado este proyecto, y de él sacaremos toda la información para poder conseguir el producto final.

6.1 Análisis de requerimientos

Si se hubiera tratado de un proyecto tradicional, en este apartado tendríamos de obtener la información de a quién va dirigido el proyecto, los actores que intervienen en el sistema. También deberíamos obtener las necesidades que originan el desarrollo de este proyecto, y cuáles son los resultados esperados, o sea deberíamos obtener del cliente las necesidades funcionales que debemos cubrir (que nos definirán el comportamiento que debe tener el nuevo sistema).

También habría que hacer un análisis de requerimientos no funcionales y el perfil de usuario al que va destinado el producto.

Debido a las limitaciones impuestas por el tipo de proyecto que estamos realizando, tomaremos como necesidades del sistema (producto) las funcionalidades requeridas en el enunciado.

Definiremos primero el perfil del usuario al que se destina el producto, después pasaremos a definir los requerimientos funcionales que definirán el comportamiento de la aplicación, para continuar después con los requerimientos no funcionales que nos impondrán restricciones en el

diseño (como requisitos de funcionamiento o estándares), y por último la arquitectura técnica que se utilizará en el proyecto.

6.1.1 Perfil del usuario

Es importante saber a quien va destinado el software que generará este proyecto, ya que será una parte importante en la implementación, principalmente de la capa de presentación.

El perfil de usuario para nuestra aplicación es la de una persona que necesita leer sus documentos de trabajo de camino a su puesto de trabajo o a la universidad y que no puede realizarlo de manera convencional ya que no se lo permite su medio de transporte. Pero también va destinado a un perfil de estudiante con dificultades para leer (o está estudiando idiomas) y que necesita el apoyo de oír el texto a medida que lo va leyendo.

La decisión que he tomado ha sido de crear un software con una interfaz gráfica sencilla y fácil de usar, que permitirá llegar a un público más amplio. Se omitirán los aspectos técnicos por pantalla.

6.1.2 Requerimientos funcionales

En este apartado debemos ser capaces de extraer del enunciado del proyecto las funcionalidades más importantes que deberá implementar el software que vamos a realizar.

Una vez estudiado el enunciado se obtiene las siguientes funcionalidades que debe proporcionar nuestra aplicación:

- Procedimiento de tratamiento automático de documentos de texto OpenDocument a partir de un fichero de configuración.
- Diseñar el fichero de configuración, en una estructura XML y su correspondiente DTD que lo valide. Este fichero debe indicar que partes del documento debemos tratar.
- Procedimiento que a partir de un texto de entrada nos genere un archivo de audio que contenga el resultado de la voz que nos lee el texto (proceso TTS). A este procedimiento se le tiene que indicar si el archivo es nuevo o debe añadir el resultado al contenido ya existente.
- El fichero de configuración deberá incluir el idioma que se quiere utilizar para la “lectura” del documento.

6.1.3 Requerimientos no funcionales

Podemos considerar como requerimientos no funcionales aquellos aspectos que pueden aportar un valor añadido:

- Gestión de errores que facilite información detallada del problema al usuario en caso de error.
- Notificar al usuario de la falta de algún archivo necesario para el proceso.
- Avisar si hay problemas para conectarse a Internet.
- Facilitar la introducción de los nombres de archivos y directorio para almacenar el resultado, habilitando la posibilidad de realizarlo de forma gráfica mediante un explorador de archivos.
- En el caso de que no exista el directorio que el usuario quiere insertar los archivos, dar la opción de crear el directorio en la ruta indicada.

6.1.4 Arquitectura técnica

En este apartado comentaremos cual es la arquitectura técnica que se ha elegido para la elaboración de esta aplicación y el motivo de estas decisiones.

Lenguaje de programación

Ya sabemos que existen muchos lenguajes de programación, cada uno de ellos con sus ventajas e inconvenientes, además algunos de ellos están muy orientados a entornos específicos. La decisión de cuál de ellos utilizar es compleja y muchas veces depende de la experiencia que el programador tiene del entorno.

Para este desarrollo la decisión del lenguaje a utilizar estaba entre dos, teníamos el C# o el Java, ya que de ambos lenguajes se disponía de librerías para el desarrollo del proyecto. En este caso me he decidido por el entorno Java por los siguientes motivos principales:

- Es el lenguaje utilizado a lo largo de la carrera por lo que dispongo de algo más de experiencia.
- La plataforma de desarrollo Java está muy extendido, y nos ofrece la portabilidad de los desarrollos a diferentes sistemas operativos. Sólo se requerirá disponer de una máquina

virtual Java para ejecutar la aplicación. La compilación del código en Java nos genera un código en lenguaje intermedio, este código generado es el que se ejecuta en la máquina virtual que normalmente utiliza un compilador JIT (Just In Time) para interpretar el conjunto de instrucciones para traducirlas al lenguaje que entiende el sistema en que se ejecuta la aplicación.

- Se puede realizar el tratamiento de ficheros OpenOffice, en cualquiera de sus formatos.
- Como se ha visto en el capítulo de estudio de esta tecnología, existe una API ODF, implementada en Java, que utiliza el paradigma orientado a objetos para representar las diferentes entidades de los documentos ODF como son los elementos: párrafo, tabla, lista, etc.
- También disponemos de clases en Java para el tratamiento de archivos XML, y archivos de audio, necesarios para el desarrollo de nuestra aplicación.

Para el desarrollo en Java se utiliza el entorno IDE de programación de Eclipse que facilita la creación de proyectos. Mediante la instalación de un *plugin* en Eclipse, llamado WindowBuilder, nos facilita el diseño de una aplicación para Windows ya que las ventanas se pueden crear en modo gráfico sin tener que escribir todo el código. WindowBuilder utiliza el paradigma WYSIWYG, “lo que ves es lo que quieres decir” (en inglés: **W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **M**ean).

La versión de Java utilizada es la 7, más concretamente el JDK1.7.0_51

Sistema operativo

El proyecto ha estado desarrollado en entorno Windows, aunque se podría ejecutar en otro sistema no Windows. El único requisito es disponer de una máquina virtual de Java (en inglés Java Virtual Machine, JVM), ya que Java sigue el axioma "escríbelo una vez, ejecútalo en cualquier parte".

Como hemos dicho antes, es la máquina virtual que hace de puente con la plataforma en la que se tiene que ejecutar la aplicación. En otras palabras la máquina virtual conoce las instrucciones de la plataforma de destino, y traduce el código en lenguaje Java (común para todas) al código nativo que es capaz de entender el hardware de la plataforma.

Software necesario

Necesitamos tener instalado la máquina virtual de Java en el ordenador para ejecutar la aplicación. La versión apropiada para nuestro ordenador la podemos descargar de la siguiente

dirección: http://www.java.com/en/download/manual_java7.jsp

Una vez instalado hay que recordarse de actualizar el path y classpath con la ubicación de esta versión, para evitar problemas en la ejecución de la aplicación.

6.2 Diseño

En este apartado se explicará el diseño de las diferentes partes del proyecto.

El diseño de nuestro sistema seguirá una arquitectura de tres capas:

- Una capa de presentación que gestionará la interacción del usuario con la aplicación.
- La capa de la lógica que contiene todas las reglas de negocio.
- Y una capa de persistencia que se basa en el sistema de archivos del sistema operativo.

Se puede ver que esta fase se divide en tres partes bien diferenciadas, por un lado tenemos la creación del fichero de configuración XML con su DTD de validación, después tenemos el diseño del software para la manipulación de documentos ODF para la obtención del texto, y por último tenemos que diseñar el tratamiento del texto para obtener archivos de audio.

6.2.1 Diseño del fichero XML de configuración

Tal como se ha especificado en los requerimientos se debe crear una estructura de fichero de configuración en XML, para el proceso del documento de texto OpenDocument.

El primer paso para realizar el diseño de este fichero ha sido pensar qué información es necesaria a nivel global del proceso, y cuál es particular permitiéndonos delimitar las secciones del documento a tratar.

La información que se ha tenido en cuenta para el fichero XML es la siguiente:

- **Idioma**: se utilizará la nomenclatura que utiliza Google para los diferentes idiomas soportados para la traducción.
- **Imágenes**: hay que indicar si se quiere tratar el texto que hay en las imágenes (toda imagen puede contener un título descriptivo). Hay que tener en cuenta que las imágenes suelen estar contenidas dentro de otros formatos como párrafos o tablas, con lo que para procesarlas habrá que seleccionar el tipo de texto que los engloba ya que en caso

contrario no los procesará.

- **Múltiples archivos**: habrá que decir al proceso si se quiere que el resultado se grabe en un archivo o en varios.
- **Prefijo**: se trata de indicar un prefijo para los archivos que se generen, para que sean fáciles de identificar dentro de la carpeta que se ha elegido para grabar el resultado.
- **Notas**: muchas veces en los documentos se insertan notas al pie de página o al final del documento. Estas notas tienen un identificador dentro del texto que sirve de referencia para identificar la nota al leer el documento, pero en realidad en el XML se encuentra junto al identificador con una etiqueta específica. Habrá que decidir si se quiere convertir estas notas en voz, si se convierten se oirán justo después del identificador.
- **Tipo**: definiremos los tipos de secciones (párrafos, *headings*, listas, tablas, secciones o todo) que queremos convertir.
- **Rango**: deberemos definir el inicio y final de cada sección a convertir del tipo definido antes. Esto nos permitirá saltarnos trozos de texto que no interesa hacer la traducción. Así mismo se permite introducir un prefijo para el rango, en el caso de que se haya definido múltiples archivos.
- **Grupo**: se permite la definición de diferentes categorías de archivos a crear, esto es independiente de si se ha elegido múltiples archivos. Por defecto un grupo es equivalente a un archivo que contendrá el tipo y los rangos seleccionados. En el caso de haber seleccionado múltiples archivos, esto nos permitirá hacer un método para clasificar el resultado obtenido.

Ahora que hemos visto la información que contendrá el fichero de configuración, veamos como queda reflejado esta estructura de información en el formato de un documento XML.

El diseño propuesto es el siguiente:

```
<Audio>
    <Idioma/>
    <Imagen/>
    <Unico/>
    <Prefijo/>
    <Nota/>
    <Traducir>
        <Tgrupo>
            <Tipo/>
            <Rango>
                <Inicio/>
                <Final/>
                <Prefijo-rango/>
            </Rango>
        </Tgrupo>
    </Traducir>
</Audio>
```

Fig. 18: Fichero configuración XML

El elemento raíz es *Audio* y será el que contendrá toda la configuración a aplicar al documento de texto a tratar. Dentro de este elemento tenemos, en primer lugar, los elementos que nos sirven para la configuración general (*Idioma, Imagen, Unico, Prefijo y Nota*), seguido del elemento *Traducir* que contendrá la definición de lo que queremos traducir a voz.

La siguiente tabla nos muestra de forma detallada que es cada elemento:

Elemento	Descripción
<Audio>	Definición de la configuración a utilizar.
<Idioma>	Identificación del idioma que hay que utilizar en el proceso TTS. La codificación es igual a la utilizada por Google.
<Imagen>	Define si hay que tener en cuenta el texto que se encuentra dentro de un <draw:frame >. Los valores que podrá contener son “Si” o “No”
<Unico>	Indica si se tiene que dividir el resultado del proceso TTS en diferentes archivos. Los valores que podrá contener son “Si” o “No”
<Prefijo>	Deberá contener la raíz del nombre que le queremos dar al archivo de audio generado. El proceso le asignará un sufijo, según el procesado realizado.
<Nota>	Nos dirá si se deben convertir a voz los textos de las notas que nos encontramos durante el procesado. Los valores que podrá contener son “Si” o “No”
<Traducir>	Definición de las partes del documento de texto que hay que convertir a voz. En este elemento se definirán los diferentes grupos de conversión.
<Tgrupo>	Definición de un grupo de texto a traducir. Por defecto cada grupo corresponderá a un archivo de audio, en el caso de que no se haya seleccionado múltiples archivos.
<Tipo>	Dice el tipo de texto que se debe seleccionar del documento. Los valores que podrá contener son “pa”=párrafos, “he”=*headings*, “li”=listas, “ta”=tablas, “tc”=indices, “se”=secciones y “to”=todo.
<Rango>	Rango de texto que se tiene que seleccionar para convertir.
<Inicio>	Posición del elemento inicial que empezaremos a realizar el proceso de conversión. La posición cero es el correspondiente a la raíz del documento.
<Final>	Posición del último elemento a tener en cuenta en el proceso.
<Prefijo-rango>	Prefijo que le queremos dar al rango seleccionado para grabar el audio. Sólo se tiene en cuenta cuando se ha seleccionado múltiples archivos, en caso contrario no se deberá informar ya que no se tendrá en cuenta.

Fig. 19: Elementos fichero configuración XML

Una vez tenemos la estructura del fichero XML nos queda crear una DTD que deberá validar los ficheros de configuración XML.

Recordemos que hay dos maneras de indicar una DTD en un fichero XML, como se explicó en el correspondiente capítulo 2, una interna y otra externa. En nuestro diseño hemos decidido de hacer la DTD externa, de esta manera el usuario podrá crear tantos ficheros de configuración como quiera y todos tendrán que hacer referencia a la misma DTD, no se verá obligado a insertar la

DTD en todos los ficheros creados.

Para hacerlo más simple para el usuario, no hará falta que éste añada la línea al principio del fichero XML indicando cual es la DTD que lo validará. Para esto la clase que validará el fichero XML ya se encargará de añadir esta línea en caso de que no esté dada de alta en el documento en el momento de cargarlo para validar. En el caso de que el usuario quiera insertar la línea será como se muestra a continuación:

```
<!DOCTYPE Audio SYSTEM "Audio.dtd">
```

