Universidad de Valladolid

E.T.S. INGENIERÍA INFORMÁTICA

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Estudio de formatos y técnicas de etiquetado de documentos y medios a bajo nivel.

Autor:
Unai Sarasola Álvarez

Tutor:
César Llamas Bello

Dedicado a aquellas personas que me han enseñado, que detrás de algo imposible, siempre hay una solución brillante.

Índice general

Resumen

Una dificultad importante en el mantenimiento de colecciones de documentos, es que a menudo nos encontramos con muchas relaciones implícitas acerca del tema que tratan o contenidos que poseen, que no somos capaces de explotar. Esta memoria pretende describir la forma y estados en las que se suelen presentar este tipo de colecciones y aportar una manera con la que explotar estas relaciones, de cara a que el usuario pueda ver y manipular estas relaciones entre las diferentes partes de los documentos.

Dado que la mayoría de contenido que nos encontramos en este tipo de colecciones es de texto, aunque también existe contenido audiovisual, hemos realizado un estudio para encontrar una manera para modificar los documentos de textos en los formatos más comunes y abiertos. Además hemos creado un sistema para ser capaces de establecer relaciones también dentro del contenido audiovisual.

Nuestra propuesta se ha basado en desarrollar un lenguaje de etiquetado que podamos incluir en los anteriores documentos. Dada la gran cantidad de conceptos que pueden aparecer dentro de los documentos de una colección, se ha establecido un mecanismo de clasificación, para lograr sin muchos tipos de etiquetas, abarcar todo el posible contenido que el usuario quiera etiquetar.

Finalmente y de cara a una posterior implementación, se han desarrollado una serie de ejemplos simples para instruir en el uso del lenguaje y para demostrar que el enfoque que se ha dado a través de toda la memoria, ha sido el válido.

Abstract

There is a difficult in the maintenance of a collection of documents, where a lot of times we have a lot of relationship between the content that appears in them, and we haven't a option to explore it. This memory pretends to describe the form and states that we can see in these kind of collections and to make a way to explore this relationships, allowing users to see and to modify this relationships in the different parts of the documents.

Because the most of the documents that we use, are based on text and some portion have audio or video content, we have made a study about the modification of the most common open formats. Also we have developed a system to mark up the relationships on the video/audio formats.

The proposal is based in the development of a markup language which allows you to mark up the documents studied. Because there are a variety of concepts that appears in the documents of a collection, we have set up a collection of tags which allow the user classify a lot of concepts without having a lot of different tags.

Finally oriented to the implementation scope, we have made examples with the language to clarify how it works, and to demonstrate that the approach used in this memory have been valid.

Capítulo 1

Introducción

En los últimos años, la sociedad ha experimentado un cambio que la ha llevado a utilizar nuevos formatos a la hora de realizar su trabajo, enseñar e incluso en el ámbito del entretenimiento. Todos aquellos libros y apuntes que ocupaban un gran espacio se han visto relegados en su mayoría por versiones digitales que nos permiten ahorrar tamaño y realizar una búsqueda y un tratamiento de la información mucho más veloz.

Por otro lado estos documentos, suelen tener una serie de contenidos asociados entre ellos, que no somos capaces de manejar. Por ejemplo dentro de un documento podemos ver como se habla de un determinado autor, o de un determinado tema, que también aparece en otro libro, pero no somos capaces de establecer relaciones entre estos dos libros de tal manera que nos sean de utilidad a la hora de buscar información o incluso de eliminarla de nuestra colección. (Por ejemplo, un profesor podría tener una serie de apuntes y una serie de exámenes de otros años, y podría querer ver que partes de los apuntes están relacionadas con cada pregunta de los exámenes).

El enfoque que se va a emplear, va a ser el de tomar un documento como un contenedor de contenido. Este contenido abarcará un espacio dentro del documento y nuestra misión deberá ser identificarlo de una manera unívoca y establecer relaciones con otros contenidos iguales o similares. Es debido a ésto, por lo que se ha buscado una manera de marcar estos documentos y establecer una jerarquía de marcas que permitan abarcar todos los casos posibles que se puedan dar.

Además, dado que esta colección de marcas con las que acotaremos las diferentes partes de los documentos, pueden o bien ser colocadas de manera externa al documento, o bien podrían ser incluidas dentro del propio documento de manera transparente al usuario, hemos decidido estudiar las diferentes posibilidades y viabilidad técnica, dentro de los tipos de documento más comunes que se emplean en el día a día. Solo se han tenido en cuenta, aquellos documentos

que son estándares oficiales o bien de facto y cuya especificación es pública.

Otro punto de estudio, deben ser los diferentes modelos que podemos tomar a la hora de relacionar partes del documento, con otras partes del documento, por lo que a lo largo de la memoria, se verán una serie de modelos de los cuales se ha analizado las ventajas y las desventajas que tiene cada uno.

Finalmente, una vez hemos escogido el modelo más adecuado para nuestros fines, desarrollaremos a lo largo de los capítulos, una jerarquía de etiquetas, con el fin de marcar el contenido y establecer relaciones entre los diferentes documentos.

1.1. Técnicas

Dado que estamos frente a un proyecto más cerca de la investigación que de lo que sería un proyecto puramente técnico donde tendríamos las fases o iteraciones propias de un proyecto de ingeniería, se han llevado a cabo las tres partes fundamentales en las que se dividiría un buen lenguaje basado sobre XML descritas en [1], que son:

- Especificación de los requisitos.
- Diseño de las etiquetas.
- Marcado de los documentos.

Debido a que en nuestro caso pretendemos insertar etiquetas o al menos comprobar su viabilidad sobre documentos ya existentes, hemos abordado esta actividad empleando los siguientes pasos:

- En un primer momento hemos abordado los diferentes formatos existentes en la actualidad sobre los que queríamos trabajar, investigando bien su contexto de uso, con el fin de ver qué necesidades surgen de éstos para relacionarlos entre sí, mediante nuestro lenguaje. (Esto podríamos considerarlo una especificación de requisitos, como ya una parte de marcado de los documentos, aunque hay que dejar claro, que finalmente será en la implementación del Software que se encargue de procesar estos documentos, en los que recaerá estas competencias, nosotros solo hemos dejado abiertas las diferentes posibilidades).
- En un segundo momento hemos definido una jerarquía de etiquetas, especificando que elementos y que etiquetas necesitamos para nuestro lenguaje, con el fin de que sea completo y usable para el usuario.

- Finalmente hemos probado su validez, aplicándolo a un ejemplo práctico y dando una serie de sugerencias de cara al desarrollo de Software posterior que pueda aprovechar el lenguaje de etiquetas aquí desarrollado.

1.2. Organización de la Memoria

La Memoria se ha distribuido en dos partes generales:

- Análisis del problema: donde se ha pretendido por un lado investigar sobre los formatos más adecuados que se emplean en el día a día. Analizar las diferentes características de cada uno para ver hasta que punto es complicado o sencillo modificarlos de manera transparente para nuestro provecho. Y una vez analizados esta parte, se han estudiado los diferentes modelos posibles de sincronización (también analizando sus ventajas y sus desventajas) para escoger el modelo que emplearemos dentro de nuestro lenguaje de etiquetado.
- Implementación del lenguaje: Una vez que se ha estudiado cual es la situación actual y se ha escogido un modelo para la sincronización, se ha ido construyendo una jerarquía de etiquetas que sirvan para nuestros propósitos. Una vez que esa jerarquía ha sido definida y validada, se ha procedido a definir el `DTD` del lenguaje, que nos permite validar de manera automática si un documento XML es válido según las restricciones especificadas en este DTD.

Parte I

Análisis del problema

Capítulo 2

Revisión de formatos

Ante los diferentes formatos que adoptan las diferentes compañías en la actualidad, se ha recopilado una serie de estándares oficiales o de facto, por dos razones:

- Investigar las diferentes posibilidades de modificación que nos dan estos formatos, con el fin de ver si sería posible incluir algún tipo de marca en ellos para su posterior sincronización.
- Ver si realmente se utilizan en los ámbitos donde pretendemos abarcar nuestro proyecto.

A pesar de habernos centrado en las posibilidades de estos formatos, se pretende que al ser un lenguaje independiente del formato del fichero, pueda ser incrustado o empleado en otros tipos de formatos.

Los formatos escogidos son los siguientes:

- Ficheros de texto plano
- PDF (ISO 32000-1:2008)
- LaTeX
- OpenDocument (ISO 26300)

Por otro lado, sería también interesante lograr la sincronización entre ficheros que no sean de texto, sino que sean de contenido `Multimedia`:

- Video
- Audio

2.1. Ficheros de texto plano

Consideramos ficheros de texto plano a los típicos ficheros txt que genera el bloc de notas de `Windows` o a cualquier fichero que podemos guardar con un editor de textos normal como puede ser el `Vi`.

Aunque no es común que las personas traten con este tipo de ficheros, si es cierto que en algunos casos donde la información tratada no entre directamente en alguno de los formatos estudiados, podría servirnos de ayuda.

Por ejemplo, dado el momento podríamos considerar como un fichero de texto plano, ficheros con código fuente, lo cual nos podría permitir relacionar secciones de un código fuente con otras secciones de otros códigos o con partes de un manual.

Debido a que realmente este tipo de ficheros existen desde los albores de la informática y engloban las características más fundamentales de los que describiremos a continuación, no les dedicaremos ni un apartado de historia, ni de ventaja ni del ambiente en el que se suelen mover.

2.2. PDF (ISO 32000-1:2008)

2.2.1. Historia

En la actualidad es uno de los formatos de fichero más empleado, tanto en un ambiente docente, como en ambientes de trabajo y de hecho es la opción favorita por muchas personas, dada su facilidad para la lectura en los diferentes lectores que existen actualmente en la red.

En 1991, John Warnock cofundador de la compañía Adobe Systems Inc. describe una serie de ideas y de tecnologías en un documento llamado "Camelot" [22], que supondrían el germen del ahora tan conocido *Portable Document Format*.

Describe que el problema más grande que había, es que no había una manera universal de comunicar y ver la información impresa electrónicamente. Ese tipo de tareas se realizaba mediante Fax, pero lo considera un medio altamente ineficiente y bajo en calidad.

De esta manera, considera que el recién inventado *PostScript* es una gran solución a nuestro problema, dado que es capaz de ser independiente a todas las plataformas tanto software como hardware.

Este documento en el que se plantea una serie de ideas acerca de como utilizar *PostScript* para poder conseguir lo planteado, desemboca finalmente en la creación de *PDF 1.0* en el año 1993.

Además dado el gran uso que tuvo, finalmente se convirtió en estándar `ISO` en el 2008, en su versión PDF 1.7 y actualmente se está trabajando en una ampliación del estándar.

En la actualidad este tipo de formato está teniendo un gran auge, dado que pese a las diferentes vulnerabilidades de seguridad que ha demostrado a lo largo de los años, podemos decir sin lugar a dudas que es probablemente el archivo más estándar a la hora de intercambiar documentos entre empresas, estudiantes, etc. Esto ha sido debido principalmente a sus características multiplataforma y a la posibilidad de leerlo en todo tipo de dispositivos, actualmente es soportado tanto en ordenadores, como en dispositivos móviles, así como los tan de moda en nuestros días: tablets y ebooks.

2.2.2. Ámbitos de uso

Siempre ha sido el formato por excelencia para intercambiar documentos, especialmente si estos no se deseaba que fueran modificables. De hecho en la actualidad se permite hasta firmar el documento para asegurarnos de que no ha sido modificado por nadie en el envío.

De acuerdo con lo descrito a lo anterior podemos establecer los siguientes ámbitos de uso:

- Ámbitos docentes y divulgativos.
- Ámbito empresarial.
- Entretenimiento.

2.2.3. Descripción interna del formato

Internamente como todos los ficheros el PDF es una secuencia de bytes que como nos indica el estándar podemos agrupar en `Tokens`. Estos tokens agrupados así mismo dan lugar a los objetos que será el nivel en el que vamos a trabajar.

Como lenguaje posee los siguientes objetos: Boolean values, Integer, Real , Strings, Names, Arrays, Dictionaries, Streams, y el objeto null. Además como en la mayoría de formatos nos

permite el uso de comentarios, por ejemplo:

```
abc% comment ( /%) bla bla bla
123
```

Este código equivaldría a los tokens abc y 123. Se hace hincapié en esta posibilidad debido a que seguramente va a ser la opción más estándar que vamos a poseer en la mayoría de los formatos y que nos va a permitir introducir la cantidad de información.

Así pues la primera manera que tendríamos de introducir datos dentro de nuestros ficheros en el caso de que finalmente nos dediquemos por no mantener ficheros externos, podria ser esta.

Otro objeto que nos puede resultar de suma utilidad son los Diccionarios, en ellos podemos definir una serie de claves/valor que suele usarse internamente para la gestión de objetos complejos dentro del documento.

Una última y curiosa manera es la que hemos encontrado a partir de una persona que buscaba cifrar información y ocultarla en un PDF: [2].

Esta técnica se basa en ocultar los stream bajo varios filtros de tal manera que quedan ocultos a la vista del documento. Esta podría ser otra opción de cara a modificar nuestro documento.

2.3. LaTeX

2.3.1. Historia

LaTeXes un lenguaje de marcado que surgió por la creación de diferentes macros que facilitaban el uso de TeX, lenguaje creado por Leslie Lamport para la creación de diferentes documentos.

El origen de TeXse data en 1984, aunque debido a su código abierto, multitud de personas fueron realizando sus aportaciones, de tal manera que acabaron coexistiendo diferentes dialectos de LaTeXincompatibles entre si. Es por esto que Leslie Lamport decidió en 1989 junto con otro grupo de personas, lanzar el proyecto LaTeXcon el fin de unificar los diferentes dialectos y macros aparecidas, en un proyecto conjunto.

En la actualidad aunque no es usado a gran escala es de vital importancia para aquellas personas relacionadas generalmente a ámbitos de investigación y dentro de éstos, a los de

ámbito científicos, debido a la posibilidad de desarrollar grandes documentos sin los problemas habituales de formato, además de su gran versatilidad para la introducción de fórmulas matemáticas complejas y otro tipo de contenido debido a la gran cantidad de `plugins` que existen en la actualidad.

2.3.2. Ámbitos de uso

La mayoría de las personas que utilizan LaTeX, son personas relacionadas con el mundo de la docencia (en especial relacionados con las ciencias) y del ámbito científico. La razón es que LaTeXsiempre ha gozado de una facilidad y rapidez para la introducción de fórmulas en documentos, que no tienen otros procesadores de textos `WYSIWYG`.

Por otro lado es empleado por muchos escritores independientes, puesto que les permite realizar una edición de calidad en un tiempo breve, sin necesidad de estar bajo el amparo de una editorial.

De acuerdo con lo anterior podemos establecer los siguientes ámbitos de uso:

- Ámbitos docentes y divulgativos.
- Ámbito empresarial, especial en la maquetación y edición de libros.

2.3.3. Descripción interna del formato

LaTeXfue diseñado para que su expansión y adición de nuevos elementos fuera muy sencillo. Dispondríamos de dos maneras diferentes para modificar el formato:

La primera como en todos los lenguajes sería mediante la inclusión de comentarios. En latex al igual que en el formato PDF, los comentarios se introducen entre símbolos de porcentaje:

```
%Esto es un comentario en LaTeX%
```

Otra manera que tenemos de introducir elementos en latex es crear introducir referencias dentro del documento: [23]. Esto es debido a que latex no provoca errores de compilación en los documentos si no encuentra referencias, de modo que solo mostrará una advertencia al compilar un documento de que no se han encontrado referencias.

Otra manera sería extender el funcionamiento de latex, esto se puede realizar mediante comandos e incluso podríamos jugar a crearnos un estilo invisible donde podamos escribir

cierta información de tal manera que lo conservemos en el fichero pero no en el documento generado.

2.4. OpenDocument (ISO 26300)

Surgió debido a la necesidad de contar con un formato que hiciera frente al formato de documentos que empleaba Microsoft, de esta manera se independizaría de un fabricante la elaboración del estándar, lo cual aseguraría que fuera abierto y sobretodo que un organismo independiente lo revisara y ampliara.

Los primeros trabajos comenzaron por parte de Sun Microsystems en el año 1999. Luego ya con la fundación de OpenOffice, se procedió a ahondar el trabajo en este formato puesto que sería soportado por la nueva suite ofimática.

Siempre ha sido el estándar del software libre desde su aparición y ha sido adoptado por la mayoría de suites de ofimática libres. En la actualidad es usado sobretodo dentro de esta comunidad y de la cantidad de gente que cree en este tipo de paradigma de software.

2.4.1. Ámbitos de uso

Al ser un formato ampliamente extendido en la actualidad debido a que es soportado por la mayoría de procesadores de texto libres, además de la posibilidad de importarlo en las últimas versiones de Microsoft Office, es empleado por personas de todos los ámbitos.

A pesar de esto cuenta con una fuerte oposición por parte del formato estrella de Microsoft, el docx que vino a sustituir al formato doc que había logrado estandarizarse con el paso del tiempo. A pesar de ello, debido a la gran influencia que está teniendo en las empresas y en las personas los entornos de software libre y abierto, como las suites de OpenOffice y LibreOffice y también a la ausencia de formatos específicos en aplicaciones en la nube como las que ofrece Google Docs este uso masivo está empezando a disminuir lo que en el futuro esperemos depare un formato más abierto y mejor documentado que el de Microsoft.

De acuerdo con lo anterior podemos establecer los siguientes ámbitos de uso:

- Ámbitos docentes y divulgativos.
- Ámbito empresarial.
- Entretenimiento.

2.4.2. Descripción interna del formato

Debido a que esta basado en XML como siempre tenemos la opción de insertar comentarios.

Además este formato en su estándar define los objetos Bookmarks y references, donde podemos insertarlo de manera que hagan de marcas para nuestro propósitos.

Si no queremos que nada de esta información quede visible de cara al usuario, debido a que es un XML siempre podremos crear nuestro propio espacio de nombres dentro del documento insertando marcas o lo que deseemos sin que afecte a la lectura del contenido.

2.5. Otros formatos

Otros formatos que podrían interesarnos son aquellos relacionados con los medios multimedia, como pueden ser el Audio y el Video.

Debido a la gran cantidad de formatos existentes en la actualidad y debido a que su modificación es bastante dificultosa puesto que la mayoría siguen sus propias cabeceras, y contenidos, en este caso no nos quedaría más remedio que proponer como solución para la sincronización el sacar fuera del propio medio la información que debemos incluir.

Un ejemplo de uso extendido de introducir etiquetas dentro de ficheros de video es la propuesta por YouTube. Como podemos observar en el siguiente artículo sobre cómo obtener las etiquetas de un video: [19] la información que emplea YouTube de manera interna para gestionar estas etiquetas es el XML, formato que se entiende estándar y totalmente portable.

En la actualidad este tipo de etiquetas está teniendo un gran éxito e incluso se han llegado a hacer tutoriales o juegos montados en este tipo de plataformas, por lo que para nuestra audiencia objetivo, sería de una gran utilidad si fuera posible mantener una colección de etiquetas por ejemplo para varias partes de un video.

Debido a que el único ejemplo algo bien referenciado encontrado ha sido el de YouTube, que está basado en XML, podríamos aportar las diferentes ideas que nos han surgido en el estudio del formato de OpenDocument como son los comentarios, o la creación de nuevos espacios de nombres para nuestras propias etiquetas.

Capítulo 3

Métodos de sincronización de contenidos

Una parte importante a tener en cuenta antes de plantear los diferentes modelos y ver hasta que punto nos son útiles, es estudiar las diferentes opciones que tenemos a la hora de mantener una sincronización entre los diferentes ficheros, ésto nos va a permitir a posteriori saber que solución es la más conveniente para nuestro caso y sobretodo nos va a permitir evaluar con una mayor eficiencia los modelos que propondremos en el capítulo siguiente.

