UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Sitio web para la conversión de lenguaje de marcas a formato QTI/IMS

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Jorge Brusel Borrero

Tutor: Antonio Martí Campoy

Curso 2017-2018

Resum

El següent projecte complix l'objectiu de crear una aplicació web que permeti als docents generar un document al sofisticat estàndard QTI/IMS a partir d'un document escrit en un senzill llenguatge de marques. L'estàndard QTI/IMS[7] és àmpliament utilitzat per plataformes de docència i, en el nostre cas, per Sakai.

Paraules clau: qti, marques, aplicació web, xml

Resumen

El siguiente proyecto cumple el objetivo de crear una aplicación web que permita a los docentes generar un documento en el sofisticado estándar QTI/IMS a partir de un documento escrito en un sencillo lenguaje de marcas. El estándar QTI/IMS es ampliamente utilizado por plataformas de docencia y, en el caso que nos ocupa, por Sakai.

Palabras clave: qti, marcas, aplicación web, xml

Abstract

The following project achieves the goal of creating a web application that allows teachers to generate a document in the sophisticated QTI/IMS standard from a document written in a simple markup language. The QTI/IMS standard is widely used by teaching platforms and, in our case, by Sakai.

Key words: qti, markup, web application, xml

Índice general

Índice de figuras

Índice de tablas

Agradecimientos

Primero quiero agradecer a mis padres por todo el apoyo que me han brindado en mi educación y por la confianza que siempre han depositado en mi.
También, a mi tutor Antonio Martí Campoy, por su ayuda, por estar siempre disponible y darme la libertad para llevar el proyecto de la forma que más me conviniera.
A mis compañeros Héctor, Diego, Josal y Jose Alberto, mi mayor apoyo durante esta etapa educativa.
Y, por último, a todas las personas que de una u otra forma han intervenido en mi educación.
Gracias a todos ellos ha sido posible realizar este proyecto.

CAPÍTULO 1
Introducción

En este apartado se da una breve descripción de qué es lo que nos ha llevado a realizar este proyecto. Se trata la motivación y los objetivos del proyecto. A su vez, se repasan brevemente los aspectos que se van a tratar en esta memoria.

1.1 Motivación

Hoy en día el uso de las plataformas docentes es una gran herramienta para gestionar la educación de los alumnos en las distintas áreas y un aspecto importante en estas plataformas es la evaluación de los estudiantes. En concreto, la herramienta Exámenes de la plataforma docente Poliformat, utilizada en la Universitat Politècnica de València permite introducir las preguntas introduciéndolas manualmente o obteniéndolas automáticamente de baterías de preguntas. Además, aleatoriza los resultados y el orden de las mismas para generar diferentes tipos de exámenes para cada usuario. Toda esta gestión de preguntas y exámenes se codifica en el estándar QTI/IMS. Para el alumno, también es una buena herramienta, pues permite la comodidad de utilizar un ordenador o un dispositivo móvil desde cualquier parte del mundo para acceder a documentos, noticias, foros, realizar exámenes, entre otras utilidades.
Una de las motivaciones para este proyecto era automatizar la tarea de introducir las preguntas una a una por parte del profesor. Por ello, se propone el uso de un lenguaje de marcas, unas herramientas sencillas para generar documentos con este formato y una traducción automática al estándar QTI/IMS. Este estándar es una extensión de XML, con etiquetas adaptadas a las necesidades del mismo. Su gran versatilidad hace que sea un lenguaje complejo y difícil de entender por un humano. Por ello, nace la idea de la creación de un **lenguaje de marcas** simplificado que pueda adaptarse a la necesidad de automatización de los exámenes y facilitar la tarea del profesor al redactar las preguntas.
Por otra parte, nos interesaba, una vez creado el traductor, tener la opción de portarlo a una aplicación web, para hacer su uso más accesible. La versión anterior de este proyecto[5] era una aplicación que solo corría bajo Windows. Ahora, con la aplicación web, se puede utilizar desde cualquier sistema operativo a través de un navegador web. También, como la filosofía de diseño desde el principio ha sido *mobile first*, la aplicación es completamente intuitiva para acceder a ella desde dispositivos móviles.