Ahora sólo queda definir como será el diseño de esta DTD que validará los documentos XML de configuración:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT Audio (Idioma, Imagen, Unico, Prefijo, Nota, Traducir)>
<!ELEMENT Idioma (#PCDATA)>
<!ELEMENT Imagen (#PCDATA)>
<!ELEMENT Unico (#PCDATA)>
<!ELEMENT Prefijo (#PCDATA)>
<!ELEMENT Nota (#PCDATA)>
<!ELEMENT Traducir (Tgrupo+)>
<!ELEMENT Tgrupo (Tipo, Rango+)>
<!ELEMENT Tipo (#PCDATA)>
<!ELEMENT Rango (Inicio, Final, Prefijo-rango?)>
<!ELEMENT Inicio (#PCDATA)>
<!ELEMENT Final (#PCDATA)>
<!ELEMENT Prefijo-rango (#PCDATA)>
```

Fig. 20: DTD del fichero de configuración

La forma de interpretar esta DTD es la siguiente:

- El elemento *Audio* deberá contener todos los datos de la configuración una única vez, y en el mismo orden, y estos son *Idioma, Imagen, Unico, Prefijo, Nota, Traducir*.
- Los elementos *Idioma, Imagen, Unico, Prefijo, Nota* son de tipo texto.
- El elemento *Traducir* deberá contener de *1* a *n* elementos *Tgrupo*.
- El elemento *Tgrupo* contiene un elemento *Tipo* y *n* elementos *Rango*.
- El contenido del elemento *Tipo* es de texto.
- El elemento *Rango* contendrá el elemento *Inicio*, *Final*. Además puede contener el elemento *Prefijo-rango* o no.
- Los elementos *Inicio*, *Final* y *Prefijo-rango* son de tipo texto. Además por programa se

permitirá que el elemento *Final* esté vacío.

6.2.2 Diseño de la interfaz del usuario

Como se ha dicho en los requerimientos, se deberá diseñar una interfaz sencilla de utilizar para cualquier tipo de usuario.

En la capa de presentación tendremos un único formulario, en el que se solicitarán los datos necesarios para la ejecución de la aplicación. Este formulario diferenciará lo que son los datos de entrada necesarios para el proceso y la información requerida para guardar el resultado del proceso, además se le añade un botón que servirá para iniciar la ejecución del proceso.

Se utilizarán ventanas emergentes para los errores que advertirán al usuario de cual ha sido el problema detectado. Se utilizan también *etiquetas de textos emergentes*[31] en cada campo.

Se facilitará en cada campo, donde el usuario tiene que introducir la dirección de una carpeta o archivo, un botón para acceder a una ventana de gestión de archivos igual al que ofrece el sistema operativo para que pueda seleccionar de una manera fácil la ubicación y nombres correspondientes.

Por último indicar que no se solicita al usuario el nombre y ubicación de la DTD que debe validar el XML porque ésta ya viene definida por defecto con la aplicación y se encontrará en el mismo directorio que el ejecutable.

6.2.3 Diseño del algoritmo de tratamiento

El diseño del tratamiento es esencial para el diseño de un software. A continuación presentamos el algoritmo general de la aplicación que tenemos que crear, en él vemos el flujo general de la aplicación con todos los procesos y validaciones relacionadas. Para hacerlo más entendedor se han definido unos subprocesos que se definirán en los siguientes dibujos, estos son el procesado del documento ODF y el proceso de conversión del texto a voz.

31 En inglés *tooltip*.

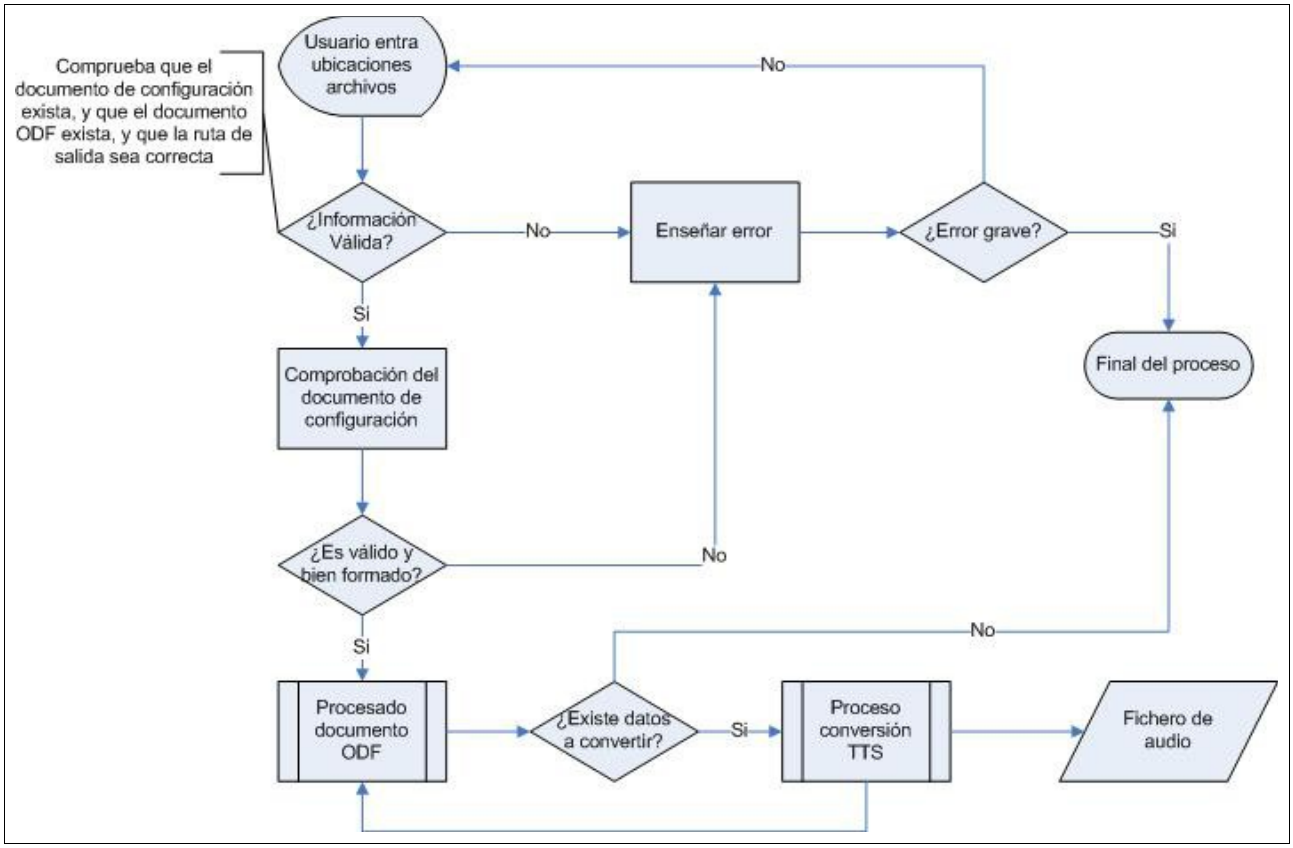


Fig. 21: Flujo principal del algoritmo de la aplicación

A nivel interno el algoritmo recibe la información de lo que tendrá que procesar por pantalla, por medio de la interfaz del usuario, a partir de esta información validará que los datos son correctos. Si todo es correcto, abrirá el documento de configuración y lo validará contra la DTD correspondiente. Llegados a este punto si el documento es válido y tiene la configuración general correcta (valores de los diferentes elementos son permitidos), se empezará el procesado del documento ODF y para cada elemento encontrado a convertir lo pasará al proceso de conversión TTS que nos generará como resultado un fichero de audio con la conversión.

A continuación presentamos el flujo del procesado del documento ODF:

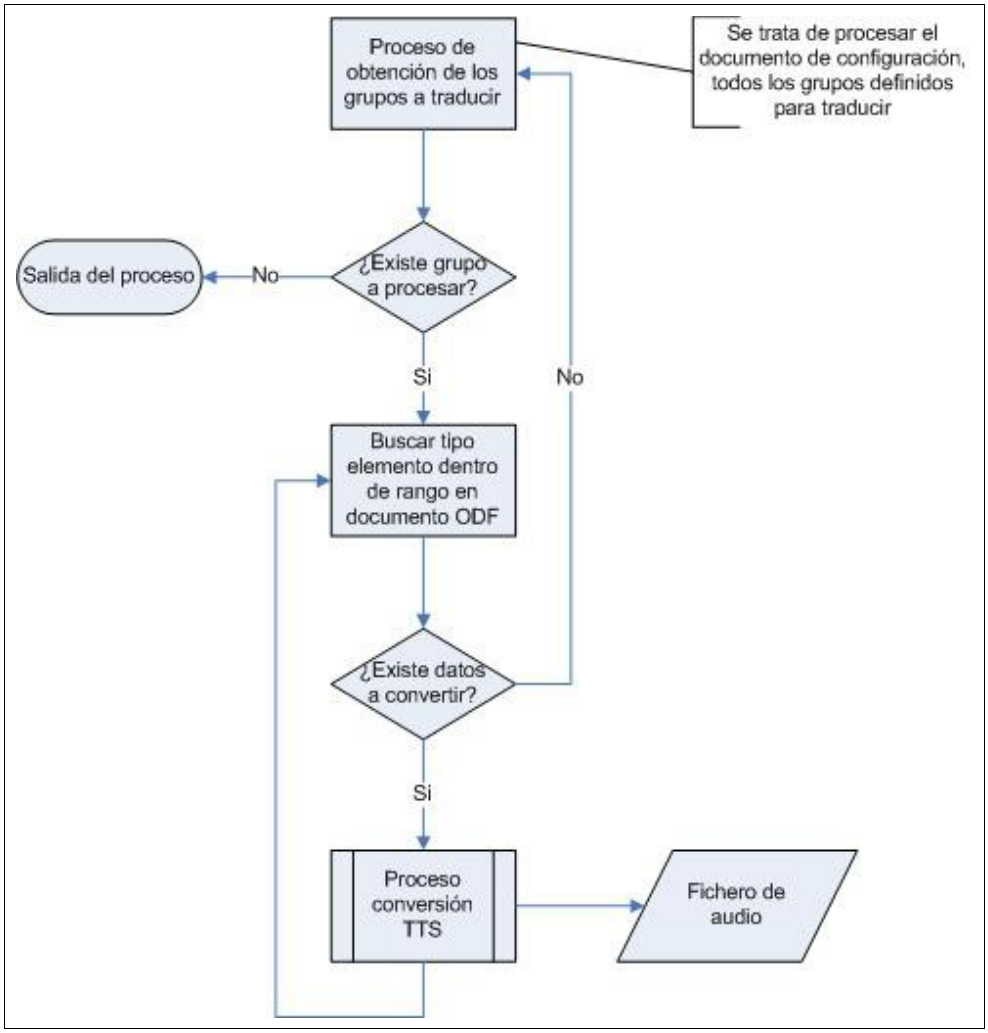


Fig. 22: Flujo procesado documento ODF

Este proceso tratará la parte variable del documento de configuración, por cada elemento *Tgrupo* encontrado deberá procesar el documento ODF en busca de las partes de texto que cumplen con la selección que estamos tratando para realizar su posterior proceso de conversión a voz y almacenaje en el archivo de audio.

A continuación presentamos el flujo del proceso de conversión TTS:

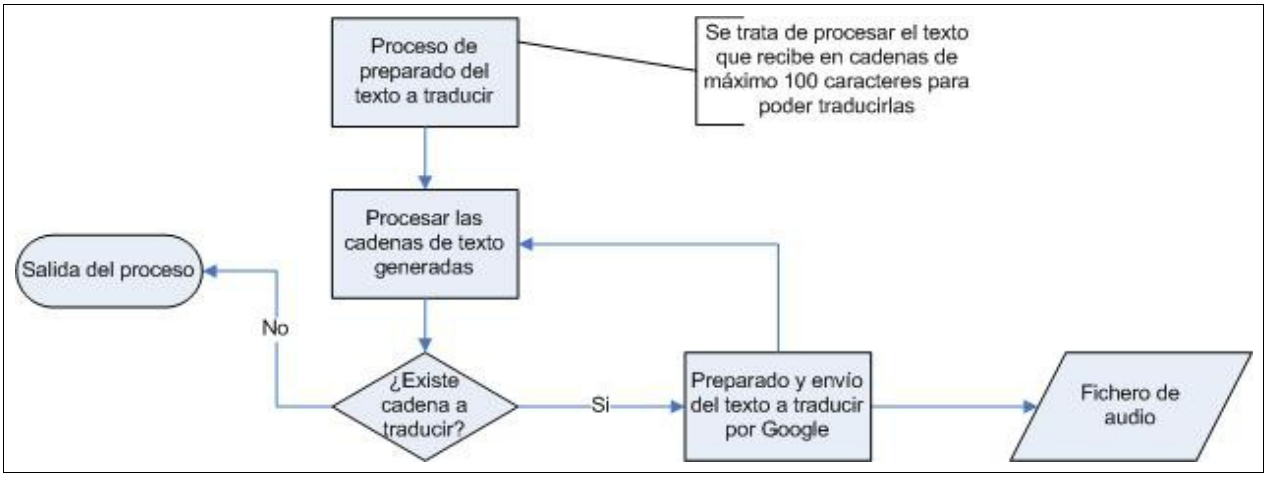


Fig. 23: Flujo procesado TTS

6.2.4 Diseño de clases

Presentamos el diseño de las clase que forman parte de la lógica de la aplicación y como se relacionan entre ellas.

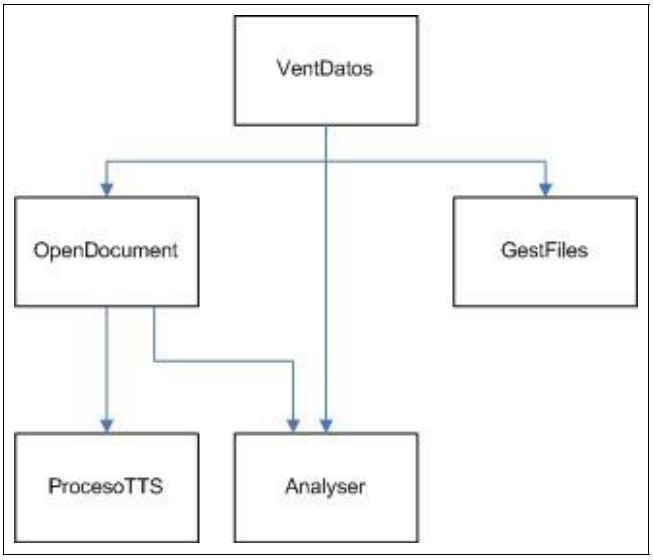


Fig. 24: Relación entre las Clases

- **VentDatos**: Se trata de la clase que arranca la aplicación y gestiona la interfaz principal con el usuario, y se encarga de hacer las llamadas a los métodos necesarios de las clases OpenDocument, GestFiles y Analyser para realizar todo el proceso.

- **OpenDocument**: Esta clase se encargará de realizar todo el tratamiento del documento ODF a partir de los datos de configuración que obtiene de la clase Analyser. Al procesar el documento ODF va obteniendo cadenas de caracteres a traducir a voz, que las enviará a la clase ProcesoTTS.

- **GestFiles**: Clase que se encargará de inicializar las rutas de los ficheros y comprobar la existencia del directorio donde se generarán los archivos, que en caso de que no exista lo creará.
- **Analyser**: Es la clase que se encargará principalmente de cargar y validar el documento de configuración XML en el sistema mediante una definición DTD. Se encarga también de comprobar que los valores de la parte general de la configuración sean correctos, para guardarlos para ser accedidos desde la clase OpenDocument. También carga una tabla que contendrá todos los idiomas, y su descripción, que podrá utilizar nuestro sistema.
- **ProcesoTTS**: Es la clase encargada de convertir el texto que le llega en voz y grabarlo en el archivo de audio definido, utilizando el idioma definido al inicio de la aplicación.

6.3 Implementación

Como ya se ha comentado en apartados anteriores la aplicación ha sido desarrollada en Java, bajo el entorno de desarrollo IDE Eclipse. Hay que tener en cuenta que los archivos de configuración XML no los genera la aplicación, por lo que los tiene que crear el usuario antes de la ejecución de la aplicación utilizando la estructura definida en el apartado de diseño.

En este apartado describiremos los puntos principales del desarrollo de la aplicación, y las soluciones emprendidas para obtener una solución ejecutable que cumpliese con los requisitos.

6.3.1 Capa de presentación

La aplicación dispone de un único formulario para introducir los datos que se encuentra en la clase VentDatos.

Para el diseño de este formulario he utilizado un *Plugin* de Eclipse llamado WindowBuilder, el cuál permite crear un formulario arrastrando componentes visuales sobre éste generando parte del código, con lo que sólo nos tenemos que preocupar del código de la lógica de nuestra aplicación.

Nuestro formulario recoge los datos del fichero de configuración y del fichero de texto, como entrada, para realizar el proceso y un directorio donde se grabarán los archivos que la aplicación pueda generar. A continuación presentamos como ha quedado el diseño de este formulario:

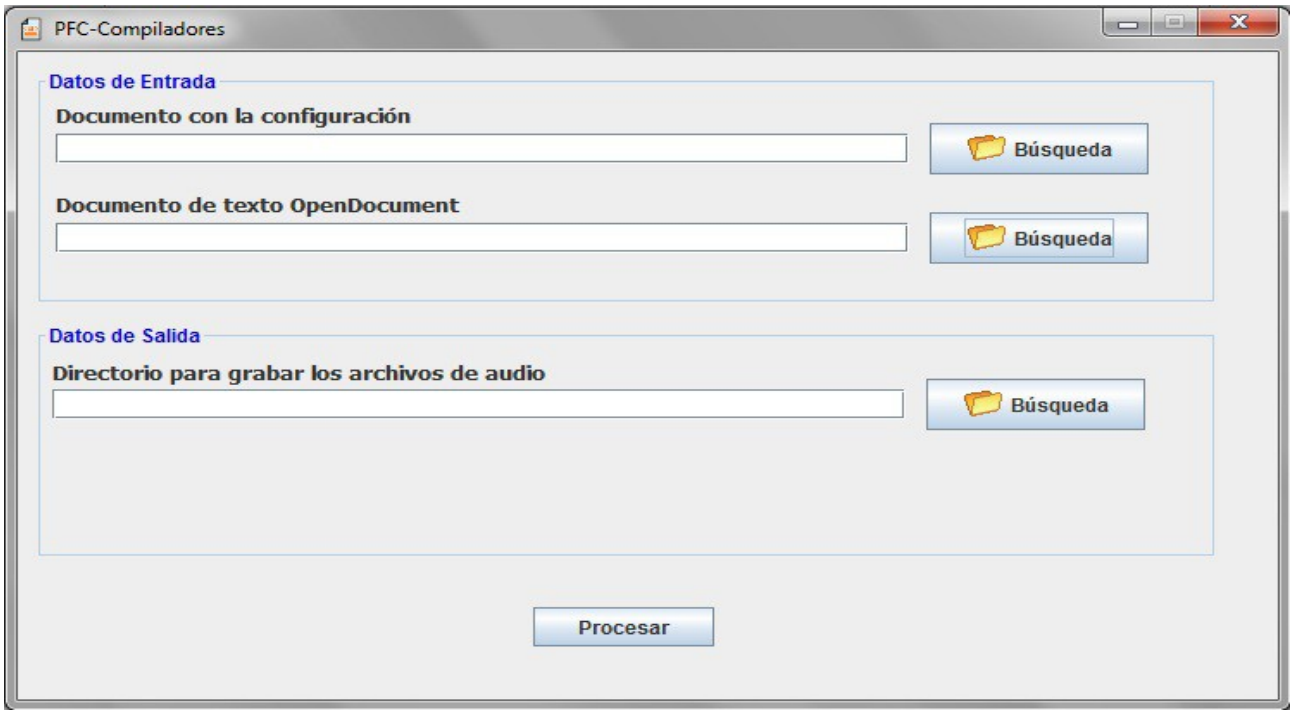


Fig. 25: Formulario de solicitud de datos

Los botones que pone "Búsqueda" permiten acceder a un diálogo genérico de gestor de archivos que se puede adaptar a cada caso. Esta parte que se adapta se trata del título y el tipo de archivos que debe seleccionar.

A continuación se muestra un ejemplo de este gestor de archivos que se parece mucho al ofrecido por el sistema operativo.

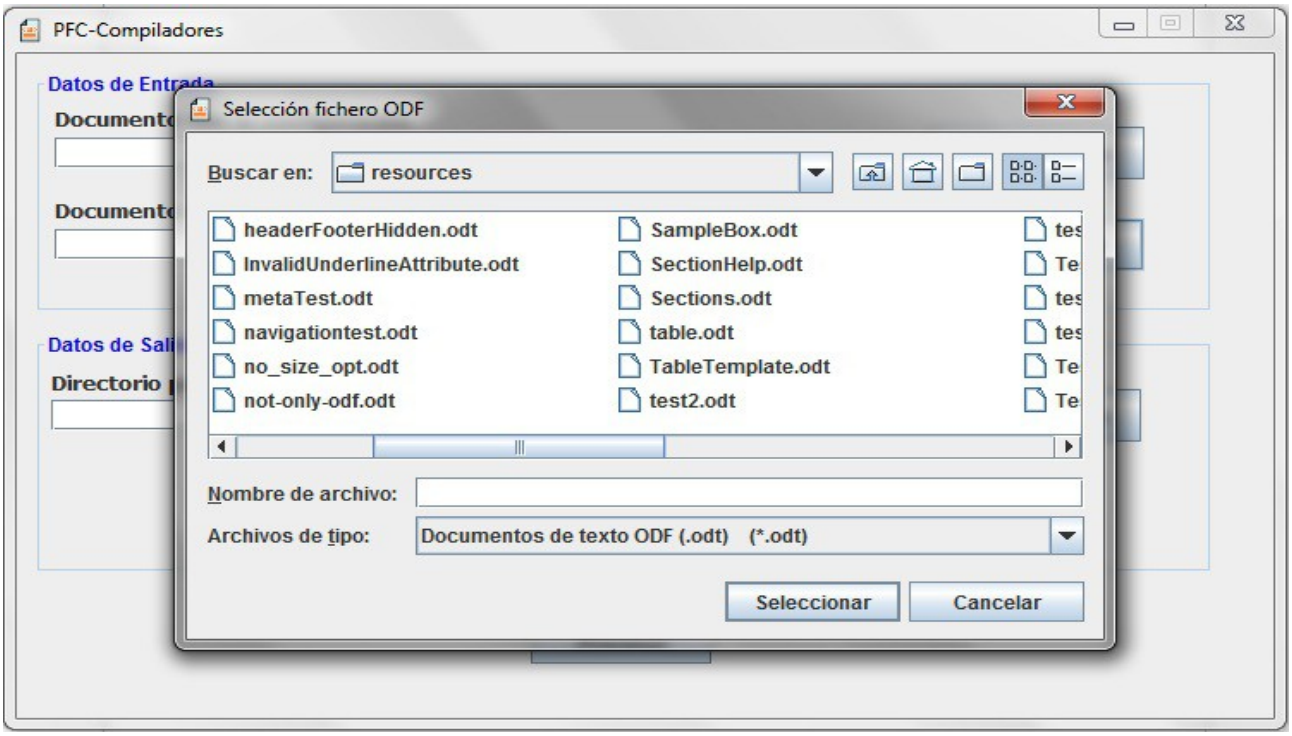


Fig. 26: Ventana para seleccionar archivos

Dado que se trata de un proyecto realizado en Java bajo el entorno de desarrollo de Eclipse, se ha generado un fichero con extensión *jar,* que es ejecutable, mediante la opción de exportar que tiene Eclipse. Con esto se consigue que llamando por la línea de comandos a VentDatos.jar o haciendo doble clic sobre su icono (en entorno gráfico) se ejecute nuestra aplicación desde el formulario inicial.

6.3.1.1 Gestión de eventos

Una vez se visualiza el formulario de la aplicación, sólo nos queda controlar las acciones que puede realizar el usuario, y no dejar que realice el proceso sin tener informado todos los datos de entrada y estos sean correctos.

Se utiliza la gestión de eventos para controlar las acciones que realiza el usuario. En nuestro caso los relacionados con los botones, que los hay para seleccionar los ficheros y directorio, y otro para iniciar el proceso.

También es importante que el usuario reciba información del resultado de sus acciones, de esta manera sabrá si la acción realizada se ha hecho correctamente o no. Para ello nuestra aplicación podrá enseñar diferentes tipos de mensajes:

- Mensajes de datos incompletos.
    - Cuando falte informar el documento de configuración XML.
    - Cuando falte informar el documento ODF.
    - Cuando falte informar el directorio de salida.
- Mensajes de datos incorrectos.
    - Cuando el documento de configuración no esté bien formado o no se haya podido validar con la DTD. En este caso se sale de la aplicación después de informar del error.
    - Si algún dato de la configuración general es errónea.
    - Cuando el documento ODF no sea correcto.
- Mensajes de finalización.
    - Finalización del proceso de manera correcta.

- Finalización del proceso provocado por un error grave.

6.3.2 Capa de negocio

En esta capa se ha realizado la implementación de las clases descritas en el proceso de diseño.

Este proceso se inicia en el momento en que el usuario pulsa el botón de “Procesar”. En esos momentos se genera un evento de procesar la información contenida en el formulario, y si los datos están todos y son correctos, que seguirá el siguiente proceso hasta su finalización:

- Inicialización de las rutas y se crea el directorio donde se guardarán los archivos de audio generados (en la clase GestFiles).
- Se crea una instancia de la clase Analyser. En este momento se carga una tabla con los idiomas que se puede utilizar para su posterior utilización en las comprobaciones de los valores de la configuración. Se carga también el documento XML para su validación con la DTD.
- Se obtienen los datos de la cabecera del documento de configuración y se validan, procedimiento realizado mediante la llamada a un método de la clase Analyser.
- Se crea una instancia de la clase OpenDocument pasandole el nombre del documento ODF a tratar. En este momento se carga el documento a procesar y comprueba que sea un documento ODF.
- Finalmente se llama al proceso de tratamiento del documento ODF, al que se le pasa una referencia de la clase Analyser y la dirección de destino de los archivos.
- Por último se cierra el documento ODF y se avisa al usuario de la finalización correcta de la aplicación.

En los siguientes apartados explicaremos los puntos más importantes de la implementación de cada una de las clases.

6.3.2.1 Clase Analyser

Esta clase es importante dentro de la aplicación porque, como hemos dicho, carga el archivo XML y lo valida contra la DTD. Para este proceso he utilizado el método *analyzeXML* de una práctica de compiladores 2, adaptándola un poco a los nuevos requerimientos.

Hay que destacar que la DTD debe encontrarse en el mismo directorio donde se encuentra el ejecutable y desde donde realizamos la ejecución, caso no exista la validación dará error. En el archivo ZIP de instalación viene entre otros este archivo que se llama “Audio.dtd”.

Esta clase se guarda una tabla hash con la relación de las claves de los idiomas que usa Google y su descripción, que se visualizará en un mensaje de confirmación del idioma escogido en el archivo de configuración.

Por último la obtención de los datos de la cabecera de la configuración se obtienen directamente utilizando accesos XPath efectuado sobre el DOM. A continuación se muestra un ejemplo:

```
XPathFactory aXPF = XPathFactory.newInstance();
XPath aXPath = aXPF.newXPath();
// Se omite código no significativo
(….)
XPathExpression aExpr = aXPath.compile("/Audio/Idioma");
Node aNodo = (Node) aExpr.evaluate(documentoValidado,XPathConstants.NODE);
String valor = aNodo.getTextContent();
      if (!this.tablaIdioma.containsKey(valor)) {
       resultado = 1;
       return resultado;       }
      setIdioma(valor);
(...)
```

Fig. 27: Ejemplo acceso XML con XPath

6.3.2.2 Clase GestFiles

Su objetivo es crear las rutas y guardarse los nombres de los ficheros y ubicaciones. En el caso de que no exista el directorio de salida para guardar los audios, lo crea de la siguiente manera:

```
public void generarDirs() {
    File directorioSalida = new File(outputPath.toString());
    if (!directorioSalida.exists()) {
        directorioSalida.mkdirs();
    }
}
```

Fig. 28: Ejemplo de generar directorios

6.3.2.3 Clase OpenDocument

Esta clase se encarga de interactuar con la librería ODF para tratar el documento a partir de los datos obtenidos del fichero de configuración, y a medida que obtiene el texto lo pasa al proceso que genera el audio.

Para tratar el documento ODF primero hay que cargar el documento y esto se hace con:

```
TextDocument outputDocument;
//  pathDocTrab es la ruta física del documento
outputDocument =  TextDocument.loadDocument(pathDocTrab);
```

Fig. 29: Carga y abertura documento ODF

Seguidamente se obtiene el contenido de la etiqueta "office:text" del archivo *content.xml* que contiene el texto del documento a tratar, y se hace mediante la siguiente instrucción:

```
contentroot = outputDocument.getContentRoot();
```

Para realizar el recorrido por el texto del documento se utiliza bastante la API DOM, pero para evitar tener que obtener cada vez los hijos de un nodo o tratar nodos que sólo tienen hijos tipo texto se utilizan funciones ofrecidas por las librerías de ODF. Los métodos más importantes utilizados son:

- Obtener el primer elemento hijo del TextDocument (*getFirstElementChild()*)
- El saber cuantos elementos nodo son hijos del elemento actual (*getChildElementCount*), esto nos permite saber si el nodo actual tiene hijos que deben ser tratados como elementos, y no obtener directamente el texto.

```
int numElem;
...
numElem = elementoTrt.getChildElementCount();
…
// Si numElem es cero quiere decir que no tiene elementos hijo, pero si contenido de texto
if (numElem == 0) {
        textoTTS = elementoTrt.getTextContent();
        procesoTTS.textToSpeech(textoTTS, analiser.getIdioma());
        return;
}
```

- Para saber si un elemento ODF está vacío utilizamos el método (*getLength*) que nos da el número de hijos del nodo que se consulta.
- Se utilizan los nombres de las clases ofrecidas por ODF para comparar si un nodo es de ese tipo, para no tener que comparar siempre el nombre del nodo con la correspondiente etiqueta del elemento. Como por ejemplo:

```
if (node instanceof TextHElement || node instanceof TextPElement) {
        // Se evita tratar elementos vacios
        if (((OdfElement)node).getLength() > 0) {
                sb.append(' ');
                sb.append(obtenerTexto((OdfElement)node));
        }
}
```

- Los tipos de elementos hijos tratados son: *TextPElement* (párrafos), *TextHElement*

(cabeceras, *headings*), *TextListElement* (listas), *TableTableElement* (tablas), *TextTableOfContentElement* (tabla de contenidos) y *TextSectionElement* (secciones).

Para controlar que el rango inicial y final del documento de configuración fueran dígitos se ha utilizado un patrón, si el rango inicial no es numérico se le asigna el valor 1 y en el caso del final si no numérico se le asigna 0. Si el rango final es 0 el proceso tratará todo el documento desde el valor inicial hasta su conclusión, en caso contrario sólo hasta el valor indicado. Ejemplo:

```
if (aChildren2.item(j).getNodeName() == "Final") {
        auxStr = aChildren2.item(j).getTextContent();
        // Hay que comprobar que el rango inicial es un numérico entero
        if (auxStr.trim().matches ("\\d+$")) {
            rangoFin = Integer.parseInt(auxStr.trim());
        }
        else {
           // Si no es numérico se considera que el final es el final del documento
           // con lo que asignamos el valor 0
           rangoFin = 0;
         }
}
```

Este rango se utiliza para seleccionar el número de elementos iguales al tipo seleccionado que se encuentran a partir del elemento raíz del documento, para su conversión a voz. Esta clase también se encarga de ir generando el nombre del archivo de audio según los datos de la configuración, posteriormente informará a la clase ProcesoTTS de este nombre para que pueda generar el correspondiente archivo.

6.3.2.4 Clase ProcesoTTS

Es la clase encargada de crear los archivos audio a partir del texto que recibe.

Hay que decir que esta clase utiliza una facilidad que ofrece Google para traducir textos, al utilizarlo de forma gratuita sólo nos permite realizar traducciones de hasta 100 caracteres cada vez. Uno de los trabajos que deberá realizar esta clase es dividir el texto recibido en cadenas de máximo 100 caracteres.

Una vez recibe el texto lo codifica a formato HTML usando UTF-8, ya que en la URL de Google hay que pasar el texto codificado, mediante la instrucción:

```
text = URLEncoder.encode(text, "utf-8");
```

Para dividir el texto en cadenas de menos de 100 caracteres se utiliza el carácter “+” para poder cortar el texto, ya que se corresponde a espacios en blanco. El signo “+” al principio de una cadena se elimina.

Se utiliza un fichero temporal para recibir el audio desde la URL de Google y después se copia a un FileOutputStream que irá guardando todo el texto traducido. En el código siguiente se ve como se hace