Una de las primeras decisiones que debemos tomar se refiere en cuanto a nuestra unidad básica de información. En informática es típico desde casi su principio trabajar con bits pero en nuestro caso a nivel de sincronización no nos aportaría nada, es por ello, que analizando los ámbitos en los que nos movemos, podríamos ver como interesantes los siguientes:

- Palabra: Es obvio que para nuestro caso la palabra sería una unidad de sincronización demasiado pequeña. El usuario que finalmente quisiera llevar a cabo una relación le resultaría muy costoso ir relacionando unas palabras con otras de otro fichero, si bien si consideramos que las mismas palabras en los ficheros deberían ir relacionadas por defecto, entonces podría sernos de utilidad. Por otro lado, tenemos el problema de que suele haber muchas palabras en un documento, si acabamos relacionando una gran cantidad de ellas con otras sin emplear criterios diferentes a simples relaciones, acabaríamos teniendo una cantidad de información de relaciones descomunal lo cual nos haría un flaco favor en cuanto a eficiencia de uso y de espacio, así como de portabilidad.
- Frase: Aunque puede ser más usable que una sola palabra, pierde la posibilidad que teníamos en la palabra de poder relacionar las iguales entre ambos ficheros de manera automática, esto es debido a que la probabilidad de que en dos ficheros se den frases

exactamente iguales o al menos muy parecidas es bastante improbable. Por otro lado si que tiene el beneficio de que podría ser más útil para el usuario debido a que podría establecer que frases del fichero hacen relación a frases del otro.

- Párrafo: Cuanto más grande va siendo la cantidad de información que relacionamos podemos ver que al usuario y a nosotros nos va a resultar más beneficioso. Tiene los mismos beneficios que la frase, pero sobretodo la mejoría respecto a esta es que la cantidad de párrafos de un documento es menor y por tanto hace que esta información de sincronización sea mucho más manejable.
- Capítulo: Para el tipo de sincronicidad que buscamos es una relación demasiado grande, dado que en general los capítulos suelen abarcar mucha información diferente y además porque hay muchos ficheros que no disponen de capítulos, si no que a lo mejor son solo una cantidad pequeña de párrafos. Por otro lado hay formatos donde nos seria bastante dificultoso establecer qué es un capítulo y qué no, puesto que en un fichero de texto plano por ejemplo tendríamos que establecer una política de que un capítulo es algo que empieza separado por espacios con un número y en mayúsculas o similares.

A la vista de estas posibilidades, parece bastante intuitivo ver el beneficio que tendría tomar como unidad el párrafo, pero debemos darnos cuenta de una cosa, aunque en principio la mayor parte de los documentos con los que vamos a tratar son de texto, también hemos pensado que seria una buena idea relacionar documentos de otros medios como pueden ser vídeos, aquí es donde los párrafos dejan de servirnos puesto que un vídeo no tiene párrafos.

Debido a esta peculiaridad que deseamos, tenemos dos opciones:

- Diseñamos nuestro lenguaje con un mecanismo de extensión para documentos de vídeo o de otros medios continuos como puede ser audio, lo cual por un lado nos permitiría jugar con una mayor variabilidad en aquellos ficheros que son puramente de texto y obtendríamos las ventajas de que nuestra extensión para medios continuos podría estar diseñada de una manera más eficiente.
- Diseñamos nuestro lenguaje de una manera que no hagamos distinción entre ficheros, de esta manera aunque perderíamos las posibilidades de llegar más a fondo en cuestiones relacionados con algunos tipos de ficheros más especiales que podamos encontrarnos en el futuro cuando queramos ampliar nuestro lenguaje, nos permite que de manera automática cualquier tipo de fichero sea el que sea, siempre y cuando cumpla una serie de criterios que habrá que especificar a la hora de desarrollar el lenguaje, sería válido para empezar a etiquetarse de acuerdo a lo desarrollado.

Como aún es pronto para tomar una decisión a este respecto, de momento no vamos a tomar partido por ninguna de estas dos vertientes, dejando esta decisión a la hora de observar los diferentes modelos de sincronización que veremos en el capítulo siguiente.

Capítulo 4

Propuestas de modelos

Una vez hemos analizado los diferentes formatos que hemos escogido, el siguiente paso va a ser explorar las diferentes posibilidades que podemos emplear a la hora de sincronizar las partes de nuestros documentos con otras. Esta tarea es básica puesto que sin una buena sincronización propiciaremos con el paso del tiempo pérdidas de información y una dificultad enorme a la hora de mantener nuestra colección de "relaciones" entre partes de los documentos.

Cuando pensamos en un primer momento en el problema que nos ocupa, surge ante nosotros la posibilidad quizás más típica y que por ello abordaremos a continuación, que sería llamada la sincronización byte-byte.

4.1. Sincronización byte-byte

La idea de esta primera propuesta se basa en mantener dos archivos diferentes sincronizados byte a byte a través de una tabla de tal manera, que cuando nosotros por ejemplo queramos saber con que está relacionada un byte del primer fichero, solo tengamos que acudir a la tabla para ver con que byte del segundo fichero está relacionada.

A partir de este modelo base debemos hacernos las siguientes preguntas, ¿Qué ocurre cuando varios ficheros necesitan ser sincronizados, es posible incrustar esta información en cada fichero?

Pues la respuesta a estas preguntas es que es un modelo demasiado arcaico y sobretodo muy poco eficiente debido a los siguientes puntos:

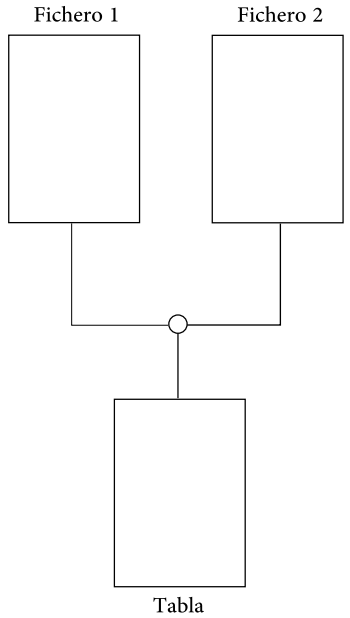

Figura 4.1: Sincronización byte-byte

- Ventajas:
  - Aunque las ventajas son más bien escasas, de una manera muy rápida recorriendo la tabla sería posible borrar todos aquellos bytes que hayamos marcado que no son de nuestro interés.
- Desventajas:
  - Si vamos a tener sincronización entre diversos ficheros, vamos a acabar necesitando o bien una tabla maestra gigante donde pongamos una por una la relación de cada sección del fichero con cada sección del resto de ficheros, o vamos a tener que dividir esta tabla en muchas subtablas donde cada tabla sea la relación de un fichero con otro fichero. El problema de esto, es que vamos a tener una cantidad de información enorme muy difícil de manipular y actualizar.
  - Dados los comentarios anteriores sería prácticamente imposible incrustar esta información dentro del propio fichero, debido a que la tabla seria enorme lo cual incrementaría muchísimo el tamaño de los ficheros.
  - Introducir otro fichero en nuestra colección que tenga relaciones con estos ficheros es un caos, tendríamos que ir actualizando la tabla de cada fichero por separado o la tabla maestra dependiendo nuestra selección lo cual lo hace prácticamente impensable.
  - ¿Qué ocurre con aquellas zonas que no tienen nada que ver con ninguno de los puntos del otro fichero? ¿Debemos no incluirlas en la tabla? Si fuéramos estrictos

en nuestro modelo sí que deberíamos hacerlo dado que suponemos que cada parte del fichero debe estar relacionada, o quizás podríamos plantearnos la solución de montar un espacio "null" en la tabla donde apunten aquellos rangos del fichero que no tienen ninguna relación.

- El byte es una unidad demasiado pequeña a la hora de sincronizar y pocas veces va a tener sentido en nuestro contesto, es por esto es que es una locura tratar de sincronizarlos de esta manera.
- La información que hay hoy, si el día de mañana ampliamos los ficheros o los modificamos va a haber que recolocarla lo cual puede ser muy costoso.

- Facilidad de implementación
  - Es otra de las principales ventajas de este sistema, aunque es poco usable, de cara al desarrollo es muy rápido y sencillo de desarrollar. Salvo en el caso de que controlemos de una manera automática la recolocación de estas tablas cuando insertamos o modificamos partes del documento, lo cual puede hacer que se complique mucho más puesto que habría que tratar de hallar una manera de actualizar las tablas de una manera eficiente de acuerdo a los cambios que ha realizado el usuario e insertando las nuevas relaciones.

Como podemos observar este tipo de modelo nos va a plantear muchos problemas, aunque por otra parte a la hora de funcionar podría ser realmente eficiente, dado que si la tabla no es demasiado grande, acceder a las relaciones sería muy muy rápido y podría facilitarnos también mucho las cosas a la hora de eliminar partes de documentos que no deseamos simplemente fijándonos en esos rangos de bytes.

Una evolución a partir del modelo anterior, podría ser en lugar de basarnos en rangos de bytes, fijarnos solamente en la posición de diferentes partes del fichero, este tipo de sincronización será la que vamos a analizar en el siguiente punto y la llamaremos: Sincronización basada en posiciones simbólicas.

4.2. Sincronización basada en posiciones simbólicas

Es una evolución directa del planteamiento anterior, aquí la idea es relacionar una posición del fichero con otra posición del segundo fichero. Así nos ahorramos tener que sincronizar demasiadas partes y además vamos a obtener numerosos beneficios.

A simple vista podemos observar que realizando algo tan sencillo como relacionar posiciones dentro de un fichero, hemos solucionado muchos de los problemas que teníamos en el anterior modelo, como son que las tablas se van a ver reducidas de una manera drástica de tamaño. De cara al usuario ya es más sencillo tratar esta cantidad de información dado que solamente tiene que marcar una zona del fichero y relacionarla con otra, no como antes

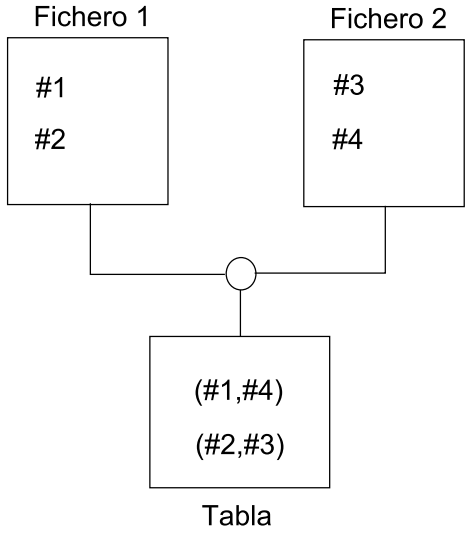

Figura 4.2: Sincronización basada en posiciones simbólicas

que teníamos que ir byte por byte relacionando y sobretodo que si el uso de ficheros de la colección no es muy grande, podemos asumir la idea de incrustar las tablas en los propios ficheros.

A continuación enumeramos las diferentes ventajas y desventajas de este modelo de sincronización:

- Ventajas:
  - La ventaja es que el tamaño de las tablas es mucho menor que en el caso anterior lo cual nos permite incrustar esta información en el propio fichero y además no nos va a suponer un problema de espacio.
  - Además para el usuario supone la ventaja de que puede escoger una zona del fichero y automáticamente mediante software podríamos saber a que dirección en bytes se refiere para conectarla con otra zona.
  - Si hemos establecido correctamente el nivel de sincronicidad (palabra, párrafo,etc) podría ser posible eliminar fácilmente información de todos los ficheros de una parte que el usuario ya no desee conservar.
- Desventajas:
  - La idea de portabilidad aquí también se ve truncada, dado que tenemos la información en tablas que se refieren a otro fichero, no podemos relacionar el fichero con otros sin necesidad de reescribir toda la tabla, lo que lo puede hacer costoso.

- Además nuevamente tenemos el problema de qué ocurre cuando los ficheros crecen o se modifican, hay que estar actualizando constantemente estas tablas y dependemos siempre de tener la versión del fichero correcta puesto que si no la información de las tablas será invalida.

- Facilidad de implementación

  - Es otra de las principales ventajas de este sistema, aunque es poco usable, de cara al desarrollo es muy rápido y sencillo de desarrollar. Salvo en el caso de que controlemos de una manera automática la recolocación de estas tablas cuando insertamos o modificamos partes del documento, lo cual puede hacer que se complique mucho más puesto que habría que tratar de hallar una manera de actualizar las tablas de una manera eficiente de acuerdo a los cambios que ha realizado el usuario e insertando las nuevas relaciones.

4.3. Sincronización mediante marcas

La idea de este modelo de sincronización es ir un poco más allá que en el paso anterior. En lugar de marcar posiciones simbólicas del fichero y relacionar esas posiciones entre ellas, la idea es fijar una marca en esa posición con un nombre, de manera que las marcas que posean el mismo nombre estuvieran conectadas.

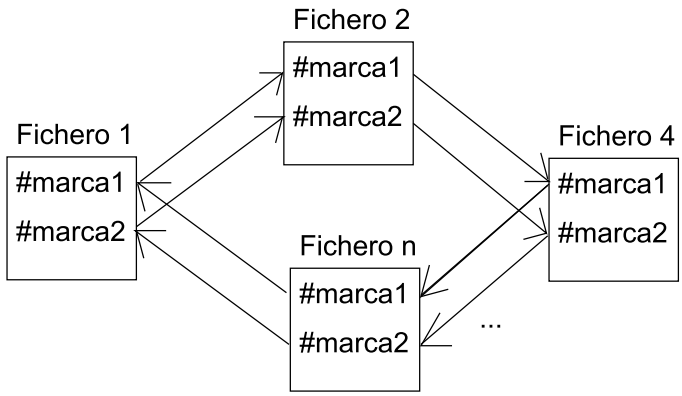

Figura 4.3: Sincronización basada en marcas

Este modelo es el primero de los anteriormente expuestos en el que logramos mantener una independencia entre ficheros de tal manera que en cualquier instante, podemos añadir un nuevo fichero a nuestra colección y mientras le añadamos las marcas que creamos convenientes, desde ese momento pasa a estar relacionado con el resto.

Además incrustar una marca en un fichero es tan solo añadir unos pocos bytes lo cual lo

hace muy atractivo de cara a que el tamaño no sube y que en este caso no seria necesaria ninguna tabla.

Así pues enumeraremos las diferentes ventajas y desventajas como hemos hecho con el resto de modelos:

- Ventajas:
  - Hemos solucionado los problemas de espacio.
  - Ahora nuestros ficheros son intercambiables con los de otras personas siempre que pactemos las marcas que vamos a usar en ellos.
  - No vamos a tener una necesidad de tablas externas lo cual aunque por un lado es una ventaja debido a que no necesitamos ficheros extras ni espacio extra, puede suponer una desventaja como analizaremos más adelante.
  - Las relaciones entre todos los documentos se vuelven transitivas de tal manera que no tenemos que almacenar las relaciones uno a uno.
  - Cuando cambiamos o modificamos el fichero, solo es necesario cambiar las marcas o renombrarlas, ya no es necesario volver a recorrer y calcular todas las tablas como en los modelos anteriores.
- Desventajas:
  - La principal desventaja se refiere en cuanto a la variabilidad de las marcas. Aunque tenemos la posibilidad en un principio puesto que no hemos puesto ninguna restricción, a que una parte del fichero lleve varias marcas, es cierto que en relaciones complicadas, a veces puede resultar útil tener una tabla de relación entre marcas, de tal manera de que yo sea capaz por ejemplo de relacionar una marca con otra sin necesidad de ser la misma.
- Facilidad de implementación
  - En principio la implementación de este modelo de cara al desarrollo no deberia ser dificultosa, puesto que únicamente tendríamos que encontrar las posiciones marcadas en un fichero y las marcadas en el otro, lo cual si hicieramos una primera pasada para tenerlo todo en memoria, debería ser muy eficiente aunque dependiendo del tamaño de los ficheros como siempre. Por otro lado si el rendimiento se viera muy reducido por el tamaño, siempre podríamos sacar las marcas fuera del fichero, indicando la posición que ocupan dentro del fichero, lo cual mejoraría el rendimiento puesto que solo habría que acudir a la tabla a consultar y luego a esa dirección dentro del fichero.

Como hemos visto, uno de los principales problemas que le vemos a este modelo, es precisamente que aunque resulta muy sencillo, simple e intuitivo, por otro lado nos iría bien un mecanismo con el que seamos capaces de transformar unas relaciones en otras.

Por ejemplo:

Imaginemos que nos movemos dentro de una colección de documentos referidos a la historia de la música, con lo cual seria muy sencillo que nos aparecieran etiquetas como: Bach, Mozart, Beethoven, Chopin, Liszt, The Beatles...

Por otro lado podríamos tener etiquetas referidas a los diferentes estilos, como pueden ser: Música Clásica, Pop, Jazz, Tangos, etc..

Imaginemos por un momento que un historiador que esta manejando esta colección de ficheros, necesita conectar los diferentes compositores habidos con guerras o épocas de guerras cercanas dado que quiere estudiar como influyeron los conflictos bélicos en la música de los compositores.

Si se da la situación anterior, el historiador tendría dos opciones:

- Reescribir toda la colección de ficheros anterior, insertando las etiquetas de las guerras o los periodos históricos que aparecieran en su colección, con los cuales quisiera conectar.
- A la inversa, reescribiendo su colección de ficheros añadiendo las etiquetas de los compositores o estilos a las etiquetas que existen en su colección de ficheros sobre historia.

Ambas ideas podemos ver rápidamente que tienen un inconveniente y es que acaba siendo imposible mantener la colección de etiquetas relacionadas en grandes colecciones de ficheros. Con unos pocos ficheros y pocas etiquetas es sencillo que la persona pueda reeditar cada fichero añadiendo las etiquetas que correspondan, pero sin embargo una vez que tenemos una gran cantidad de datos, esto es imposible. Y de hecho esto entorpecería mucho la labor a la hora de aprovechar el trabajo que ya se ha realizado previamente en otras colecciones que no son nuestras, lo que le resta usabilidad y portabilidad.

Es por ello, que vamos a plantear una evolución de este modelo, intentando mejorarlo con tablas, el cual llamaremos Modelo de sincronización de marcas basado en tablas de relación.

4.3.1. Modelo de sincronización de marcas basado en tablas de relación

La idea de este modelo es aumentar las posibilidades que tenemos del modelo anterior, que aunque es muy flexible y eficiente, hemos visto que tiene una carencia en cuanto a relacionar diferentes partes de colecciones que en un principio no tienen un mismo conjunto de etiquetas.

La manera de solventar esto, va a ser añadir al modelo anterior una tabla de equivalencias, donde nosotros podamos especificar que marcas se relacionan con otras. Debido a que si

el número de relaciones fueran muy grandes podemos correr el riesgo de que todo acabe relacionado con todo, nuestras tablas van a ser intercambiables, es decir, tendremos una tabla por ejemplo que llamaremos Periodos musicales y que emplearemos cuando queramos relacionar nuestra colección mediante ese criterio. Tendremos otra tabla, llamada Guerras Bélicas, que relacionará nuestra colección mediante ese criterio, de tal manera que en el futuro nosotros podamos ser capaces de ir añadiendo filtros o quitándolos, como si estuviéramos jugando con la lente de una cámara fotográfica.

De esta manera si observamos la imagen, podremos comprender rápido el funcionamiento de este modelo:

Por un lado partimos de la colección 1, que posee una serie de ficheros desde 1 hasta n. Estos ficheros por un lado tienen relacionados a través de marcas fragmentos del fichero con otros. (Por ejemplo, la marca 1 o la marca 2).

Por otro lado la colección 2, tiene también una serie de ficheros de 1 hasta n, que a su vez de nuevo tienen una serie de marcas en común, que serían la marca 3 y la marca 4.

Si llegado un momento tenemos la necesidad de relacionar la colección 1, podremos hacerlo a través de una de las tablas que hemos planteado en la imagen, como serían la relación A y la relación B. Si por ejemplo empleamos la relación A, a partir de ese momento además de las relaciones individuales que se tenían en ese modelo, aquellos fragmentos con la marca1 estarán relacionados con la marca3. Si por el contrario decidimos emplear la relación B, estarán los de la marca1 con los de la marca4. Y si deseamos aplicar los dos al mismo tiempo, tendremos relacionados los de la marca1 con la marca3 y la marca4.

Esto nos aporta unas posibilidades infinitas para que no sea necesario emplear un lenguaje de marcas común entre todas las personas que requieran hacer uso de una colección de ficheros, cada persona puede crearse sus tablas de criterios para empastar sus diferentes colecciones de ficheros con temáticas diferentes o donde simplemente no había usado la mismas marcas.