1.2 Objetivos

Cuatro son los grandes objetivos que se persiguen en este proyecto: Creación de un lenguaje de marcas sencillo para la escritura de exámenes, la creación de una aplicación web que permita traducir al instante y desde cualquier lugar los documentos generados en el lenguaje de marcas al estándar QTI/IMS, la posibilidad de almacenar los documentos previamente traducidos mediante la generación de usuarios y ofrecer a los usuarios herramientas sencillas para la generación automática de baterías de preguntas.
Para el primer objetivo, Antonio Martí Campoy, mi tutor, y Héctor Herráiz Muñoz habían desarrollado ya un lenguaje de marcas sencillo para su proyecto final de carrera.
Uno de los requisitos importantes para este objetivo era que el lenguaje creado fuera accesible, tanto para personas con un nivel de programación alto, como para docentes no relacionados con el área de la informática.
Para el segundo objetivo se han valorado varias posibilidades de creación de la aplicación web. Los requisitos que nos planteábamos a la hora de construir la web eran diversos, pero los más importantes eran: conseguir una interfaz amigable e intuitiva para los diversos tipos de usuarios, no requerir ningún tipo de instalación ni descarga por parte del cliente, realizar la traducción del lenguaje de marcas al estándar QTI/IMS y que funcionara en la mayoría de navegadores de forma óptima (hemos apuntado a Microsoft Edge, Opera, Mozila Firefox y Google Chrome, con esto se cubre el 99 % de cuota de mercado)[9]
En cuanto al tercer objetivo, teníamos dos aspectos importantes que necesitábamos cubrir. En primer lugar, la gestión de usuarios dentro de la aplicación, que el cliente pueda registrarse, tener su perfil personal, identificarse mediante su palabra clave y que estos datos sean guardados de forma segura (no accesibles fácilmente por terceros ni por los administradores de nuestra web) y persistente. Por otra parte, la necesidad de implementar un sistema de archivos para que el usuario, desde su perfil, pueda recuperar todos los documentos previamente traducidos, tanto en lenguaje de marcas como el archivo en estándar QTI/IMS.
Para facilitar el acceso a todo tipo de usuarios, se permite la traducción de documentos sin registro, pero se advierte de que éstos no serán almacenados, por lo que al descargarlos se eliminarán del servidor.
Para el cuarto objetivo, nos hemos planteado la posibilidad de ofrecer plantillas ya preparadas para que los usuarios simplemente tengan que introducir los datos básicos de las preguntas y puedan generar automáticamente una batería grande de preguntas en el lenguaje de marcas. Las posibilidades planteadas son, en principio, hacer uso de código en C y la combinación de correspondencia de los distintos procesadores de textos.
Así mismo, para construir el traductor de lenguaje de marcas al estándar QTI/IMS nos hemos apoyado tanto en la documentación oficial como en exámenes generados ya previamente por el generador de Sakai. El principal objetivo era usar esta aplicación para subir exámenes a la plataforma docente Sakai, por lo que apoyarnos en ejemplos de la misma era éxito asegurado.

1.3 Contenido de la memoria

La siguiente memoria se estructura en 9 grandes apartados.
En el primero, la introducción, se hace un breve repaso de qué y porqué se persiguen estos objetivos en el proyecto.
En el segundo, el contexto, se hace un breve repaso de las plataformas a las que nos hemos dirigido en la creación de la aplicación web. No se intenta dar una clase magistral sobre las mismas, sino unas pinceladas por si el lector desconociera totalmente alguna de las mismas, por ello animamos a cualquiera que quiera saber más a consultar la bibliografía de la memoria.
El tercer apartado detalla los requisitos y las funcionalidades propuestos para la aplicación.
En el cuarto apartado, se da una visión general del diseño propuesto para la aplicación. Para ello se utiliza la arquitectura en tres capas. A su vez, se detalla el lenguaje de marcas utilizado y se dan opciones de generación automática de las baterías de preguntas.
En el quinto apartado se da una descripción breve de las tecnologías utilizadas en el proyecto.
En el sexto apartado, partiendo del diseño, se detalla como se han implementado las funcionalidades de la aplicación web.
En el séptimo apartado, de implantación y pruebas, se describe como se ha desplegado el proyecto y como se han probado las funcionalidades del mismo.
En el octavo apartado, de ampliaciones futuras, se hace un repaso de lo conseguido en la aplicación y se dan un par de ideas de futuro para cualquier persona que quiera seguir desarrollando la misma.
Por último, se recogen las conclusiones del proyecto en el apartado 5 y se ofrece un manual de usuario como anexo, para que cualquier potencial usuario pueda consultarlo.

CAPÍTULO 2
Contexto

El proyecto se apoya en las plataformas que se nombran a continuación. Para introducir y poner en contexto este trabajo, nos ha parecido necesario hacer una breve descripción de qué son y para qué se usan estas plataformas. También repasaremos los aspectos más destacados de las mismas que nos han hecho decidirnos por su uso en este caso.

