ÍNDICE

1. INTRODUCCIÓN

La evaluación ha sido siempre una parte importante del proceso de enseñanza y aprendizaje. Por una parte, los profesores necesitan conocer cual es el conocimiento que ha adquirido los alumnos tras el proceso de enseñanza para actuar en consecuencia, y por otra, los propios alumnos necesitan contrastar de manera no subjetiva el conocimiento adquirido. De igual manera en el campo de la Enseñanza Asistida por Ordenador, EAO y de los Sistemas Tutores Inteligentes (STI), especialmente en estos últimos, la evaluación ha sido siempre una parte importante.

Uno de los mecanismos de evaluación mas extendidos, por su facilidad de corrección es la realización de tests. La realización de tests tiene la ventaja de sistematizar la evaluación, por lo que ha sido ampliamente usada en aplicaciones de enseñanza asistida por ordenador y en sistemas tutores inteligentes. En los sistemas de enseñanza tradicional, el uso de tests de papel y lápiz también tiene sus inconvenientes: las preguntas del test son en general las mismas para todos los alumnos, el numero de preguntas tampoco varia, el tipo de preguntas se limita a unas o varias opciones, o a una respuesta corta, etc. Por otra parte la evaluación que se obtiene tras la realización de un test mediante papel y lápiz raramente tiene en cuenta mas que el numero de respuestas acertadas, y no suelen tener en cuenta la variabilidad en la dificultad de las preguntas, ni otros factores como la probabilidad de acertar una pregunta al azar, sin realmente saber la respuesta.

En el campo de psicometría, en el que la realización de tests ha sido siempre tradicional, surge en los años 1960s-1970s la Teoría de Respuesta al Ítem con el objeto de mejorar la evaluación y cuantificar de manera probabilista los resultados del test y los posibles errores de evaluación debidos al azar. En la teoría de respuesta al ítem se asume que el conocimiento del alumno puede medirse mediante una única variable denominada rasgo. A partir de las respuestas del alumno a la secuencia de preguntas del test y mediante un procedimiento estadístico que tiene en cuenta la dificultad de las preguntas se obtiene un estimador de este rasgo, a partir de la función de distribución de las probabilidades de que el conocimiento real del alumno tome determinado valor.

La teoría de Tests Adaptativos Informatizados (TAI), evoluciona a principios de los años 1980s, en general basándose en la aplicación mediante ordenador de teoría de respuesta al ítem. Básicamente el objeto de esta nueva teoría es mejorar la evaluación mediante dos mecanismos: la selección de las preguntas mas adecuadas a cada alumno según su nivel de conocimiento estimado en cada momento y la realización del mínimo numero de preguntas necesarias para garantizar una cota superior del error de estimación.

Por otra parte, en la ultima década, los sistemas de enseñanza asistida por ordenador y los sistemas tutores inteligentes han hallado en Internet, y mas concretamente en la World Wide Web, WWW un medio ideal de difusión. De hecho, ya son muchos los centros de enseñanza y universidades que incluyen en sus portales al menos el soporte para la difusión de contenidos y material docente. A este tipo de enseñanza se le domina por diversos términos como teleformación, formación en línea, formación virtual, o simplemente con el término inglés *e-learning*. Desde el punto de vista de los sistemas tutores inteligentes la WWW ofrece muchas ventajas en comparación con otros medios mas tradicionales de difusión de programas: no requiere de instalaciones adicionales al usuario; se puede acceder al sistema desde cualquier lugar sin necesidad de transportar el software; los contenidos se distribuyen y mantienen actualizados de forma inmediata; facilita la construcción de sistemas distribuidos; existe la posibilidad de obtener datos sobre la efectividad de la instrucción y consiguientemente mejorarla, etc. Sin embargo, uno de los inconvenientes del uso de la WWW es la complejidad técnica que requiere tanto el desarrollo de sistemas personalizados como la creación de contenidos.

SIETTE es una aplicación eficiente de la teoría de test adaptativos informatizados y de la teoría de respuesta al ítem para la realización de tests a través de WWW. SIETTE aporta técnicas propias para sacar el máximo provecho de las características de este nuevo medio, ofreciendo un valor añadido a la evaluación mediante test. SIETTE incluye además una herramienta de autor para que los profesores puedan crear los tests también a través de Internet, y de un simulador para estudiar el funcionamiento de los tests en condiciones controladas, estudiar su efectividad, y calibrar los parámetros de un conjunto de preguntas. SIETTE ha sido

diseñado como componente autónomo, aunque puede también integrarse como parte de un sistema tutor inteligente.

IMS Global Learning Consortium es un grupo independiente, sin ánimo de lucro que inició su labor en 1997 impulsado por el NLII (National Learning Infrastructure Initiative) que es una organización apoyada por Educase. Aunque inicialmente surgió como una iniciativa en EEUU, ahora en IMS participan instituciones educativas de todo el mundo (desde universidades a pequeñas empresas de formación), fabricantes, y vendedores de aplicaciones software para la educación.

Actualmente es el principal promotor y desarrollador de especificaciones abiertas orientadas a la enseñanza electrónica. Su objetivo es que, a partir de estas especificaciones, se consiga la interoperabilidad de aplicaciones y servicios en la enseñanza electrónica para que los autores de contenidos y de entornos puedan trabajar conjuntamente.

IMS tiene muchas especificaciones (actualmente tiene 16 especificaciones) ya que cada una de ellas está enfocada en una necesidad distinta del proceso de enseñanza. Hay especificaciones que se refieren a metadatos de los objetos educativos, al formato de empaquetamiento y distribución de los cursos, a la información del usuario, a la secuenciación de contenidos educativos, o incluso al diseño de la actividad educativa en su conjunto.

IMS QTI (Question & Test Interoperability Specification) es la especificación sobre la que trabajaremos. Esta especificación contempla una estructura básica que describe la forma de representar preguntas individuales o ítems (assesment item) y gestionar evaluaciones o exámenes completos (assessment). Su objetivo es conseguir que tanto las evaluaciones cómo los resultados sean intercambiables entre los diferentes LMS. Así, podemos disponer de almacenes de preguntas y bases de datos con los resultados obtenidos por los alumnos a los que cualquier sistema de enseñanza electrónica podrá acceder.

Con este propósito se plantea y se documenta un formato de contenido para almacenar las preguntas o ítems independientemente del sistema o herramienta de autoría utilizada para crearlas. Esto permite, por ejemplo, el uso de las mismas

preguntas en diversos LMS (*Learning Management System*) o en sistemas de evaluación electrónica, o la integración en un único LMS de preguntas o exámenes desarrollados con distintas herramientas. Por otro lado se propone un sistema coherente para que los sistemas puedan informar de cuál es el resultado de una evaluación. El QTI esta especificado con el metalenguaje XML.

1.1 Objetivos

El objetivo principal de este proyecto consiste en realizar un reproductor del QTI (que después se integrara al sistema SIETTE, de esta forma le estamos sumando a SIETTE otra funcionalidad que consiste en dar suporte a los estándares que están teniendo mucha popularidad en los sistemas de E-learning, en este caso el estándar QTI).

El reproductor QTI debía ser un Servlet de Java. Dado un fichero XML que contenga la estructura de un ítem según la especificación QTI, el reproductor debería ser capaz de presentar la pregunta en el navegador Web, ser capaz de evaluar la respuesta del usuario y en caso de error (el fichero QTI no esta bien formado o no es valido) el reproductor debe de avisar de tal caso.

1.2 Estructura de la memoria

Esta memoria esta estructurada en nueve capítulos:

En el primero de ellos, el capitulo presente, hemos realizado una breve introducción presentando las claves fundamentales que han hecho posible la realización de este proyecto.

En el segundo capitulo, hablaremos de las tecnologías utilizadas, haciendo especial hincapié en la tecnología XML.

A continuación, en el capitulo tercero, hablaremos del sistema SIETTE.

En el cuarto capitulo, presentaremos los diferentes tipos de ítems del sistema SIETTE.

A continuación, en el capitulo quinto, trataremos el sistema QTI y describiremos los componentes que componen un ítem del QTI.

En el sexto capitulo será donde trataremos todo el desarrollo software llevado a cabo para la realización del reproductor QTI.

En el capitulo siete, evaluaremos nuestra aplicación sobre una variedad de ejemplos de preguntas QTI.

En el capitulo ocho, enunciaremos las conclusiones que se pueden extraer del trabajo realizado y también de las posibles líneas de trabajo futuro.

Por último se enumera la bibliografía que se ha utilizado para la elaboración de este proyecto y las referencias usadas en la memoria.

2. TECNOLOGIAS UTILIZADAS

A lo largo de este capitulo vamos a presentar las diferentes tecnologías que se han utilizado para el desarrollo de este proyecto.

Debido a que se trata de desarrollar una aplicación Web, podemos dividir las tecnologías utilizadas en dos grandes grupos: las tecnologías del servidor y las del cliente. Estas dos categorías serán de las que nos ocupamos en los dos primeros apartados. Por no poderla clasificar como una tecnología de cliente o servidor y por su importancia en este proyecto, a continuación describiremos con mas detalle el lenguaje XML.

2.1 Tecnologías servidor

2.1.1 Java

Sun Microsystems, líder en servidores para Internet, uno de cuyos lemas desde hace tiempo es "the network is the computer" (el verdadero ordenador es la red en su conjunto y no cada maquina individual). Es quien ha desarrollado el lenguaje Java en un intento de resolver simultáneamente todos los problemas que se le plantean a los desarrolladores de software por la proliferación de arquitecturas incompatibles, tanto entre las diferentes maquinas como entre los diversos sistemas operativos y sistemas de ventanas que funcionaban sobre una misma maquina, añadiendo la dificultad de crear aplicaciones distribuidas en una red como Internet.

Podemos hallar varias versiones distintas que intentan explicar el verdadero origen de Java, desde la que dice que fue un proyecto que reboto durante mucho tiempo por distintos departamentos de Sun sin que nadie le prestara mucha atención hasta que encontró su nicho en Internet; hasta la mas difundida, que justifica a Java como lenguaje de pequeños electrodomésticos. Por ello, lo único que podemos verificar es que en agosto de 1995 Sun Microsystems lo presento ante la sociedad como un nuevo lenguaje de programación.

La primera característica que llama la atención de Java es que es independiente de la plataforma, es decir, que podemos ejecutarlo sobre distintas arquitecturas y sistemas operativos sin que sea necesario realizar ningún tipo de cambio en el código del programa. Los programas en Java logran esta independencia mediante una maquina virtual que toma los programas de Java compilados y traduce sus instrucciones en código nativo de la maquina sobre la que se esta ejecutando. Los programas Java están compilados en un formato especial llamado código bytes (bytecodes), que es el formato que se le pasa a la maquina virtual. Para generar dicho código de bytes debemos compilar el código fuente con un compilador Java. Tanto el compilador java como la maquina virtual son específicos para cada plataforma, lo que implica que para poder ejecutar un programa java en una determinada plataforma, debe de existir previamente una maquina virtual para dicha plataforma. Por ello, Sun Microsystems dispone de un entorno de ejecución (JRE: Java Runtime Environment) para la mayoría de las plataformas actuales (Solaris 2.x, SunOs 4.1x, Windows95, Windows NT, Linux, Iris, Aix, Mac, Apple).

Otra ventaja a destacar de Java es que cuenta con una completísima librería de clases que no deja de aumentar. En ella podemos encontrar clases con las que hacer desde cosas tan simples como leer la hora del sistema, hasta clases más elaboradas con las que construir gráficos 3D.

2.1.2 J2EE

Java Platform Enterprise Edition es una plataforma de programación para desarrollar y ejecutar software escrito en Java basándose en una arquitectura distribuida en niveles basada en componentes modulares que se ejecutaran sobre un servidor de aplicaciones.

J2EE define gran número de especificaciones, entre ellas las de los servlets y las Java Server Pages (JSP). Los servlets son objetos Java que se ejecutan en un servidor y que permiten proporcionar contenido dinámico dentro de aplicaciones Web. Las JSP son otra forma de generar contenido dinámico en las aplicaciones Web incluyendo código Java dentro de código estático, como puede ser HTML.

La mayor parte de las aplicaciones Web de SIETTE están implementadas usando estas tecnologías de JSP y servlets.

2.1.3 Apache Tomcat

Apache Tomcat es un contenedor de aplicaciones desarrollado por Apache Software Fundation. Implementa las especificaciones de JSP y servlets por lo que se usa como contenedor Web.

SIETTE se despliega sobre un servidor Tomcat en su versión 4.1.31.

2.2 Tecnologías cliente

2.2.1 HTML

El lenguaje HTML o HyperText Markup Language es un lenguaje de marcas mediante el cual se le dice a un explorador de Internet como mostrar una página Web. El marco de trabajo de su invención y creación han sido atribuidos a Tim Berners-Lee, un programador del Centro Europeo para Física de Partículas (CERN). Berners-Lee desarrollo el HTML para:

- **ofrecer un medio que permitiera a los científicos publicar, buscar y recibir información 24 horas al día.**
- **Crear un lenguaje internacional de codificación en ordenadores que facilitara el acceso universal independientemente de la plataforma, red o terminal.**

Básicamente, el HTML consta de una serie de órdenes o directivas, que indican al visor que estemos utilizando, la forma de representar los elementos (texto, gráficos, etc.) que contenga el documento.

2.2.2 El lenguaje JAVASCRIPT

Javascript es un lenguaje interpretado que permite incluir macros en páginas Web. Estas macros se ejecutan en el ordenador del visitante de nuestras paginas, y no en el

servidor (algo muy interesante, porque los servidores Web suelen estar sobrecargados, mientras que los PC's de los usuarios no suelen estarlo).

Javascript proporciona los medios para:

- **Controlar las ventanas del navegador y el contenido que muestran.**
- **Programar páginas dinámicas simples.**
- **Evitar depender del servidor Web para cálculos sencillos.**
- **Capturar los eventos generados por el usuario y responder a ellos sin salir Internet.**
- **Simular el comportamiento de las macros CGI cuando no es posible usarlas.**
- **Comprobar los datos que el usuario introduce en un formulario antes de enviarlos.**
- **Comunicarse con el usuario mediante diversos métodos.**

La característica de Javascript que más simplifica la programación es que, aunque el lenguaje soporta cuatro tipos de datos, no es necesario declarar el tipo de las variables, argumentos de funciones ni valores de retorno de las funciones. El tipo de las variables cambia implícitamente cuando es necesario, lo que dificulta el desarrollo de programas complejos, pero ayuda a programar con rapidez macros sencillos. En esto, Javascript se separa de lenguajes como C, C++ o Java.

Javascript ha sido inventado por Nestscape, que comenzó a ofrecerlo como parte de su navegador v.2.0. El nombre original de Javascript fue LiveScript, pero se modifico en el último momento, aparentemente para aprovechar el tirón de Java. Al ser código interpretado, Javascript es más lento que Java, pero en la práctica no suele ser un factor de importancia.

Obviamente el objetivo de Netscape al introducir Javascript es tratar de establecer un estándar de programación de macros ejecutables en el navegador Web, que de ser adoptado por los Webmasters, facilitaría la implantación de los navegadores de Netscape en el mercado. En respuesta a este reto, Microsoft soporta una versión parcial de Javascript, con el nombre de Jscript, en su Internet Explorer.

2.3 Otras tecnologías

2.3.1 Lenguaje XML

2.3.1.1 Fundamentos de XML

XML (eXtensible Markup Language) no es, como su nombre podría sugerir, un lenguaje de marcas. XML es una especificación que nos permite definir lenguajes de marcado adecuados a usos determinados, es decir, un meta-lenguaje.

El meta-lenguaje XML es un subconjunto de SGML (Standard Generalised Markup Language), pero simplificado y adaptado a Internet. XML fue adoptado como estándar por el w3c (World Wide Web Consortium) en Febrero de 1998.

Aunque a primera vista, un documento XML puede parecerse a HTML, existen claras diferencias:

- **Como diferencia principal, HTML tiene marcas predefinidas mientras que XML permite definir sus propias marcas cuyos nombres pueden ser representativos de la situación.**
- **HTML define más la presentación que el contenido, mientras que XML separa radicalmente el contenido o información de la presentación.**
- **El uso de HTML se limita a las páginas Web mientras que el XML se centra en el almacenamiento de información mediante bases de datos.**
- **HTML no sigue una estructura fija y tiene datos mal definidos mezclados con elementos de formato. Su procesamiento se hace complejo, al contrario que XML, cuyo procesamiento es sencillo gracias a la estructura jerárquica característica de los documentos XML.**

Todos los documentos XML deben estar bien formados, y este es el requisito mínimo que deben cumplir los documentos XML. Cuando hablamos de ficheros bien formados nos referimos a ficheros con determinadas características:

- **Un solo elemento raíz: Los documentos XML solo permiten un elemento raíz del que todos los demás sean parte.**

- **Estructura jerárquica: Los documentos seguirán una estructura estrictamente jerárquica. Una etiqueta debe estar debidamente incluida en otra y además debe estar correctamente cerrada.**
- **Contenido de los atributos: Los atributos de los elementos deben estar siempre entre comillas dobles (") o comillas simples ('), se suele usar las comillas simples cuando el contenido del atributo contiene comillas dobles, si tiene ambos entonces se usa ' y ".**
- **Mayúsculas y minúsculas: XML es sensible a las mayúsculas y minúsculas, si un elemento esta definido como "algo" no podemos usar "Algo".**
- **Etiquetas vacías: XML permite etiquetas sin contenido, como por ejemplo:<etiqueta></etiqueta> pero esto es muy incómodo, por lo que se usa la notación <etiqueta/> simplemente.**
- **Nombre de las etiquetas: Los nombres de las etiquetas comienzan con una letra o uno o más signos de puntuación, y continúa con letras, dígitos, rayas, dos puntos, denominados de forma global como caracteres de nombre.**

En un documento XML, aparte de elementos y atributos, puede haber otras cosas: comentarios, que se procesan de forma diferente al texto, y que van precedidos por <!-- y acaban con -->; entidades, que representan símbolos "atómicos", que habitualmente deben ser entendidos por el navegador, las entidades van encerradas entre los caracteres '&' y ';', podemos ver las entidades predefinidas en la tabla siguiente; secciones CDATA, que sirven para extraer del documento XML una sección, que va a ser interpretada tal cual, sin hacer ninguna modificación. Puede servir, por ejemplo para introducir HTML dentro de un documento XML.

Entidad	Carácter
&quot;	"
&apos;	'
&lt;	<
&gt;	>
&amp;	&

Tabla1: Entidades predefinidas

2.3.1.2 Definición de documentos XML

Como ya hemos comentado anteriormente, XML nos permite definir nuestras propias etiquetas. Para ello existen varias formas de establecer los nombres de las etiquetas, atributos, estructura, etc., que queremos usar en nuestro documento.

Cuando el documento se ajusta a estas formas, se denomina "válido". El concepto de validez es diferente al de estar "bien-formado". Un documento bien-formado simplemente respeta la estructura y sintaxis definidas por la especificación de XML. Un documento "bien-formado" puede además ser "válido". También existen documentos que no se ajustan a ninguna de las formas anteriores, en ese caso no son "válidos", pero tampoco son "inválidos", simplemente "bien-formados" o no.

Las formas de las que hemos hablado son principalmente dos, las DTDs y los Esquemas.

2.3.1.3 DTD

La primera opción que podemos usar es una DTD (Document Type Definition). Una DTD es una especificación y estructuración mediante una gramática de contexto libre, que permite validar el contenido estructural y formal de un documento XML. Las DTDs se pueden usar para la definición de modelos de contenido, es decir, en

qué orden y que elementos pertenecen a un elemento de orden superior en la jerárquica del documento; además permiten, aunque de modo muy limitado, imponer ciertas restricciones sobre el tipo de los elementos.

Una DTD puede residir en un fichero externo, y quizá compartido por varios documentos. O bien, puede estar contenido por el propio documento XML como parte de la declaración del documento. Para especificar la DTD que vamos a usar se hace de la siguiente manera:

- **Si la DTD se encuentra dentro del documento, entonces se incluye lo siguiente: <!DOCTYPE elemento _ raíz SYSTEM [contenido_DTD]> donde elemento _ raíz es el elemento principal del documento, y Contenido_DTD es la sintaxis de la DTD.**
- **Si la DTD está en un fichero externo, entonces, lo que hay que incluir en el documento es parecido a lo anterior sólo que cambiando el contenido por la URI (Universal Resource Identifier) de la DTD. Es algo como: <!DOCTYPE elemento _ raíz SYSTEM URI>**

Para entender el contenido de un fichero DTD necesitamos conocer los siguientes elementos:

- <!-- Se usa para introducir comentarios en la DTD.

- <!ELEMENT se usa para definir elementos, va seguido del nombre del elemento y del tipo de éste. El tipo puede ser (#PCDATA) para indicar que es texto, EMPTY para indicar un nodo vació, ANY para indicar cualquier contenido (No se suele usar), o puede contener otros elementos; en este caso se indica entre paréntesis y separado por comas el nombre y su cardinalidad. Para su cardinalidad se usan los caracteres:

- **Ningún carácter ► elemento obligatorio (1 vez)**
- **‘+’ → elemento necesario y repetible (1 o mas veces)**
- **‘*’ → elemento opcional y repetible (0 o más veces)**
- **‘?’ → elemento opcional (0 ó 1 vez)**

- **‘|’ → indica disyunción entre elementos.**

Un ejemplo de aplicación de este apartado puede ser:

<!ELEMENT cartel (encabezado?, (imagen | texto+)*)>

- <ATTLIST se usa para crear listas de atributos, va seguido del nombre del elemento al que pertenecen y de una lista de atributos, cada elemento de la lista está compuesta por el nombre del atributo, el tipo del atributo, y un modificador. El tipo puede ser:

- **CDATA cadena de texto**
- **NMTOKEN cadena de caracteres legales para nombres en XML**
- **NMTOKENS lista de tokens separados por espacio**
- **ID nombre único entre el resto de atributos del tipo ID**
- **IDREF referencia a un atributo de tipo ID**
- **IDREFS lista de IDREFs separados por espacio**
- **ENTITY para indicar que el atributo es el nombre de una entidad**
- **ENTITIES para indicar lista de entidades**
- **NOTATION nombre de una notación declarada en el DTD**
- **Puede tener una lista de valores posibles entre paréntesis y separados por ‘|’ para indicar un enumerado**

Con respecto al modificador puede ser:

- **#REQUIRED indica que el atributo es obligatorio**
- **#IMPLIED el atributo es opcional y no toma valor por defecto**
- **#FIXED "valor" el atributo siempre tiene un valor fijo**
- **"valor" el atributo es opcional y si no está toma por defecto el valor dado**

2.3.1.4 Esquema

La otra opción son los esquemas , creados para solventar las limitaciones de las DTDs. Estas limitaciones y las diferencias se detallan en el próximo apartado.

Los esquemas utilizan un mecanismo de reutilización de etiquetas o tags: los namespaces , que permiten la creación de librerías para su uso en documentos. Los namespaces indican el lugar de donde se toman elementos que se usan en el documento. Para indicar el namespace al cual pertenece un elemento, se pone a este elemento un prefijo que consta de una referencia y el carácter ':'.

Un esquema sigue una estructura jerárquica. Al igual que un fichero XML, tiene un elemento raíz dentro del cual se extiende un árbol de elementos con sus atributos.

Hay ciertos elementos que se usan para definir los esquemas de QTI que son necesario conocer para comprender cual es su función dentro del esquema:

-Un esquema suele comenzar con el elemento schema: <xs:schema xmlns:xs=http://www.w3.org/2001/XMLSchema> donde se especifica el namespace asociado a xs para su posterior uso a la hora de definir elementos y atributos.

- <element se usa para definir elementos, va seguido del nombre y tipo del nuevo elemento: <xs:element name="nombre" type="xs:string">. Se suele poner como prefijo el namespace al que pertenece el elemento element. Puede tener como atributos *minOccurs* y *maxOccurs* cuyo contenido debe ser un número, y se usa para indicar la cardinalidad del elemento. Si estos atributos no se ponen, se toman por defecto 1 para ambos, podemos indicar que el elemento se puede repetir infinitas veces con el valor *unbounded*. El tipo puede ser uno definido por el usuario, o uno de los tipos básicos, en este último caso se suele poner al tipo un prefijo que corresponde con la abreviatura del namespace donde están definidos estos tipos. En la especificación de los esquemas se definen más de 30 tipos básicos distintos, de ellos los principales son los siguientes: *string, integer, long, short, decimal, float, double, boolean, time, date.*

- <attribute se usa para definir atributos al crear nuevos tipos con <complexType. Va seguido del nombre y el tipo: <xs:attribute name="edad" type="xs:integer" use="required"/>. También tiene el atributo opcional use, que indica si el atributo que se está definiendo es necesario o no. Puede tomar los valores *opcional* y *required*, por defecto su valor es opcional.

- <complexType permite definir nuevos tipos que no están definidos. Dentro de este elemento se definen los atributos y elementos que puede tener el nuevo tipo. Estos elementos van incluidos dentro de las siguientes etiquetas:

- **<sequence>: Esta etiqueta se usa para establecer el orden de los elementos que se definen dentro. En el mismo orden en el que aparezcan aquí deben aparecer en el documento.**
- **<choice> Esta etiqueta permite definir una disyunción de los elementos que se definen dentro. Puede contener los atributos *minOccurs* y *maxOccurs*, para indicar el número de elementos. Por defecto el número de elementos es 1.**

Si no se definen elementos y el contenido es un tipo básico se usa el elemento <simpleContent>, dentro del cual se pone <extension base="tipo"> con lo que se especifica el tipo y se definen los atributos. En otro caso los atributos se ponen directamente en el elemento *complexType*. Para definir un enumerado, dentro de los atributos se usa el elemento <simpleType>, y dentro de éste el elemento <restriction base="NMTOKEN">. Cada elemento del enumerado se define con <enumeration value="valor_del_enumerado"/>.

2.3.1.5 Comparación entre esquema y DTD

Dado que el surgir de una nueva tecnología donde ya existía otra es síntoma de que un grupo de gente cree que en el actual estado de ese campo, o bien las cosas no están bien hechas, o bien falta algo por hacer. Vamos a mostrar que limitaciones tenían las DTDs para que fuese necesaria la aparición de XML Schema.

- **Posee un lenguaje propio de escritura, lo cual deriva en problemas a la hora de:**
  - **El aprendizaje: no sólo hay que aprender XML si no que además hay que conseguir hacerse con el lenguaje de las DTDs.**
  - **Procesado del documento: las herramientas y parsers que se empleen para tratar los documentos de XML deben ser capaces de procesar las DTDs.**

- **No permite el uso de namespaces, estos permiten definir elementos con igual nombre dentro del mismo contexto, siempre y cuando se anteponga un prefijo al nombre del elemento.**
- **Tiene un tipado para los datos del documento extremadamente limitado, no permite definir el que un elemento pueda ser de un tipo número o de un tipo fecha, sólo presenta variaciones limitadas sobe strings.**
- **El mecanismo de extensión es complejo y frágil, está basado en sustituciones sobre "strings". Lo peor de dichas extensiones es que realmente no hace explícitas las relaciones en ningún momento, es decir dos elementos que tienen definido el mismo modelo de contenido no presentan ninguna relación.**

A parte de solventar los problemas antes expuestos, XML Schema, permite una serie de ventajas adicionales que se consideraron importantes:

- **Una estructura de tipos mucho más rica. En la segunda parte de la especificación de XML Schema (XML Schema Part 2: Datatypes) se definen los tipos base que se pueden emplear dentro de esquema de XML: integer, boolean, string, date, etc.**
- **Es posible agrupar atributos, haciendo más comprensible el uso de un grupo de aspectos de varios elementos distintos, pero con denominador común, que deben ir juntos en cada uno de estos elementos.**
- **Permite tipos definidos por el usuario, llamados Arquetipos, dándoles un nombre y que se pueden emplear en distintas partes dentro del Schema.**
- **El trabajo con namespaces (XML Schema Part 0: Primer) está especificado, permitiendo, dentro de la dificultad que conlleva trabajar con ellos, validar documentos con varios namespaces.**
- **Sin embargo, la característica que más resalta la utilidad de XML Schema es la posibilidad de extender Arquetipos de un modo especifico, es decir permite lo que en términos de orientación a objetos se llama herencia. Considérese un esquema que extiende otro previamente hecho, se permite refinar la especificación de algún tipo de elemento para, por ejemplo, indicar que puede contener algún nuevo elemento del tipo que sea; pero dejando el resto del esquema antiguo completamente intacto.**

2.3.2 XML y Java

El pequeño conjunto de normas que define XML y la posibilidad de definir reglas, como son los esquemas, para especificar los tags válidos, permite construir parsers genéricos que son capaces de comprobar que el documento XML está "bien-formado" y es "válido".

Los analizadores o parsers de XML pueden implementarse en cualquier lenguaje (Java, C++, Perl, …). No obstante, hay una tendencia natural a utilizar Java para XML. De hecho, una característica compartida de Java y XML es la independencia de la plataforma. Al utilizar Java para implementar la tecnología XML se obtiene como valor añadido capacidad multiplataforma a nivel binario, de forma que incluso las herramientas que se utilizan para analizar y depurar código XML son independientes de la plataforma.

En este apartado se trata de explicar lo relacionado con el procesado de documentos XML. Se describen brevemente las etapas típicas a la hora de trabajar con estos documentos y las diferentes herramientas disponibles para ello.

2.3.2.1 Fases en el procesado de XML

El procesado de documentos XML consiste en la obtención de información del fichero fuente y su tratamiento. Son tres las fases típicas a la hora de realizar este procesado:

1- Procesamiento de entrada:

- **Analizar y validar el documento fuente.**
- **Reconocer/buscar información importante basándose en su localización o en su etiquetado en el documento fuente.**
- **Extraer la información importante una vez que se ha localizado.**
- **Opcionalmente, mapear/unir la información recuperada.**

2- Manejo de la información:

- **El procesamiento real de la información de entrada resultando opcionalmente en la generación de información de salida.**

3- Procesamiento de salida:

- **Construir un modelo de documento para generar con DOM, JDOM, etc.**
- **Aplicar hojas de estilo SXLT o serializar directamente a XML.**

2.3.2.2 Principales APIs para XML

Una API (Aplication Program Interface) consta de una serie de clases, con sus correspondientes métodos. Estos APIs nos permiten trabajar con un documento XML desde un programa escrito en Java pudiendo acceder a los datos, comprobar si ésta bien formado, si es válido, etc.

Los principales son lo siguientes:

- **SAX, Simple API for XML: la principal característica de SAX es que el documento se lee secuencialmente de principio a fin, sin cargar todo el documento en memoria. Es un modelo basado en eventos, al leerse determinados elementos se harán determinadas cosas. SAX es una especificación y como tal no podemos trabajar con ella, pero si con sus implementaciones: los parsers Xerces, Crimson, etc.**

  Tiene sus características positivas y negativas:

  - **La ventaja es la eficiencia en cuanto al tiempo y la memoria empleados en el análisis.**
  - **La desventaja es que con SAX no disponemos de la estructura en árbol de los documentos, luego no podemos modificar ni crear documentos XML, ni recorrerlos jerárquicamente, solamente analizarlos secuencialmente.**
- **DOM, Document Object Model API from W3C: la principal característica de DOM es que el documento con el que se trabaja se carga entero en la memoria, en una estructura de árbol, por lo que la forma de trabajar con DOM es muy distinta a la de SAX. Al igual que SAX, DOM es una**

especificación y por tanto no se puede trabajar directamente con ella. Se establece dos niveles de actuación:

- **Nivel 1: se refiere a la parte interna, y modelos para HTML y XML. Contiene funcionalidades para la navegación y la manipulación de documentos. Tiene dos partes: el core o parte básica, referida a documentos XML, y la parte HTML, referida precisamente a los HTML.**
- **Nivel 2: incluye un modelo de objetos interface de acceso a las características de estilo del documento, definiendo funcionalidades para manipular la información sobre el estilo del documento. También incluirá un modelo de eventos para suportar los namespaces de XML, y consultas enriquecidas.**
- **Posteriores niveles: especificaran interfaces a posibles sistemas de ventanas, manipulación de DTD y modelos de seguridad.**

Las principales ventajas e inconvenientes son:

- **La ventaja es que con DOM, al disponer de la estructura del documento, podemos acceder a los datos en función de la jerarquía de elementos, así como modificar el contenido de los documentos y incluso crearlos desde cero.**
- **La desventaja es el coste en tiempo y memoria que conlleva construir el árbol para un documento, sobre todo si tiene cierto tamaño.**

- **XPath, XML Path Lenguaje from W3C: XPath esta construido sobre DOM, es por ello que el documento XML se carga en una estructura de árbol. Proporciona al usuario una forma particular de acceder a los elementos, atributos, y cualquier otra información contenida dentro de un fichero XML. Se usa un "path" o ruta para acceder a cada elemento. Esta ruta está formada por elementos del árbol separados por el carácter '/' que en conjunto indica la ruta.**

  Las ventajas y desventajas son las mismas que las de DOM, pero XPath trabaja de forma más directa que DOM. XPath facilita el manejo al usuario

proporcionando métodos que evitan tener que recorrer cada nodo y sus hijos para acceder a un nodo determinado dentro del árbol.

- **JDOM, "Java optimizad" document object model API from jdom.org: JDOM es un API pensada específicamente para el procesado de documentos XML con Java. Nos proporciona una capa de abstracción en el tratamiento de ficheros XML, ya que JDOM se sitúa entre un parser y el usuario. Además de las ventajas e inconvenientes que arrastra del parser sobre el cual trabaja (ya sea una implementación de SAX o de DOM) tenemos las siguientes:**
  - **La principal ventaja es que es un API pensado especialmente para Java y por tanto mejor integrado en este lenguaje, además es fácil de usar.**
  - **Desventajas: Al ser un API diseñado para Java resulta poco portable a otros lenguajes de programación. El carecer del concepto de Nodo dificulta en cierta medida la navegación a través del árbol del documento. Por ejemplo, al obtener los hijos de un elemento, el resultado es una lista de objetos genéricos, y debemos comprobar qué representan esos objetos para saber si son elementos, atributos, etc.**

2.3.2.3 Principales parsers para XML

Sobre los APIs descritos anteriormente se definen los procesadores de XML (parsers) para analizar, manipular y validar los documentos. Los parsers son las implementaciones de las especificaciones SAX, DOM, etc. Como ya hemos citado, java y XML van cogidos de la mano gracias a la portabilidad de ambos sistemas, es por ello que java ha sido el lenguaje de programación usado en este proyecto.

A la hora de elegir el parser para el tratamiento de documentos XML nos encontramos con muchas posibilidades, las más conocidas son las siguientes:

- **Xerces, del proyecto Apache. La versión mas actual, la 2.4.0 tiene como características principales las siguientes:**
  - **Disponible para java, C++ y Perl.**
  - **Es de código abierto**
  - **Soporta XML 1.0 Second Edition Recommendation**
  - **Soporta namespaces**

  - **Soporta SAX 2.0 y DOM nivel 2**
  - **Soporta JAXP 1.2**
  - **Soporta W3C XML Schema Recommendation 1.0**
  - **Permite la validación contra DTDs y contra esquemas**
- **Crimson, también de Apache. La versión actual es la 1.1 y sus características son:**
  - **Disponible para Java**
  - **Es de código abierto**
  - **Soporta XML 1.0**
  - **Soporta SAX 2.0 y DOM nivel 2**
  - **Soporta JAXP 1.1**

- **Oracle XML Parser, de Oracle [15]. La versión actual para java es la 2.0.5. Sus principales características son:**
  - **Disponible para Java, C, C++ y PL/SQL**
  - **Se puede bajar gratuitamente de Internet**
  - **Soporta namespaces**
  - **Soporta validación contra DTDs**
  - **Soporta SAX 2.0 y DOM 2.0**
  - **Soporta XSLT (XSL Transformations)**
- **XML4J, de IBM [8]. La versión actual es la 4.1.4 y sus características más importantes son:**
  - **Disponible para Java**
  - **Parser bajo licencia de IBM Alphaworks**
  - **Soporta W3C XML Schema Recommendation 1.0**
  - **Soporta SAX hasta 2.0 y DOM hasta nivel 2 y algunos rasgos experimentales de nivel 3**
  - **Soporta JAXP 1.2**
  - **Permite validación contra esquemas y DTDs.**

En nuestro proyecto se ha optado usar el parser xerces y la validación contra DTD. Se ha usado concretamente la API DOM por facilidad de programación que ofrece y por la necesidad de tener en memoria la estructura del documento XML para poder interpretarlo y reproducirlo al usuario en el navegador Web.

3. EL SISTEMA SIETTE

3.1 Descripción general

SIETTE es un sistema de evaluación mediante tests adaptativos de contenido equilibrado basado en la teoría de respuesta al ítem, que ha sido diseñada para ser usada sobre la World Wide Web.

Utilizando un navegador estándar como interfaz gráfica de usuario, ofrece a los alumnos un aula virtual en el que realizar tests y permite a los profesores crear nuevos tests o añadir nuevas cuestiones a los tests ya creados. Además, dichos tests no sólo destacaran por ser tests adaptativos sobre WWW, sino que aceptaran como cuestiones ciertas plantillas que el usuario podrá definir y que facilitaran la creación final de test. A su vez, estas cuestiones y plantillas podrán incluir objetos multimedia, tales como: video-clips, ficheros de audio, animación y gráficos; mejorando así el aprendizaje del alumno y el método de evaluar su conocimiento.

Para comprender mejor el sistema SIETTE, se expone a continuación sus principales características:

- **Reduce el coste de evaluación de los alumnos.**
- **Suministra a los usuarios dos tipos de herramientas. Por un lado, un conjunto de herramientas de edición de tests con las que se construye la base de conocimiento (especificaciones de tests, cuestiones, plantillas de cuestiones y objetos multimedia) del sistema. Por otro lado, el sistema de generación de tests, que construye el test más apropiado dependiendo del examinado. Además, se ha construido una nueva herramienta grafica que permite a los profesores analizar los resultados de los alumnos que hayan realizado tests de sus asignaturas. Será posible acceder a estadísticas sobre el número de preguntas realizadas por los alumnos, numero de preguntas que han sido respondidas, etc.**
- **Confidencialidad de la información existente. El acceso tanto al editor de tests como al generador de tests, esta restringido mediante palabras claves.**

- **Arquitectura distribuida: el sistema permite que haya múltiples alumnos realizando el mismo test, y a su vez permite que haya profesores editando las preguntas del test o insertando nuevas preguntas. La coherencia de la base de conocimiento (banco de ítems) se mantiene gracias a un mecanismo automático de validación de datos, que determina el momento mas apropiado para llevar acabo las actualizaciones en la base de datos.**
- **Adaptable al gusto del usuario. El formato y apariencia de las cuestiones son muy flexibles. Basta con conocer la sintaxis del lenguaje HTML para que estas obtenga la apariencia que el creador del test considera oportuna.**
- **Uso de contenido multimedia. Las cuestiones del test pueden contener objetos multimedia (imágenes, videos, sonido, etc.) si el diseñador del test si lo desea. Además a través de la biblioteca de esquemas de preguntas, es posible proponer a los alumnos casi cualquier tipo de pregunta.**
- **Accesible a través de Internet. El acceso al sistema y sus resultados están disponibles a cualquier hora y desde cualquier lugar.**
- **Administración automática e inteligente del test. Se construye tests adaptativos, los cuales el número de preguntas necesarias para que el alumno obtenga su calificación es mucho menor en los clásicos tests de papel y lápiz, en los que todos los alumnos, independientemente de su nivel de conocimiento, tienen que responder un número fijo de cuestiones. En SIETTE, a un alumno solo se le plantearan las preguntas adecuadas para su nivel de conocimiento, evitando así la frustración o el aburrimiento de dicho alumno. De este modo, en el sistema se han tenido en cuenta factores como: el grado de adivinanza a cada pregunta del test, dificultad de las preguntas, así como el factor de distracción. La precisión del sistema se incrementa al aumentar el número de cuestiones que el creador del test introduce en la base de conocimiento de la que hace uso el sistema. Se favorece así, la no repetición de preguntas en la misma sesión de test, entre alumnos del mismo nivel de conocimiento, etc.**
- **Aprendizaje de la dificultad de las preguntas. A través de procedimiento de aprendizaje nos aseguramos de que las preguntas están correctamente calibradas, ya que el profesor cuando elabora el test solo da una estimación personal de la dificultad real de la pregunta, no siempre sucederá así, a**

través de los datos empíricos obtenidos a partir de la ejecución de los tests por parte de los alumnos, el sistema de aprendizaje será capaz de asignar cuales la verdadera dificultad de cada pregunta y a su vez de cada una de las respuestas a esta.