A continuación, como hemos hecho en el resto de modelo, vamos a evaluar las diferentes, ventajas y desventajas del modelo:

- Ventajas:
    - La primordial es que podemos establecer relaciones entre colecciones en un principio diseñadas con otras marcas y para otro fin, de manera que es mucho más flexible su uso.
    - Seguimos teniendo pocos problemas de espacio, puesto que la idea debido a que las tablas no tienen sentido dentro del propio fichero, es que las marcas queden anotadas como antes en el fichero y nuestras tablas queden en ficheros aparte de tal manera que podamos usarlas cuando nos convenga. Además estas tablas por

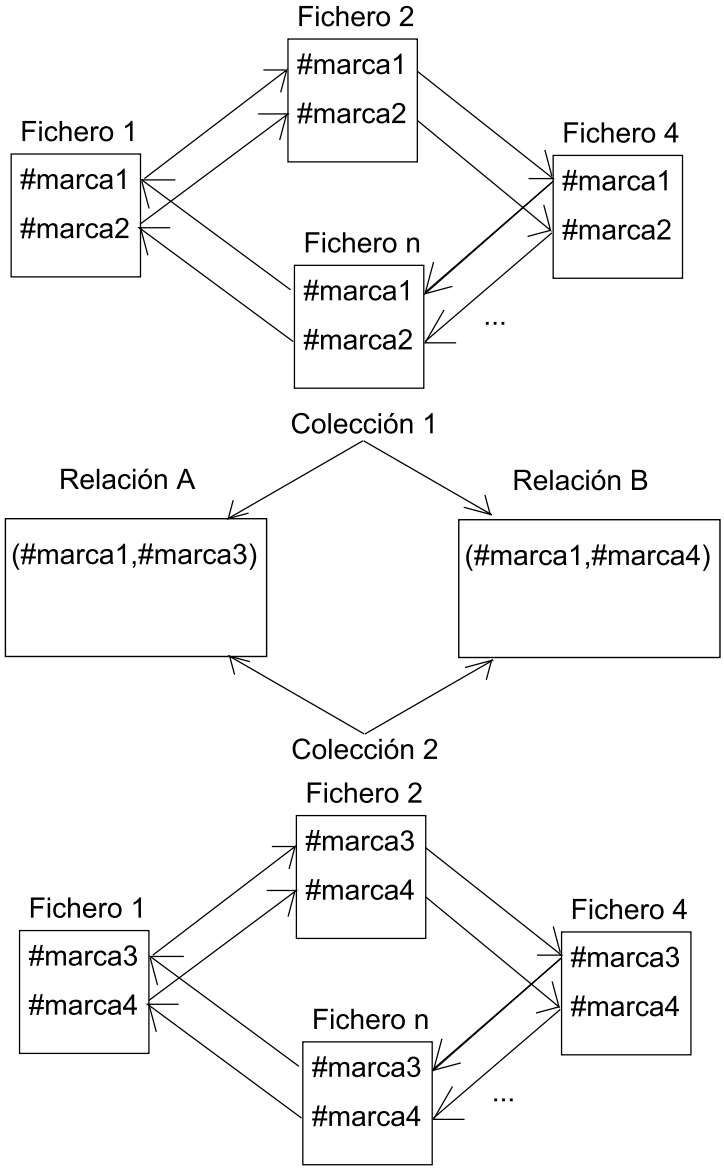

Figura 4.4: Sincronización de marcas basado en tablas

lo general debido a que poseen una información pequeña, no deberían crecer a tamaños desmesurados con el tiempo.

- Se mantiene la transitividad a lo largo de todas las tablas de relación, lo cual es un mecanismo muy potente cuando tengamos muchas colecciones diferentes de datos

y queramos movernos entre ellas.

- Desventajas:

  - La principal desventaja es que en algunos momentos puede resultar molesto para el usuario poseer muchas tablas para establecer relaciones, en lugar de tenerlo todo en el propio fichero, pero a la hora de diseñar una aplicación que se ocupe de esto, no debería ser dificultoso encontrar soluciones eficientes e interesantes.

- Facilidad de implementación

  - De nuevo la implementación del modelo tiene los mismos pros y contras que el solo basado en marcas, con el añadido de que ahora tendremos que ir investigando las tablas que nos muestre el usuario para establecer relaciones más allá de lo que nos dice el propio fichero. Podría suponer problemas de tiempo a la hora de hacerlo en grandes colecciones, pero en principio no debería suponer un problema muy grande.

Este modelo parece suficiente para todos nuestros propósitos, pero plantearemos otro, que aunque en estos momentos creemos que no sería posible, o al menos sería muy dificultoso de hacer de cara a la implementación y al desarrollador, sería interesante para un futuro, lo llamaremos Modelo de sincronización de marcas relacionadas semánticamente.

4.4. Modelo de sincronización de marcas relacionadas semánticamente

La idea de este modelo es dar un paso más allá en todos los modelos introducidos anteriormente. En los modelos anteriores, siempre teníamos que introducir nosotros una serie de marcas, o unas tablas para que se recogieran todas las relaciones que queríamos obtener. Este modelo lo que pretende, es que dado que todos los ficheros que estamos tratando, con excepción de los medios continuos, son legibles en texto plano si se abren con cualquier editor y además están bien definidas por cada lenguaje aquellas partes que pueden tener texto, que seamos capaces de relacionar el contenido del texto por su significado, con otros contenidos del texto.

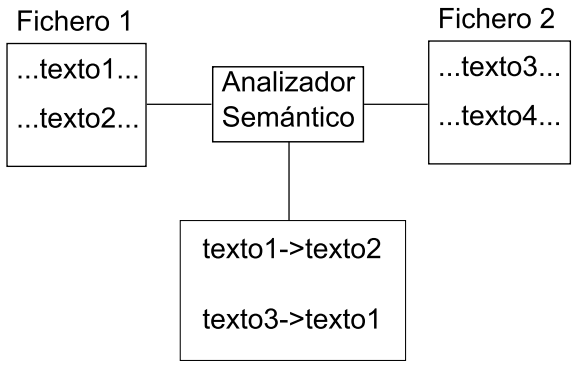

Figura 4.5: Sincronización de marcas relacionadas semánticamente

Esta idea nos va a recordar un poco al procesado que llevamos a cabo cuando compilamos un código fuente, en el que este código va pasando a través de una serie de analizadores como si fuera una cadena de montaje, para de esta manera obtener el código ejecutable final, como podemos observar en la imagen.

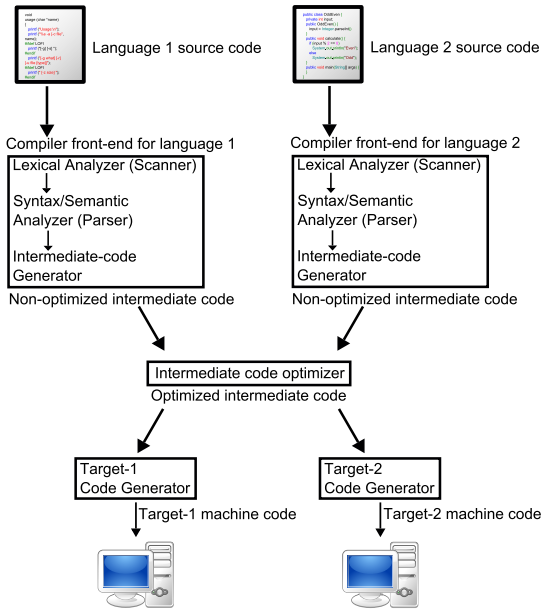

Figura 4.6: Fases de la compilación

A través de una analizador semántico, nosotros podríamos establecer una serie de marcas durante el tiempo de procesado, para poder establecer relaciones entre ellas. El problema

de este modelo va a ser que es muy costoso, no solo como comentábamos por el coste de implementación, si no porque cada vez que añadamos un fichero nuevo a la colección, u otra colección, va a haber que revisar todos los ficheros para intentar establecer relaciones entre ellos.

Es por esto que para este proyecto no nos planteamos en ningún momento escoger este modelo, debido a las dificultades tecnológicas que conlleva si en el futuro desea implementarse, si bien, si se llevara a cabo, podríamos plantearlo con una serie de etapas parecidas a las llevadas a cabo en la compilación de tal manera, que podríamos obtener algo parecido a la imagen.

Figura 4.7: Posible modelo de etapas para obtención de marcas

A continuación, como hemos hecho en el resto de modelo, vamos a evaluar las diferentes, ventajas y desventajas del modelo:

- Ventajas:
  - El usuario ya no tiene por qué preocuparse de establecer el marcas, si no que se establecen a partir del contenido del fichero, aunque seguramente el usuario pueda realizar sugerencias a los analizadores.
- Desventajas:
  - Hay que reabrir todos los ficheros para establecer nuevas relaciones cada vez que uno se añade.
  - En tiempo es poco eficiente, y además puede darnos lugar a una cantidad de marcas desmesuradas.
- Facilidad de implementación
  - Muy difícil de implementar sobretodo la fase semántica donde se sea capaz de extraer el significado de un texto para relacionarlo con otro. Aunque por otro lado este modelo de tuberías lo haría de una facilidad descomunal a la hora de establecer nuevas fases en el establecimiento de relaciones.

4.5. Conclusión: Elección del modelo

Una vez vistos los modelos anteriores, nos vamos a quedar con uno de esos modelos e intentaremos cerrar algunos temas pendientes que nos puedan aparecer.

Nuestro modelo seleccionado es el Modelo de sincronización de marcas basado en tablas de relación, puesto que a la vista de lo analizado anteriormente es el que más ventajas nos va a proveer.

Así pues, hemos decidido que para implementar este modelo lo ideal va a ser establecer los siguiente componentes:

- Cada fichero dispondrá de una serie de marcas que bien la persona o un software externo habrá introducido en él, de manera que esas marcas van a viajar siempre con ese fichero.

- Por cada tabla externa de relaciones que queramos definir, definiremos un fichero que relacionará unas marcas con otras en el caso de que el usuario lo necesite. Estos ficheros podrán ser aplicados de manera conjunta, como se vio en la definición de este modelo.

- Debido a que el modelo en principio no proponía ninguna solución para aquellos archivos dependientes de medios, debido a la dificultad que supone insertar contenido en forma de marca a un fichero de este tipo, hemos decidido que aquellos ficheros de medios que queramos tener dentro de nuestra colección, irán acompañados cada uno de un fichero donde se establezca mediante las posibilidades que nos ofrezca nuestro lenguaje, las zonas marcadas de ese documento. La gestión de este tipo de contenidos la veremos con más profundidad a lo largo de la siguiente parte.

Así pues, a continuación mostramos lo que va a ser un esquema base de la “arquitectura” de nuestro lenguaje:

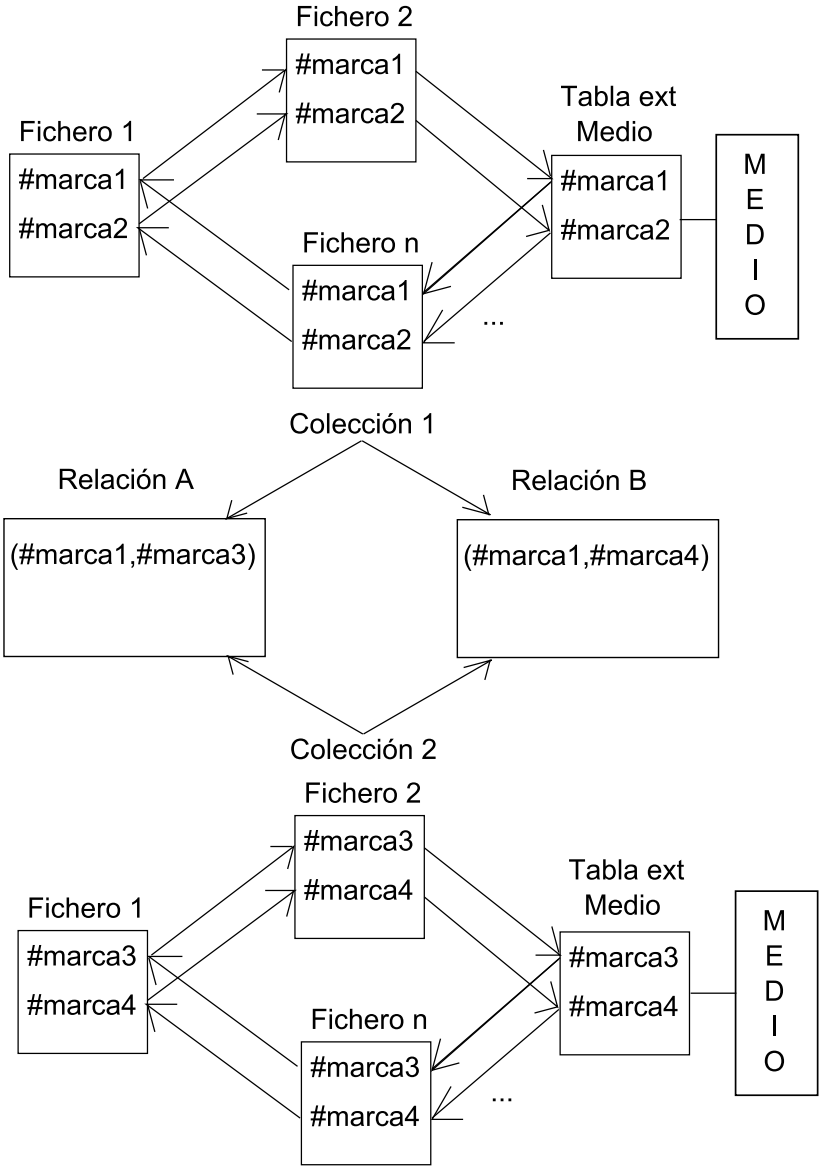

Figura 4.8: Modelo final

Parte II

Implementación del lenguaje

Capítulo 5

Especificaciones W3C para XML

Una vez hemos hecho un análisis de las diferentes posibilidades que teníamos a nuestro alcance, es hora de empezar a realizar una implementación de nuestro lenguaje. Para ello, nos basaremos en el popular lenguaje `XML`, siguiendo una serie de recomendaciones que hemos observado en la bibliografía.

Antes de eso, haremos un pequeño repaso por las especificaciones de XML para dar unas nociones aunque sean a nivel básico de lo que éste lenguaje de etiquetado nos puede aportar.

XML son las siglas de eXtensible Markup Language y surgió como heredero de `SGML`, con el objetivo de desarrollar un lenguaje neutro (anteriormente se había comenzado a desarrollar `HTML` pero se le dio una excesiva orientación hacia las páginas web) con el que poder generar otros lenguajes de etiquetas, pero que simplificara mucho SGML el cual tenía tantas reglas y sintaxis que dificultaba enormemente el trabajo con él, especialmente si nunca lo habías empleado.

Es por esto, que cuando se redactó la especificación por la `W3C`, se marcaron las siguientes metas:

1. XML shall be straightforwardly usable over the Internet.
2. XML shall support a wide variety of applications.
3. XML shall be compatible with SGML.
4. It shall be easy to write programs which process XML documents.
5. The number of optional features in XML is to be kept to the absolute minimum, ideally zero.

6. XML documents should be human-legible and reasonably clear.
7. The XML design should be prepared quickly.
8. The design of XML shall be formal and concise.
9. XML documents shall be easy to create.
10. Terseness in XML markup is of minimal importance.

Y así fue como en el año 1988 se saco la primera versión de XML 1.0, que en poco tiempo se vería muy usado de tal manera que posteriormente se decidió sacar la versión 1.1 de la que hablaremos más adelante, con el fin de evitar algunos problemas que se dieron con los juegos de caracteres.

Debido al hecho de que es un lenguaje de marcas con el que podemos derivar nuevos lenguajes, (por ejemplo tenemos una implementación de HTML que sigue reglas de XML como es XHTML), lo denominamos metalenguaje.

Su principal característica que lo hace muy manejable, es la idea de que se maneje en ficheros de texto normales, de tal manera que una persona corriente puede abrir ficheros XML, comprender su interior y modificarlos a su antojo, sin necesidad de programas complicados, solo con el uso de un editor.

Debido a estas ventajas, ha habido muchas tecnologías que han empezado a usarlo como formato de intercambio, por ejemplo hoy en día todos estamos familiarizados con la tecnología `AJAX` (Asynchronous Javascript And XML), que se basa en realizar peticiones a un servidor de manera asíncrona mediante HTTP, empleando XML como formato de intercambio.

5.1. Escogiendo entre XML 1.0 y XML 1.1

A continuación a modo de resumen plantearemos las pequeñas diferencias que existen entre la primera versión de XML (1.0) y la segunda revisión (1.1), que principalmente se refieren al uso de la codificación. (Al principio XML no había planteado el lenguaje de una manera que se pudieran emplear sistemas UTF8 y similares).

En el artículo de este ingeniero de IBM [8] se explica como la principal razón por la que se ha sacado la nueva versión, es debido a que cuando se implementó el primer XML sobre la base de `Unicode`, éste no había acabado de implementarse en su totalidad y conforme fueron avanzándose nuevas versiones, XML 1.0 acababa siendo totalmente incompatible con las nuevas versiones de Unicode.

Además se nos dice que se ha mejorado el soporte de tal manera que caracteres que antes no se soportaban dentro de ficheros XML, ahora tienen cabida, así como algunas pequeñas modificaciones en el uso de los espacios de nombres.

La cuestión a partir de estos cambios, es ver realmente en la actualidad, qué versión realmente se está empleando. Si miramos por las diferentes implementaciones de lenguajes que emplean XML como Ibatis, etc, veremos que la mayoría ha adoptado el 1.1 por el hecho de que realmente es bastante similar y da un mejor soporte a zonas donde no se emplean caracteres latinos.

Sin embargo es curioso observar, que en el momento en el que surgió XML 1.1, la mayoría de personas que estaba trabajando con XML 1.0 siguió trabajando con esta versión y de hecho se recomendaba en la época que si no ibas a usar juegos de caracteres extraños, siguieras usando la versión 1.0.

Así pues visto que queremos "internacionalizar" al máximo nuestro lenguaje, de tal manera que no importe de donde proceden los ficheros que vamos a relacionar, optaremos por una implementación XML 1.1 para nuestro lenguaje.

Capítulo 6

Definiendo nuestra jerarquía de etiquetas

Una de las primeras cosas que se nos indica en todos los libros didácticos sobre el uso de XML es cuál va a ser nuestra jerarquía de etiquetas.

A partir de esta jerarquía, iremos definiendo que necesitamos dentro de esa etiqueta como atributo, etc, y además iremos viendo casos especiales como son los medios continuos o medios multimedia.

6.1. ¿Etiqueta o atributo?

Si partimos del hecho de que una etiqueta puede tener numerosos atributos, podemos llegar a la conclusión correcta, de que perfectamente todos nuestros datos, podrían ser usados en una etiqueta.

Por ejemplo, imaginemos el libro: Don Quijote de la Mancha, escrito por Don Miguel de Cervantes Saavedra, escrito en el 1605.

Nosotros podríamos almacenar toda esta información dentro de una sola etiqueta con atributos:

```
<libro titulo="Don Quijote de la Mancha"
    año="1605" escritor="Miguel de Cervantes Saavedra"/>
```

Como vemos, esto dentro una sintaxis XML sería perfectamente válida. El problema que tiene es que estructuralmente es muy inestable. Es difícil, añadir nuevos datos a nuestra colección, como pueden ser datos relacionados con el escritor, si solo mantenemos en una etiqueta toda la información.

Por esto, quedaría mucho más natural, en lugar de abusar de atributos, emplear la siguiente sintaxis:

```
<libro titulo="Don Quijote de la Mancha" año="1605">
    <escritor nombre="Miguel"
        1apellido="de Cervantes" 2apellido="Saavedra" />
</libro>
```

Como vemos pues, es muy importante definir primero en qué vamos a centrar nuestro lenguaje para declarar esos elementos de importancia como etiquetas. Y luego a la hora de estructurar y añadir información secundaria, aplicar atributos o nuevas etiquetas que se puedan derivar.

Para ello vamos a necesitar establecer cual va a ser nuestra unidad base de trabajo para poder comenzar a definir elementos XML para nuestro lenguaje. La unidad básica de trabajo debido a que vamos a trabajar con relaciones entre diferentes tipos de documentos, podría ser relacionable, de tal manera que todas esas partes marcadas como elementos relacionables serían aquellas partes que el software posteriormente implementado, basado en una serie de atributos que se incluirán con el fondo de hacer estos datos más manejables, sería capaz de relacionarlas.

¿Qué clase de elementos/atributos necesitamos en nuestra relación?