2.1 Lenguaje de marcas

Un **lenguaje de marcado** es una forma de codificar un texto, incorporando etiquetas o marcas que contienen información adicional para que sea interpretado por un programa informático. Pese a la creencia popular, no son lenguajes de programación, puesto que estos no contienen funciones aritméticas, variables ni métodos. Las marcas otorgan información adicional acerca de la estructura del texto o de su presentación.
A diferencia de un código de programación, los documentos con **lenguaje de marcado** son fácilmente interpretables por un usuario a poco que esté familiarizado con el lenguaje que se está utilizando.
Este tipo de lenguajes debe su popularidad al lenguaje HTML (*HyperText Markup Language*, por sus siglas en inglés). En el capítulo 3 se describe este lenguaje con más profundidad.
La madurez de este tipo de lenguajes tuvo su esplendor con la creación de XML (*eXtensible Markup Language*, por sus siglas en inglés). Este metalenguaje permite la creación de etiquetas adaptadas a las necesidades de cada usuario.
En la figura siguiente se puede observar la estructura de un documento XML[2] sencillo. En este caso, un catálogo con dos libros. Como comentábamos anteriormente, el **lenguaje de marcas** es muy intuitivo y fácil de interpretar por el usuario, incluso sin tener conocimiento del mismo, como se puede ver en la figura 2.1.

```xml
<?xml version="1.0"?>
<catalog>
  <book id="bk101">
    <author>Sallinger, J.D.</author>
    <title>El guardian entre el centeno</title>
    <genre>Novela</genre>
    <price>12.95</price>
    <publishdate>1951-07-16</publishdate>
    <description>La novela cuenta la historia de Holden Caulfield, un
        joven neoyorquino que acaba de ser expulsado de Pencey Prep, su
         escuela preparatoria.
    </description>
  </book>

  <book id="bk102">
    <author>Karl Marx</author>
    <title>El capital</title>
    <genre>Ensayo</genre>
    <price>Gratis</price>
    <publishdate>1867</publishdate>
    <description>Tratado de critica de la economia politica. Al mismo
        tiempo, ha sido tambien leido como una obra de filosofia, como
        un tratado de economia, o como un tratado politico sobre las
        relaciones de dominacion entre las clases, de un lado los
        proletarios y de otro los burgueses.
    </description>
  </book>
</catalog>
```

Figura 2.1: Ejemplo código XML

La relación de nuestro proyecto con los **lenguajes de marcas** es variada. Por una parte, nos apoyamos en el lenguaje HTML para la creación de la interfaz de la aplicación web. Por otra parte, utilizamos XML para la creación de los documentos en el formato estándar QTI/IMS. Este estándar es una extensión de XML, con etiquetas adaptadas a las necesidades del mismo. Su gran versatilidad hace que sea un lenguaje complejo y difícil de entender por un humano. Por ello, nace la idea de la creación de un **lenguaje de marcas** simplificado que pueda adaptarse a la necesidad de automatización de los exámenes y facilitar la tarea del profesor al redactar las preguntas.

2.1.1. Estándar QTI/IMS

La especificación de **QTI/IMS** es una extensión de XML y se trata de un estándar para permitir el intercambio de información entre sistemas que trabajan con plataformas docentes, que se centra en la definición de un modelo de preguntas y respuestas tipo test, rellenar huecos, múltiples opciones...
La idea era poder reflejar cualquier interacción o condición de un examen en un documento XML que pueda ser ejecutado por cualquier sistema que interprete el estándar **QTI**.
El estándar **QTI** fue desarrollado por el consorcio IMS Global. La necesidad de creación de un estándar surge para paliar la pérdida de preguntas cuando la tec-

nología cambia. Así pues, es posible cambiar de LMS (por ejemplo, de Sakai a Moodle) y seguir manteniendo las baterías de exámenes generados en un documento XML.
Lo que diferencia este lenguaje de marcas de otros más conocidos, como HTML, por ejemplo, es su complejidad. En la figura 2.2 se muestra una parte de un ejemplo de una pregunta sencilla del tipo *rellenar huecos* ya preparado para su publicación dentro de Poliformat (no se muestra el contenido completo, puesto que contiene más de mil líneas) en la que se puede observar que el entendimiento de las etiquetas es mucho más complicado que en ejemplo del catálogo de libros del apartado anterior.

```
<flow class="Block">
<response_str ident="FIB00" rcardinality="Ordered" rtiming="No">
<render_fib charset="ascii-us" columns="5" encoding="UTF_8"
fibtype="String" prompt="Box" rows="1" />
</response_str>
<material>
<mattext charset="ascii-us" texttype="text/plain" xml:space="default">
<![CDATA[<br><br>¿En qué posición de memoria se encuentra uch? (En
hexadecimal, con 8 bits y sin el 0x) 0x]]>
</mattext>
</material>
```

Figura 2.2: Ejemplo código en formato estándar QTI/IMS

Es de esta complejidad de donde surge la idea inicial del proyecto. Simplificar y automatizar la creación de baterías de preguntas.