```
(….)
File tempFile = File.createTempFile("FichAudio", ".mp3");
tempFile.deleteOnExit();

for (String snippet : snippets) {
  //Cadena con la URL que utilizaremps con el texto (snippet) y el idioma (language)
 String urlString = new String ("http://www.translate.google.com/translate_tts?ie=UTF-8&q=" + snippet
+ "&tl=" + language+"&prev=input");
  BinaryResource res = new Resty().bytes(new URI(urlString));
  res.save(tempFile);
  // Copia fichero temporal al fichero que contendrá todo el audio
  Files.copy(Paths.get(tempFile.getCanonicalPath()), os);
  //se asigna os a un FileOutputStream que añade audio al fichero de destino
  //sólo se tiene que hacer una vez ya que después se utiliza la misma referencia
  if (shouldAppend) {
      os.close(); //cierra la referencia al output streams
      os = new FileOutputStream(new File(destination + ".mp3"),true);
      shouldAppend = false;
   }
}
//cierra la referencia al output streams
os.close();
(….)
```

El audio que se recibe de Google viene en formato MP3, y no es estéreo, con lo que tendremos que convertirlo en formato AIFF y mantener que sea mono para que se pueda oír bien. Esto se realiza una vez completado todo el archivo de audio, mediante el método de esta clase llamado *generaAif()*, este método utiliza funciones de Java Sound API que permite crear un archivo de audio con un formato predefinido a partir de una fuente de audio de entrada. A continuación vemos el código principal de este método

```
File file = new File(destination + ".mp3");
// Si el fichero no existe no hay que hacer la transformación a AIFF
if (!file.exists())
    return;
AudioFileFormat aff;

try {
    aff = AudioSystem.getAudioFileFormat(file);
    AudioFormat baseFormat = aff.getFormat();
    AudioInputStream ini = AudioSystem.getAudioInputStream(file);
    AudioFormat decodedFormat =  new AudioFormat(AudioFormat.Encoding.PCM_SIGNED, baseFormat.getSampleRate(), 16, 1, 2,
            baseFormat.getSampleRate(),
            true);
    AudioInputStream din = null;
    din = AudioSystem.getAudioInputStream(decodedFormat, ini);
    AudioSystem.write(din, AudioFileFormat.Type.AIFF, new File(nomArch + ".aif"));
    ini.close();
    din.close();
```

Fig. 30: Conversión de formato MP3 a AIFF

Lo que hace este proceso, básicamente, es crear un formato de codificación de audio que lo guarda en la variable *decodedFormat*, después crea un *AudioInputStream* que tendrá el audio que se va convirtiendo y es el que se utiliza para escribir en el fichero con formato *AIFF*.

6.3.3 Librerías utilizadas

Para realizar este proyecto además de las librerías estándares de la versión 7 de Java he tenido que utilizar otras, algunas de ellas se necesitan utilizar porque las utiliza otra librería de la que utilizo algún método.

Empezamos por las librerías relacionadas con el tratamiento de documentos ODF, en este caso he necesitado dos librerías básicas *odfdom-java-0.8.7.jar* y *simple-odf-0.8.1-incubating.jar* ambas librerías se han descargado de la página web de 'Apache ODF Toolkit (incubating)'[32].

Algunas clases de las que he utilizado de estas librerías usan referencias o importaciones de otras dos librerías que se han tenido que adjuntar ya que si no provocaban errores, también se han descargado de la misma página de Internet que las anteriores. Estas librerías son: *xercesImpl.jar* y *xml-apis.jar*. Son utilizadas para trabajar con elementos XML y métodos especiales para tratar nodos (como por ejemplo el método *getFirstElementChild* o *getChildElementCount*).

Para la comunicación con la URL de Google se ha utilizado la librería Resty[33] (*resty-0.3.2.jar*), nos realiza la conexión con la URL y nos establece el canal para recibir los datos, también dispone de un método para guardar los datos descargados en un fichero.

También se ha utilizado la librería *commons-io-2.4.jar*[34] que proporciona utilidades para facilitar el uso de los ficheros, en particular he utilizado la clase FileUtils.

Finalmente para el tratamiento del audio también se ha necesitado de librerías extras. Cabe recordar que se dijo que se utilizaría la librería estándar ofrecida por Java Sound API, pero al usar Google para obtener la conversión a audio, éste se obtiene en formato mp3. La API por si sola no reconoce el formato mp3, por lo que no se puede realizar la conversión a otro formato.

La solución a este problema ha sido la instalación de un Plugin en forma de librerías, que dan la información necesaria a la API para trabajar con otros formatos, como mp3. Es totalmente transparente, ya que no se utiliza ningún método de estas librerías dentro del programa, pero la

32 La dirección es https://incubator.apache.org/odftoolkit/index.html
33 Descarga realizada desde http://beders.github.io/Resty/Resty/Download.html
34 Se puede descargar de la página de Apache Commons IO http://commons.apache.org/proper/commons-io/download_io.cgi

API va a buscar información complementaria a la que dispone. La librería utilizada es 'MP3 SPI for Java™Sound'[35] (*mp3spi1.9.5.jar*), al descargar esta librería nos añade dos más que son las que se basa su implementación (*jl1.0.1.jar* y *tritonus_share.jar*).

6.4 Ejemplos y pruebas de funcionamiento

Una vez tenemos diseñado e implementado todas las partes de este proyecto, toca realizar una serie de pruebas para verificar su correcto funcionamiento. En este capítulo expondremos alguno de estos test.

Hay que destacar que el proceso puede tardar tiempo en completarse debido al tiempo que se tarda en traducir el texto a voz y generar el archivo de audio, este tiempo depende en gran medida del volumen de texto de cada prueba.

También es importante recordar que esta aplicación necesita acceso a Internet para que funcione.

Me he encontrado con problemas para verificar la corrección de la traducción sobre todo en la parte de los índices y tablas, principalmente, en el caso de los índices, por la manera de pronunciar y no hacer pausas en los saltos de línea y los títulos los junta con los números (el número de página lo junta con la numeración del capítulo). En el caso de las tablas el problema está en distinguir cuando cambia de celda y conseguir que haga una pausa al cambiar de celda.

Estos problemas no los he conseguido solventar ya que depende de la traducción de Google, y a pesar de pasarle saltos de página y espacios en blanco, la traducción no es del todo correcta para el oído humano.

Inicialmente tuve problemas para obtener el texto de ciertos elementos ya que utilizaba el método de obtener el contenido de texto de un elemento nodo, pero en el caso de que este elemento tuviese hijos, añadía el texto de los hijos del elemento que se estaba tratando. Esto sería correcto en el caso de querer traducir todo el contenido, en los demás casos no. Para solucionarlo se ha tenido que hacer funciones recursivas que fueran buscando el texto en los elementos hijos y cuando se llega a un elemento terminal se devuelve su texto si lo tiene, de esta forma nos podemos saltar las partes que no interesan o no están contempladas para traducir.

35 MP3SPI es un Service Provider Interface de Java que añade soporte al formato de audio MP3 para las plataformas Java, está basado en librerías Jlayer y Tritonus. Se puede descargar de la página http://www.javazoom.net/mp3spi/mp3spi.html

6.4.1 Caso de prueba 1

Esta prueba la he dividido en dos, para comprobar el funcionamiento del detalle de la configuración y como actúa sobre los resultados obtenidos.

Los ficheros de configuración utilizados son *pruebaConf1b.xml* y *pruebaConf1c.xml*, y los dos se ejecutan sobre el mismo archivo ODF que se llama *emarquet_PAC2A.odt*.

Ambos utilizan el idioma castellano y quieren que el resultado salga en varios archivos de audio, y sí quieren que se traduzcan las notas de pie de página del texto seleccionado para traducir. Aunque prácticamente ambos ficheros de configuración son iguales, los resultados obtenidos son diferentes. Veamos el contenido de ambos ficheros XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<Audio>
    <Idioma>es</Idioma>
    <Imagen>SI</Imagen>
    <Unico>NO</Unico>
    <Prefijo>Prueba6</Prefijo>
    <Nota>SI</Nota>
  <Traducir>
    <Tgrupo>
      <Tipo>pa</Tipo>
      <Rango>
        <Inicio>1</Inicio>
        <Final>20</Final>
        <Prefijo-rango>Parrafo</Prefijo-rango>
      </Rango>
      <Rango>
        <Inicio>37</Inicio>
        <Final>45</Final>
        <Prefijo-rango>Parrafo</Prefijo-rango>
      </Rango>
    </Tgrupo>
    <Tgrupo>
      <Tipo>li</Tipo>
      <Rango>
        <Inicio>1</Inicio>
        <Final></Final>
        <Prefijo-rango>Lista</Prefijo-rango>
      </Rango>
    </Tgrupo>
    <Tgrupo>
      <Tipo>tc</Tipo>
      <Rango>
        <Inicio>1</Inicio>
        <Final>1</Final>
        <Prefijo-rango>Indice</Prefijo-rango>
      </Rango>
    </Tgrupo>
  </Traducir>
</Audio>
```

Fig. 31: Ejemplo pruebaConf1b.xml

El primer caso nos generará tres archivos de audio, uno por cada *Tgrupo*, aunque se ha configurado varios archivos. Esto es así porque se ha definido en el primer *Tgrupo* dos grupos con el mismo *Prefijo-rango* por lo que el segundo sustituye al primero, y su resultado se perderá. Los nombres de los archivos creados son: *Prueba6G1Parrafo.aif*, *Prueba6G2Lista.aif* y *Prueba6G3Indice.aif*.

Para comprobar que los archivos de audio contienen la voz para todo el texto seleccionado,

primero he abierto el documento de texto y he puesto en marcha el audio. En el caso del audio que contiene las listas y el del índice no es complicado seguirlos, sólo hay que ir leyendo la parte de texto correspondiente para confirmar la corrección. Pero en este caso de párrafo es más complicado porque los párrafos vacíos también cuentan y visualmente desde el editor es difícil de seguir. Para este caso resulta mejor abrir el correspondiente archivo content.xml y se tienen que contar las etiquetas *<text:p>* que son hijos de la etiqueta <office:text> ya que son las que se cuentan para hacer la traducción, las que podamos encontrar en un nivel inferior no cuentan (aunque pueden ser tenidas en cuenta para la traducción, según el caso). De esta manera también se puede ver mejor las notas de pie de página ya que se encuentran justo después de donde está el número de nota y no al final de la página como se ve en el editor.

```
<?xml version="1.0" encoding="UTF-8"?>
<Audio>
    <Idioma>es</Idioma>
    <Imagen>SI</Imagen>
    <Unico>NO</Unico>
    <Prefijo>Prueba7</Prefijo>
    <Nota>SI</Nota>
  <Traducir>
    <Tgrupo>
      <Tipo>pa</Tipo>
      <Rango>
        <Inicio>1</Inicio>
        <Final>20</Final>
        <Prefijo-rango>Parrafo</Prefijo-rango>
      </Rango>
      <Rango>
        <Inicio>37</Inicio>
        <Final>45</Final>
        <Prefijo-rango/>
      </Rango>
    </Tgrupo>
    <Tgrupo>
      <Tipo>li</Tipo>
      <Rango>
        <Inicio>1</Inicio>
        <Final></Final>
        <Prefijo-rango>Lista</Prefijo-rango>
      </Rango>
    </Tgrupo>
    <Tgrupo>
      <Tipo>tc</Tipo>
      <Rango>
        <Inicio>1</Inicio>
        <Final>1</Final>
        <Prefijo-rango>Indice</Prefijo-rango>
      </Rango>
    </Tgrupo>
  </Traducir>
</Audio>
```

Fig. 32: Ejemplo pruebaConf1c.xml

En cambio en este segundo caso si que nos genera cuatro archivos, esto es así porque el segundo grupo del primer *Tgrupo* no se ha dado valor a *Prefijo-rango* por lo que su nombre se obtiene del que da la aplicación por defecto (se hubiera obtenido el mismo tipo de resultado si se definiesen dos valores diferentes de *Prefijo-rango*). Los nombres de los archivos creados han sido: *Prueba7G1Parrafo.aif*, *Prueba7G1PA2.aif*, *Prueba7G2Lista.aif* y *Prueba7G3Indice.aif*.

También en este caso para comprobar que el resultado era el correcto he hecho servir el archivo content.xml, para los dos archivos de los párrafos. En el primer archivo deberíamos tener el contenido de 20 párrafos y sólo nos traduce una línea de texto, pero viendo en el xml se pude comprobar que es correcto porque los diecinueve primeros párrafos están vacíos. Lo demás se comprobó como en el caso anterior.

6.4.2 Caso de prueba 2

En este segundo caso de ejemplo de prueba también utiliza el idioma castellano, pero en este caso se ha elegido que el resultado esté en un fichero de audio único y sólo tenemos un grupo a tratar de tipo párrafo y desde el inicio hasta que se hayan tratado 10 párrafos. A continuación vemos el contenido del archivo de configuración (pruebaConf2.xml):

```
<?xml version="1.0" encoding="UTF-8"?>
<Audio>
    <Idioma>es</Idioma>
    <Imagen>SI</Imagen>
    <Unico>SI</Unico>
    <Prefijo>Prueba1</Prefijo>
    <Nota>SI</Nota>
  <Traducir>
    <Tgrupo>
      <Tipo>pa</Tipo>
      <Rango>
        <Inicio>1</Inicio>
        <Final>10</Final>
      </Rango>
    </Tgrupo>
  </Traducir>
</Audio>
```

Fig. 33: Ejemplo pruebaConf2.xml

El archivo ODF sobre el que se realiza el tratamiento se llama *HelloWorldb.odt*.

El resultado obtenido es un archivo de nombre *Prueba1G1.aif*, en el que nos ha grabado todo el texto que debía. Si miramos el documento de texto con el editor es difícil realizar la comprobación, se trata de un texto en que parte está escrito en inglés y parte en castellano, con lo que la parte en inglés la pronuncia como si fueran palabras en castellano. También la imagen contiene texto oculto que es traducido (se trata del título y la descripción) y también tiene nota al pie de página que se traduce antes del final de página. El texto '*Que me escribirá???*' que se traduce antes que '*Prueba primera*' esto con el editor parece erróneo, pero no lo es porque el primer texto se encuentra dentro de un text-box (en una imagen) y éste está anclado antes del otro texto, esto se comprueba muy bien abriendo el archivo content.xml, además podemos comprobar que este texto sólo consta de ocho párrafos, con lo que traduce hasta el final del documento.

6.4.3 Caso de prueba 3

En este caso de prueba probaremos con otro idioma, que será el inglés. En este caso se ha puesto todos los campos de la cabecera de configuración a “SI” y como prefijo “Prueba5”. Al tener un sólo grupo definido en el detalle, se espera que como resultado obtengamos un solo archivo de audio cuyo nombre deberá empezar por *Prueba5*. El fichero sobre el que realizaremos el tratamiento es el *Sections.odt* y el de configuración *pruebaConf5.xml*. Veamos como es el fichero de configuración:

```
<?xml version="1.0" encoding="UTF-8"?>
<Audio>
    <Idioma>en</Idioma>
    <Imagen>SI</Imagen>
    <Unico>SI</Unico>
    <Prefijo>Prueba5</Prefijo>
    <Nota>SI</Nota>
  <Traducir>
    <Tgrupo>
      <Tipo>to</Tipo>
      <Rango>
        <Inicio>1</Inicio>
        <Final></Final>
      </Rango>
    </Tgrupo>
  </Traducir>
</Audio>
```

Fig. 34: Ejemplo pruebaConf5.xml

El resultado obtenido es la traducción a voz de todo el este archivo de prueba ODF, que ha sido grabado en *Prueba5G1.aif*. En este caso si que el acento inglés es correcto ya que traduce un texto escrito en el mismo idioma. Es fácil de reseguir mirando el documento con el editor, aunque tiene referencias a otro documento cuyo contenido no es traducido, también hay que tener en cuenta que las notas son traducidas en el punto donde está la marca de referencia. En este caso también, como segunda comprobación, he utilizado el archivo content.xml del documento para comprobar que efectivamente se había traducido correctamente.

Se ha detectado problemas con la tabla que contiene el documento, que es que no se aprecia con nitidez todos los cambios de celdas. También el programa descarta las etiquetas de comentarios '*<office:annotation>*' y también el campo que registra el seguimiento de la modificación de texto sólo traduce el texto actual el anterior a la modificación no lo traduce (etiqueta '*<text:tracked-changes>*'), ya que el texto modificado se encuentra al inicio del documento y está oculto, y en el lugar donde el editor visualiza el histórico de cambios sólo se encuentran las referencias a donde está el texto.