Esta pregunta podemos enfocarla de dos maneras diferentes:

- Mediante una colección enorme de etiquetas, de tal manera que en cualquier temática tengamos ya un elemento definido para ajustarse perfectamente al contenido del texto. Por ejemplo dentro de la Química podríamos tener definiciones para: Elementos, Fórmulas, Propiedades, Leyes, etc... Para Biología tendríamos animales, especies, clases, géneros, familias... Este enfoque aunque por un lado al ser muy detallista haga que la información de la que disponemos en las relaciones es muy rica semánticamente, por otro lado le resta utilidad, debido a que vamos a tener una gran cantidad de etiquetas que no siempre va a estar dispuesto el usuario o incluso nuestro software a definir.
- Mediante una colección de elementos genérica, esta solución propone la definición de unos tipos básicos con los que consideremos que podemos abarcar un gran porcentaje de supuestos y en caso de necesidad, el usuario podrá definir otros más especializados en base a estos. Con esto por un lado quizás no vamos a tener una información tan rica

de todo el documento, pero por otro lado tenemos la ventaja que para el usuario al ser muchas menos etiquetas a manejar, va a ser más sencillo que las rellene por completo en el documento, bien por él mismo, o bien por un software externo.

Existen ya algunos modelos para relacionar información como el de los Tesauros. [29]

Los Tesauros se basan en organizar diferentes entidades relacionadas mediante la creación de unas listas de sinónimos (también suele contrastarse con otra de antónimos) y que tiene una gran utilidad sobretodo en el ámbito científico, especialmente en temas donde el vocabulario es muy especializado. Debido a que este enfoque puede ser una manera externa de establecer relaciones para nosotros, vamos a proveer además de las entidades, a nivel global la posibilidad de incorporar una serie de palabras clave a cada parte relacionable del texto, de tal manera que al final puedan tener una función similar a las palabras que ya se especifican en las páginas `HTML` con el fin de ayudar a los robots de búsqueda.

Así pues se establecen los siguientes necesidades de atributos/elementos en nuestro lenguaje, que posteriormente jerarquizaremos y profundizaremos decidiendo si son elementos o atributos y completando lo que sería nuestro modelo de datos para las etiquetas.

Obligatorias:

- Un id que haga nuestra relación única dentro de nuestro documento. (Además de que cada elemento sería aconsejable que tuviera un Id de tal manera que puedan montarse colecciones a nivel externo para que sea más sencillo compartir colecciones con otras personas). Por ejemplo si establezco que el Id de Wolfang Amadeus Mozart es el 1865 en una colección externa. Aunque yo haya definido de una manera diferente esa etiqueta en cada una de las relaciones, o otra persona de otro país haya rellenado de otra manera los atributos, podemos saber si ambos estamos usando el mismo sistema de Ids de referencia, que estamos hablando de la misma persona.

Opcionales (estos atributos/elementos tendrían un fin de establecer relaciones a mayor nivel si el usuario de la aplicación final así lo quisiera):

- Una descripción opcional: esta descripción aunque no es un elemento que permita clasificar elementos o establecer elementos, puede ser de mucha utilidad de cara al uso posterior en una aplicación, puesto que cuando el usuario está viendo partes de documento a relacionar o ya relacionadas, puede ver estas anotaciones que se hayan realizado, o realizarlas el mismo de cara a que se entienda que forma parte de esa relación.
- Una serie de palabras claves con la que establecer relaciones a mayores: Como hemos visto con los tesauros e incluso en muchas páginas webs dentro de la parte meta del encabezado, se permite la inclusión de una serie de palabras claves, también conocidas

como keywords, que permiten indexar de una manera más eficaz a buscadores o similares. El uso de estas palabras para realizar relaciones y aprovechar los mecanismos que nos proporcionan los Tesauros, son un elemento muy a tener en cuenta en nuestro lenguaje.

- Un Tema: la idea es que el tema o los temas de los que pueda constar la parte relacionable, nos resulte útil a la hora de establecer materias o áreas de estudio. Por ejemplo un tema podría ser: Ciencias de la Naturaleza y el Medioambiente, o podría ser Programación Orientada a Objetos. Sería interesante que cada parte pudiera tener varios temas, debido a que muchas veces dentro de un texto, tenemos varios temas en conjunto. Por ejemplo, en un determinado momento podemos hablar de Física dentro de una parte en la que se está explicando un principio matemático, por eso es algo que intentaremos tener en cuenta a la hora de implementar el lenguaje.

- Una franja de tiempo en el que podamos situar el contenido a relacionar, esto es importante porque aunque podríamos usar los temas para clasificar este tipo de contenido, como por ejemplo decir, aquí se está hablando del Siglo de Oro Español, parece más interesante poder establecer una serie de fechas asociadas a un nombre o indicativos. De esta manera podemos reutilizarla desde para decir que este periodo de tiempo se denomina Modernismo y se da en una situación geográfica (ver más adelante su definición) de tal año a tal año, como que una persona vivió en una franja de tiempo. Es por ello que parece un elemento bastante importante a la hora de clasificar.

- Una zona geográfica donde podamos situar el contenido a relacionar: al igual que el contenido del tiempo, quizás podría englobarse dentro de la parte de tema, pero es un clasificador excelente para realizar conexiones con otras colecciones, puesto que podríamos relacionar diferentes textos basándonos en la ubicación del contenido de estos textos.

- Una persona: a nivel básico a la hora de hablar de escritores, cantantes, compositores, científicos, ingenieros parece una pieza de lo más básica y útil a la hora de clasificar contenido.

- Un objeto: nos referimos a objetos de cualquier tamaño, que para que resulte de más utilidad deberemos dotar de una serie de atributos para que tengan un sentido completo. La idea es poder describir objetos inertes desde un planeta, hasta una molécula de agua.

- Un ser vivo: para clasificar todos los Seres Vivos que abarcan el Universo. Debe ser capaz de describir desde un animal complejo como puede ser un mamífero, hasta a una célula como puede ser una neurona.

- Colecciones de todo lo anterior: Es decir, conjuntos de Seres Vivos, de personas, de objetos, de tal manera que podamos definir agrupaciones. Por ejemplo Bandas de Música, o incluso podamos especificar que ciertos objetos son en realidad agregaciones de otros muchos, como pueden ser minerales formados a partir de conjuntos químicos, etc.

En el fondo aunque en la descripción hemos dicho que solo uno, es cierto que podemos incluir dentro de esa parte varias personas, varias zonas geográficas, etc con el fin de no establecer al final un nivel tan pequeño que acabemos con demasiadas etiquetas para fragmentos de documentos muy muy pequeños lo que al final sería de escasa utilidad para todos.

De tal manera que la jerarquía de nuestras etiquetas quedaría representada por la siguiente imagen:

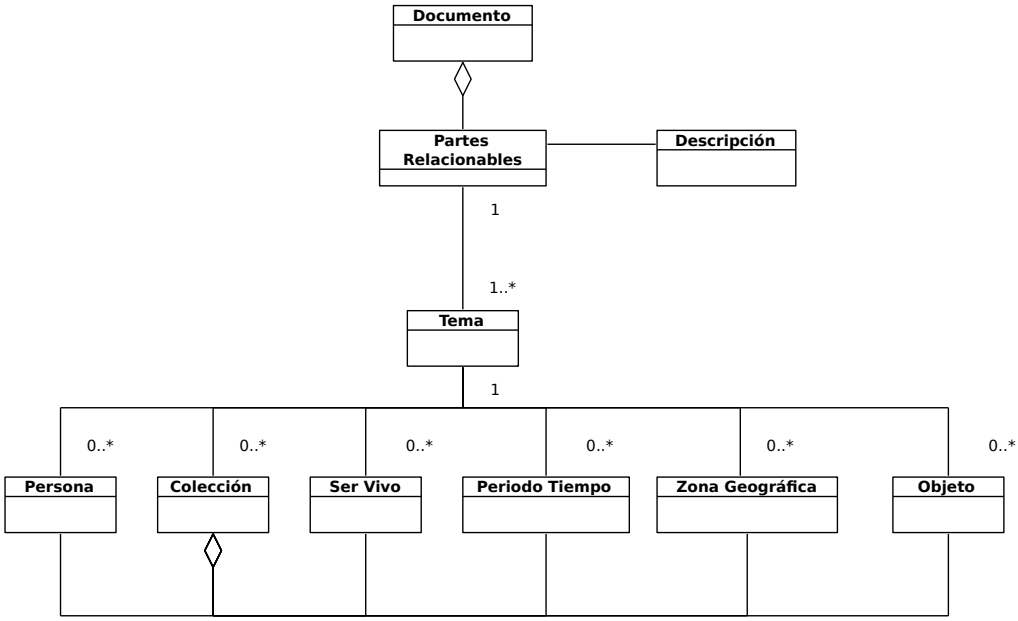

Figura 6.1: Jerarquía de elementos a tener en cuenta.

A partir de esta imagen, podemos profundizar en los diferentes atributos que puede tener cada elemento. Sobretodo las conclusiones que extraemos del visionado de la imagen, es que la mayoría de los elementos van destinados a catalogar cosas a muy alto nivel, de una manera muy genérica para que se puedan reutilizar sencillamente y por otro lado, cabe destacar la posibilidad de hacer coleccione, según el diagrama, de cualquier tipo de objetos, pero más adelante investigaremos si es viable esto en el uso de XML.

6.2. Extensión para medios continuos

Por otro lado tenemos que tener en cuenta la posibilidad de incorporar medios continuos a nuestra colección y lo más importante poder relacionar este contenido con el resto. Para

esto no nos sirve lo anterior, puesto que la inmensa mayoría de formatos de vídeo no permite la inclusión de datos en mitad de un vídeo, aparte de que sería muy costoso la gestión de estos datos. Es por esto, que para definir nuestras necesidades y tomar una decisión final, podemos ver como trabajan algunos ejemplos de fichero de textos con medios continuos.

Probablemente uno de los formatos menos conocidos y que sin embargo vemos más a nuestro alrededor es el que implementa YouTube.

En YouTube hay cabida dos tipos de formatos que trabajan en dos áreas muy diferenciadas. Uno es el formato que se emplea en el `API` que YouTube pone a disposición de los programadores que podemos encontrar en [3].

El otro es un formato interno, que sabemos por ejemplos encontrados [19] que emplea para superponer los diferentes comentarios y etiquetas a un vídeo. Para este formato no tenemos la estructura pública, pero con un poco de imaginación podemos imaginarnos como funciona y tomar unas cuantas ideas que pueden ser muy útiles de cara a nuestro proyecto.

6.2.1. Estudiando el API de YouTube

Obviamente, para nuestros propósitos no vamos a necesitar una cantidad inmensa de material, ni un dominio absoluto del API de YouTube puesto que lo único que buscamos es un análisis de los diferentes mensajes que se intercambian, para ver que clase de atributos y necesidades podemos aprovechar para nuestro cometido de éstos.

En un primer vistazo al API, vemos que está dividida en dos partes, la parte en la que habla de como insertar el reproductor en una página web, y la otra que habla de cómo podemos hacer invocaciones para obtener datos de vídeos, realizar búsquedas, etc. Obviamente la primera no es necesaria, puesto que en nuestro lenguaje cosas como el tamaño, resolución o propiedades del vídeo consideramos que no van a ser de excesiva utilidad a la hora de relacionar vídeos con otros vídeos y documentos.

Si analizamos los mensajes de ejemplo que aparecen por ejemplo en la búsqueda de canales y en la obtención de datos de vídeo, obtenemos las siguientes conclusiones:

- Todos los vídeos tienen un id y un título, como era de esperar.
- Además, tiene una fecha de modificación y de subida, que en nuestro caso no vamos a considerar necesario.
- Contiene un autor, con una serie de datos asociados.
- Tiene una descripción y una serie de palabras clave.
- El vídeo pertenece a una categoría.

- Tenemos disponible la duración del vídeo.
- Finalmente tenemos datos como puntuación, comentarios y estadísticas que para nuestro propósito no aportan nada.

A partir de estos datos, podemos ver que realmente los datos más característicos de un vídeo y que además podemos aprovechar para nuestra definición en el lenguaje, coinciden casi plenamente con lo que habíamos propuesto en la parte anterior. Básicamente tenemos una descripción, autores, palabras clave y categorías (que en el contexto anterior podría encajar perfectamente con nuestro elemento Tema).

Esto quiere decir que con todo lo que ya teníamos definido anteriormente, quizás añadiendo alguna parte más, sería posible incluir vídeos en nuestra colección. Como obviamente estas etiquetas no podemos tenerlas insertadas dentro del propio vídeo, habría que tener en cuenta que tendrían que ir en ficheros externos asociados.

Por desgracia, aún no es suficiente. Los vídeos y en general los medios continuos, precisamente por su característica de continuos pueden versar sobre muchos temas y estar presentes partes muy diferentes dentro del mismo contenido, es decir, que a lo mejor lo que yo estoy viendo en el minuto 10, no tiene nada que ver con lo que se escucha en el minuto 66. Es por esto por lo que para poder incorporar medios continuos a nuestras colecciones, mediante nuestro lenguaje de etiquetado, necesitamos ser aún más específicos, asociando cosas por ejemplo al minuto y segundo exacto del vídeo que está siendo etiquetado. Con el fin de analizar ésto e incluso observar si nos da alguna idea, o podemos aprovechar algo más, vamos a echar un vistazo al formato que tiene definido YouTube para realizar las anotaciones.

6.2.2. Analizando como funcionan las etiquetas que emplea YouTube para anotaciones

El sistema que haya empleado YouTube para las anotaciones de los vídeos, puede ser de gran utilidad, debido a que cada anotación aparece en un determinado momento del vídeo, mostrando a veces mensajes sobre lo que está pasando en la escena (la anotación aparece en una determinada posición) y por otro lado porque a veces tiene una funcionalidad especial, como permitirnos escoger entre dos posibilidades o incluso redireccionarnos hacia otra página web.

A través de la herramienta de [19] obtenemos el XML para verlo y lo analizamos(solo se muestra un trozo):

```
<document latest_timestamp="1323711029115108"
    polling_interval="30">
<requestHeader video_id="tbEei0I3kMQ"/>
```

```
 <annotations>
      <annotation author="werneroi" id="annotation_122476"
         style="popup" type="text">
         <TEXT>Don't be shy.I'm sure you will get it right!
         Click a card now.</TEXT>
         <segment>
             <movingRegion type="rect">
                <rectRegion h="7.5" t="0:00:54.00"
                    w="85.4169998169"
                    x="6.66699981689"
                    y="91.9440002441"/>
                <rectRegion h="7.5" t="0:01:2.30"
                    w="85.4169998169"
                    x="6.66699981689"
                    y="91.9440002441"/>
             </movingRegion>
         </segment>
      </annotation>
...
   </annotations>
```

Como podemos ver, la información que incluye es sobretodo referida al momento en el que ha de aparecer esta anotación, el tiempo que ha de aparecer, su texto y el cuadrado que pinta YouTube encima del vídeo para posicionarla.

En nuestro caso la información útil para nosotros puede ser el Id, cosa que teníamos ya contemplada en nuestro esquema anterior y sobretodo podemos aprovechar la parte del tiempo, por lo que queda claro, que debemos establecer una serie de limitadores en nuestro lenguaje para delimitar una parte de tiempo del vídeo, que es la que se relacionará con documentos o otras partes del vídeo.

6.2.3. Otros formatos...

Existen en la actualidad otros formatos para describir lo que está apareciendo en ellos. De los cuales los más conocidos son aquellos que nos permiten visualizar subtítulos en los videos.

Obviamente, estos formatos tienen un parecido muy similar con el que hemos visto de anotaciones de YouTube, si bien son mucho más simples, debido a que deben ser fáciles de procesar en una pequeña porción de tiempo, con el objetivo de mostrar el subtítulo en el momento que corresponde.

Los más conocidos son el SubRip (extensión srt) y el MicroDVD (extensión sub). Ambos tienen una estructura similar, muy básica, con un formato como el que sigue:

```
{frame inicial}{frame final}Texto del subtítulo|Texto2(opcional)
```

Es debido a su sencillez, por lo que se ha decidido que era más interesante en un primer momento analizar el de YouTube, donde se preveían más complejidades y posibilidades de cara a introducir otras etiquetas.

6.3. Definiendo una estructura

Así pues y visto lo anterior, debemos diseñar una estructura básica que nos permita definir relaciones en ciertos intervalos del vídeo. Es por ello que vamos a optar por introducir un elemento denominado Parte Multimedia, que heredará todos los atributos de nuestras Partes Relacionables, pero que además ha de tener una serie de atributos a mayores que nos permitan esta gestión. Estos atributos al menos, deben ser:

- Tiempo de Inicio: Seleccionamos como medida el tiempo en un formato que definiremos más adelante, debido a que otras medidas como hemos visto a lo largo de la investigación como son los frames, pueden no tener sentido en medios que no sean de video, como pueden ser audio o similares. Obviamente un problema y algo en lo que hay que hacer hincapié, es que los frames son una medida absoluta, mientras que el tiempo es una medida relativa, dado que está en relación entre los frames de un vídeo y su velocidad de reproducción (por ejemplo los televisores PAL reproducen a 25 frames por segundo).
- Tiempo de Fin: Momento hasta el cual dura la parte que vamos a relacionar de ese medio.
- Duración: Aunque este atributo es derivable de los dos anteriores, siempre es interesante de cara a ahorrarse un esfuerzo de cálculo introducir esta información. Es por esto que quizás consideraremos en el futuro que debería ser un atributo opcional.

Debemos tener en cuenta de que entonces nuestra antigua jerarquía estaba formada por un nodo padre que era Documento y que ahora seguramente haya que añadir algunos atributos para que sea posible especificar cierta información acerca de medios audiovisuales que pueden contener una serie de parámetros que otro tipo de documentos como son presentaciones, o documentos de texto, no tienen.

Así pues después de estos añadidos podemos definir que el diagrama de clases que define las clases del dominio y las relaciones entre éstas, salvo que varíe en los siguientes puntos

cuando definamos elemento por elemento, la estructura final de la que vamos a disponer, debería quedar así:

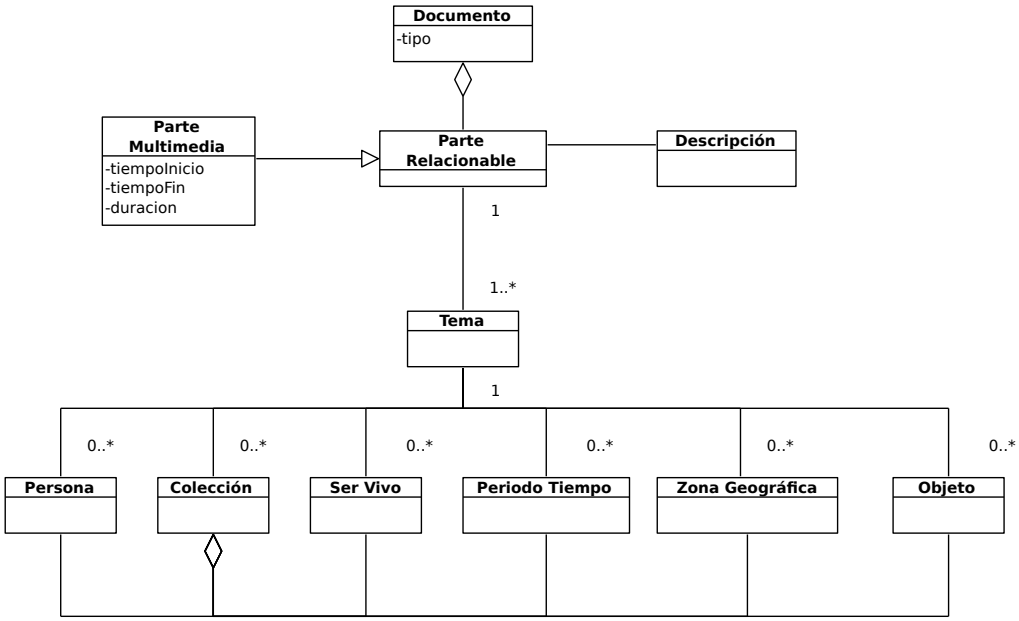

Figura 6.2: Jerarquía con la inclusión de medios continuos.

Como podemos observar en la jerarquía, además de incorporar Parte Multimedia se ha incorporado el atributo tipo al elemento Documento. Esto es debido a que de alguna manera tenemos que controlar el tipo de documento al que pertenecen las partes, puesto que no tiene sentido que un documento de texto tenga Partes Multimedias o viceversa, que un Documento Multimedia, tenga otro tipo de partes que no son Multimedia.