2.2 Sistemas de gestión de aprendizaje

Los **LMS**[10](*Learning Management System*, por sus siglas en inglés. Sistemas de gestión de aprendizaje, en castellano) se crean con el objetivo de apoyar a la educación tradicional y a la educación a distancia. Son plataformas online[6] y, a pesar de lo que su nombre parece indicar, sirven de apoyo tanto al personal docente, valga la redundancia, como a los alumnos.
En la educación universitaria se han convertido en una herramienta imprescindible. Proporcionan ventajas a la comunidad universitaria, tales como:

- Compartir recursos educativos
- Sistemas de comunicación a distancia entre profesor y alumno (foros, chats, llamadas de voz, videollamadas...)
- Envío de tareas.
- Realización de exámenes a distancia.
- Consulta de resultados académicos.

Existen diversas **plataformas docentes**, pero, en el caso que nos ocupa dentro de la *Universitat Politècnica de València*, se utiliza la **plataforma docente** Sakai, cuyo uso se detalla en los siguientes apartados.

2.2.1. Sakai

Sakai[8] es un *Learning Management System* de código abierto. Ellos mismos se definen como una alternativa a los LMS comerciales, que tiene su origen en la educación superior para satisfacer las demandas dinámicas de la comunidad académica global.
El proyecto **Sakai** nace en 2005 como propiedad de la *Sakai Foundation*, aunque actualmente **Sakai** es un proyecto propiedad de la *Fundación Apereo*, una organización formada por la fusión de la *Sakai Foundation* y *Jasig*. La *Fundación Apereo* promueve la construcción de comunidades entre individuos, instituciones académicas, organizaciones sin fines de lucro y organizaciones comerciales y proporciona a sus miembros un marco institucional para sus proyectos.
El proyecto está en continuo desarrollo y proceso de mejora, valorando la participación de todos sus contribuidores, desde docentes hasta desarrolladores de las diferentes instituciones que adoptan el sistema (o revisiones del mismo, como es el caso de la UPV con Poliformat), trabajando juntos para transformar las ideas y quejas que van surgiendo en una realidad en las nuevas versiones del proyecto.
Uno de los pilares fundamentales en los que pretende apoyarse **Sakai** es la educación colaborativa. Sus funcionalidades, que ayudan a perseguir este objetivo, incluyen un lector de noticias RSS, distribución de material docente, calificaciones, foros de discusión, chat en vivo, tareas en línea, gestión de trabajos, etc. Al ser un proyecto de código abierto, esta sujeto a constante evolución y múltiples autores han desarrollado funcionalidades externas que se pueden integrar en la aplicación.
La herramienta permite la creación de un sitio web para cada curso y gestionar de forma eficiente los roles de los usuarios, estableciendo diferentes permisos para cada uno de ellos. Un ejemplo muy simple sería a la hora de entregar una tarea. Los alumnos pertenecerían a un rol de usuario, que solo podría subir su propia tarea y consultar su calificación. Por su parte, el profesor tendría un rol con más permisos desde el que podría consultar todas las tareas y otorgar calificaciones a todas ellas.
Todo esto es algo que parece sencillo, y este es precisamente el punto fuerte de **Sakai** frente a otros LMS comerciales, la sencillez de manejo. También destaca su estabilidad, pudiendo albergar a una gran cantidad de usuarios simultáneos sin ver perjudicado su funcionamiento.

2.2.2. Poliformat

Poliformat es la plataforma docente que utiliza actualmente la *Universitat Politècnica de València*. Es una adaptación del LMS Sakai.
Entre sus características se pueden encontrar diversos apartados, entre los cuales destacamos, por ser los más utilizados en el transcurso de una asignatura:

- ***Recursos***: Sirve para la compartición de recursos educativos entre el personal docente y los alumnos.
- ***Exámenes***: Esta es la parte más importante en nuestro proyecto, pues aquí es donde el profesor sube los exámenes (que puede haber generado con nuestra aplicación web) y los distintos alumnos del curso deben resolverlos.
- ***Foros y Chats***: Se utilizan para la comunicación directa entre el personal docente y los alumnos o entre los propios alumnos.
- ***Anuncios***: El profesor o el resto de personal del departamento anuncian fechas importantes, como la entrega de alguna tarea, un examen, un día festivo, etc.