Capítulo 7: Conclusiones

7.1 Logros conseguidos

Si miramos atrás nos habíamos puesto como metas a alcanzar en este proyecto:

- Estudio del formato XML y de las DTD para su validación.
- Estudio del formato OpenOffice.
- Estudio de los formatos de audio.
- Estudio de la tecnología Text-To-Speech.
- Elaboración de una aplicación que utilizase la tecnología estudiada en este proyecto.

Podemos ver que estos puntos se han alcanzado con mayor o menor medida, dándome la posibilidad de desarrollar una aplicación, dentro de los tiempos previstos en la planificación.

7.2 Metas no alcanzadas

Inicialmente me había propuesto para el desarrollo de la aplicación el poder permitir seleccionar por páginas el texto a traducir a voz, pero tras buscar una manera de saber en que página se encontraba el trozo de texto que se seleccionaba en ese momento y no encontrar nada que fuera más o menos directo de obtener, tuve que decidirme por la opción de elegir el punto de inicio y final contando elementos.

Para poder llegar a saber la página se tendría que saber el tamaño del papel, la dimensión que ocupa cada letra y su ancho, para poder calcular en que posición de la página nos encontramos. Para esto además hay que ir mirando los diferentes estilos de letras que se utilizan en el texto ya que cada uno puede tener un tamaño diferente. Se podría también haber ido contando los diferentes saltos de página encontrados en el texto, pero hay que tener en cuenta que también hay estilos que tienen definido un salto de página, con lo que se complica mucho. Además hay que decir que en la propia librería ODF los métodos que crean automáticamente una tabla de contenidos[36] (índices) advierte de que las referencias a las páginas les pone el valor 1 y que hay que entrar después en el menú de AOO[37] y seleccionar la opción de actualizar todos los índices y

36 Véase la clase TextDocument.class dentro de la librería simple-odf-0.8.1-incubating.jar por ejemplo el método createDefaultTOC

37 Abreviación de Apache OpenOffice

tablas para poner los valores correctos.

También había pensado en hacer la aplicación para que no tuviese que acceder a Internet, pero después de ver la facilidad ofrecida por Google y la posibilidad de trabajar con 80 idiomas diferentes me decanté por esta última opción, a pesar de que Google trabaja con ficheros mp3 y se haya tenido que realizar un proceso de conversión de tipo de audio.

7.3 Valoración personal

Antes de empezar el proyecto era consciente de la importancia de los formatos libres, pero desconocía totalmente lo que era OpenDocument a pesar de utilizar OpenOffice.org para redactar documentos.

Ahora, después del estudio realizado sobre este formato, he descubierto su arquitectura interna y el gran potencial de los documentos XML en el que se basa fundamentalmente estos documentos ODF. Sabía que este lenguaje de etiquetas se estaba utilizando cada vez más, pero nunca me hubiera imaginado que se usase en una Suite ofimática.

En lo referido a la tecnología de la conversión de texto a voz, considero que es una tecnología muy importante que se está aplicando cada vez más en el entorno que nos rodea. Ya sea para ayudar a personas con discapacidades visuales en el día a día, como en el ámbito de la enseñanza. He podido encontrar muchas librerías para su tratamiento y procesado pero la mayoría para el idioma inglés, faltando voces en castellano o catalán, y la mayoría de las veces con una calidad baja. Sin embargo las utilidades de pago tenían una calidad mejor.

Con el estudio realizado para el proyecto también me ha dado la oportunidad de descubrir que los formatos de audio que conocía son formatos propietarios, y para no tener que pagar derechos por sus utilizaciones se deben utilizar formatos abiertos libres y que la mayoría de ellos no tiene nada que envidiar de los propietarios.

En definitiva este proyecto me ha dado la oportunidad de adquirir unos conocimientos que estoy seguro que podré utilizar, en gran parte, en mi vida laboral porque es la nueva tendencia y muchos documentos oficiales ya requieren ser presentados en formatos XML determinados.

Glosario

AIFF: Acrónimo de Audio Interchange File Format. Es un estándar de formato de audio usado para almacenar datos de sonido en computadoras personales. Los datos no están comprimidos y se almacenan en big-endian y emplea una modulación por impulsos codificados (PCM).

Android Market: Actualmente se llama Google Play Store, es una plataforma de distribución digital de aplicaciones móviles para los dispositivos Android, así como una tienda en línea desarrollada y operada por Google.

API: Acrónimo del inglés Application Programming Interface. Es el conjunto de funciones, variables y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción, permitiendo a los programadores beneficiarse de las ventajas que proporciona evitándose el trabajo de programar todo desde el principio.

ASR: Acrónimo del inglés Automatic Speech Recognition. Se trata del reconocimiento de voz automatizado.

Bitrates: Tasa de bits por segundo que tiene un archivo multimedia.

Códec: Es la abreviatura de codificador-decodificador.

DOM: Acrónimo del inglés Document Object Model es una interfaz de programación de aplicaciones (API) que proporciona un conjunto estándar de objetos para representar documentos HTML y XML, gracias a esto permite acceder y manipular estos documentos.

DTD: Acrónimo del inglés Document Type Definition, es el archivo que permite validar el contenido de un documento XML al que hace referencia, mediante unas simples reglas sintácticas que definen los elementos y su orden dentro del documento.

FLAC: Acrónimo del inglés Free Lossless Audio Codec. Es un códec de audio que permite comprimir el audio sin pérdidas de tal manera que el archivo se ve reducido entre un 50 a 60% sin que se pierda ningún tipo de información.

GML: Generalized Markup Language fue inventado en los años setenta por IBM para cubrir el problema que tenían para almacenar grandes cantidades de información.

ISO/IEC: Acrónimo de International Organization for Standardization.

Glosario

Java Sound API: Librería Java que proporciona soporte de bajo nivel a operaciones de audio tales como la reproducción de audio y captura (grabación), mezcla, secuenciación MIDI y síntesis MIDI en un framework extensible y flexible.

Matroska: Es un formato contenedor de estándar abierto que puede contener una cantidad ilimitada de contenidos audiovisuales y multimedia dentro de un solo archivo. Permite reproducir el archivo tanto en ordenadores como en otros dispositivos con la suficiente potencia de procesamiento.

MIME: Acrónimo de Multipurpose Internet Mail Extensions (en español "extensiones multipropósito de correo de internet") son una serie de convenciones o especificaciones dirigidas al intercambio a través de Internet de todo tipo de archivos de forma transparente para el usuario y que se puedan identificar fácilmente.

OASIS: Open Document Format for Office Applications, también referido como formato OpenDocument (ODF).

ODF: OpenDocument es un formato de fichero abierto y estándar para almacenar documentos ofimáticos como hojas de cálculo, textos, gráficos y presentaciones, creado por OASIS.

ODFDOM: Es una librería gratuita de de OpenDocument Format, tiene como propósito ofrecer una manera fácil de crear, acceder y manipular ficheros ODF.

Ogg: Es un formato libre y abierto de contenedor, desarrollado y mantenido por la Fundación Xiph.Org. Puede multiplexar varios flujos independientes para audio, vídeo, texto y metadatos.

SGML: Acrónimo de Standard Generalized Markup Language, consiste en un sistema para la organización y etiquetado de documentos.

Síntesis concatenativa: Es la síntesis que se basa en la concatenación de segmentos de voz grabados.

Síntesis de formantes: Es la síntesis que no usa muestras de voz humana, sino que la salida se crea usando un modelo acústico.

Text-To-Speech: La conversión de texto-voz es la generación por medios automáticos de una voz artificial que genera el sonido que produciría una persona al leer un texto cualquiera en voz alta o una voz artificial. Se conoce también por las siglas CTV o por las siglas en inglés TTS (Text To Speech).

Transductor: Dispositivo que recibe la potencia de un sistema mecánico, electromagnético o

acústico y la transmite a otro, generalmente en forma distinta (ejemplo: el micrófono y el altavoz son transductores).

Vorbis: Códec de audio digital general con pérdidas, desarrollado por la Fundación Xiph.Org, utiliza el formato de archivo Ogg.

Voz sintética: Se trata de una voz artificial (no grabada) que está generada por un proceso de sintetización del habla, esto es la producción artificial de habla humana.

W3C: World Wide Web Consortium es un consorcio internacional que produce recomendaciones para la World Wide Web.

XML: Acrónimo del inglés e**X**tended **M**arkup **L**anguage, se trata de un metalenguaje extensible de etiquetas. Es una simplificación y adaptación del SGML, y permite el intercambio de datos entre diferentes plataformas.

Bibliografía

Wikipedia. Extensible Markup Language.
http://es.wikipedia.org/wiki/Extensible_Markup_Language [on-line] Fecha última consulta: 28/09/14

RI5. http://www.ri5.com.ar/ayuda07.php [on-line] Fecha última consulta: 28/09/14

Documentos XML. F. Javier García Castellano. http://flanagan.ugr.es/xml/documento.htm [on-line]
Fecha última consulta: 28/09/14

DTD: Definición de Tipo de Documento XML. Bartolomé Sintes Marco.
http://www.mclibre.org/consultar/xml/lecciones/xml_dtd.html#L843 [on-line]
Fecha última consulta: 01/10/14

DTDs y XML Esquema. María Jesús Lamarca Lapuente.
http://www.hipertexto.info/documentos/dtds.htm [on-line] Fecha última consulta: 01/10/14

XML: DTD. http://mercurio.ugr.es/pedro/tutoriales/cursos/xml/dtd.htm [on-line]
Fecha última consulta: 01/10/14

FAO.org. http://www.fao.org/docrep/009/ae908s/ae908s03.htm [on-line]
Fecha última consulta: 01/10/14

Wikipedia. Apache OpenOffice. http://es.wikipedia.org/wiki/Apache_OpenOffice [on-line]
Fecha última consulta: 01/10/14

OpenOffice.org. About Apache OpenOffice. http://www.openoffice.org/about/ [on-line]

Fecha última consulta: 02/10/14

LibreOffice Help. Formatos de archivos XML.
https://help.libreoffice.org/Common/XML_File_Formats/es [on-line]
Fecha última consulta: 02/10/14

Wikipedia. OpenDocument. http://es.wikipedia.org/wiki/OpenDocument [on-line]
Fecha última consulta: 03/10/14

Observatorio Tecnológico. Open Document.
http://recursostic.educacion.es/observatorio/web/ca/software/software-general/660-open-document
[on-line] Fecha última consulta: 03/10/14

Bibliografía

OASIS. Open Document Format for Office Aplications. https://www.oasis-open.org/committees/download.php/12572/OpenDocument-v1.0-os.pdf Fecha última consulta: 05/10/14

OASIS OpenDocument Essentials. http://books.evc-cit.info/odbook/book.html Fecha última consulta: 05/10/14

AODL. Apache OpenOffice Wiki. https://wiki.openoffice.org/wiki/AODL [on-line] Fecha última consulta: 05/10/14

Apache Incubator. Apache ODF Toolkit Incubation Status. http://incubator.apache.org/projects/odftoolkit.html [on-line] Fecha última consulta: 05/10/14

Apache OpenOffice Wiki. Java Eclipse Tuto. https://wiki.openoffice.org/wiki/JavaEclipseTuto [on-line] Fecha última consulta: 06/10/14

Apache OpenOffice Wiki. ODF Toolkit. https://wiki.openoffice.org/wiki/ODF_Toolkit [on-line] Fecha última consulta: 06/10/14

Apache Incubator. Apache ODF Toolkit (Incubating). http://incubator.apache.org/odftoolkit/index.html [on-line] Fecha última consulta: 06/10/14

Wikipedia. Formato de archivo de audio. http://es.wikipedia.org/wiki/Formato_de_archivo_de_audio [on-line] Fecha última consulta: 08/10/14

Doc.ubuntu-es. Formatos libres. http://doc.ubuntu-es.org/Formatos_libres [on-line] Fecha última consulta: 08/10/14

Diseño de Materiales Multimedia. Formatos de audio. http://www.ite.educacion.es/formacion/materiales/107/cd/audio/audio0102.html [on-line] Fecha última consulta: 08/10/14

El Baúl del Programador. ¿Cual es la diferencia entre los distintos formatos de audio y cual debería elegir? http://elbauldelprogramador.com/cual-es-la-diferencia-entre-los-distintos-formatos-de-audio-y-cual-deberia-elegir/ [on-line] Fecha última consulta: 08/10/14

Wikipedia. Matroska. http://es.wikipedia.org/wiki/Matroska [on-line] Fecha última consulta: 09/10/14

Wikipedia. Vorbis. http://es.wikipedia.org/wiki/Vorbis [on-line] Fecha última consulta: 09/10/14

Bibliografía

Wikipedia. Ogg. http://es.wikipedia.org/wiki/Ogg [on-line] Fecha última consulta: 10/10/14

Matroska.info. ¿Que es Matroska?. http://www.matroska.info/que-es-matroska#track [on-line] Fecha última consulta: 09/10/14

Wikipedia. Free Lossless Audio Codec. http://es.wikipedia.org/wiki/FLAC [on-line] Fecha última consulta: 11/10/14

Wikipedia. Au (formato de archivo). http://es.wikipedia.org/wiki/Au_(formato_de_archivo) [on-line] Fecha última consulta: 12/10/14

Wikipedia. Audio Interchange File Format. http://es.wikipedia.org/wiki/Audio_Interchange_File_Format [on-line] Fecha última consulta: 12/10/14

Informática-hoy. Formatos de audio digital AIFF. http://www.informatica-hoy.com.ar/multimedia/Formatos-audio-digital-AIFF.php [on-line] Fecha última consulta: 12/10/14

JavaZoom. Ogg Vorbis SPI. http://www.javazoom.net/vorbisspi/vorbisspi.html [on-line] Fecha última consulta: 12/10/14

Java FLAC Codec. Project Information. http://jflac.sourceforge.net/project-info.html [on-line] Fecha última consulta: 12/10/14

The Java Tutorials. Trail: Sound. http://docs.oracle.com/javase/tutorial/sound/ [on-line] Fecha última consulta: 12/10/14

Wikipedia. Conversor texto-voz. http://es.wikipedia.org/wiki/Conversor_texto-voz [on-line] Fecha última consulta: 13/10/14

Wikipedia. Síntesis de habla. http://es.wikipedia.org/wiki/Voz_sint%C3%A9tica [on-line] Fecha última consulta: 14/10/14

W3C. Speech Synthesis Markup Language (SSML) Version 1.1. http://www.w3.org/TR/speech-synthesis11/ [on-line] Fecha última consulta: 17/10/14

Thierry Dutoit. A Short Introduction to Text-to-Speech Synthesis. http://tcts.fpms.ac.be/synthesis/introtts_old.html [on-line] Fecha última consulta: 18/10/14

ZAMZAR. Convert documents into speech. http://blog.zamzar.com/2009/04/07/convert-documents-into-speech-yes-speech-doc-pdf-txt-and-more-to-mp3/ [on-line] Fecha última consulta: 18/10/14

Bibliografía

Herramientas 2.0 para la enseñanza de idiomas. Text-to-speech. https://sites.google.com/site/herramientasidiomas/4a-sesion [on-line] Fecha última consulta: 18/10/14

Findbestopensource. 37 Open source text-to-speech. http://www.findbestopensource.com/tagged/text-to-speech [on-line] Fecha última consulta: 18/10/14

JSAPI:Home. Java Speech API. http://jsapi.sourceforge.net/index.html [on-line] Fecha última consulta: 20/10/14

Oracle. Java Speech API Frequently Asked Questions. http://www.oracle.com/technetwork/java/jsapifaq-135248.html [on-line] Fecha última consulta: 20/10/14

Wikipedia. FreeTTS. http://es.wikipedia.org/wiki/FreeTTS [on-line] Fecha última consulta: 20/10/14

Sourceforge. FreeTTS 1.2. http://freetts.sourceforge.net/docs/index.php [on-line] Fecha última consulta: 20/10/14

Wikipedia. eSpeak. http://en.wikipedia.org/wiki/ESpeak [on-line] Fecha última consulta: 20/10/14

Sourceforge. eSpeak text to speech. http://espeak.sourceforge.net/ [on-line] Fecha última consulta: 20/10/14

iSpeech. Text-to-speech. http://www.ispeech.org/#/home [on-line] Fecha última consulta: 20/10/14

Code.Google.com. Java-google-translate-text-to-speech. http://code.google.com/p/java-google-translate-text-to-speech/ [on-line] Fecha última consulta: 21/10/14

Sergio. De texto a MP3 a través de URL con Google Translate. http://www.enlanubetic.com.es/2012/02/de-texto-mp3-traves-de-url-con-google.html [on-line] Fecha última consulta: 21/10/14

Programa de conversión de texto a voz

Manual de Usuario

INDICE

1 Instalación

Lo primero que hay que tener en cuenta es que esta aplicación necesita tener instalado la máquina virtual de Java versión 7 o superior.