Capítulo 7

Construyendo el lenguaje

En este capítulo se pretende obtener una estructura final del lenguaje, ya con todos los posibles elementos y atributos de éstos bien formalizados y declarados sintácticamente, para en el siguiente capítulo, ver por último un ejemplo de funcionamiento del lenguaje y comprobar que es usable y que puede ser fácilmente empleado por una aplicación con el fin de gestionar todas las etiquetas de una colección.

7.1. Definiendo el esquema

Aunque XML permite un uso libre de las etiquetas, siemrpe y cuandos e respeten las reglas de formación y sintácticas del lenguaje, empleando cualquier tipo de elementos y de atributos, con el fin de restringir y por tanto hacer nuestro lenguaje más robusto, debemos definir una serie de restricciones que especifiquen cuándo un documento está bien formado, siguiendo las pautas de nuestro lenguaje.

En primer lugar, realizaremos unas tablas con cada Elemento, desglosando los diferentes atributos que puede tener y que valores podrán ir tomando esos atributos. En un segundo paso, especificaremos en un fichero el esquema con el que podremos validar que el etiquetado se ha realizado de una manera correcta.Este esquema se definirá siguiendo el estándar de DTD que aporta la W3C sobre XML.

Para realizar esto de una manera estructurada, lo abordaremos de la manera que acontece:

- Primero se definirá el elemento con todos sus atributos, indicando en cada uno el tipo que pueden tener (de acuerdo a la especificación del DTD podrán contener otros

elementos, o ser de tipo vacío (no contienen nada), o bien podrán contener texto y elementos, o bien solo texto.

- A continuación después de la tabla en la que se describen, se especificará esa parte del DTD ya formateada correctamente de acuerdo a las reglas de formación de la W3C.

Finalmente, cuando ya tengamos todo correctamente definido, mostraremos como quedaría nuestra especificación del DTD final (ya en el siguiente capítulo) y analizaremos su contenido para ver si es posible realizarla algunas mejoras, o añadidos.

La manera en la que vamos a ir definiendo los elementos va a ser de abajo hacia arriba, es decir, primero describiremos y definiremos los elementos más básicos sobre los que además probablemente se van a ir apoyando el resto de los elementos de la jerarquía e iremos definiendo y especificando conforme avancemos los elementos más altos de la jerarquía, que contienen a estos elementos.

Para la definición de los elementos y sus atributos seguiremos la siguiente plantilla:

Nombre del Elemento	
Atributo 1	Descripción del atributo 1
...	
Atributo n	Descripción del atributo n

7.1.1. Zona Geográfica

El elemento Zona Geográfica, tiene como misión describir posibles emplazamientos como son países, continentes, regiones, ciudades, que aparezcan dentro del contenido que pretendemos describir.

Para lo cual se proponen los siguientes atributos:

Zona Geográfica	
Id	Identificador de la Zona Geográfica, este Id será obligatorio y único dentro del elemento.Todos aquellos elementos de la colección que sean Zonas Geográficas con el mismo Id, estarán refiriéndose forzosamente a la misma Zona Geográfica.[1]
Nombre	Nombre de la Zona Geográfica, también tiene que ser obligatorio y único dentro del elemento.
Tipo	Campo opcional donde se permitirá especificar si hablamos de un país, una ciudad, un continente, etc.
Coordenadas geográficas	Campo opcional donde se permitirá introducir coordenadas con la posición del elemento a describir. Para lo cual definiremos coordenada con dos atributos a mayores que serán obligatorios que van a ser latitud y longitud, que van a contener la latitud y longitud de la posición. De cara a no volverse loco a la hora de establecer relaciones con las diferentes coordenadas y dado que existen diferentes sistemas de proyecciones, recomendamos emplear siempre la proyección WGS84 que abarca todas las posiciones posibles del mundo y emplear el sistema decimal para especificar las coordenadas.
Descripción	Campo opcional donde se permite introducir descripciones sobre la Zona Geográfica que se habla. Puede ser útil a la hora de ir analizando las diferentes etiquetas para el usuario.

Como fruto de este elemento, además nos ha surgido el elemento coordenada que tendrá como se ha descrito anteriormente, esta estructura:

Coordenada	
Longitud	Campo obligatorio donde vendrá la longitud. (Como se ha descrito en el punto anterior, en formato decimal y proyección WGS84).
Latitud	Campo obligatorio donde vendrá la latitud. (Como se ha descrito en el punto anterior, en formato decimal y proyección WGS84).

Así pues del elemento Coordenada tendríamos la siguiente especificación en el DTD:

```
<!ELEMENT coordenada EMPTY>
<!ATTLIST coordenada longitud CDATA #REQUIRED>
<!ATTLIST coordenada latitud CDATA #REQUIRED>
```

Este DTD indica que por un lado el elemento coordenada en su interior no puede contener

[1] A lo largo de todos los Ids que definiremos en el documento se verá que no se emplea el tipo de atributo ID de XML. Esto es debido a que los Ids en XML no pueden repetirse a lo largo de un documento, mientras que en un texto corriente es habitual que podamos encontrarnos varias referencias a la misma persona a lo largo del texto, es por ello que se ha decidido implementar con ese tipo y dejarlo como si fuera un atributo normal que se encargará de procesar el Software.

nada y que tiene que tener necesariamente dos atributos de tipo CDATA para que sea válido (CDATA en XML sería cualquier tipo de cadena salvo aquellas que contienen caracteres especiales propios del etiquetado como son comillas, ángulos, etc).

Un ejemplo de como podría ser un elemento coordenada sería la siguiente:

```
<coordenada longitud="-1.343212" latitud="40.2312322" />
```

Por otro lado la Zona Geográfica nos quedaría con un DTD como sigue:

```
<!ELEMENT zona_geografica (#PCDATA|coordenada)*>
<!ATTLIST zona_geografica id CDATA #REQUIRED>
<!ATTLIST zona_geografica nombre CDATA #REQUIRED>
<!ATTLIST zona_geografica tipo CDATA>
<!ATTLIST zona_geografica descripcion CDATA >
```

Un ejemplo de Zona Geográfica que podríamos encontrarnos podría ser similar a este:

```
En un lugar de la
<zona_geografica id="mancha" nombre="Castilla La Mancha"
    tipo="comunidad"
    descripcion="comunidad española que aparece en el Quijote">
    Mancha
    <coordenada longitud="-1.254221" latitud="39.584654"/>
</zona_geografica>
```

7.1.2. Periodo de Tiempo

El elemento periodo de tiempo tiene como misión establecer desde rangos hasta fechas concretas con el fin de marcar en nuestros documentos una fecha, o una época en la que ocurrió algo.

Para este elemento se van a proponer los siguientes atributos:

Periodo de Tiempo	
Id	Identificador del Periodo de Tiempo, este Id será obligatorio y único dentro del elemento.Todos aquellos elementos de la colección que sean Periodos de Tiempo con el mismo Id, estarán refiriéndose forzosamente al mismo Periodo de Tiempo.
Nombre	Nombre que queramos darle a ese Periodo de Tiempo, también tiene que ser obligatorio y único dentro del elemento.
Tipo	Campo opcional donde se permitirá especificar el tipo de Periodo de Tiempo. Por ejemplo aquí podemos indicar si hablamos de una época, un movimiento, un día señalado, un rango de fechas, etc
Día de Inicio	Campo opcional para marcar en que día comienza el periodo de Tiempo.
Mes de Inicio	Campo opcional para marcar en que mes comienza el periodo de Tiempo.
Año de Inicio	Campo opcional para marcar en que año comienza el periodo de Tiempo.
Día de Fin	Campo opcional para marcar en que día acaba el periodo de Tiempo.
Mes de Fin	Campo opcional para marcar en que mes acaba el periodo de Tiempo.
Año de Fin	Campo opcional para marcar en que año acaba el periodo de Tiempo.
Descripción	Campo opcional donde se permite introducir descripciones sobre el Periodo de Tiempo del que habla. Puede ser útil a la hora de ir analizando las diferentes etiquetas para el usuario.

Por lo que nuestra especificación en el DTD de nuestro lenguaje sería similar a:

```
<!ELEMENT periodo_tiempo(#PCDATA)*>
<!ATTLIST periodo_tiempo id CDATA #REQUIRED>
<!ATTLIST periodo_tiempo nombre CDATA #REQUIRED>
<!ATTLIST periodo_tiempo tipo CDATA>
<!ATTLIST periodo_tiempo dia_inicio CDATA>
<!ATTLIST periodo_tiempo mes_inicio CDATA>
<!ATTLIST periodo_tiempo ano_inicio CDATA>
<!ATTLIST periodo_tiempo dia_fin CDATA>
<!ATTLIST periodo_tiempo mes_fin CDATA>
<!ATTLIST periodo_tiempo ano_fin CDATA>
<!ATTLIST periodo_tiempo descripcion CDATA>
```

A partir del cual un ejemplo que podríamos obtener sería el siguiente:

```
Don Cristobal Colón descubrió Ámerica el día
```

```
<periodo_tiempo id="fechaDescubrimientoAmerica"
    nombre="descrubimientoDeAmerica"
    tipo="dia"
    dia_inicio="12"
    mes_inicio="Octubre"
    ano_inicio="1492"
    descripcion="Fecha en la que se descubre America">
    12 de Octubre de 1492 cuando Rodrigo de Triana gritó:
    Tierra a la vista!
</periodo_tiempo>
```

7.1.3. Objeto

Este elemento ha de ser empleado para etiquetar apariciones de objetos inertes de todos los tipos. Desde por ejemplo un cometa, un asteroide, hasta algo tan pequeño como un átomo, o un electrón.

Para este elemento se van a proponer los siguientes atributos:

Objeto	
Id	Identificador del Objeto, este Id será obligatorio y único dentro del elemento.Todos aquellos elementos de la colección que sean Objetos con el mismo Id, estarán refiriéndose forzosamente al mismo Objeto.
Nombre	Nombre del objeto que estamos etiquetando, también tiene que ser obligatorio y único dentro del elemento.
Tipo	Permite establecer un tipo de objeto a la hora de etiquetar que puede ser útil a la hora de realizar clasificaciones posteriores.
Descripción	Campo opcional donde se permite introducir descripciones sobre el Objeto del que habla. Puede ser útil a la hora de ir analizando las diferentes etiquetas para el usuario.
Tamaño	El usuario podrá especificar opcionalmente el tamaño del objeto que está etiquetando, de cara a posibles clasificaciones o búsquedas que desee realizar posteriormente.

Por lo que nuestra especificación en el DTD de nuestro lenguaje sería similar a:

```
<!ELEMENT objeto (#PCDATA)*>
<!ATTLIST objeto id CDATA #REQUIRED>
<!ATTLIST objeto nombre CDATA #REQUIRED>
<!ATTLIST objeto tipo CDATA>
```

```
<!ATTLIST objeto descripcion CDATA>
<!ATTLIST objeto tamano CDATA>
```

Un ejemplo de etiquetado de un objeto dentro de un documento, podría ser similar al siguiente:

```
<objeto id="Saturno" nombre="Saturno"
    tipo="Planeta"
    descripcion="Planeta del sistema Solar"
    tamano="Enorme">
     Saturno
</objeto> es el sexto planeta del Sistema Solar
y cuenta con un sistema de
<objeto id="anillo" nombre="anillo" tipo="joya"
    tamano="pequeño">
    anillos
</objeto> visible desde nuestro planeta.
```

7.1.4. Ser Vivo

Este elemento debe ser capaz de etiquetar y describir a todos los seres vivos que existen o existieron sobre la faz de la tierra. Será el elemento que emplee el usuario para etiquetar apariciones de Anfibios, Reptiles, etc...

Es por ello, que proponemos como criterio de clasificación la típica que se ha ido desarrollando dentro del mundo de la Biología durante años de Dominios y Reinos, realizada por primera vez por Linneo en 1735. [12]

Para este elemento se van a proponer los siguientes atributos:

Ser Vivo	
Id	Identificador del Ser Vivo, este Id será obligatorio y único dentro del elemento.Todos aquellos elementos de la colección que sean Seres Vivos con el mismo Id, estarán refiriéndose forzosamente al mismo Ser Vivo.
Nombre	Nombre del Ser Vivo que estamos etiquetando, también tiene que ser obligatorio y único dentro del elemento. Si el usuario ha especificado el nombre de la especie, como nombre puede emplearse el nombre común del ser vivo, o incluso si está hablando de su mascota puede emplear ese nombre para clasificarlo.
Dominio	Nombre del Dominio dentro del cual se clasifica al ser vivo.(Opcional)
Reino	Nombre del Reino dentro del cual se clasifica al ser vivo.(Opcional)
Filo	Nombre del Filo dentro del cual se clasifica al ser vivo.(Opcional)
Subfilo	Nombre del Subfilo dentro del cual se clasifica al ser vivo. Se incluye puesto que se ha visto que en numerosas ocasiones debido a la enorme cantidad de filos existentes, existe también el Subfilo.(Opcional)
Clase	Nombre de la Clase dentro del cual se clasifica al ser vivo.(Opcional)
Orden	Nombre del Orden dentro del cual se clasifica al ser vivo.(Opcional)
Familia	Nombre de la Familia dentro del cual se clasifica al ser vivo.(Opcional)
Género	Nombre del Género dentro del cual se clasifica al ser vivo.(Opcional)
Especie	Nombre de la Especie dentro del cual se clasifica al ser vivo.(Opcional)
Descripción	Campo opcional donde se permite introducir descripciones sobre el Ser Vivo del que habla. Puede ser útil a la hora de ir analizando las diferentes etiquetas para el usuario.

Por lo que nuestra especificación en el DTD de nuestro lenguaje sería similar a:

```
<!ELEMENT ser_vivo (#PCDATA)*>
<!ATTLIST ser_vivo id CDATA #REQUIRED>
<!ATTLIST ser_vivo nombre CDATA #REQUIRED>
<!ATTLIST ser_vivo dominio CDATA>
<!ATTLIST ser_vivo reino CDATA>
<!ATTLIST ser_vivo filo CDATA>
<!ATTLIST ser_vivo subfilo CDATA>
<!ATTLIST ser_vivo clase CDATA>
<!ATTLIST ser_vivo orden CDATA>
<!ATTLIST ser_vivo familia CDATA>
```

```
<!ATTLIST ser_vivo genero CDATA>
<!ATTLIST ser_vivo especie CDATA>
<!ATTLIST ser_vivo descripcion CDATA>
```

Un ejemplo de cómo podría aparecer una etiqueta dentro de un documento que trate sobre algún ser vivo, sería el siguiente:

```
El
<ser_vivo id="lobo" nombre="lobo"
    dominio="eucarionte"
    reino="animalia"
    filo="chordata"
    subfilo="vertebrata"
    clase="mammalia"
    orden="carnivora"
    familia="canidae"
    genero="canis"
    especie="canis lupus"
    descripcion="Lobo común que aparece en los
    textos del libro de Biología de primero
    de Bachillerato">
    Lobo
</ser_vivo> podríamos definirlo como uno de
los animales más bellos de nuestro litoral.
```

7.1.5. Persona

Este elemento permitirá etiquetar a aquellas personas que aparecen en el documento. Tenemos que poner especial atención en el hecho de que algunas de las personas que pueden aparecer en los documentos, pueden ser personajes de ficción y que no por ello debemos dejar de etiquetarlos, aunque si que deberemos brindar la posibilidad de discernir entre ellos mediante algún atributo.

Además debido a ello, es posible que si bien etiquetar personas reales, puede ser ciertamente sencillo conocer el valor de todas los atributos que se pueden emplear en ello, para personajes de ficción o del pasado, sería posible desconocer alguno de esos datos, es por eso que estos atributos deberán ser forzosamente opcionales.

Para este elemento se van a proponer los siguientes atributos:

Persona	
Id	Identificador de la Persona, este Id será obligatorio y único dentro del elemento.Todos aquellos elementos de la colección que sean Personas con el mismo Id, estarán refiriéndose forzosamente a la misma Persona.
Nombre	Nombre de la persona que estamos etiquetando. Será un campo obligatorio dentro de la etiqueta.
Primer Apellido	Primer apellido de la persona que estamos etiquetando.(Opcional)
Segundo Apellido	Segundo apellido de la persona que estamos etiquetando.(Opcional)
Lugar de Nacimiento	Se especificará empleando el elemento Zona Geográfica anteriormente especificado.(Opcional)
Fecha de Nacimiento	Se especificará empleando el elemento Periodo de Tiempo anteriormente especificado.(Opcional)
Lugar de Defunción	Se especificará empleando el elemento Zona Geográfica anteriormente especificado.(Opcional)
Fecha de Defunción	Se especificará empleando el elemento Periodo de Tiempo anteriormente especificado.(Opcional)
Ficción	Aquí le daremos posibilidad al usuario determinar si es un personaje real o de ficción.(Opcional)
Oficio	Podrá especificarse el oficio o oficios que tuviera esta persona.(Opcional)
Descripción	La etiqueta contendrá una posible descripción de este personaje. Puede ser por ejemplo a modo de biografía o simplemente de cara a reconocer mejor las marcas del etiquetado. (Opcional)

Si observamos, nos damos enseguida cuenta de una incoherencia que podría producirse al emplear XML. Esta incoherencia está en que si empleamos directamente los elementos de Zona Geográfica o de Periodo de Tiempo dentro de nuestro elemento, no sabremos cuando se está refiriendo a la fecha o el lugar de nacimiento y cuando se está refiriendo a la de defunción. Para resolver este problema, tenemos dos maneras de solucionarlo:

- Establecemos una recomendación en el lenguaje en el que por ejemplo estos elementos posean un Id característico con el cual se pueda averiguar esta información. Por ejemplo si la persona tiene Id “Mozart” propondríamos que el lugar de nacimiento tendría un Id “Mozart_lugar_nac”.
- Establecemos cuatro elementos nuevos que solo van a aparecer por debajo de este elemento y que van a contener una Zona Geográfica o un Periodo de Tiempo. El problema es que XML no permite establecer que solo tenga que aparecer uno, si no que tendría que ser trabajo del Software subyacente validar si solo aparece uno o no.

En este caso creemos que lo más conveniente es añadir nuevas etiquetas para delimitar dentro del elemento estas partes. Tiene el inconveniente de que hacemos más complejo el

lenguaje y hay más cosas que añadir, pero por otro lado nos va a resultar muy útil a la hora de clasificar nuestra información correctamente.

Por ello vamos a añadir dos subelementos al elemento Persona que van a ser Nacimiento y Muerte. Dos etiquetas que podrán contener Zonas Geográficas y Periodos de Tiempo.

A continuación se muestra, como quedaría la especificación en el DTD de estos dos elementos (que como se puede observar va a ser muy simple y sin atributos):

```
<!ELEMENT nacimiento (#PCDATA|periodo_tiempo|zona_geografica)>
<!ELEMENT muerte (#PCDATA|periodo_tiempo|zona_geografica)>
```

Y por lo que finalmente la especificación en el DTD el elemento persona, sería así:

```
<!ELEMENT persona
 (#PCDATA|periodo_tiempo|zona_geografica|nacimiento|muerte)*>
<!ATTLIST persona id CDATA #REQUIRED>
<!ATTLIST persona nombre CDATA #REQUIRED>
<!ATTLIST persona primer_apellido CDATA>
<!ATTLIST persona segundo_apellido CDATA>
<!ATTLIST persona ficcion (si | no)>
<!ATTLIST persona oficio CDATA>
<!ATTLIST persona descripcion CDATA>
```

Al igual que en anterior ocasiones mostraremos un pequeño ejemplo de cómo funcionaría la etiqueta en un caso real:

```
Uno de los compositores que es considerado
el puente entre el
<periodo_tiempo id="Clasicismo"
    nombre="Clasicismo Musical"
    tipo="epoca"
    ano_inicio="1750"
    ano_fin="1800">
    Clasicismo
</periodo_tiempo>
y el
<periodo_tiempo id="Romanticismo"
    nombre="Romanticismo Musical"
    tipo="epoca"
    ano_inicio="1800"
```

```
    ano_fin="1912">
    Romanticismo
</periodo_tiempo>
es
<persona id="Beethoven"
nombre="Ludwig"
primer_apellido="van Beethoven"
ficcion="no"
oficio="músico">
Beethoven que
<nacimiento>nació en la
conocida ciudad de
  <zona_geografica id="Bonn"
    nombre="NacimientoBeethoven" tipo="ciudad">
    Bonn
  </zona_geografica> en el año
  <periodo_tiempo id="16-dic-1770"
    nombre="NacimientoBeethoven"
    tipo="dia"
    dia_inicio="16"
    mes_inicio="Diciembre"
    ano_inicio="1770"
  </periodo_tiempo>
</nacimiento>
</persona>
```

7.1.6. Colección

Al pretender etiquetar colecciones tenemos que tener en cuenta los siguientes factores:

- Podemos emplear un enfoque en el que se fuerce a que las colecciones tengan que contener el mismo tipo de objetos, con el problema de que no es posible restringir mediante XML en el DTD este enfoque salvo que creemos un elemento de colección diferente por cada elemento que pretendamos insertar en la colección.
- Debido al punto anterior, creemos que es más interesante no establecer ningún tipo de restricción en aquellos elementos que pueda contener. Las colecciones afectarán a los elementos descritos anteriormente en el documento.
- A veces debido a las características del documento vamos a querer etiquetar en alguna parte una colección particular y sin embargo no van a aparecer sus componentes en el texto, para lo cual el caso de que solo se haga referencia a la colección sin que aparezcan sus componentes ha de estar disponible en el DTD. Un claro ejemplo podría

ser cuando hablamos de sociedades como un equipo de fútbol que podríamos considerar una colección de personas o incluso de personas y objetos si consideramos todos sus bienes.