- **Calificación generada por el SIETTE. El objetivo final del sistema SIETTE es estimar el nivel de conocimiento del alumno proponiendo a este las preguntas mas adecuadas para su nivel de conocimiento estimado en cada momento, y con el menor numero de preguntas posibles. Por tanto, la calificación generada por el sistema esta formada por: una distribución de un número establecido a priori de probabilidades y una estimación del nivel de conocimiento del alumno según esa distribución.**
- **SIETTE crea un historial del alumno de tal modo que si un alumno abandona un test de autoevaluación antes de que este finalice y posteriormente vuelve a realizar dicho test, el sistema le planteara preguntas según el nivel de conocimiento inicial. En cambio, si el test es de evaluación, el alumno solo podrá realizar el test una sola vez, impidiéndose así que pueda modificar los resultados estimados en la sesión anterior.**
- **Refuerzo inmediato. Los resultados del test son calculados inmediatamente y están disponibles en formato grafico. Dichos resultados junto con otros datos sobre el alumno son almacenados para posteriores análisis sobre el perfil de aprendizaje de dicho alumno. El alumno puede ver las respuestas correctas a las preguntas que se les hace (ya que éstas no serán repetidas en la sesión actual de test) bien después de cada pregunta, o bien al finalizar el test.**

3.2 Arquitectura del sistema

La arquitectura del sistema SIETTE, contiene los principales componentes de un test adaptativo agrupados en seis módulos principales, su representación grafica se muestra en la figura 3.1.

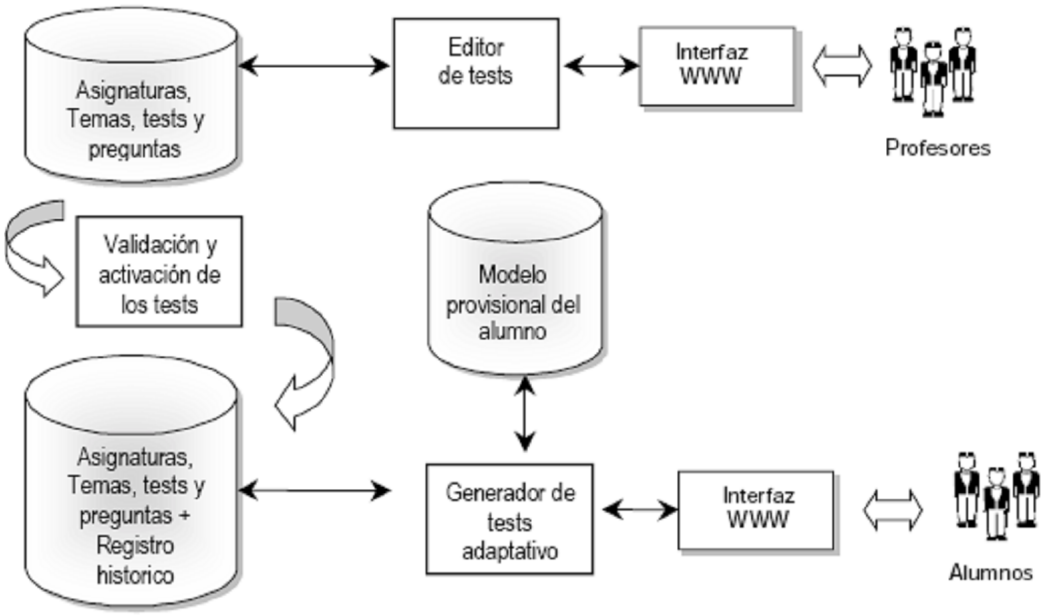


Figura 3.1 Arquitectura del sistema SIETTE

- **La base de conocimiento: Está compuesta por el currículum o estructura del temario, las especificaciones de tests y el banco de ítems, que constituye la colección de posibles cuestiones a presentar en un test, todas ellas calibradas con una serie de parámetros.**
- **El generador de tests: Es el modulo principal del sistema SIETTE. Es el encargado de seleccionar las preguntas a plantear al alumno, según las especificaciones del test y del modelo temporal del alumno.**
- **El modulo de edición: Permite a los profesores acceder a la base de conocimientos, donde podrán añadir nuevas preguntas, respuestas, conceptos, tests, etc. Actualmente hay dos editores: una aplicación Java que permite modificar la base de conocimiento de forma off-line y un editor Web que la modifica de forma on-line. Ambos ofrecen la misma funcionalidad. En el caso de la aplicación Java, una vez haya hecho los cambios oportunos podrán conectarse con la base de conocimiento para transferir los cambios realizados.**
- **Modelos temporales de los alumnos: Es creado y actualizado por SIETTE por cada estudiante que hace un test. Básicamente, el modelo de un estudiante consiste en una estimación de su nivel de conocimiento.**
- **El modulo de validación y activación de tests: Las modificaciones realizadas en la base de conocimiento a través del editor de tests, no son inmediatamente**

actualizadas. Si el proceso de actualización fuera realizado en tiempo real, el sistema SIETTE podría tener problemas de inconsistencia. Por ello, necesitamos de un mecanismo de validación y activación de tests. De esta forma, los profesores hacen sus modificaciones en una base de conocimiento distinta de la base de conocimiento del generador de tests. Una vez al día, un proceso interno bloquea los elementos modificados y los actualiza en la base de conocimiento del generador de tests.

- **El modulo de aprendizaje: Se encarga de realizar una calibración de los parámetros de los ítems a partir de la información obtenida de forma empírica tras las sucesivas ejecuciones de los tests por parte de los estudiantes. Esta calibración es necesaria ya que los valores asociados a los ítems que introducen los profesores, solo son aproximaciones iniciales.**

Además de los componentes que han sido descritos anteriormente, SIETTE dispone de un simulador que permite el estudio empírico de las variables estadísticas que se manejan, así como de la influencia de los parámetros de los ítems, e incluye mecanismos para el aprendizaje automático de estos parámetros.

3.2.1 La Base de Conocimientos

SIETTE es un sistema para evaluación. Esta estructurado por materias o asignaturas independientes. Cada una de estas asignaturas tiene una base de conocimientos distinta. La base de conocimiento de SIETTE está formada por tres tipos de objetos: Los *conceptos*, que son los temas o elementos en los que se descompone la asignatura. Están estructurados jerárquicamente, formando un *curriculum*. Los *tests*, que representan las sesiones de evaluación. Cada test está formado por un conjunto de ítems o cuestiones. Los ítems están asociados a uno o varios conceptos o temas. La definición de los tests se hace en función de los temas sobre los que se desea evaluar.

3.2.2 El Generador de Tests

Una vez que el profesor ha elaborado un test, los alumnos pueden acceder a él a través de una interface Web denominada *aula de tests*. Las preguntas se generan

dinámicamente y de forma individualizada para cada alumno según la materia y el test que haya elegido.

El primer paso que tiene que dar el alumno es identificarse dentro del sistema. SIETTE permite el acceso tanto de forma identificada como anónima. Para la primera opción, el usuario deberá introducir un identificador personal y una contraseña. En caso de que sea la primera vez que el usuario accede al sistema, deberá dar sus datos personales: nombre, apellidos, correo electrónico, así como seleccionar un identificador personal y una contraseña. Una vez terminado el proceso de identificación, se le muestran los tests disponibles para él. Cuando el usuario selecciona el test que quiere realizar, SIETTE le mostrara información sobre este test antes de empezar a formularle cuestiones. Si posteriormente quisiera volver a realizar otro o el mismo test que resolvió anteriormente, únicamente tendrá que introducir en el sistema su identificador y palabra de paso, puesto que ya consta como usuario registrado.

Para los usuarios registrados, el sistema tiene en cuenta los tests realizados anteriormente, y es capaz de mantener el modelo temporal como punto de partida para una nueva evaluación. Durante el proceso de realización del test, el sistema mantiene un registro de la evolución del alumno, que se tiene en cuenta en el proceso de selección del siguiente ítem; y un registro histórico de las respuestas a cada cuestión, que servirá como fuente de información para el aprendizaje automático de los parámetros de las preguntas.

La figura 3.2 y 3.3 muestran un estado intermedio durante la realización de un test. Ambas corresponden a páginas generadas dinámicamente por el generador de tests de SIETTE para un test de licencias y permisos de conducción. Una vez que el alumno responde a una pregunta, si se ha seleccionado esta opción, el generador indicara cual es la solución correcta. En este caso, como se puede apreciar, el alumno no ha respondido correctamente, el generador muestra su respuesta con una marca en rojo, si la respuesta estuviera correcta la marca será en verde.

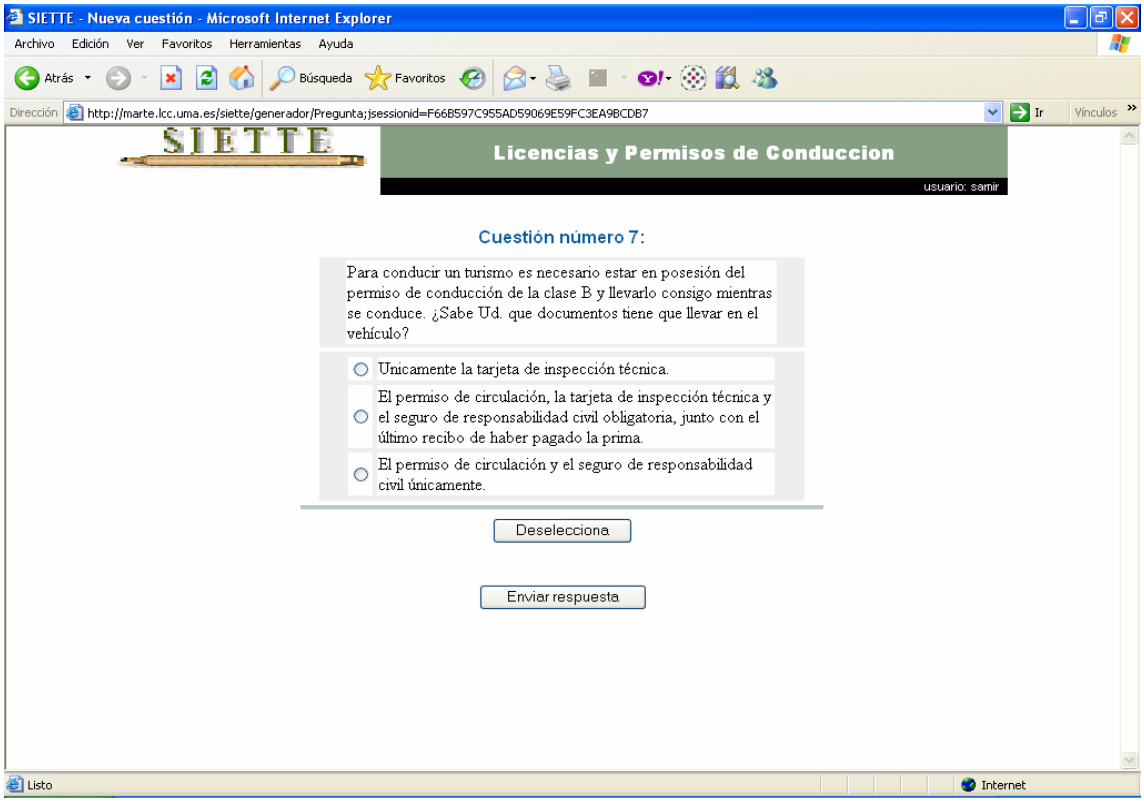


Figura 3.2

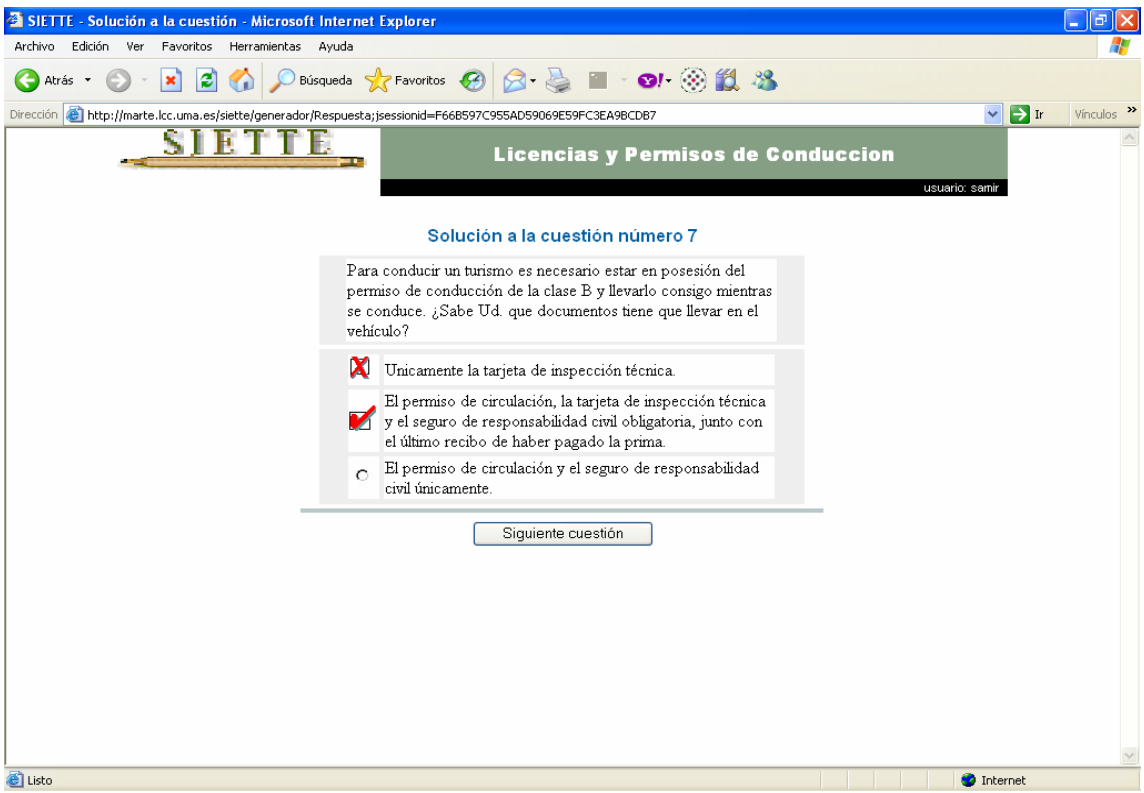


Figura 3.3

Una vez finalizado el test, el sistema devolverá (Figura 3.4) el nivel estimado de conocimiento del alumno, el numero de preguntas que ha realizado, así como el numero de preguntas que han sido respondidas correctamente. Alternativamente, si SIETTE se usa integrado en un sistema instructor inteligente, al terminar la evaluación esta información pasa al sistema tutor mediante la llamada a una URL distinta para cada nivel de conocimientos, o mediante una misma URL a la que se pasan estos datos mediante parámetros.

Además se mantiene un registro temporal de la evolución del alumno durante la sesión que se tiene en cuenta en el proceso de selección de preguntas y un registro histórico de las respuestas a cada pregunta que servirá como fuente de información para el aprendizaje automático de los parámetros de las preguntas.

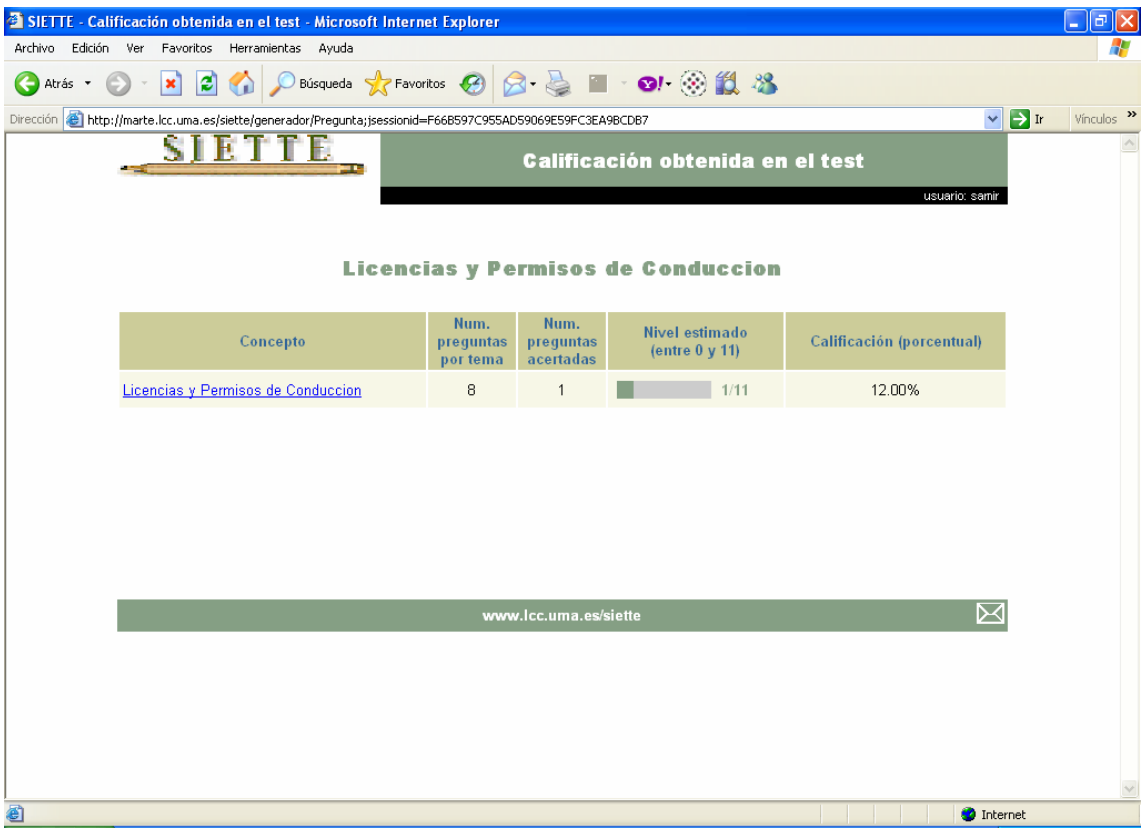


Figura 3.4: Evaluación final de un test.

3.2.3 Los Módulos de Edición

El editor de tests es una herramienta de autor que permite a los profesores insertar cuestiones, y definir los tests que pueden realizar los alumnos. La información generada por el editor se almacena en una base de conocimiento temporal hasta que

el modulo de validación y activación las transfiera a la base de conocimiento del generador de tests, estando a partir de este momento a disposición de los estudiantes.

Actualmente coexisten dos editores. Un editor a través de la Web, que fue desarrollado en la primera versión de SIETTE, y una nueva herramienta para edición y gestión de tests desarrollada en Java, que no requiere el uso del navegador.

Para acceder a la interfaz del editor Web, el profesor debe disponer de un identificador y una contraseña válida para la asignatura que desee modificar. Cada asignatura o materia se subdivide en temas en la forma que el profesor crea mas adecuada, usando para ello las secciones para añadir, modificar o eliminar temas. Los temas representan grandes bloques de la asignatura sobre los que se desea evaluar y no necesariamente conceptos concretos. Una vez definidos los temas, el profesor definirá los ítems o cuestiones que compondrán los tests. Para ello rellena una ficha en la que se incluye el enunciado, la respuesta correcta y una o varias alternativas de respuestas incorrectas. El sistema ha sido diseñado de forma que también almacene una posible ayuda, asociada al enunciado del ítem, en caso de que el alumno solicite mas información para resolver la pregunta. También es posible añadir un refuerzo por cada una de las respuestas, de forma que si el alumno no ha acertado al resolver la pregunta, el sistema puede indicarle por qué su respuesta no es valida. Esta característica convierte a SIETTE en un sistema no sólo evaluador sino también en cierto modo instructor.

Los textos de las preguntas y las respuestas son trozos de codigo HTML, a los que se puede añadir JavaScript, Applets de Java, e incluso matacódigo en PHP, PERL, JSP, etc. El formato de las cuestiones admite cualquier objeto multimedia. Además del enunciado, las respuestas, la ayuda y los refuerzos, el profesor debe proporcionar algunos datos sobre la cuestión, como por ejemplo, el número de alternativas incorrectas a mostrar, el factor de dificultad y debe asociar cada pregunta a uno o varios temas de entre los definidos anteriormente. Opcionalmente, puede proporcionar otros parámetros como el factor de adivinanza, el factor de discriminación, la distribución en pantalla, etc. Dado que el enunciado y las respuestas permiten el uso de metacódigo y Applets, cada cuestión introducida puede ser en realidad un esquema generador de cuestiones, lo que permite una gran

variedad de preguntas. También es posible simular como quedara la pregunta al presentarla. Esto es especialmente útil en el caso de preguntas generativas.

Una vez definidas las preguntas de una materia, asociadas cada una de ellas a uno o varios temas, el profesor puede definir los tests que se van a realizar. Un test se compone de un conjunto de preguntas seleccionadas según diversos criterios tanto para la selección de preguntas como para la finalización del test. Por cuestiones practicas se fija un numero mínimo y máximo de preguntas para garantizar que en cualquier caso el test tiene un final, se alcance o no el criterio de finalización estadístico.

La nueva herramienta de edición mantiene las características ofrecidas por el anterior editor y añade otras nuevas. Permite dos modos de trabajo, según exista conexión permanente a Internet durante el proceso de modificación del test, o la conexión se realice únicamente al final para llevar a cabo la transferencia de las nuevas especificaciones de los tests al sistema SIETTE.

Esta herramienta muestra la estructura del currículum de la asignatura en forma de árbol. Para realizar una modificación o nueva inserción de un tema o de una cuestión, el profesor sólo tiene que seleccionar el elemento con el botón de la derecha del ratón y elegir la opción que desee.

Tras especificar los temas, el profesor debe añadir las cuestiones que compondrán el test. En el editor a través de la Web tanto el enunciado de las cuestiones como sus respuestas tenían que ser directamente insertados en código HTML. Este nuevo editor ofrece un conjunto de facilidades adicionales de edición del texto, ofreciendo ciertas ventajas al insertar código HTML, objetos multimedia y applets Java, ya que dispone de un entorno mas elaborado. En concreto, permite configurar fácilmente los parámetros necesarios para un conjunto de applets predefinidos que forman la biblioteca de ítems.

3.2.4 El modulo de Comprobación y Activación de Tests

El modulo de comprobación y activación de tests es un proceso fuera de línea que se ejecuta en el servidor donde esté ubicado el motor de base de datos cada cierto

tiempo. Sólo de este modo las modificaciones o creaciones de nuevas especificaciones de tests, realizadas por los profesores, se harán visibles en el generador de tests, manteniendo la coherencia en el caso en que se usen varios editores.

Este módulo comprueba que la definición de las preguntas y los tests es coherente desde el punto de vista computacional, por ejemplo, si existe un número suficiente de preguntas de cada tema, si se han incluido suficientes respuestas alternativas para cada pregunta, etc.

Por otra parte la separación entre la base de conocimientos empleada en la generación de tests y la base de conocimientos que crea el profesor permite separar las tareas de edición del test de la realización, eliminando los posibles problemas de inconsistencia en el caso de edición y realización simultanea. Un test en ejecución no se ve alterado por la inclusión, modificación o eliminación de ítems.

4. TIPOS DE ÍTEMS DE SIETTE

SIETTE es capaz de trabajar con diferentes tipos de ítems, que pueden ser combinados en el mismo test utilizando el mismo modelo de respuesta. A continuación se detallan cada uno de ellos.

4.1 Ítems Dicotómicos

Estos ítems son los más utilizados en los tests. Se caracterizan en que solamente pueden tener dos tipos de respuestas: verdadero o falso. Esto limita mucho el tipo de preguntas que se pueden proponer, pues se debe plantear para responderse de forma verdadero-falso. Un ejemplo de este tipo de ítems se muestra en la siguiente figura:

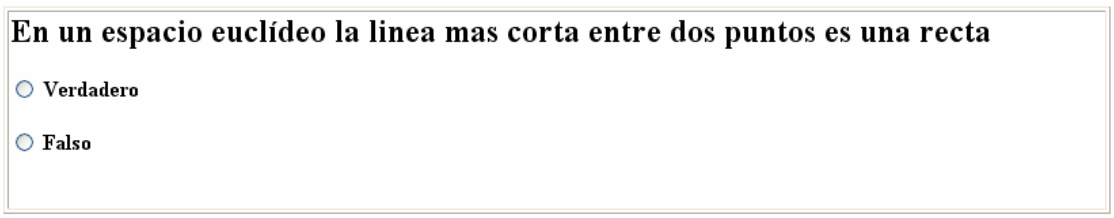


Figura 4.1: Ítems dicotómicos

4.2 Ítems Opción Múltiple

Son aquellos que tienen más de dos respuestas para un mismo enunciado, entre las cuales el alumno debe seleccionar sólo una. SIETTE dispone las alternativas aleatoriamente cada vez que muestra la pregunta. Un ejemplo de este tipo de ítems se muestra en la siguiente figura:

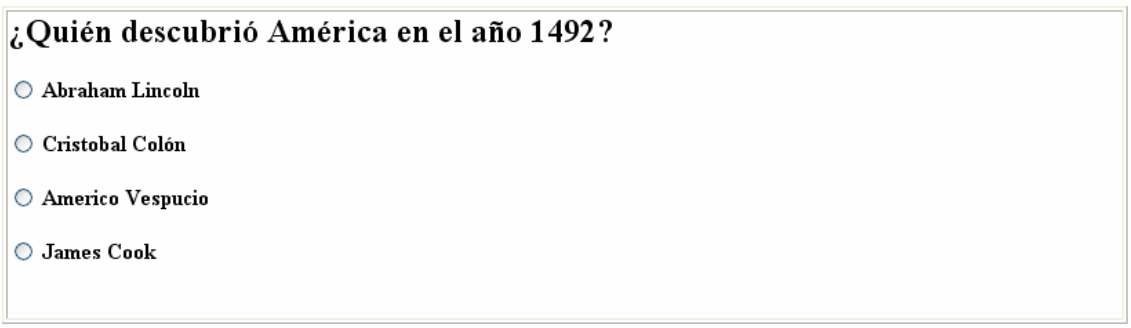


Figura 4.2: Ítems de opción múltiple

Desde el punto de vista de las curvas características, los ítems de opción múltiple en SIETTE se definen asociando a cada posible respuesta un vector característico de probabilidad condicionada. Las posibles respuestas a un ítem de opción múltiple con *r* respuestas alternativas son *r+1*, una por cada una de las posibles respuestas y otra más que corresponde a la contestación en blanco.

Dado que los ítems dicotómicos no son más que un tipo especial de ítems de opción múltiple con solo dos opciones, en la implementación de SIETTE se utiliza internamente este tipo de ítems en ambos casos.

4.3 Ítems Politomicos de Respuesta Independiente

Los ítems politomicos son ítems en los que la respuesta viene dada como un conjunto de opciones entre las que el alumno debe seleccionar aquellas que considera correctas. En este primer caso se considera que las opciones son independientes entre sí, por lo que la evaluación de una respuesta (una selección) no tiene ninguna influencia en la corrección de las restantes. En todo caso, la evaluación de estos ítems será mejor cuantas mas selecciones sean correctas. El efecto de plantear uno de estos ítems politomicos es el mismo que plantear un ítem dicotómico para cada una de las opciones. Por ejemplo, el ítem politómicos mostrado en la Figura 4.3 seria equivalente al siguiente conjunto de ítems dicotómicos: ¿Pertenece Francia con pleno derecho a la Unión Europea en el año 2001?, ¿Pertenece Japón con pleno derecho a la Unión Europea en el año 2001?, etc.

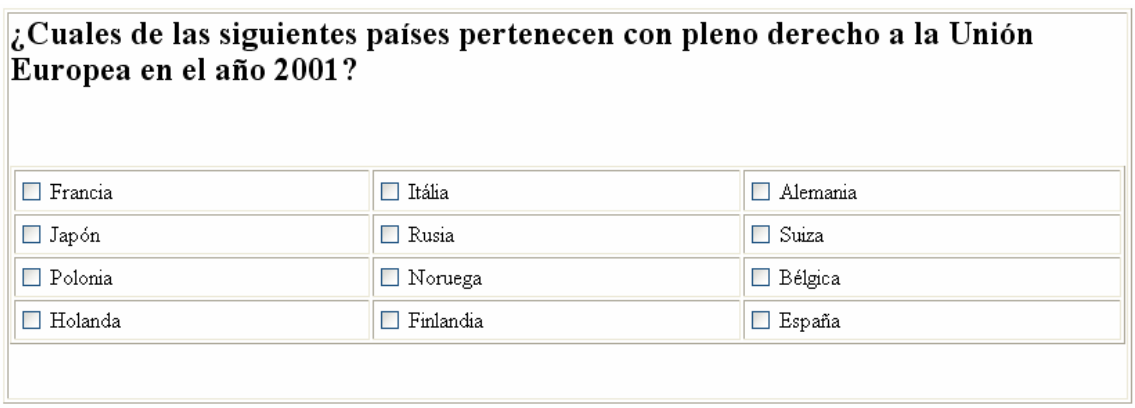


Figura 4.3: Ítems politómicos de respuesta independiente

4.4 Ítems Politomicos de Respuesta Dependiente

Un caso diferente aunque externamente similar al anterior se plantea cuando no puede suponerse independencia entre las respuestas a cada una de las opciones. En este caso la opción correcta es solamente una combinación de un cierto numero de opciones, aunque ocasionalmente mas de una combinación puede resultar valida. En el ítem de la Figura 4.4 la solución es: carbón, azufre y clorato potásico; aunque también es correcta la combinación carbón, azufre y nitrato potásico.

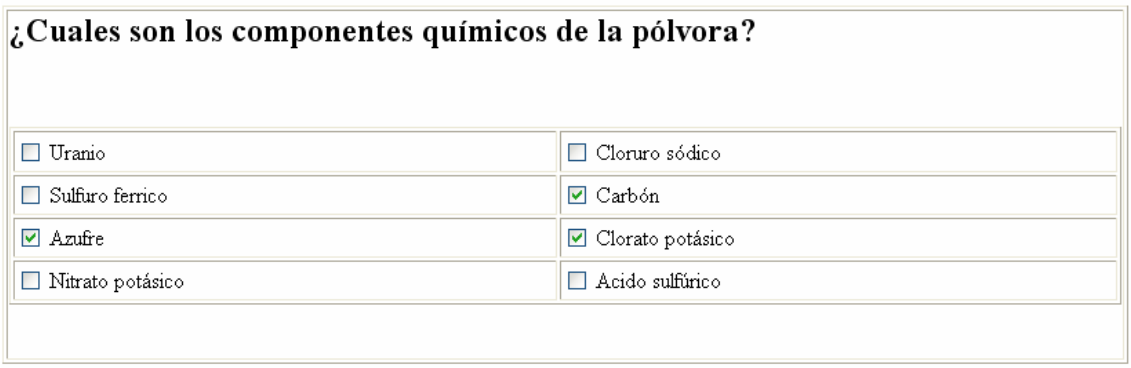
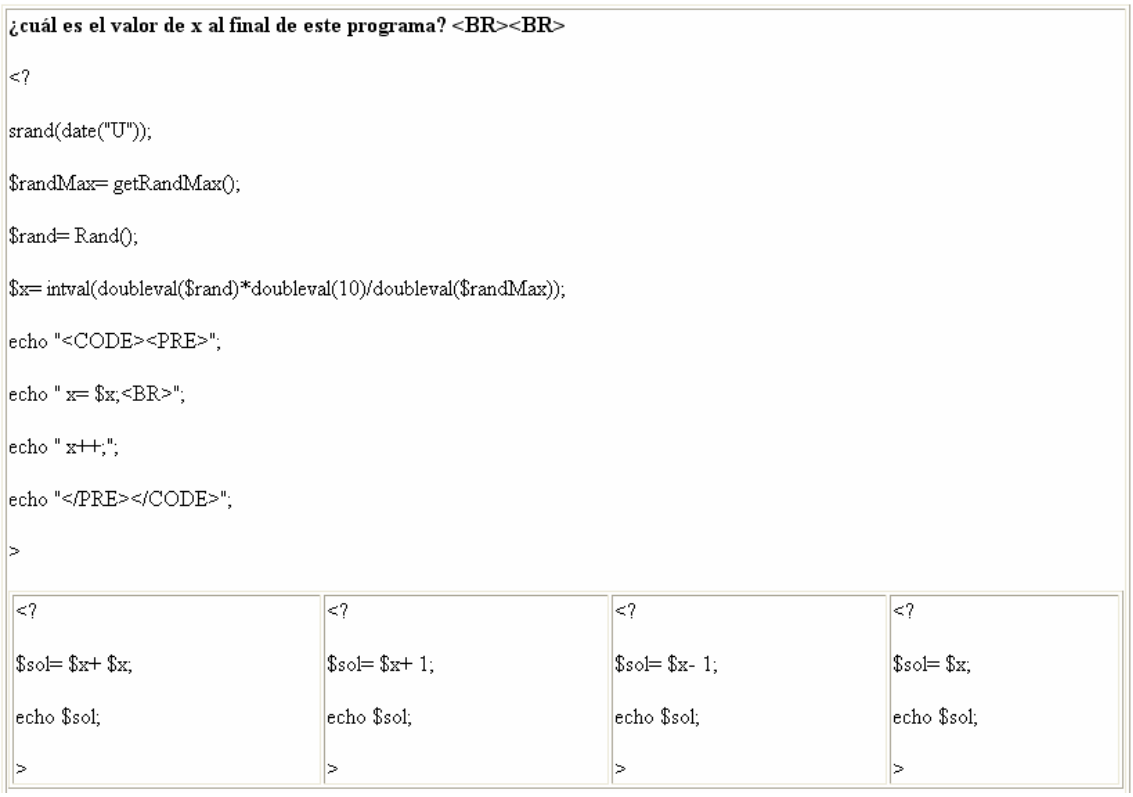


Figura 4.4: Ítems politómicos de respuesta dependiente

Este tipo de ítems es equivalente a un conjunto de ítems dicotómicos que preguntan si una combinación de respuestas es verdadera. Por consiguiente, el cardinal del conjunto de ítems dicotómicos es el factorial del número de posibles respuestas.

4.5 Plantillas Generativas de Ítems

Estas plantillas o esquemas de ítems se instancian en tiempo de ejecución dando origen a un ítem concreto. El desarrollo de estos esquemas es independiente del tipo de ítem. Con la definición de esquemas se minimiza el riesgo de plantear preguntas repetidas, y de que el alumno pueda llegar a memorizar las preguntas existentes en el banco. El desarrollo de los esquemas se implementa actualmente mediante el uso del lenguaje PHP que es una extensión del lenguaje HTML que se interpreta en el servidor, dando lugar dinámicamente a un texto en HTML. Otros tipos de lenguajes y extensiones de HTML como PERL, JSP, etc. serian también admisibles si se encuentran instalados en el servidor. En la Figura 4.5 podemos observar un ejemplo de estas plantillas en PHP.



```
¿cuál es el valor de x al final de este programa? <BR><BR>
<?
srand(date("U"));
$randMax= getRandMax();
$rand= Rand();
$x= intval(doubleval($rand)*doubleval(10)/doubleval($randMax));
echo "<CODE><PRE>";
echo " x= $x;<BR>";
echo " x++;";
echo "</PRE></CODE>";
>
```

<? $sol= $x+ $x; echo $sol; >	<? $sol= $x+ 1; echo $sol; >	<? $sol= $x- 1; echo $sol; >	<? $sol= $x; echo $sol; >

Figura 4.5: Esquema de ítem en PHP

El código en PHP de la Figura 4.5 es guardado en la base de conocimiento junto al resto de los ítems. La primera fila de la figura corresponde a la pregunta y a la base del programa, y cada columna de la segunda fila a cada una de las respuestas posibles. Una vez el ítem ha sido seleccionado, el generador preprocesa este esquema mandándolo al servidor de la Web dentro de una pagina HTML. El servidor interpreta la página y tras capturar el resultado del código en PHP, construye la pregunta en HTML que reemplaza al esquema de PHP. Al final, el ítem es mostrado a los estudiantes como se ve en la Figura 4.6.

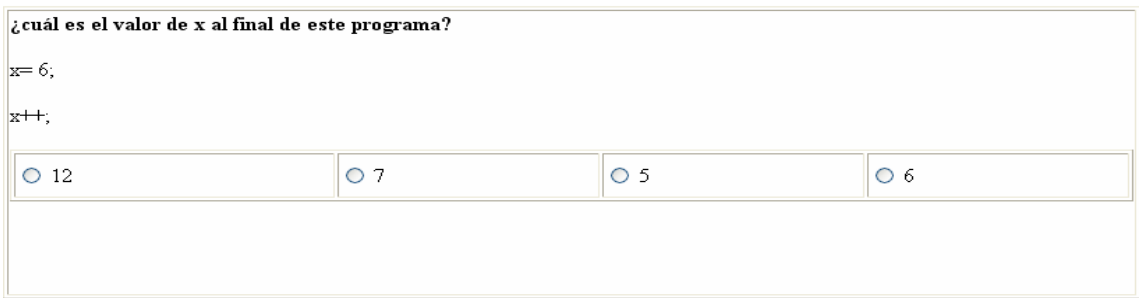


Figura 4.6: Ítem generado a partir del esquema de la Figura 4.5

Los esquemas de preguntas son una herramienta potente, pero a la vez complicada y no exenta de complicaciones. En primer lugar hacen necesaria el conocimiento del

lenguaje PHP o de cualquier otro que se emplee, lo cual no facilita precisamente la creación de plantillas a los no-programadores. Este problema es de difícil solución salvo para dominios concretos en los que se puedan desarrollar bibliotecas de funciones y herramientas especificas de edición para ayudar a los profesores no-programadores en la creación de esquemas. Por otra parte, es necesario un especial cuidado cuando se trabaja con esquemas, dado que ciertas instancias de un esquema pueden ser incorrectas como ítems. El esquema presentado anteriormente como ejemplo falla para x=0 y para x=1, ya que daría origen a ítems con dos alternativas iguales.

4.6 Ítems Controlados por Applets de Java

Puesto que SIETTE presenta las preguntas al usuario por medio de páginas Web, una posibilidad interesante es la utilización de programas incluidos en estas paginas mediante los conocidos applets en lenguaje JAVA. Existen dos modalidades para la utilización de estos applets.