En el caso de que no lo tengamos se puede obtener esta versión en el siguiente enlace http://www.oracle.com/technetwork/java/javase/downloads/java-se-jre-7-download-432155.html

Una vez comprobado que se tiene la máquina virtual necesaria para la ejecución, deberemos instalar (o colocar) los siguientes ficheros en la misma carpeta (directorio de archivos):

Audio.dtd

VentDatosPFC.jar

Estos dos archivos vienen dentro del empaquetado en el que se ha entregado la memoria.

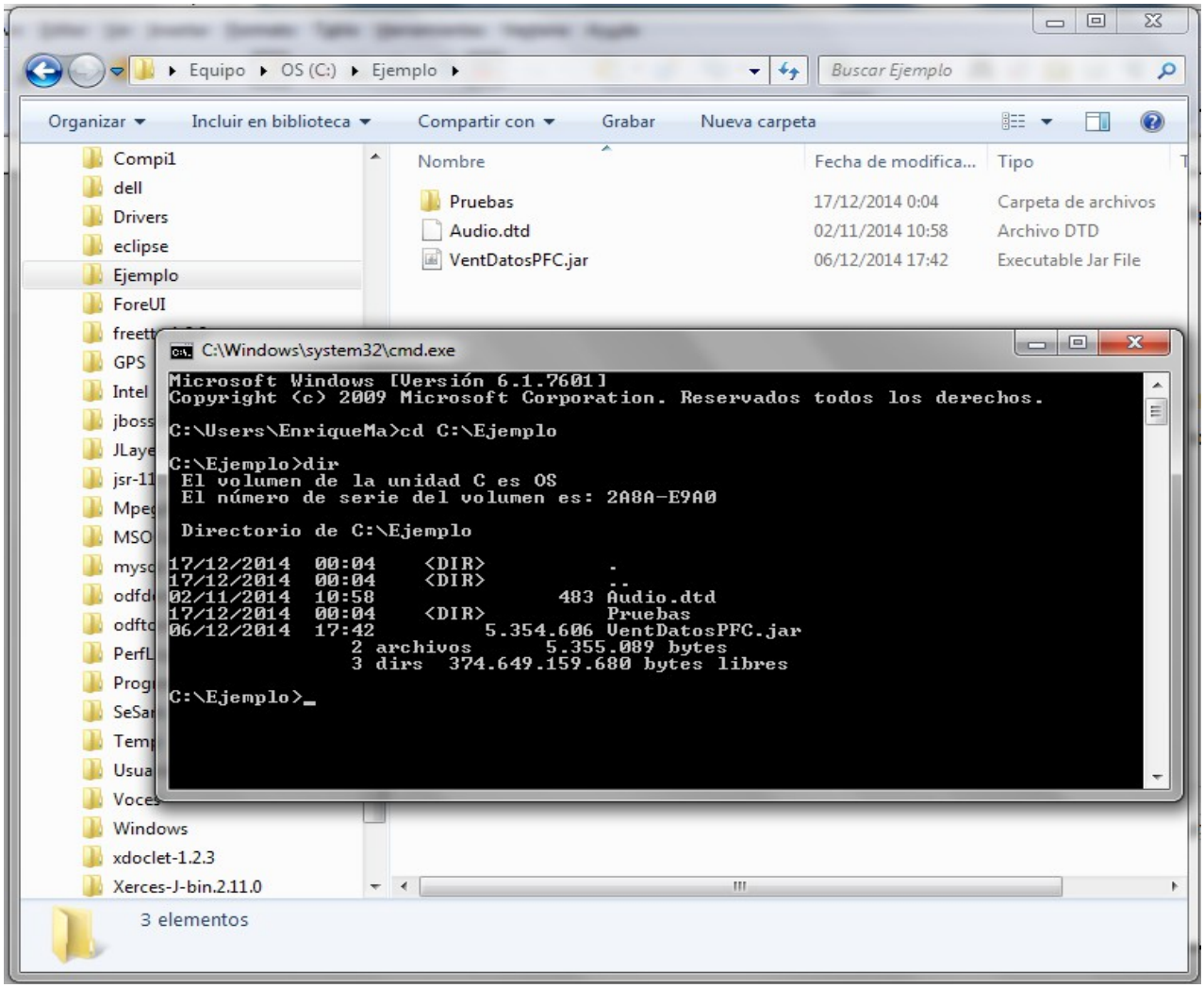
2 Preparación para ejecutar la aplicación

Para la ejecución de la aplicación necesitaremos disponer de al menos un fichero de configuración y un documento de texto con extensión odt.

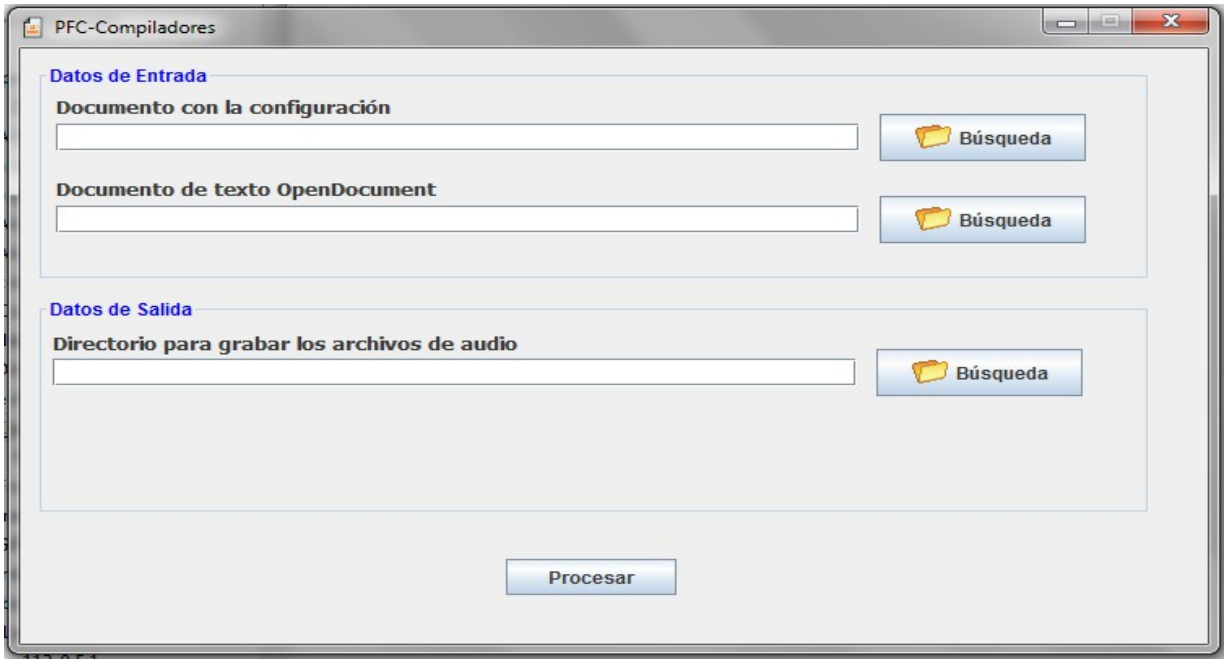
3 Ejecución de la aplicación

Para ejecutar la aplicación, lo podremos hacer de dos maneras diferentes, desde la línea de comandos o desde el icono de la aplicación que se se encuentra en el directorio que hemos hecho la instalación.

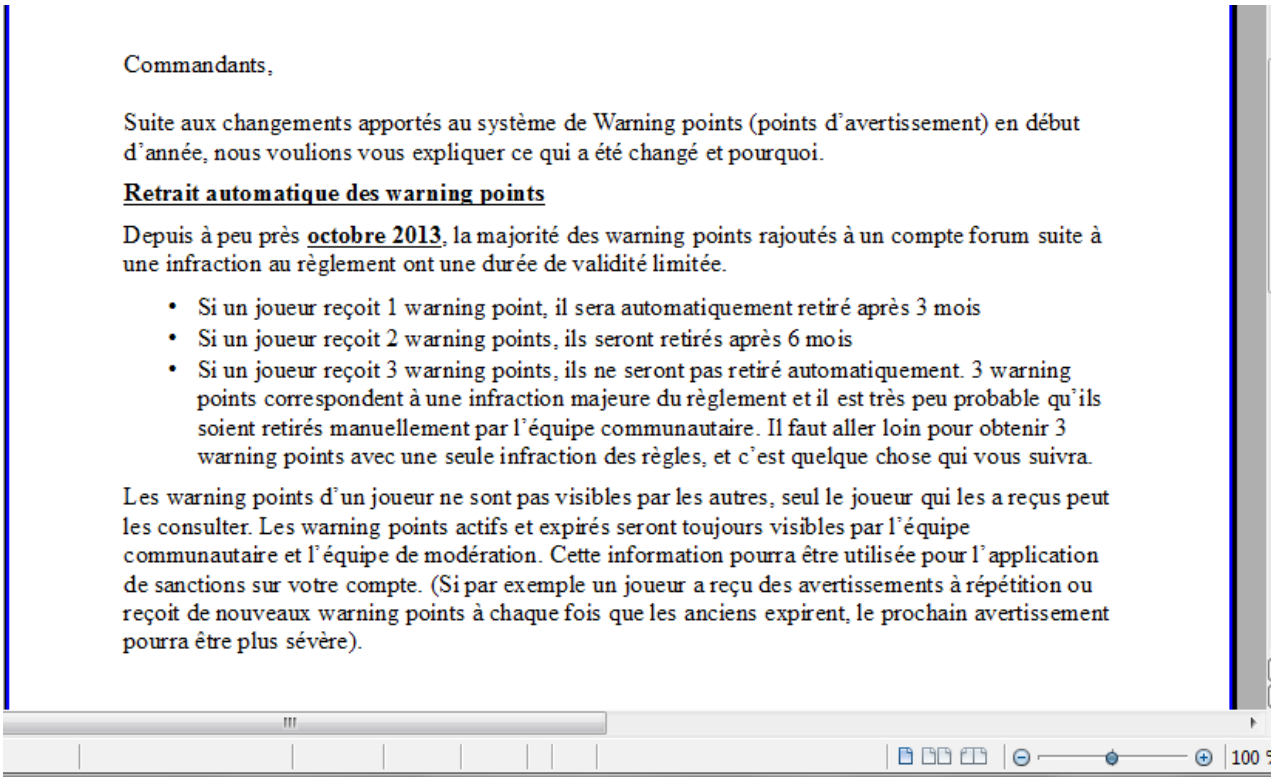
En ambos casos se tiene que ejecutar el programa VentDatosPFC.jar. Para hacer las pruebas propuestas en este manual se ha instalado el programa en el directorio C:\Ejemplo como podemos ver en la imagen siguiente:



En este ejemplo lo ejecutaremos haciendo doble *clic* en el correspondiente icono del programa, obteniendo la ventana principal del programa



En este primer ejemplo miraremos como podemos traducir un texto que está en el idioma Francés y conseguir obtener un audio en este idioma del texto que tenemos escrito, de esta manera nos permitirá a través del resultado poder practicar en el estudio del idioma, viendo como se pronuncian las palabras. El documento tiene como nombre *TextoFrances.odt*.



Commandants,

Suite aux changements apportés au système de Warning points (points d'avertissement) en début d'année, nous voulions vous expliquer ce qui a été changé et pourquoi.

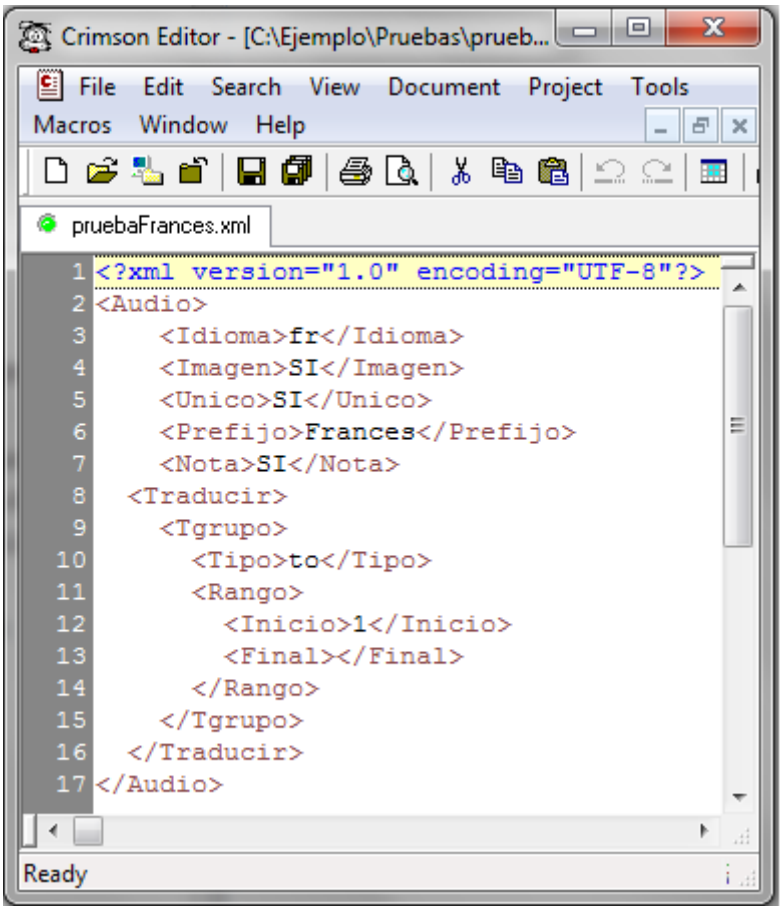
Retrait automatique des warning points

Depuis à peu près **octobre 2013**, la majorité des warning points rajoutés à un compte forum suite à une infraction au règlement ont une durée de validité limitée.