Para este elemento se van a proponer los siguientes atributos:

Colección	
Id	Identificador de la Colección, este Id será obligatorio y único dentro del elemento.Todos aquellos elementos de la colección que sean Colecciones con el mismo Id, estarán refiriéndose forzosamente a la misma Colección.
Nombre	Nombre de la colección que estamos etiquetando. Será un campo obligatorio dentro de la etiqueta.
Tipo	Nos permitirá darle un tipo a la colección. Por ejemplo podemos decir que el tipo de la colección es una Sociedad, o una Agrupación, o un Equipo de Fútbol...(Opcional)
Descripción	La etiqueta contendrá una posible descripción la colección, como por ejemplo que se está hablando de una colección de Mamíferos, o de un equipo de Fútbol, etc... (Opcional)

Por lo que nuestra especificación en el DTD de nuestro lenguaje sería similar a:

```
<!ELEMENT coleccion
 (#PCDATA|periodo_tiempo|zona_geografica|objeto
 |ser_vivo|persona)*>
<!ATTLIST coleccion id CDATA #REQUIRED>
<!ATTLIST coleccion nombre CDATA #REQUIRED>
<!ATTLIST coleccion tipo CDATA>
<!ATTLIST coleccion descripcion CDATA>
```

Puede que el lector se sorprenda del porqué de no incluir que haya colecciones que puedan contener otras colecciones. Esta pregunta tiene una fácil solución. XML no permite definir elementos que se contengan así mismos de manera recursiva, por lo que no podemos especificar el DTD que una colección puede contener colecciones, porque a partir de ese momento violaríamos el estándar, los validadores existentes fallarían y además seguramente el Software que implemente este lenguaje, acabará teniendo múltiples problemas de implementación, dado que la mayoría de librerías tendrán esto en cuenta.

Un ejemplo de lo que podría ser una colección podría ser el siguiente:

```
<coleccion id="listaCompra"
  nombre="Lista de la Compra"
```

```
  descripcion="Lista de la compra de
    los Lunes">
  <objeto id="lechugaIceberg"
     nombre="lechuga" tipo="comida"
     descripcion="Lechuga Iceberg">
  lechuga,
  </objeto>
  <objeto id="tomatePera"
     nombre="tomate" tipo="comida"
     descripcion="Tomate Pera">
  tomate
  </objeto>
  y
  <objeto id="aceiteOliva"
     nombre="aceite" tipo="liquido"
     descripcion="Aceite de Oliva">
  aceite.
  </objeto>
</coleccion>
```

7.1.7. Tema

El Tema es uno de nuestros elementos claves dentro de las parte de un documento. Debido a nuestra definición en la jerarquía que aparece en los capítulos anteriores, pueden aparecer varios temas dentro de cada parte que se desea relacionar.

Estos Temas deberían ser las diferentes categorías en las que se puede clasificar el contenido de los elementos/texto que contiene en su interior. Por ejemplo un tema podría ser Guerra Civil en España, o Biología, o Ajedrez.

Para este elemento se van a proponer los siguientes atributos:

Tema	
Id	Identificador del Tema, este Id será obligatorio y único dentro del elemento.Todos aquellos elementos de la colección que sean Temas con el mismo Id, estarán refiriéndose forzosamente al mismo Tema.
Nombre	Nombre del tema que estamos tratando. Será un campo obligatorio dentro de la etiqueta.
Descripción	La etiqueta contendrá una posible descripción del Tema que se está tratando. Puede tratarse simplemente de un breve resumen sobre el contenido del texto, o del tema que se está tratando en el texto. (Opcional)

Por lo que nuestra especificación en el DTD de nuestro lenguaje sería similar a:

```
<!ELEMENT tema
 (#PCDATA|periodo_tiempo|zona_geografica
 |objeto|ser_vivo|persona|coleccion)*>
<!ATTLIST tema id CDATA #REQUIRED>
<!ATTLIST tema nombre CDATA #REQUIRED>
<!ATTLIST tema descripcion CDATA>
```

Mostraremos un ejemplo de como emplear la etiqueta al final de la descripción de todos los atributos donde ya veremos un ejemplo completo que incluya todo tipo de etiquetas y que será más rico para el usuario. Los anteriores elementos se habían puesto ejemplos uno a uno al ser pequeños y no ocupar una parte considerable, además de que para el lector que fuera leyendo el documento, le resultara más cómodo. Ahora que ya ha visto el funcionamiento y la dinámica en los anteriores elementos, creemos que no es necesario seguir incluyendo ejemplos de estas partes que tienen mayor densidad debido a la cantidad de etiquetas que puede contener.

7.1.8. Parte Relacionable

Como se ha descrito ya en la jerarquía es el elemento que contendrá todo el trozo de documento que se pretenda relacionar con otros. Debido a que solo tiene esa misión, se adjuntaba de una descripción como elemento en la jerarquía.

Visto el enfoque que hemos tomado en los anteriores elementos de incluir un campo de descripción, finalmente decidimos que mejor cuadra como atributo dentro del elemento con la diferencia de que para el usuario y en general en la hora posterior de recopilar información sobre el Texto, puede ser de gran ayuda, este campo será obligatorio.

Para este elemento se van a proponer los siguientes atributos:

Parte Relacionable	
Id	Identificador de la Parte Relacionable. Obligatoriamente no deberá existir otro Id similar dentro de todo el documento, puesto que cada parte es única dentro del documento.
Nombre	Nombre que queramos dotar a la Parte Relacionable. Será un atributo obligatorio.
Descripción	Una descripción todo lo larga que desee el usuario, donde se especifique de qué habla esa parte del documento.(Opcional)
Palabras Claves	Una serie de palabras claves que ayuden al usuario a la hora de realizar búsquedas sobre partes dentro del documento. Tendrán una utilidad similar a las keywords que se especifican en el Meta de HTML. Además recomendamos para hacer un mejor uso de esta posibilidad emplear un mecanismo basado en `Tesauros` como el que ya se habló anteriormente.

Por lo que nuestra especificación en el DTD de nuestro lenguaje sería similar a:

```
<!ELEMENT parte_relacionable (#PCDATA|tema)*>
<!ATTLIST parte_relacionable id CDATA #REQUIRED>
<!ATTLIST parte_relacionable nombre CDATA #REQUIRED>
<!ATTLIST parte_relacionable descripcion CDATA>
<!ATTLIST parte_relacionable keywords CDATA #REQUIRED>
```

7.1.9. Parte Multimedia

El fin de este elemento es mediante un fichero externo a un fichero multimedia, especificar las partes que posee este documento y ser capaz de describir que contienen, por ello se emplean los mismos atributos que en la Parte Relacionable, pero además se marcan una serie de atributos que hacen referencia al tiempo en el que comienza y acaba cada parte.

Para este elemento se van a proponer los siguientes atributos:

Parte Multimedia	
Id	Identificador de la Parte Multimedia. Obligatoriamente no deberá existir otro Id similar dentro de todo el documento, puesto que cada parte es única dentro del documento.
Nombre	Nombre que queramos dotar a la Parte Multimedia. Será un atributo obligatorio.
Descripción	Una descripción todo lo larga que desee el usuario, donde se especifique de qué habla esa parte del medio.
Palabras Claves	Al igual que ya se ha explicado en la Parte Relacionable.
Inicio	Especificará en un formato: HH:mm:SS el momento del medio en el que comienza la parte. Este atributo es obligatorio.
Fin	Especificará en un formato: HH:mm:SS el momento del medio en el que finaliza la parte. Este atributo es obligatorio.
Duración	Especificará en un formato: HH:mm:SS el tiempo que dura esa parte. Este atributo es obligatorio.

Por lo que nuestra especificación en el DTD de nuestro lenguaje sería similar a:

```
<!ELEMENT parte_multimedia (#PCDATA|tema)*>
<!ATTLIST parte_multimedia id CDATA #REQUIRED>
<!ATTLIST parte_multimedia nombre CDATA #REQUIRED>
<!ATTLIST parte_multimedia descripcion CDATA>
<!ATTLIST parte_multimedia keywords CDATA #REQUIRED>
<!ATTLIST parte_multimedia inicio CDATA #REQUIRED>
<!ATTLIST parte_multimedia fin CDATA #REQUIRED>
<!ATTLIST parte_multimedia duracion CDATA #REQUIRED>
```

7.1.10. Documento

Tal y como indica la especificación de XML, todos lenguaje basado en XML debe contener un elemento raíz que contenga al resto de elementos. En nuestro caso este rol lo va a desempeñar el documento. El documento además podremos especificar un tipo, para definir si estamos dentro de un documento basado en texto como es el caso de los OpenDocument, PDF, Latex, etc (que se ha demostrado diferentes posibilidades para inserción de etiquetas en ellos), o por el contrario estamos en un Documento Multimedia, el cual llevará asociado un fichero que es al que hace referencia.

Para este elemento se van a proponer los siguientes atributos:

Documento	
Id	Identificador del documento. Puede ser de utilidad de cara a organizar nuestros documentos dentro de la colección mediante un Software Externo.
Nombre	Nombre del documento.Será un atributo obligatorio.
Tipo de Documento	Atributo obligatorio en el que se indicará si es un documento multimedia o no.
Fichero Externo	Atributo que debemos usar en el caso de los ficheros multimedia puesto que nos indicará el nombre y ubicación del fichero que está describiendo este archivo externo en XML.(Se recomienda describir una ruta relativa, para que de esta manera sea más fácil el intercambio entre colecciones de etiquetas de diferentes personas).

Por lo que nuestra especificación en el DTD de nuestro lenguaje sería similar a:

```
<!ELEMENT documento (#PCDATA|parte_relacionable|parte_multimedia)*>
<!ATTLIST documento id CDATA #REQUIRED>
<!ATTLIST documento nombre CDATA #REQUIRED>
<!ATTLIST documento tipo (texto | multimedia) #REQUIRED>
<!ATTLIST documento fichero_externo CDATA>
```

7.1.11. Jerarquía final

Finalmente, si recopilamos todas las definiciones que hemos ido haciendo a lo largo de las subsecciones y las fundimos en un único DTD ahora ya sí en un formato DTD completo, tendríamos lo siguiente:

```
<?xml version="1.1" encoding="UTF-8"?>
<!--Declaracion raiz-->
<!DOCTYPE documento [
  <!--Nodo raiz Documento-->
  <!ELEMENT documento (#PCDATA|parte_relacionable|parte_multimedia)*>
  <!ATTLIST documento id CDATA #REQUIRED>
  <!ATTLIST documento nombre CDATA #REQUIRED>
  <!ATTLIST documento tipo (texto | multimedia) #REQUIRED>
  <!ATTLIST documento fichero_externo CDATA>

  <!--Parte multimedia-->
  <!ELEMENT parte_multimedia (#PCDATA|tema)*>
  <!ATTLIST parte_multimedia id CDATA #REQUIRED>
  <!ATTLIST parte_multimedia nombre CDATA #REQUIRED>
```

```
<!ATTLIST parte_multimedia descripcion CDATA>
<!ATTLIST parte_multimedia keywords CDATA #REQUIRED>
<!ATTLIST parte_multimedia inicio CDATA #REQUIRED>
<!ATTLIST parte_multimedia fin CDATA #REQUIRED>
<!ATTLIST parte_multimedia duracion CDATA #REQUIRED>

<!--Parte relacionable-->
<!ELEMENT parte_relacionable (#PCDATA|tema)*>
<!ATTLIST parte_relacionable id CDATA #REQUIRED>
<!ATTLIST parte_relacionable nombre CDATA #REQUIRED>
<!ATTLIST parte_relacionable descripcion CDATA>
<!ATTLIST parte_relacionable keywords CDATA #REQUIRED>

<!--Tema-->
<!ELEMENT tema
  (#PCDATA|periodo_tiempo|zona_geografica
   |objeto|ser_vivo|persona|coleccion)*>
<!ATTLIST tema id CDATA #REQUIRED>
<!ATTLIST tema nombre CDATA #REQUIRED>
<!ATTLIST tema descripcion CDATA>

<!--Coleccion-->
<!ELEMENT coleccion
  (#PCDATA|periodo_tiempo|zona_geografica|objeto
   |ser_vivo|persona)*>
<!ATTLIST coleccion id CDATA #REQUIRED>
<!ATTLIST coleccion nombre CDATA #REQUIRED>
<!ATTLIST coleccion tipo CDATA>
<!ATTLIST coleccion descripcion CDATA>

<!--Persona-->
<!ELEMENT persona
  (#PCDATA|periodo_tiempo|zona_geografica|nacimiento|muerte)*>
<!ATTLIST persona id CDATA #REQUIRED>
<!ATTLIST persona nombre CDATA #REQUIRED>
<!ATTLIST persona primer_apellido CDATA>
<!ATTLIST persona segundo_apellido CDATA>
<!ATTLIST persona ficcion (si | no)>
<!ATTLIST persona oficio CDATA>
<!ATTLIST persona descripcion CDATA>
<!ELEMENT nacimiento (#PCDATA|periodo_tiempo|zona_geografica)>
<!ELEMENT muerte (#PCDATA|periodo_tiempo|zona_geografica)>

<!--Ser vivo-->
<!ELEMENT ser_vivo (#PCDATA)*>
```

```
  <!ATTLIST ser_vivo id CDATA #REQUIRED>
  <!ATTLIST ser_vivo nombre CDATA #REQUIRED>
  <!ATTLIST ser_vivo dominio CDATA>
  <!ATTLIST ser_vivo reino CDATA>
  <!ATTLIST ser_vivo filo CDATA>
  <!ATTLIST ser_vivo subfilo CDATA>
  <!ATTLIST ser_vivo clase CDATA>
  <!ATTLIST ser_vivo orden CDATA>
  <!ATTLIST ser_vivo familia CDATA>
  <!ATTLIST ser_vivo genero CDATA>
  <!ATTLIST ser_vivo especie CDATA>
  <!ATTLIST ser_vivo descripcion CDATA>

  <!--Objeto-->
  <!ELEMENT objeto (#PCDATA)*>
  <!ATTLIST objeto id CDATA #REQUIRED>
  <!ATTLIST objeto nombre CDATA #REQUIRED>
  <!ATTLIST objeto tipo CDATA>
  <!ATTLIST objeto descripcion CDATA>
  <!ATTLIST objeto tamano CDATA>

  <!--Periodo de Tiempo-->
  <!ELEMENT periodo_tiempo(#PCDATA)*>
  <!ATTLIST periodo_tiempo id CDATA #REQUIRED>
  <!ATTLIST periodo_tiempo nombre CDATA #REQUIRED>
  <!ATTLIST periodo_tiempo tipo CDATA>
  <!ATTLIST periodo_tiempo dia_inicio CDATA>
  <!ATTLIST periodo_tiempo mes_inicio CDATA>
  <!ATTLIST periodo_tiempo ano_inicio CDATA>
  <!ATTLIST periodo_tiempo dia_fin CDATA>
  <!ATTLIST periodo_tiempo mes_fin CDATA>
  <!ATTLIST periodo_tiempo ano_fin CDATA>
  <!ATTLIST periodo_tiempo descripcion CDATA>

  <!--Zona Geografica-->
  <!ELEMENT zona_geografica (#PCDATA|coordenada)*>
  <!ATTLIST zona_geografica id CDATA #REQUIRED>
  <!ATTLIST zona_geografica nombre CDATA #REQUIRED>
  <!ATTLIST zona_geografica tipo CDATA>
  <!ATTLIST zona_geografica descripcion CDATA >
  <!ELEMENT coordenada EMPTY>
  <!ATTLIST coordenada longitud CDATA #REQUIRED>
  <!ATTLIST coordenada latitud CDATA #REQUIRED>

]>
```

Por tanto y debido a que hemos introducido una serie de elementos nuevos para mejorar la usabilidad y la eficiencia de nuestro lenguaje, a continuación se muestra la evolución del diagrama UML que se mostró en el capítulo anterior, mostrando los nuevos elementos que se han añadido así como todos los atributos que pueden poseer estos elementos.

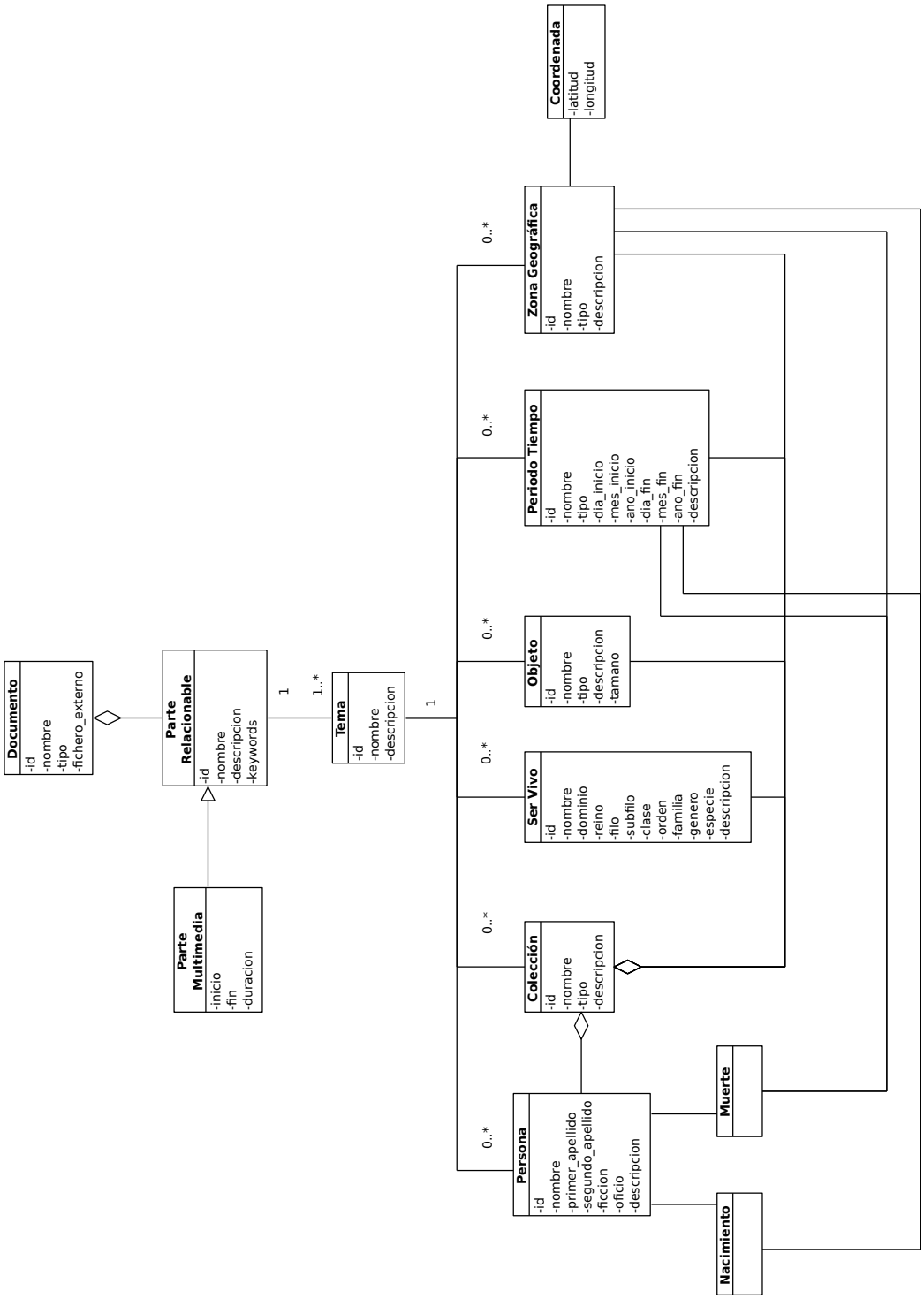

Figura 7.1: Jerarquía Final

7.2. Comprobando que es correcto en términos de la especificación W3C

A continuación comprobaremos que la especificación de nuestro lenguaje es válida y daremos un repaso a aquellas reglas de la W3C más importantes que afectan a la hora de realizar un documento con XML.