La primera consiste en la mera incorporación de los applets a las secciones de enunciado o a las secciones de respuesta. De esta forma es posible construir preguntas en las que el enunciado se muestra dinámicamente, se pide al usuario que lo observe, y finalmente se le plantean varias opciones entre las que ha de decidir. Este mecanismo permite incluir en SIETTE temas y enunciados que midan capacidades sensoriales, visuales y/o auditivas; capacidad de atención y percepción, etc. para el desarrollo de este tipo de preguntas, y su incorporación a un test realizado mediante SIETTE, solo es necesario escribir de forma independiente el applet e incluirlo mediante el editor en la sección de código HTML correspondiente al enunciado o las respuestas, al igual que cualquier otro elemento multimedia.

La segunda modalidad consiste en sustituir el mecanismo de evaluación de las preguntas que hace SIETTE incluyéndolo en el propio applet. En este caso, se plantearía al alumno un enunciado que contiene un pequeño programa que se ejecuta y muestra al usuario dinámicamente y que recoge su respuesta de forma interactiva. No son necesarias por tanto en este caso utilizar un conjunto de respuestas fijo, de entre las cuales el usuario debe elegir una, sino que el propio

applet que aparece en el enunciado incluye la posibilidad de determinar si la respuesta es o no correcta. Este modo ofrece posibilidades de evaluación sensiblemente mejores que los métodos basados en selección de opciones. Por ejemplo mediante este tipo de preguntas es posible minimizar el efecto de los factores de adivinanza, controlar otros factores como el tiempo de respuesta, medir conocimientos espaciales difícilmente medibles mediante un catalogo de opciones, etc.

Este tipo de preguntas pueden integrarse en un mismo test junto con otras multirespuesta. El mecanismo de engranaje lo lleva a cabo el propio applet que se encarga de evaluar en términos de las posibles alternativas, la solución que ha obtenido el alumno mediante la interacción con el applet. Finalmente la respuesta es tratada al igual que un ítem politomico.

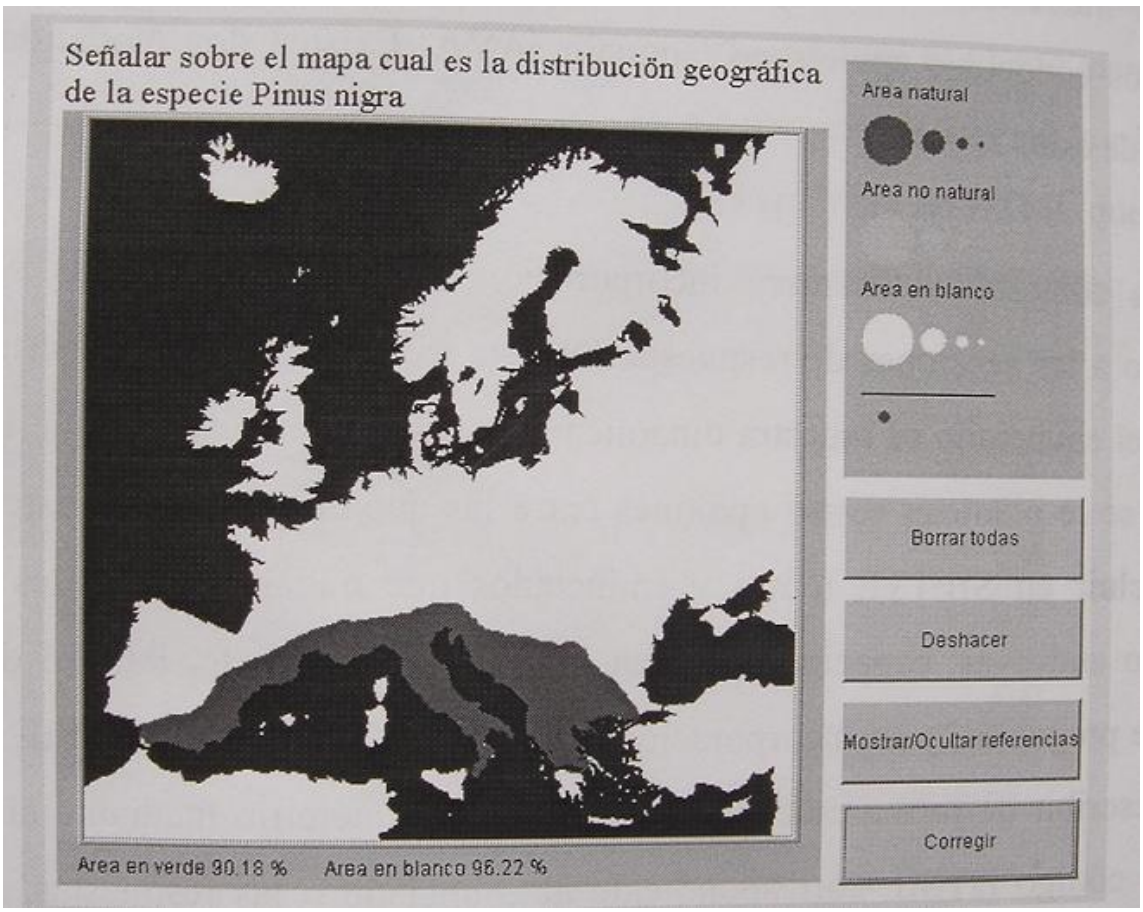


Figura 4.7: Ítems controlados por Applets en Java

La figura 4.7 muestra un ejemplo de este tipo de preguntas preparadas como parte de un test de conocimientos botánicos. En concreto se pretende averiguar como parte

del conocimiento de esta materia, si el alumno conoce la distribución geográfica de una cierta especie. Para ello mediante un applet se muestra un mapa de Europa sobre el cual, usando una brocha verde para pintar y blanca para borrar, al estilo de los programas de dibujo, el usuario debe señalar en verde la localización de la especie. Una vez completada esta tarea, el usuario deberá pulsar el botón de "Corregir", y el applet comparará la distribución indicada por el alumno con la distribución patrón preestablecida, evaluando si la respuesta es correcta o no, de acuerdo con los márgenes de error que se consideren admisibles y que han sido programados en el applet. Una vez determinado el tipo de respuesta, el applet transmite al sistema SIETTE el resultado de la evaluación. La figura 4.5 muestra este ítem justo después de pulsar el botón "Corregir".

Gracias a este método SIETTE amplia sus posibilidades al permitir la inclusión de cualquier conjunto de ítems que puedan ser implementados por Applets en Java, aunque ciertamente esta posibilidad esta limitada a desarrolladores de tests con conocimientos de programación en Java.

5. SISTEMA QTI

5.1 ¿Qué Es?

Una parte indispensable de la enseñanza es su método de evaluación al alumno. En la enseñanza electrónica también se considera fundamental ofrecer, cómo recta final del proceso de aprendizaje, una herramienta que permita examinar el trabajo realizado por la persona que trata de aprender.

Para ello nace esta especificación de IMS, que ha sido creada con la intención de ofrecer una estructura básica que describa la forma de representar evaluaciones (assessments) y sus calificaciones correspondientes. El objetivo es conseguir una forma de representar exámenes y resultados de manera que estos sean intercambiables entre los diferentes LMS (Learning Management System). Así, podríamos disponer de almacenes de preguntas y bases de datos con los resultados obtenidos por los alumnos a los que cualquier sistema de enseñanza electrónica podría acceder.

5.2 Qti y el Proceso de Enseñanza

En el siguiente gráfico se ilustra dónde encaja esta nueva especificación dentro del complicado proceso de enseñanza, y las partes de las que se compone.

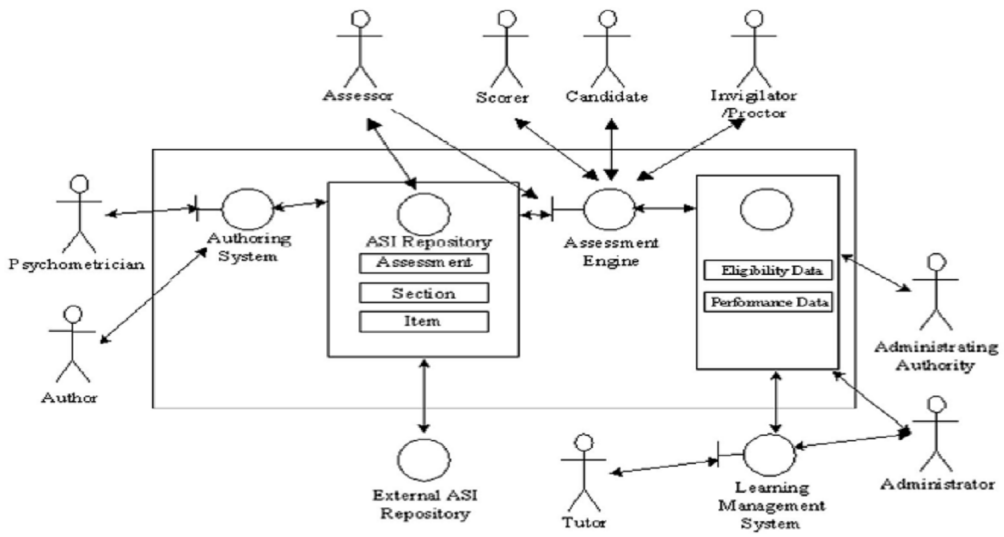


Figura 5.1. Componentes del sistema de exámenes. (IMSQTI_INFO 2002)

Aquí se pueden diferenciar las siguientes partes (IMSQTI_INFO 2002):

- **Sistema de autoría (authoring system). Es el encargado de facilitar la creación y edición de los exámenes. Estos están compuestos por Ítems, Sections, y Assessments (representados por las siglas ASI).**
- **Repositorio ASI (ASI Repository). Es una base de datos local dónde se almacenan los ASIs. Esta base de datos debe proporcionar funcionalidades para la búsqueda y recuperación de los objetos que almacena para así facilitar el trabajo al sistema de autoría, al motor de evaluación, o para cualquier consulta independiente. El External ASI Repository se refiere a la capacidad de importar objetos de un repositorio externo gracias a la interoperabilidad de esta especificación.**
- **Motor de evaluación (assessment engine). Se encarga de generar los resultados de los test en función de los objetos del repositorio ASI que se hayan usado, ya que cada uno de ellos lleva asociada una puntuación que se le otorga al candidato en caso de que se responda correctamente.**
- **Repositorio de datos de los candidatos (Elegibility Data, y Performance Data). Es dónde se almacenan los datos de los resultados obtenidos por los alumnos en cada uno de los exámenes. Es importante remarcar que este repositorio es accesible directamente por el LMS (Learning Managment System), y que no es parte de la especificación *Learner Information Profile*, aunque ya se ha propuesto la inclusión de estos datos en versiones posteriores.**

Learning Managment System o LMS, es el proceso o sistema responsable de toda la arquitectura de enseñanza.

5.3 QUE ASPECTO TIENE

Después de una serie de tentativas por parte de IMS de definir cuales son las partes de las que se compone un examen, se llega a la especificación ASI. Por tanto, ASI no es más que una forma de representar un examen y las diferentes partes de las que se compone.

Las siglas ASI son la abreviatura de: Assessment, Section, Item, que son las diferentes partes de una evaluación según esta especificación. A continuación explicamos el significado de cada una:

- **Item. Se usan para la representación de las preguntas individuales. Contienen además de la pregunta, información de presentación de la misma, la respuesta correcta, información para la evaluación de las posibles respuestas, las pistas para ayudar al alumno a responderla, y metadatos sobre la misma. Estos objetos no permiten anidamiento, es decir, un ítem no puede estar compuesto por otros, lo que hace de ellos la unidad mínima de intercambio en IMS QTI.**
- **Section. Se usan para estructurar los exámenes y para ayudar al posterior secuenciamiento de las preguntas. Cada sección puede estar compuesta por ítems o por otras secciones.**
- **Assessments. Se usan cómo contenedores de los anteriores y su equivalente sería el examen que trata de determinar los conocimientos del que lo realiza. Tienen que contener una sección cómo mínimo ya que no pueden albergar en su interior ítems directamente. Poseen la información de secuenciamiento de las secciones y gracias a la puntuación que cada ítem lleva asociada, son capaces de generar la puntuación total del test.**

Además de estas definiciones hay una más que debemos tener en cuenta, se trata de los Object Bank, que son cualquier agrupación de objetos, ya sean Sections o Ítems. Por tanto contiene los objetos necesarios para la creación de Assessments.
La definición de este nuevo elemento responde exclusivamente a la idea de facilitar la organización de objetos en los repositorios de exámenes.

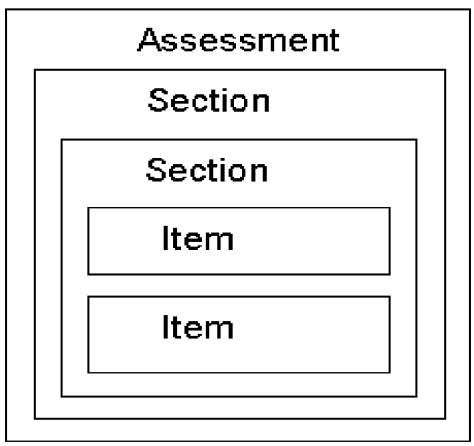


Figura 5.2. Estructura propuesta por la especificación ASI.

5.4 UNA CLASIFICACION DE LOS TIPOS DE ELEMENTOS DE UNA EVALUACION BASADA EN RESPUESTAS

Cómo hemos visto, el objeto fundamental definido en esta especificación es el Item. Pero surge un problema a la hora de clasificar estos nuevos elementos. No pueden hacerse según la información que almacena ya que esta es variada. Tampoco se los puede clasificar según la pregunta que plantean porque estas a su vez pueden resultar inclasificables. Y es aquí dónde nos encontramos ante una idea innovadora. Para solucionar este problema, se ha optado por clasificarlas por el tipo de respuesta que ofrecen. En la siguiente figura observamos la taxonomía que se ha creado basándose en las respuestas.

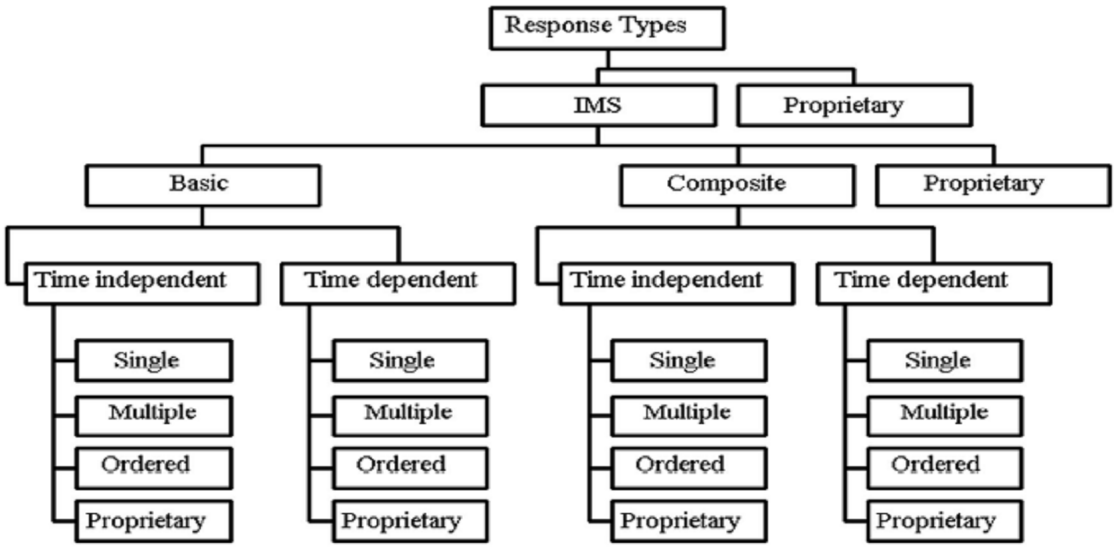


Figura 5.3. Taxonomía del tipo de respuesta

La primera clasificación con que nos encontramos si seguimos el camino de IMS es:

- **Respuesta Básica. Es aquella que contiene un solo tipo de respuesta.**
- **Respuesta Compuesta. Funciona cómo un contenedor de respuestas en el que podemos tener respuestas de diferentes tipos.**
- **Propietary (propietarias). Este campo nos indica que cualquiera que adopte esta especificación puede definir su propio tipo de respuesta de acuerdo con sus intereses.**

La siguiente bifurcación nos lleva a dos tipos nuevos de respuestas:

- **Respuestas dependientes del tiempo. En las cuales el tiempo que se tarda en responder la pregunta es relevante y tiene que ser almacenado. Este tipo de respuestas resultan útiles para preguntas en las que el examinador quiere tener en cuenta la rapidez de contestación del candidato.**
- **Respuestas independientes del tiempo. Dónde el tiempo tardado en responder es irrelevante y no se almacena.**

La última clasificación depende de lo que el alumno debe hacer para contestar la pregunta:

- **Single (simple). La respuesta es sólo una.**
- **Múltiple. Existen varias respuestas válidas.**
- **Ordered (ordenadas). No sólo existen varias respuestas sino que el orden de las mismas también es importante.**

Una vez vista la taxonomía de las respuestas por la que quedan clasificados los Ítems, vamos a ver los tipos de respuestas que propone y que admite esta especificación. Nos ofrecen cinco tipos de respuestas. Estos cinco tipos pasan a ser el cuerpo de la clasificación, mientras que las tres categorías (Simples, Múltiples y Ordenadas) pasan a ser atributos de esos cinco tipos. En la Tabla 5.1 se muestran los tipos básicos que se pueden utilizar, a estos habría que añadirle los tipos compuestos, que no son más que combinaciones de los básicos, y los propietarios.

Tipo de respuesta	Estructura de datos	Formatos de presentación		
		Simples	Múltiples	Ordenadas
Identificador Lógico (LID)	Cada respuesta o una lista de las respuestas puestas en orden.	-Múltiples posibilidades. -Verdadero/ Falso. -Número acotado.	-Respuesta múltiple.	-Ordenar objetos. -Conectar los puntos. -Objeto coincidente -Ordenar objetos. -Llevar un objeto. -Acumular en destino
Coordenadas X-Y (XY)	Las coordenadas x-y del centro del objeto o una lista ordenada de las coordenadas.	-Imagen sensitiva.	-Ordenar objetos.	-Conectar los puntos.

Cadena (STR)	La cadena para cada respuesta.	-Rellenar el espacio en blanco. -Seleccionar un texto -Respuesta corta - Redacción		
Numérica (NUM)	El número para cada respuesta.	-Rellenar el espacio en blanco - Número acotado.		
Grupos Lógicos (GRP)	Una serie de tuplas para cada grupo de objetos y para cada respuesta.	- Objeto coincidente - Llevar un objeto. - Acumular en destino.	- Objeto coincidente - Llevar un objeto. - Acumular en destino	- Objeto coincidente. - Ordenar objetos.

Tabla 2.5.1. Tipos básicos de respuestas.

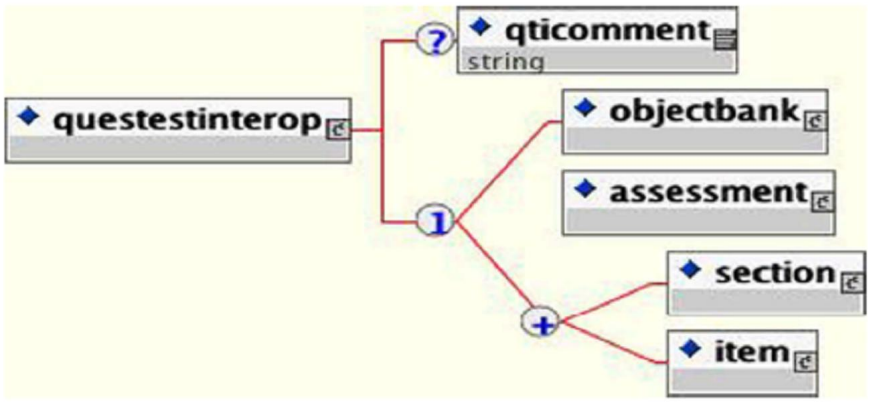
Uno de los grandes avances en esta clasificación de respuestas es que se instaura un vocabulario hasta ahora inexistente que nos servirá para denominar los objetos de un examen. Además, supone un estudio en profundidad de los tipos de respuestas, y por tanto de preguntas, que podemos utilizar a la hora de crear exámenes.

5.5 ESTRUCTURA DE FICHEROS XML PARA EL SISTEMA QTIASI 1.2

En este apartado describiremos el contenido y la estructura de los ficheros XML que cumplen con las especificaciones del sistema QTIASI. Utilizaremos la notación grafica en lugar de extraer el contenido del esquema o DTD de la especificación.

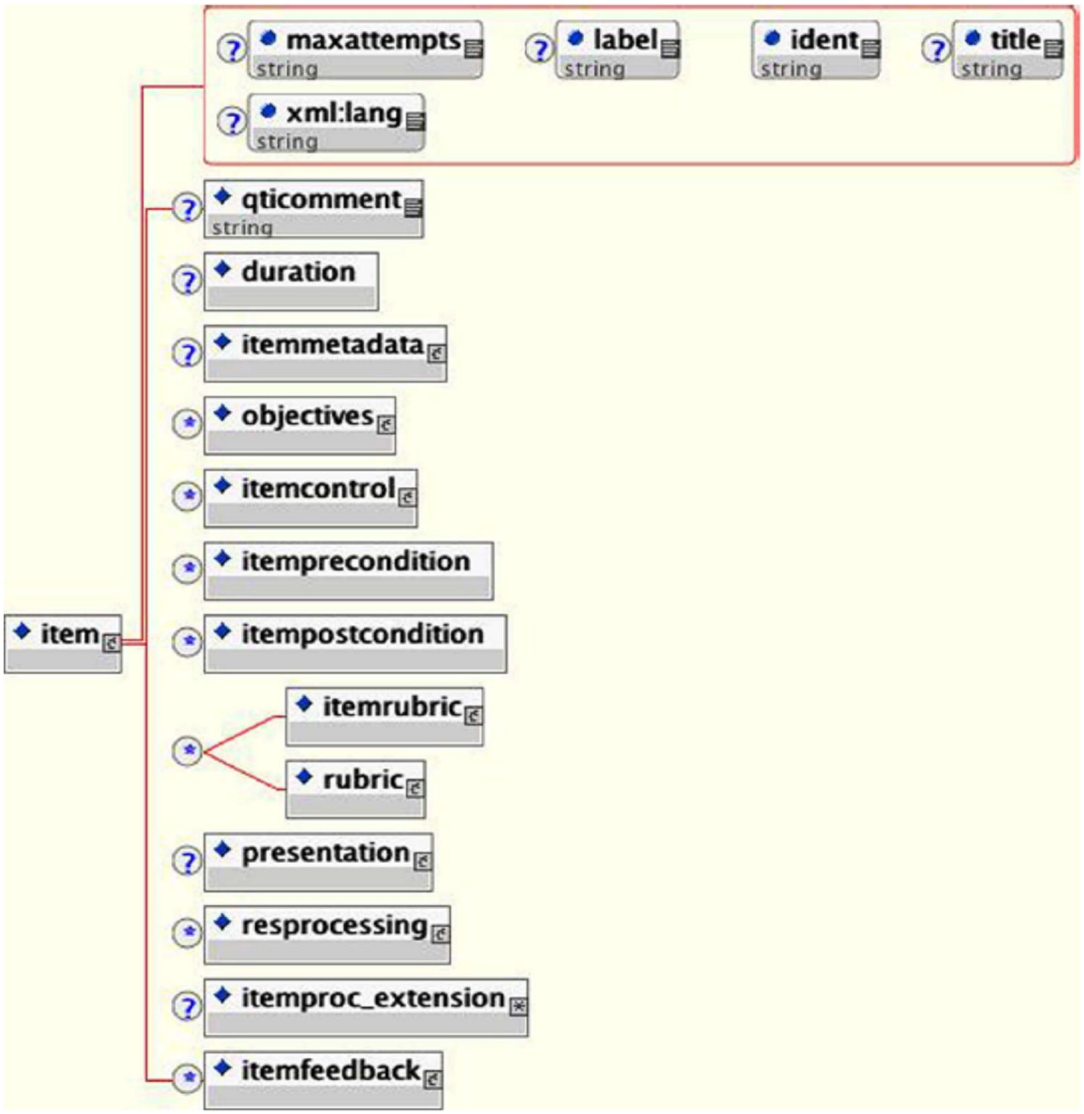
Hay que mencionar que en este proyecto se va a concentrar el trabajo sobre el elemento *ítem* (los elementos *objectbank*, *assessment* y *section* quedan fuera del objetivo de este proyecto fin de carrera, este hecho es debido simplement a que el sistema SIETTE es el encargado de suministrar ítems al reproductor QTI), y además se va a explicar en este apartado solo los elementos que se habían implementado a la hora de realizar el reproductor (algunos elementos no son significativos ni a la hora de representar las preguntas ni tampoco a la hora de la evaluación de las mismas).

5.5.1 EL ELEMENTO <*questestinterop*>



<u>Descripción</u>: el principal elemento del esquema QTI es *questestinterop*, se usa para almacenar las preguntas con sus respuestas y rasgos de evaluación.

5.5.2 EL ELEMENTO <*item*>

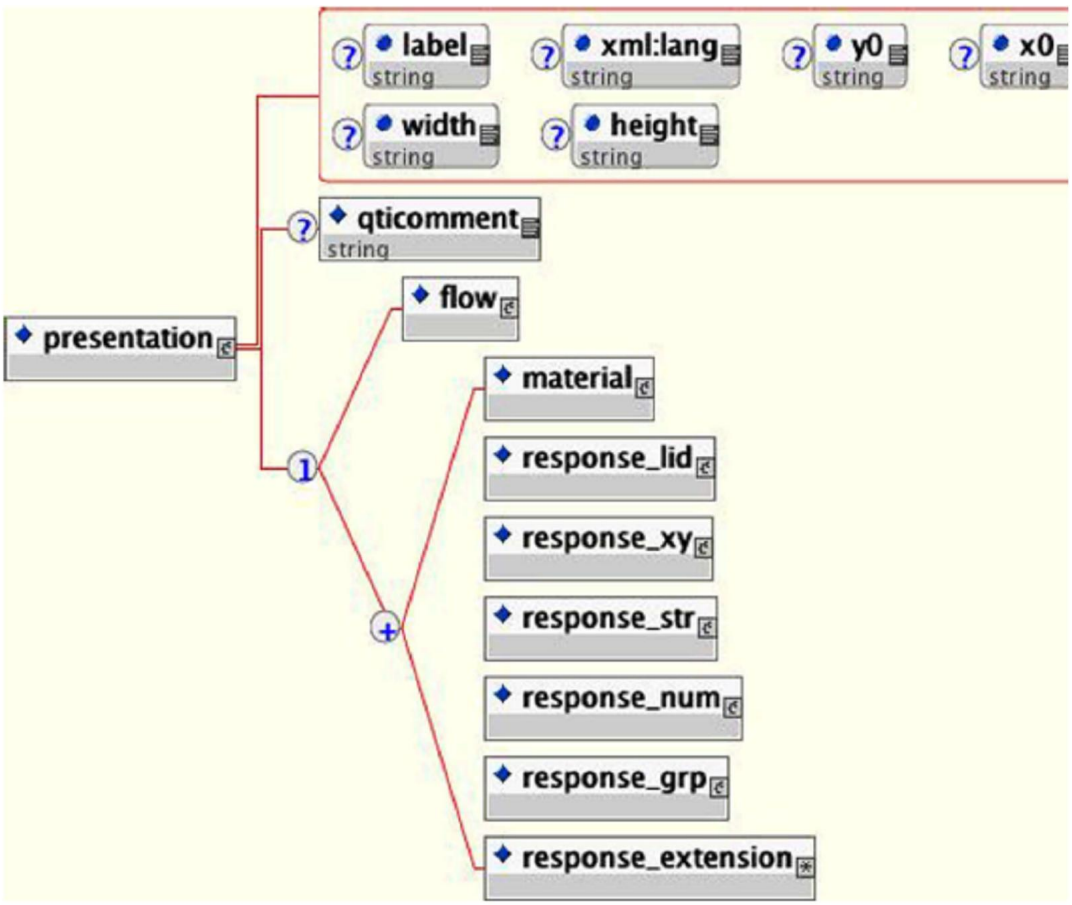


Descripción: **un ítem es una pregunta. Contiene las respuestas, refuerzos o "feedback" y otros elementos usados para la presentación al usuario y procesado de las respuestas.**

Atributos:

Nombre	Descripción	Uso
maxattempts	Numero de intentos permitidos para el usuario	opcional
label	Etiqueta que puede ser usada por las aplicaciones de creación de tests para identificar los rasgos importantes.	opcional
ident	Identificador único para el item	obligatorio
title	Titulo del ítem	opcional
xml:lang	El lenguaje por defecto del texto utilizado en el item	opcional

5.5.3 EL ELEMENTO <*presentation*>



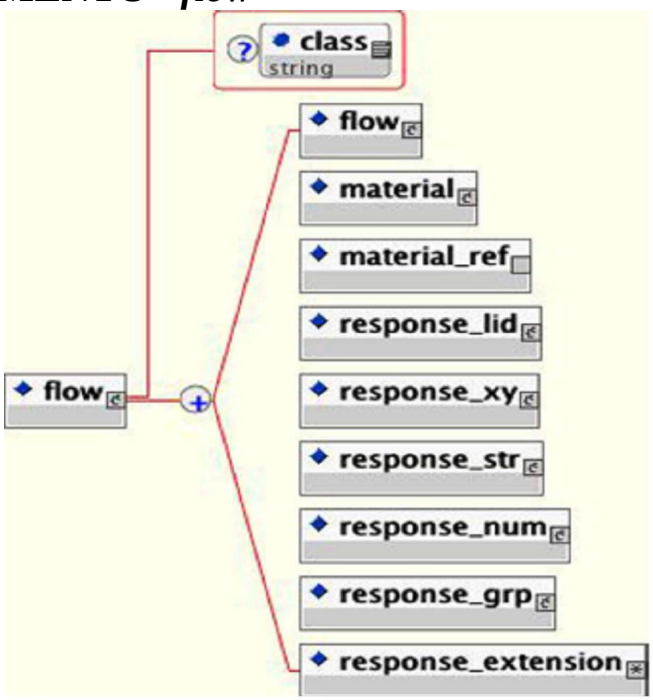
Descripción: **Este elemento contiene todas las instrucciones para la presentación de la pregunta durante el test. Esta información incluye el material utilizado para**

presentar la pregunta (textos, imágenes, etc.) y los identificadores de las posibles respuestas que serán utilizados después a la hora de la evaluación.

Atributos:

Nombre	Descripción	Uso
label	**Etiqueta que puede ser usada por las aplicaciones de creación de tests para identificar los rasgos importantes.**	**opcional**
xml:lang	**El lenguaje por defecto del texto contenido en la presentación del item.**	**opcional**
y0	**La coordenada del punto superior izquierdo del contenido de la presentación del item.**	**opcional**
x0	**La abscisa del punto superior izquierdo del contenido de la presentación del item.**	**opcional**
width	**El ancho del contenido de la presentación del item.**	**opcional**
height	**La altura del contenido de la presentación del item.**	**opcional**

5.5.4 EL ELEMENTO <*flow*>

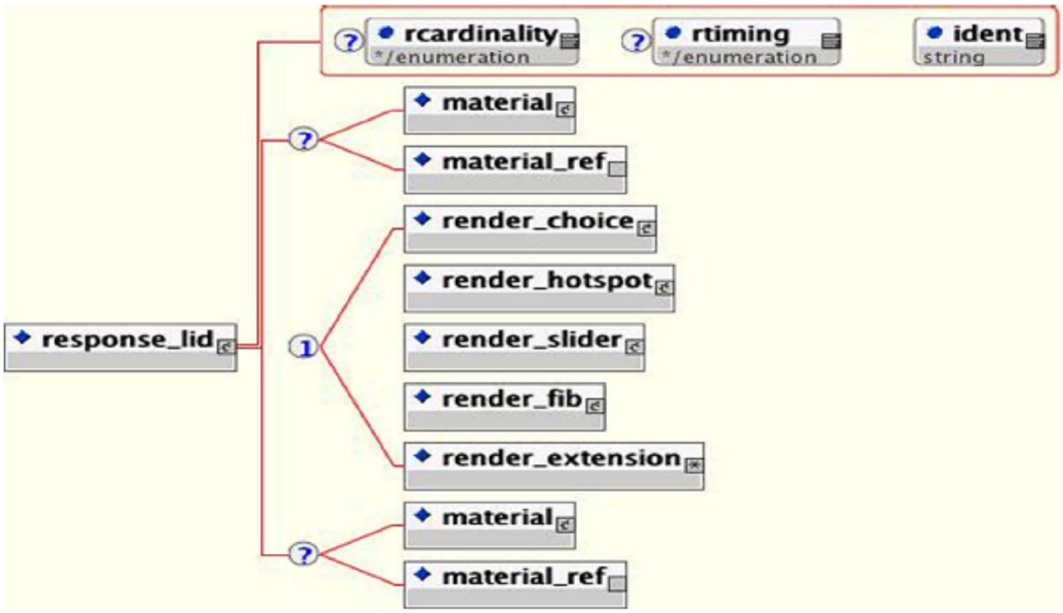


Descripción: Este elemento se utiliza para presentar la información al usuario en forma de párrafos o bloques.

Atributos:

Nombre	Descripción	Uso
class	**El tipo de párrafo o bloque que se quiere aplicar al contenido.**	**opcional**

5.5.5 EL ELEMENTO <*response_lid*>

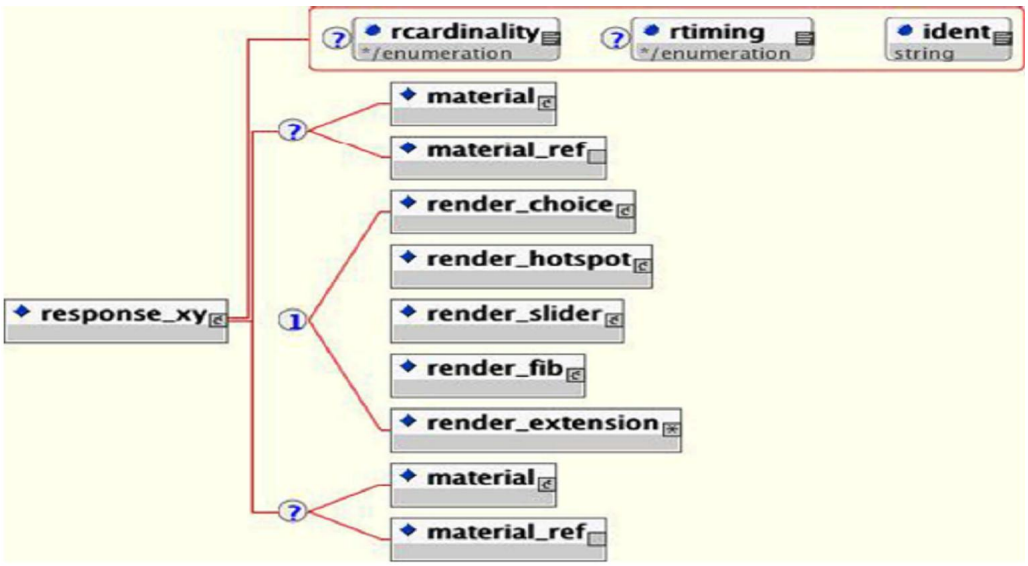


Descripción: **Contiene la lista de respuestas posibles para una pregunta y otros elementos que definen el como van a ser mostradas estas respuestas.**

Atributos:

Nombre	Descripción	Uso
rcardinality	Indica el número de respuestas esperadas del usuario, puede ser respuesta única o múltiple.	Opcional
rtiming	Indica si las respuestas serán temporizadas.	Opcional
ident	Identificador único para el elemento response_lid dentro del elemento presentaction. Es del tipo cadena y se usa dentro del elemento resprocessing para asegurar que es procesado el correspondiente conjunto de etiquetas.	Obligatorio

5.5.6 EL ELEMENTO <*response_xy*>

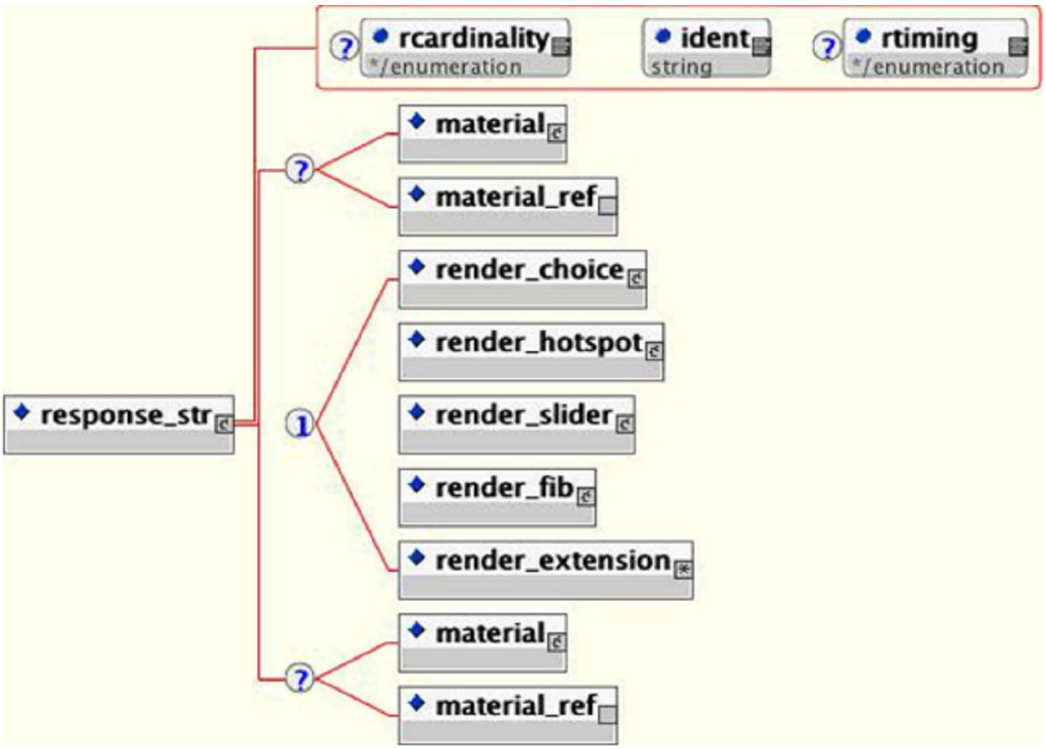


Descripción: **Este elemento contiene las instrucciones para la presentación de preguntas cuya respuesta será las coordenadas (x, y) de la respuesta elegida.**

Atributos:

Nombre	Descripción	Uso
rcardinality	Indica el número de respuestas esperadas del usuario, puede ser respuesta única o múltiple.	Opcional
rtiming	Indica si las respuestas serán temporizadas.	Opcional
ident	Identificador único para el elemento response_lid dentro del elemento presentaction. Es del tipo cadena y se usa dentro del elemento resprocessing para asegurar que es procesado el correspondiente conjunto de etiquetas.	Obligatorio

5.5.7 EL ELEMENTO <*response_str*>



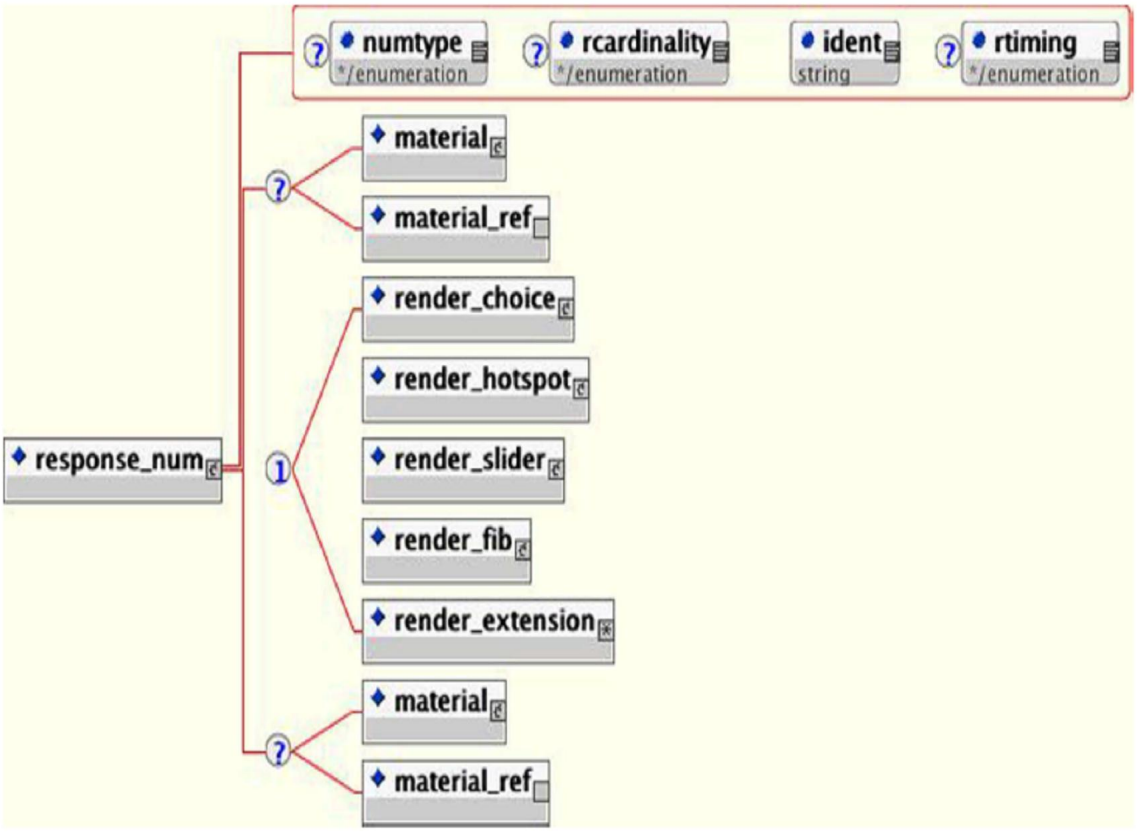
Descripción: **Este elemento contiene las instrucciones para la presentación de preguntas cuya respuesta será una cadena de caracteres.**

Atributos:

Nombre	Descripción	Uso
rcardinality	Indica el número de respuestas esperadas del usuario, puede ser respuesta única o múltiple.	Opcional
Rtiming	Indica si las respuestas serán temporizadas.	Opcional
ident	Identificador único para el elemento response_lid	Obligatorio

	dentro del elemento presentaction. Es del tipo cadena y se usa dentro del elemento resprocessing para asegurar que es procesado el correspondiente conjunto de etiquetas.	

