Universidad
Rey Juan Carlos

ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

INGENIERÍA INFORMÁTICA

Curso académico 2011-2012

Proyecto de fin de carrera

Plataforma de seguimiento on-line del nivel de autoaprendizaje de los estudiantes: Adaptación de documentos y generación de una base de datos.

Autor: Miguel Ángel Jimémez Sánchez

Tutor: Jose Centeno González

A mis padres,

y a mis amigos...

Agradecimientos

La exposición de un proyecto de fin de carrera puede ser considerado como el fin de una etapa y el comienzo de otra. A lo largo de todos los años de universidad he podido adquirir numerosos conocimientos tanto académicos como humanos, que han permitido evolucionar como persona.

Quisiera aprovechar la oportunidad que se me brinda para agradecer a todas esas personas que me han apoyado durante este periodo de mi vida. En primer lugar como no podía ser de otra manera agradecer a mis padres todo su apoyo, tanto en los momentos buenos como en los más difíciles, donde nunca me han fallado, además de darme la oportunidad de elegir mi futuro.

En segundo lugar, quisiera agradecer a Jose Centeno, mi tutor, permitirme realizar este proyecto, siempre que he tenido alguna duda ha tratado de orientarme de la mejor manera posible y ha estado pendiente del proyecto en los momentos decisivos. Gracias también a Pedro de las Heras y Eva Castro por acompañarme en las reuniones donde exponía la evolución del proyecto. Debo agradecer ese buen ambiente en todas nuestras reuniones que me invitaba a continuar adelante.

Por último, pero no por ello menos importante, a todos amigos que siempre han confiado en mí, por su apoyo y amistad que te da la fortaleza necesaria para afrontar los momentos difíciles. En especial a Mónica por aportarme algo más que confianza y apoyo.

Resumen

Este proyecto surge con la intención de ofrecer a los alumnos un nuevo sistema que permita mejorar su capacidad de autoaprendizaje y que sirva de apoyo a su desarrollo académico. Para llevar a cabo esta empresa ha sido necesario realizar un análisis de los métodos de evaluación actuales, considerando, tras el análisis, la necesidad de realizar un sistema de autoevaluación on-line. Este sistema debe permitir una realimentación tanto para el alumno como para el profesor, de manera se mejore la experiencia docente.

Se utilizan los materiales de años anteriores para crear una base de datos estructurada, de esta manera estos datos se utilizarán para crear nuevos ejercicios de autoevaluación. Para realizar el proceso de adaptación de los documentos antiguos al nuevo sistema se ha creado un estándar que permite unificar documentos heterogéneos. Este estándar se introduce en los documentos creando en ellos una estructura analizable. Al mismo tiempo se ha creado una herramienta para analizar este estándar e introducir sus datos en la base de datos.

Una vez desarrollado el analizador, al tratarse de una herramienta on-line, este ha sido integrado en un sistema web. Este sistema web además de realizar el análisis utilizando la herramienta desarrollada proporciona otras utilidades: como facilitar la inserción del estándar en documentos de evaluación o verificar que un documento esté correctamente estructurado con el estándar. Al mismo tiempo se encarga de mantener la base de datos generada tras el análisis de los diversos documentos.

A lo largo del desarrollo se han realizado diversas pruebas con distintos documentos de entrada que han permitido verificar el correcto análisis de los documentos. Utilizando estos datos almacenados durante las pruebas se han generado nuevos archivos entrelazando de manera aleatoria los datos de los documentos de entrada. Este proceso será un punto de partida de la segunda parte del sistema de autoevaluación on-line.

Índice general

Índice de figuras

Capítulo 1

Introducción

Hoy en día las nuevas tecnologías se encuentran presentes en todos los ámbitos de la vida cotidiana. La docencia es un ámbito donde se debe aprovechar este desarrollo de las tecnologías, de forma que permita innovar en los métodos de estudio y en los métodos de evaluación. [1]

[1] Este proyecto forma parte del trabajo desarrollado en el Proyecto de Innovación Docente: Seguimiento on-line del nivel de autoaprendizaje de los estudiantes en asignaturas impartidas por el área de Ingeniería Telemática

1.1. Motivación del proyecto

El presente proyecto surge del interés mostrado por parte del personal docente del departamento de Sistemas Telemáticos y Computación de la Universidad Rey Juan Carlos por desarrollar un conjunto de herramientas on-line que permita a los alumnos comprobar su nivel de aprendizaje en las asignaturas impartidas por este departamento. Estas herramientas ofrecerán pruebas de autoevaluación similares a las que los profesores utilizarán en las asignaturas como parte del proceso de evaluación. Debido a su carácter on-line, los alumnos podrán realizarlas en el momento en que consideren que se encuentran suficientemente preparados, obteniendo como resultado información sobre su nivel de aprendizaje, lo que repercutirá en una mejora del proceso continuo de su formación. El sistema también generará información valiosa para el profesor que le permitirá a éste analizar qué competencias y habilidades de la asignatura precisan de un esfuerzo extra tanto por parte de los alumnos como del profesor para mejorar la docencia.

Se pretenden alcanzar una serie de objetivos docentes, enumerados a continuación:

- **Favorecer el aprendizaje autónomo del estudiante** en las asignaturas que participarán en este proyecto. Normalmente, un alumno utiliza la documentación de teoría y prácticas, la cual se encuentra disponible en la página web de la asignatura para estudiar los contenidos de la misma. Adicionalmente, el alumno dispone de los enunciados tipo test y las soluciones de las pruebas de evaluación continua de años anteriores. Cada alumno lleva un ritmo de estudio diferente dentro de una asignatura. Por este motivo, el desarrollo de una herramienta on-line, permitirá que realicen su evaluación en el momento que se consideren suficientemente preparados. Los alumnos obtendrán de las herramientas que se desarrollarán una explicación de los resultados obtenidos en la evaluación con el objetivo de que entiendan las razones de los errores que hayan cometido y comprendan mejor los contenidos sobre los que han realizado dicha prueba. Hasta ahora, el alumno sólo podía disponer de esta explicación si el profesor se encontraba con él en el momento de realizar los ejercicios, mientras que a partir de este proyecto el alumno podría disponer de esta información aunque se encuentre realizando los ejercicios en su casa o en las aulas de prácticas fuera del horario lectivo.

- **Mejorar el rendimiento académico de los estudiantes**, de forma que

  disminuyan la tasas de fracaso y abandono. Existe un colectivo de alumnos que tiene serias dificultades para asistir a clase y a tutorías presenciales. Estadísticamente el rendimiento académico de estos alumnos es inferior al del resto de los alumnos que si que utilizan la asistencia presencial y asisten a las tutorías.

  Incluso aunque estos alumnos dispongan de todos los materiales didácticos publicados en la página web de las asignaturas, la falta de las explicaciones del profesor y sus aclaraciones a los fallos en los ejercicios introducen una seria dificultad en su aprendizaje.

  Por otro lado, incluso entre los alumnos que asisten siempre a clase, tienden a realizar los ejercicios propuestos hacia el final del proceso de aprendizaje, en una ubicación temporal muy próxima a la de su evaluación presencial, lo que dificulta enormemente el proceso de evaluación continua. Cuando el alumno se enfrenta a estos ejercicios se encuentra normalmente estudiando en su casa y no dispone del profesor al que consultarle la razón por la que es incorrecta la respuesta que él ha dado a una cuestión.

  Por tanto se espera que con las herramientas que se desarrollarán en este proyecto el alumno pueda tener realimentación inmediata de su aprendizaje, con lo que se verá mejorado su rendimiento académico.

Con el desarrollo de estas herramientas se espera que los alumnos obtengan una serie de beneficios:

- **Incremento de la motivación y aumento de la implicación de los alumnos en el proceso de aprendizaje** gracias al uso de los nuevos materiales de aprendizaje autónomo que se generarán debido a las herramientas a desarrollar en el proyecto.

- **Mejor aprovechamiento, por parte del alumno, del tiempo** dedicado al estudio y trabajo personal, al permitir al alumno continuar el proceso de aprendizaje fuera del aula, realizando ejercicios similares a las pruebas de evaluación continua

que se realizan a lo largo del curso. Dichos ejercicios permitirán que cada alumno pueda conocer los errores cometidos y obtener una explicación inmediata de los mismos. Este nuevo entorno mejora sustancialmente la experiencia de aprendizaje con respecto a un entorno donde no se tiene acceso rápido a una explicación de los errores que se han cometido.

- **Modificación de la metodología docente utilizada en las partes del temario que resultan más complicadas para el alumno**. Al recoger el sistema, del cual se pretende desarrollar la información de los resultados obtenidos por los alumnos en cada uno de los temas, el profesor puede conocer aquellos aspectos del temario que resulten más difíciles en el proceso de aprendizaje del alumno. El profesor

  podrá adaptar la metodología utilizada en la impartición de dichos contenidos para que los alumnos puedan superar sus dificultades.

1.2. Estado del arte

Para la creación de estas herramientas on-line será necesario realizar un análisis sobre las técnicas y herramientas actuales que permiten realizar este tipo de tareas o tareas similares.

El sistema de evaluación on-line debe contener una serie de pruebas de evaluación que se presentarán a los alumnos. Estas pruebas tendrán su origen en pruebas evaluativas y prácticas realizadas en cursos académicos anteriores, por lo que se debe estudiar cuales son los métodos que facilitarán la tarea de adaptar estos contenidos al servicio on-line.

Al mismo tiempo, ya que estos documentos se encuentran escritos en formato LaTeX, será necesario estudiar como manejarlos y que herramientas existen en caso de crear un estándar basado en XML.

Por lo tanto, las principales tecnologías y aplicaciones analizadas tienen como objetivo el manejo de documentos científicos, tanto en formato LaTeX como XML, y, además, se han analizado técnicas de autoevaluación y ayuda al aprendizaje de forma que permitiera comprender cuáles son los problemas actuales y cuál puede ser una buena forma de

evolucionar la docencia. Conocer estas herramientas, técnicas y tecnologías es clave para el desarrollo posterior del proyecto.

1.2.1. Sistemas de autoevaluación y ayuda al aprendizaje

El término autoevaluación consiste en valorar uno mismo su propia capacidad, así como la calidad del trabajo realizado, en especial en el campo pedagógico.

Un sistema de autoevaluación dinámico trata de generar en un usuario un interés por evolucionar sus conocimientos en algún ámbito. Para lograr este objetivo centramos la atención del usuario sobre una serie de preguntas o problemas completos, para los cuales se facilitan una multitud de opciones posibles. Entre estas opciones sólo una es la correcta descartando las demás como erróneas (cuando el usuario escoge una pregunta errónea el sistema debe proporcionar una respuesta ante este evento). En general, cuando un usuario no responde correctamente ante una pregunta, el sistema trata de incentivar el aprendizaje aportando una pista de cuál puede ser la correcta y de por qué la respuesta elegida no es válida.

En la docencia tradicional a un estudiante se le solicita resolver una serie de problemas de mayor o menor complejidad, con mayor o menor contenido teórico, pero todos estos problemas tienen en común la necesidad de un profesor encargado de su corrección. Una propuesta de cambio o evolución puede ser la autocorrección o corrección automática de los problemas a los que el estudiante o alumno se ha enfrentado. Esto permitiría al profesor dedicarse a la creación de nuevos retos o preguntas con las que desafiar los conocimientos del alumnado.

Este sistema permite por un lado un posible sistema de logros con el que incentivar al alumnado. El propio alumno recibe realimentación instantánea ante su trabajo: correcciones instantáneas, explicaciones ante respuestas erróneas, etc. Por otro lado al estar centrado en un sistema software sería posible crear un sistema de evaluación clasificado por asignatura, tipo de preguntas, nivel de dificultad, etc. Además de la posibilidad de proponer preguntas de manera aleatoria para evitar que dos alumnos intercambien únicamente el número de la pregunta y su respuesta correcta. Tras lo expuesto anteriormente, podemos observar que el modelo deriva en un sistema de

evaluación personalizado para cada alumno donde el profesor únicamente se ha encargado de mantener el sistema de evaluación con los contenidos apropiados.

Este proyecto está centrado en la creación de una base de datos sólida, donde poder almacenar todos los datos que este sistema de evaluación dinámico necesitaría. Durante años de docencia se han acumulado una serie de documentos en formato estático que ahora pueden ser aprovechados en el nuevo sistema dinámico, para ello es necesario desarrollar un software que adapte estos documentos a nuestro nuevo sistema de la forma más sencilla posible. En cuanto al sistema de evaluación y autocorrección quedará relegado a un segundo trabajo, como se comenta en la sección de trabajos futuros del presente documento.

1.2.2. Herramientas de transformación de documentos LaTeX

Como se ha mencionado en el punto anterior es necesario añadir datos a un sistema de evaluación. Estos datos pueden encontrarse en diversos formatos; en el presente proyecto nos centraremos en el caso particular de documentos almacenados en formato LaTeX aunque también se posibilitará la opción de insertar documentos en texto plano.

LaTeX

LaTeX es un sistema de composición de textos, orientado especialmente a la creación de libros, documentos científicos y técnicos que contengan fórmulas matemáticas. Por ello ha sido el sistema preferido durante años para editar contenidos en el ámbito académico.

LaTeX nace en 1984, con la intención de facilitar el uso del lenguaje de composición tipográfica TeX. Su gran extensión dentro del ámbito académico se debe a la alta calidad de los documentos generados al utilizarlo. Mediante LaTeX se pueden escribir todo tipo de documentos: artículos académicos, tesis o libros técnicos, con una calidad equiparable a cualquier editorial científica.

Como se ha mencionado anteriormente el formato LaTeX es un lenguaje de macros de composición tipográfica. El contenido o información se encuentra dentro de determinadas órdenes o macros. Es por tanto necesario interpretar el documento para abstraer el contenido de cómo se representa esta información. Una vez se consiga abstraer únicamente

el contenido se podrá representar posteriormente en otros muchos formatos: HTML, DOC, LaTeX XML, texto plano.

Es en este punto donde existen múltiples herramientas que permiten interpretar un documento escrito en LaTeX y generar otro documento en otro formato. A lo largo del proceso de análisis sobre el estado de estas herramientas se han tenido en cuenta una serie de criterios. El principal objetivo es convertir el documento escrito en LaTeX en un formato procesable de forma sistemática, preferentemente HTML + MathML o, más generalmente, a XML. Además de cumplir el requisito anterior se han tenido en cuenta otra serie de requisitos que se exponen a continuación:

- La herramienta debe ser libre y de código abierto.
- La herramienta debe producir un documento correcto en XHTML + MathML o XML como salida.
- La herramienta debe ser capaz de manejar las definiciones de macros mediante comandos estándar de LaTeX. Cuantas menos restricciones imponga mejor será calificado.
- Se considerará una ventaja la posibilidad de añadir soporte para paquetes adicionales de LaTeX (por ejemplo, natbib o hyperref).
- Las herramientas deben requerir escasa o nula intervención por parte del usuario, preferentemente deben poder ser utilizadas ejecutando un Script.
- Las herramientas deben tratar el documento sin realizar modificaciones en él.

Es relevante destacar la gran cantidad de herramientas que se pueden encontrar cumpliendo buena parte de estos requisitos. Algunas de estas herramientas son las siguientes:

- *LaTeXML:* LaTeXML es un módulo escrito en Perl que analiza el documento LaTeX y emite una salida XML para su posterior post-procesado (por ejemplo, para la conversión a XHTML + MathML). Con una adecuada hoja de estilo XSLT, se

puede obtener un documento en XHTML + MathML como salida. El proyecto fue creado en el 2004 y la ultima versión (0.7.0) es del año 2011 y está bien documentado.

- *Tralics:* Tralics está escrito en C++ y también directamente analiza el código fuente de LaTeX (es igualmente muy rápido). El archivo XML que genera se puede convertir a XHTML + MathML usando una hoja de estilos. Esta herramienta, que se ejecuta desde consola, tiene una gran cantidad de argumentos de entrada con lo que le aporta versatilidad. Está licenciado bajo la marca francesa CeCILL licencia de código abierto que es compatible con la GPL.

- *Tex4ht:* Tex4ht, disponible en el paquete Debian tex4ht, es probablemente la herramienta más utilizada que transforma LaTeX a HTML. Soporta la conversión a HTML, XHTML + MathML, OpenDocument, y DocBook.

  La conversión directa a XHTML + MathML parece un gran progreso, pero la salida no es clara y su interpretación puede llegar a ser complicada. Parece que es posible personalizar mucho la salida si se desea, pero los métodos para hacerlo no son precisamente obvios.

- *LXir:* LXir es un analizador de documentos DVI, por tanto tiene ciertos requisitos para analizar documentos LaTeX. En primer lugar, se debe incluir el paquete de LXir en el documento LaTeX para obtener un archivo DVI. A continuación, se debe ejecutar LXir sobre el archivo DVI y se producirá e archivo XML.