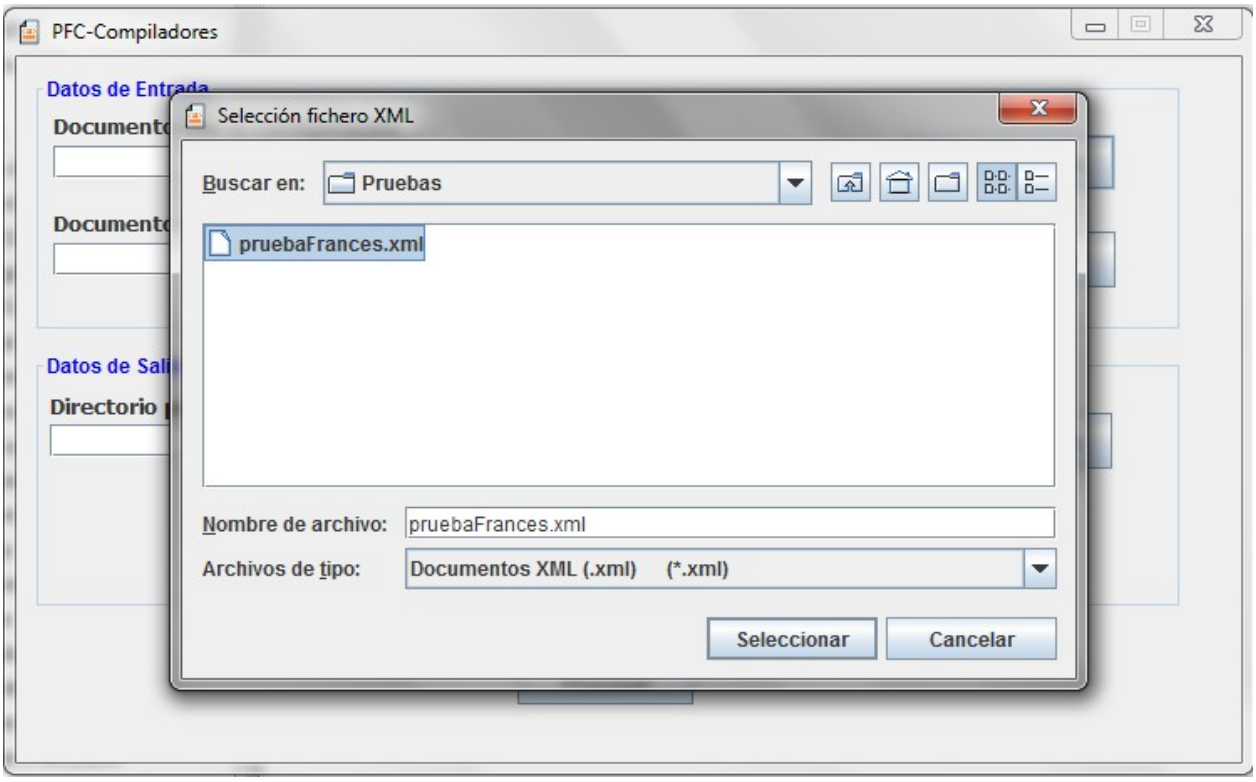
- Si un joueur reçoit 1 warning point, il sera automatiquement retiré après 3 mois
- Si un joueur reçoit 2 warning points, ils seront retirés après 6 mois
- Si un joueur reçoit 3 warning points, ils ne seront pas retiré automatiquement. 3 warning points correspondent à une infraction majeure du règlement et il est très peu probable qu'ils soient retirés manuellement par l'équipe communautaire. Il faut aller loin pour obtenir 3 warning points avec une seule infraction des règles, et c'est quelque chose qui vous suivra.

Les warning points d'un joueur ne sont pas visibles par les autres, seul le joueur qui les a reçus peut les consulter. Les warning points actifs et expirés seront toujours visibles par l'équipe communautaire et l'équipe de modération. Cette information pourra être utilisée pour l'application de sanctions sur votre compte. (Si par exemple un joueur a reçu des avertissements à répétition ou reçoit de nouveaux warning points à chaque fois que les anciens expirent, le prochain avertissement pourra être plus sévère).

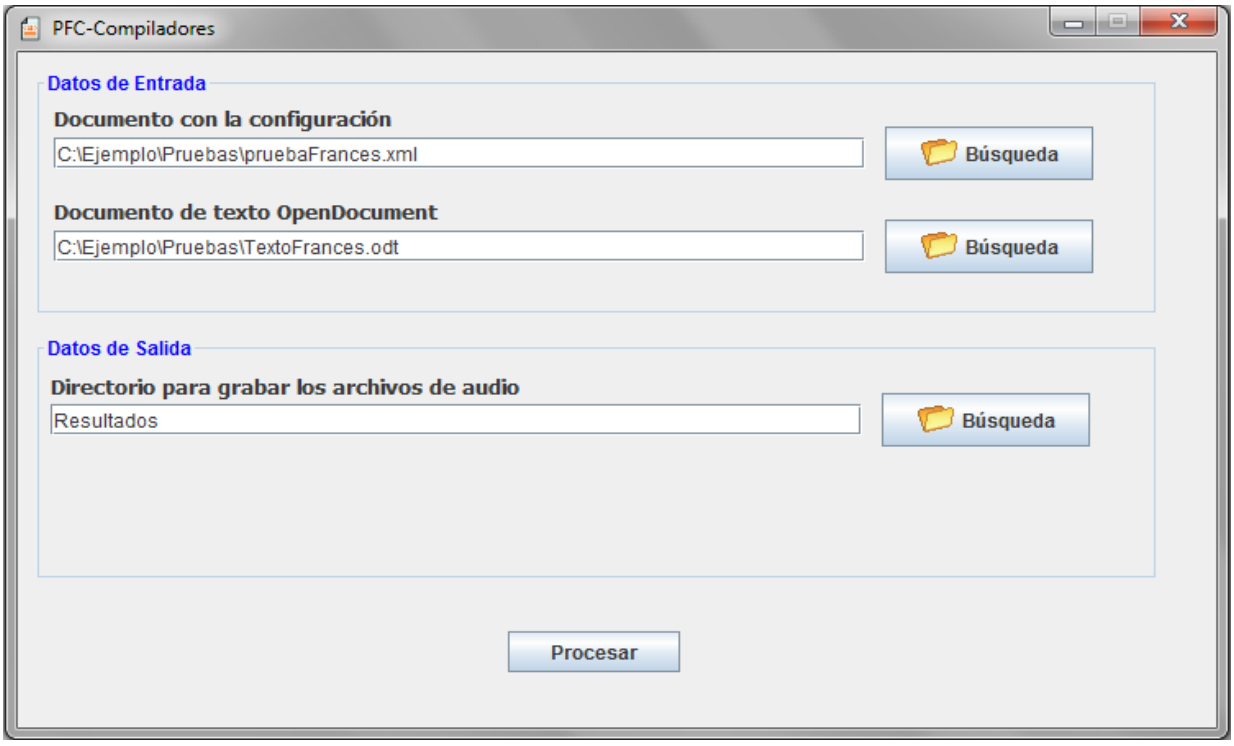
Para realizar la traducción utilizamos el siguiente documento de configuración *pruebaFrances.xml*, con los siguientes valores:



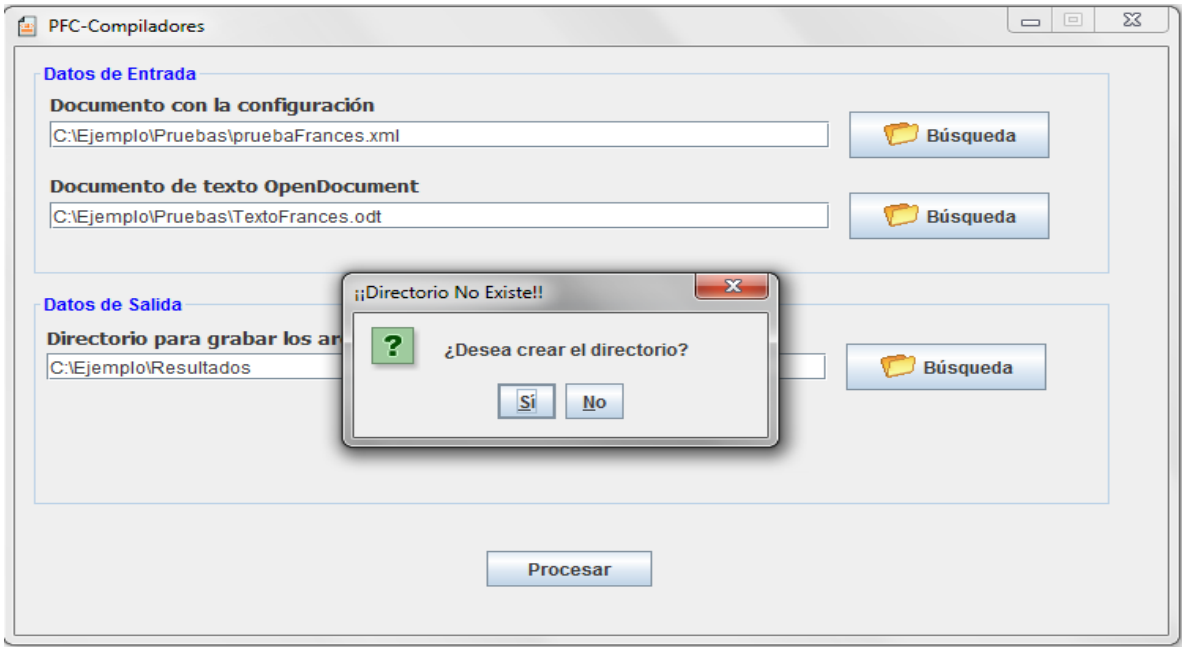
Ahora que tenemos estos dos documentos podemos proceder a probar el programa, el primer dato que nos pide lo podemos introducir escribiendo la ruta completa y el nombre del fichero, o sólo el nombre si nos encontramos trabajando en el mismo directorio en que se encuentra. En este caso completaré la información utilizando el botón de *Búsqueda* que se encuentra a la derecha del campo



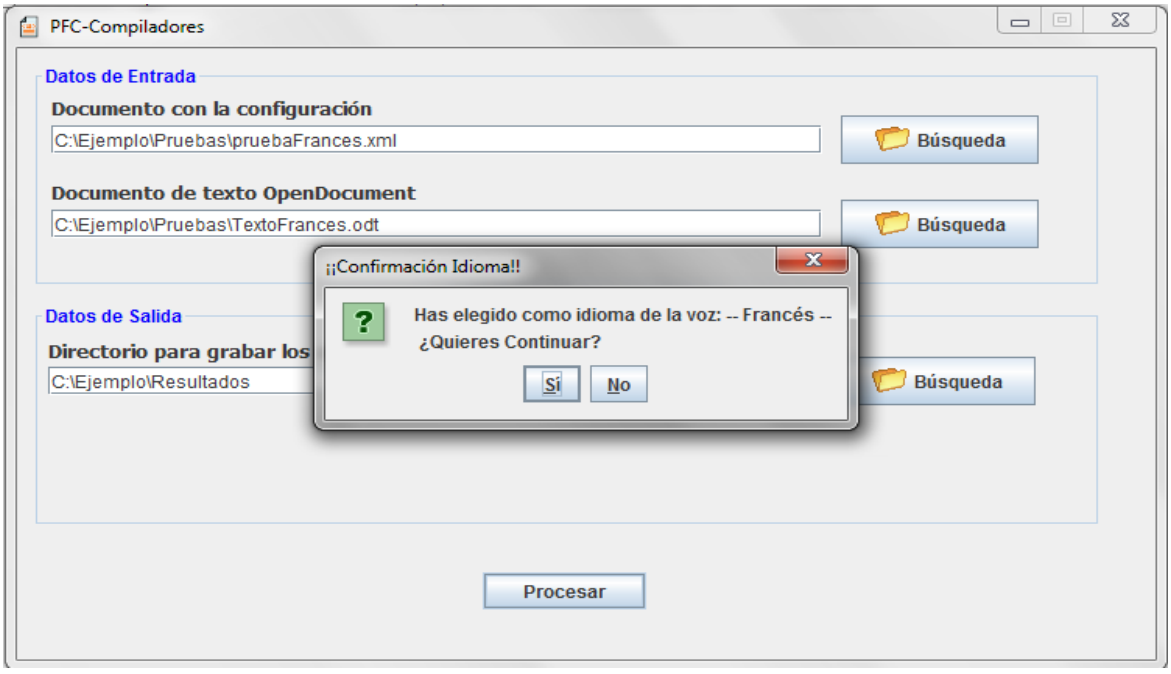
Para el siguiente dato, el documento ODF utilizo el botón de la derecha del correspondiente campo. Para el último campo que solicita, que es el directorio donde guardar el resultado, pondremos un directorio que no existe



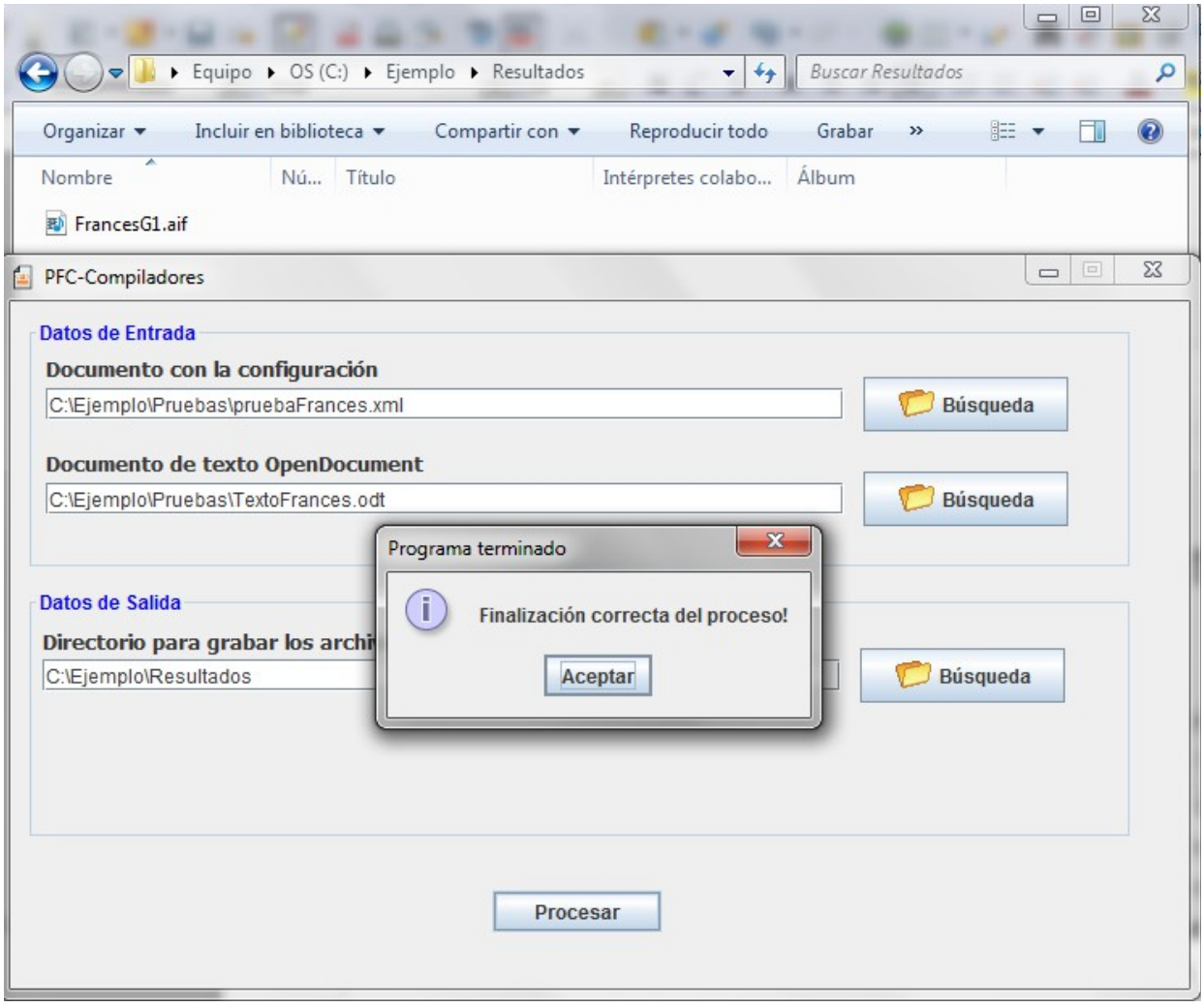
Al pulsar la tecla 'enter' (Intro) para aceptar lo que hemos escrito, el programa rellena la ruta completa (la ruta al directorio que estamos ejecutando el programa) y nos visualiza un mensaje preguntando si queremos crear el directorio, si contestamos que no nos permitirá modificarlo, en nuestro caso contestamos que si y cuando pulsemos el botón de procesar será el momento en que nos creará el nuevo directorio.