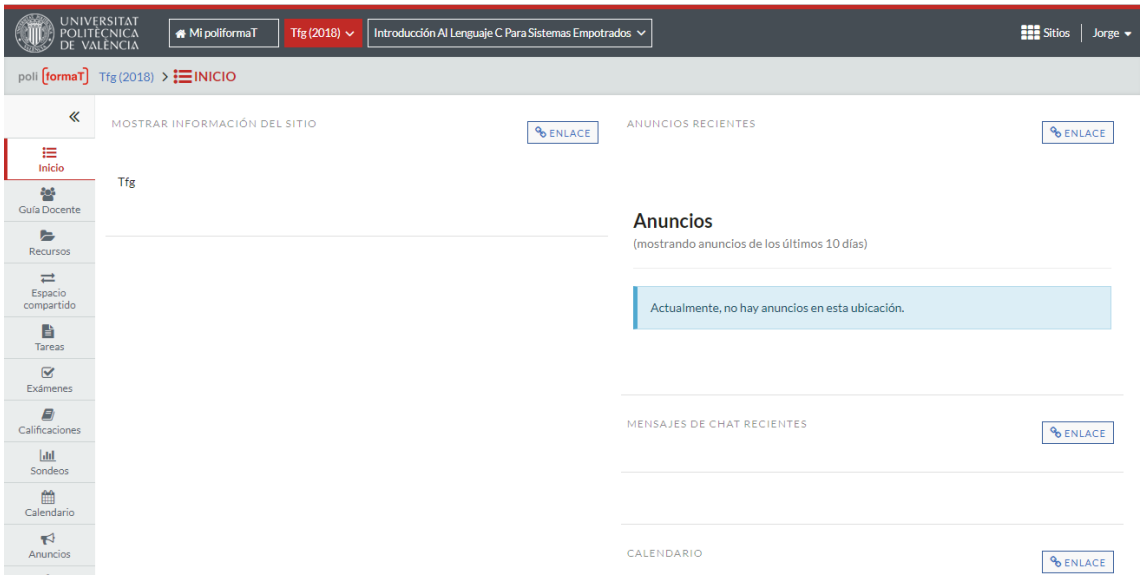


Figura 2.3: Interfaz Poliformat

Además de estos apartados existen muchos otros, que según la asignatura tienen mayor o menor uso, tales como: Espacio compartido (una nube personal del alumno que el profesor puede consultar), Sondeos *(para preguntar la opinión del alumnado sobre algún aspecto)*, Tareas *(para la entrega de los distintos trabajos)*, Calificaciones *(para consultar los resultados académicos)*, etc.
Con respecto a la sección de Exámenes, aquí es donde apunta nuestra aplicación web. El personal docente, a la hora de la creación de los exámenes, tiene varios métodos para hacerlo. Se puede subir un archivo en formato estándar QTI/IMS, conteniendo todas las preguntas, o bien introduciendo una a una todas las preguntas mediante el asistente web. Nuestra aplicación apunta a la primera opción de creación de exámenes y trata en parte de suplir alguna de las carencias del asistente web.
Como escribir un archivo en el formato estándar QTI es una tarea bastante tediosa, se creó el lenguaje de marcas que se utiliza en la aplicación web, ya no solo para automatizar el proceso de crear una gran batería de preguntas, sino también para facilitar la creación de cada pregunta individualmente.
Si uno busca recursos para facilitar la tarea de crear preguntas en formato QTI puede encontrar diversos editores online del mismo. Pero a la hora de su uso

presentan grandes inconvenientes, tales como: Algunos son de pago, no tienen las funcionalidades deseadas o no permiten el nivel de automatización deseado.

CAPÍTULO 3
Requisitos

Dentro de este apartado se van a definir los requisitos que debe cumplir la aplicación web. Para la definición de los mismos se va a seguir el estándar IEEE 830/1998, en el cual se define la estructura a seguir para el desarrollo de un documento de requisitos estándar.

3.1 Introducción

3.1.1. Propósito

El objetivo de esta especificación es definir de manera exacta las funciones que la aplicación puede llevar a cabo. También se mencionará las funcionalidades que no son posibles.

3.1.2. Ámbito del sistema

El objetivo del proyecto es desarrollar una aplicación web que permita traducir un sencillo lenguaje de marcas al estándar QTI/IMS para la creación de exámenes en Poliformat.
La aplicación permite a un docente crear baterías de preguntas de forma sencilla y automatizada.
La aplicación, además, permite la recuperación de los archivos que se hayan tratado con anterioridad en la web, mediante la creación de un usuario propio para cada cliente.
El principal beneficio de la aplicación para su uso docente es hacer la tarea de creación de exámenes más fácil, intuitiva y rápida.

3.1.3. Definiciones, acrónimos y abreviaturas

- XML: es un tipo de documento utilizado para representar información mediante el etiquetado de lenguaje de marcas.
- QTI/IMS: Es un lenguaje de marcas creado como estándar para la creación de exámenes en los diferentes sistemas de gestión de aprendizaje.

3.1.4. Referencias

IEEE Recommended Practice for Software Requeriments Specification. ANSI/IEEE 830/1998.