  LXir parece prometedor, pero todavía tiene algunos problemas. Actualmente, tiene ciertos errores si se encuentra con comandos u órdenes de paquetes no soportados, y también produce errores cuando trata de analizar fórmulas matemáticas, muy comunes en documentos escritos en LaTeX. En general es un proyecto con proyección pero no parece preparado para su uso en producción.

- *Hermes:* Hermes es una gramática basada en un analizador de archivos DVI para la traducción de LaTeX a Unicode con codificación XML + MathML. Es necesario incluir un conjunto de macros de TeX en el documento original para generar el documento DVI. A continuación, construye salida XML sobre el análisis del archivo DVI.

Hermes es muy completo en términos de funcionalidad, pero todavía mantiene ciertos problemas técnicos o entre otros, que tiene problemas para manejar adecuadamente los espacios. También se requiere de dos pasos para obtener el archivo XML: primero incluir una serie de macros en el documento original y después leer el documento DVI con lo que se convierte en una herramienta complicada para su uso automatico.

Tras este breve repaso sobre las tecnologías actuales de transformación de documentos escritos en LaTeX a XML, se puede adelantar que la herramienta elegida ha sido Tralics. En un capítulo posterior se realizará un análisis más exhaustivo de la herramienta así como cuál ha sido su uso.

1.2.3. Herramientas para el manejo de documentos XML

En el presente proyecto se trata de realizar un proceso de conversión automático, con la finalidad de transformar un documento LaTeX en XML y almacenarlo en una base de datos. Como se ha visto en el punto anterior, podemos obtener un documento en formato XML desde otro escrito con macros de LaTeX utilizando una herramienta de procesado, pero sigue siendo necesario realizar una lectura de este documento XML para obtener el contenido final. Además del contenido propiamente dicho será necesario tratar cada contenido en su contexto, es decir, obtener la semántica del texto: qué es o qué tipo de información contiene una determinada parte del documento. Esto será ampliado en capítulos posteriores.

XML

XML, cuyo nombre procede de las siglas en inglés *eXtensible Markup Language*, es un metalenguaje de etiquetas desarrollado por el *World Wide Web Consortium* (W3C). Es una simplificación y adaptación del SGML[2], que permite definir la gramática de lenguajes específicos. Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades, de ahí que se le denomine

[2]Standard Generalized Markup Language

metalenguaje. Algunos de los lenguajes que utilizan XML para su definición son XHTML, SVG o MathML.

XML no esta dirigido únicamente a su uso en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Este será el uso que se aprovechará en este proyecto. Adicionalmente, se puede usar en bases de datos, editores de texto, hojas de cálculo, etc.

XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Actualmente tiene una amplia relevancia y es utilizado en muchos ámbitos para el intercambio de información.

A lo largo de este proyecto se utilizarán librerías especializadas en leer o modificar documentos en XML.

- *DOM:* DOM, cuyo nombre procede del acrónimo inglés *Document Object Model*, es un lenguaje de convención independiente y multiplataforma para representar e interactuar con objetos representados en HTML, XHTML y XML. Es un estándar creado por el *World Wide Web Consortium* y su primera versión fue lanzada en 1994. Los elementos son representados como objetos en un árbol conocido como árbol DOM y pueden ser manipulados utilizando métodos diseñados para tal efecto.

- *SAX:* SAX, cuyo nombre procede del acrónimo inglés *Simple API for XML*, es un analizador de documentos XML basado en una lectura secuencial del documento. SAX como alternativa a la librería DOM anteriormente mencionada no opera en el documento como un conjunto sino que los analizadores SAX operan en cada pieza del documento XML de manera secuencial. Esto les permite realizar una lectura más rápida y eficiente, pero por el contrario no les permite modificar el documento.

Ambas librerías están desarrolladas para la mayoría de lenguajes de programación. Durante el desarrollo del presente documento se especificará en qué punto ha sido necesario el uso de cada una de ellas y por qué se ha utilizado.

1.2.4. Representación de fórmulas matemáticas

El propósito final del proyecto es crear una base de datos con documentos escritos en LaTeX. Adicionalmente, pueden contener fórmulas matemáticas representadas con una librería matemática para LaTeX y deben conservar su significado tras el análisis del documento. Para conseguir mantener estas fórmulas matemáticas es necesario utilizar

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

Figura 1.1: Representación de una matriz

una herramienta que nos permita tal propósito, representando dichas formulas en algún formato estándar que posibilite ser interpretado.

Las herramientas encontradas a tal efecto son pocas ya que en su mayor parte, o no son estándar o requieren de software adicional para su correcta visualización. Otras alternativas serían la representación de las fórmulas por medios gráficos, como datos adjuntos a los documentos, con el trabajo que ello supone. Por estos motivos el método elegido ha sido utilizar el estándar MathML.

MathML o *Mathematical Markup Language* es un lenguaje de marcado basado en XML. Su objetivo es representar notación matemática de forma estándar de manera que pueda ser interpretada por distintas máquinas. Su uso principal se realiza en combinación con XHTML en páginas web para intercambio de información entre programas de edición matemática de propósito general. Un ejemplo de representación matemática puede incluir una explicación de números complejos, podemos observar sus representaciones tanto en LaTeX como su equivalente en MathML en las figuras 1.1, 1.2 y 1.3.

1.3. Estructura del proyecto

En este apartado se indica cual ha sido la división del presente documento, cada uno de los capítulos de los que consta el mismo y se detalla una pequeña descripción de su

```
\begin{equation*}
    \begin{pmatrix}
       0 & 1 & 0 \\
       0 & 0 & 1 \\
       1 & 0 & 0
    \end{pmatrix}
\end{equation*}
```

Figura 1.2: Representación de una matriz utilizando LaTeX

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
   <matrix>
      <matrixrow>
           <cn> 0 </cn> <cn> 1 </cn> <cn> 0 </cn>
           </matrixrow>
           <matrixrow>
           <cn> 0 </cn> <cn> 0 </cn> <cn> 1 </cn>
           </matrixrow>
           <matrixrow>
           <cn> 1 </cn> <cn> 0 </cn> <cn> 0 </cn>
      </matrixrow>
  </matrix>
</math>
```

Figura 1.3: Representación de una matriz utilizando MathML

contenido.

- *Capítulo 1:* Capítulo donde se muestra la motivación y estado del arte de las tecnologías empleadas durante el desarrollo del proyecto.
- *Capítulo 2:* Capítulo donde se presentan los objetivos del proyecto.
- *Capítulo 3:* Capítulo donde se detalla la metodología empleada en el proyecto, una enumeración de los requisitos a cumplir y una descripción de las herramientas utilizadas.
- *Capítulo 4:* Capítulo donde se describe como se han solucionado cada uno de los requisitos de manera informática.
- *Capítulo 5:* Capítulo donde muestran las diferentes pruebas realizadas tanto en el análisis de los documentos como en el servicio web.
- *Capítulo 6:* Capítulo donde se realiza un análisis general de las conclusiones del proyecto, con los hitos alcanzados y los posibles trabajos futuros.

Junto con el presente documento se entrega en soporte electrónico un CD con el código fuente implementado, y en el que además se incluye este documento en formato LaTeX y pdf.

Capítulo 2

Objetivos

Tras realizar una descripción de la naturaleza del proyecto es necesario definir unos objetivos que permitan dividir el proyecto en fases. El análisis de estas fases permite evaluar la magnitud del proyecto a desarrollar y afrontar el diseño del mismo.

Del análisis de la motivación del proyecto surge la necesidad de su división en fases, de tal manera que pueda abordarse el problema de forma estructurada. El proyecto completo debe constar de las siguientes fases:

- FASE 1: Selección y/o definición de un formato intermedio de representación de la información (DTD, esquema XML/JSON o esquema relacional de una base de datos) que resulte de aplicación para cuestionarios, como los de las asignaturas objetivo del proyecto. Dicho formato debe permitir representar el enunciado de las preguntas, las respuestas alternativas, especificar la respuesta correcta, y anotar las respuestas erróneas con explicaciones de por qué lo son (en relación a la respuesta correcta).

- FASE 2: Desarrollo de herramientas que permitan convertir de forma automática los cuestionarios de las pruebas de evaluación existentes en la actualidad (en formato LaTeX/PDF) en el formato intermedio especificado en la Fase 1, así como completar por parte de los administradores del sistema (los profesores) las explicaciones de las respuestas de los cuestionarios.

- FASE 3: Desarrollo de la aplicación web con la que los alumnos responderán los cuestionarios de autoevaluación, tomando como partida dichos cuestionarios representados en el formato intermedio. Dicha aplicación permitirá al alumno conocer el resultado obtenido en su autoevaluación y, saber de igual manera las preguntas contestadas erróneamente, y la razón de por qué lo son. Además la aplicación generará estadísticas de las preguntas acertadas y falladas que podrán ser consultadas por los administradores del sistema (los profesores), así como estadísticas sobre el número de alumnos que utiliza el sistema.

- FASE 4: Evaluación continua de los resultados del proyecto, en base al análisis de las estadísticas que va recogiendo la aplicación web.

- FASE 5: Evaluación final de los resultados del proyecto al concluir el mismo tras analizar: las estadísticas finales de la aplicación web, los resultados académicos obtenidos por los alumnos y los resultados de las encuestas telemáticas que realizarán los alumnos para valorar el proyecto.

El proyecto completo posee una envergadura tal que no permite ser abordada en un único proyecto de fin de carrera, por ello se ha considerado necesaria su división en dos proyectos. Ambos proyectos se describen a continuación:

- La primera parte centra su objetivo principal en la adaptación de pruebas de evaluación escritas en LaTeX a una base de datos, clasificando previamente su contenido. Esto permitirá posteriormente manejar la información contenida en la base de datos de manera dinámica. El contenido almacenado en esta base de datos será utilizado como base para la segunda parte.

- La segunda parte consiste en crear un sistema de autoevaluación que permita realizar un seguimiento de la progresión de los alumnos. Además, debe proporcionar las herramientas necesarias para que puedan progresar en su aprendizaje de manera autónoma. Para cumplir estos objetivos, debe crear un sistema capaz de manejar los contenidos almacenados en la base de datos por la primera parte.

2.1. Objetivos principales

Una vez ha sido presentado el marco general del proyecto se enumerarán los objetivos principales:

1. *Trasladar documentación estática a contenido dinámico:* El objetivo principal del presente proyecto consiste en la creación de una plataforma que permita transformar el contenido de diversos documentos escritos en un formato estático a contenidos dinámicos clasificados en una base de datos. Estos contenidos dinámicos podrán ser utilizados posteriormente para fines diversos.

   El contenido estático en general lo formarán ejercicios de evaluación del alumnado procedentes de años anteriores. Éstos deben ser estructuradas semánticamente, de tal manera que un software pueda ser capaz de analizar y clasificar el contenido. Este contenido se podrá utilizar de forma independiente de la prueba de evaluación a la que pertenecía. Esto permitirá, por ejemplo, utilizar dos pruebas de evaluación

y generar otra con el contenido mezclado de ambas, en distinto orden o con alguna temática concreta.

2. *Unificación de contenidos:* Otro de los objetivos principales del proyecto consiste en la unificación de todos los contenidos de varias asignaturas del departamento de Sistemas Telemáticos y Computación de la Universidad Rey Juan Carlos, lo que permitirá mantener organizados documentos antiguos y dotarles de nueva utilidad.

   Así, asignaturas que compartan una parte de la guía docente o temario, podrán verse beneficiadas.

2.2. Objetivos secundarios

Tras enunciar los objetivos principales del proyecto se enumeran otra serie de objetivos secundarios que tienen como finalidad apoyar su desarrollo:

1. *Diseño de un pseudolenguaje:* Para la clasificación de los datos que contienen los ficheros es necesario utilizar una serie de marcas que aporten contenido semántico, es decir, añadir metadatos a los datos de tal forma que éstos puedan ser clasificados de forma automática. Adicionalmente, se puede generar un sistema que facilite el marcado de los documentos originales y abstraiga del pseudolenguaje utilizado.

2. *Creación de un sistema de lectura de datos:* Una vez diseñado el pseudolenguaje descrito en el punto anterior, se debe crear un sistema capaz de leer los ficheros con los metadatos y clasificarlos.

3. *Creación de una base de datos:* Para almacenar los datos de los documentos es necesario diseñar una base de datos que permita posteriormente utilizarlos. La base de datos generada en este apartado será el punto de partida del proyecto posterior que se encargará de realizar el sistema de evaluación.

A lo largo de los siguientes capítulos se desarrollaran soluciones para cada uno de los objetivos.

Capítulo 3

Especificación y diseño

Una vez planteados los objetivos del proyecto es necesario llevar a cabo un diseño estructurado de cada uno de los elementos que lo componen. Así mismo es necesario definir un modelo de desarrollo que permita mantener un control sobre los progresos realizados.

3.1. Metodología empleada

Durante todo el desarrollo del proyecto se ha utilizado un modelo en espiral. Según indica este modelo cada ciclo de la espiral representa el ciclo de vida de una actividad. Cada actividad a su vez está dividida en una serie de tareas:

1. Determinar los objetivos a conseguir en la iteración, se decide que se debe conseguir: requisitos, restricciones, etc.

2. Análisis de los riesgos: se realiza un estudio de posibles amenazas, complicaciones o contratiempos que puedan surgir. Con este estudio se toma una serie de decisiones que permitan mitigar los posibles efectos de los riesgos.

3. Desarrollo, verificación y pruebas: se implementan y se evalúan los algoritmos necesarios. Dependiendo de la magnitud de la iteración puede utilizarse o no una metodología independiente para esta fase.

4. Planificación del siguiente ciclo: se realiza una revisión de los progresos conseguidos y se planifica la siguiente ruta de acción.

El alcance del proyecto viene determinado por el número de iteraciones de la espiral. A medida que se avanza en la espiral, el proyecto añade funcionalidad y avanza en los objetivos. Tras un determinado numero de iteraciones se considera que no es necesario realizar mas iteraciones y se da por finalizado el desarrollo.

Este modelo favorece la independencia entre las diferentes tareas que componen el proyecto, y, permite orientar el desarrollo del proyecto según los resultados de cada una de las iteraciones. Por ello se ha considerado un modelo apropiado para el desarrollo del proyecto.

3.2. Análisis de requisitos

El desarrollo software por norma general utiliza un diagrama de casos de uso para orientar y obtener los requisitos. En este proyecto únicamente existe un actor que se

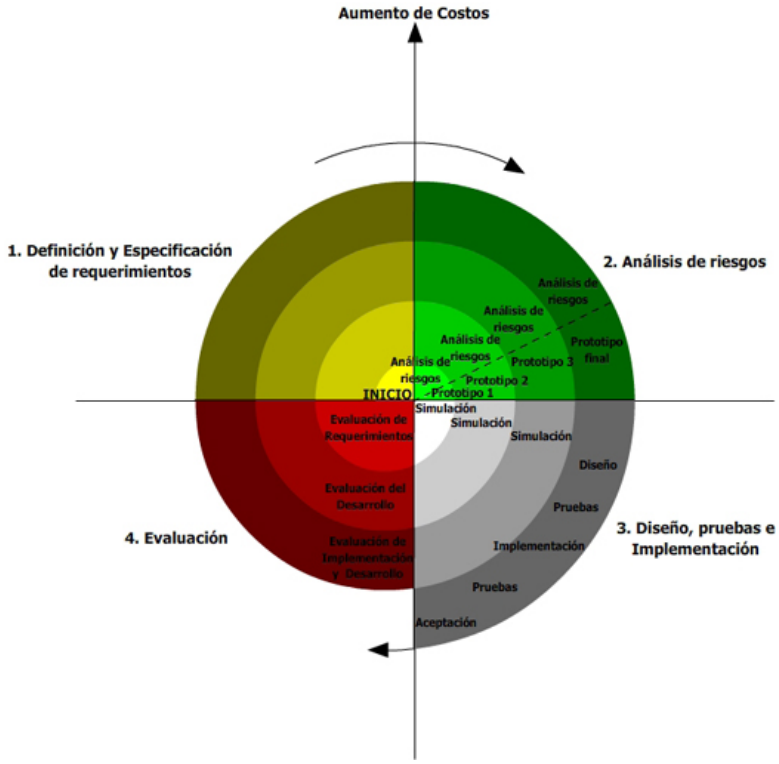


Figura 3.1: Diagrama del modelo de desarrollo en espiral

encargará de iniciar el proceso de transformación o análisis de los datos, a su vez la interacción con el sistema será mínima, por lo que se ha considerado relevante realizar sólo pequeñas alusiones gráficas al proceso en lugar de realizar todos y cada uno de los diagramas.

A continuación se realizará una especificación de requisitos sobre los diferentes objetivos del proyecto.

3.2.1. Trasladar documentación estática a contenido dinámico

Para realizar esta transformación es necesario conocer cuáles son los diferentes contenidos estáticos, sus distintos formatos e identificar cuáles van a ser los formatos de salida.

Requisitos funcionales:

- *RF001:* Es necesario que el software desarrollado sea capaz de analizar cada uno

de los documentos de entrada. Estos documentos de entrada han de encontrarse en dos posibles formatos, LaTeX o texto plano.

- *RF002:* El sistema debe generar un documento de salida en XML. Este documento XML podrá ser utilizado posteriormente como documento de entrada.

- *RF003:* El sistema debe ser capaz de notificar un error en caso de detectar que un documento no se encuentra en un formato válido. El usuario debe saber que la transformación no ha sido realizada debido a un problema con el formato del documento de entrada.

- *RF004:* En caso de que la transformación se haya realizado correctamente, el sistema debe comportarse de forma transparente hacia el usuario. Es posible que realice una pequeña notificación de su correcta o fallida transformación al finalizar el proceso.

- *RF005:* El sistema debe ser capaz de organizar la información contenida en los documentos de entrada mediante un método de clasificación.

Requisitos no funcionales:

- *RNF001:* El software desarrollado debe funcionar sobre un sistema operativo Linux. Adicionalmente puede evaluarse la posibilidad de ejecutar el software sobre un sistema operativo diferente. Ej Windows o MacOS.

- *RNF002:* El software desarrollado debe tratar la información siguiendo unas condiciones mínimas de *velocidad*. Es decir, debe proporcionar una respuesta al usuario en un tiempo mínimo.

3.2.2. Unificación de contenidos

Los documentos de entrada se encuentran en diversos formatos pero su contenido debe conservarse de manera íntegra.

Requisitos funcionales:

- *RF006:* Los documentos tras la transformación deben conservar su contenido original. Debido a que los contenidos se encuentran en formato LaTeX es necesario diferenciar la forma en que se visualiza el contenido del contenido en si mismo.

- *RF007:* Es necesario que los documentos mantengan cierta información semántica antes de ser almacenados. Esta información hace referencia al tipado de los textos o formulas matemáticas. Esto debe conservarse en un formato estándar que permita representarlo posteriormente.

3.2.3. Diseño de un pseudo-lenguaje

La heterogeneidad de los documentos de entrada hace necesaria la creación de un estándar. Este estándar tiene como finalidad unificar la estructura de los contenidos. A continuación se muestran los requisitos que debe cumplir dicho estándar.

Requisitos funcionales:

- *RF008:* Debe estar formado por órdenes que permitan clasificar el contenido de los documentos de entrada.

- *RF009:* Debe introducirse en los documentos de entrada de manera transparente para un compilador de documentos LaTeX.

- *RF010:* Debe estar diseñado de tal manera que permita una fácil conversión a XML para su posterior procesado.

- *RF011:* Debe tener una estructura fácilmente ampliable. Esto permitirá en un futuro añadir nueva información.

- *RF012:* Debe facilitar la clasificación del contenido de los documentos. Añadirá metainformación al contenido de los archivos de manera que será posible su clasificación.

3.2.4. Creación de un sistema de lectura de datos

Como complemento al sistema de traducción se ha considerado necesario realizar un sistema que permita la lectura de los documentos, comprenda el pseudo-lenguaje creado y clasifique el contenido de los documentos.

Requisitos funcionales:

- *RF013:* Debe ser capaz de leer los documentos en cualquiera de los formatos indicados.
- *RF014:* Debe ser capaz de comprender el pseudo-lenguaje desarrollado y utilizarlo para clasificar los datos que contienen los documentos.
- *RF015:* Debe a su vez procesar los ficheros de entrada y generar un documento XML estándar con el contenido de los documentos de entrada.

Requisitos no funcionales:

- *RNF003:* Debe funcionar sobre un sistema operativo Linux. En caso de que el sistema de entrada contenga un método de introducción de datos WEB es necesario que funcione sobre el navegador Mozilla Firefox.
- *RNF004:* Debe tratar la información siguiendo unas condiciones mínimas de *velocidad*. El proceso de transformación debe ser rápido y ágil. El usuario debe recibir una respuesta en el menor tiempo posible.

3.2.5. Creación de una base de datos

Para la creación de una base de datos con todos los contenidos de los documentos analizados, son necesarios una serie de requisitos:

Requisitos funcionales:

- *RF016:* Debe mantener una estructura que permita gran versatilidad a la hora de almacenar los distintos contenidos de los documentos.

- *RF017:* Todos los contenidos de los documentos de entrada deben quedar registrados en esta base de datos.
- *RF018:* Si alguno de los documentos de entrada contiene contenido adicional (imágenes, archivos de datos) también deben quedar almacenados en la base de datos, o, en último caso, se debe guardar una referencia al mismo.

Requisitos no funcionales:

- *RNF005:* La base de datos debe ser desarrollada sobre SQLite, sistema de bases de datos sobre el cual se hablará en el capítulo siguiente. Es posible que en el futuro sea necesario trasladar la información a una base de datos de mayor envergadura por lo que es importante tenerlo en cuenta en el diseño.
- *RNF006:* El acceso a los datos de la base de datos debe mantener unos tiempos de acceso y tiempos de escritura *aceptables*.

3.3. Diseño de un pseudo-lenguage de adaptación

Para completar los objetivos principales del proyecto, leer, analizar y obtener información de un documento, se ha considerado necesario realizar un marcado sobre cada uno de los documentos de origen. Este marcado tiene como finalidad orientar el proceso de análisis y permitir clasificar el contenido de los documentos. También puede añadir información extra al documento. En la figura 3.2 se puede observar como el marcado que se realizó, permite al software identificar cada una de las partes del documento y clasificarlos en la BD.

Para realizar este marcado se ha diseñado un lenguaje de adaptación. Dicho lenguaje define una estructura que deben cumplir todos los documentos que vayan a ser analizados. Es esta estructura la que permite identificar cada una de las partes del documento, independientemente de cual sea su contenido o formato.

Este lenguaje de marcado se ha diseñado con la capacidad de poder ser ampliado para añadir otro tipo de información relevante, como referencias a otros documentos, imágenes, enlaces, etc.

Figura 3.2: Representación del proceso de transformación de los documentos

Los documentos que van a ser transformados tienen carácter académico, y su contenido siempre representa una prueba de evaluación en formato de test. Este lenguaje permitirá mezclar los contenidos de varios documentos. La finalidad es poder manejar toda la información de los documentos por separado y de forma dinámica (figura 3.2).

Este lenguaje de marcado está dirigido por una serie de órdenes básicas y órdenes especificas o especializadas. Todas siguen un patrón predefinido:

- `%Command:Orden(Begin);`
- `%Command:Orden(End);`

Donde `Orden` hace referencia a cada uno de las diferentes marcas de las que dispone el lenguaje. Algunas de estas órdenes tienen elementos adicionales que le permiten añadir información. Estas órdenes siguen el siguiente formato:

- `%Command:Orden(Begin)( Nombre='Valor')*;`
- `%Command:Orden(End);`

Donde `Nombre` es el identificador del contenido y `Valor` es el contenido en si mismo.

Este lenguaje ha sido definido siguiendo unas características, de las cuales, se mencionan a continuación las más importantes.

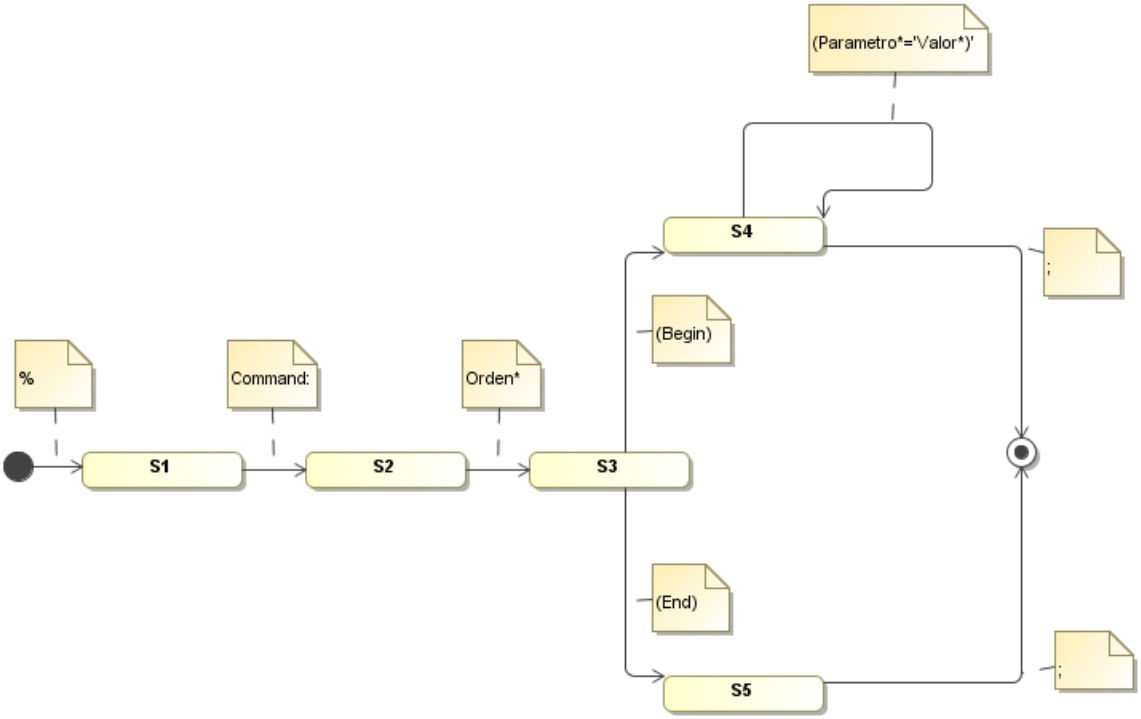


Figura 3.3: Diagrama del lenguaje

- *Transparente:* El lenguaje debe ser transparente en caso de analizar el documento LaTeX en un editor externo a este proceso. Por ello, se ha decidido que este lenguaje quede enmarcado dentro de los comentarios que permite el lenguaje LaTeX utilizando para ello el carácter %.