5.5.8 EL ELEMENTO <*response_num*>

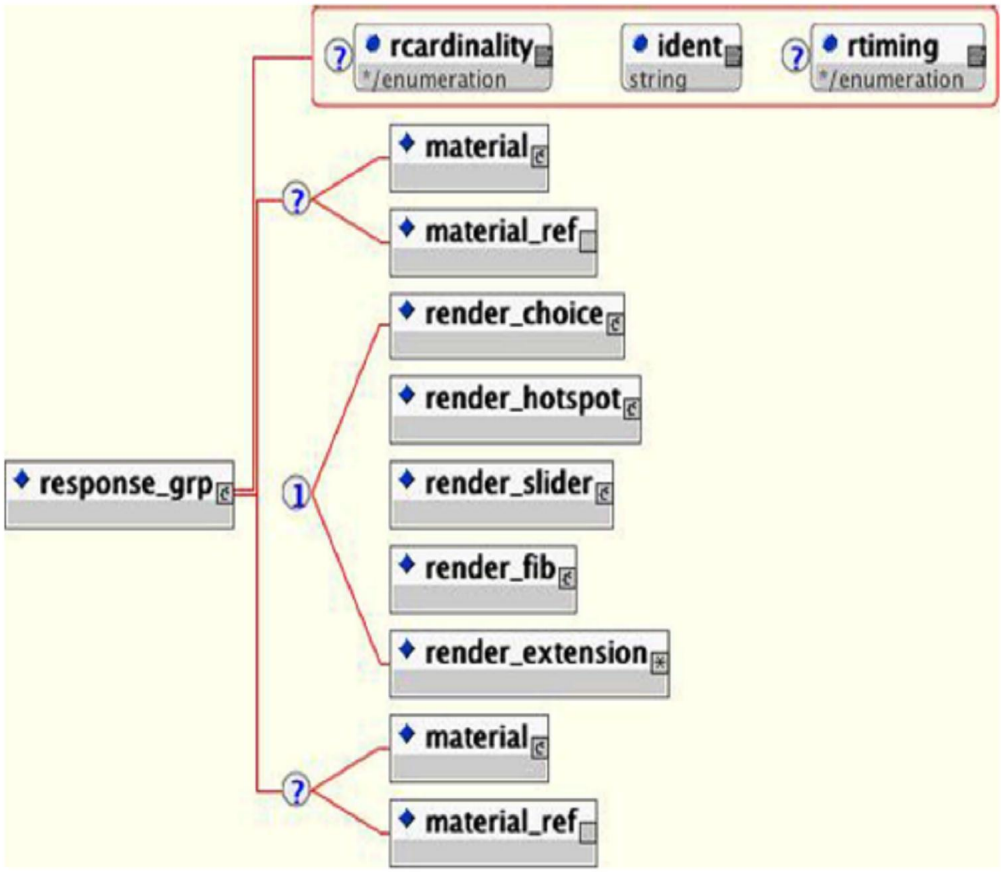


Descripción: **Este elemento contiene las instrucciones para la presentación de preguntas cuya respuesta será un número. Es igual que <*response_str*>, la única diferencia reside en que hay que interpretar la cadena de caracteres de la respuesta como un numero.**

Atributos:

Nombre	Descripción	Uso
numtype	**Indica el tipo de la respuesta numérica. Puede ser: Integer, Decimal, Scientific.**	**opcional**
rcardinality	**Indica el número de respuestas esperadas del usuario, puede ser respuesta única o múltiple.**	**Opcional**
rtiming	**Indica si las respuestas serán temporizadas.**	**Opcional**
ident	**Identificador único para el elemento response_num dentro del elemento presentaction. Es del tipo cadena y se usa dentro del elemento resprocessing para asegurar que es procesado el correspondiente conjunto de etiquetas.**	**Obligatorio**

5.5.9 EL ELEMENTO <*response_grp*>

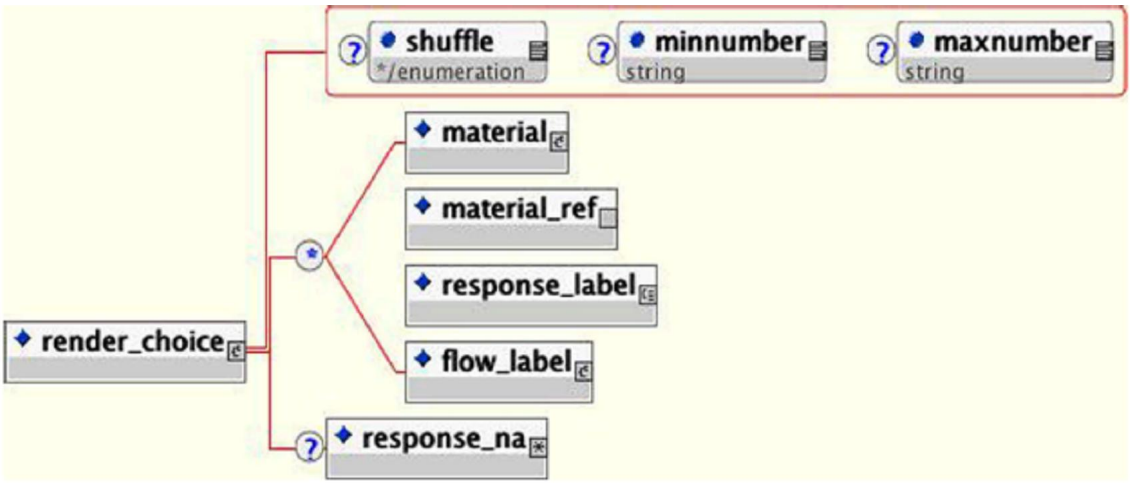


Descripción: **Este elemento contiene las instrucciones para la presentación de preguntas cuya respuesta será un grupo de identificadores lógicos. Se suele utilizar en las preguntas de relación.**

Atributos:

Nombre	Descripción	Uso
rcardinality	Indica el número de respuestas esperadas del usuario, puede ser respuesta única o múltiple.	Opcional
rtiming	Indica si las respuestas serán temporizadas.	Opcional
ident	Identificador único para el elemento response_lid dentro del elemento presentaction. Es del tipo cadena y se usa dentro del elemento resprocessing para asegurar que es procesado el correspondiente conjunto de etiquetas.	Obligatorio

5.5.10 EL ELEMENTO <*render_choice*>

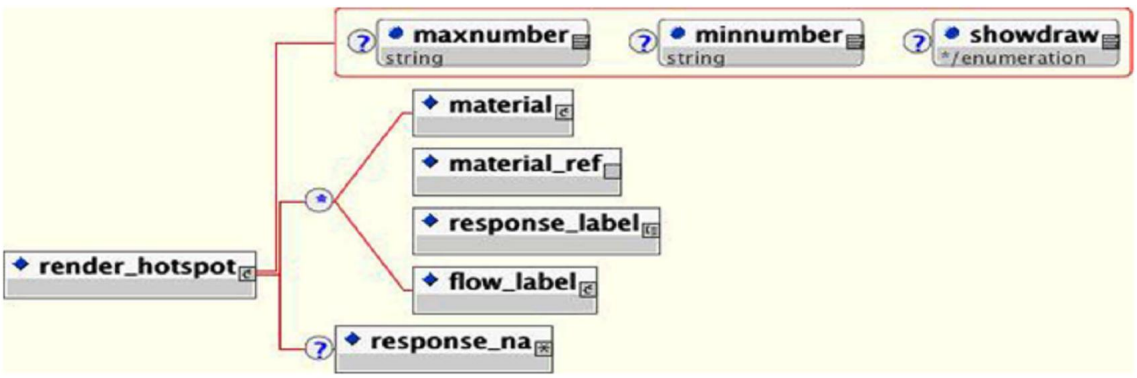


Descripción: **Este elemento indica al reproductor del QTI a presentar la pregunta utilizando el formato clásico de respuestas múltiple. El numero de respuestas posibles viene determinado por el numero de elementos <*response_label*> incluidos dentro de <*render_choice*>.**

Atributos:

Nombre	Descripción	Uso
shuffle	**Los valores que puede tomar son Yes o No. Indica si las posibles respuestas serán barajadas entre realizaciones consecutivas de la pregunta. Afecta a todas las respuestas, esto es, en apariciones consecutivas de la pregunta al usuario, todas las respuestas de la pregunta aparecerán en distinto orden.**	**opcional**
minnumber	**Numero mínimo de respuestas que deben ser proporcionadas por el usuario.**	**opcional**
maxnumber	**Numero máximo de respuestas que pueden ser proporcionadas por el usuario.**	**opcional**

5.5.11 EL ELEMENTO <*render_hotspot*>

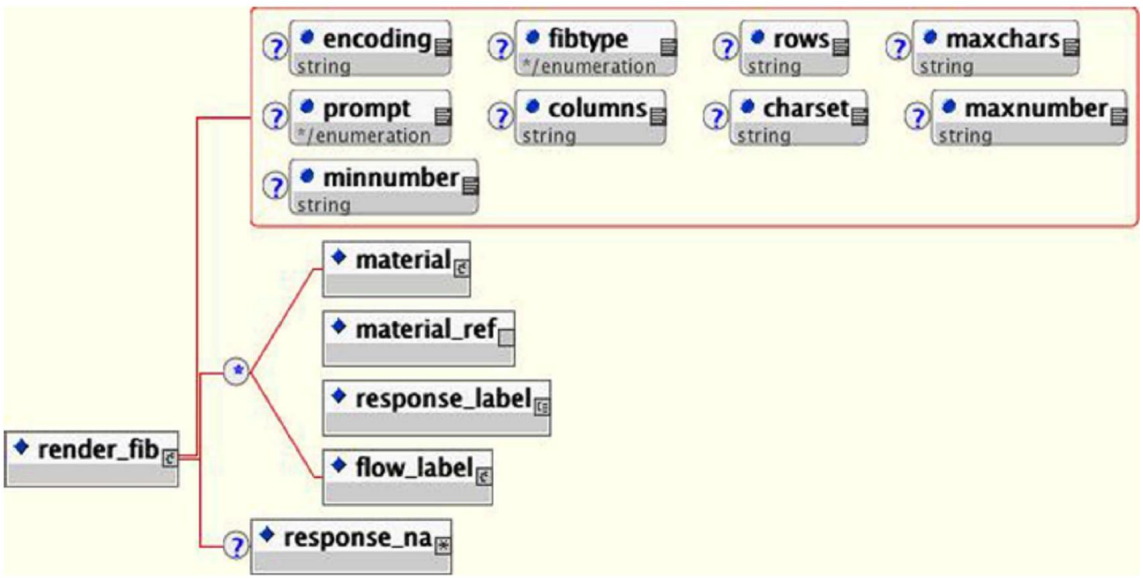


Descripción: **El elemento <render_hotspot> indica al procesador QTI a presentar la pregunta utilizando el clásico formato "image hot-spot format" (las posibles respuestas están distribuidas en puntos concretos a lo largo de una imagen).**

Atributos:

Nombre	Descripción	Uso
showdraw	**Informa al procesador QTI que las respuestas identificadas por el usuario deben ser conectadas (se utiliza líneas dibujadas para conectar las respuestas).**	**opcional**
minnumber	**Numero mínimo de respuestas que deben ser proporcionadas por el usuario.**	**opcional**
maxnumber	**Numero máximo de respuestas que pueden ser proporcionadas por el usuario.**	**opcional**

5.5.12 EL ELEMENTO <*render_fib*>



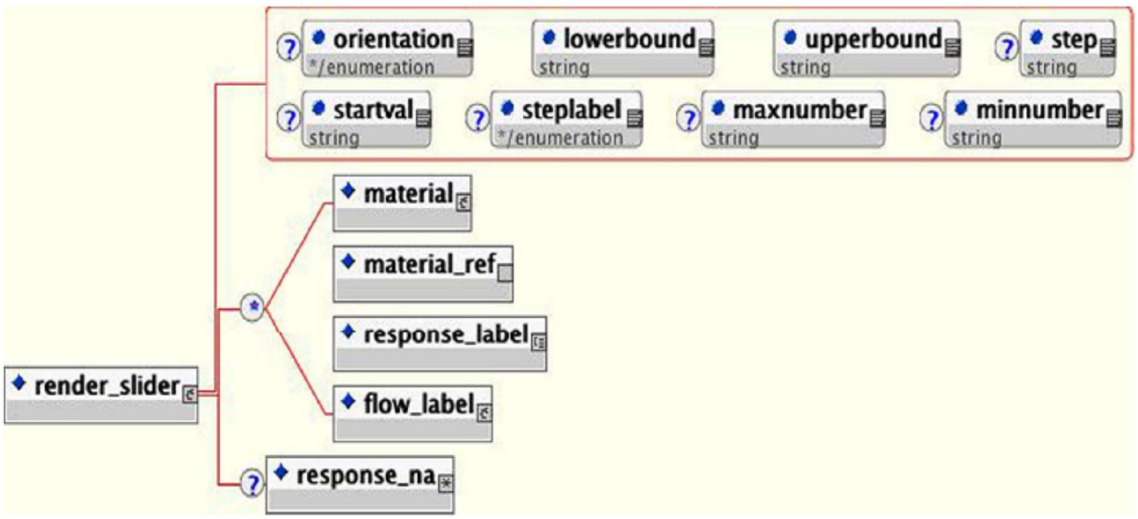
Descripción: **Este elemento indica al procesador QTI a presentar la pregunta utilizando el clásico formato de respuesta libre, es decir hay que dejar un espacio en blanco para que el usuario pueda introducir su respuesta directamente.**

Atributos:

Nombre	Descripción	Uso
encoding	**Indica el código utilizado para la presentación del texto, por defecto es 'UTF-8'**	**opcional**
fibtype	**Indica el tipo de datos que se espera del usuario, este atributo puede tomar los siguientes valores enumerados (String, Integer, Decimal, Scientific, Boolean).**	**opcional**

rows	**Indica el número de filas disponibles para la introducción de los datos.**	**opcional**
maxchars	**Indica el número máximo de caracteres disponibles para los datos.**	**opcional**
prompt	**Indica el tipo o la apariencia del espacio en blanco presentado al usuario donde hay que introducir los datos o la respuesta. Puede tomar la siguiente lista de valores (Box, Dashline, Asterisk, Underline)**	**opcional**
columns	**Indica el número de columnas disponibles para los datos.**	**opcional**
charset	**Indica el conjunto de caracteres que se utilizara para la presentación del texto.**	**opcional**
maxnumber	**Indica el número máximo de respuestas que el usuario puede proporcionar.**	**opcional**
minnumber	**Indica el número mínimo de respuestas que el usuario debe proporcionar.**	**opcional**

5.5.13 EL ELEMENTO <*render_slider*>



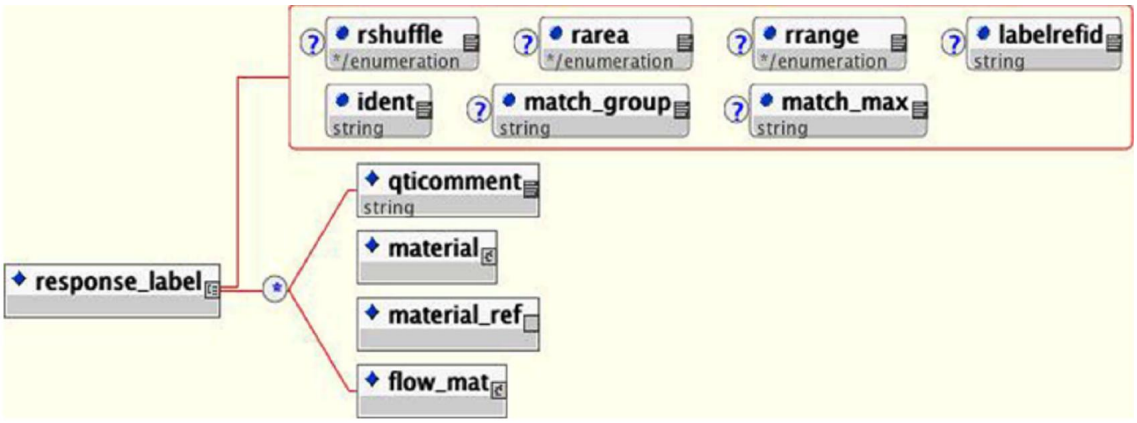
Descripción: **<render_slider> indica al procesador QTI que la pregunta debe ser presentada al usuario utilizando el componente grafico barra deslizante o escala.**

Atributos:

Nombre	Descripción	Uso
orientation	**Indica la orientación de la barra deslizante. Este atributo puede ser *Horizontal o Vertical*.**	**opcional**
lowerbound	**Indica el valor mas bajo de la barra deslizante.**	**obligatorio**
upperbound	**Indica el valor más alto de la barra deslizante.**	**obligatorio**
step	**Indica el valor de incremento de la barra deslizante**	**opcional**
startval	**Es el valor por defecto de la barra deslizante.**	**opcional**
steplabel	**Indica si los valores posibles deben ser mostrados o no.**	**opcional**
maxnumber	**Indica el número máximo de respuestas que el usuario puede proporcionar.**	**opcional**
minnumber	**Indica el número mínimo de respuestas que el**	**opcional**

	usuario debe proporcionar.	

5.5.14 EL ELEMENTO <*response_label*>



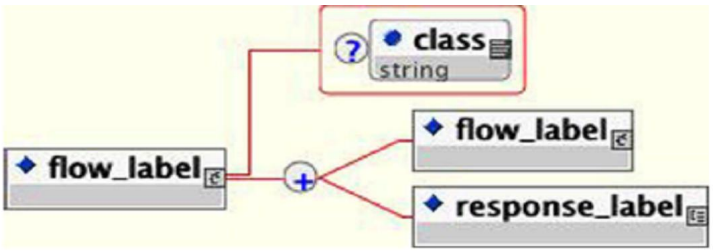
Descripción: **Cada elemento de este tipo es una opción o respuesta. Contiene tanto la etiqueta que será mostrada al usuario como información útil para procesar las respuestas en el elemento resprocessing.**

Atributos:

Nombre	Descripción	Uso
rshuffle	**Los valores que puede tomar son Yes o No. Indica si la respuesta a la que pertenece este atributo será barajada entre realizaciones consecutivas de la pregunta. Este atributo solo tiene sentido si el valor del atributo shuffle del elemento render_choice es Yes. Es útil con una respuesta del tipo "Todas las anteriores", ya que una requiere estar en una posición fija a diferencia del resto.**	**opcional**
rarea	**Indica el tipo de área o zona que se utilizara en la presentación de posibles respuestas en el caso de preguntas de tipo <render_hotpost>. Puede tomar la siguiente lista de valores (Ellipse, Rectangle, Bounded).**	**opcional**
rrange	**Indica la precisión de la respuesta numérica. Se utiliza en las preguntas de tipo libre.**	**opcional**
labelrefid	**Etiqueta que puede ser usada por las aplicaciones de creación de tests para identificar rasgos importantes.**	**opcional**
ident	**Identificador único para cada respuesta dentro del item. Es de tipo cadena y es usado por el mecanismo de procesado para identificar la respuesta seleccionada.**	**obligatorio**
match_group	**Es una lista separada por coma que define el conjunto de identificadores de los <response_label> a los que este objeto puede ser relacionado. Se utiliza en las preguntas de**	**opcional**

	relación.	
match_max	**Define el número máximo de veces en los que este objeto puede ser relacionado con otro. Se utiliza este atributo en las preguntas de tipo relación.**	**opcional**

5.5.15 EL ELEMENTO <*flow_label*>

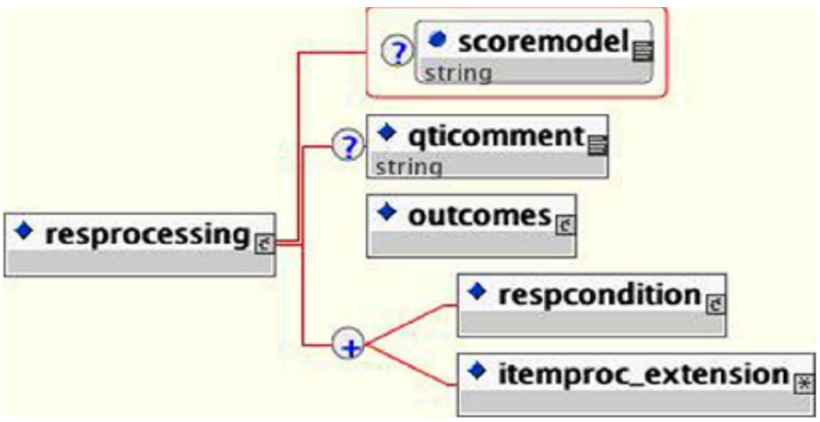


Descripción: **Este elemento se utiliza para establecer un párrafo o salto de línea para la representación del elemento <response_label>.**

Atributos:

Nombre	Descripción	Uso
class	**El tipo de párrafo o bloque que se quiere aplicar al contenido.**	**opcional**

5.5.16 EL ELEMENTO <*resprocessing*>

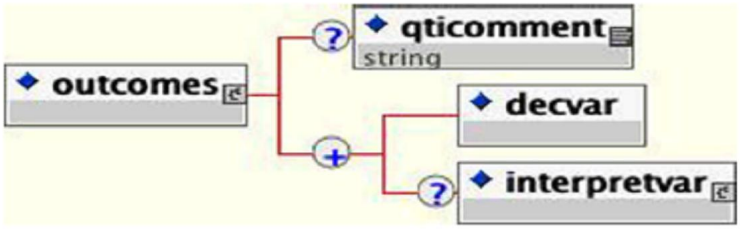


Descripción: **Este tipo contiene todas las instrucciones necesarias para procesar las respuestas. Incluye variables para guardar la puntuación de la respuesta recibida del usuario.**

Atributos:

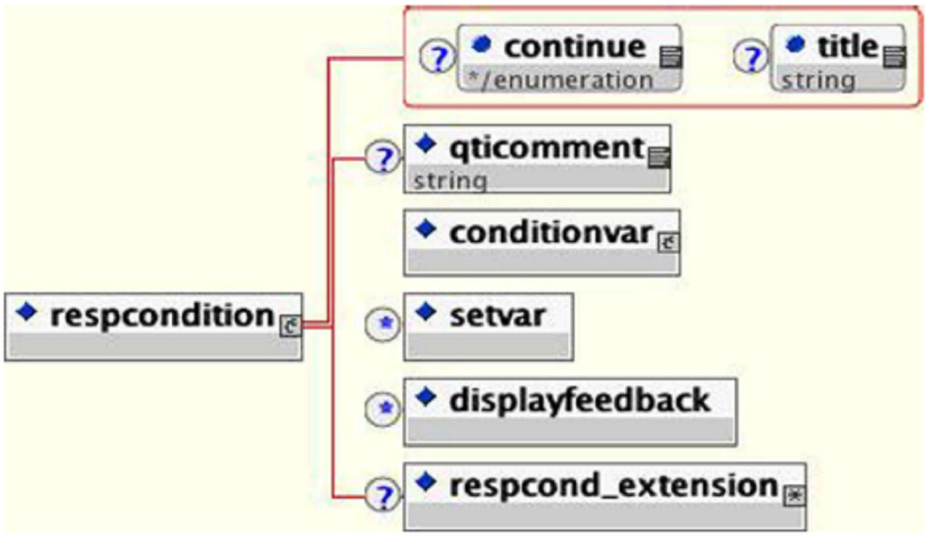
Nombre	Descripción	Uso
scoremodel	**Indica el algoritmo de puntuación que se utilizara. Por defecto se utiliza el 'SumOfScores', es decir la suma de los puntos obtenidos por el usuario.**	**opcional**

5.5.17 EL ELEMENTO <*outcomes*>



Descripción: **este elemento sirve para definir variables de puntuación.**

5.5.18 EL ELEMENTO <*respcondition*>

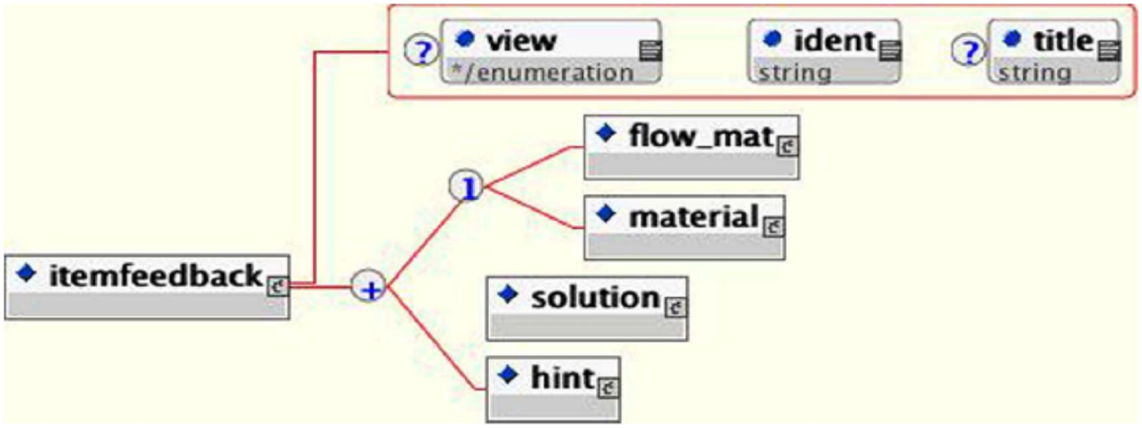


Descripción: **Este elemento se usa para evaluar la respuesta del usuario, para asignar un valor a la variable de puntuación y para asociar un refuerzo a respuestas.**

Atributos:

Nombre	Descripción	Uso
continue	**Solo puede tomar los valores Yes o No. Por defecto toma el valor No. Es usado para controlar el orden del procesado de los elementos resprocessing. Si el valor de este atributo es No y la respuesta dada por el usuario esta representada en el elemento conditionvar entonces se acaba el procesado de los elementos respcondition. Este atributo no se tiene en cuenta en algunos editores de preguntas.**	**opcional**
title	**Titulo del elemento respcondition. Puede servir como descripción de su contenido, un ejemplo podría ser "respuesta correcta"**	**opcional**

5.5.19 EL ELEMENTO <*itemfeedback*>



Descripción: **Este elemento contiene elementos material que serán mostrados al usuario tras dar su respuesta, cuando para esa respuesta exista un elemento displayfeedback que mediante su atributo linkrefid haga referencia al elemento itemfeedback.**

Atributos:

Nombre	Descripción	Uso
view	La vista determina a que usuarios será mostrada la información dependiendo de su categoría. Los valores posibles son: All, Administrador, AdminAuthority, Asesor, Autor, Candidate, InvigilatorProctor, Psychometrician, Scorer, Tutor. Por defecto toma el valor All.	opcional
ident	Identificador único para itemfeedback dentro de un item. Es del tipo cadena y es usado para saber que elemento mostrar como consecuencia del procesado de la respuesta del usuario.	obligatorio
title	Titulo del elemento	opcional

5.5.20 EL ELEMENTO <*qticomment*>

Descripción: **Este elemento se usa para añadir comentarios al fichero QTI.**

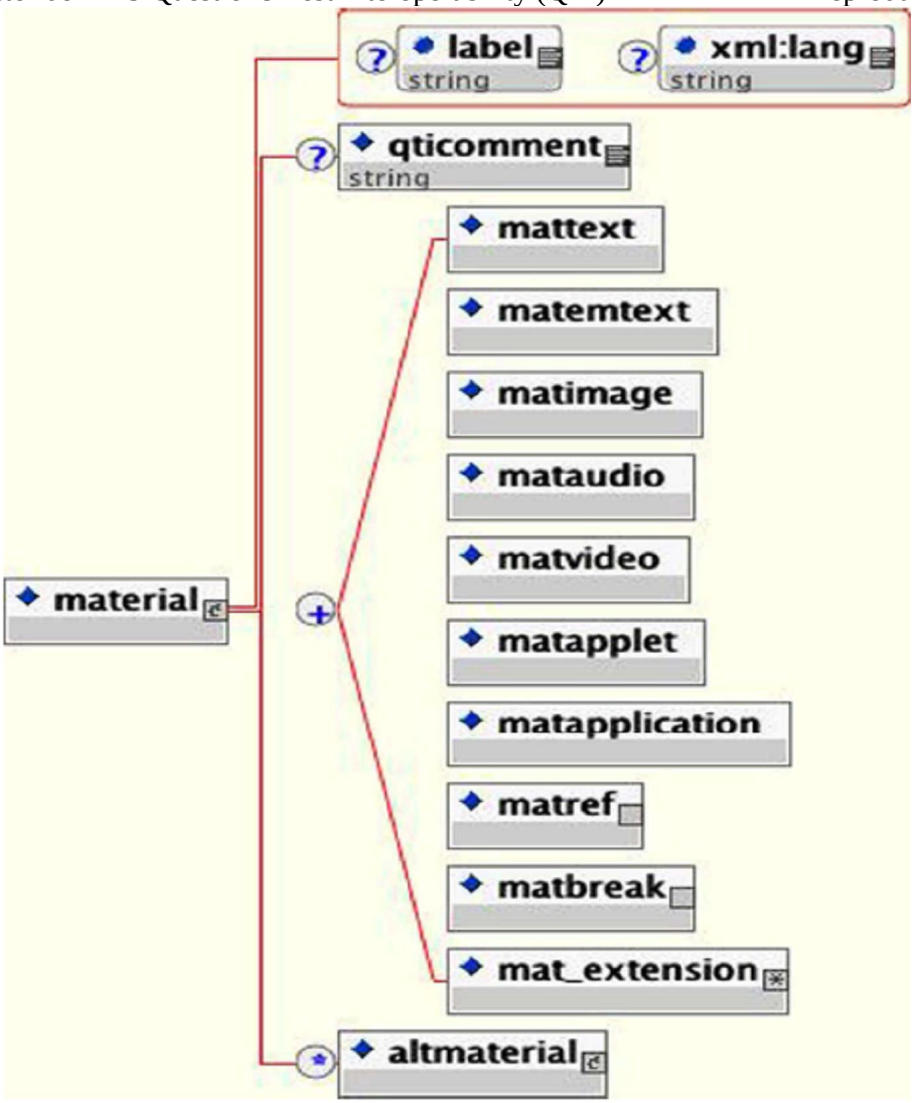
Atributos:

Nombre	Descripción	Uso
xml_lang	Indica el lenguaje del texto del comentario.	opcional

5.5.21 EL ELEMENTO <*duration*>

Descripción: **Se utiliza este elemento cuando se quiera establecer un tiempo límite al usuario para responder a la pregunta.**

5.5.22 EL ELEMENTO <*material*>



Descripción: **Este elemento es el contenedor de cualquier contenido que se quiera presentar al usuario. Este contenido puede ser texto, imágenes, audio, video, aplicación y applet.**

Atributos:

Nombre	Descripción	Uso
label	**Etiqueta que puede ser usada por las aplicaciones de creación de tests para identificar rasgos importantes.**	**opcional**
xml_lang	**Indica el lenguaje del texto.**	**opcional**

5.5.23 EL ELEMENTO <*mattext*>

Descripción: **este elemento se utiliza para almacenar contenido de tipo text en el elemento material.**

Atributos:

Nombre	Descripción	Uso
texttype	**Indica el tipo de texto que contiene. Por defecto toma el valor text/plain, los posibles valores están definidos en el RFC1521, uno de ellos muy usado es text/html.**	**opcional**
charset	**Es el conjunto de caracteres que es usado para representar la cadena de texto.**	**opcional**
label	**Etiqueta que es usada para identificar singularmente este texto. Esta etiqueta sirve para referenciar el contenido texto desde un elemento matref.**	**opcional**
entityref	**Un mecanismo alternativo para identificar una referencia externa que contiene el texto a mostrar.**	**opcional**
height	**Identifica la altura de la zona de texto. Limitación máxima de 32digitos.**	**opcional**
width	**Identifica la anchura de la zona de texto. Limitación máxima de 32digitos.**	**opcional**
x0	**La abscisa del punto superior izquierdo de la zona de texto.**	**opcional**
y0	**La coordenada del punto superior izquierdo de la zona de texto.**	**opcional**
xml:lang	**Determina el lenguaje del texto.**	**opcional**
xml:space	**Determina si hay que preservar los espacios en blanco.**	**opcional**

5.5.24 EL ELEMENTO <*matemtext*>

Descripción: **Este elemento se usa para almacenar contenido de tipo texto en el elemento material, con la peculiaridad de que el texto será enfatizado al ser mostrado al usuario. La forma en la que el texto se mostrara dependerá de la aplicación o herramienta que se use para hacer el test. Se suele utilizar para marcar o denotar alguna palabra o frase que es importante.**

Atributos:

Nombre	Descripción	Uso
texttype	**Indica el tipo de texto que contiene. Por defecto toma el valor text/plain, los posibles valores están definidos en el RFC1521, uno de ellos muy usado es text/html.**	**opcional**
charset	**Es el conjunto de caracteres que es usado para representar la cadena de texto.**	**opcional**
label	**Etiqueta que es usada para identificar singularmente este texto. Esta etiqueta sirve para referenciar el contenido texto desde un elemento matref.**	**opcional**
entityref	**Un mecanismo alternativo para identificar una referencia externa que contiene el texto a mostrar.**	**opcional**

height	**Identifica la altura de la zona de texto. Limitación máxima de 32digitos.**	**opcional**
width	**Identifica la anchura de la zona de texto. Limitación máxima de 32digitos.**	**opcional**
x0	**La abcisa del punto superior izquierdo de la zona de texto.**	**opcional**
y0	**La coordenada del punto superior izquierdo de la zona de texto.**	**opcional**
xml:lang	**Determina el lenguaje del texto.**	**opcional**
xml:space	**Determina si hay que preservar los espacios en blanco.**	**opcional**

5.5.25 EL ELEMENTO <*matimage*>

Descripción: **Este elemento se usa para almacenar contenido de tipo imagen en el elemento material. También tiene información sobre las características de la imagen como es el atributo imagtype.**

Atributos:

Nombre	Descripción	Uso
imagtype	**Indica el tipo de la imagen que contiene. Por defecto toma el valor image/jpeg, los posibles valores están definidos en el RFC1521.**	**opcional**
label	**Etiqueta que es usada para identificar singularmente este elemento imagen. Esta etiqueta sirve para referenciar el contenido texto desde un elemento matref.**	**opcional**
uri	**Indica una URI en la cual se encuentra un contenido de tipo imagen para mostrar. Generalmente es un nombre de fichero o una URL.**	**opcional**
entityref	**Un mecanismo alternativo para identificar una referencia externa que contiene la imagen a mostrar.**	**opcional**
height	**Identifica la altura de la imagen.**	**opcional**
width	**Identifica la anchura de la imagen.**	**opcional**
x0	**Es la abscisa del punto superior izquierdo de la posición de la imagen.**	**opcional**
y0	**La coordenada del punto superior izquierdo de la posición de la imagen.**	**opcional**
embedded	**Define el tipo de codificación de la imagen si ésta está embebida dentro del mismo documento XML. Por defecto toma el valor base64.**	**opcional**

5.5.26 EL ELEMENTO <*mataudio*>

Descripción: **Este elemento se usa para almacenar contenido de tipo audio en el elemento material. También tiene información sobre las características del audio como es el atributo audiotype.**

Atributos:

Nombre	Descripción	Uso
audiotype	**Indica el tipo de audio que contiene. Por defecto toma el valor audio/base. Los posibles valores están definidos en el RFC1521.**	**opcional**
label	**Etiqueta que es usada para identificar singularmente este audio.**	**opcional**
uri	**Indica una URI en la cual se encuentra un contenido de tipo audio para mostrar. Generalmente es un nombre de fichero o una URL.**	**opcional**
entityref	**Un mecanismo alternativo para identificar una referencia externa que contiene el audio a mostrar.**	**opcional**
embedded	**Define el tipo de codificación del audio si éste está embebido dentro del mismo documento XML. Por defecto toma el valor base64.**	**opcional**

5.5.27 EL ELEMENTO <*matvideo*>

Descripción: **Este elemento se usa para almacenar contenido de tipo video en el elemento material. También tiene información sobre las características del video como es el atributo videotype.**

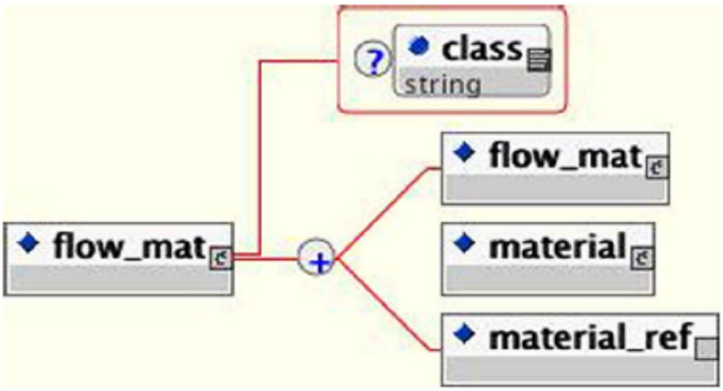
Atributos:

Nombre	Descripción	Uso
videotype	**Indica el tipo del video que contiene. Por defecto toma el valor video/mpeg. Los posibles valores están definidos en el RFC1521.**	**opcional**
label	**Etiqueta que es usada para identificar singularmente este video.**	**opcional**
uri	**Indica una URI en la cual se encuentra un contenido de tipo video para mostrar. Generalmente es un nombre de fichero o una URL.**	**opcional**
entityref	**Un mecanismo alternativo para identificar una referencia externa que contiene el video a mostrar.**	**opcional**
height	**Identifica la altura del video.**	**opcional**
width	**Identifica la anchura del video.**	**opcional**
x0	**Es la abscisa del punto superior izquierdo de la posición del video.**	**opcional**
y0	**La coordenada del punto superior izquierdo de la posición del video.**	**opcional**
embedded	**Define el tipo de codificación del video si éste está embebido dentro del mismo documento XML. Por**	**opcional**

	defecto toma el valor base64.	