3.1.5. Visión general del documento

El documento consta de tres secciones: Introducción, descripción general y requisitos específicos. En la introducción se ha proporcionado una visión general de la aplicación y el ámbito para el cual fue pensada. En la descripción general se describen los factores que afectan a la aplicación y a sus requisitos. Por último, en el apartado de requisitos específicos, se entra en detalle sobre los mismos.

3.2 Descripción general

3.2.1. Perspectiva del producto

La aplicación desarrollada requerirá del uso de un navegador web para acceder a la misma. Esta es la única herramienta a través de la cual se podrá acceder a la web. Se permite el acceso desde cualquier dispositivo.

3.2.2. Funciones del producto

- Traducción de archivos de lenguaje de marcas al estándar QTI/IMS
- Generación de un documento XML válido para la herramienta Exámenes en Poliformat
- Gestión de errores para traducciones fallidas.
- Permitir a un usuario registrarse en la página web.
- Almacenar un historial de archivos previamente traducidos.
- Conseguir un diseño amigable y funcional en la aplicación.
- Desarrollar un sistema de recuperación de contraseñas vía email.
- Posibilitar al usuario el cambio de contraseña.
- Integrar un editor online para los archivos.
- Permitir al usuario darse de baja de la aplicación, con el consiguiente borrado de datos y archivos.
- Garantizar la seguridad de los datos de los usuarios.

3.2.3. Características de los usuarios

La aplicación está pensada para ser usada por docentes, pero puede tener acceso cualquier usuario que esté interesado en las funcionalidades de la misma. Por lo tanto, se entiende que el usuario medio tiene unos mínimos conocimientos de las tecnologías utilizadas por Poliformat para la creación de exámenes.

3.2.4. Restricciones

Las principales restricciones que nos definimos para el proyecto son:

- La aplicación debe funcionar correctamente en los navegadores web Microsoft Edge, Google Chrome, Mozilla Firefox y Opera.
- La aplicación debe visualizarse correctamente en pantallas de 1920x1080 y en dispositivos móviles con resolución 720x1280.

3.2.5. Suposiciones y dependencias

La aplicación se ha diseñado para funcionar en los navegadores web antes mencionados, por lo que utilizando otros, pueden surgir problemas de visualización que no han sido comprobados. De todas formas, el código HTML5 cumple los estándares del consorcio W3C, por lo que cualquier navegador que los cumpla debería poder mostrar correctamente la aplicación. Al ser una aplicación web alojada en servidores, su rendimiento puede variar por factores externos, como la velocidad y disponibilidad del servidor que se esté utilizando. La web se ha diseñado pensando en las resoluciones de pantalla más extendidas en el mercado actual, aunque también se han hecho pruebas de visualización en otros estándares. Como en el caso de los navegadores web, puede haber fallos de visualización al usar resoluciones no tan comunes.

3.2.6. Requisitos futuros

En el futuro sería interesante añadir otros tipos de preguntas al traductor de lenguaje de marcas, tales como encuesta o ordenación, por citar algunas.

3.3 Requisitos específicos

3.3.1. Interfaces

Para el desarrollo de la aplicación se ha diseñado una interfaz externa, con partes ocultas para usuarios no registrados que están disponibles para los usuarios con registro.

3.3.2. Funciones

A continuación se van a explicar todas las funciones que debe llevar a cabo la aplicación. Para esto se va a seguir el estándar IEEE 830/1998, para describirlas con un nivel de detalle suficiente para su posterior implementación.
Las funciones se dividen entre usuarios registrados y no registrados, aunque algunas de ellas se comparten entre ambos.
Primero se definen las asociadas a un usuario no registrado:

Número	1
Requisito	Traducción de archivos de marcas a QTI/IMS.
Prioridad	Alta
Descripción	La aplicación debe ser capaz de aceptar la subida de un archivo en el lenguaje de marcas creado y traducirlo a un archivo en el formato estándar QTI/IMS, listo para ser aceptado como examen por las plataformas de aprendizaje.

Número	2
Requisito	Registrarse
Prioridad	Alta
Descripción	Gestionar los datos del usuario y almacenarlos en una base de datos para la identificación de los mismos y el almacenaje de su historial de archivos

Número	3
Requisito	Editor online
Prioridad	Baja
Descripción	Dar la posibilidad al usuario de escribir preguntas usando el lenguaje de marcas directamente en la aplicación web, para que no tenga la necesidad de generar un archivo de texto por su cuenta

Número	4
Requisito	Gestión de errores
Prioridad	Alta
Descripción	Avisar de errores en la generación del examen. El usuario así podrá corregirlos antes de encontrarse que el examen no es correctamente interpretado por la plataforma de aprendizaje

A continuación se describen los requisitos para usuarios registrados:

Número	5
Requisito	Iniciar sesión
Prioridad	Alta
Descripción	Lectura de los datos del usuario y comparación con los almacenados en la base de datos para que el usuario pueda acceder a las funciones restringidas para usuarios registrados