  ```
  %Command:Orden(Begin-End)( Nombre='Valor')*;
  ```

- *Diferenciado:* El lenguaje debe diferenciarse del resto de comentarios escritos en el documento y por tanto pueden convivir con estos sin que estos comentarios influyan en el resultado del análisis. Para ello se ha utilizado la palabra clave `Command` que permite diferenciar una marca de este lenguaje de otro comentario cualquiera.

  %Command:`Orden(Begin-End)( Nombre='Valor')*;`

- *Versátil:* El lenguaje debe ser versátil y debe permitir añadir nuevas órdenes o mandatos. Para ello se ha diseñado de tal manera que permita añadir nuevas órdenes utilizando un nombre que la identifique.

  `%Command:`**Orden**`(Begin-End)( Nombre='Valor')*;`

- *Por bloque de información:* El lenguaje limita los bloques de información identificando su contenido de forma inequívoca. Para ello se ha decidido que este lenguaje utilice una marca para iniciar un bloque de información y otra para finalizar el bloque de información.

  `%Command:Orden(Begin-End)( Nombre='Valor')*;`

- *Parametrizable:* El lenguaje debe permitir que cada una de las órdenes pueda incluir determinados valores que sirvan como modificadores o contengan información adicional. Para ello se permiten parámetros al final del comando, estos parámetros constan de una clave y un valor. La clave permite identificar el parámetro y el valor es el contenido adicional.

  `%Command:Orden(Begin-End)( Nombre='Valor')*;`

Las diferentes órdenes que pertenecen al lenguaje serán descritas a continuación.

3.3.1. Órdenes básicas

Las órdenes o etiquetas básicas, son aquellas que deben estar presentes en un documento que se desee analizar. Estas órdenes tienen como finalidad construir la estructura del documento, indicando donde comienza y acaba cada una de las secciones que componen el mismo. A continuación se define cada una de ellas y su significado.

1. ***Document:*** Delimita el contenido estructurado con el de marcado.

   Se utilizan dos órdenes que enmarcan la parte del documento a ser tratado por el software. La primera etiqueta indica el lugar donde se debe comenzar a analizar y la segunda donde finaliza la parte a analizar, el resto del documento será ignorado.

   - `%Command:Document(Begin);`
   - `%Command:Document(End);`

2. ***Group:*** Dentro de un documento pueden existir distintos grupos de información y todos ellos serán procesados por separado. Esta orden delimita cada uno de esos grupos.

Se utilizan dos órdenes que enmarcan la parte del documento a ser tratada como un único bloque de información. Las órdenes que enmarcan estas agrupaciones son las siguientes:

- `%Command:Group(Begin);`
- `%Command:Group(End);`

3. ***GroupIntro:*** Cada grupo de información contiene un texto introductorio. Este texto es considerado como una descripción de su contenido.

   Esta información queda enmarcada dentro de las órdenes que se describen a continuación:

   - `%Command:GroupIntro(Begin)(Type='')(Desc='');`
   - `%Command:GroupIntro(End);`

   Como se puede observar contienen elementos adicionales que se utilizan para añadir información al contenido de todo el grupo.

   - `"Type"`: Esta etiqueta hace referencia al tipo de contenido del grupo. Si utilizamos el mismo tipo para enmarcar dos grupos en distintos documentos, el contenido de estos documentos queda relacionado y almacenado de manera uniforme.
   - `"Desc"`: Esta etiqueta puede ser utilizada para realizar una descripción más extensa del contenido del grupo. La descripción es independiente para cada grupo.

4. ***Question:*** Cada grupo está compuesto por diversas cuestiones. Éstas serán clasificadas en función del grupo al que pertenezcan.

   Esta información queda enmarcada dentro de las órdenes que se describen a continuación:

   - `%Command:Question(Begin);`
   - `%Command:Question(End);`

5. ***QuestionIntro:*** Cada una de las cuestiones que forman parte de un grupo contienen un enunciado. Éste representa el enunciado de cada una de las preguntas que contiene el documento de evaluación analizado.

   Esta información queda enmarcada dentro de las órdenes que se describen a continuación:

   - `%Command:QuestionIntro(Begin);`
   - `%Command:QuestionIntro(End);`

6. ***QuestionPossibility:*** Es utilizado para enmarcar el contenido que hace referencia a una posible respuesta a la cuestión a la que pertenece.

   Esta información queda enmarcada dentro de las órdenes que se describen a continuación:

   - `%Command:QuestionPossibility(Begin)(Exp='Explicacion')(Correct='F');`
   - `%Command:QuestionPossibility(End);`

   Al igual que la orden `"QuestionIntro"` . Pueden especificarse parámetros adicionales para añadir información a dicha respuesta. Estos parámetros son:

   - `"Correct"`: Esta etiqueta es utilizada para indicar si la respuesta es correcta o incorrecta en relación a la pregunta a la que pertenece.
   - `"Exp"`: Este parámetro es utilizado para indicar una posible explicación de por qué es o no correcta la respuesta.

3.3.2. Órdenes específicas

Las órdenes especificas o adicionales son aquellas que no tienen por qué estar presentes en el documento a analizar, es decir, son opcionales. En caso de estar presentes, representan información adicional relacionada con cada elemento al que pertenecen. A continuación se enumera cada una de las posibles órdenes adicionales.

1. ***QuestionImage:*** Representa un recurso gráfico dentro del ámbito en el que se utilice, este recurso será almacenado en la base de datos. Puede encontrarse dentro del ámbito de las órdenes *Question* y *Group*. En caso de formar parte de *Group* el recurso gráfico se almacenará relacionado con cada una de las etiquetas question que contenga el grupo analizado, si por el contrario esta asociado a la orden *Question* el recurso gráfico únicamente se almacenará con la etiqueta Question a la que pertenece.

   Esta información queda enmarcada dentro de las órdenes que se describen a continuación:

   - `%Command:QuestionImage(Begin)(Image='Ruta Imagen');`
   - `%Command:QuestionImage(End);`

   La etiqueta Question Image contiene un parámetro que permite añadir contenido adicional a la orden.

   - `"Image":` Esta etiqueta es utilizada para indicar la ruta al recurso gráfico.

2. ***QuestionResource:*** Representa un recurso de cualquier tipo dentro del ámbito en el que se utilice. Este recurso será almacenado en la base de datos. Puede encontrarse dentro del ámbito de las órdenes *Question* y *Group*, en caso de formar parte de *Group* el recurso se almacenará relacionado con cada una de las etiquetas Question que contenga el grupo analizado, si por el contrario está asociado a la orden *Question* el recurso únicamente se almacenará con la cuestión a la que pertenece.

   Esta información queda enmarcada dentro de las órdenes que se describen a continuación:

   - `%Command:QuestionResource(Begin)(Resource='Ruta Recurso');`
   - `%Command:QuestionResource(End);`

   La etiqueta Question Resource contiene un parámetro que permite añadir contenido adicional a la orden.

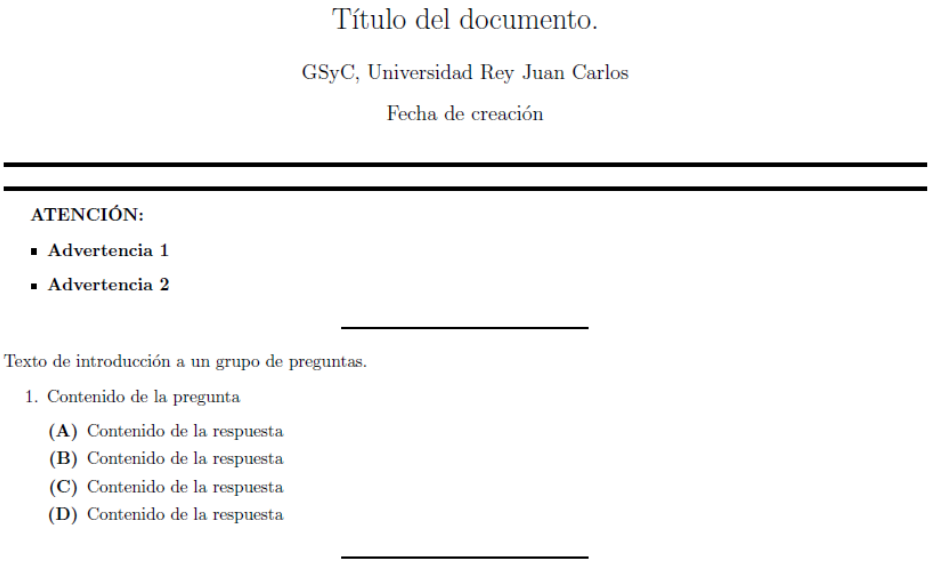
   - `"Resource":` Esta etiqueta es utilizada para indicar la ruta al recurso.

3.3.3. Plantilla

Todos los documentos de entrada deben ser marcados utilizando las órdenes anteriormente descritas. Este marcado puede dar lugar a un proceso tedioso, por ello más adelante se definen soluciones para evitar introducir las órdenes a mano y otros métodos de introducción de datos basados en plantillas. Un ejemplo de documento escrito en LaTeX con el marcado insertado puede verse en la figura 3.4 y el resultado generado listo para ser introducido en la base de datos sería el indicado en la figura 3.6:

```
%Command:Document(Begin);
\documentclass[a4paper]{article}
\usepackage[spanish]{babel}
\usepackage[utf8]{inputenc}
\usepackage[scale=0.89]{geometry}
\usepackage{setspace}
\usepackage{amsmath}
\usepackage{amssymb}
\usepackage{enumerate}
\usepackage{epsfig}
\usepackage{graphicx}
\usepackage{verbatim}
\usepackage{color}
\newcommand{\res}[1]{\textcolor{darkred}{#1}}
\title{Título del documento.}
\author{GSyC, Universidad Rey Juan Carlos}
\date{Fecha de creación}
\begin{document}
\maketitle
\hrule width \textwidth height 2pt
\vspace{2mm}
\vspace{2mm}
\hrule width \textwidth height 2pt
\vspace{3mm}
{\bf
  ATENCIÓN:
  \begin{itemize}
  \item Advertencia 1
  \item Advertencia 2
  \end{itemize}
}
\begin{center}
  \rule{5cm}{1pt}
\end{center}
%Command:Group(Begin);
%Command:GroupIntro(Begin)(Type='Tipo De preguntas.')(Desc='Descripción de la
categoría.');
Texto de introducción a un grupo de preguntas.
%Command:GroupIntro(End);
\begin{enumerate}
\setcounter{enumi}{0}
%Command:Question(Begin);
%Command:QuestionIntro(Begin);
\item Contenido de la pregunta
%Command:QuestionIntro(End);
     \begin{enumerate}[\bf (A)]
       %Command:QuestionPossibility(Begin)(Exp='Explicación')(Correct='F');
         \item Contenido de la respuesta
       %Command:QuestionPossibility(End);
       %Command:QuestionPossibility(Begin)(Exp='Explicación')(Correct='F');
         \item Contenido de la respuesta
       %Command:QuestionPossibility(End);
       %Command:QuestionPossibility(Begin)(Exp='Explicación')(Correct='F');
         \item Contenido de la respuesta
       %Command:QuestionPossibility(End);
       %Command:QuestionPossibility(Begin)(Exp='Explicación')(Correct='T');
         \item Contenido de la respuesta
       %Command:QuestionPossibility(End);
     \end{enumerate}
%Command:QuestionImage(Begin)(Image='Ruta Imagen');
%Command:QuestionImage(End);
%Command:QuestionResource(Begin)(Resource='Ruta Recurso');
%Command:QuestionResource(End);
%Command:Question(End);
\end{enumerate}
\begin{center}
  \rule{5cm}{1pt}
\end{center}
%Command:Group(End);
%Command:Document(End);
\end{document}
```

Figura 3.4: Plantilla de ejemplo de un posible documento de entrada



Título del documento.

GSyC, Universidad Rey Juan Carlos

Fecha de creación

ATENCIÓN:

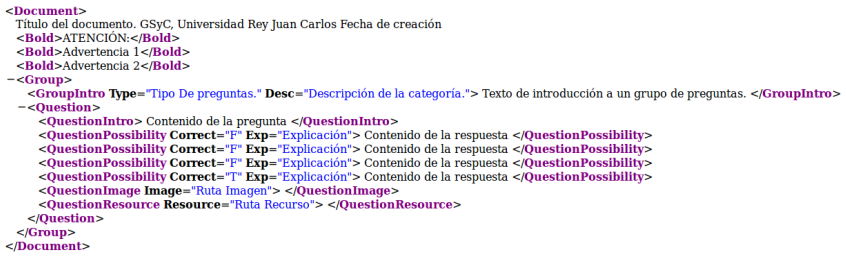
- **Advertencia 1**
- **Advertencia 2**

Texto de introducción a un grupo de preguntas.

1. Contenido de la pregunta

   (A) Contenido de la respuesta
   (B) Contenido de la respuesta
   (C) Contenido de la respuesta
   (D) Contenido de la respuesta

Figura 3.5: Salida del documento latex como pdf



```
<Document>
  Título del documento. GSyC, Universidad Rey Juan Carlos Fecha de creación
  <Bold>ATENCIÓN:</Bold>
  <Bold>Advertencia 1</Bold>
  <Bold>Advertencia 2</Bold>
 −<Group>
    <GroupIntro Type="Tipo De preguntas." Desc="Descripción de la categoría."> Texto de introducción a un grupo de preguntas. </GroupIntro>
   −<Question>
      <QuestionIntro> Contenido de la pregunta </QuestionIntro>
      <QuestionPossibility Correct="F" Exp="Explicación"> Contenido de la respuesta </QuestionPossibility>
      <QuestionPossibility Correct="F" Exp="Explicación"> Contenido de la respuesta </QuestionPossibility>
      <QuestionPossibility Correct="F" Exp="Explicación"> Contenido de la respuesta </QuestionPossibility>
      <QuestionPossibility Correct="T" Exp="Explicación"> Contenido de la respuesta </QuestionPossibility>
      <QuestionImage Image="Ruta Imagen"> </QuestionImage>
      <QuestionResource Resource="Ruta Recurso"> </QuestionResource>
    </Question>
  </Group>
</Document>
```

Figura 3.6: Salida del documento latex como xml

3.4. Diseño de la base de datos

Se ha diseñado una base de datos que permite unificar los archivos analizados. Esta base de datos se convertirá en el punto de cohesión de este proyecto con un proyecto futuro, por lo tanto, es necesario que el diseño sea lo más transparente posible y deba permitir su ampliación en el futuro.

El diseño de la base de datos se representa en la figura 3.7. Se ha escogido este diseño porque se ajusta muy bien a la estructura de los documentos de entrada y al

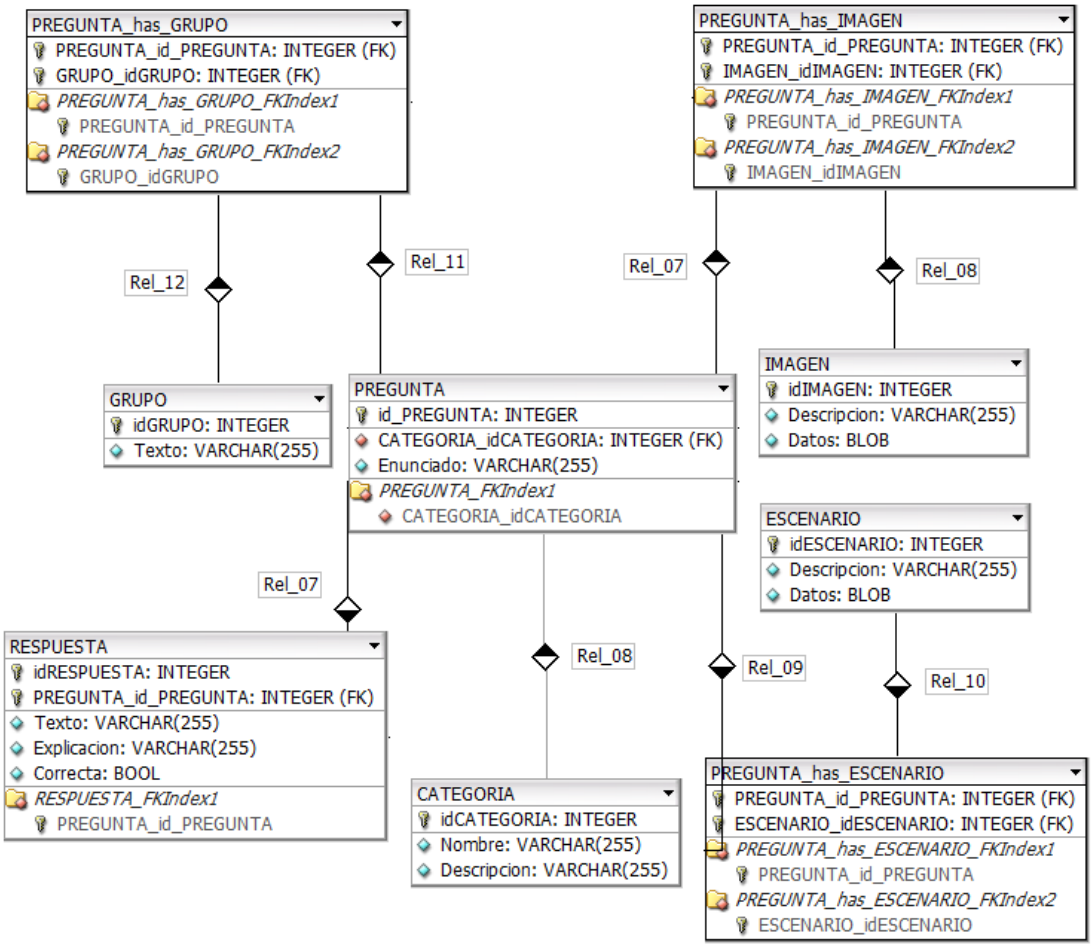


Figura 3.7: Diseño de la base de datos

pseudo-lenguaje diseñado. La base de datos se convierte por tanto en un aglutinador de documentos, lo que permite visualizar la base de datos como un único documento que une todos los documentos que han sido procesados.

1. *Pregunta*:

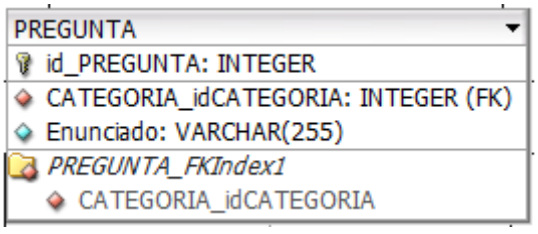


Figura 3.8: Tabla pregunta

Almacena los datos referentes a los enunciados de las preguntas que contienen los documentos de entrada.

- `id_PREGUNTA`: Identificador único de la pregunta. Es generado de forma incremental al introducir la pregunta en la base de datos.
- `CATEGORIA_idCATEGORIA`: Relaciona una pregunta con una categoría de preguntas, de forma que una pregunta debe pertenecer a una categoría concreta.
- `Enunciado`: Hace referencia al contenido del texto de la pregunta, es decir, el contenido que originalmente contenía el documento de entrada relacionado con esta cuestión concreta.

2. *Categoría*:

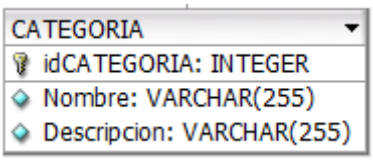


Figura 3.9: Tabla categoria

Almacena las diferentes categorías o tipos de preguntas que contiene la base de datos.

- `id_CATEGORIA`: Identificador único de la categoría. Es generado de forma incremental al introducir la categoría en la base de datos.
- `Nombre`: Nombre de la categoría.
- `Enunciado`: Descripción del contenido de la categoría.

3. *Grupo*:

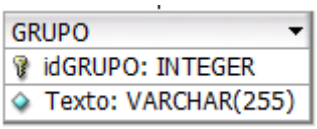


Figura 3.10: Tabla grupo

Almacena agrupaciones de preguntas. Esta tabla permite realizar agrupaciones de preguntas de diversas categorías y representa la agrupación de preguntas que originalmente tiene un documento de entrada.

- `idGRUPO`: Identificador único del grupo. Es generado de forma incremental al introducir el grupo en la base de datos.
- `Texto`: Cualquier tipo de texto descriptivo del grupo.

4. *Respuesta*:

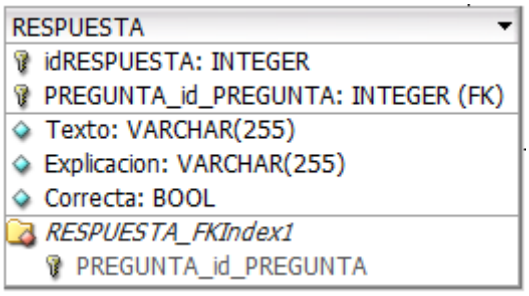


Figura 3.11: Tabla respuesta

Almacena las diferentes respuestas posibles para cada uno de los enunciados almacenados en la tabla preguntas.

- `idRESPUESTA`: Identificador único de la respuesta. Es generado de forma incremental al introducir la respuesta en la base de datos.
- `PREGUNTA_id_PREGUNTA`: Relaciona la respuesta con una pregunta de la tabla PREGUNTA.
- `Texto`: Texto que representa cual puede ser una posible respuesta a la pregunta a la que pertenece.
- `Explicación`: En este campo se realiza un breve texto indicando por qué la respuesta es correcta o no.
- `Correcta`: Indica si la respuesta es correcta o no.

5. *Imagen*:

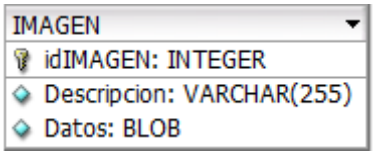


Figura 3.12: Tabla imagen

Almacena la imagen o imágenes que puede contener una pregunta.

- `idIMAGEN:` Identificador único de la imagen. Es generado de forma incremental al introducir la imagen en la base de datos.
- `Descripción:` Contiene un breve análisis de la imagen
- `Datos:` Almacena los datos relacionados con la imagen.

6. *Escenario*:

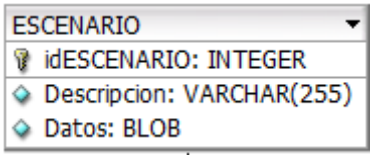


Figura 3.13: Tabla escenario

Almacena posibles recursos adicionales que puede contener una pregunta.

- `idESCENARIO:` Identificador único del recurso adicional. Es generado de forma incremental al introducir el recurso en la base de datos.
- `Descripción:` Contiene una breve descripción del recurso.
- `Datos:` Almacena los datos relacionados con el recurso.

El resto de tablas se encargan de realizar las relaciones entre los diferentes elementos principales de la base de datos, por lo que su contenido no se describe en este documento.

3.5. Herramientas utilizadas

Durante el desarrollo del proyecto se ha tratado de favorecer el desarrollo sobre software libre utilizando herramientas y utilidades de código abierto.

A continuación se analizarán las principales herramientas y utilidades junto con su descripción y su aplicación en el proyecto.

3.5.1. Tralics

En este proyecto es necesario el uso de una herramienta de transformación de documentos escritos en LaTeX capaz de generar documentos en XML o un formato HTML + MathML. Después de su correspondiente estudio, se ha decido utilizar la herramienta Tralics. Tralics es una herramienta para transformar documentos escritos en LaTeX a XML, más concretamente realiza una transformación de documentos escritos en LaTeX a XHTML + MathML. Es una herramienta escrita en C++ que comenzó su desarrollo en 2003, y continúa activo hoy día. La última versión de Tralics corresponde a la versión 2.14.4 lanzada en noviembre de 2011.

En el capítulo siguiente se describirá como se ha utilizado en el proyecto.

3.5.2. MathML

MathML (Mathematical Markup Language) es un lenguaje de marcas dirigido a la representación de fórmulas matemáticas, permite su procesado de forma dinámica, por ello está basado en una serie de etiquetas XML. Se han publicado varias versiones de MathML: MathML 1.0 (1998), MathML 1.01 (1999), MathML 2.0 (2001), MathML 2.0 (2º edición) (2003) y MathML 3.0 (octubre de 2010).

En el proyecto se ha utilizado MathML para almacenar fórmulas y símbolos matemáticos de forma que puedan ser reproducidos con posterioridad en un visor de documentos XHTML.

3.5.3. Python como lenguaje de cohesión

Phyton es un lenguaje de programación interpretado, lo cual tiene la principal ventaja de que evita tiempos de compilado y enlazado, ya que esas tareas las realiza el intérprete. Nace en 1991, siendo la última versión estable Python 2.7.3 y Python 3.2.3. Se mencionan

dos versiones como finales puesto que existen diferencias entre las versiones 2.x y 3.x que las hacen incompatibles. En este proyecto se ha optado por la compatibilidad y actualmente la versión más indicada para desarrollos siguen siendo las basadas en Python 2.x. Python se desarrolla como un proyecto de software libre amparado por la Python Software Foundation.

Una de las características fundamentales que hace que este lenguaje tenga numerosos seguidores es la sencillez y elegancia que presenta a la hora de desarrollar programas. Prueba de ello es la posibilidad de modularizar el código de un programa de tal manera que cada uno de esos módulos sean aprovechables desde otras aplicaciones, haciendo más eficiente la estructuración del código a la hora de programar. En general, estas características se enuncian en un listado de sentencias que conforman la "filosofía"de Python[1].

Python es un lenguaje de programación sencillo y multiplataforma. Permite simplificar los tratamientos de documentos, este es el principal motivo por el que se ha elegido este lenguaje. En concreto se han generado una serie de scripts que procesan los documentos de entrada para generar y almacenar en la base de datos los datos extraídos. Por otro lado y, como se podrá ver más adelante, se ha utilizado Django Python para desarrollar un sistema web de introducción de datos.

3.5.4. SQLite

SQLite es un sistema de gestión de bases de datos relacional compatible con ACID, contenido en una biblioteca relativamente pequeña escrita en C.

A diferencia de los sistemas de gestión de bases de datos cliente-servidor, el motor de SQLite no es un proceso independiente con el que el programa principal se comunica. En lugar de eso, la biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo. El conjunto de la base de datos (definiciones, tablas, índices, y los propios datos), son guardados como un sólo fichero estándar en la máquina host. Este diseño simple se logra bloqueando todo el fichero de base de datos al principio de cada

[1]Estas sentencias se encuentran disponibles en `http://c2.com/cgi/wiki?PythonPhilosophy`, o ejecutando el comando "import this" en una shell cualquiera de Python.

transacción. En su versión 3, versión utilizada en este proyecto, SQLite permite bases de datos de hasta 2 Terabytes de tamaño, y también permite la inclusión de campos tipo BLOB.

Se ha utilizado este sistema de gestión de base de datos por ser un sistema ligero y fácilmente portable a ampliaciones posteriores, puesto que los datos se encuentran en un fichero. El diseño del software debe permitir realizar una migración a otro tipo de base de datos en el futuro.

3.5.5. Django

Django es un framework web de código abierto escrito en Python que permite construir aplicaciones web más rápido y con menos código. Django fue inicialmente desarrollado para gestionar aplicaciones web de páginas orientadas a noticias de World Online, más tarde se liberó bajo licencia BSD. Django se centra en automatizar todo lo posible y se adhiere al principio DRY (Don't Repeat Yourself).

En este proyecto se ha utilizado Django para diseñar un servicio web. Éste será el encargado de facilitar una interfaz simple para la introducción de documentos de entrada, de manera que sea posible almacenar documentos utilizando un navegador web y debe ser el encargado de manejar los scripts generados para el análisis de los documentos y proporcionar un tratamiento de los mismos transparente de cara al usuario.

3.5.6. Otras herramientas y tecnologías

Además de las anteriormente mencionadas se han utilizado otras herramientas y tecnologías como pueden ser:

- *Javascript y HTML*: Mediante javascript, un lenguaje de programación interpretado, y HTML, un lenguaje de marcado, han sido desarrolladas las vistas presentadas en el navegador del usuario al utilizar el servicio web desarrollado en Django.
- *XMLindent*: Es un software de código abierto utilizado para identar correctamente documentos XML. Este software ha sido utilizado para presentar los documentos

XML generados tras el procesado del documento de entrada.

- *SQLiteman y SQLiteManager*: Ambos son programas para manejar el contenido de bases de datos SQLite. El primero de ellos es un software de escritorio proporcionado por los repositorios de Ubuntu. El segundo es un complemento disponible para el navegador *Mozilla Firefox*.

Capítulo 4

Implementación del sistema principal.

Este proyecto trabaja con documentos que mantienen una estructura definida. Este hecho permite la creación de una herramienta sistemática capaz de procesar los datos que contienen, dividiéndolos y clasificándolos. Para llevar a cabo este análisis y transformación de documentos es necesario dividir el problema en fases, en cada una de las cuales será necesario identificar los problemas y buscar sus soluciones concretas para cada uno de ellos.

4.1. Transformación del documento de entrada a XML.

Con el objetivo de transformar el documento de entrada y convertirlo en un documento XML estructurado es necesario que este documento lleve a cabo una serie de pasos:

1. Preprocesado del documento de entrada.

2. Tratamiento del documento utilizando Tralics.

3. Generar un documento final XML.

4. Análisis del documento generado.

Antes de comenzar a definir cada uno de los pasos necesarios para realizar la transformación o análisis, se van a describir cada uno de los posibles documentos de entrada. Actualmente el software desarrollado es capaz de admitir tres tipos de documentos de entrada. Todos ellos deben tener en común una serie de marcas[1] que el software necesita para interpretar los datos que contienen y ser capaz de clasificar su contenido. Estas marcas están formadas por las órdenes del pseudo–lenguaje definido en el capitulo anterior. Los documentos admitidos son los siguientes::

- *Documento LaTeX* El software del proyecto recibirá un documento escrito en LaTeX previamente procesado con las órdenes descritas en el pseudo–lenguaje diseñado. El archivo será interpretado utilizando la herramienta Tralics. En la figura 4.1 podemos ver un fragmento de un posible documento de entrada escrito en LaTeX.

[1] Estas marcas fueron definidas en el capítulo anterior como órdenes o comandos.