Todo documento XML así como su DTD debe tener un prólogo, este prólogo tiene la siguiente misión:

- Marcar el documento como texto XML.
- Declarar cuál es la versión de XMl utilizada para elaborar el documento.
- Aportar información sobre la codificación empleada.
- Incluir una declaración de si el documento es autónomo.

Como se puede observar es correcto, está definida por la parte:

```
<?xml version="1.1" encoding="UTF-8"?>
```

Todos los documentos XML tienen obligatoriamente que tener un elemento raíz (uno y solo uno), un documento XML sin el elemento raíz o con dos o más elementos raíz no está bien formado.

En nuestro caso el elemento raíz sería documento. Es obvio que nosotros al definir el DTD no podemos controlar si el usuario luego crea más de uno, para lo cual el propio XML establecerá que el documento no es válido, pero sin embargo sí es nuestra obligación darle la definición del elemento raíz.

Ningún elemento de la especificación de XML puede llamarse XML en cualquiera de sus formas (mayúsculas, minúsculas, alternadas).

En nuestro caso como se puede comprobar, tampoco hemos utilizado en ningún momento ni definido ese nombre.

Los nombres de los elementos, deben componerse al menos de una letra, (desde la "a" a la "z" o desde la "A" a la "Z"), de un guión bajo o de un carácter de dos puntos. El carácter inicial puede estar seguido de otros caracteres, definidos dentro del grupo "NameChar", es decir, una o más letras, dígitos,puntos,guiones,guiones bajos en cualquier combinación.

En nuestro caso como se puede comprobar, todos los nombres tanto de los elementos como de los atributos cumplen estas normas perfectamente.

Todas las declaraciones de tipo de Elementos se componen de una cadena "<!ELEMENT" seguida de uno o más espacios en blanco, un nombre XML, espacios en blanco, la especificación del contenido del elemento, espacios en blanco opcionales y el carácter de cierre.

Como podemos ver en cualquier elemento, se ha definido siguiendo esa regla, por ejemplo:

```
<!ELEMENT zona_geografica (#PCDATA|coordenada)*>
```

Todas las declaraciones de listas de Atributos comienzan con la cadena "<!ATTLIST" a continuación y separados por al menos un espacio, el nombre del elemento al que pertenecen los atributos. Seguidamente el nombre del atributo que se está declarando, espacios, el tipo del atributo y su declaración por defecto.

Como podemos ver en las listas definidas en el DTD, son correctas y cumplen este formato.

```
<!ATTLIST zona_geografica nombre CDATA #REQUIRED>
```

El lector puede mostrarse confundido debido a que pueda parecer que "#REQUIRED" no es un valor por defecto, sin embargo la especificación de XML define las siguientes posibilidades como valor por defecto:

- #REQUIRED: si un atributo se declara así, en el ejemplar del documento se debe dar valor a todos los atributos de este tipo. No habría valor por defecto.
- #IMPLIED: los atributos que se declaran así, pueden especificarse opcionalmente en los elementos del tipo para el que se declara el atributo. No hay valor por defecto y el analizador XML pasará un valor en blanco a la aplicación para que lo cambie por su propio valor si lo desea.
- "valor del atributo": en este caso el atributo si no tiene ningún valor, irá con el que se haya especificado por defecto.
- #FIXED "valor del atributo": estos atributos no cambian su valor, están obligados a llevar este valor siempre.

Por otro lado tal y como se ha explicado anteriormente por qué no podíamos forzar a que solo exista una aparición de un elemento dentro de otro, aquí dejamos las posibles variaciones que se pueden emplear para definir la frecuencia de aparición de un elemento en XML:

- ?: indica opción, el elemento o grupo se puede repetir cero o una vez.
- +: indica una o más. Asegura una ocurrencia como mínimo.
- *: indica cero o más.

Finalmente debido a que no hemos empleado otros elementos como tokens, entidades, ids, etc, estas serían las reglas que más afectarían a nuestra especificación. Luego es obvio que el usuario o el software deberán tener en cuenta las reglas de buena formación de XML para etiquetar el diferente contenido que aparezca en el documento de una manera apropiada y de manera que el analizador XML lo considere válido, de acuerdo a las reglas de XML y de acuerdo a las reglas especificadas en nuestro DTD.

Capítulo 8

Comprobando que nuestro lenguaje es usable y válido

El objetivo de este capítulo, será exponer un ejemplo entre dos ficheros no muy extensos y mostrar aquellas relaciones que debería ser capaz de establecer el Software que implemente nuestro lenguaje de etiquetado.

Para realizar una muestra de ejemplo del potencial que tiene nuestro lenguaje de etiquetado y con el fin de dejar más claro cuál es el funcionamiento que creemos es el más adecuado para su empleo y que hay que tener en cuenta en futuras implementaciones, se va a describir un ejemplo ficticio en el que por ejemplo un profesor tuviera una colección de preguntas sobre una materia para incluirlas en un examen y unos apuntes sobre esa materia.

En este caso, no vamos a tener en cuenta ningún tipo de fichero específico puesto que eso emborronaría demasiado el ejemplo, por lo que en nuestro caso supondremos un documento de texto plano típico y se supondrá que el visor que ha empleado el profesor a la hora de rellenar este documento, oculta las marcas, de manera que para él es totalmente transparente el uso de marcas o no en ese documento. Esto desde luego sería un requisito muy a tener en cuenta para cualquier aplicación de tratamiento de textos que se implementara empleando este lenguaje.

Dado que aunque no son muchas etiquetas, si ponemos un ejemplo de todas, el documento va a ser muy difícil de leer, emplearemos algunas de las más básicas y comunes, y para la consulta sobre usos de otras etiquetas, sugerimos que acudan a los ejemplos de empleo que se expusieron en el capítulo anterior, cuando fuimos definiendo el DTD de nuestro lenguaje. Así mismo, el CD contiene todos estos ejemplos en ficheros ya hechos, para que puedan usarse, manipularse o copiar al antojo del lector.

Imaginemos que tenemos los siguientes documentos:(Definiciones extraídas de [4]

```
APUNTES DE LENGUA

Sustantivo: Parte variable de la oración que sirve para designar a
personas, animales o cosas.

Verbo: El verbo es la parte variable de la oración que indica
existencia, estado o acción.

Adjetivo: Es la parte variable de la oración que se une al sustantivo
para calificarlo o determinarlo.

Artículo: Es la parte variable de la oración que sirve para precisar
su género y número.
```

Por otro lado, imaginemos que tenemos las siguientes preguntas que tiene un profesor recopiladas aparte sobre estos temas:

```
PREGUNTAS DE EXÁMENES ANTERIORES

1. Describe lo que es un sustantivo y pon tres ejemplos.

2. Indica cuáles son las diferencias entre un verbo y un adjetivo.

3. ¿Qué tienen en común los verbos, adjetivos, sustantivos
   y artículos?

4. Define artículo y escribe un ejemplo en masculino singular.
```

Una vez que tenemos estos dos ficheros, que como podrá observar el lector, están muy resumidos para que sea más fácil, explicar cómo creemos que es más sencillo la inclusión de etiquetas y que relaciones tienen que acabar existiendo.

Ahora imaginemos que por un momento, tenemos ya un editor que realice estas operaciones de incrustado de etiquetas. Nosotros abriríamos nuestro fichero en el editor y ya de partida, el editor debería detectar dado que no hay presencia de etiquetas, que el fichero está virgen para nuestros propósitos. Por lo cual un primer paso muy recomendable que podría hacer el editor es establecer el encabezado y el nodo raíz empleando por ejemplo el nombre del fichero:

```
<?xml version="1.1" encoding="UTF-8"?>
<!DOCTYPE documento SYSTEM "general.dtd">
<documento id="definiciones.txt" nombre="Sin Nombre" tipo="texto">
APUNTES DE LENGUA

Sustantivo: Parte variable de la oración que sirve para designar a
personas, animales o cosas.

Verbo: El verbo es la parte variable de la oración que indica
existencia, estado o acción.

Adjetivo: Es la parte variable de la oración que se une al sustantivo
para calificarlo o determinarlo.

Artículo: Es la parte variable de la oración que sirve para precisar
su género y número.
</documento>


<?xml version="1.1" encoding="UTF-8"?>
<!DOCTYPE documento SYSTEM "general.dtd">
<documento id="examen.txt" nombre="Sin Nombre" tipo="texto">
PREGUNTAS DE EXÁMENES ANTERIORES

1. Describe lo que es un sustantivo y pon tres ejemplos.

2. Indica cuáles son las diferencias entre un verbo y un adjetivo.

3. ¿Qué tienen en común los verbos, adjetivos, sustantivos y
   artículos?

4. Define artículo y escribe un ejemplo en masculino singular.
</documento>
```

A partir de este momento, nuestro documento estaría preparado para incorporar nuestras etiquetas puesto que si nos fijamos, ya dispone la ubicación del DTD (ficticia en general.dtd y que se corresponderá con la ruta donde esté el DTD), y tiene declarado el nodo raíz. Ahora nos faltaría definir las partes relacionables y dentro de estas el tema sobre el que está hablando.

Dado que el usuario a lo mejor no está interesado en fijar directamente las partes relacionables, se sugiere la siguiente idea:

Lo ideal de cara a la Usabilidad de un editor de este tipo, es que por ejemplo, yo seleccione un trozo del texto (además puedo ver que estoy seleccionando, porque como en cualquier

editor de texto que se precie, me lo pintará en negro y las letras en blanco), y a partir de ese trozo de texto, mediante una opción en un menú contextual activado por ejemplo con el ratón, o bien mediante algún botón, pudiera fijar, que eso es una parte, o que eso es una etiqueta de algún tipo.

Dado que las partes, suelen tener relación con la estructura del documento, otra manera de pensar, podría fijar un nivel de detalle para que el propio editor fuera capaz de deducir cuales son las partes. Por ejemplo, podríamos definir que cada párrafo es una parte. O podríamos definir que cada sección es una parte. En nuestro caso, siguiendo un criterio parecido a estos, presupondremos que cada sección es una parte. Por lo que si consideramos a las preguntas del examen secciones unitarias y los apuntes, serán una gran sección. El aspecto del documento, una vez que se han añadido las secciones, quedaría similar a éste:

```
<?xml version="1.1" encoding="UTF-8"?>
<!DOCTYPE documento SYSTEM "general.dtd">
<documento id="definiciones.txt" nombre="Sin Nombre" tipo="texto">
<parte_relacionable id="rel1" nombre="Sin Nombre" keywords="-">
APUNTES DE LENGUA

Sustantivo: Parte variable de la oración que sirve para designar a
personas, animales o cosas.

Verbo: El verbo es la parte variable de la oración que indica
existencia, estado o acción.

Adjetivo: Es la parte variable de la oración que se une al sustantivo
para calificarlo o determinarlo.

Artículo: Es la parte variable de la oración que sirve para precisar
su género y número.
</parte_relacionable>
</documento>
```

```
<?xml version="1.1" encoding="UTF-8"?>
<!DOCTYPE documento SYSTEM "general.dtd">
<documento id="examen.txt" nombre="Sin Nombre" tipo="texto">
PREGUNTAS DE EXÁMENES ANTERIORES
<parte_relacionable id="rel1" nombre="Sin Nombre" keywords="-">
1. Describe lo que es un sustantivo y pon tres ejemplos.
</parte_relacionable>
<parte_relacionable id="rel2" nombre="Sin Nombre" keywords="-">
2. Indica cuáles son las diferencias entre un verbo y un adjetivo.
</parte_relacionable>
```

```
<parte_relacionable id="rel3" nombre="Sin Nombre" keywords="-">
3. ¿Qué tienen en común los verbos, adjetivos, sustantivos y
   artículos?
</parte_relacionable>
<parte_relacionable id="rel4" nombre="Sin Nombre" keywords="-">
4. Define artículo y escribe un ejemplo en masculino singular.
</parte_relacionable>
</documento>
```

Por ahora, no existiría ningunas relaciones entre el documento. Si bien si se hubieran rellenado las keywords, sería posible ya que el editor asociara ciertas Partes Relacionables a las del otro documento, por la coincidencia de estas keywords. Pero como en el caso, suponemos que - sería el símbolo neutro de indicar que esa parte no tiene keywords entendemos que aún no se han definido ninguna relación entre las partes del documento.

Ahora supongamos lo siguiente: El usuario marcaría seleccionado por ejemplo la definición de artículo e inmediatamente el editor le preguntaría qué es esa selección. El usuario indicaría que está hablando de un Tema, y que ese Tema trata sobre el artículo. Cabe destacar aquí, que si no queremos profundizar empleando por ejemplo el elemento Objeto para un Artículo, podemos hacerlo puesto que con el Tema ya la relación se establecería totalmente. Por lo que el editor, dejaría el fichero a algo parecido a esto:

```
<?xml version="1.1" encoding="UTF-8"?>
<!DOCTYPE documento SYSTEM "general.dtd">
<documento id="definiciones.txt" nombre="Sin Nombre" tipo="texto">
<parte_relacionable id="rel1" nombre="Sin Nombre" keywords="artículo">
APUNTES DE LENGUA

Sustantivo: Parte variable de la oración que sirve para designar a
personas, animales o cosas.

Verbo: El verbo es la parte variable de la oración que indica
existencia, estado o acción.

Adjetivo: Es la parte variable de la oración que se une al sustantivo
para calificarlo o determinarlo.
<tema id="artículo" nombre="Definición del Artículo">
Artículo: Es la parte variable de la oración que sirve para precisar
su género y número.
</tema>
</parte_relacionable>
</documento>
```

Como vemos además, el editor podría ser capaz de inferir ya ciertas keywords mientras se va escribiendo el contenido del texto. Ahora por ejemplo procedería a definir que la última pregunta del examen, tiene como tema los artículos, por lo cual quedaría así etiquetado:

```
<?xml version="1.1" encoding="UTF-8"?>
<!DOCTYPE documento SYSTEM "general.dtd">
<documento id="examen.txt" nombre="Sin Nombre" tipo="texto">
PREGUNTAS DE EXÁMENES ANTERIORES
<parte_relacionable id="rel1" nombre="Sin Nombre" keywords="-">
1. Describe lo que es un sustantivo y pon tres ejemplos.
</parte_relacionable>
<parte_relacionable id="rel2" nombre="Sin Nombre" keywords="-">
2. Indica cuáles son las diferencias entre un verbo y un adjetivo.
</parte_relacionable>
<parte_relacionable id="rel3" nombre="Sin Nombre" keywords="-">
3. ¿Qué tienen en común los verbos, adjetivos, sustantivos y
   artículos?
</parte_relacionable>
<parte_relacionable id="rel4" nombre="Sin Nombre" keywords="articulo">
<tema id="artículo" nombre="Preguntas sobre el Artículo">
4. Define artículo y escribe un ejemplo en masculino singular.
</tema>
</parte_relacionable>
</documento>
```

Ahora mismo es cuando ya se habrían establecido dos relaciones entre nuestros documentos (aunque como habrá observado el lector, contienen lo mismo). La primera, entre las partes relacionables puesto que ambas contienen como keyword la palabra artículo y nuestro editor debería ser capaz ya de reconocer, que por tanto es muy probable que estén relacionadas. Y sobretodo porque ahora existe un tema, que tiene en ambos el mismo id. Al tener en ambos el mismo id, inmediatamente el editor asociaría que la definición, está asociada con esa pregunta del examen.

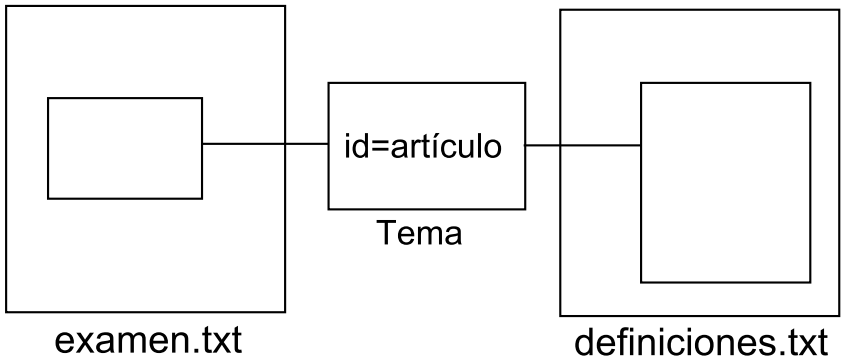

Figura 8.1: Relación entre los documentos en el tema

Otra manera de etiquetar este documento, hubiera sido definiendo que en realidad, los cuatro puntos donde se definen los diferentes elementos de una oración, son en realidad un tema llamado definiciones de palabras y por ejemplo declarara la primera aparición de sustantivo indicando que es un objeto y sus atributos:

```
<?xml version="1.1" encoding="UTF-8"?>
<!DOCTYPE documento SYSTEM "general.dtd">
<documento id="definiciones.txt" nombre="Sin Nombre" tipo="texto">
<parte_relacionable id="rel1" nombre="Sin Nombre" keywords="artículo">
<tema id="definiciones_partes_oracion" nombre="Definiciones de las
  partes de la Oración">
APUNTES DE LENGUA

<objeto id="sustantivo" nombre="Sustantivo" tipo="concepto">
Sustantivo</objeto>:
Parte variable de la oración que sirve para designar a
personas, animales o cosas.

Verbo: El verbo es la parte variable de la oración que indica
existencia, estado o acción.

Adjetivo: Es la parte variable de la oración que se une al sustantivo
para calificarlo o determinarlo.

Artículo: Es la parte variable de la oración que sirve para precisar
su género y número.
</tema>
</parte_relacionable>
</documento>
```

Si de la misma manera el usuario decidiera seleccionar que por ejemplo en la primera pregunta cuando aparece la palabra sustantivo, se está refiriendo al objeto sustantivo que ya existe en la colección, nos quedaría un etiquetado similar a este: (Para simplificar se deja solo una parte relacionable y un tema como antes).

```
\begin{verbatim}
<?xml version="1.1" encoding="UTF-8"?>
<!DOCTYPE documento SYSTEM "general.dtd">
<documento id="examen.txt" nombre="Sin Nombre" tipo="texto">
<parte_relacionable id="parte1" nombre="Sin Nombre" keywords="examen">
<tema id="preguntas_exam" nombre="Preguntas Diciembre 2010">
PREGUNTAS DE EXÁMENES ANTERIORES

1. Describe lo que es un <objeto id="sustantivo" nombre="Sustantivo">
sustantivo</objeto> y pon tres ejemplos.

2. Indica cuáles son las diferencias entre un verbo y un adjetivo.

3. ¿Qué tienen en común los verbos, adjetivos, sustantivos y
   artículos?

4. Define artículo y escribe un ejemplo en masculino singular.
</tema>
</parte_relacionable>
</documento>
```

Como vemos, se establecería automáticamente una relación entre los dos temas y las dos partes relacionables, puesto que aparece dentro de ellas un objeto con el mismo id (también el criterio podría establecerse por mismo nombre o similitud de nombre a mayores) por lo que estas dos partes, quedarían ya totalmente relacionadas y el editor debería ser capaz de indicarle al usuario que entre examen.txt y definiciones.txt existe una relación a través del id sustantivo.

Como vemos, las posibilidades son infinitas, una vez que el usuario puede colocar marcas dentro de su texto, pueden inferirse muchas relaciones bien directamente, debido a que ambas relaciones contienen los mismos elementos con los mismos ids, pero también inferirse de otras maneras, como puede ser relacionando las keywords de las partes relacionables con los ids de algunos contenidos, o con sus nombres.