5.5.28 EL ELEMENTO <*flow_mat*>

Descripción: Este elemento sirve para presentar información agrupada en flujos de datos. Este flujo suele ser párrafos o bloques de datos.



Atributos:

Nombre	Descripción	Uso
class	**Indica el tipo de flujo o bloque que se utilizará para agrupar el contenido o material a presentar. *Bloque* es el valor por defecto. También puede tomar el valor *List* que indicara que el contenido se agrupara en forma de lista vertical.**	**opcional**

5.5.29 EL ELEMENTO <*decvar*>

Descripción: Este elemento se utiliza para declarar las variables de puntuación. Se puede indicar en sus atributos el nombre, el tipo y el valor por defecto además de otros atributos.

Atributos:

Nombre	Descripción	Uso
varname	**Indica el nombre de la variable que se va a declarar. Por defecto este nombre es *SCORE*.**	**opcional**
vartype	**Es el tipo de la variable. Este atributo puede tomar un valor de la siguiente lista de valores (String, Decimal, Scientific, Boolean, Integer, Enumerated). El valor por defecto es Integer.**	**obligatorio**
defaultval	**Valor con el que se inicializa la variable. Tiene como rango máximo 16 dígitos.**	**opcional**
cutvalue	**Este atributo sirve para indicar que si la variable iguala o supera este valor, entonces el objetivo esta superado.**	**opcional**
minvalue	**Indica el valor mínimo permitido a la variable de puntuación numérica.**	**opcional**

maxvalue	**Es el valor máximo permitido a la variable numérica.**	**opcional**
members	**Indica que la variable es un enumerado, es decir que solo puede tomar un valor de una lista predeterminada de valores.**	**opcional**

5.5.30 EL ELEMENTO <*setvar*>

Descripción: Este elemento es el responsable de cambiar y actualizar la variable de puntuación debido a la evaluación de la pregunta.

Atributos:

Nombre	Descripción	Uso
varname	**Indica el nombre de la variable a la que hay que cambiar el valor como consecuencia de la evaluación de la pregunta. Por defecto es SCORE.**	**opcional**
action	**Indica el tipo de la operación que hay que realizar sobre la variable para actualizarla. Los valores posibles son (Set, Add, Subtract, Multiply, Divide, por defecto =Set)**	**opcional**

5.5.31 EL ELEMENTO <*displayfeedback*>

Descripción: Este elemento se encarga de asociar un elemento itemfeedback a las respuestas que representa un elemento conditionvar. Contiene elementos <material> que serán mostrados al usuario tras dar su respuesta, cuando para esa respuesta exista un elemento displayfeedback que mediante su atributo linkrefid haga referencia al elemento itemfeedback.

Atributos:

Nombre	Descripción	Uso
feedbacktype	**Tipo del elemento itemfeedback que va a ser disparado. Puede ser Response, Solution y Hint.**	**opcional**
linkrefid	**Identificador del elemento itemfeedback asociado.**	**obligatorio**

5.5.32 EL ELEMENTO <*conditionvar*>

Descripción: Este elemento es el que realiza la evaluación sobre la respuesta del usuario. Gracias a la variedad de elementos que puede contener este elemento, se puede realizar variadas evaluaciones sobre las respuestas del usuario.

5.5.33 EL ELEMENTO <*varequal*>

Descripción: Este elemento es de equivalencia. Evalúa si la respuesta del usuario es igual a la que esta dentro de este elemento.

Atributos:

Nombre	Descripción	Uso
Respident	**Este atributo apunta a un identificador de una pregunta.**	**obligatorio**
Index	**Se utiliza este atributo en las preguntas de tipo ordenar donde el orden de las respuestas del usuario es relevante.**	**opcional**
Case	**Indica si hay que respetar o ignorar la mayúscula y la minúscula de las dos cadenas de caracteres al compararlas.**	**opcional**

5.5.34 EL ELEMENTO <*varlt*>

Descripción: Este elemento es el operador lógico 'inferior a' (<). Se suele utilizar en las preguntas tipo libre con valores numéricos.

Atributos:

Nombre	Descripción	Uso
Respident	**Este atributo apunta a un identificador de una pregunta.**	**obligatorio**
Index	**Se utiliza este atributo en las preguntas de tipo ordenar donde el orden de las respuestas del usuario es relevante.**	**opcional**

5.5.35 EL ELEMENTO <*varlte*>

Descripción: Este elemento es el operador lógico 'inferior o igual a' (<=). Se suele utilizar en las preguntas tipo libre con valores numéricos.

Atributos:

Nombre	Descripción	Uso
Respident	**Este atributo apunta a un identificador de una pregunta.**	**obligatorio**
Index	**Se utiliza este atributo en las preguntas de tipo ordenar donde el orden de las respuestas del usuario es relevante.**	**opcional**

5.5.36 EL ELEMENTO <*vargt*>

Descripción: Este elemento es el operador lógico 'superior a' (>). Se suele utilizar en las preguntas tipo libre con valores numéricos.

Atributos:

Nombre	Descripción	Uso
Respident	**Este atributo apunta a un identificador de una pregunta.**	**obligatorio**

Index	**Se utiliza este atributo en las preguntas de tipo ordenar donde el orden de las respuestas del usuario es relevante.**	**opcional**

5.5.37 EL ELEMENTO <*vargte*>

Descripción: **Este elemento es el operador lógico 'superior a' (>=). Se suele utilizar en las preguntas tipo libre con valores numéricos.**

Atributos:

Nombre	Descripción	Uso
Respident	**Este atributo apunta a un identificador de una pregunta.**	**obligatorio**
Index	**Se utiliza este atributo en las preguntas de tipo ordenar donde el orden de las respuestas del usuario es relevante.**	**opcional**

5.5.38 EL ELEMENTO <*varsubstring*>

Descripción: **Este elemento comprueba si la respuesta del usuario es una subcadena de una respuesta completa. Se usa sobre todo en las preguntas de tipo libre.**

Atributos:

Nombre	Descripción	Uso
Respident	**Este atributo apunta a un identificador de una pregunta.**	**obligatorio**
Index	**Se utiliza este atributo en las preguntas de tipo ordenar donde el orden de las respuestas del usuario es relevante.**	**opcional**
case	**Indica si hay que respetar o ignorar la mayúscula y la minúscula al comparar dos cadenas de caracteres.**	**opcional**

5.5.39 EL ELEMENTO <*not*>

Descripción: **Este elemento es el operador lógico 'not'. Hace el complementario de una expresión lógica.**

5.5.40 EL ELEMENTO <*and*>

Descripción: **Este elemento es el operador lógico 'and'.**

5.5.41 EL ELEMENTO <*or*>

Descripción: **Este elemento es el operador lógico 'or'.**

5.5.42 EL ELEMENTO <*unanswered*>

Descripción: **Este elemento representa el caso en el que no se responda la respuesta determinada por el contenido del atributo respident.**

Atributos:

Nombre	Descripción	Uso
Respident	**Este atributo apunta a un identificador de una pregunta.**	**obligatorio**

6. REPRODUCTOR QTI

El reproductor QTI es una aplicación Web construida siguiendo el patrón de arquitectura software Modelo Vista Controlador. Este patrón de arquitectura es uno de los más utilizados a la hora de realizar aplicaciones que posean una interfaz de usuario más o menos compleja, como puede ser una interfaz Web. Esta basado principalmente en separar la lógica de los datos de la interfaz grafica que permita al usuario acceder a ellos, colocando entre ambos elementos una capa intermedia que interactúe con ambos, el controlador.

En el reproductor QTI tendremos los tres elementos de la arquitectura bien diferenciados:

-modelo: conjunto de clases Java que actúan sobre el fichero QTI y sacan toda la información necesaria para la presentación de la pregunta y la evaluación de la misma.

-vista: la interfaz Web diseñada para poder representar la pregunta al usuario y facilitar la interacción con ella.

-controlador: los servlets Java encargados de acceder en el servidor a los ficheros QTI y de controlar el flujo de datos entre el servidor y el usuario.

La aplicación ira montada sobre un servidor Apache Tomcat que tendrá acceso a las clases del modelo.

6.1 MODELO

En este apartado vamos a explicar el funcionamiento interno del reproductor QTI. La siguiente figura muestra de forma general la lógica de la implementación:

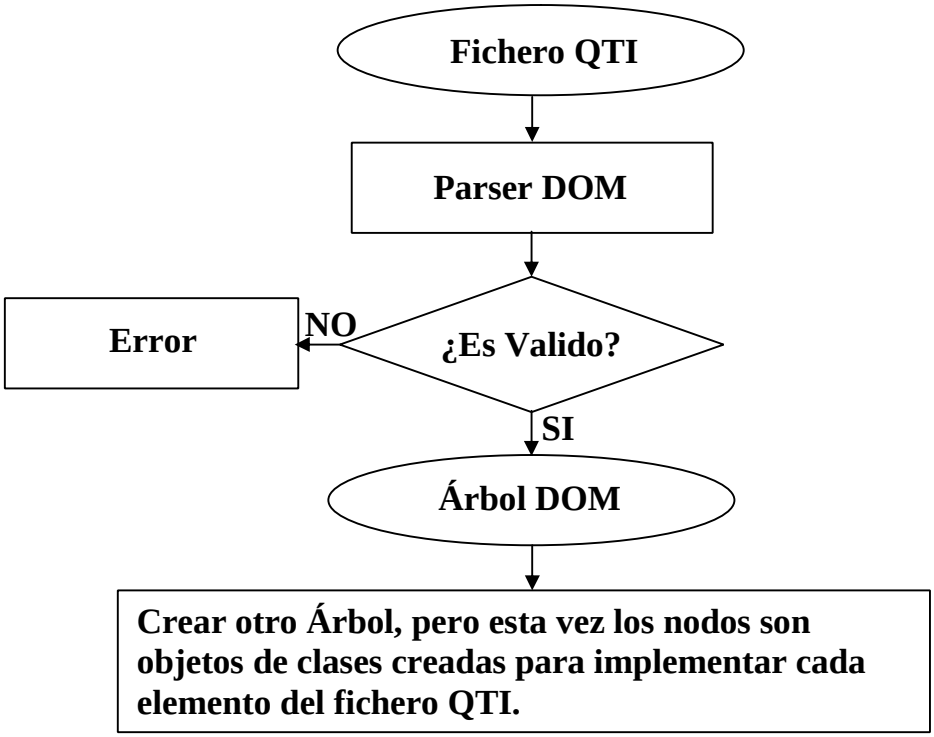


Figura 6.1.1

6.1.1 Analizar el fichero QTI.

Como ya se comento en el capitulo de la tecnologías utilizadas, se ha optado por DOM frente a SAX, ya que este API permite tener en memoria la estructura del documento para recorrerla y consultarla según sea necesario.

Un árbol DOM esta compuesto de nodos y esos nodos pueden ser elementos, atributos, texto, comentarios, etc. que se obtienen al procesar un fichero XML con un parser. Hay que mencionar que en Java el concepto de nodo es distinto al de elemento, atributo, etc. de hecho existe una interfaz nodo de la que son subinterfaces los nodos de texto, elementos, comentarios, etc.

Vamos a comentar la clase *QtiDom* que se encarga de analizar el fichero QTI.

El siguiente código muestra el constructor de la clase *QtiDom*:

```
public QtiDom(StringReader sr) throws IOException, SAXException, ParserConfigurationException
{
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        factory.setNamespaceAware(true);
        factory.setValidating(true);

        InputSource flujo= new InputSource(sr);
        DocumentBuilder builder = factory.newDocumentBuilder();

        builder.setErrorHandler(new ManejadorError());
        doc = builder.parse(flujo);
}
```

- **El constructor tiene como parámetro un objeto de tipo *StringReader* que es un flujo de caracteres cuyo origen es una cadena de caracteres (contenido del fichero QTI).**
- **La clase *DocumentBuilderFactory* se utiliza para obtener objetos *DocumentBuilder*, que producen árboles de objetos DOM a partir de documentos XML. Puesto que tiene constructor protegido, las aplicaciones utilizan el método estático *newInstance()* de esta clase para crear objetos factoría. Se han establecido algunos parámetros de configuración de los objetos *DocumentBuilder* que crea esta factoría. El método *setNamespaceAware(boolean)* indica si la factoría creara instancias de analizadores que tienen en cuenta los espacios de nombres. El método *setValidating()* indica si los analizadores devueltos deben validar documentos al analizarlos.**
- **La clase *DocumentBuilder* acepta un documento XML a partir del cual crea un objeto *org.w3c.dom.Document*.**
- **El método *setErrorHandler()* permite que la aplicación especifique un objeto SAX *errorHandler* para que sea utilizado por la implementación DOM subyacente para reportar todos los errores y advertencias que surjan en el análisis.**
- **El método *parse* analiza el contenido de una fuente de entrada de datos dada como documento XML y devuelven el objeto *org.w3c.dom.Document* que contiene el árbol DOM resultante.**

Ahora que ya tenemos el árbol DOM en memoria, nos interesaría buscar algunos elementos que son interesantes para llevar acabo la reproducción del fichero QTI. Por este motivo se ha implementado algunos métodos en la misma clase para tener acceso a esos relevantes elementos.

El siguiente código muestra una forma de buscar el elemento *item* dentro del árbol DOM:

```
private void getNodoItem(Node documento) {
        for (Node nodo= documento.getFirstChild(); nodo!=null; nodo=  nodo.getNextSibling()) {
                if (nodo.getNodeName().equals(ClaseUtil.ITEM)) {
                        item= nodo;
                        break;
                }
                else {
                        getNodoItem(nodo);
                }
        }
}
```

- **El método *getFirstChild()* devuelve el primer hijo del nodo.**
- **El método *getNextSibling()* devuelve el nodo inmediatamente siguiente a este nodo.**

6.1.2 Inspeccionar la validez del fichero QTI.

Si una aplicación necesita ejecutar manipulación de errores personalizadas, debe implementar la interfaz *ErrorHandler* y después registrar una instancia utilizando el método *setErrorHandler()*. El analizador SAX informará entonces de todos los errores y advertencias a través de esta interfaz.

La clase *ManejadorError* implementa esa interfaz, veamos su código:

```
public class ManejadorError implements ErrorHandler
{
        public void warning(SAXParseException e) throws SAXException
        {
                throw e;
        }

        public void error(SAXParseException e) throws SAXParseException
        {
                throw e;
        }

        public void fatalError(SAXParseException e) throws SAXParseException
        {
                throw e;
        }
}
```

- **El método *warning()* recibe las notificaciones de advertencias. Los analizadores utilizaran este método para reportar condiciones que no son errores ni errores fatales tal como los define la Recomendación XML 1.0. El comportamiento predeterminado es no llevar a cabo ninguna acción. El analizador continuará proporcionando sucesos normales de análisis después**

de invocar a este método, de forma que seguirá siendo posible para la aplicación procesar el documento hasta el final.

- **El método *error()* recibe las notificaciones de errores que admiten recuperación. Esto corresponde a la definición de "error" de la Recomendación XML 1.0 del W3C. Por ejemplo, un analizador de validación utilizaría esta llamada para informar de la violación de una restricción de validez. El comportamiento predeterminado es no llevar a cabo ninguna acción. El analizador continuará proporcionando sucesos normales de análisis después de invocar a este método, de forma que seguirá siendo posible para la aplicación procesar el documento hasta el final.**
- **El método *fatalError()* recibe las notificaciones de errores que no permiten recuperación. Esto corresponde a la definición de "error fatal" de la Recomendación XML 1.0 del W3C. por ejemplo, un analizador utilizaría esta llamada para reportar la violación de una restricción de documento bien formado. La aplicación debe asumir que no se puede usar el documento una vez que el analizador haya invocado a este método y solo debe continuar procesando (si lo hace) para buscar errores adicionales. Los analizadores pueden dejar de proporcionar informes de sucesos una vez que se ha invocado a este método.**

6.1.3 Reproducción del fichero QTI.

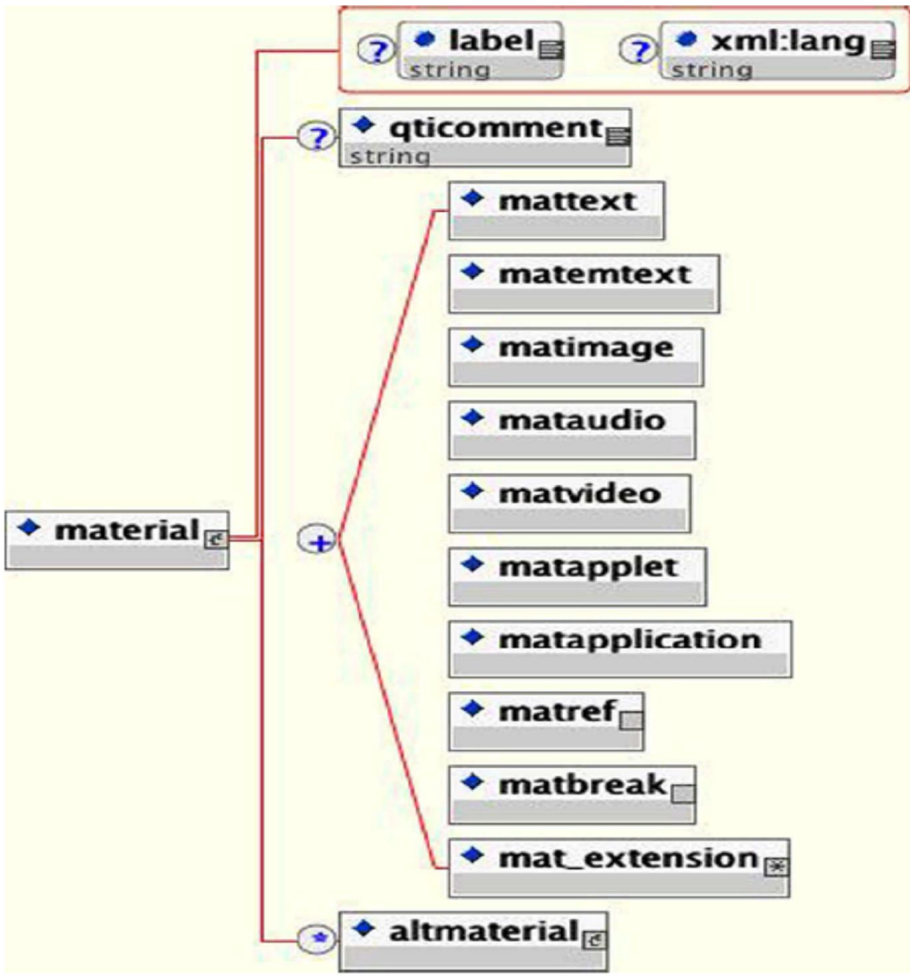
Ahora que ya tenemos en la memoria el árbol DOM, vamos a interpretar cada nodo o elemento según su función en la especificación QTI. Lo que se ha hecho es crear una clase para cada elemento y después volver a crear todo el árbol pero esta vez, los nodos son objetos que ya pueden ofrecer métodos para desarrollar su función.

Todos los elementos tienen un comportamiento común, por este motivo se creo una clase abstracta *Elemento* que define de forma general el comportamiento de todos los elementos. Veamos su código:

```
/* clase abstracta que representa de forma general el comportamiento de cualquier
 * elemento del arbol DOM del fichero QTI
 */

public abstract class Elemento
{

	//presentacion del elemento
	protected StringBuffer presentacion;

	//Elementos hijos que contiene el elemento
	protected Vector hijos= null;

	//Metodo get para devolver el miembro hijos
	protected Vector getHijos()
	{
		return this.hijos;
	}

	/*metodo abstracto que es comun para cualquier elemento
	 *este metodo saca los nodos hijos de un nodo concreto y crea objetos del tipo
	 *Elemento y las pone en el miembro "hijos"
	 */
	abstract protected void tratar(Node nodo);


	/*metodo que es comun para cualquier elemento
	 *asigna los valores a los atributos del elemento
	 */
	abstract protected void asignarAtributos(Node nodo);
}
```

- **El miembro *presentacion* es la cadena de caracteres que contendrá como valor la reproducción o la interpretación del elemento. Algunos elementos devolverán una cadena vacía puesto que no son representables como son los elementos para la evaluación de la respuesta del usuario (operadores lógicos).**
- **El miembro *hijos* es una estructura que contendrá todos los hijos del elemento.**
- **El método *getHijos()* devuelve los hijos del elemento.**
- **El método abstracto *tratar(Node nodo)* es el que trata el elemento, es decir es el encargado de asignar todos los atributos del elemento a los miembros de la clase y devolver los hijos del elemento.**
- **El método abstracto *asignarAtributos(Node nodo)* es el que asigna los atributos del elemento a los miembros de la clase. Este método es llamado dentro del método *tratar(Node nodo)*.**

Vamos a mostrar con la ayuda de un ejemplo la manera de interpretar un elemento del fichero QTI. Escogemos el elemento <*material*>, La siguiente figura muestra la especificación del mismo:



El siguiente código es la clase *Material*:

```
public class Material extends Elemento {

	//atributos
	private String label;
	private String lang;

	//constructor de la clase
	public Material(Node nodo){
		this.hijos= new Vector();
		this.tratar(nodo);
	}

	//definir el metodo asignarAtributos() heredado de la clase abstracta padre Elemento
	protected void asignarAtributos(Node nodo){
		if (nodo!=null && nodo.hasAttributes()) {
			NamedNodeMap atributos= nodo.getAttributes();
			for (int i=0; i< atributos.getLength(); i++) {
				Node atr= atributos.item(i);
				String nombreAtr= atr.getNodeName();
				String valorAtr= atr.getNodeValue();
				if (nombreAtr.equals(ClaseUtil.LABEL)) {
					this.label= valorAtr;
				}
				else if (nombreAtr.equals(ClaseUtil.LANG)) {
					this.lang= valorAtr;
				}
			}
		}
	}

	//definir el metodo tratar() heredado de la clase abstracta padre Elemento
	protected void tratar(Node nodo) {
		this.asignarAtributos(nodo);
		if (nodo!=null && nodo.hasChildNodes()) {
			NodeList hijos= nodo.getChildNodes();
			for (int i=0; i< hijos.getLength(); i++) {
				Node elem= hijos.item(i);
				int tipo= this.getTipo(elem.getNodeName());
				switch (tipo) {
					case MATTEXT:
						MatText texto= new MatText(elem);
						this.hijos.add(texto);
						break;
					case MATEMTEXT:
						MatEmText textoEm= new MatEmText(elem);
						this.hijos.add(textoEm);
						break;
					case MATIMAGE:
						MatImage imagen= new MatImage(elem);
						this.hijos.add(imagen);
						break;
					case MATAUDIO:
						MatAudio audio= new MatAudio(elem);
						this.hijos.add(audio);
						break;
					case MATVIDEO:
						MatVideo video= new MatVideo(elem);
						this.hijos.add(video);
						break;
```

```
                        }
                    }
                }
            }

            public String toString() {
                this.presentacion= new StringBuffer();
                if (this.hijos!=null && this.hijos.size()>0) {
                    Iterator iter= this.hijos.iterator();
                    while (iter.hasNext()) {
                        Elemento elem= (Elemento) iter.next();
                        if (elem instanceof MatText) {
                            MatText matText= (MatText)elem;
                            this.presentacion.append(matText.toString());
                        }
                        else if (elem instanceof MatEmText) {
                            MatEmText matEmText= (MatEmText)elem;
                            this.presentacion.append(matEmText.toString());
                        }
                        else if (elem instanceof MatImage) {
                            MatImage matImage= (MatImage)elem;
                            presentacion.append(matImage.toString());
                        }
                        else if (elem instanceof MatAudio) {
                            MatAudio matAudio= (MatAudio)elem;
                            this.presentacion.append(matAudio.toString());
                        }
                        else if (elem instanceof MatVideo) {
                            MatVideo matVideo= (MatVideo)elem;
                            this.presentacion.append(matVideo.toString());
                        }
                    }
                }
                return this.presentacion.toString();
            }
```

Vamos a comentar el código de la clase *Material*:

- **La clase *Material* extiende de la clase *Elemento* y por tanto tiene que implementar los métodos abstractos.**
- **Atributos: Puesto que el elemento <*material*> tiene dos atributos *label* y *xml:lang*, se ha incorporado esos atributos a la clase Material (xml:lang se le ha llamado simplemente lang puesto que no se permite xml:lang como nombre de una variable en Java).**
- **Constructor *Material(Node nodo)*: En el constructor de la clase *Material* se ha llamado al método *tratar(Node nodo)* donde *nodo* es un nodo del árbol DOM que contiene al elemento material del fichero XML.**
- **El método *asignarAtributos(Node nodo)*: Aquí donde se asignan los valores de los atributos label y lang.**

- **El método *tratar(Node nodo):* Aquí donde se llama al método *asignarAtributos(Node nodo)* y después se busca todos los hijos del elemento <*material*> para almacenarlos en la estructura *hijos*.**
- **El método *toString()*: Este método devuelve la cadena de caracteres que será la interpretación del elemento en cuestión, en este caso el elemento <*Material*>. Este método va concatenando las interpretaciones de los hijos (<*mattext*>, <*matemtext*>, <*matimage*>, <*matvideo*>, <*mataudio*> etc.).**

De esta manera conseguiremos al final construir un árbol donde los nodos son instancias de clases que implementan cada elemento del fichero QTI. Puesto que el elemento <*presentation*> es el elemento que contiene todas las instrucciones para pintar la pregunta, basta con llamar al método toString() sobre el objeto de la clase *Presetation* para obtener la reproducción de la pregunta.

6.2 Controlador

El controlador es la parte o el componente de la aplicación Web que se encarga de controlar la comunicación entre el cliente y el servidor donde esta alojada la aplicación.

Antes, la forma tradicional para añadir funcionalidad a un servidor Web era a través de scripts CGI, pero los CGI presentaban muchos inconvenientes a nivel de eficiencia y consumo de memoria, esos inconvenientes han influido en el desarrollo de nuevas tecnologías que ofrezcan mejores resultados como es el caso de los servlets de Java.

Los servlets de Java son clases normales Java que se crean cuando se necesitan y se destruyen cuando ya no se van a usar. Los servlets se ejecutan en una caja restringida, mediante un motor especial, que es el encargado de la creación y destrucción de cada uno de los servlets de la aplicación Web, mediante los métodos *init()* y *destroy()*, respectivamente. Este motor de servlets es la implementación de la especificación Servlet y es el responsable de mantener el ciclo de vida del servlet, y se puede usar solo o en combinación con un servidor Web. La implementación oficial de Sun de la especificación Servlet y, por tanto, el motor de servlets oficial, es el contenedor Yakarta-Tomcat, desarrollado por Apache Software Foundation.

La clase *HttpServlet* es una clase que implementa la interfaz *Servlet* incorporando además métodos específicos para servidores Web. Un uso típico de *HttpServlet* es el procesamiento de formularios html.

El método *service()* de la clase *HttpServlet* lanza diferentes peticiones a distintos métodos Java para sistemas de petición diferentes. Reconoce los métodos estándar en formato http/1.1 y no es conveniente sobrecargarlo en subclases, a no ser que se necesiten implementar métodos adicionales. Los sistemas o métodos de petición que reconoce son GET, HEAD, PUT, POST, DELETE, OPTIONS y TRACE; cualquier otro método obtendrá como respuesta un error HTTP de tipo Bad Request.

Los datos de la petición se pasan a todos los métodos como primer argumento de tipo *HttpServletRequest,* que es una subclase de la clase más general *ServletRequest.* Las respuestas que pueden crear los distintos métodos se devuelven en el segundo argumento de tipo *HttpServletResponse,* que es una subclase de *ServletResponse*.

6.3 Vista

Una vez que hemos visto el modelo de nuestra aplicación y el controlador, vamos a ver ahora la interfaz Web desarrollada.

Para el desarrollo de las paginas Web, hemos optado por las paginas JSP, estas ultimas nos permite separar la parte dinámica de la pagina (código Java y etiquetas especiales) de la parte estática (código HTML), es decir nos permite separar la lógica de la programación y la presentación de los contenidos.

La interfaz Web esta dividida en tres partes como muestra la siguiente figura:

<table>
<tr><td>PUNTUACION</td><td>TITULO</td><td>TIEMPO RESTANTE</td></tr>
<tr><td colspan="3">CONTENIDO DE LA PREGUNTA</td></tr>
<tr><td colspan="3">BOTONES PARA EL ENVIO DE LA RESPUESTA</td></tr>
</table>

- **Cabecera: hay tres elementos en la cabecera de la página Web.**
  - **Puntuación: la puntuación del usuario, solo es visible cuando haya enviado el usuario su respuesta y se haya evaluado.**
  - **Titulo: simplemente es el titulo de nuestra aplicación que es "*Reproductor QTI*".**
  - **Tiempo restante: cuando la pregunta es dependiente del tiempo, se muestra al usuario el tiempo que le falta para responder a la pregunta.**
- **Contenido de la pregunta: es la parte principal de la interfaz de nuestra aplicación, en esta zona donde se muestra la reproducción del fichero QTI.**
- **Pie de la página: en esta zona se ha añadido algunos botones para que el usuario puede enviar su respuesta.**

La siguiente figura muestra el aspecto de nuestra aplicación:

Figura 6.1

6.4 Camino hacia la integración al sistema SIETTE

Como ya se ha comentado antes, aunque la integración del reproductor QTI al sistema SIETTE queda fuera del objetivo de este proyecto, hacia falta introducir al reproductor QTI mecanismos para acercar la filosofía de SIETTE a la del QTI y de esta forma facilitar la labor de la integración.

El sistema SIETTE divide los ítems en 6 tipos diferentes como muestra la siguiente figura:

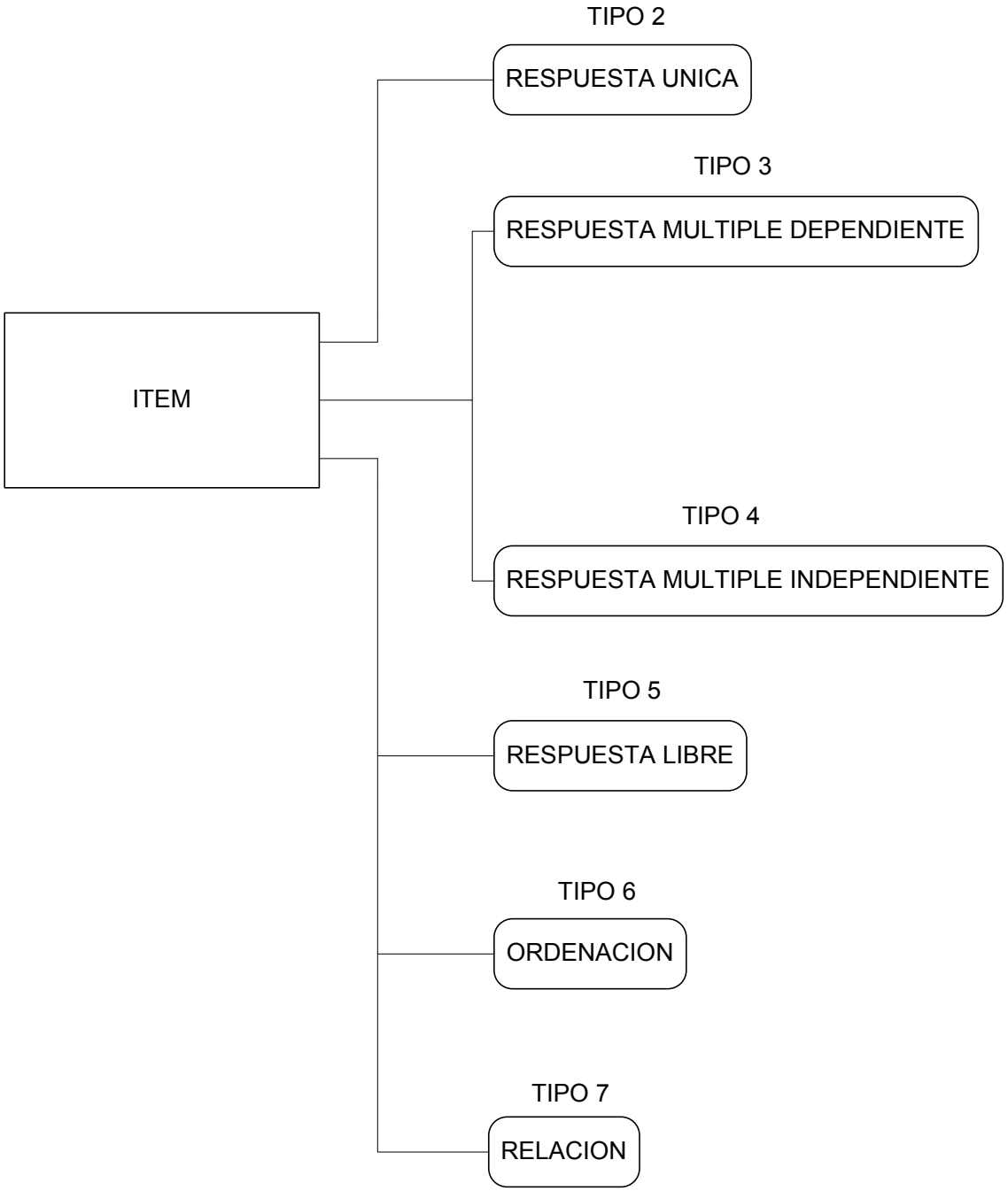


Figura 6.2

A estas alturas, el reproductor QTI debe ser capaz de suministrar a SIETTE un conjunto de información que es el tipo de pregunta, las respuestas posibles y la respuesta correcta.

Para ofrecer este conjunto de información a SIETTE, se ha desarrollado el servlet *QtiParserServlet* que analiza el fichero QTI.

La siguiente figura muestra el funcionamiento de QtiParserServlet:

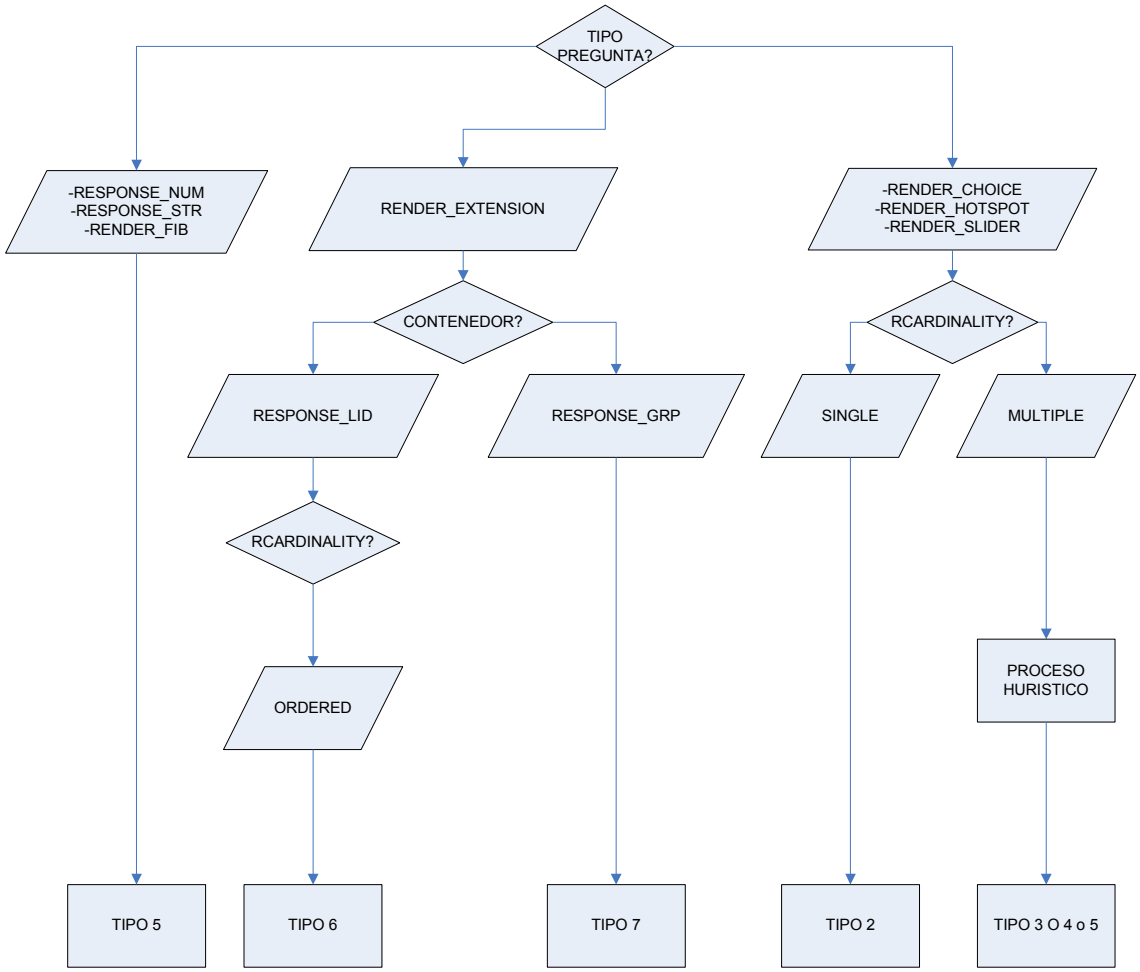


Figura 6.3

En la figura 6.3 se ve claramente como se ha obtenido el tipo de pregunta analizando el fichero QTI.

La siguiente figura muestra el funcionamiento del *proceso huristico* que se muestra en la figura anterior para determinar si es de tipo 3 o 4 o 5.