```
\begin{enumerate}
\setcounter{enumi}{0}
%Command:Question(Begin);
%Command:QuestionIntro(Begin);
\item Contenido de la pregunta
%Command:QuestionIntro(End);
     \begin{enumerate}[\bf (A)]
       %Command:QuestionPossibility(Begin)(Exp='Explicación')(Correct='F');
         \item Contenido de la respuesta
       %Command:QuestionPossibility(End);
       %Command:QuestionPossibility(Begin)(Exp='Explicación')(Correct='F');
         \item Contenido de la respuesta
       %Command:QuestionPossibility(End);
       %Command:QuestionPossibility(Begin)(Exp='Explicación')(Correct='F');
         \item Contenido de la respuesta
       %Command:QuestionPossibility(End);
       %Command:QuestionPossibility(Begin)(Exp='Explicación')(Correct='T');
         \item Contenido de la respuesta
       %Command:QuestionPossibility(End);
\end{enumerate}
%Command:QuestionImage(Begin)(Image='Ruta Imagen');
%Command:QuestionImage(End);
%Command:QuestionResource(Begin)(Resource='Ruta Recurso');
%Command:QuestionResource(End);
%Command:Question(End);
\end{enumerate}
```

Figura 4.1: Ejemplo de una pregunta procesada en LaTeX

- *Documento en texto plano:* El software del proyecto recibirá un documento con texto plano, el cual habrá sido previamente procesado con las órdenes descritas en el pseudo-lenguaje diseñado. En la figura 4.2 podemos ver un fragmento de un posible documento de entrada escrito en texto plano.

```
%Command:Question(Begin);
%Command:QuestionIntro(Begin);
Contenido de la pregunta
%Command:QuestionIntro(End);
       %Command:QuestionPossibility(Begin)(Exp='Explicación')(Correct='F');
       Contenido de la respuesta
       %Command:QuestionPossibility(End);
       %Command:QuestionPossibility(Begin)(Exp='Explicación')(Correct='F');
       Contenido de la respuesta
       %Command:QuestionPossibility(End);
       %Command:QuestionPossibility(Begin)(Exp='Explicación')(Correct='F');
       Contenido de la respuesta
       %Command:QuestionPossibility(End);
       %Command:QuestionPossibility(Begin)(Exp='Explicación')(Correct='T');
       Contenido de la respuesta
       %Command:QuestionPossibility(End);
%Command:QuestionImage(Begin)(Image='Ruta Imagen');
%Command:QuestionImage(End);
%Command:QuestionResource(Begin)(Resource='Ruta Recurso');
%Command:QuestionResource(End);
%Command:Question(End);
```

Figura 4.2: Ejemplo de una pregunta procesada en un archivo TXT

- *Documento XML:* El software del proyecto además de poder recibir un documento en texto plano o escrito en lenguaje LaTeX es capaz de tratar directamente un archivo generado en un procesamiento anterior. Esto permite almacenar estos documentos y modificarlos.

El proceso de análisis de cada uno de ellos es diferente. En la figura 4.3 se puede observar cual es el ciclo de procesado que sigue cada uno de ellos.

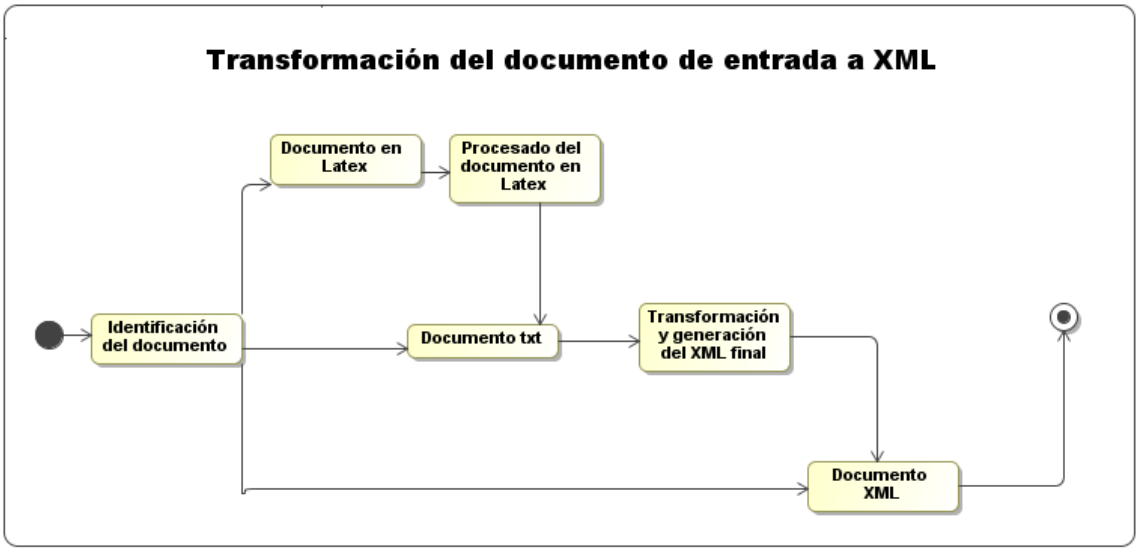


Figura 4.3: Proceso de Análisis del documento de entrada

Para la realización de este proceso se ha diseñado un programa escrito en *Python*. Es importante describir el programa así como cada uno de sus argumentos de entrada.

- ***parserXML***

- *Usage: parserXML.py [options]*

  Options:

```
-h, --help            show this help message and exit

-t INPUT_FILE_TEX, --tex=INPUT_FILE_TEX
                      Input .tex file.With commands tags

-x INPUT_FILE_XML, --xml=INPUT_FILE_XML
                      Input .xml file. With commands in xml

-r INPUT_FILE_OTHER, --raw=INPUT_FILE_OTHER
                      Input .* file. With commands tags

-o OUTPUT_FILE, --output=OUTPUT_FILE
                      Output .* file.