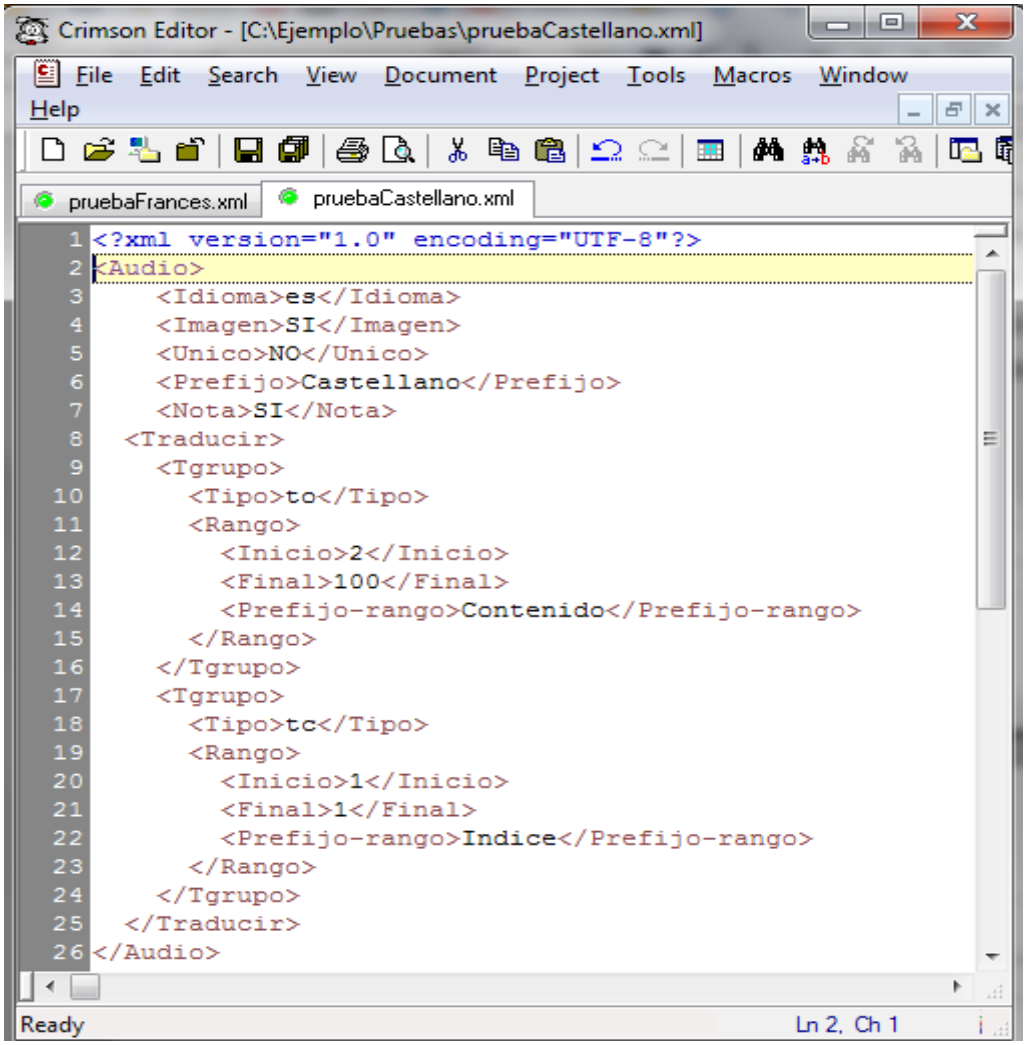
Ahora una vez aceptado este mensaje ya podemos proceder a pulsar el botón de *Procesar*. Nos visualizará un mensaje para que confirmemos el idioma seleccionado para la traducción

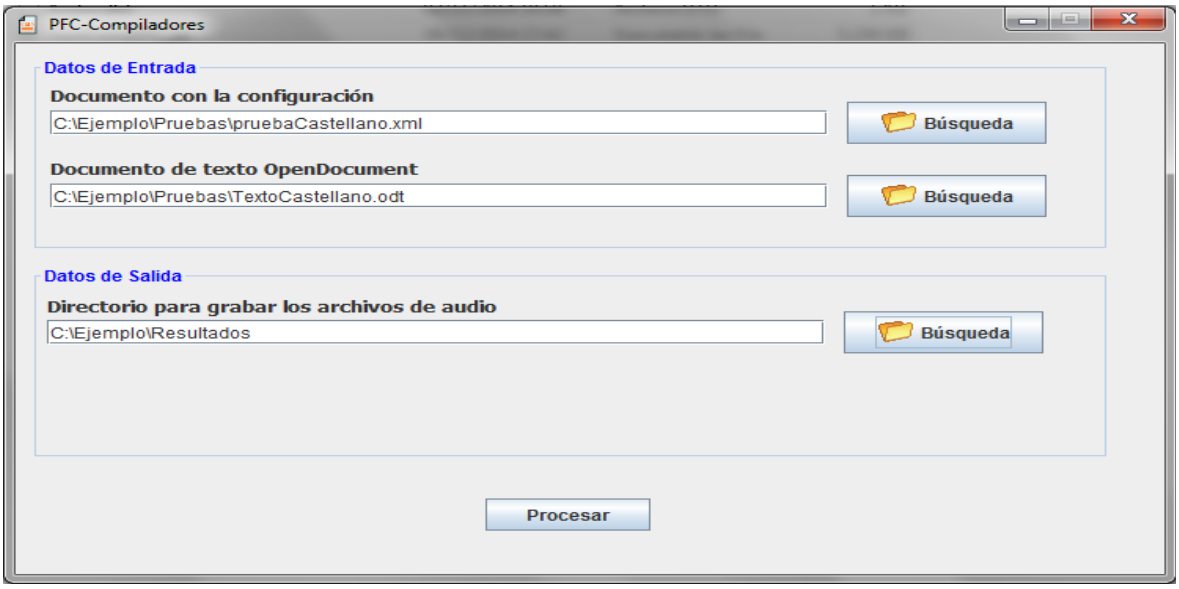


Al confirmar que queremos continuar, nos generará un archivo de audio con una locución correspondiente a la lectura del texto en el idioma Francés. El programa también nos avisa de cuando ha finalizado el proceso, en la siguiente imagen vemos como nos ha generado el archivo en el sitio solicitado, con el prefijo que habíamos puesto en el archivo XML, seguido de un sufijo que lo pone la aplicación porque algunas veces debe generar más de un archivo.

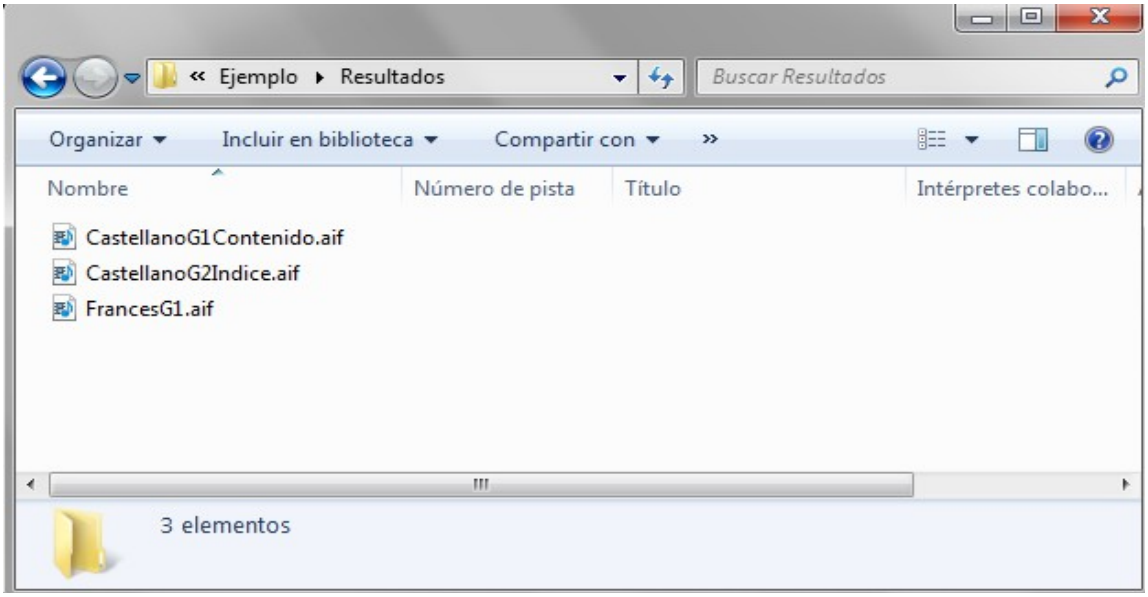


Como segunda prueba, utilizaremos un texto reducido en castellano de la PAC2 de este proyecto, en el que separaremos el resultado en dos archivos de audio, el primero contendrá el contenido y el segundo el índice de este contenido. Para conseguirlo hemos insertado dos grupos diferentes de traducción en el archivo XML, cada uno tendrá sufijos diferentes, pero el prefijo inicial será "*Castellano*", en la imagen siguiente vemos el contenido de este fichero XML que tratará el documento TextoCastellano.odt


Crimson Editor - [C:\Ejemplo\Pruebas\pruebaCastellano.xml]
File Edit Search View Document Project Tools Macros Window
Help
pruebaFrances.xml
pruebaCastellano.xml
<?xml version="1.0" encoding="UTF-8"?>
<Audio>
<Idioma>es</Idioma>
<Imagen>SI</Imagen>
<Unico>NO</Unico>
<Prefijo>Castellano</Prefijo>
<Nota>SI</Nota>
<Traducir>
<Tgrupo>
<Tipo>to</Tipo>
<Rango>
<Inicio>2</Inicio>
<Final>100</Final>
<Prefijo-rango>Contenido</Prefijo-rango>
</Rango>
</Tgrupo>
<Tgrupo>
<Tipo>tc</Tipo>
<Rango>
<Inicio>1</Inicio>
<Final>1</Final>
<Prefijo-rango>Indice</Prefijo-rango>
</Rango>
</Tgrupo>
</Traducir>
</Audio>
Ready
Ln 2, Ch 1

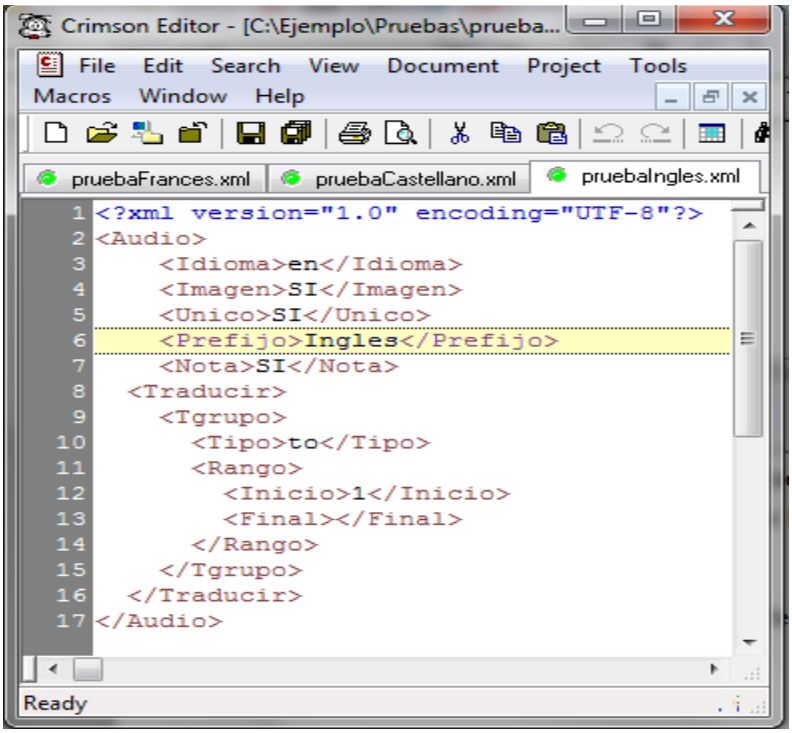

PFC-Compiladores
Datos de Entrada
Documento con la configuración
C:\Ejemplo\Pruebas\pruebaCastellano.xml
Búsqueda
Documento de texto OpenDocument
C:\Ejemplo\Pruebas\TextoCastellano.odt
Búsqueda
Datos de Salida
Directorio para grabar los archivos de audio
C:\Ejemplo\Resultados
Búsqueda
Procesar

Una vez finalizado el proceso, podemos comprobar que en el directorio de Resultados tenemos dos nuevos archivos, uno para el índice y otro para el contenido.

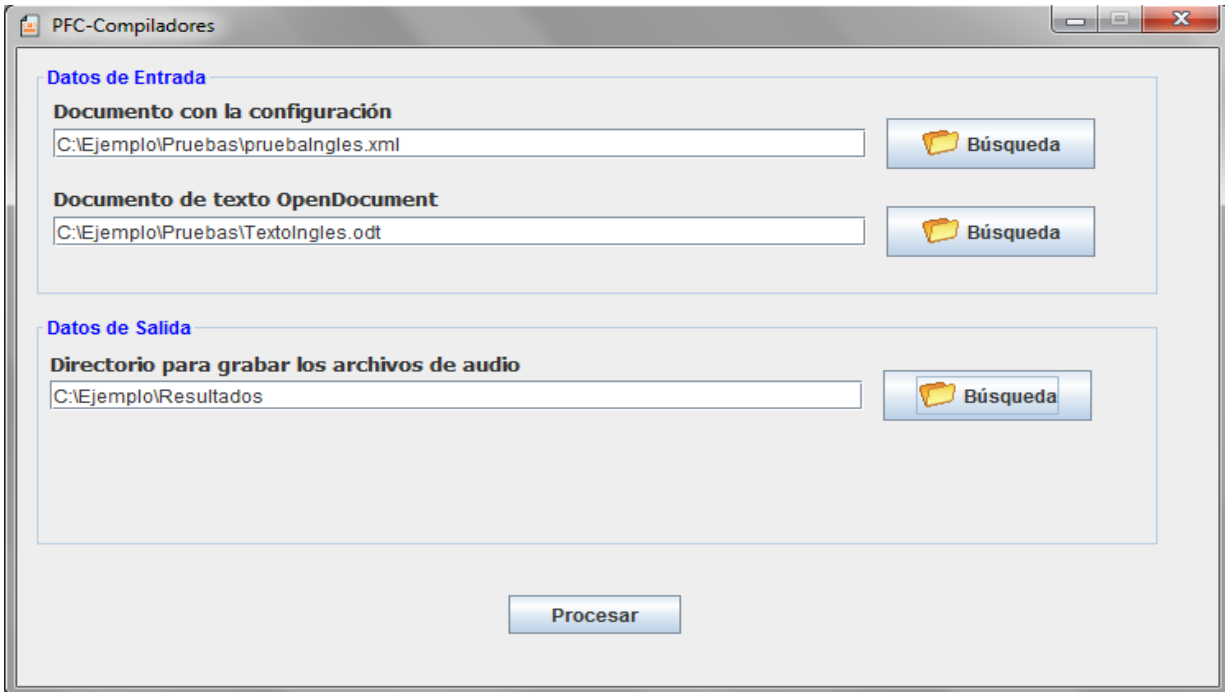


La última prueba que veremos es la traducción de un cuento infantil en inglés, ya que una de las utilidades de esta aplicación es facilitar la lectura y el poder oír el texto mientras lo leemos facilita la comprensión del mismo.

Para ello cogeremos como archivo de texto TextoIngles.odt donde tenemos introducido el cuento y utilizaremos el siguiente archivo XML:



Seguidamente ejecutamos el programa poniendo la siguiente información en los campos solicitados



Y como resultado obtendremos, *InglesG1.aif*, un archivo de audio con la voz que nos lee el cuento en inglés.