Número	6
Requisito	Recuperar contraseña
Prioridad	Media
Descripción	Si el usuario olvida sus datos de usuario, puede pedirle a la aplicación que le envíe por correo una contraseña alternativa para iniciar sesión

Número	7
Requisito	Modificar contraseña
Prioridad	Media
Descripción	Permitir al usuario cambiar su contraseña, pensando sobretodo en el caso que la olvide y la aplicación le genere una nueva

Número	8
Requisito	Darse de baja
Prioridad	Baja
Descripción	El usuario podrá borrar sus datos de la BBDD y todo su historial de archivos

Número	9
Requisito	Historial de archivos
Prioridad	Alta
Descripción	El usuario podrá recuperar tanto el fichero fuente como el fichero traducido de todos los exámenes que haya generado en la aplicación

Número	10
Requisito	Borrado de archivos
Prioridad	Baja
Descripción	Permitir al usuario borrar los exámenes que ya no vaya a necesitar más

3.3.3. Requisitos de rendimiento

Los requisitos de rendimiento, como en cualquier aplicación web dependen, sobre todo, del servidor en el que esté alojada y de la capacidad de la conexión

del usuario que haga uso de la misma. El sistema no tiene que soportar una gran carga de trabajo, y los archivos que se tratan de media ocupan 1MB de espacio, por lo que, hoy en día, teniendo en cuenta la capacidad siempre creciente de los espacios de almacenamiento y las velocidades de las conexiones, no es una exigencia demasiado grande. Aún así, para mantener cierto control sobre el uso, el tamaño máximo de archivo permitido es de 8MB. Esto, en términos prácticos, supone más de 4000 preguntas contenidas en un archivo, por lo que estimamos es un tamaño más que suficiente.

3.3.4. Restricciones de diseño

El diseño de la interfaz de la aplicación está limitado a su portabilidad. Es decir, el principal objetivo ha sido adaptar la aplicación a todo tipo de dispositivos y resoluciones, por lo que se ha prescindido del uso de animaciones o etiquetas específicas de HTML5 que no son interpretadas igual por los diferentes navegadores. Se han minimizado así los errores de visualización posibles.

3.3.5. Atributos del sistema

- Fiabilidad: La aplicación debe funcionar y ofrecer todas las funciones especificadas en los requisitos.
- Seguridad: Se lleva a cabo un control de acceso a la aplicación mediante la gestión de usuarios. La política de contraseñas también es otro aspecto de la seguridad.
- Portabilidad: La aplicación se puede instalar en cualquier servidor ASP.NET y es accesible desde cualquier dispositivo que haga uso de un navegador.

CAPÍTULO 4
Diseño

4.1 Arquitectura de tres capas

En este proyecto se ha optado por una arquitectura en tres capas que separa nuestra aplicación en tres capas diferenciadas: capa de presentación, capa de negocio y capa de acceso a datos. En la figura 4.1 se puede observar como interactúan las tres capas entre sí:

Figura 4.1: Arquitectura de tres capas

4.1.1. Capa de presentación

La capa de presentación incluye el código necesario para mostrar la aplicación web, de forma que la información se muestre gráficamente para poder realizar las funciones que se esperan.
Para diseñar la interfaz se nos abrían dos posibilidades. La primera y la elegida finalmente, era el diseño en HTML y CSS como una web estándar. La segunda opción era hacer uso de los controladores propios de ASP. La decisión fue fácil y, aunque el abanico de opciones que nos ofrecía ASP podía compararse con el de HTML, lo cierto es que, pensando tanto en nuestros conocimientos anteriores, que ya habíamos trabajado con HTML, y en los conocimientos de otras personas que se podrían encargar en un futuro de la web (es más usual encontrar gente experta en HTML que en controladores ASP), nos decantamos por combinar HTML y CSS para el diseño.

Para el diseño primitivo de la interfaz hemos dividido cada pantalla en dos partes para mantener la coherencia en el diseño. En la parte superior se encuentra el *header*, que va a seguir la misma estructura en todas las pantallas. Allí se encuentran el logo, los botones para las funciones principales, el menú y una breve descripción de la funcionalidad que ofrece la pantalla en concreto. En la parte inferior, llamada *footer*, que no es necesaria en las pantallas simples como login o registro, entre otras, se encuentra la funcionalidad principal de la aplicación. Por citar dos ejemplos, en la pantalla principal se encuentra el formulario de subida de archivos y en el perfil personal el historial de archivos.
En la figura 4.2 se muestra el primer prototipo de la pantalla principal. El resto de ellas siguen la misma estructura y se puede ver la versión final en la implementación.