-v, --verbose         Verbose mode
```

Cada una de las opciones posibles se define a continuación:

- *-h*: El argumento *h* muestra la ayuda. En la ayuda podemos observar los posibles argumentos de entrada y una breve descripción de cada una de ellas.

- *-t*: El argumento *t* seguido de un nombre de fichero se utiliza para indicar al analizador que debe tratar el fichero como un fichero escrito en LaTeX. Este fichero debe haber sido marcado previamente con las órdenes o mandatos definidos en el pseudo-lenguaje del capítulo anterior.

- *-x*: El argumento *x* seguido de un nombre de fichero se utiliza para indicar al analizador que debe tratar el fichero como un fichero XML, analizando la correcta

composición de su estructura.

- *-r*: El argumento *r* seguido de un nombre de fichero se utiliza para indicar al analizador que debe tratar el fichero como un fichero en texto plano. Este fichero debe haber sido marcado previamente con las órdenes o mandatos definidos en el pseudo-lenguaje del capítulo anterior.

- *-o*: El argumento *o* seguido de un nombre de fichero se utiliza para indicar al analizador el nombre que se desea añadir al archivo XML de salida. En caso de omitir este argumento el archivo de salida tendrá el mismo nombre que el archivo de entrada especificado con un argumento *-t*,*-x* u *-r*.

- *-v*: El argumento *v* provoca que durante el proceso de análisis de los documentos se muestren ciertos mensajes relacionados con dicho proceso.

4.1.1. Preprocesado del documento de entrada

Independientemente del tipo de documento de entrada todos siguen un proceso de transformación que permite manejar su contenido de manera uniforme, creando un flujo de datos desde la entrada hasta la salida en forma de fichero XML. Podemos observar este flujo en la figura 4.3. No obstante cada tipo de archivo sigue una evolución distinta hasta llegar a un XML común.

Esta fase, es la primera del procesado y se centra en dos aspectos fundamentales: Detección de errores y, preparación del documento escrito en LaTeX para su análisis con las herramientas Tralics.

Detección de errores:

- *Lenguaje de marcado:* En caso de detectar un error en el lenguaje de marcado es necesario detener el proceso de análisis, notificando el error con la mayor precisión posible. Algunos de estos errores pueden ser: Error en la sintaxis de alguna orden, ausencia de alguna orden básica, etc.

- *Tipo de archivo:* En caso de introducir un archivo que no cumpla las características de los documentos de entrada permitidos, es necesario detener el proceso de análisis notificando el error con la mayor precisión posible.

Preparación del documento escrito en LaTeX para su análisis con la herramienta Tralics:

- *Tratamiento de las órdenes:* Tralics debe interpretar las órdenes escritas en el pseudo-lenguaje descrito. Para ello es necesario realizar un marcado de estas ordenes como si fueran comentarios especiales.
- *Información adicional:* La información contenida en etiquetas como *Exp* en la orden *QuestionPossibility* puede contener información en LaTeX y debe ser tratada con Tralics.

4.1.2. Tratamiento del documento con Tralics

Una vez verificada la validez del documento de entrada y realizado el análisis previo, es necesario tratar el documento según el tipo de que se trate.

En caso de identificar un documento de entrada en formato LaTeX es necesario utilizar la herramienta Tralics para generar un documento con la información en un XHTML + MathML, debido a que Tralics es una herramienta muy completa y posee multitud de opciones, las cuales pueden encontrarse en la pagina de Tralics:

`http://www-sop.inria.fr/marelle/tralics/options.html`

Argumentos indicados al ejecutar Tralics:

$$\text{tralics Documento.tex -shell-escape –noxmlerror -utf8 -utf8output -output}_{dir.} > /dev/null$$

- *-shell-escape:* Permite ejecutar órdenes y scripts dentro de un documento escrito en latex. Puede ser útil si es necesario realizar alguna tarea concreta al procesar el documento LaTeX.

- *–noxmlerror:* Evita generar en el documento de salida elementos `<error>`. Estos elementos son útiles cuando el comando se utiliza por separado para identificar el error dentro del árbol XML. Pero en nuestro proyecto en el caso de que el documento escrito en LaTeX no puediese ser procesado indicaríamos que es un documento LaTeX mal formado sin generar el árbol completo.

- *–utf8:* Indica a Tralics que el documento de entrada será escrito por defecto en utf8 excepto si se indica en la primera línea que la codificación es "iso-8859- 1". En ese caso se tratará como Latin1.

- *–utf8output:* Indica a Tralics que el archivo de salida debe ser codificado utilizando utf8.

Una vez analizado el documento con el *software* Tralics, es necesario realizar un análisis del documento XHTML + MathML generado, con la intención de detectar posibles errores. Debido a que este archivo generado no tiene aún una estructura que permita su clasificación se utilizarán las órdenes para dotarle de esta estructura.

Detección de errores:

- *Error en la sintaxis:* Si existe un error en el documento escrito en LaTeX es el punto del proceso en el que se detecta. Es necesario detener el proceso de análisis, notificando el error con la mayor precisión posible.

- *Error de compatibilidad:* Pese a que Tralics maneja la mayoría de paquetes utilizados en LaTeX es posible que exista algún problema de compatibilidad con alguno de ellos en concreto. Es necesario detener el proceso de análisis, notificando el error con la mayor precisión posible.

Contenido del documento:

- *Órdenes del pseudo–lenguaje:* Se conservan las órdenes insertadas en la fase previa para crear el documento final XML en una fase posterior.

- *Texto del documento:* Es necesario conservar todo el contenido original

  del documento: párrafos, saltos de linea.

- *Contenido adicional*: Se conservan las fórmulas generadas en MathML y elementos de formato como negrita, cursivas o subrayado.

4.1.3. Documento XML

El análisis de un documento con Tralics y el procesado posterior generan un documento con el mismo formato que un documento de entrada en texto plano marcado con el pseudo-lenguaje. Es por tanto, el momento de transformar las órdenes insertadas en un documento XML de forma que todo el contenido de los ficheros adquiera una estructura procesable por una librería de lectura de documentos XML como DOM o SAX.

El proceso de transformación consiste en traducir estas órdenes de pseudo-lenguaje a etiquetas XML:

1. ***Document:***
   - `%Command:Document(Begin);` → `<Document>`
   - `%Command:Document(End);` → `<\Document>`
2. ***Group:***
   - `%Command:Group(Begin);` → `<Group>`
   - `%Command:Group(End);` → `<\Group>`
3. ***GroupIntro:***
   - `%Command:GroupIntro(Begin)(Type='')(Desc='');` → `<GroupIntro Type='' Desc=''>`
   - `%Command:GroupIntro(End);` → `<\GroupIntro>`
4. ***Question:***
   - `%Command: Question (Begin);` → `<Question>`
   - `%Command: Question (End);` → `<\Question>`
5. ***QuestionIntro:***

- %Command:QuestionIntro(Begin); → <QuestionIntro>
- %Command:QuestionIntro(End); → <\QuestionIntro>

6. ***QuestionPossibility:***

- %Command:QuestionPossibility(Begin)(Exp='Explicacion')(Correct='F'); → <QuestionPossibility Correct='F' Exp='Explicacion'>
- %Command:QuestionPossibility(End); → <\QuestionPossibility>

7. ***QuestionImage:***

- %Command:QuestionImage(Begin)(Image='Ruta Imagen'); → <QuestionImage Image='Ruta Imagen'>
- %Command:QuestionImage(End); → <\QuestionImage>

8. ***QuestionResource:***

- %Command:QuestionResource(Begin)(Resource='Ruta Recurso'); → <QuestionResource Resource='Ruta Recurso'>
- %Command: QuestionResource (End); → <\QuestionResource>

Este documento XML está definido por el documento de definición de tipos, DTD, de la figura 4.4

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT QuestionImage (#PCDATA)>
<!ATTLIST QuestionImage
Image CDATA #IMPLIED
>
<!ELEMENT QuestionResource (#PCDATA)>
<!ATTLIST QuestionResource
Resource CDATA #IMPLIED
>
<!ELEMENT QuestionPossibility (#PCDATA)>
<!ATTLIST QuestionPossibility
Correct (T | F) #REQUIRED
Exp CDATA #IMPLIED
>
<!ELEMENT QuestionIntro (#PCDATA)>
<!ELEMENT Question ((QuestionIntro, QuestionPossibility+,
QuestionImage+, QuestionResource+))>
<!ELEMENT GroupIntro (#PCDATA)>
<!ATTLIST GroupIntro
Desc CDATA #IMPLIED
Type CDATA #IMPLIED
>
<!ELEMENT Group ((GroupIntro, Question+))>
<!ELEMENT Document (#PCDATA | Group)*>
```

Figura 4.4: Documento de definición de tipos del XML Final

La última fase del proceso consiste en analizar este documento XML para introducirlo en una Base de Datos.

4.2. Análisis del documento XML

Una vez generado el documento XML final se debe analizar para clasificar su contenido en la base de datos. Para ello se ha desarrollado un programa en Python que se encarga de realizar este proceso. El proceso de análisis se lleva acabo utilizando una librería DOM. Esta librería la estructura del archivo completamente en memoria y posteriormente se realiza su lectura.

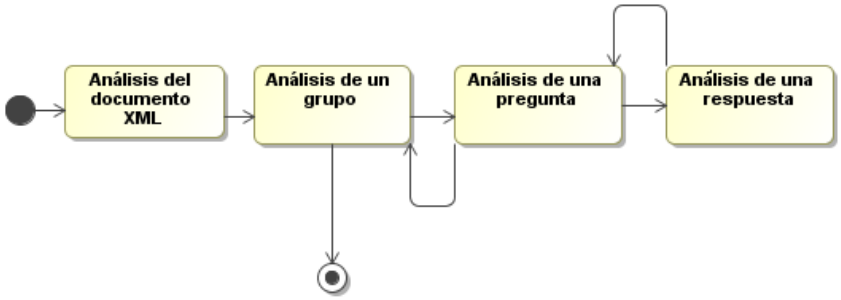


Figura 4.5: Diagrama del análisis del documento XML final

El proceso se realiza de manera iterativa analizando cada uno de los elementos del documento. El pseudo-código utilizado en el proceso es el descrito en la figura 4.6

Cada uno de los bucles corresponde con cada uno de los elementos principales que componen los documentos:

- *Grupo*: Cada uno de los grupos se almacena utilizando un identificador único asociado a su tipo. Junto con este identificador se almacena la descripción del grupo en caso de existir. Para cada una de las preguntas asociadas a dicho grupo se almacena una relación entre el grupo y la pregunta en una tabla independiente creada para tal fin.

- *Pregunta*: Cada una de las preguntas contienen un identificador único que permite relacionarlas con el grupo al que pertenecen. Con cada pregunta se almacena también su enunciado y en caso de tener asociados recursos o imágenes también se utiliza el identificador para relacionarlo con la pregunta.

```
AnalisisXML
    CargarDocumento
    Para cada Grupo
       ObtenerTipo
       ObtenerDescripción
       Para cada Pregunta
            ObtenerEnunciado
            Para cada Respuesta
                 ObtenerEnunciado
                 ObtenerExplicacion
                 AlmacenarRespuesta
       AlmacenarPregunta
  AlmacenarGrupo
```

Figura 4.6: Pseudo-codigo del programa de análisis

- *Respuesta*: Cada respuesta también tiene un identificador único que permite relacionarla con la pregunta a la que pertenece. Al mismo tiempo se almacena su enunciado, explicación y si es correcta o no.

4.3. Servicio Web de introducción de datos

Una vez realizado el software encargado de transformar y analizar los documentos de entrada, se ha tratado de dotar al proyecto de una mayor funcionalidad creando una herramienta web. Esta herramienta web permite unificar las funcionalidades y los datos del proyecto así como facilitar su uso desde cualquier sistema operativo. Para hacer uso del software desarrollado, la plataforma ofrece la funcionalidad a través de una serie de llamadas HTTP que facilitan el manejo del software utilizando un navegador web.

El *framework* o plataforma de desarrollo web elegida ha sido Django Python. Al desarrollar todo el código sobre esta plataforma y utilizar Python como lenguaje en todo el proyecto, se puede integrar de manera natural el sistema software de análisis generado

hasta el momento, con una serie de funcionalidades adicionales.

4.3.1. Patrón MVC en nuestro sistema

Mediante Django se puede utilizar un patrón de arquitectura de software MVC (Modelo Vista Controlador) que permite separar los datos de una aplicación, la interfaz de usuario y la lógica de negocio en tres componentes distintos. En el proyecto actual estos tres componentes quedan claramente delimitados.

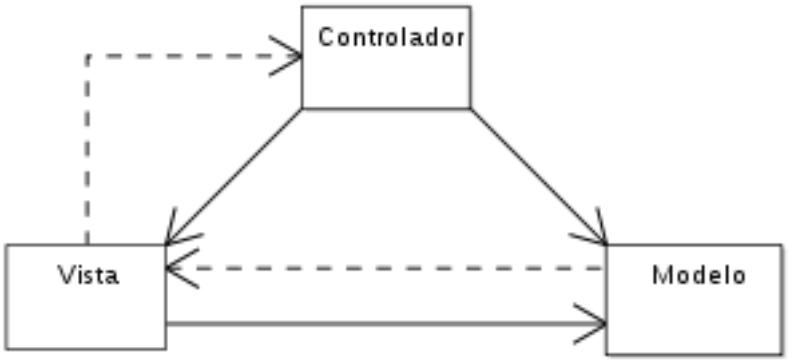


Figura 4.7: Imagen del MVC

1. *Vista:* En el patrón MVC, la vista tiene la función de presentar una interfaz con la que el usuario pueda interacturar. En nuestro sistema está representada por cada uno de los formularios HTML presentados al usuario y por cada una de las paginas de resultado generadas tras la ejecución de alguno de los métodos.

   El usuario interacciona con el sistema utilizando el navegador web, desde este el usuario tiene acceso a toda la funcionalidad que ofrece el sistema. Esta funcionalidad y cada una de las vistas que ofrece el sistema serán definidas más adelante.

2. *Controlador:* En el patrón MVC, el controlador tiene la función de recibir los eventos producidos en la vista y realizar una acción. En nuestro sistema el controlador queda definido por la lógica de Django que permite asociar URL's de llamada a recursos con funcionalidad asociada a esa llamada.

   Para ello Django utiliza el archivo urls.py donde se definen cada una de las llamadas http que el sistema puede manejar. En este archivo se define una correspondencia

entre la llamada http y la función encargada de manejar la lógica de la llamada. Estas funciones se encuentran definidas en el fichero views.py.

3. *Modelo:* En el patrón MVC, el modelo se encarga de administrar la lógica de negocio y la interacción con los datos de la aplicación. En el sistema implementado representa al proceso de análisis e inserción en la base de datos.

   Django proporciona un mecanismo de manejo de la base de datos modelo-relacional que permite manejar las tablas de la base de datos como si fueran clases de objetos. Django se encarga de almacenar los datos de los atributos de la clase en los campos de la tabla en SQL. Pese a que Django permite este tipo de manejo de la base de datos, en el proyecto se ha desarrollado una clase encargada de manejar la base de datos ejecutando directamente una serie de sentencias SQL sobre la base de datos. Quedará propuesto en la sección de trabajos futuros adaptar la clase encargada del manejo de la base de datos a la filosofía propuesta por Django para evitar el hecho de tener que ejecutar las sentencias de la base de datos directamente.

4.3.2. Funcionalidad implementada en el sistema

Ese sistema de introducción de datos ofrece al profesor un entorno donde manejar todas las herramientas software creadas a lo largo del proyecto, el sistema no solo permite introducir datos en la base de datos sino que también permite verificar la correcta estructura de marcas del archivo antes de realizar su inserción en la base de datos o mostrar una vista previa de como quedará el documento una vez analizado.

Las funcionalidades concretas del sistema se enumeran a continuación:

- *Estructurar un documento:* El sistema de introducción se encarga de proporcionar una interfaz de introducción de documentos on-line. Consta de una pagina web con un formulario, dicho formulario permite introducir los datos de los ficheros de manera sencilla insertando las diferentes partes del documento en distintos campos del formulario.

  Una vez completado el formulario se envía al servicio web, que se encarga de clasificar

el contenido e insertar las marcas del pseudo–lenguaje realizar el análisis y mostrar una pagina XHTML con el resultado del análisis.

- *Verificar un documento:* El sistema de introducción permite comprobar si un documento marcado con el pseudo–lenguaje implementado tiene una estructura correcta para su análisis.

  El servicio web recibe un documento de entrada, lo analiza e indica si se encuentra en un estado óptimo para el análisis e inserción en la base de datos. En caso contrario trata de localizar el error y mostrarlo al usuario de manera descriptiva.

- *Analizar un documento LaTeX:* El sistema de introducción se encarga de analizar un documento LaTeX marcado con el pseudo–lenguaje implementado.

  En caso de analizar correctamente el documento devuelve un archivo XHTML que permite una visualización del resultado del análisis en el navegador.

- *Analizar un documento en texto plano:* El sistema de introducción se encarga de analizar un documento TXT marcado con el pseudo–lenguaje implementado.

  En caso de analizar correctamente el documento devuelve un archivo XHTML que permite una visualización del resultado del análisis en el navegador.

- *Analizar un documento XML:* El sistema de introducción permite utilizar un documento XML generado en un análisis previo para evitar analizar el documento.

  En caso de analizar correctamente el documento devuelve un archivo XHTML que permite una visualización del resultado del análisis en el navegador.

- *Insertar un documento en la base de datos:* El sistema de introducción de datos analiza el contenido del documento insertado. Si el análisis se produce de forma correcta, se insertan los valores en la base de datos. En el navegador se muestra un documento XHTML como en los casos previos.

Capítulo 5

Resultados y pruebas

Tras el desarrollo del software descrito en los capítulos anteriores, ha sido necesario comprobar su correcto funcionamiento, detectando posibles errores o mejoras que permitan orientar la evolución del proyecto.

5.1. Análisis del documento

La necesidad de realizar un análisis sobre un documento escrito utilizando lenguaje de formato como es LaTeX hace indispensable diferenciar la información que se desea transmitir (contenido), de cómo se desea mostrar (formato).

Adicionalmente, era necesario clasificar la información contenida en dicho fichero según una estructura definida a priori. Por lo tanto, para desarrollar el método final descrito en capítulos anteriores, ha sido necesario realizar una serie de pruebas y análisis a lo largo de todas las fases del proyecto.

A continuación se describen los resultados y pruebas relacionados con la selección del traductor de LaTeX la selección del pseudo-lenguaje y los documentos de prueba sobre los que se ha trabajado.

5.1.1. Documentos de prueba

Para la realización de todas las pruebas relacionadas con el proceso de análisis ha sido necesario utilizar una serie de documentos escritos en LaTeX que permitan trabajar con un entorno que se asemeje en la mayor medida posible al entorno final donde se utilizará el proyecto.

Para ello se han empleado pruebas de evaluación reales, utilizadas en años anteriores. Esto ha permitido adaptar el proyecto a un entorno real y generar un patrón que pueda ser utilizado en el caso general.

5.1.2. Selección del traductor LaTeX

Durante el capítulo de introducción se han presentado diferentes herramientas relacionadas con el análisis de documentos escritos en formato LaTeX. Estas herramientas han sido evaluadas para finalmente tomar la decisión acerca de cual se debe seleccionar:

- *LaTeXML:* LaTeXML permite generar documentos XML pero hace necesario utilizar una serie de hojas de estilo XSLT para generar una salida correcta. Estas

hojas de estilo varían entre los diferentes documentos LaTeX analizados, según los paquetes utilizados. Es por tanto una limitación, que se debería franquear.

- *Tex4ht:*Tex4ht , es uno de los analizadores más extendidos. Permite generar a partir de un documento LaTeX, un documento en multitud de formatos HTML, XML, XHTML+MathML. Es necesario incluir en el documento de origen el paquete `usepackage[html]{tex4ht}` y en ocasiones genera diversos ficheros de salida que es necesario relacionar. Es una de las herramientas más completas de las analizadas pero a su vez una de las más lentas y complejas de utilizar.

- *Tralics:* Tralics es una herramienta para generar únicamente XML + MathML, esto la convierte en una herramienta muy especializada. Dispone de un uso ágil y rápido lo que la convierte en una gran alternativa para el propósito de este proyecto.

- *LXir:* LXir es un analizador de documentos DVI, es capaz de generar documentos HTML a partir de dichos DVI, pero todavía se encuentra en una fase temprana de desarrollo y por ello posee una gran incompatibilidad con ciertos paquetes del lenguaje LaTeX. Pese tener un futuro prometedor y realizar alguna prueba con la herramienta, la barrera de la compatibilidad la convierte en una herramienta poco apropiada para este proyecto.