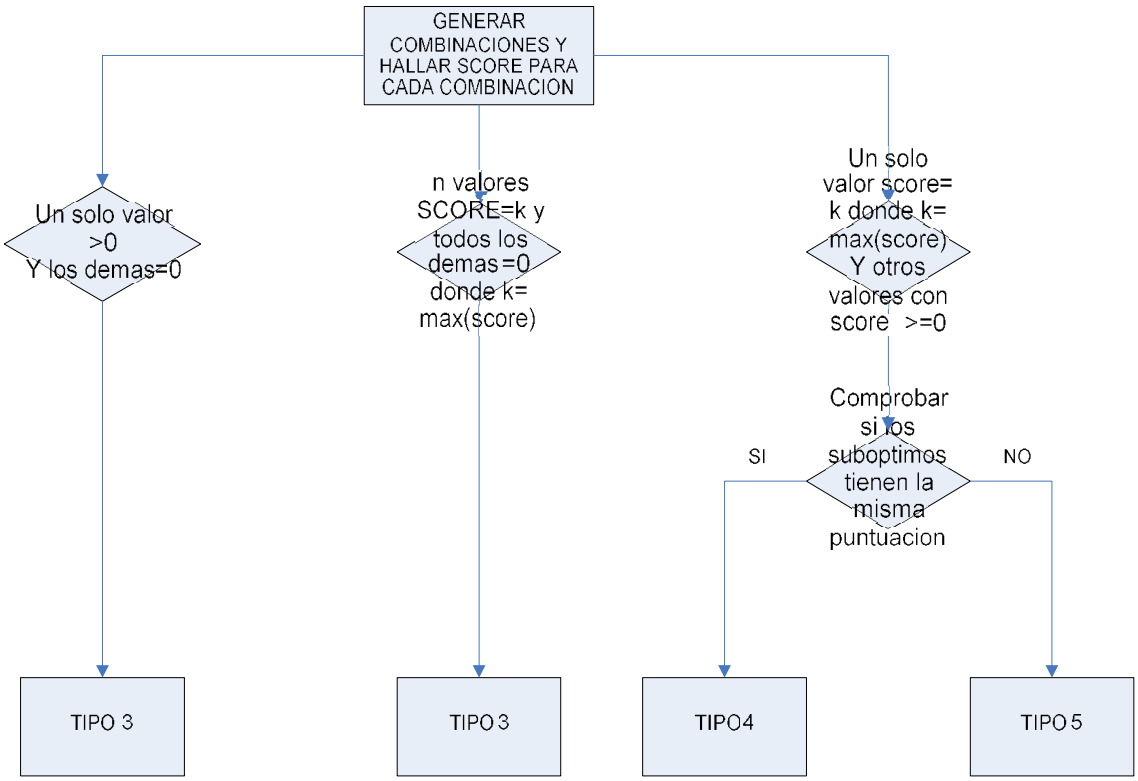

GENERAR COMBINACIONES Y HALLAR SCORE PARA CADA COMBINACION
Un solo valor >0 Y los demas=0
n valores SCORE=k y todos los demas=0 donde k= max(score)
Un solo valor score= k donde k= max(score) Y otros valores con score >=0
Comprobar si los suboptimos tienen la misma puntuacion
SI
NO
TIPO 3
TIPO 3
TIPO4
TIPO 5

Figura 6.4

7. Evaluación de la aplicación

En toda aplicación, parte fundamental de su diseño, es la evaluación de su funcionalidad y de los resultados que se obtienen. No es una etapa concreta, más bien es algo que va con la implementación, pues la aplicación se prueba a medida que se va programando, para solucionar posibles errores.

En este capitulo vamos a probar nuestra aplicación para los diferentes tipos de preguntas que reconoce QTI. Pero antes vamos a describir un poco el funcionamiento de nuestra aplicación.

7.1 funcionamiento de la aplicación

El reproductor QTI es una aplicación Web desarrollada para ser ejecutada en un servidor Apache Tomcat. Por tanto se va a ejecutar en un navegador Web.

Para ejecutarla, basta con poner la dirección http://localhost/qti/servlet/Siette en el navegador.

Podemos resumir el funcionamiento de nuestra aplicación en la siguiente figura:

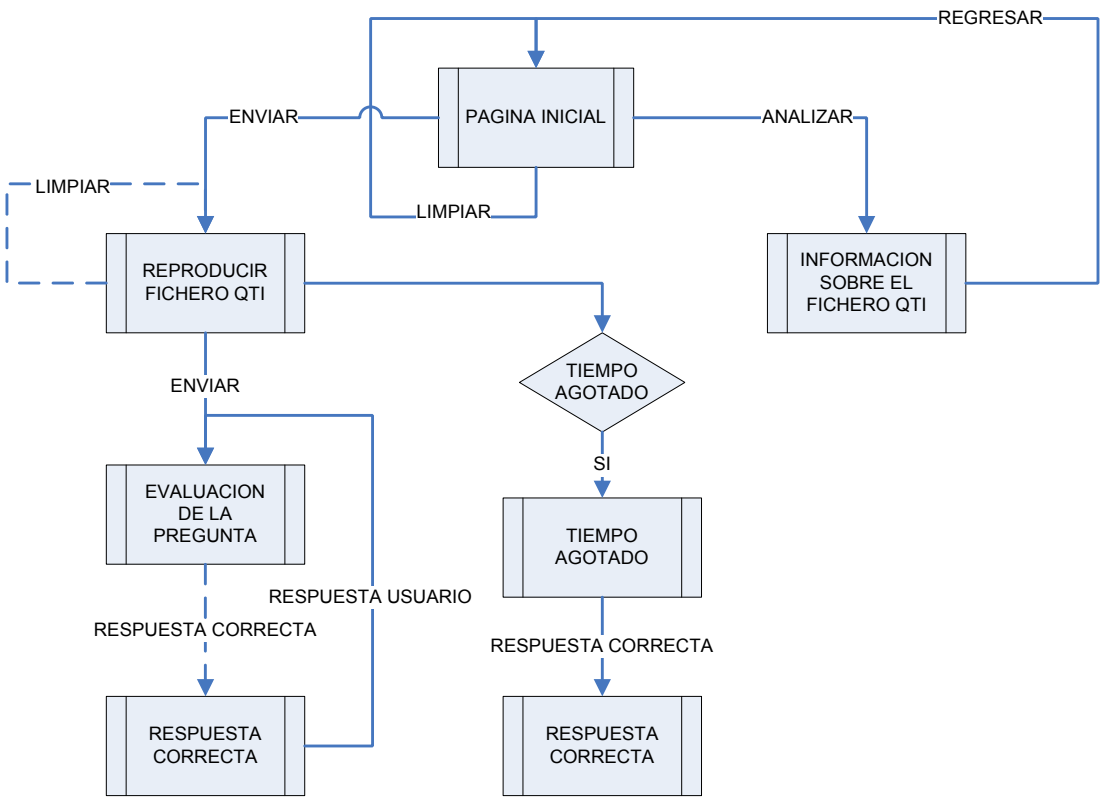


Figura 7.1

7.1.1 Pagina inicial

La siguiente figura muestra la apariencia de la página inicial.

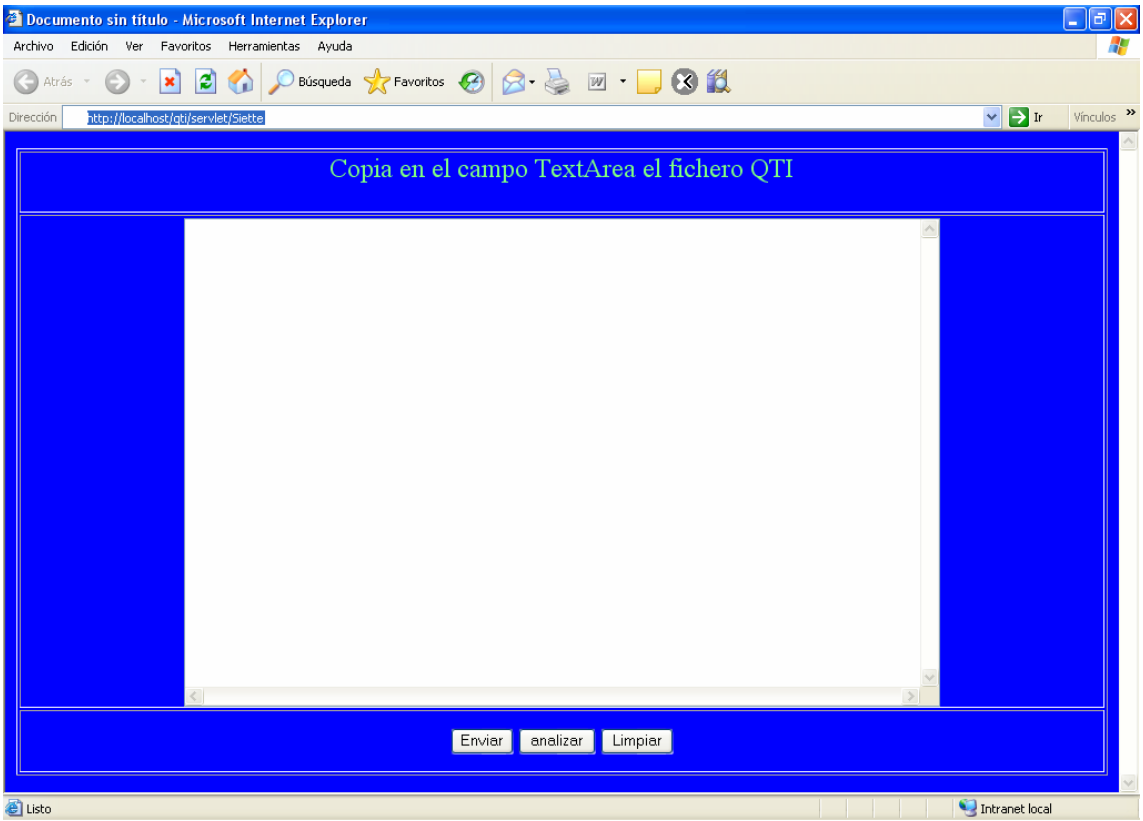


Figura 7.2

Para simular el hecho de que el sistema SIETTE suministra el contenido del fichero QTI al reproductor QTI, se ha desarrollada la pagina anterior (de allí viene el nombre Siette en la dirección Web).

La página inicial consta de los siguientes componentes de interfaz de usuario:

- **Un área de texto en la que se va a pegar el contenido del fichero QTI.**
- **El botón *Enviar* que llama al reproductor QTI proporcionándole el contenido del componente área de texto.**
- **El botón *Analizar* que analiza el contenido del área de texto proporcionando información sobre el tipo de pregunta, respuestas posibles y las respuestas correctas.**
- **El botón *Limpiar* simplemente limpia o borra el contenido del componente área de texto.**

7.1.2 Reproducir fichero QTI

Al pulsar el botón Enviar de la pagina inicial obtenemos el resultado de reproducir la pregunta. La siguiente figura muestra un ejemplo:

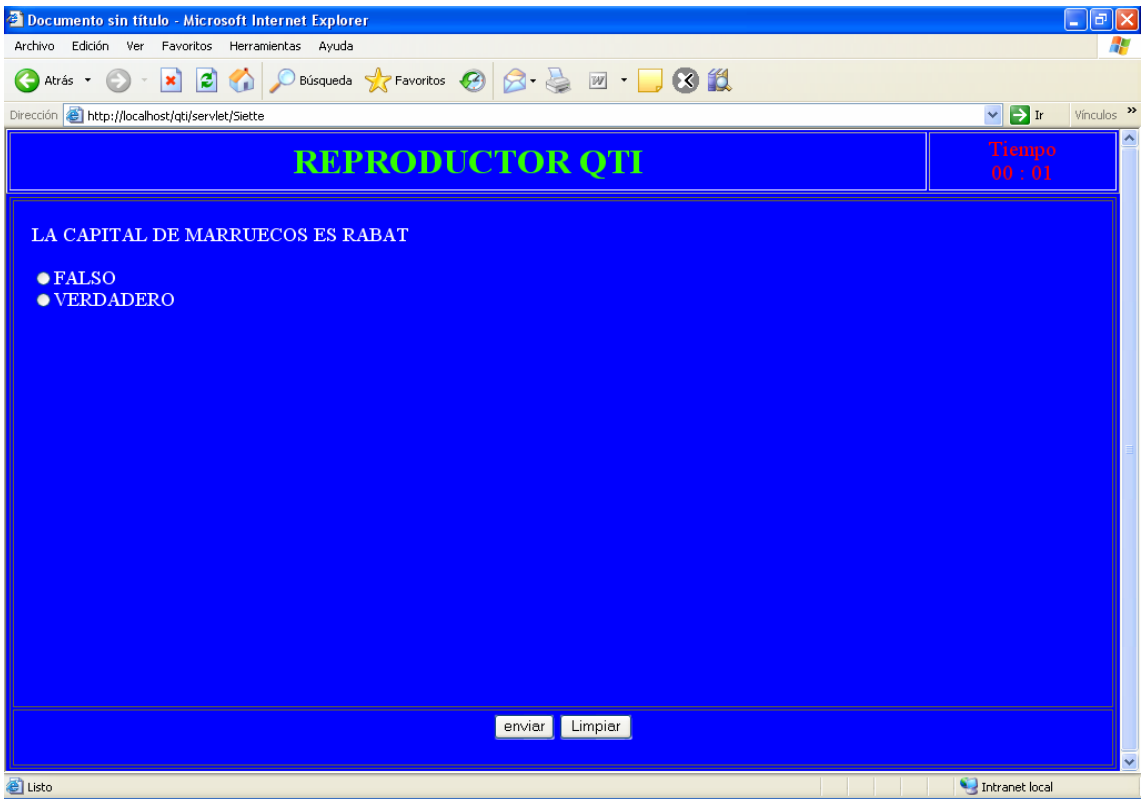


Figura 7.3

Esta página es el resultado de reproducir el fichero QTI. Se han incluido algunos botones para interaccionar con la pregunta. Cuando se trata de una pregunta simple en el que no hace falta un Applet, se ha puesto dos botones enviar y limpiar y cuando se trata de applet se ha puesto solo el botón enviar (de allí la línea cortada en la figura 7.1).

- **El botón enviar que sirve para enviar la respuesta del usuario.**
- **El botón limpiar cuando existe sirve para limpiar el formulario.**

La siguiente figura muestra el aspecto de un ejemplo con un Applet y solo el botón enviar:

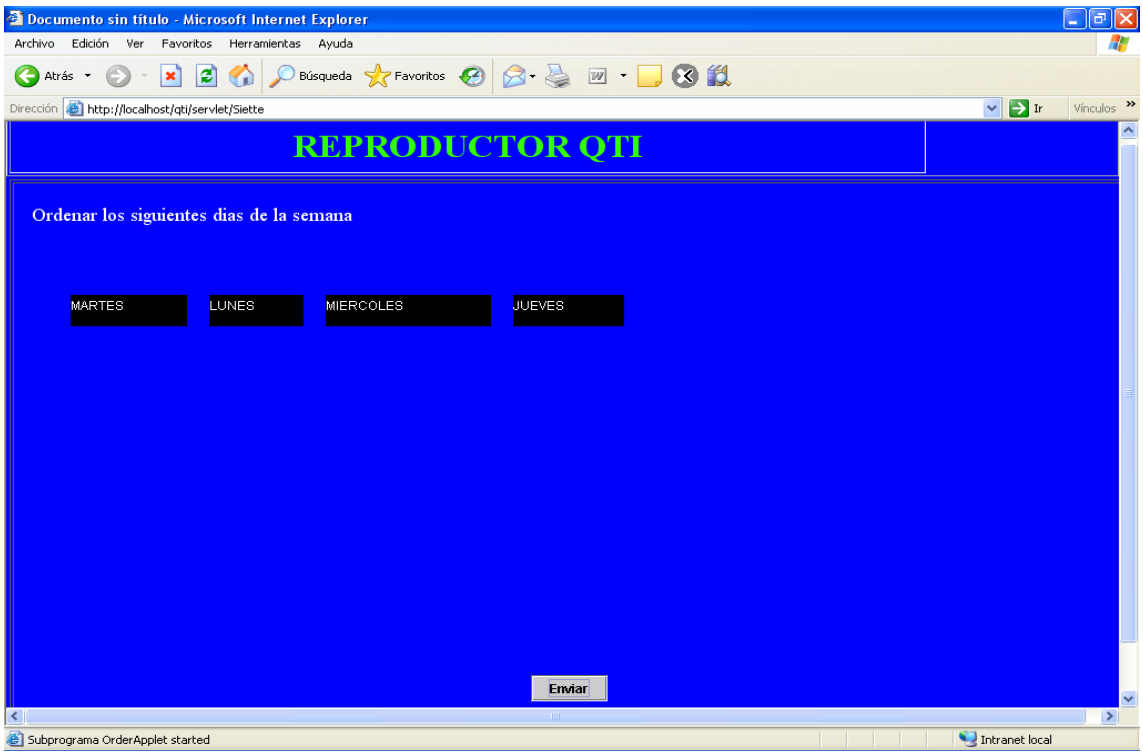


Figura 7.4

7.1.3 Evaluación de la pregunta

Tras enviar la respuesta del usuario, se evalúa la pregunta mostrando la puntuación obtenida y cuando es posible la corrección de la misma en la misma pagina. La siguiente figura ilustra un ejemplo en el que la evaluación y la corrección se muestran en la misma página:

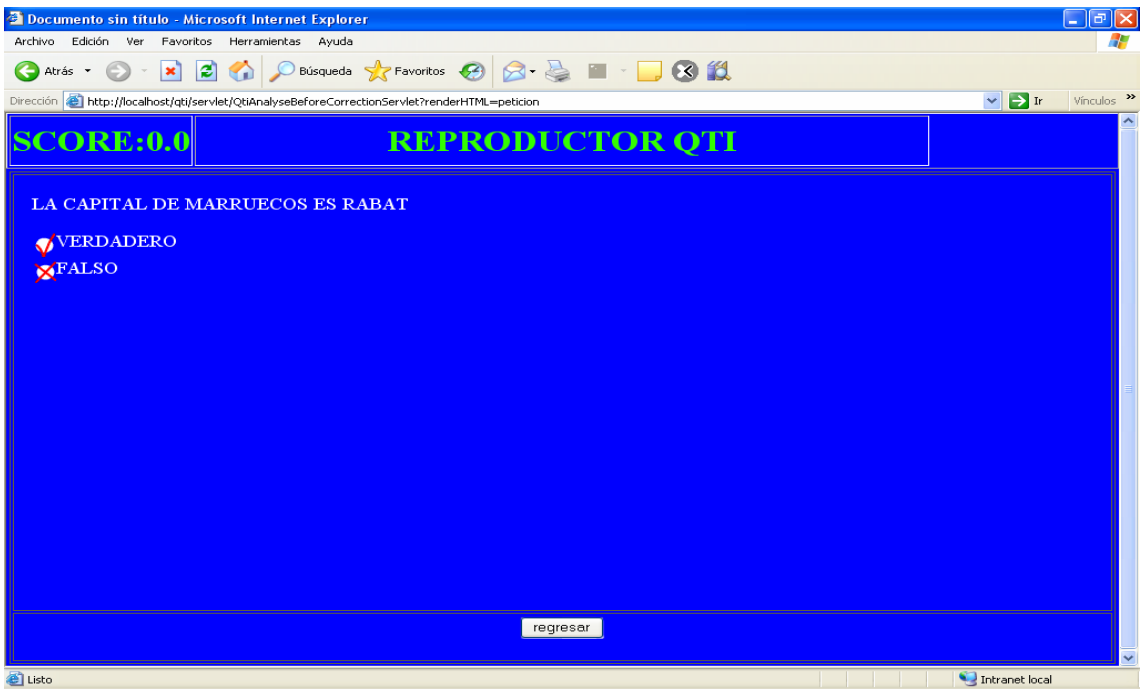


Figura 7.5

- **El botón regresar simplemente sirve para regresar a la pagina principal.**

La siguiente figura ilustra un ejemplo en el que no es recomendable mostrar la evaluación y la corrección de la pregunta en la misma página (no es cómodo de visualizar):

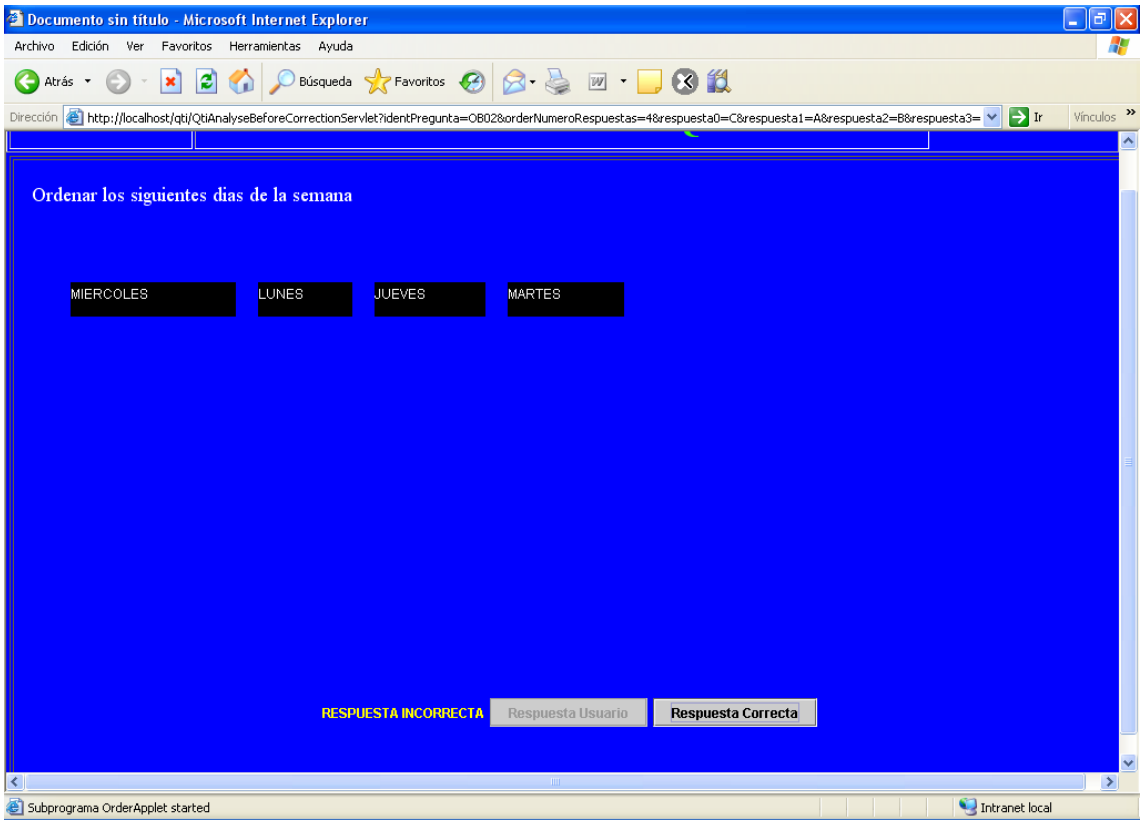


Figura 7.6

- **el botón *Respuesta Usuario* cuando esta activado muestra la respuesta del usuario.**
- **El botón *Respuesta Correcta* cuando esta activado muestra la respuesta correcta. Así podemos ir comparando la respuesta del usuario con la correcta.**

7.1.4 Tiempo agotado

Cuando se trata de una pregunta dependiente del tiempo y en caso de haber expirado el tiempo, se muestra una pagina informando de dicho suceso. La siguiente figura muestra el aspecto de esta página:

Figura 7.7

- **El botón *respuestaCorrecta* muestra la respuesta correcta.**

7.1.5 Información sobre el fichero QTI

Se ha añadido a la pagina inicial un botón cuyo nombre es Analizar que nos va a permitir a consultar la información sobre el tipo de pregunta, las respuestas posibles y la respuesta correcta del fichero QTI que a la hora de integrar el reproductor a SIETTE, toda esa información será transmitida implícitamente desde el reproductor al sistema SIETTE.

La siguiente figura muestra el aspecto de esta página:

Figura 7.8

- **El botón regresar nos lleva a la pagina inicial.**

7.2 Ejemplos de preguntas QTI.

En este apartado vamos a probar nuestra aplicación sobre ejemplos de preguntas QTI. Mostraremos la parte de presentación de la pregunta extraída del fichero QTI y la parte de la evaluación de la misma.

7.2.1 Pregunta de tipo verdadero/falso.

7.2.1.1 Presentación

La siguiente figura muestra la parte responsable de la presentación de la pregunta.

```
<duration>5000</duration>
<presentation label = "QTIExample006">
        <material>
                <mattext>LA CAPITAL DE MARRUECOS ES RABAT</mattext>
        </material>
        <response_lid ident = "MCb_01" rcardinality = "Single" rtiming = "No">
                <render_choice shuffle = "Yes" maxnumber="-1" minnumber="-1">
                        <flow_label>
                                <response_label ident = "A">
                                        <material>
                                                <mattext>VERDADERO</mattext>
                                        </material>
                                </response_label>
                        </flow_label>
                        <flow_label>
                                <response_label ident = "B">
                                        <material>
                                                <mattext>FALSO</mattext>
                                        </material>
                                </response_label>
                        </flow_label>
                </render_choice>

        </response_lid>
</presentation>
```

Figura 7.9

La siguiente figura muestra como se ha producido la pregunta.

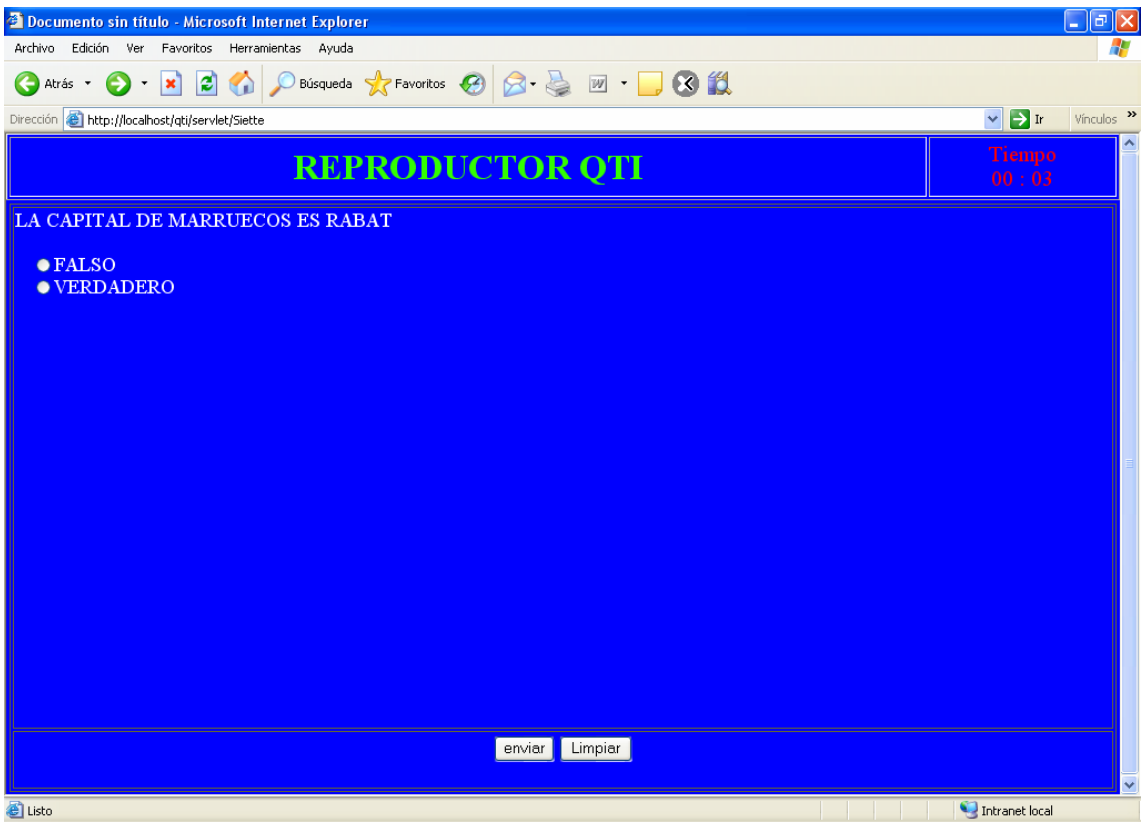


Figura 7.10

7.2.1.2 Evaluación

La parte del fichero QTI responsable de la evaluación de la pregunta es la siguiente:

```
<resprocessing>
        <outcomes>
                <decvar vartype = "Integer" defaultval = "0"/>
        </outcomes>
        <respcondition title = "Correct">
                <conditionvar>
                        <varequal respident = "MCb_01">A</varequal>
                </conditionvar>
                <setvar action = "Set">1</setvar>
                <displayfeedback feedbacktype = "Response" linkrefid = "Correct"/>
        </respcondition>
        <respcondition title = "inCorrect">
                <conditionvar>
                        <not>
                                <varequal respident = "MCb_01">B</varequal>
                        </not>
                </conditionvar>
        <setvar action = "Set">0</setvar>
        <displayfeedback feedbacktype = "Response" linkrefid = "inCorrect"/>
        </respcondition>
</resprocessing>
```

Figura 7.11

La siguiente figura muestra el resultado de la evaluación tras elegir la opción VERDADERO que es la correcta.

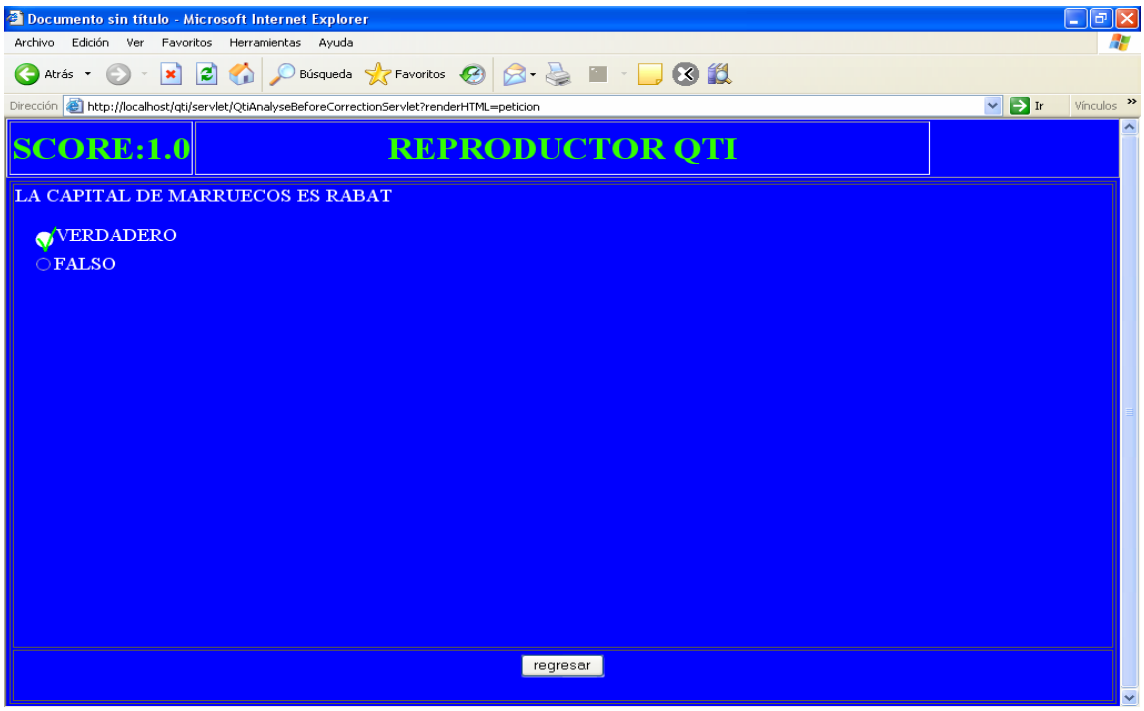


Figura 7.12

7.2.2 Preguntas de opción múltiple.

Hay dos categorías en este tipo de preguntas, por un lado tenemos preguntas de opción múltiple de respuesta única en la que solo se permite al usuario elegir una opción de entre muchas y por otro lado preguntas de opción múltiple de respuesta múltiple en la que el usuario tiene la posibilidad de seleccionar mas de una respuesta.

7.2.2.1 Preguntas de opción múltiple de respuesta única.

7.2.2.1.1 Presentación

La siguiente figura muestra la parte de presentación de la pregunta.

```
<presentation label = "QTIExample006">
   <material>
      <mattext>Cual de estos monumentos se encuentra en Paris ?</mattext>
   </material>
   <response_lid ident = "MCb_01" rcardinality = "Single" rtiming = "No">
      <render_choice shuffle = "Yes" minnumber="1" maxnumber="1">
         <flow_label>
            <response_label ident = "A">
               <material>
                  <matimage uri="http:/qti/imagenes/reinoUnido.jpg"></matimage>
               </material>
            </response_label>
         </flow_label>
         <flow_label>
            <response_label ident = "B">
               <material>
                  <matimage uri="http:/qti/imagenes/francia.jpg"></matimage>
               </material>
            </response_label>
         </flow_label>
         <flow_label>
            <response_label ident = "C">
               <material>
                  <matimage uri="http:/qti/imagenes/italia.jpg"></matimage>
               </material>
            </response_label>
         </flow_label>
      </render_choice>
   </response_lid>
</presentation>
```

Figura 7.13

Las posibles respuestas en este ejemplo son imágenes.

La siguiente figura muestra la reproducción de la pregunta:

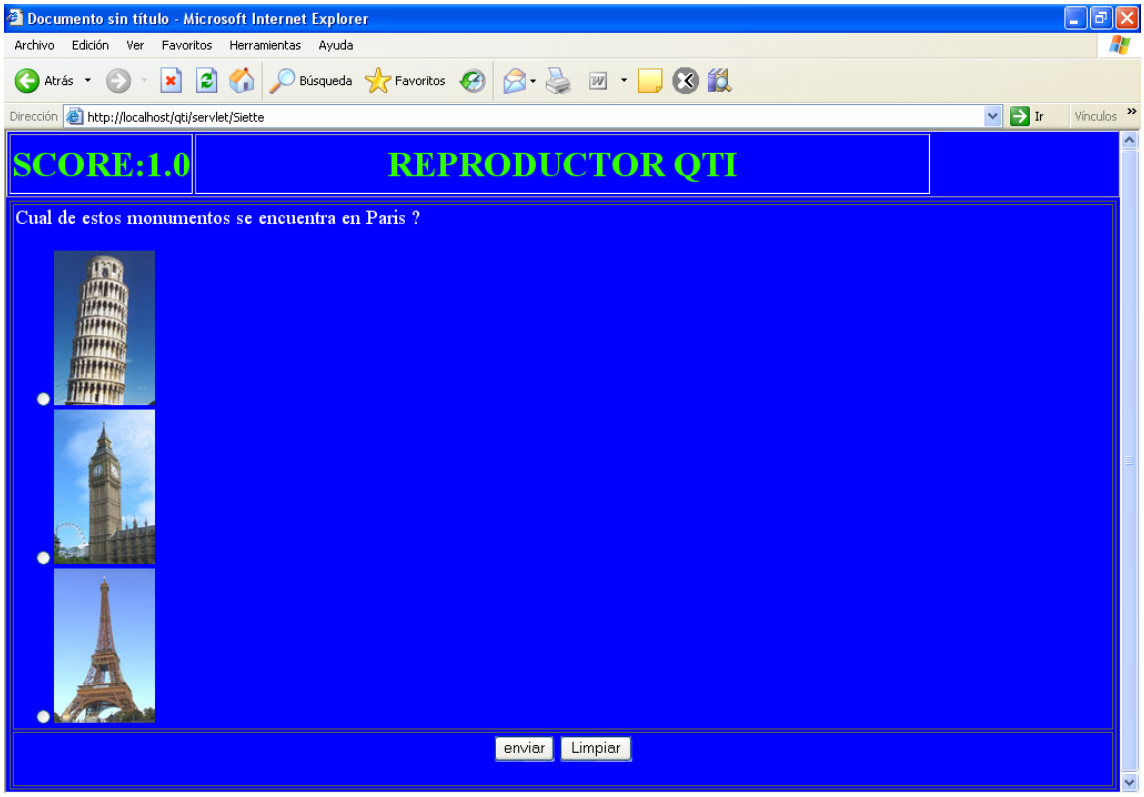


Figura 7.14

7.2.2.1.2 Evaluación

La evaluación de la pregunta de la figura anterior viene determinada por el código de la siguiente figura.

```
<resprocessing>
        <outcomes>
                <decvar vartype = "Integer" defaultval = "0"/>
        </outcomes>
        <respcondition title = "Correct">
                <conditionvar>
                        <varequal respident = "MCb_01">B</varequal>
                </conditionvar>
                <setvar action = "Set">1</setvar>
                <displayfeedback feedbacktype = "Response" linkrefid = "Correct"/>
        </respcondition>
        <respcondition title = "inCorrect">
                <conditionvar>
                        <not>
                                <varequal respident = "MCb_01">B</varequal>
                        </not>
                </conditionvar>
                <setvar action = "Set">0</setvar>
                <displayfeedback feedbacktype = "Response" linkrefid = "inCorrect"/>
        </respcondition>
</resprocessing>
```

Figura 7.15

La siguiente figura muestra la evaluación tras seleccionar la foto del *Big Ben*.

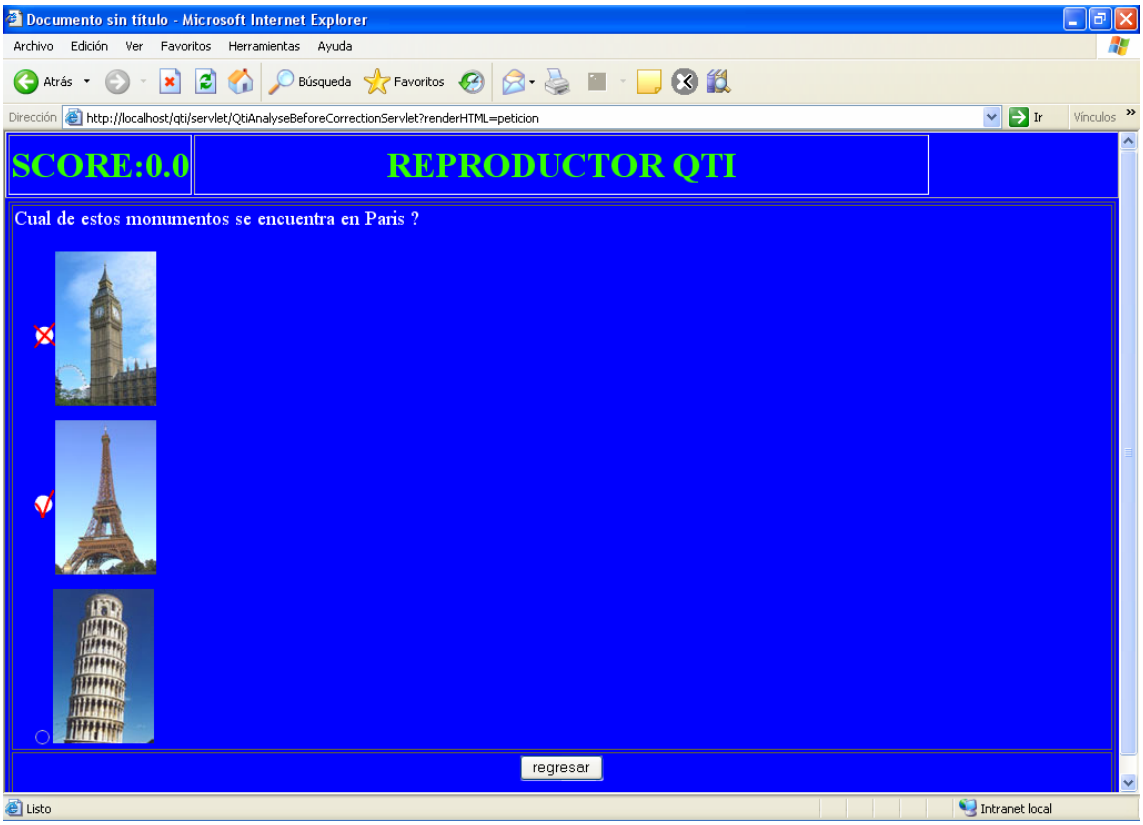


Figura 7.16

7.2.2.2 Preguntas de opción múltiple de respuesta múltiple

7.2.2.2.1 Presentación

La siguiente figura muestra la parte de presentación de la pregunta.

```
<presentation label = "BasicExample005b">
        <material>
                <mattext>Cuales de los siguientes elementos forman el agua ?</mattext>
        </material>
        <response_lid ident = "MR01" rcardinality = "Multiple" rtiming = "No">
                <render_choice shuffle = "Yes" minnumber = "1" maxnumber = "2">
                        <flow_label>
                                <response_label ident = "A">
                                        <material>
                                                <mattext>Hydrogen</mattext>
                                        </material>
                                </response_label>
                        </flow_label>
                                <flow_label>
                                        <response_label ident = "B">
                                                <material>
                                                        <mattext>Helium</mattext>
                                                </material>
                                        </response_label>
                                </flow_label>
                        <flow_label>
                                <response_label ident = "C">
                                        <material>
                                                <mattext>Carbon</mattext>
                                        </material>
```

```
                    </response_label>
                </flow_label>
                <flow_label>
                    <response_label ident = "D">
                        <material>
                            <mattext>Oxygen</mattext>
                        </material>
                    </response_label>
                </flow_label>
                <flow_label>
                    <response_label ident = "E">
                        <material>
                            <mattext>Nitrogen</mattext>
                        </material>
                    </response_label>
                </flow_label>
                <flow_label>
                    <response_label ident = "F" rshuffle="No">
                        <material>
                            <mattext>Chlorine</mattext>
                        </material>
                    </response_label>
                </flow_label>
            </render_choice>
        </response_lid>
```

Figura 7.17

La siguiente figura muestra como se ha presentado la pregunta.