Es misión del nivel que quiera emplear el Software establecer más relaciones a mayores, puesto que según la especificación de nuestro documento, **solo se relacionarán automáti-**

camente dos partes entre ellas, solo si contienen elementos similares con el mismo id o mismo nombre.

Por ello recomendamos en la implementación, que se tomen como fijas aquellas relaciones que contentan el mismo Id y sean del mismo tipo, y posteriormente el usuario a través de la herramienta, se le sugieran otras muchas o incluso pueda cambiar los criterios para establecer relaciones. Por ejemplo pueda decir, dime que partes están relacionadas, si tenemos en cuenta las anteriores, y además ese Id aparece en las keywords, o por ejemplo comparten keywords.

Para ello, lo ideal sería que se analizaran la dificultad de establecer estos mecanismos en la aplicación y en tiempo de desarrollo se decida cuales se van a proporcionar al usuario y cuales no, atendiendo a la dificultad/tiempo de implementación así como a si es realmente una funcionalidad requerida por el usuario.

Conclusiones y posibles ampliaciones

8.1. Conclusiones

XML es un lenguaje que como hemos visto nos permite de una manera muy sencilla crear nuevos lenguajes basados en él y definir un esquema para comprobar su validez de una manera inmediata. Es por eso que desde su aparición se ha considerado como una de las mejores herramientas para crear Metalenguajes y hoy en día lo podemos encontrar, desde como formato de intercambio de mensajes, como por ejemplo en peticiones AJAX, como en definiciones de una GUI como se emplea en `Android` o incluso para definir todos los puntos de entrada a los controladores de una aplicación, como es el caso de `Apache Struts` y `Spring` en `J2EE`.

Además de todas las ventajas que supone trabajar con algo que sabes que funciona, de cara a posteriores desarrollos, es muy sencillo extraer información de un XML. Existen un montón de librerías para cualquier lenguaje que permite explorar el árbol de nodos e ir extrayendo información de éstos. También es posible realizar búsquedas de información muy rápidas gracias a tecnologías como XPath, que permite a través de expresiones regulares manejar el acceso a los nodos de una manera muy simple y eficiente.

A pesar de todas estas ventajas, no debemos tampoco olvidar ciertas desventajas que posee a la hora de usarse. Una de las primeras desventajas es su tamaño. Los documentos que acaban siendo etiquetados, o que se han basado en XML, a pesar de que es uno de los estándares que se emplea para transmisión de datos, acaban siendo demasiado grandes. Si el número de etiquetas es muy grande, además el parser tiene que realizar un gran esfuerzo para sacar esta información y ser capaz de procesarla, es por ello, que en la actualidad se está empezando a movilizar la tecnología hacia otras soluciones como el empleo de `JSON`.

Por otro lado, aunque trae mecanismos que permiten saber si un documento es válido, es imprescindible que el propio lenguaje traiga una serie de recomendaciones y restricciones

implementadas mediante Software, debido a que XML no es capaz de restringir el uso de las etiquetas a muy bajo nivel. Uno de los problemas por ejemplo, es que podemos especificar si un documento puede contener cero o más etiquetas (*), una o más etiquetas (+), o ninguna etiqueta, pero sin embargo, no podemos fijar un límite estableciendo que en el documento solo puede aparecer una etiqueta hija. Esto en ocasiones, como hemos visto con las coordenadas, o como hemos visto con el nacimiento y muerte de una persona, provoca incoherencias, dado que el XML puede estar bien formado y ser válido por el DTD, y sin embargo no corresponderse con lo que nosotros deseamos emplear.

Otra desventaja en este aspecto es por ejemplo el empleo de los IDs o de las ENTITY, que permiten realizar una validación y una especificación mucho más profunda. Por un lado es cierto que pueden ser muy útiles cuando uno está etiquetando un documento XML, pero cuando se está definiendo como en este caso, de una manera muy general, no se puede realmente extraer un potencial muy grande de ellos. Por ejemplo, en nuestro caso no hemos podido emplear ID en los Ids de nuestros elementos, debido a que nuestros elementos podían aparecer varias veces en el documento refiriéndose a lo mismo y las reglas de XML establecen que un XML no es válido si aparece el mismo ID repetido a lo largo de un documento, tiene que ser único. Las entidades por ejemplo son de gran utilidad empleando ficheros externos o sabiendo la referencia al Software que tiene que procesar un documento, puesto que dan mecanismos para establecer estos parámetros dentro del documento, pero en nuestro caso, dado que son elementos dinámicos, no podemos hacer uso de ellas.

Es por ello que podemos concluir que XML es un excelente mecanismo para realizar lenguajes o emplearse a la hora de guardar información y de procesarla en cualquier tipo de aplicación en el que dispongamos de un tiempo y un espacio razonable, siempre teniendo en cuenta las restricciones y limitaciones que hemos venido contando hasta este punto.

8.2. Ampliaciones

Como hemos visto, a la hora de clasificar los diferentes elementos que van a generar relaciones dentro de nuestros textos, hemos realizado un enfoque generalista, en el que con unas pocas etiquetas pretendemos ser capaces de clasificarlo todo. Este enfoque puede funcionar en una inmensa mayoría de casos, pero en otros muchos casos, sobretodo de personas que trabajen en ámbitos muy especializados, es probable que por el tipo de lenguaje y conceptos que manejan, no sea un enfoque suficiente.

Por esto, es por lo que una de las posibles ampliaciones que sería posible realizar, es la adición de nuevas etiquetas al lenguaje que permitan delimitar mejor aún los elementos que ya tenemos. Por ejemplo si estamos en un ámbito de informática, podrían surgir etiquetas como: Sistema Operativo, Lenguaje de Programación, Librería...Si estuviéramos dentro de un ámbito por ejemplo de literatura, podrían surgir: Novela, Poema, Género Literario, etc...

No se estiman más ampliaciones que esas, puesto que como el fin último es que finalmente se implemente una aplicación sobre este lenguaje, probablemente durante el desarrollo de esa aplicación, se observarán algunas carencias o mejoras que pueden facilitar la vida al programador y que quizás en este caso no hemos tenido en cuenta.

8.3. Datos de interés

Para la realización de este proyecto se ha empleado:

- Texmaker 3.0.2
- Inkscape 0.48
- TortoiseSVN 1.6.5
- Dropbox
- Google Chrome 19
- Gvim

Escrito en LaTeX

Agradecimientos

Al igual que en cualquier libro que se precie, me gustaría agradecer su ayuda a las siguientes personas, que me han hecho la vida más fácil:

- A mis padres, por encargarse que no hubiera un día que no recordara que tenía que escribir este trabajo y por todas las facilidades que me han dado a lo largo de la vida para estudiar lo que me gusta y disfrutar haciendo lo que hago.
- A mis abuelos, puesto que después de tanto recorrido que yo no tengo en esta vida, nunca dejo de aprender cosas nuevas y de ver cosas que debo aprender para caminar en ella con soltura. Y por supuesto al resto de mi familia, tíos, primos, etc...
- A mis amigos, pero en especial a los señores *Pablo Díez* y *Luís González*, porque siempre que quieras pasar unos ratos locos y divertidos, ellos siempre están disponibles para alegrarte el día.
- A mis compañeros de trabajo (tanto los de *Thales Information Systems* aka *Connectis*, como a los de *Safran Engineering Services*), puesto que siempre me han animado a hacerlo y sus consejos han sido muy valiosos para la realización de este trabajo.
- Y finalmente a *César Llamas Bello* porque a pesar de que ambos tenemos unos horarios totalmente incompatibles, hemos logrado hallar la manera de estar comunicados y lo que es más importante, éste ya es el segundo proyecto de estas características que llevamos a cabo. César, me lo paso muy bien haciéndolos, pero ¡Esperemos que sea ya el último!;)

Contenido del CD

Carpeta	**Contenido**
`ejemplos/`	Contiene los ejemplos del lenguaje aparecidos en la memoria.
`lenguaje.dtd`	El DTD para validar nuestro lenguaje.
`memoria.pdf`	Contiene la memoria del proyecto.

Glosario

AJAX

(Asynchronous Javascript And XML) Tecnología que se basa en el uso de realizar peticiones asíncronas de XML a través de HTTP, de tal manera que no necesitamos cambiar página web para obtener contenido. 40

Android

Popular sistema operativo para móviles desarrollado y mantenido por Google. 89

Apache Struts

Popular framework para Web en J2EE que provee de un eficaz MVC. 89

API

(Application Programming Interface) Es el conjunto de funciones que nos proporciona un lenguaje de programación o una librería. 48

DTD

Document Type Definition (Definición de tipo de documento), es una descripción de la estructura y sintaxis de un documento XML con el que podemos comprobar que es válido. 9

HTML

HyperText Markup Language (Lenguaje de Marcado de Hipertexto), es el lenguaje de marcado predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. 39, 45

ISO

International Organization for Standardization (Organización Internacional para la Estandarización) es la encargada de promulgar estándares de calidad para que los fabricantes los adopten y por lo tanto intentar mantener una calidad en sus productos. 13

J2EE

Java for Enterprise Edition. Versión para entornos empresariales de la popular distribución de la máquina virtual de Java. 89

JSON

(JavaScript Object Notation) Especificación con la cual se pueden transmitir objetos con sus propiedades y métodos a través de cualquier protocolo para ser empleado en Javascript. 89

Multimedia

Que utiliza conjuntamente diversos medios a la vez como puede ser texto, imágenes, sonido, etc. 11

plugins

Pequeños programas que podemos acoplar a un sistema de manera que extienden su funcionamiento. 14

SGML

Standard Generalized Markup Language (Estándar de lenguaje de marcado generalizado) es uno de los primeros metalenguajes de marcas que se emplearon, pero que debido a su complejidad ha sido poco empleado y ampliamente sustituido por algunos de sus hijos como HTML o XML. 39

Spring

Popular framework para J2EE que puede convivir junto a Struts y se emplea sobretodo por su novedoso concepto de inyección de dependencias. 89

Tesauros

Es una lista de palabras con significados similares sinónimos, habitualmente acompañada por otra lista de antónimos. Un ejemplo sería un tesauro dedicado a un campo especializado, que contiene la jerga que se emplea en dicho campo del conocimiento. 67

Tokens

Son cadenas de caracteres que tienen un significado coherente en cierto lenguaje de programación o en cierta parte de un sistema. 13

Unicode

Estándar de codificación de caracteres que abarca una gama de caracteres muchísimo más amplia que el Ascii. 40

Vi

Popular editor de texto de los sistemas UNIX. Destaca por sus grandes posibilidades para emplear macros y comandos sobre él. 12

W3C

(World Wide Web Consortium)Es una comunidad internacional que desarrolla estándares que aseguran el crecimiento de la Web a largo plazo. 39

Windows

Microsoft Windows es una serie de sistemas operativos desarrollados por Microsoft desde 1981, año en que el proyecto se denominaba Interface Manager. Tuvo gran éxito debido a su interfaz de ventanas. En la actualidad es uno de los sistemas más usados por usuarios domésticos, pero también de los más criticados por fallos de seguridad. 12

WYSIWYG

Acrónimo de What You See Is What You Get. Se refiere a aquellos editores de texto o de gráficos, donde lo que vemos en pantalla es el resultado final que obtendremos al imprimir o generar el documento. 15

XML

Extensible Markup Language (lenguaje de marcas extensible), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML). 39

Bibliografía

[1] Raúl Martínez Abraham Gutiérrez. *XML a través de ejemplos*. Ra-Ma, primera edición edición, 2001. Libro con gran cantidad de ejemplos del lenguaje. Es muy útil y nos ha servido de cara a ver como estructurar nuestro lenguaje de etiquetado, comparando con los posibles ejemplos que aparecían en el libro.

[2] Jose Miguel Esparza. Ocultando información en un pdf, 2009. URL `http://blog.s21sec.com/2009/03/ocultando-informacion-en-un-pdf.html`. Interesantísimo artículo en el que se habla de las diferentes maneras que es posible incorporar texto oculto a un documento PDF. Es una de las maneras que hemos propuesto para incorporar nuestras etiquetas dentro de este tipo de formatos.

[3] Google. Youtube apis and tools, 2012. URL `http://gdata.youtube.com/demo/index.html`. Descripción y uso del API de YouTube documentada por Google. De aquí se ha extraído el formato de mensaje con el que trabaja YouTube con el fin de estudiar su estructura y ver en que partes podía servirnos para la incorporación de medios continuos en nuestro lenguaje de etiquetado.

[4] La Oración Gramatical. Gramática: La oración, análisis y clasificación, 2006. URL `http://www.actiweb.es/laoraciongramatical/partes_variables_de_la_oracion.html`. Descripción de las partes de la oración. He escogido esta porque era la que más me recordaba a las definiciones que me daba mi profesor cuando era pequeño en la materia de Lengua.

[5] TeX Users Group. History of tex, 2011. URL `http://www.tug.org/whatis.html`. Historia de como surge el TeX (padre de LaTeX) a través de su más conocida comunidad de usuarios.

[6] Elliotte Rusty Harold. *XML Bible*. IDG Books, first edición, 1999. Aunque un poco anticuado, trae absolutamente todo de la primera versión que se sacó de XML, es muy útil de cara a buscar cosas muy específicas que a veces no es posible encontrar en los libros de iniciación a este lenguaje.

[7] David Hunter *et al. Iniciación a XML*. Inforbook's Ediciones, primera edición, 2000. Libro muy básico pero muy útil de cara a iniciarse en el mundo de XML, trae de manera

básica todas las maneras en los que uno puede adoptar emplear XML en diferentes modelos de negocio y como sacarle partido en tus propios desarrollos.

[8] IBM. Xml 1.1 and namespaces 1.1 revealed, 2004. URL `http://www.ibm.com/developerworks/xml/library/x-xmlns11/index.html`. Documento de IBM orientado hacia desarrolladores que emplean XML para crear nuevos espacios de nombres dentro de documentos XML. En este documento nos hemos basado a la hora de encontrar las maneras de insertar las etiquetas del formato OpenDocument. (Basado en XML). Además ha sido una buena base para plantear mecanismos de expansión dentro de nuestro lenguaje de etiquetas.

[9] Adobe Systems Inc. *PDF 32000-1:2008*. ISO, first edition edición, 2008. Especificación de la propuesta que realizó Adobe al organismo ISO para que PDF pudiera ser estándar. Se han extraído de este documento las nociones elementales de como funciona el PDF por dentro, con el fin de poder elaborar de una manera fiable, las diferentes posibilidades de inclusión de marcas dentro del propio fichero.

[10] Adobe Systems Inc. Adobe-pdf:the global standard for trusted,electronic documents and forms, 2011. URL `http://www.adobe.com/pdf/about/history/`. Entrada dentro de la propia Adobe donde cuenta como fue el nacimiento e historia de su formato más conocido, el PDF.

[11] Lauren Leurs. La historia del pdf, 2001. URL `http://www.gusgsm.com/historia_pdf`. Pequeña historia de como a lo largo del tiempo surgió el PDF y acabó desarrollándose y evolucionando a lo largo de sus diferentes versiones.

[12] C. Linneo. *Systema Naturae, sive regna tria naturae, systematics proposita per classes, ordines, genera, species*. Theodorum Haak, 1735. Libro que publicó Linneo en 1735 donde aparecería por primera vez el popular mecanismo para clasificar seres vivos basado en Dominios y Reinos. Aunque obviamente no se ha leído para el trabajo, dado que es la matriz de donde procede nuestros atributos del sistema de etiquetado, lo citamos como bibliografía.

[13] Antonio Gabriel López. Latex avanzado: Crear nuevos comandos y colores, 2009. URL `http://hallsi.ugr.es/cursoLatex/cursoLaTeX_4_1.pdf`. Documento donde habla como emplear LaTeX de manera avanzada, extendiéndolo y haciendo posible la creación de comandos propios que en nuestro caso podrían ser empleados para insertar las marcas dentro de los documentos de una manera sencilla.

[14] Sun Microsystems. Open document format - open systems for open minds, 2005. URL `http://xml.gov/presentations/sun/odf.pdf`. Documento de Sun Microsystems donde habla de todos los beneficios que tiene el formato Open Document para el futuro.

[15] Frank Mittelbach *et al. The Latex companion*. Addison-Weasley, second edition edición, 2004. Libro de referencia para aprendizaje y uso de LaTeX. Se han extraído de este documento las diferentes posibilidades contempladas a lo largo de este PFC para introducir marcas dentro de documentos LaTeX. También se investigó la posibilidad de

insertar marcas dentro del propio PDF generado a través del postscript intermedio que permite generar LaTeX.

[16] OASIS. History of opendocument, 2008. URL http://opendocument.xml.org/milestones. Historia del nacimiento y las diferentes evoluciones que ha sufrido a lo largo de los años, el formato OpenDocument.

[17] Latex Project. Latex - a document preparation system, 2011. URL http://www.latex-project.org. Página principal de LaTeX donde se puede encontrar información sobre su estructura, documentación para emplearlo, así como su historia y otros datos de interés.

[18] Committee Specification1. *Open Document Format for Office Applications (OpenDocument) v1.0*. OASIS, second edition edición, 2006. Especificación entregada por OASIS a la ISO como propuesta de estándar para el formato OpenDocument. Se han extraído todas las posibilidades de incluir marcas de manera interna dentro de esta clase de documentos, además de para realizar un estudio estructural del mismo.

[19] Stefan Sundin. How to copy youtube annotations, 2011. URL http://stefansundin.com/blog/277. Artículo que muestra como copiar las anotaciones de los videos de YouTube, a otro video nuestro. De aquí se ha extraído el conocimiento necesario para recuperar ese formato XML en el que se basan las anotaciones y poder realizar su estudio.

[20] Erik T.Ray. *Learning XML*. O'Reilly, second edition edición, 2003. Otro de los libros de XML en el que nos hemos basado, es mucho más completo que los de iniciación básica, pero llega al nivel de profundidad suficiente como para que su tamaño sea tan desmesurado como para encontrar cosas. Este ha sido probablemente uno de los pilares fundamentales en el aprendizaje de XML.

[21] W3C. Extensible markup language (xml) 1.1 (second edition), 2006. URL http://www.w3.org/TR/xml11/. Especificación de la versión 1.1 de XML. Nos ha servido como consulta para realizar el lenguaje de etiquetado, así como para ver las diferencias existentes entre esta versión y su versión anterior, la 1.0.

[22] Dr John Warnock. The camelot paper, 1991. URL http://www.planetpdf.com/enterprise/article.asp?ContentID=6519. Documento histórico en el que se habla por primera vez del proyecto matriz que acabaría desembocando en el nacimiento del PDF.

[23] Wikibooks. Labels and cross referencing, 2011. URL http://en.wikibooks.org/wiki/LaTeX/Labels_and_Cross-referencing. Describe el uso de las etiquetas dentro de LaTeX y ha sido uno de las páginas estudiadas para incorporar estas marcas a LaTeX a través de otra manera diferente que no fueran los comandos.

[24] Wikipedia. Compiler, 2011. URL http://en.wikipedia.org/wiki/Compiler. Descripción de las diferentes fases de un compilador, descritas en su entrada de la Wikipedia.

[25] Wikipedia. John warnock, 2011. URL `http://en.wikipedia.org/wiki/John_Warnock`. Pequeña entrada sobre la historia del padre del PDF, además contiene también detalles sobre la historia y como llegó a crearse.

[26] Wikipedia. Open document, 2011. URL `http://en.wikipedia.org/wiki/OpenDocument`. Entrada de la wikipedia sobre el Open Document, donde cuenta la historia del formato, especificación, etc.

[27] Wikipedia. Portable document format, 2011. URL `http://en.wikipedia.org/wiki/Portable_Document_Format`. Entrada en la Wikipedia inglesa que contiene datos y la historia sobre el PDF.

[28] Wikipedia. Extensible markup language, 2012. URL `http://es.wikipedia.org/wiki/Extensible_Markup_Language`. Entrada principal de la Wikipedia donde se habla del XML. Aquí aparece su historia, estructura básica y algunos de los lenguajes de etiquetas más famosos en los que ha desembocado XML.

[29] Wikipedia. Tesauro, 2012. URL `http://es.wikipedia.org/wiki/Tesauro`. Explicación breve y detallada de como funcionan los Tesauros. Se han propuesto los Tesauros como un mecanismo complementario a emplear dentro de las palabras clave que emplea nuestro lenguaje de etiquetado.