- *Hermes:* Hermes es una herramienta de conversión de archivos DVI a diversos formatos, HTML, XHTML+MathML, etc. Pese a su versatilidad, posee limitaciones similares a las proporcionadas por Lxir.

Tras analizar las diferentes herramientas expuestas anteriormente ha sido necesario tomar una decisión sobre cual utilizar. En este proyecto la herramienta utilizada para realizar las transformaciones ha sido **Tralics**.

La elección final se ha tomado en función de diversos criterios:

- *Versatilidad:* La herramienta debe ser capaz de configurar ciertos aspectos tanto de la entrada como de la salida. En este sentido Tralics permite dicha configuración.

- *Claridad:* El documento obtenido debe ser un documento lo más "*legible*" posible. Una herramienta genera documentos legibles cuando no introduce elementos que no

estuvieran incluidos en el documento original. Tralics puede ser configurado para omitir en su salida ciertos elementos del documento original.

- *Progresión:* La herramienta debe formar parte de un proyecto de desarrollo activo, adicionalmente debe tratarse de un proyecto maduro con un largo periodo de pruebas que certifiquen su calidad. En este aspecto la herramienta elegida forma parte de un proyecto iniciado en 2003 y su última versión fue publicada en 2011.

- *Código abierto:* La herramienta debe pertenecer a un proyecto de código abierto donde exista la posibilidad de acceder al código. Esto puede permitir adaptarla a diferentes entornos junto con el proyecto general. Tralics pertenece a un proyecto de código abierto y permite modificar su código fuente.

- *Velocidad:* La herramienta debe generar una salida en el menor tiempo posible. En caso de introducir el análisis en el servicio web, será necesario realizar un análisis ágil que devuelva una traducción en un breve periodo de tiempo. Tralics es la herramienta, de las analizadas, que genera una salida en el menor tiempo.

Una herramienta que podría remplazar a Tralics en un futuro sería Tex4ht, herramienta que cumple los requisitos de manera similar a Tralics.

5.1.3. Selección del pseudo-lenguaje

Como ya se ha mencionado en numerosas ocasiones a lo largo del presente documento, la finalidad del proyecto consistía en clasificar de manera estructurada el contenido de diferentes documentos. Por lo que ha sido necesario comprender cuál es el significado de cada uno de los párrafos de los que consta el documento.

Tras seleccionar la herramienta de transformación de documentos LaTeX a XML se han realizado numerosos experimentos y pruebas. A continuación se enumeran las dos pruebas más relevantes:

- *Analizar el documento según su organización:* En el documento XML generado se ha tratado de buscar correspondencias entre párrafos y el contenido que contienen.

Es equivalente a tratar de identificar el enunciado de una prueba de evaluación, por encontrarse en todos los documentos situada de la misma manera y con el mismo formato.

Realizar el análisis del documento utilizando este método permitió identificar la heterogeneidad de los documentos de entrada. No todos los enunciados se encontraban estructurados de la misma manera, por ejemplo, diferentes espacios entre preguntas, diferentes elementos dentro del enunciado o carencia de enunciado.

- *Analizar el documento marcando ciertos puntos:* Ciertas partes en el documento de entrada se marcaron con palabras clave que tras el análisis con la herramienta de transformación, eran localizados y permitían delimitar los contenidos.

  Este método permitía delimitar los contenidos de las pruebas de evaluación de manera correcta, permitiendo eliminar la barrera de la heterogeneidad de los documentos de entrada en su estructura.

De los dos métodos de análisis realizados, el segundo método aportaba un número de ventajas mayor que el primero. Aun así era necesario perfeccionar las marcas introducidas en el documento por ello se ha implementado el lenguaje de marcas definido en los capítulos anteriores. Para su finalidad se ha diseñado el lenguaje definido previamente en el documento.

5.1.4. Selección del sistema de representación de formulas matemáticas

Las pruebas de evaluación analizadas pueden contener formulas matemáticas. Para representarlas de manera estándar se ha utilizado MathML, no obstante se han realizado numerosas pruebas con distintos navegadores.

Una página web que contiene elementos MathML es un documento compuesto que contiene tanto elementos XHTML como MathML. El tipo del documento tiene que ser por tanto XHTML 1.1 + MathML 2.0 y se debe servir al navegador con el tipo MIME application/xhtml+xml.

Normalmente los servidores sirven los documentos que tienen la extensión .html con el tipo MIME text/html y los documentos que tienen la extensión .xhtml con el tipo MIME application/xhtml+xml, por lo que conviene guardar los documentos que incluyan elementos MathML con la extensión .xhtml.

Se puede comprobar fácilmente la capacidad de un navegador mostrando elementos MathML visitando la página `https://www.eyeasme.com/Joe/MathML/MathML_browser_test`. Esta página muestra fórmulas matemáticas representadas en diversos formatos: el primero de ellos muestra la imagen utilizando un sistema de tipografía TEX y, el segundo, utiliza la representación de MathML en Firefox con las fuentes STIX instaladas.

En los navegadores que no son capaces de mostrar elementos MathML, las fórmulas matemáticas deben ser representadas como imágenes, lo que impide aprovechar las posibilidades de MathML (hojas de estilos, zoom, facilidad de edición, etc.). Algunos sitios detectan si el navegador admite o no MathML y envían una versión con MathML o con imágenes, como por ejemplo la Biblioteca Digital de Fórmulas Matemáticas del NIST[1].

Debido a que ciertos navegadores no implementan de forma nativa el soporte para MathML, y éste será utilizado en el proyecto, se ha considerado necesario evaluar las limitaciones de cuatro de los navegadores más utilizados del momento. Según las estadísticas obtenidas de *StatCounter* estos navegadores son los siguientes: Internet Explorer, Google Chrome, Firefox y Safari. En la figura 5.1 podemos observar la evolución del uso de estos navegadores en el tiempo.

Estado de MathML en los navegadores:

- *Firefox:* Firefox es capaz de mostrar páginas con elementos MathML, aunque es necesario instalar ciertas fuentes para poder mostrar correctamente todos los caracteres matemáticos.

  Actualmente, las fuentes recomendadas son las fuentes STIX, una fuentes de miles de caracteres creadas por el consorcio STI Pub, formado por varias organizaciones científicas norteamericanas y la editorial Elsevier. Estas fuentes son de uso libre y

[1] Instituto Nacional de Estándares y Tecnologías de los Estados Unidos `http://dlmf.nist.gov/`

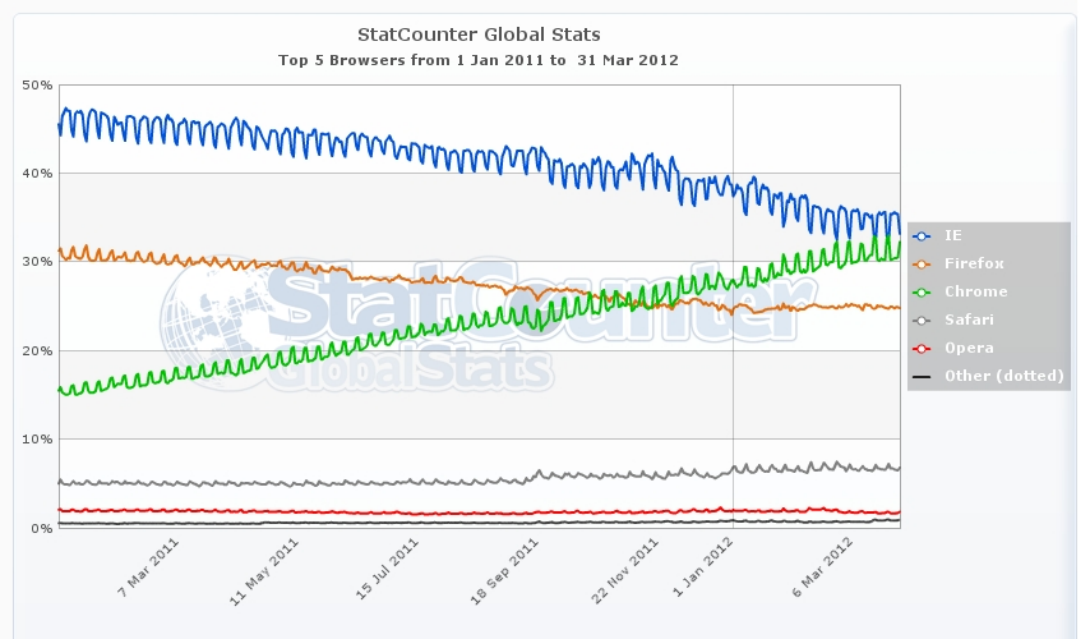


Figura 5.1: Evolución temporal del uso de los navegadores Web

pueden obtenerse en el siguiente enlace:
$http://www.cdlibre.org/consultar/catalogo/Fuentes_OTF.html$

- *Internet explorer:* Internet Explorer no admite el tipo MIME application/ xhtml+xml, por lo que no es capaz de mostrar directamente páginas con elementos MathML, pero existen plug-in gratuitos para conseguirlo, por ejemplo MathPlayer de DesignScience. Se ofrecen dos versiones de MathPlayer:
  - MathPlayer 2.2, de febrero de 2010, que funciona en Internet Explorer 8 y anteriores
  - MathPlayer 3.0, de diciembre de 2011, que funciona en Internet Explorer 9
- *Safari y Google Chrome:* Actualmente tanto Google Chrome como Safari utilizan un WebWit como motor del navegador el cual admite MathML desde agosto de 2010. Google Chrome tiene deshabilitada esta funcionalidad por lo que no es capaz de mostrar elementos MathML, se esta trabajando para que la siguiente versión del navegador este disponible con compatibilidad con MathML, por el contrario el navegador Safari 5.1 (publicado en julio de 2011) sí que tiene habilitadas todas las funcionalidades del Webkit por lo que tiene compatibilidad con MathML.

5.2. Servicio web

Una vez implementado el analizador y desarrollado el sistema web que permite el manejo de dicho analizador, el siguiente paso ha sido desarrollar una prueba que evalúe el proceso de análisis desde la introducción del documento hasta la correcta visualización del documento en una página web XHTML+MathML y correcto almacenamiento en la base de datos.

Se ha desarrollado una prueba utilizando un test de evaluación, cedido por el departamento de sistemas telemáticos, en el servicio web. Para realizar la prueba

Se han seguido una serie de pasos descritos a continuación:

1. Introducir los datos en el sistema utilizando uno de los modos propuestos en este caso se han utilizado 2 modos:

   a) El primero consiste en introducir el test de evaluación mediante el formulario de estructuración de documentos. En la figura 5.2 puede observarse como ha sido introducido dicho documento.

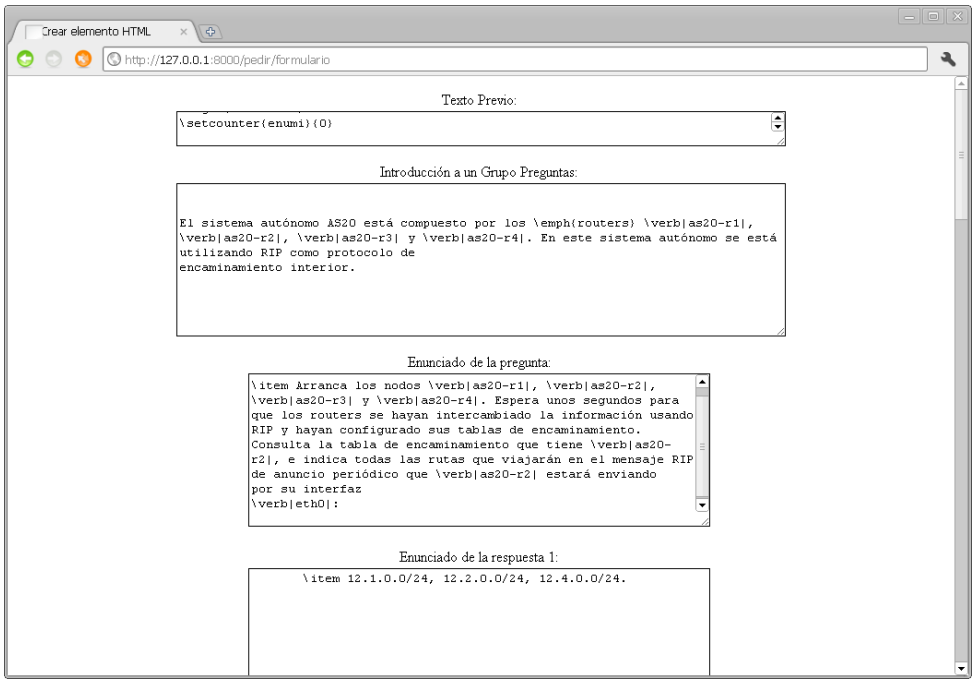


Figura 5.2: Formulario de entrada

b) El segundo modo consiste en introducir el pseudo–lenguaje diseñado directamente en el documento de entrada generando un documento que pueda ser analizado por el servicio web directamente. Quedando este documento marcado de forma similar al ejemplo mostrado en la figura 3.4 del capítulo 3

2. Generar el fichero de análisis intermedio con los datos introducidos en el primer paso. El resultado obtenido es el de la figura 5.3

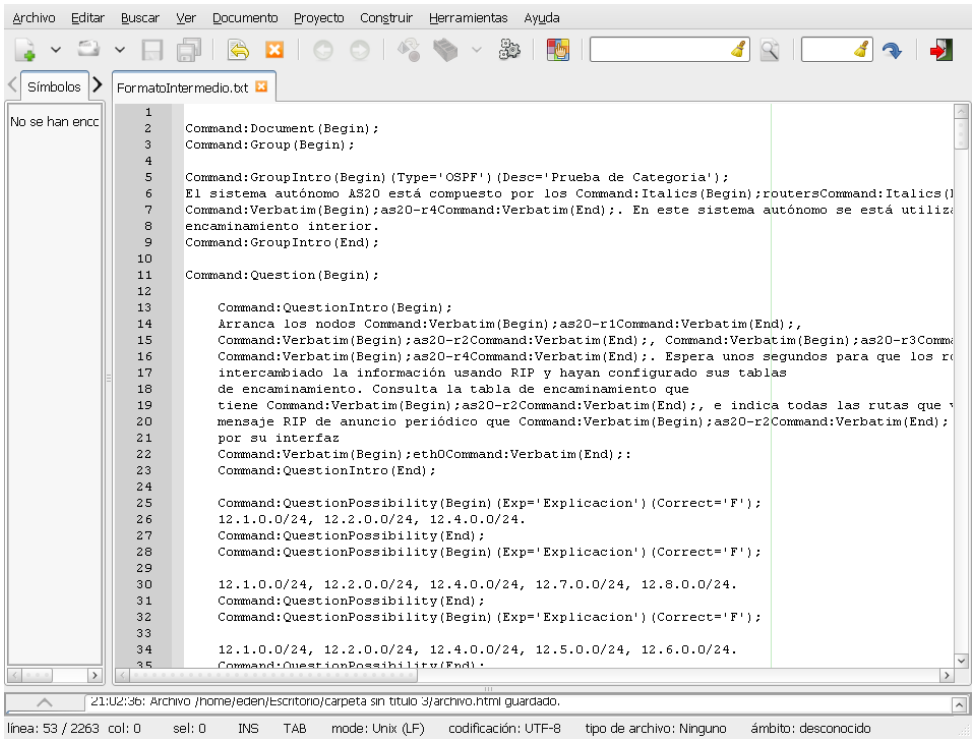


Figura 5.3: Documento en formato intermedio

3. Generar un fichero XML utilizando el estándar definido en la figura 4.4, el documento obtenido podemos verlo en la figura 5.4

4. Analizar el documento XML, generando finalmente un documento XHTML+MathML e insertando los datos contenidos en dicho documento en la base de datos como se puede observar en las figuras 5.5 y 5.6.