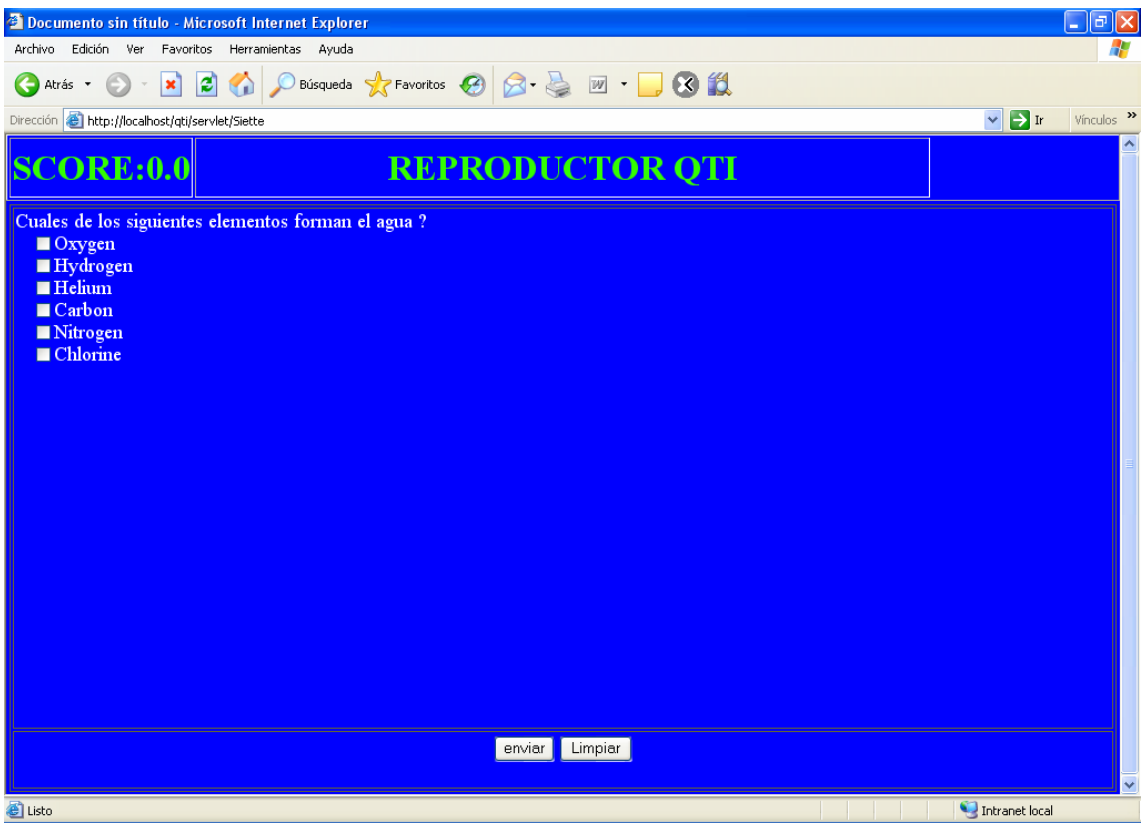


Figura 7.18

7.2.2.2.2 Evaluación

La evaluación de la pregunta de la figura anterior viene determinada por el siguiente codigo.

```
<resprocessing>
        <outcomes>
                <decvar varname = "SCORE1" vartype = "Decimal" defaultval = "0"/>
        </outcomes>
        <respcondition continue="Yes">
                <conditionvar>
                        <varequal respident = "MR01">A</varequal>
                </conditionvar>
                <setvar action = "Add" varname = "SCORE1">1</setvar>
        </respcondition>
        <respcondition continue="Yes">
                <conditionvar>
                        <varequal respident = "MR01">D</varequal>
                </conditionvar>
                <setvar action="Add" varname="SCORE1">1</setvar>
        </respcondition>
        <respcondition continue="Yes">
                <conditionvar>
                        <varequal respident = "MR01">B</varequal>
                </conditionvar>
                <setvar action="Add" varname="SCORE1">-1</setvar>
        </respcondition>
        <respcondition continue="Yes">
                <conditionvar>
                        <varequal respident = "MR01">C</varequal>
                </conditionvar>
                <setvar action="Add" varname="SCORE1">-1</setvar>
        </respcondition>
        <respcondition continue="Yes">
                <conditionvar>
                        <varequal respident = "MR01">E</varequal>
                </conditionvar>
                <setvar action="Add" varname="SCORE1">-1</setvar>
        </respcondition>
        <respcondition continue="Yes">
                <conditionvar>
                        <varequal respident = "MR01">F</varequal>
                </conditionvar>
                <setvar action = "Add" varname = "SCORE1">-1</setvar>
        </respcondition>
</resprocessing>
```

Figura 7.19

La siguiente figura muestra la evaluación de la pregunta tras elegir como respuesta la opción Helium y Oxygen. La respuesta correcta es la opción Oxygen y la opción Hydrogen.

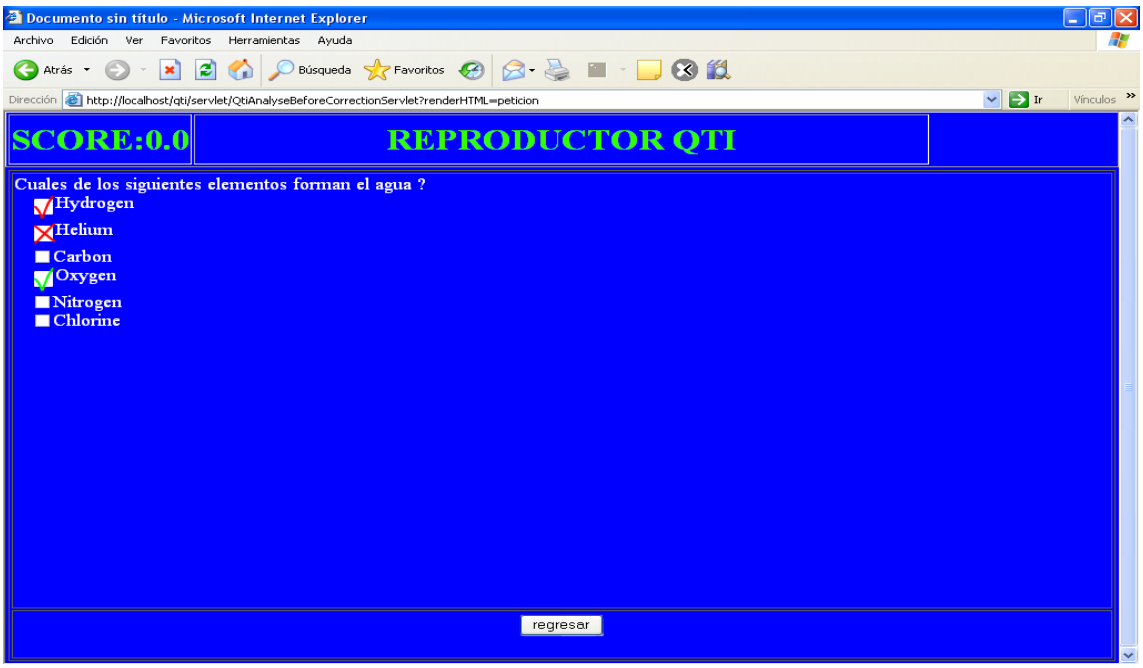


Figura 7.20

7.2.3 Pregunta de tipo respuesta libre

Este tipo de pregunta permite al usuario introducir libremente su respuesta en una caja de texto. La caja de texto puede ser de una sola línea o de muchas líneas. Vamos a ver el ejemplo de una sola línea.

7.2.3.1 Presentación

El código que determina el aspecto de la pregunta viene determinado por la siguiente figura.

```
<presentation label = "BasicExample012b">
    <flow>
        <material>
            <mattext>Completa la secuencia: </mattext>
        </material>
        <flow>
            <material>
                <mattext>Winter, Spring, Summer, </mattext>
            </material>
            <response_str ident = "FIB01" rcardinality = "Single" rtiming = "No">
                <render_fib fibtype = "String" prompt = "Dashline" maxchars = "6">
                    <response_label ident = "A"/>
                        <material>
                            <mattext>.</mattext>
                        </material>
                </render_fib>
            </response_str>
        </flow>
    </flow>
</presentation>
```

Figura 7.21

Después de haber reproducido el código de la figura anterior, obtenemos la siguiente figura:

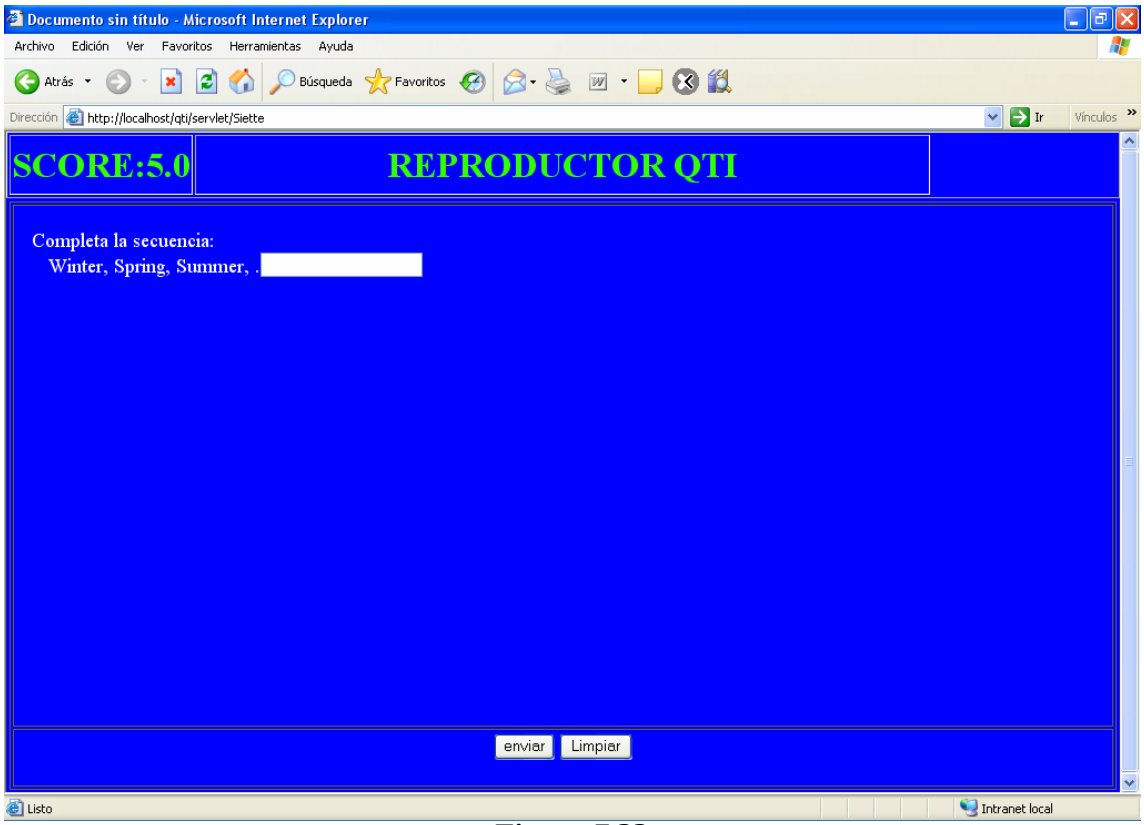


Figura 7.22

7.2.3.1 Evaluación

El código que controla la autoevaluacion de la respuesta del usuario viene detallado en la siguiente figura:

```
<resprocessing>
        <outcomes>
                <decvar varname = "FIBSCORE" vartype = "Integer" defaultval = "0"/>
        </outcomes>
        <respcondition>
                <conditionvar>
                        <varequal respident = "FIB01" case = "Yes">Autumn</varequal>
                </conditionvar>
                <setvar action = "Set" varname = "FIBSCORE">1</setvar>
        </respcondition>
        <respcondition>
                <conditionvar>
                        <not>
                                <varequal respident = "FIB01" case = "Yes">Autumn</varequal>
                        </not>
                </conditionvar>
                <setvar action = "Set" varname = "FIBSCORE">0</setvar>
        </respcondition>
</resprocessing>
```

Figura 7.23

Según el código de la evaluación, la respuesta debe de ser igual a *Autumn* y además hay que respetar la mayúscula y la minúscula.

La siguiente figura muestra el aspecto de la autoevaluacion de nuestra aplicación.

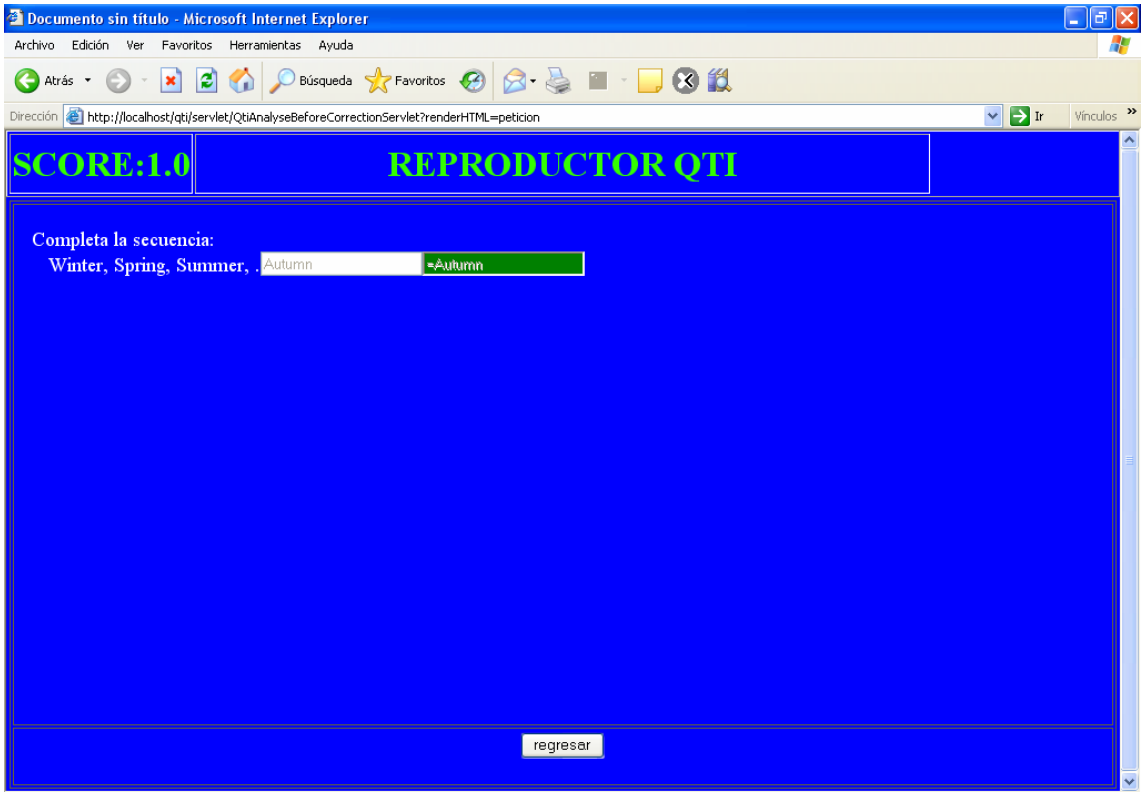


Figura 7.24

En este caso presentamos junto a la caja de texto donde ha introducido el usuario su respuesta otra caja de texto que contiene el patrón de la respuesta correcta, en este caso el patrón es =*Autumn*.

7.2.4 Pregunta de tipo Image Hotspot

Este tipo de pregunta se utiliza cuando se quiera que el usuario identifica una zona dentro de una imagen, como es el caso de una ciudad dentro de una mapa etc.

Hay dos tipos de preguntas dentro de este tipo. La primera es simplemente identificar una zona dentro de una imagen y la segunda con posibilidad de conectar puntos de una imagen. Se ha utilizado los Applets de Java para reproducir este tipo de preguntas.

7.2.4.1 Image Hotspot sin la posibilidad de conectar los puntos seleccionados

7.2.4.1.1 Presentación.

La siguiente figura muestra el código de presentación de la pregunta de tipo Image Hotspot donde no es posible conectar los puntos seleccionados.

```
<presentation label = "BasicExample006b">
    <response_lid ident = "MC04" rcardinality = "Single" rtiming = "No">
        <material>
            <mattext>Cual es la capital de ESPAÑA?</mattext>
        </material>
        <render_hotspot showdraw="No" minnumber="1" maxnumber="1" >
            <material>
            <matimage imagtype = "image/gif" uri = "http:/qti/imagenes/mapa1.gif" x0 = "150" y0 = "51"
/>
            </material>
            <response_label ident = "A" rarea = "Ellipse">311,82,10,10</response_label>
            <response_label ident = "B" rarea = "Ellipse">457,87,10,10</response_label>
            <response_label ident = "C" rarea = "Ellipse">513,144,10,10</response_label>
            <response_label ident = "D" rarea = "Ellipse">597,150,10,10</response_label>
            <response_label ident = "E" rarea = "Ellipse">436,187,10,10</response_label>
            <response_label ident = "F" rarea = "Ellipse">532,222,10,10</response_label>
            <response_label ident = "G" rarea = "Ellipse">615,215,10,10</response_label>
            <response_label ident = "H" rarea = "Ellipse">406,282,10,10</response_label>
            <response_label ident = "I" rarea = "Ellipse">372,301,10,10</response_label>
            <response_label ident = "J" rarea = "Ellipse">417,326,10,10</response_label>
        </render_hotspot>
    </response_lid>
</presentation>
```

Figura 7.25

Después de haber reproducido el código anterior, obtenemos lo siguiente.

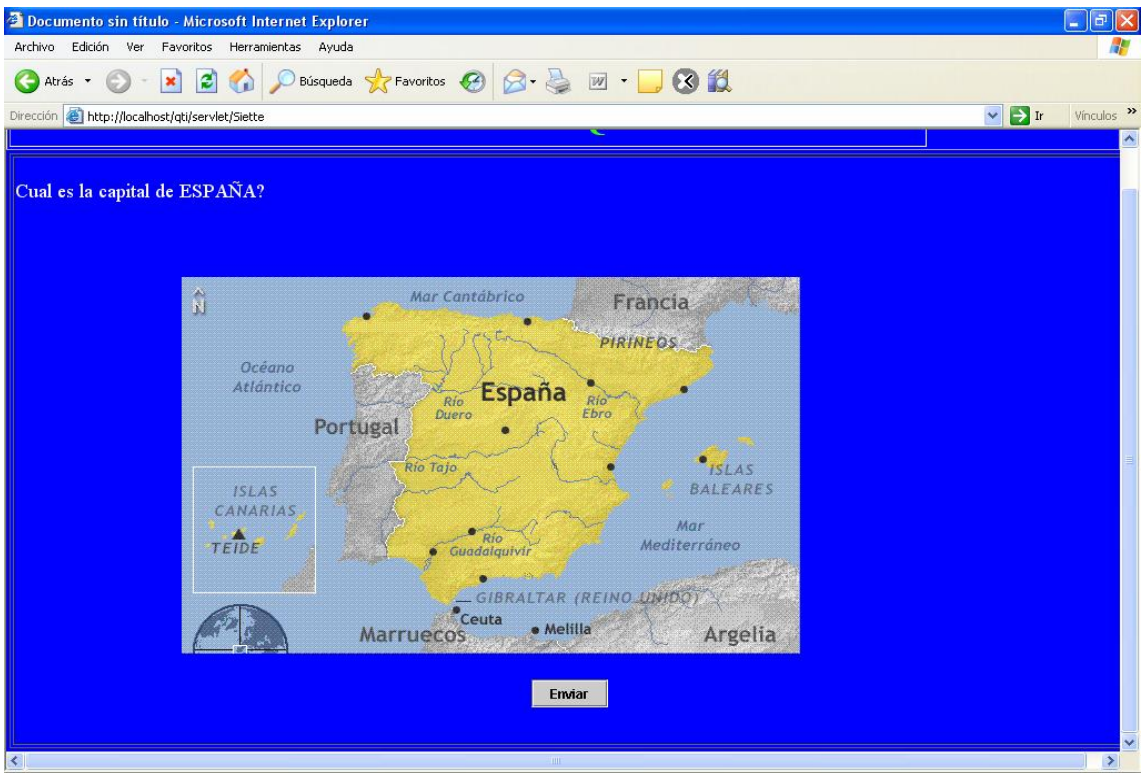


Figura 7.26

7.2.4.1.2 Evaluación

El código que controla la autoevaluacion de la respuesta del usuario viene detallado en la siguiente figura.

```
<resprocessing>
        <outcomes>
                <decvar varname = "SCORE1" vartype = "Integer" defaultval = "0"/>
        </outcomes>
        <respcondition>
                <conditionvar>
                        <varequal respident = "MC04">E</varequal>
                </conditionvar>
                <setvar action = "Set" varname = "SCORE1">10</setvar>
        </respcondition>
        <respcondition>
                <conditionvar>
                        <not>
                                <varequal respident = "MC04">E</varequal>
                        </not>
                </conditionvar>
                <setvar action = "Set" varname = "SCORE1">0</setvar>
        </respcondition>
</resprocessing>
```

Figura 7.27

La siguiente figura muestra la respuesta del usuario que en este caso no es la correcta.

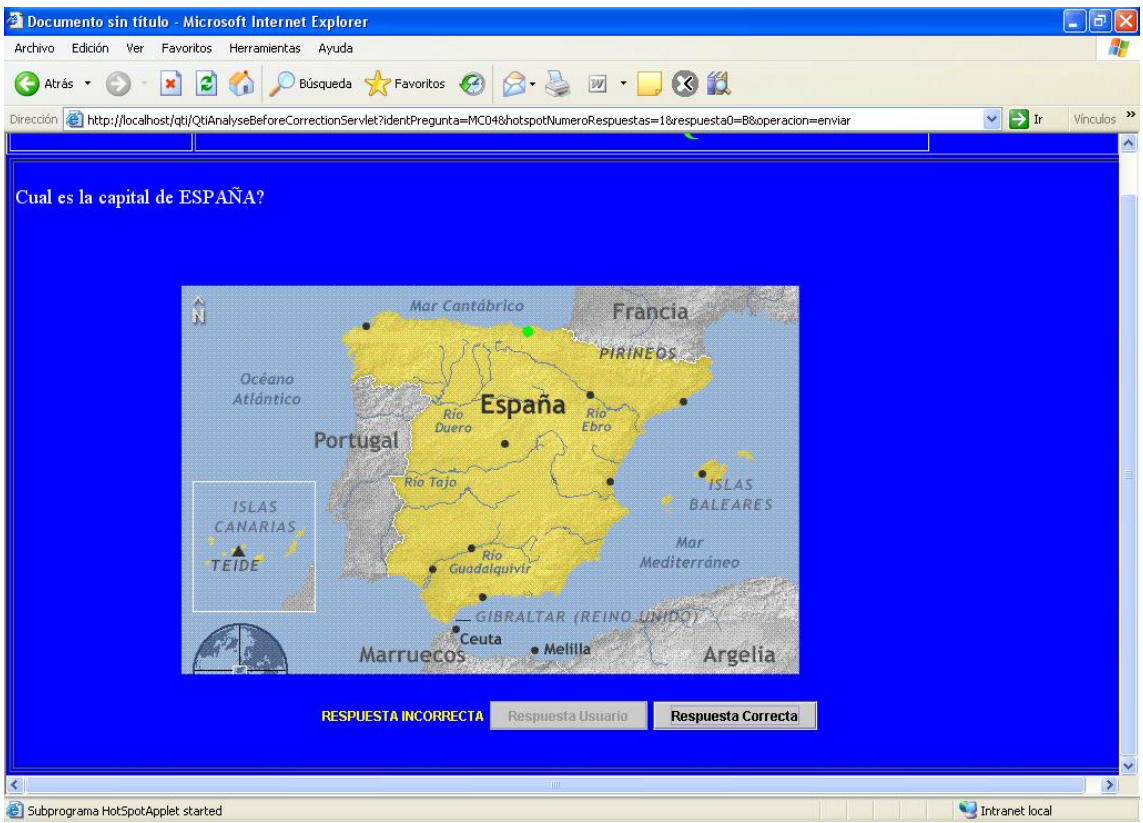


Figura 7.28

Si nos fijamos en la figura anterior, la respuesta del usuario viene marcada por un círculo verde. Hay una etiqueta de texto avisando de que la respuesta del usuario no es la correcta. Para ver la respuesta correcta simplemente tenemos que pulsar en el botón *Respuesta Correcta*. La siguiente figura es el resultado de pulsar en dicho botón.

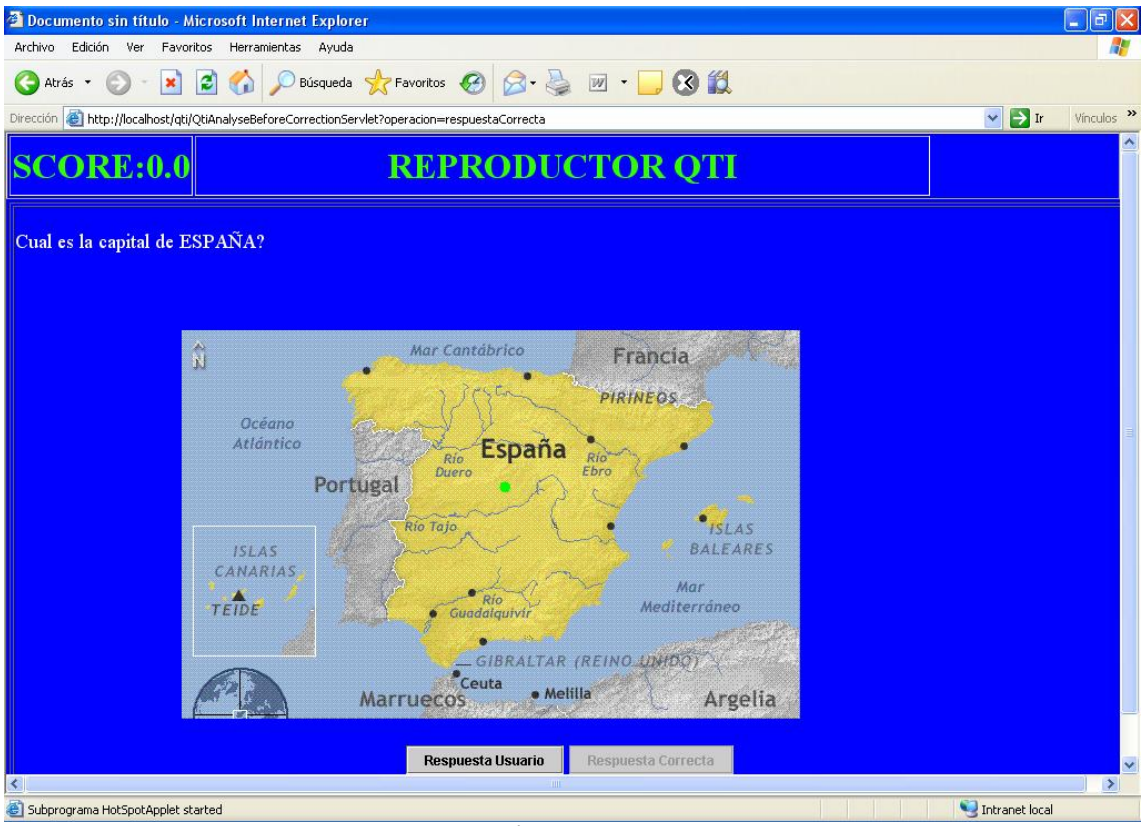


Figura 7.29

La respuesta correcta es el punto del medio que ahora esta marcado por un circulo verde. Si nos fijamos, ahora el botón *Respuesta Usuario* esta habilitado, de esta manera podemos ir alternando entre respuesta de usuario y la respuesta correcta para poder compararlas.

7.2.4.2 Image Hotspot con la posibilidad de conectar los puntos seleccionados

7.2.4.2.1 Presentación

La siguiente figura muestra el código de la presentación de nuestro ejemplo.

```
<presentation label = "BasicExample018">
    <response_lid ident = "CTP01" rcardinality = "Multiple" rtiming = "No">
        <material>
            <mattext>Connecte los puntos para formar un triangulo recto</mattext>
        </material>
        <render_hotspot showdraw = "Yes" maxnumber="3" minnumber="3">
            <material>
              <matimage imagtype = "image/gif" uri = "http:/qti/imagenes/circulo.gif" x0 = "0" width
= "400" y0 = "0" height = "200"/>
            </material>
            <response_label ident = "A" rarea = "Ellipse">199,32,15,15</response_label>
            <response_label ident = "B" rarea = "Ellipse">270,52,15,15</response_label>
            <response_label ident = "C" rarea = "Ellipse">283,135,15,15</response_label>
            <response_label ident = "D" rarea = "Ellipse">203,165,15,15</response_label>
            <response_label ident = "E" rarea = "Ellipse">124,142,15,15</response_label>
            <response_label ident = "F" rarea = "Ellipse">131,52,15,15</response_label>
        </render_hotspot>
   </response_lid>
</presentation>
```

Figura 7.30

Hay que destacar que la única diferencia entre este ejemplo y el anterior es el atributo *showdraw* del elemento *render_hotspot*. Cuando *showdraw* vale Yes es que tenemos la posibilidad de conectar los puntos, en el caso contrario no es posible esa conexión de puntos.

La siguiente figura muestra el aspecto de nuestro ejemplo.

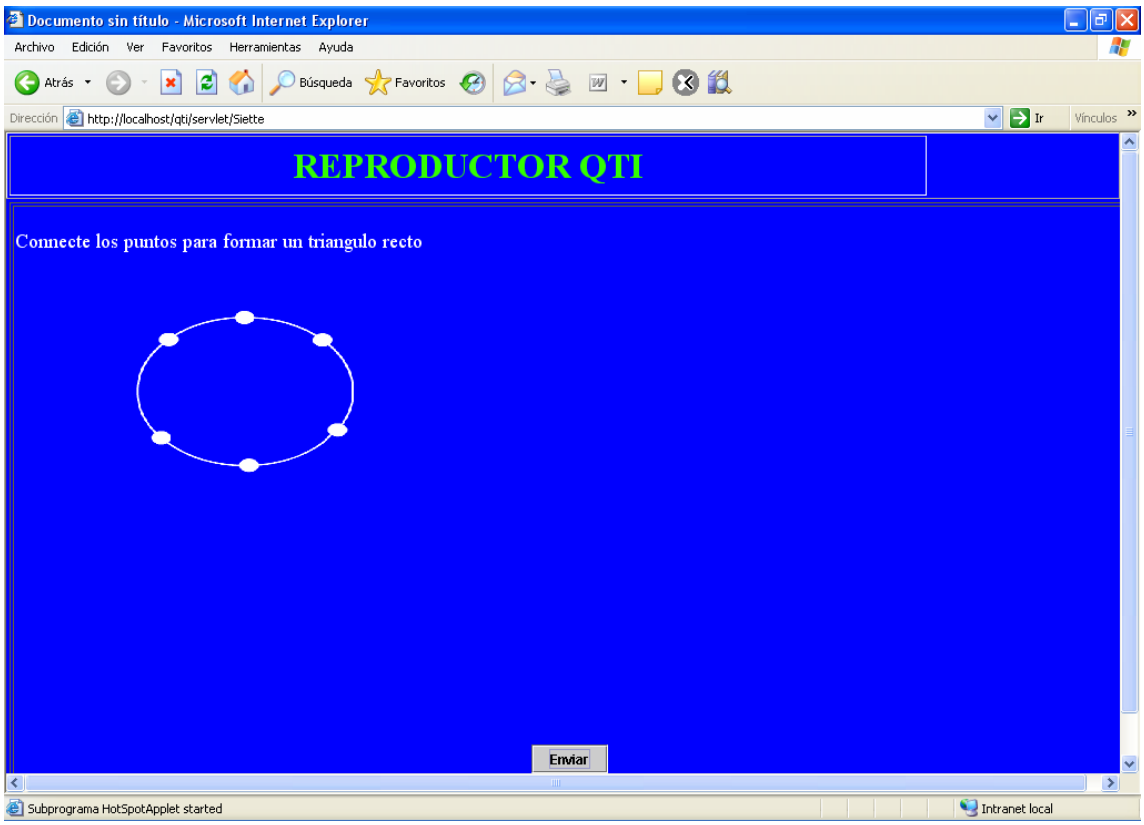


Figura 7.31

7.2.4.2.2 Evaluación.

El código que controla la autoevaluacion de la respuesta del usuario viene detallado en la siguiente figura.

```
<resprocessing>
    <outcomes>
        <decvar varname = "CTPCHOICE" vartype = "Integer" defaultval = "0"/>
    </outcomes>
    <respcondition>
        <conditionvar>
            <or>
                <and>
                    <varequal respident = "CTP01" index="1">A</varequal>
                    <varequal respident = "CTP01" index="2">B</varequal>
                    <varequal respident = "CTP01" index="3">D</varequal>
                    <not>
                        <varequal respident = "CTP01">C</varequal>
                    </not>
                    <not>
                        <varequal respident = "CTP01">E</varequal>
                    </not>
                    <not>
                        <varequal respident = "CTP01">F</varequal>
                    </not>
                </and>
                <and>
                    <varequal respident = "CTP01" index="1">A</varequal>
                    <varequal respident = "CTP01" index="2">D</varequal>
                    <varequal respident = "CTP01" index="3">B</varequal>
                    <not>
                        <varequal respident = "CTP01">C</varequal>
                    </not>
                    <not>
                        <varequal respident = "CTP01">E</varequal>
                    </not>
                    <not>
                        <varequal respident = "CTP01">F</varequal>
                    </not>
                </and>
                <and>
                    <varequal respident = "CTP01" index="1">B</varequal>
                    <varequal respident = "CTP01" index="2">A</varequal>
                    <varequal respident = "CTP01" index="3">D</varequal>
                    <not>
                        <varequal respident = "CTP01">C</varequal>
                    </not>
                    <not>
                        <varequal respident = "CTP01">E</varequal>
                    </not>
                    <not>
                        <varequal respident = "CTP01">F</varequal>
                    </not>
                </and>
                <and>
                    <varequal respident = "CTP01" index="1">B</varequal>
                    <varequal respident = "CTP01" index="2">D</varequal>
                    <varequal respident = "CTP01" index="3">A</varequal>
```

```
                        <not>
                            <varequal respident = "CTP01">C</varequal>
                        </not>
                        <not>
                            <varequal respident = "CTP01">E</varequal>
                        </not>
                        <not>
                            <varequal respident = "CTP01">F</varequal>
                        </not>
                    </and>
                    <and>
                        <varequal respident = "CTP01" index="1">D</varequal>
                        <varequal respident = "CTP01" index="2">A</varequal>
                        <varequal respident = "CTP01" index="3">B</varequal>
                        <not>
                            <varequal respident = "CTP01">C</varequal>
                        </not>
                        <not>
                            <varequal respident = "CTP01">E</varequal>
                        </not>
                        <not>
                            <varequal respident = "CTP01">F</varequal>
                        </not>
                    </and>
                    <and>
                        <varequal respident = "CTP01" index="1">D</varequal>
                        <varequal respident = "CTP01" index="2">B</varequal>
                        <varequal respident = "CTP01" index="3">A</varequal>
                        <not>
                            <varequal respident = "CTP01">C</varequal>
                        </not>
                        <not>
                            <varequal respident = "CTP01">E</varequal>
                        </not>
                        <not>
                            <varequal respident = "CTP01">F</varequal>
                        </not>
                    </and>
                </or>
            </conditionvar>
            <setvar action = "Set" varname = "CTPCHOICE">5</setvar>
        </respcondition>
    </resprocessing>
```

Figura 7.32

La siguiente figura muestra una respuesta del usuario.

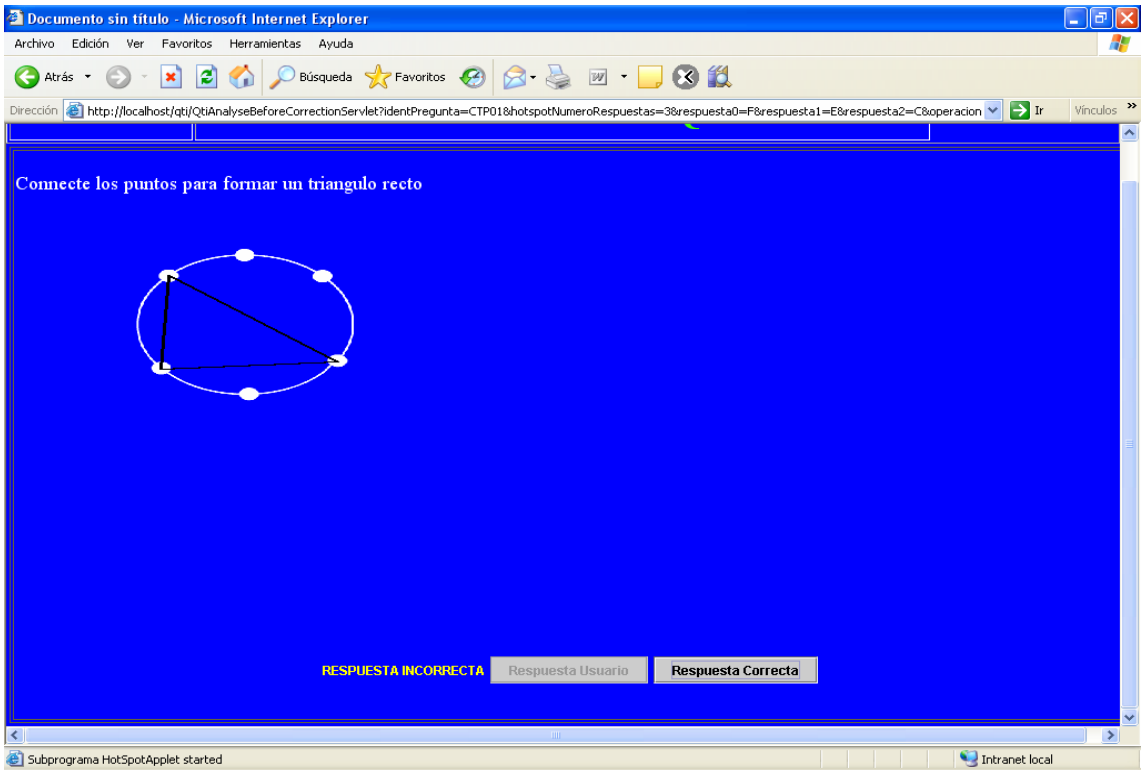


Figura 7.33

La respuesta no es correcta según lo muestra la etiqueta amarilla. Para poder visualizar la respuesta correcta, pulsemos el botón *Respuesta Correcta*, obtenemos lo siguiente.