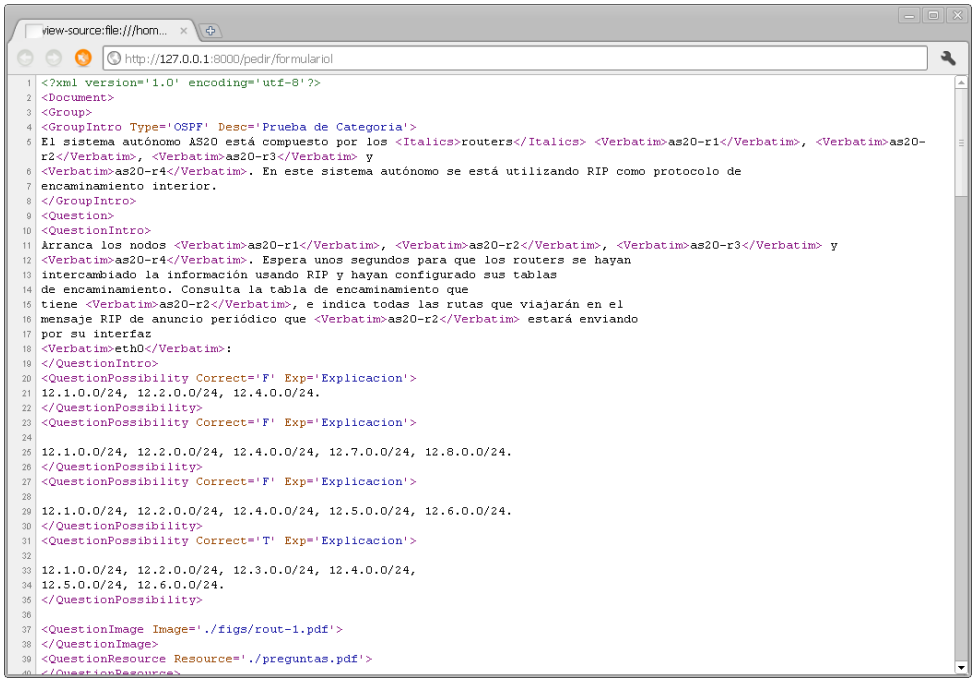


Figura 5.4: Documento en formato XML

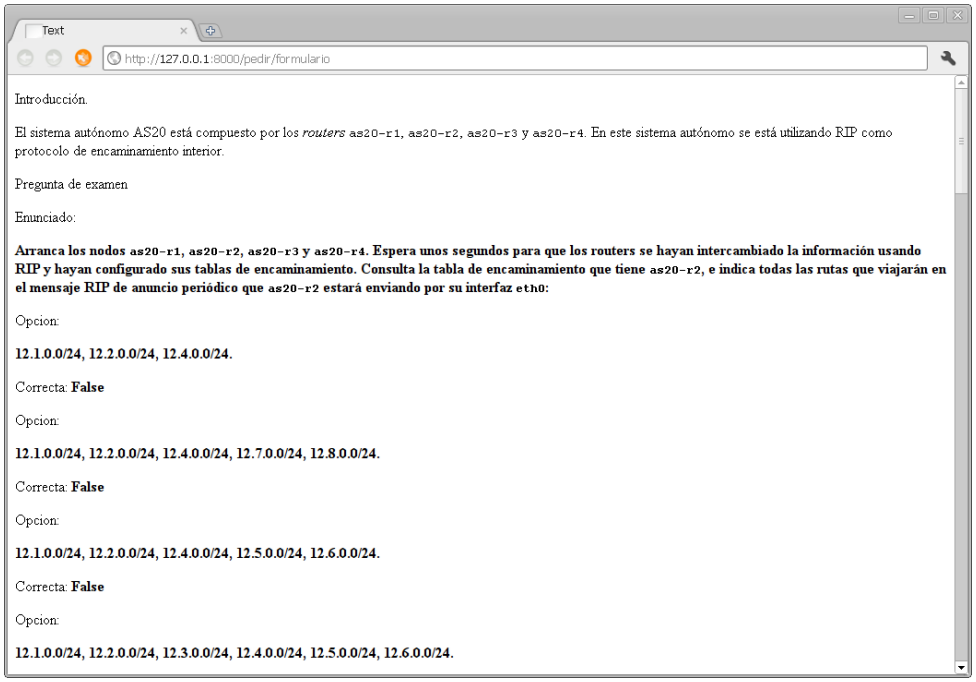


Figura 5.5: Vista preliminar de los datos analizados

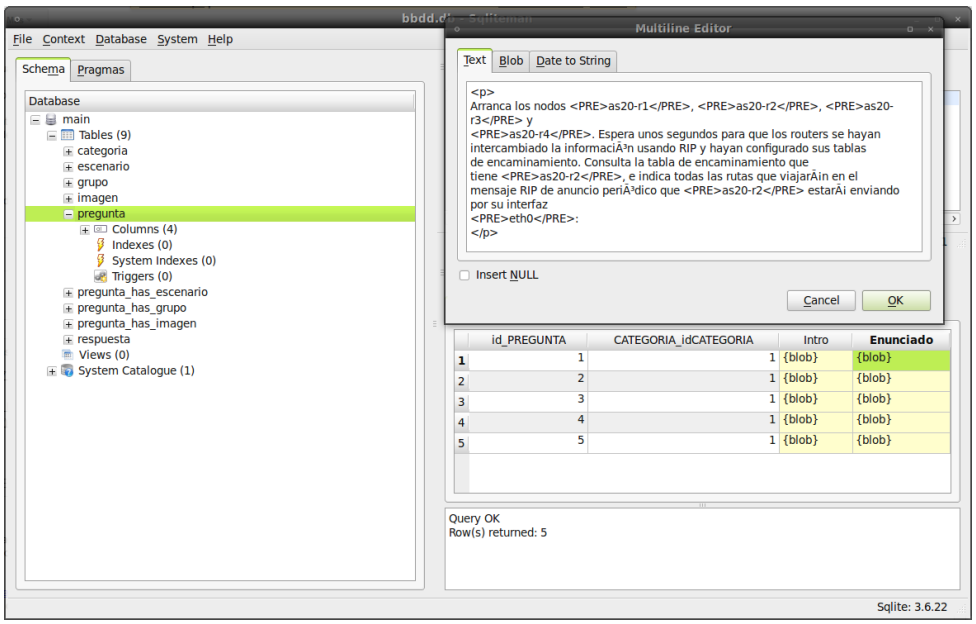


Figura 5.6: Fragmento del documento analizado en la base de datos

Capítulo 6

Conclusiones

En el inicio del presente documento se han enunciado una serie de objetivos, estos tienen como finalidad solucionar un problema. En capítulos posteriores se ha tratado de mostrar un análisis y diseño de cómo abordar el problema, junto con los métodos y las soluciones propuestas. En este capítulo se trata de evaluar el resultado obtenido después de ejecutar todo el proceso descrito con anterioridad, así como un breve análisis de los conocimientos adquiridos y un análisis que identifique posibles trabajos futuros.

6.1. Logros alcanzados

Los objetivos del proyecto estaban centrados en la creación de un software de unificación de datos a partir de documentos ya existentes y de diverso origen. Para la realización de este objetivo era necesario satisfacer 2 objetivos secundarios, unificación de los datos de entrada procedentes de diversos documentos con diversos formatos y unificación de la salida generando un modelo de datos que pueda ser utilizado posteriormente de forma automática. A lo largo del presente documento hemos podido comprobar cuáles han sido las medidas adoptadas para solucionar cada uno de los objetivos propuestos. Para realizar un análisis del estado de la solución propuesta, se describe el resultado de forma independiente:

- *Unificación de la entrada:* Proceso de unificar todos los documentos de entrada independizando el origen de los datos y su tipo. Para realizar este proceso ha sido necesario solucionar primero cuáles pueden ser los documentos de entrada válidos, desarrollar un estándar que permita tratarlos todos de la misma manera y facilitar al usuario un método de introducción de datos simple. Este problema, como se ha descrito a lo largo del capítulo anterior, ha sido solucionado utilizando un servicio web donde el usuario introduce los datos de sus documentos en un formulario dinámico que se adapta a la estructura y formato del documento de entrada. Esto permite una gran versatilidad, posibilitando una posterior ampliación del servicio web que permita nuevas entradas de datos.

- *Unificación de la salida:* Proceso de unificar los datos de entrada, generando un fichero de salida estándar e independiente del método utilizado para introducir los datos, así como una base de datos unificada que permita manejar todos los datos obtenidos independientemente de su documento de origen. Esta base de datos puede ser utilizada por un proceso posterior que, al tener los datos organizados y estructurados puede manejarlos con facilidad para cualquier posible finalidad. Algunas de estas finalidades se describen en el capítulo de trabajos futuros en este mismo documento.

Ambos objetivos han sido cumplidos y se ha establecido una serie de pautas que permiten su uso y adaptación a otro software distinto a este proyecto.

6.2. Conocimiento adquirido

Para el desarrollo del presente proyecto ha sido necesario hacer uso de los conocimientos adquiridos a lo largo de la titulación, además ha sido necesario adquirir ciertos conocimientos adicionales algunos de los cuales se enumeran a continuación:

- *Lenguajes de programación:* Ha sido necesario adquirir y utilizar conocimientos en varios lenguajes de programación el principal ha sido Python, pero también han sido necesarios: javascript, XML, HTML, etc.

- *Investigación:* Se han desarrollado habilidades en el campo de la investigación, esto ha sido necesario tanto para reconocer el estado del arte como para adquirir conocimientos en distintas tecnologías utilizadas para el desarrollo del proyecto.

- *Desarrollo de servicios web:* Se han adquirido conocimientos en el mundo del desarrollo de aplicaciones web. Django ha sido la opción elegida y por tanto se ha profundizado más en su aprendizaje.

- *Control de versiones:* El presente proyecto ha sido desarrollado utilizando una metodología en espiral, en cada ciclo de la espiral tenemos una versión con mayor funcionalidad. Estos ciclos han sido controlados utilizando GIT, un sistema de control de versiones. Así ha sido posible identificar cambios entre hitos y solucionar errores manteniendo siempre un desarrollo sólido.

- *Experiencia en desarrollo:* El desarrollo de un proyecto de gran magnitud ha permitido adquirir conocimientos y experiencia de cara a proyectos reales en un entorno laboral. Esto ha permitido diferenciar el desarrollo de una práctica en el entorno de una asignatura concreta, del desarrollo de un proyecto de propósito general donde la toma de decisiones influye en todos los aspectos del resultado final.

- *Escritura de documentos:* Dentro de los conocimientos adquiridos podemos enmarcar también la escritura de documentos científicos. Se ha adquirido habilidad utilizando la herramienta LaTeX.

Todos estos conocimientos han sido y serán de gran utilidad para un posible futuro tanto laboral como profesional.

6.3. Trabajos futuros

Este proyecto puede formar parte de un proyecto de mayor envergadura por lo que las posibilidades de ampliación son muy grandes. Se puede afirmar que este proyecto enmarca una primera parte de los posibles objetivos que se podrían alcanzar utilizando este como base. Algunos ejemplos pueden ser los siguientes:

- *Sistema de autoevaluación:* Como se ha enunciado a lo largo del proyecto, sobretodo en el capitulo de introducción y objetivos, este proyecto forma parte de un proyecto cuya finalidad es crear un sistema completo de autoevaluación, destinado al alumnado. Por tanto uno de los principales proyectos que se pueden abordar a partir del presente proyecto sería la creación este sistema de autoevaluación. Ya que el presente proyecto permite la recolección de datos a partir de documentos que contienen pruebas de evaluación, será posible la creación de un sistema software que permita el uso de los datos almacenados para generar nuevas pruebas como combinación de las almacenadas. Este sistema podría ofrecer diversas funciones para evaluar a un alumno o usuario del sistema, como la selección aleatoria de preguntas de la base de datos, selección de preguntas centradas en una temática concreta, etc. Ya que este sistema que se pretende desarrollar estaría basado en la interacción con el usuario, adicionalmente permitiría conocer la evolución académica de los usuarios del sistema, lo que ofrecería nuevas posibilidades de mejora de la docencia.

- *Sistema de evaluación móvil:* Además del sistema de autoevaluación mencionado anteriormente, otra posible línea de investigación y trabajo podría estar encaminada hacia la creación de pruebas de evaluación sobre dispositivos móviles. El auge de las

tecnologías móviles hace posibles nuevas formas de plantear la docencia, mejorando la eficiencia del aprendizaje. Por ejemplo, pruebas de evaluación colectivas evaluables de forma inmediata y sin mediación de un profesor, test de conocimientos al finalizar una clase o foros interactivos que mejoren las opciones actuales.

- *Sistema de creación de documentos:* El software del presente proyecto está adaptado para la clasificación del contenido e diversos documentos en una base de datos, pero también sería posible generar documentos en formato LaTeX o TXT con los datos almacenados.

6.4. Tiempo y esfuerzo

En esta sección se trata de dar una estimación del tiempo empleado al proyecto. Así mismo se trata de evaluar su progreso en el tiempo indicando cuales han sido los meses de mayor dedicación.

Cabe destacar que dicho proyecto ha sido desarrollado en conjunto con las asignaturas del Máster en Sistemas Telemáticos e Informáticos, esto explica la distribución de las horas de trabajo en el tiempo. Como se puede ver en la figura 6.1 ese tiempo está distribuido de forma uniforme entre los diferentes meses de los que consta el curso académico, a excepción de las épocas de exámenes y periodos previos a dichas épocas, donde se aprecia una disminución del tiempo invertido.

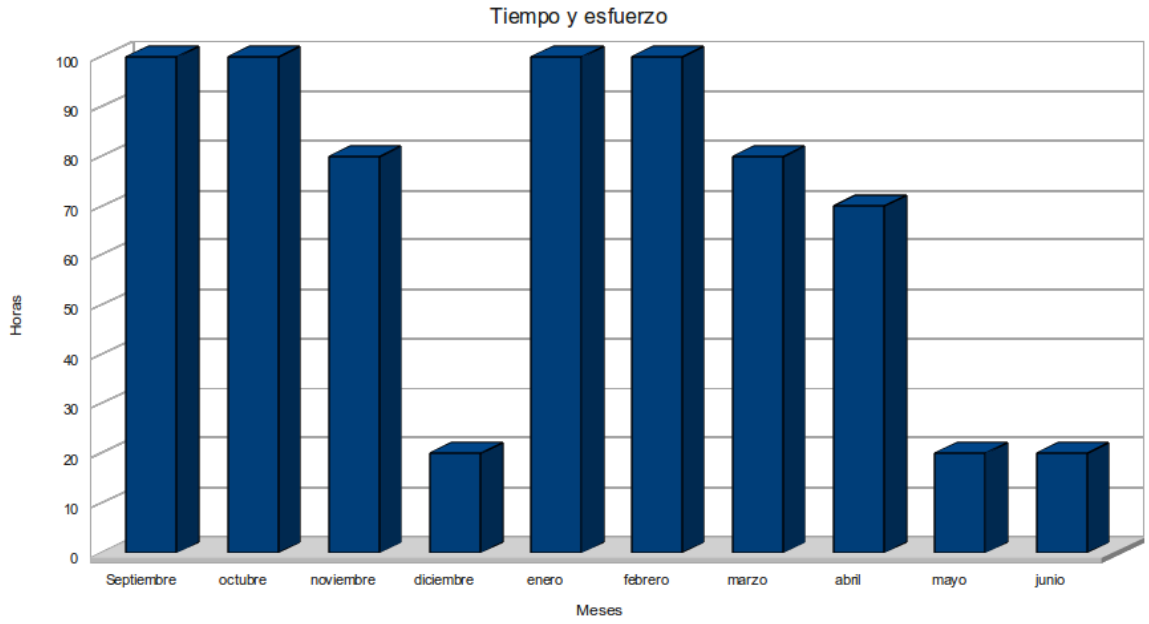


Figura 6.1: Tiempo y esfuerzo dedicado al proyecto

Bibliografía

[Corporation2012] Gradiance Corporation. Gradiance overview. http://www.gradiance.com/idea.html, Junio 2012.

[Duque2012] Raúl González Duque. Python para todos. http://mundogeek.net/tutorial-python/, Junio 2012.

[Foundation2012] The Python Software Foundation. The document object model api. http://docs.python.org/library/xml.dom.html, Junio 2012.

[geek2012] Mundo geek. Interactuar con webs en python. http://mundogeek.net/archivos/2008/04/15/interactuar-con-webs-en-python/, Junio 2012.

[Holovaty2010] Adrian Holovaty. La guía definitiva de django. http://www.anayamultimedia.es/cgigeneral/ficha.pl?id_sello_editorial_web=23&codigo_comercial=2315599#, Junio 2010.

[Marco2012] Bartolomé Sintes Marco. Mathml. http://www.mclibre.org/consultar/amaya/xhtml/xhtml_mathml.html, Febrero 2012.

[Pritchard2012] David Pritchard. Beginners guide. http://wiki.python.org/moin/BeginnersGuide, Junio 2012.

[Tralics2012] Tralics. Tralics: a latex to xml translator. http://www-sop.inria.fr/marelle/tralics/, Junio 2012.

[Ullman2012] Jeffrey D. Ullman. Gradiance on-line accelerated learning guide for authors. http://www.gradiance.com/downloads/auth-guide.pdf, Junio 2012.

[Valdivieso2012] Pedro A. Castillo Valdivieso. Dtd. http://atc.ugr.es/pedro/tutoriales/cursos/xml/dtd.htm, Junio 2012.

[w3c2012] w3c. What is mathml? http://www.w3.org/Math/, Junio 2012.

[Wikipedia2012] Wikipedia. Mathml. http://es.wikipedia.org/wiki/MathML, Junio 2012.