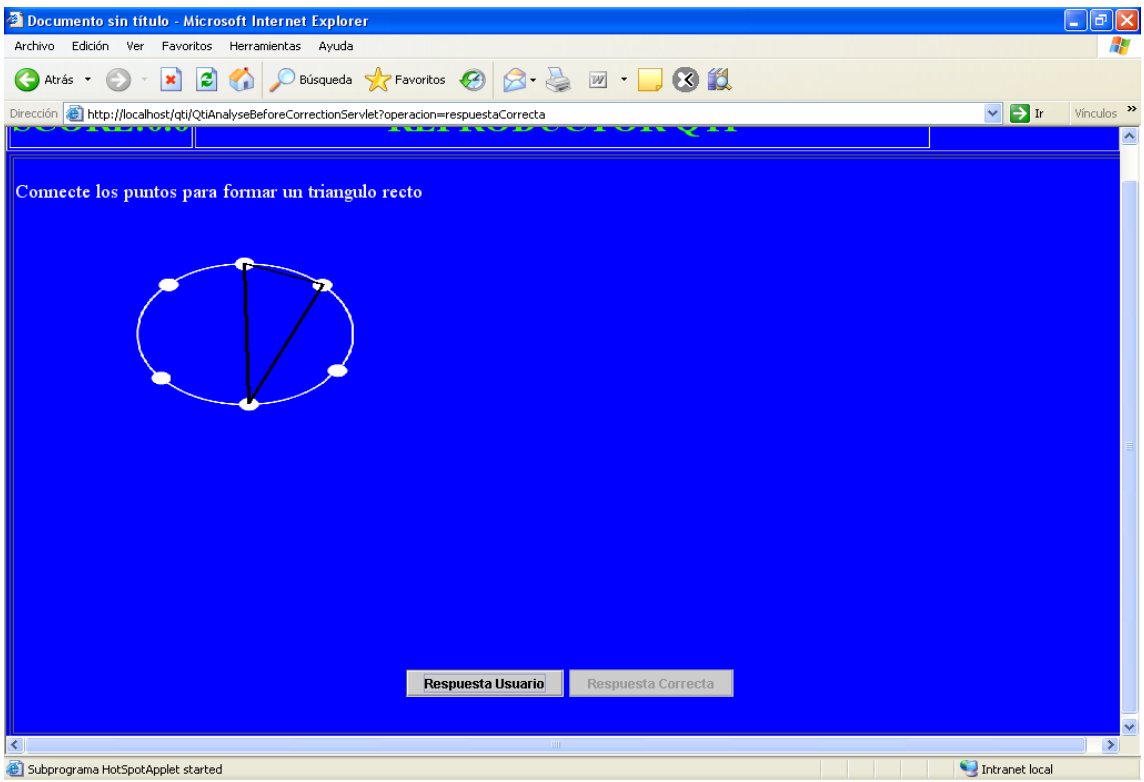


Figura 7.34

7.2.5 Pregunta de tipo ordenación

Consiste en ordenar objetos, ya sea texto o imágenes.

7.2.5.1 Presentación.

La siguiente figura muestra el código del ejemplo.

```
<presentation label = "BasicExample010a">
   <material>
      <mattext>Put these objects in ascending order of size, starting with the smallest on the left hand
side.</mattext>
   </material>
   <response_lid ident = "OB02" rcardinality = "Ordered" rtiming = "No">
   <render_extension>
      <ims_render_object shuffle = "No" orientation = "Row">
         <flow_label>
            <response_label ident = "A">
               <material>
                  <matimage imagtype = "image/gif" uri = "http:/qti/imagenes/imagen0.gif"/>
               </material>
            </response_label>
            <response_label ident = "B">
               <material>
                  <matimage imagtype = "image/gif" uri = "http:/qti/imagenes/imagen1.gif"/>
               </material>
            </response_label>
            <response_label ident = "C">
               <material>
                  <matimage imagtype = "image/gif" uri = "http:/qti/imagenes/imagen2.gif"/>
               </material>
            </response_label>
            <response_label ident = "D">
               <material>
                  <matimage imagtype = "image/gif" uri = "http:/qti/imagenes/imagen3.gif"/>
               </material>
            </response_label>
         </flow_label>
      </ims_render_object>
   </render_extension>
   </response_lid>
</presentation>
```

Figura 7.35

Después de reproducir el ejemplo, obtenemos lo siguiente.

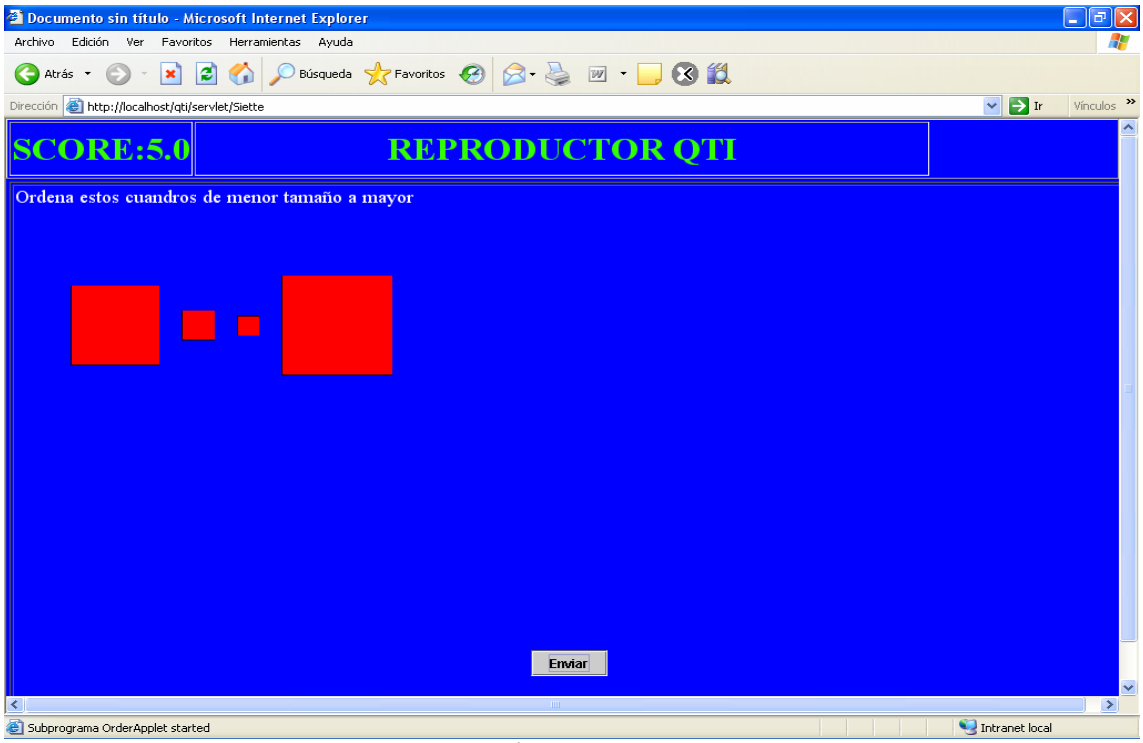


Figura 7.36

7.2.5.2 Evaluación

El código que controla la autoevaluacion de la respuesta del usuario viene detallado en la siguiente figura.

```
<resprocessing>
        <outcomes>
                <decvar varname = "SCORE1" vartype = "Integer" defaultval = "0"/>
        </outcomes>
        <respcondition>
                <conditionvar>
                        <varequal respident = "OB02" index="1">C</varequal>
                        <varequal respident = "OB02" index="2">B</varequal>
                        <varequal respident = "OB02" index="3">A</varequal>
                        <varequal respident = "OB02" index="4">D</varequal>
                </conditionvar>
                <setvar action = "Set" varname = "SCORE1">5</setvar>
        </respcondition>
        <respcondition>
                <conditionvar>
                        <not>
                                <and>
                                        <varequal respident = "OB02" index="1">C</varequal>
                                        <varequal respident = "OB02" index="2">B</varequal>
                                        <varequal respident = "OB02" index="3">A</varequal>
                                        <varequal respident = "OB02" index="4">D</varequal>
                                </and>
                        </not>
                </conditionvar>
                <setvar action = "Set" varname = "SCORE1">0</setvar>
        </respcondition>
</resprocessing>
```

Figura 7.37

La siguiente figura muestra la evaluación de la respuesta correcta.

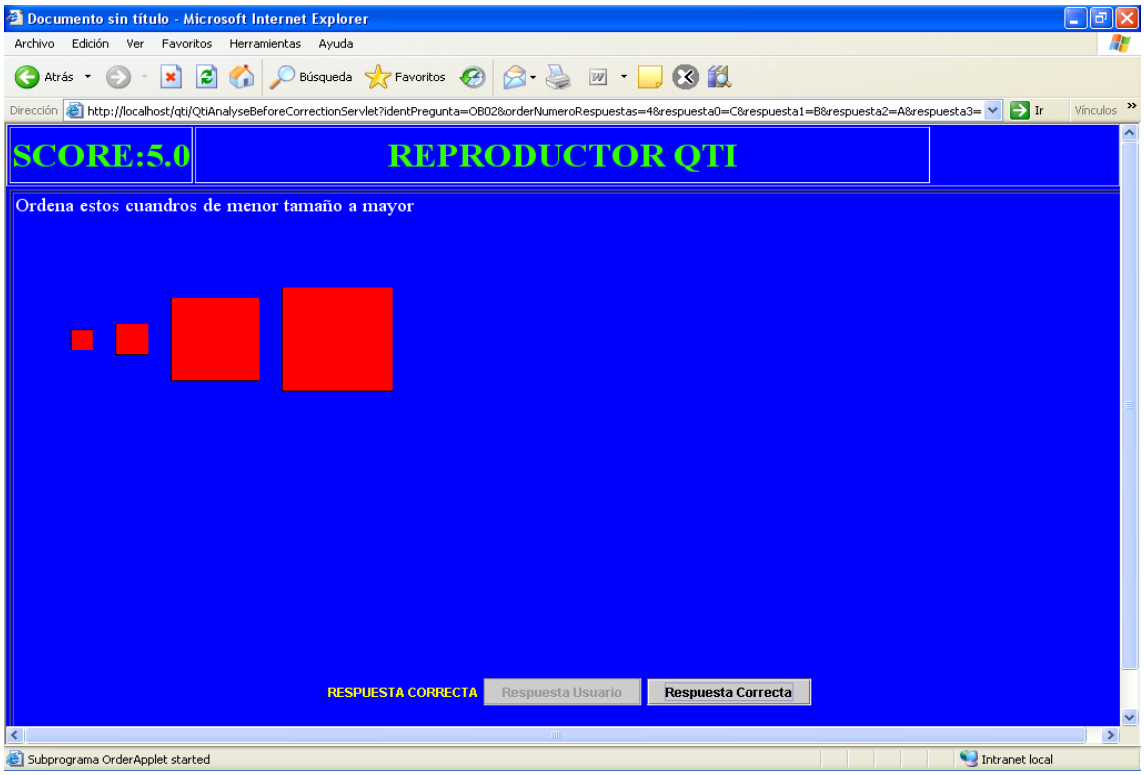


Figura 7.38

7.2.6 Pregunta de tipo Render Slider

En este tipo de pregunta, se le presenta al usuario las diferentes respuestas en una barra vertical u horizontal y a través de un puntero deslizante a lo largo de esa barra, el usuario puede elegir una o varias respuestas.

7.2.6.1 Presentación

La parte responsable de la presentación de la pregunta viene dada por el siguiente código.

```
<presentation label = "BasicExample008b">
        <material>
                <mattext>What is the value of 2 * 3 ?</mattext>
        </material>
        <response_lid ident = "MC05" rcardinality = "Single" rtiming = "No">
                <render_slider lowerbound = "2" upperbound = "10" step = "2" startval = "4"
steplabel = "Yes"  orientation="Vertical" maxnumber="2" >
                        <response_label ident = "A" rrange = "Exact">2  </response_label>
                        <response_label ident = "B" rrange = "Exact">4</response_label>
                        <response_label ident = "C" rrange = "Exact">6</response_label>
                        <response_label ident = "D" rrange = "Exact">8</response_label>
                        <response_label ident = "E" rrange = "Exact">10</response_label>
                </render_slider>
        </response_lid>
</presentation>
```

Figura 7.39

Tras reproducir el código anterior, obtenemos lo siguiente.

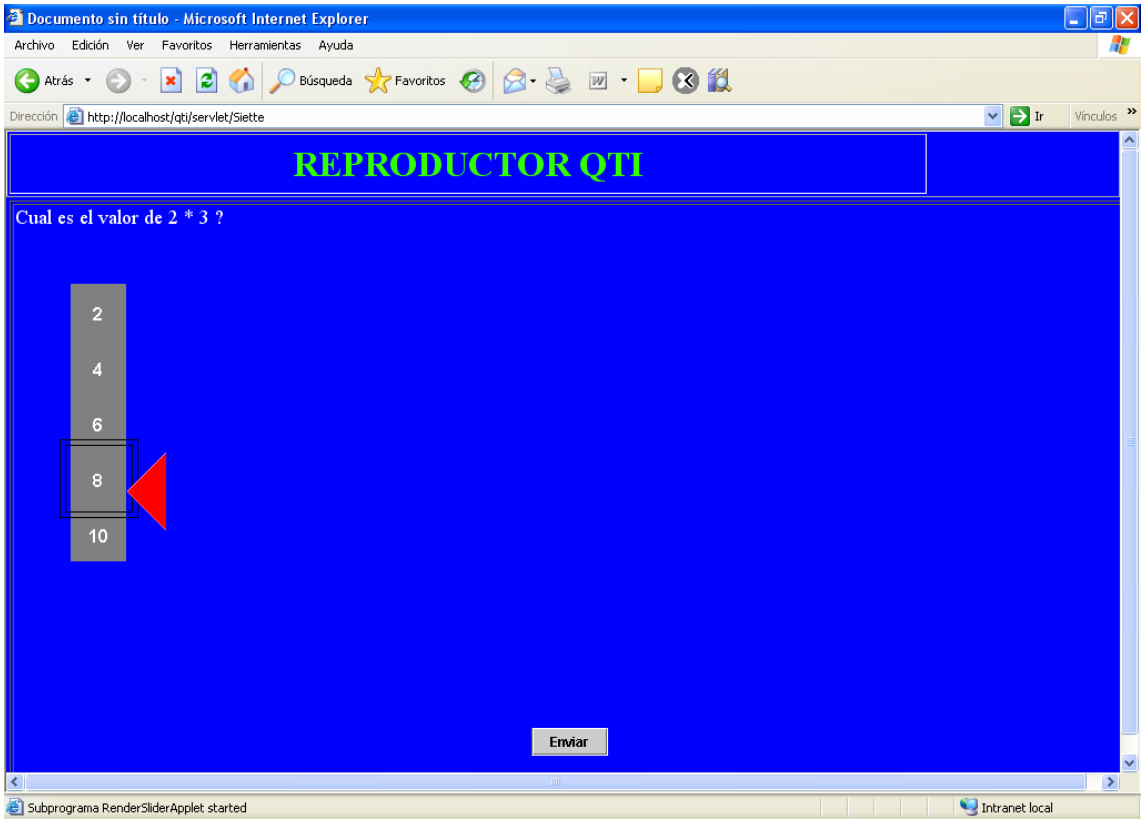


Figura 7.40

7.2.6.2 Evaluación

El siguiente código es el responsable de la autoevaluacion de la pregunta anterior.

```
<resprocessing>
        <outcomes>
                <decvar varname = "SLIDECHOICE" vartype = "Integer" defaultval = "0"/>
        </outcomes>
        <respcondition>
                <conditionvar>
                        <varequal respident = "MC05">C</varequal>
                        <not>
                                <varequal respident = "MC05">A</varequal>
                        </not>
                        <not>
                                <varequal respident = "MC05">B</varequal>
                        </not>
                        <not>
                                <varequal respident = "MC05">D</varequal>
                        </not>
                        <not>
                                <varequal respident = "MC05">E</varequal>
                        </not>
                </conditionvar>
                <setvar action = "Add" varname = "SLIDECHOICE">5</setvar>
        </respcondition>
        <respcondition>
                <conditionvar>
                        <not>
```

```
                        <and>
                            <varequal respident = "MC05">C</varequal>
                            <not>
                                <varequal respident = "MC05">A</varequal>
                            </not>
                            <not>
                                <varequal respident = "MC05">B</varequal>
                            </not>
                            <not>
                                <varequal respident = "MC05">D</varequal>
                            </not>
                            <not>
                                <varequal respident = "MC05">E</varequal>
                            </not>
                        </and>
                    </not>
                </conditionvar>
                <setvar action = "Set" varname = "SLIDECHOICE">0</setvar>
            </respcondition>
        </resprocessing>
```

Figura 7.41

La siguiente figura muestra la evaluación de nuestra aplicación a la respuesta 8.

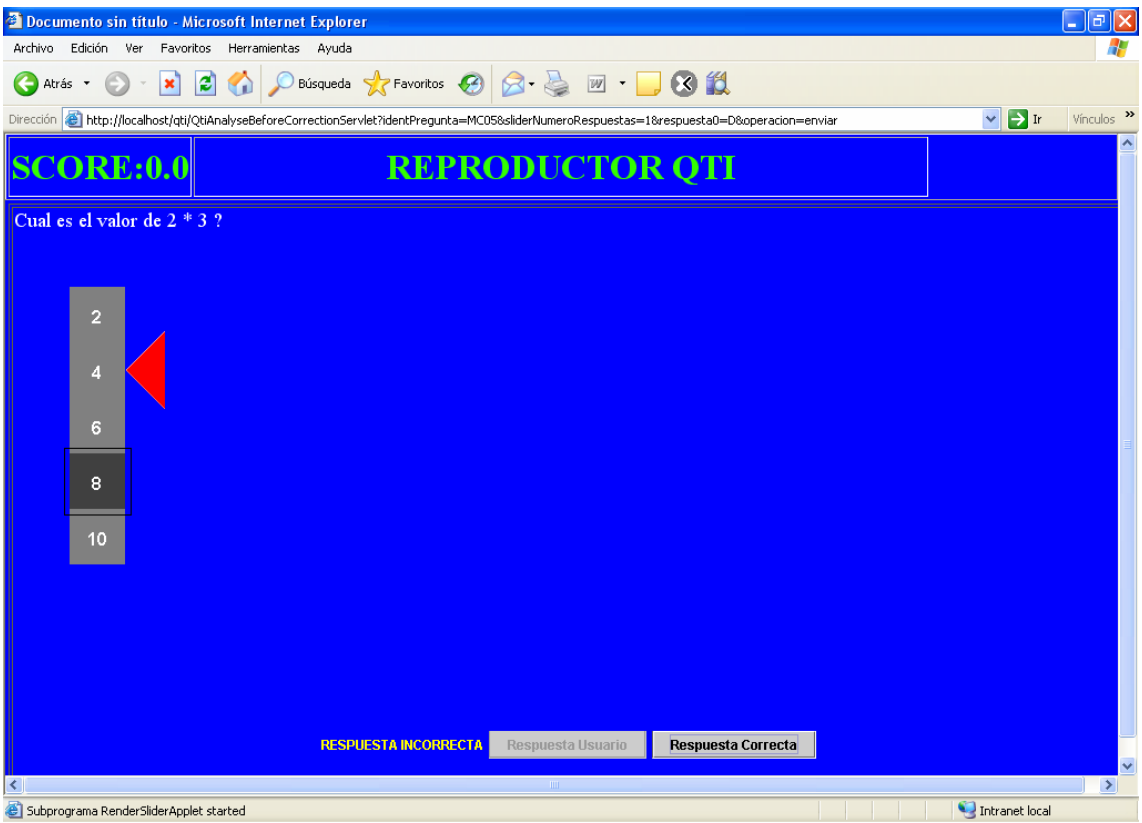


Figura 7.42

Después de pulsar sobre el botón Respuesta Correcta para compararla con la del usuario, obtenemos lo siguiente.

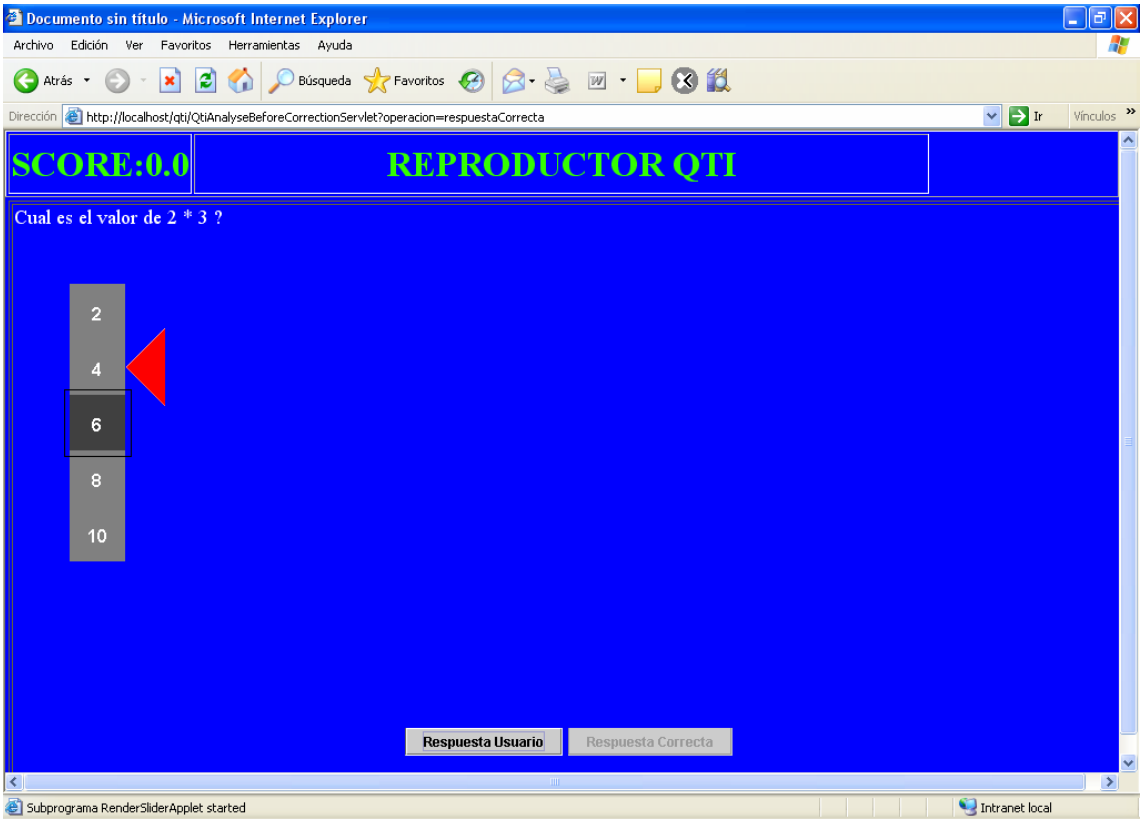


Figura 7.43

7.2.7 Pregunta tipo Drag Drop

Este tipo de pregunta se caracteriza por el hecho de que el usuario tiene la posibilidad de mover objetos y colocarlos en el sitio adecuado. Podemos llamar a este tipo *pregunta de relación*.

7.2.7.1 Presentación

El siguiente código muestra la parte responsable de la presentación de la pregunta.

```
<presentation>
        <material>
                <mattext texttype="text/plain">Pon cada Monumento en su sitio</mattext>
        </material>
        <response_grp ident="DnD_question" rcardinality="Single">
            <render_extension>
                <ims_render_object orientation="Row" shuffle="Yes">
                    <material>
                        <matimage uri="http:/qti/imagenes/dragDrop.gif" x0="0" y0="0" />
                    </material>
                   <response_label ident="a" match_group="a_target,b_target,c_target,d_target"
                match_max="1">
                    <material>
                            <matimage imagtype="image/jpeg" uri="http:/qti/imagenes/EEUU.jpg"
                            x0="50" y0="50"/>
                </material>
                </response_label>
                <response_label ident="b" match_group="a_target,b_target,c_target,d_target"
        match_max="1">
```

```
                    <material>
                        <matimage imagtype="image/jpeg"
                    uri="http:/qti/imagenes/italia.jpg" x0="150" y0="50"/>
                    </material>
                </response_label>
                <response_label ident="c" match_group="a_target,b_target,c_target,d_target"
                match_max="1">
                    <material>
                      <matimage imagtype="image/jpeg"
                    uri="http:/qti/imagenes/reinoUnido.jpg"  x0="50" y0="200"/>
                    </material>
                </response_label>
                <response_label ident="d" match_group="a_target,b_target,c_target,d_target"
                match_max="1">
                    <material>
                      <matimage imagtype="image/jpeg" uri="http:/qti/imagenes/francia.jpg"
                        x0="150" y0="200"/>
                      </material>
                </response_label>
                <response_label ident="a_target" match_max="1" >258,343,90,138</response_label>
                <response_label ident="b_target" match_max="1"
                    area="Rectangle">652,344,90,138</response_label>
                <response_label ident="c_target" match_max="1" >388,346,90,138</response_label>
                <response_label ident="d_target" match_max="1"
                    rarea="Rectangle">521,344,90,138</response_label>
            </ims_render_object>
        </render_extension>
    </response_grp>
</presentation>
```

Figura 7.44

Después de haber reproducido el código anterior, obtenemos lo que muestra la siguiente figura.

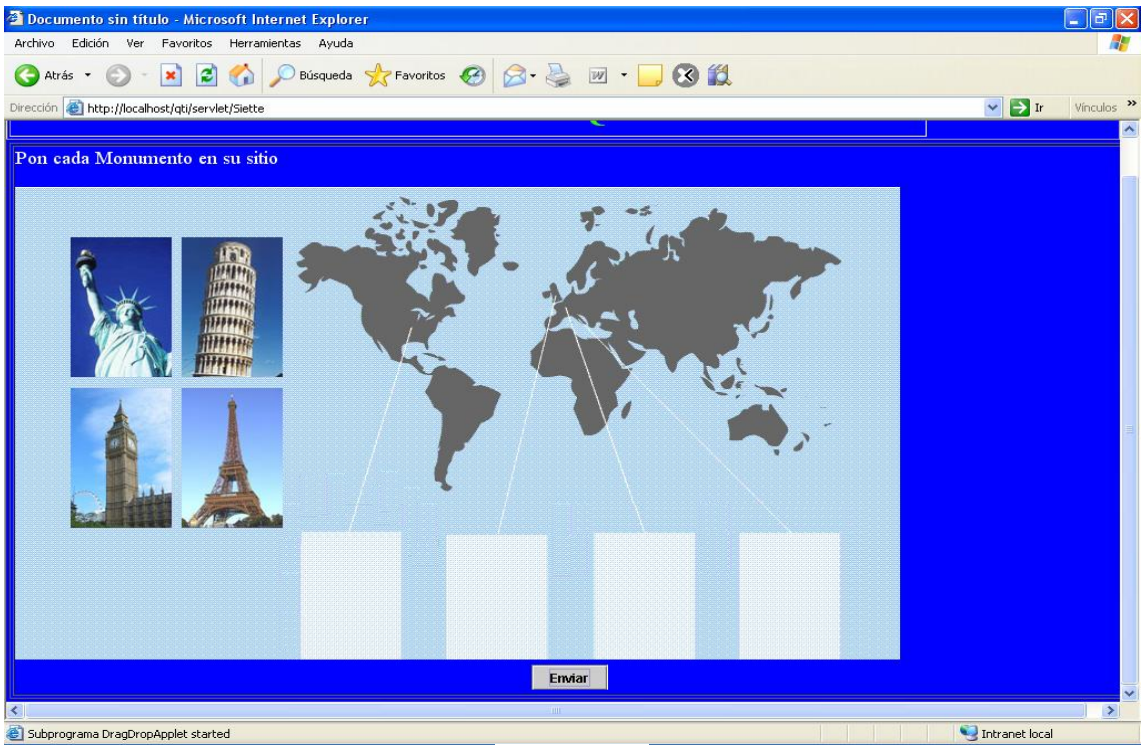


Figura 7.45

7.2.7.2 Evaluación

El siguiente código es el que controla la autoevaluación de la pregunta anterior.

```
<resprocessing>
    <outcomes>
        <decvar varname="SCORE" vartype="Decimal" defaultval="0" maxvalue="1"
minvalue="0"/>
    </outcomes>
    <respcondition title="Correct" continue="Yes">
    <conditionvar>
        <and>
            <varsubset respident="DnD_question" setmatch="Exact">a,a_target</varsubset>
            <varsubset respident="DnD_question" setmatch="Exact">b,b_target</varsubset>
            <varsubset respident="DnD_question" setmatch="Exact">c,c_target</varsubset>
            <varsubset respident="DnD_question" setmatch="Exact">d,d_target</varsubset>
        </and>
    </conditionvar>
    <setvar varname="SCORE" action="Add">5</setvar>
    </respcondition>
    <respcondition title="Incorrect" continue="Yes">
    <conditionvar>
        <not>
          <and>
            <varsubset respident="DnD_question" setmatch="Exact">a,a_target</varsubset>
            <varsubset respident="DnD_question" setmatch="Exact">b,b_target</varsubset>
            <varsubset respident="DnD_question" setmatch="Exact">c,c_target</varsubset>
            <varsubset respident="DnD_question" setmatch="Exact">d,d_target</varsubset>
          </and>
        </not>
   </conditionvar>
   <setvar varname="SCORE" action="Add">-5</setvar>
   </respcondition>
</resprocessing>
```

Figura 7.46

Vamos a mostrar lo que hemos obtenido al responder incorrectamente a la pregunta, hemos puesto los monumentos en los sitios que no corresponden, además hemos dejado un monumento sin tocar. La siguiente figura muestra la respuesta del usuario.

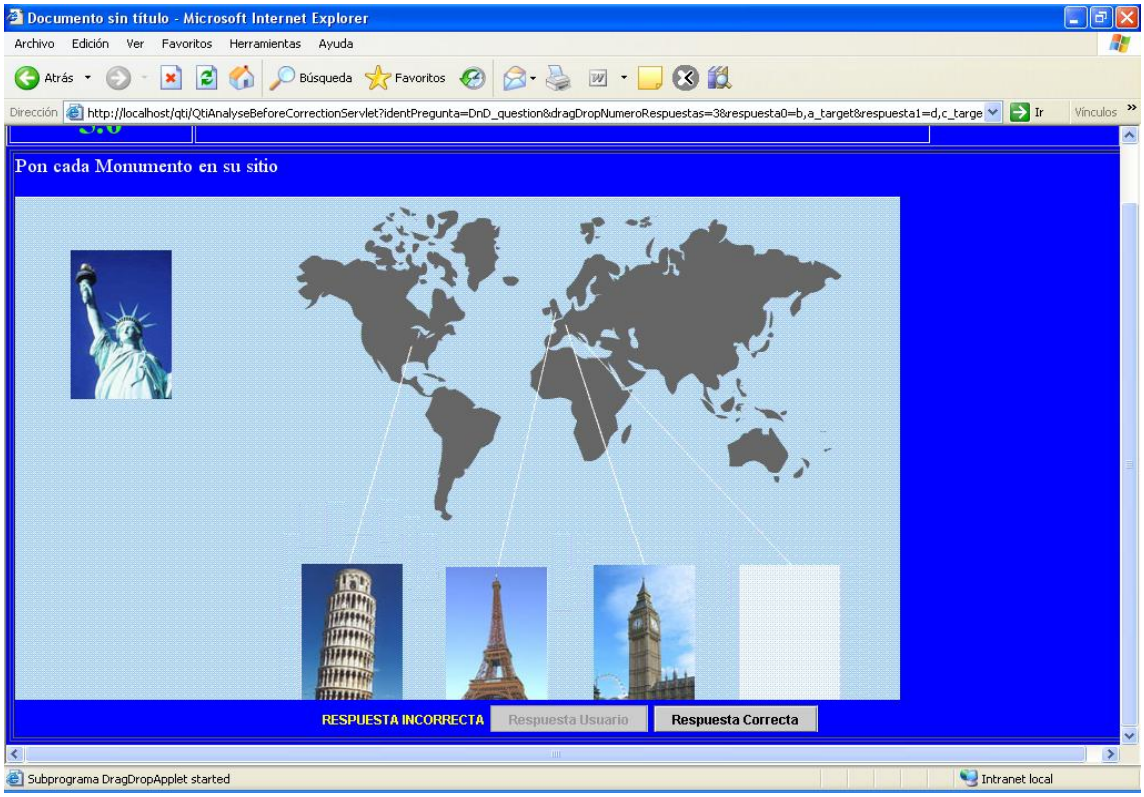


Figura 7.47

Como muestra la figura anterior, la etiqueta en amarillo nos esta avisando de que la respuesta es incorrecta. Para corregirla debemos de pulsar el botón *Respuesta Correcta*. Obtenemos la siguiente figura.

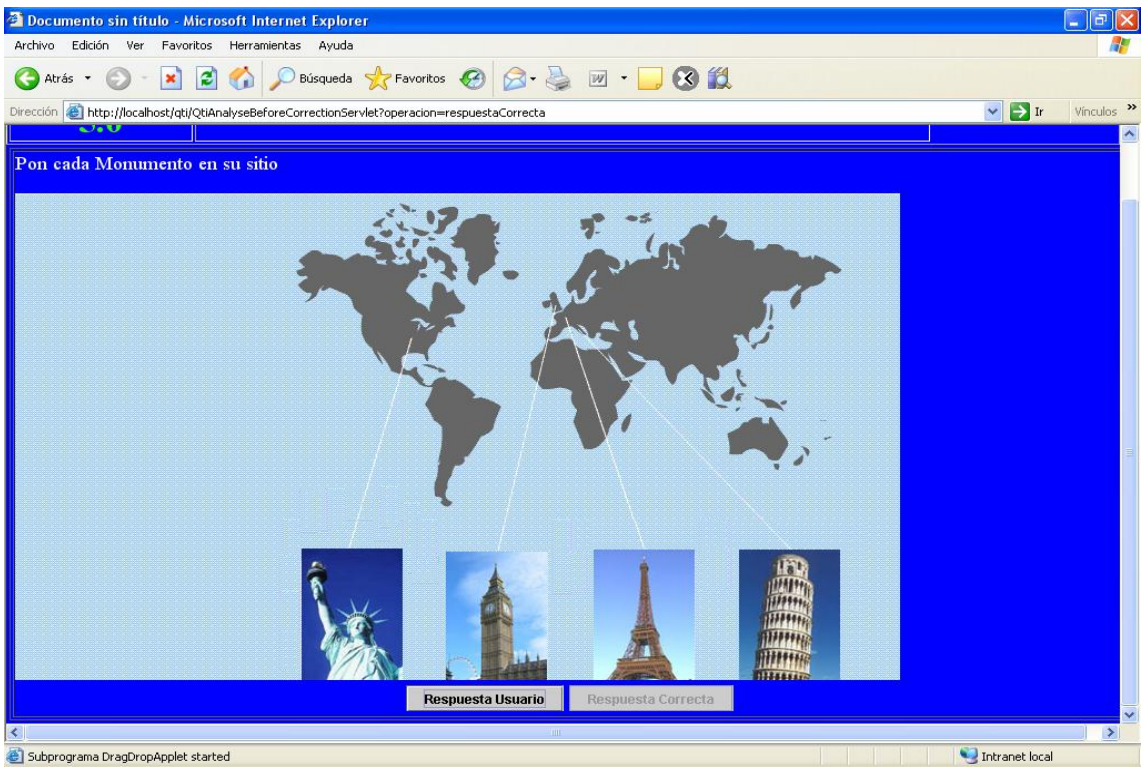


Figura 7.48

8. Conclusiones

8.1 Aportaciones

La elaboración de este proyecto ha supuesto una investigación y estudio centrada fundamentalmente, en el tratamiento de ficheros XML y su enfoque en el sistema QTI. Además, la creación de Applets gráficos en Java ha jugado otro papel importante a la hora de mejorar el aspecto de las preguntas un poco mas complicadas. Pero la parte central, el desarrollo del núcleo, ha sido meramente practico (programación), siendo necesario, para la elaboración de esta parte el conocimiento de los conceptos teóricos. Este requisito de conocimiento previo ha motivado la metodología de trabajo: primero se ha aprendido a manejar toda la teoría necesaria para, posteriormente, poder establecer unas ideas y formas de actuación ante el problema planteado; después hemos llevado a cabo estas ideas mediante un programa, ampliando cuando era necesario los conceptos teóricos; y finalmente se ha elaborado esta memoria.

8.2 Limitaciones

Como en cualquier proyecto fin de carrera, habrá algunas limitaciones o algunos detalles que no se han conseguido realizar, en nuestro caso hemos destacado las siguientes:

- **A lo largo del desarrollo de esta aplicación, se ha utilizado una sola versión del servidor Apache Tomcat que es la versión 4.0 y por tanto no se sabe si la aplicación funcionaria bien en otra versión.**
- **Para mantener la compatibilidad de esta aplicación con el sistema SIETTE, se ha utilizado el marco JDK1.4 en lugar de aprovechar las novedades de la versión 1.5.**

8.3 Futuras líneas

Como en todo desarrollo de una aplicación, la creación de nuevas herramientas abre, en últimos términos, líneas de trabajo y de investigación, ya sea para mejorar dichas herramientas ya creadas o bien para desarrollar otras nuevas con éstas como base. En nuestro caso, las posibles líneas futuras de trabajo que se abren con las siguientes:

- **Podría haberse utilizado el DHTML y Javascript para reproducir las preguntas complicadas (ordenación, relación, etc.…) en lugar de utilizar los Applets de Java puesto que estos últimos suelen ser más lentos a la hora de la ejecución.**
- **Puesto que ya ha salido una nueva versión del QTI que es la versión 2.0, vendría mejor extender el trabajo desarrollado a esta nueva versión.**
- **Realizar la integración de esta aplicación al sistema SIETTE.**

9. BIBLIOGRAFÍA

- **Biblioteca de plantillas de preguntas para SIETTE. Proyecto fin de carrera realizado por Juan Andrés Riveros Vera y dirigido por Eduardo Guzmán de los Riscos y tutorizado por Ricardo Conejo Muñoz. Diciembre de 2001.**

- **Java 2: Manual de usuario y tutorial (3ª Edición). Froufe. Editorial Ra-Ma.**

- **The complete Reference Java 2 Fifth Edition, Herbert Schiltd. McGraw-Hill.**

- **La Biblia Java 2 v5.0. Anaya Multimedia.**

- **Java y XML referencia para programadores. Mohammed Akif, Steven Brodhead Andrei Cioroianu, James Hart, Eric Jung, Dave Writz. Editorial Anaya Multimedia.**

- **Piensa en Java. Segunda Edición. Bruce Eckel. Traducción Jorge González Barturen. Prentice Hall.**

- **XML. José Antonio Ramírez Pérez.**

- **JavaScript Bible. Danny Goodman. GOLD EDITION.**