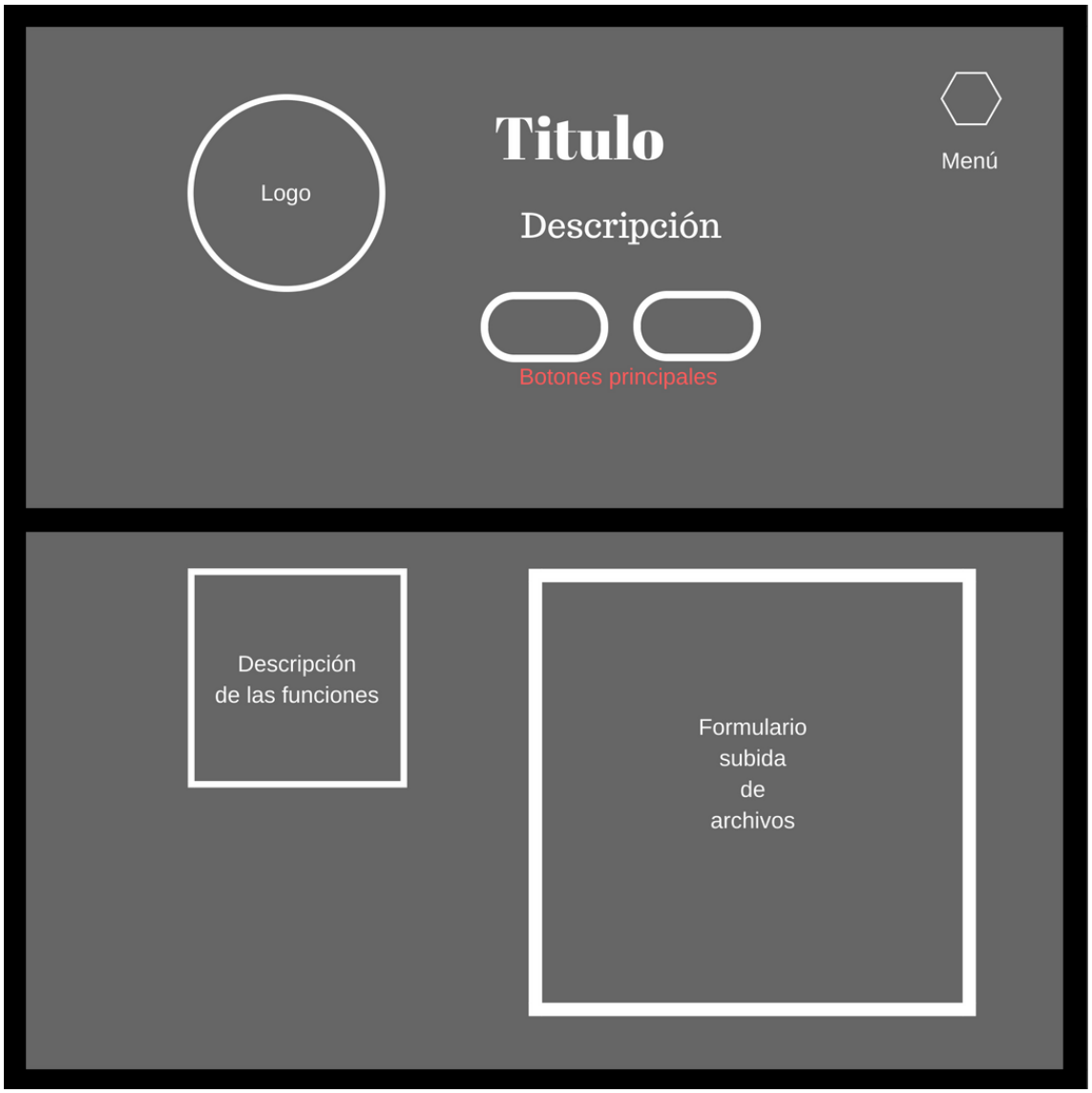


Figura 4.2: Prototipo de la pantalla principal

Se puede destacar que la interfaz de la aplicación está totalmente pensada para ser usada en todo tipo de dispositivos, esto se controla mediante los *media query* y la web se adapta al dispositivo que estemos utilizando.

4.1.2. Capa de negocio

La capa de negocio o capa intermedia funciona como un puente entre los datos y la interfaz y también se encarga de resolver las peticiones de la capa de presentación para mostrar los datos correctos. En esta capa se encuentran todos los códigos asociados a los archivos de ASP.NET. En la figura 4.3 se muestra un prototipo del diseño inicial que se planteó para la aplicación. En él se pueden ver las interacciones entre los diferentes archivos de código y dónde se producen las llamadas a la base de datos. Como ya se ha dicho, este diseño es una primera aproximación y todo se detallará posteriormente, en la fase de implementación final.

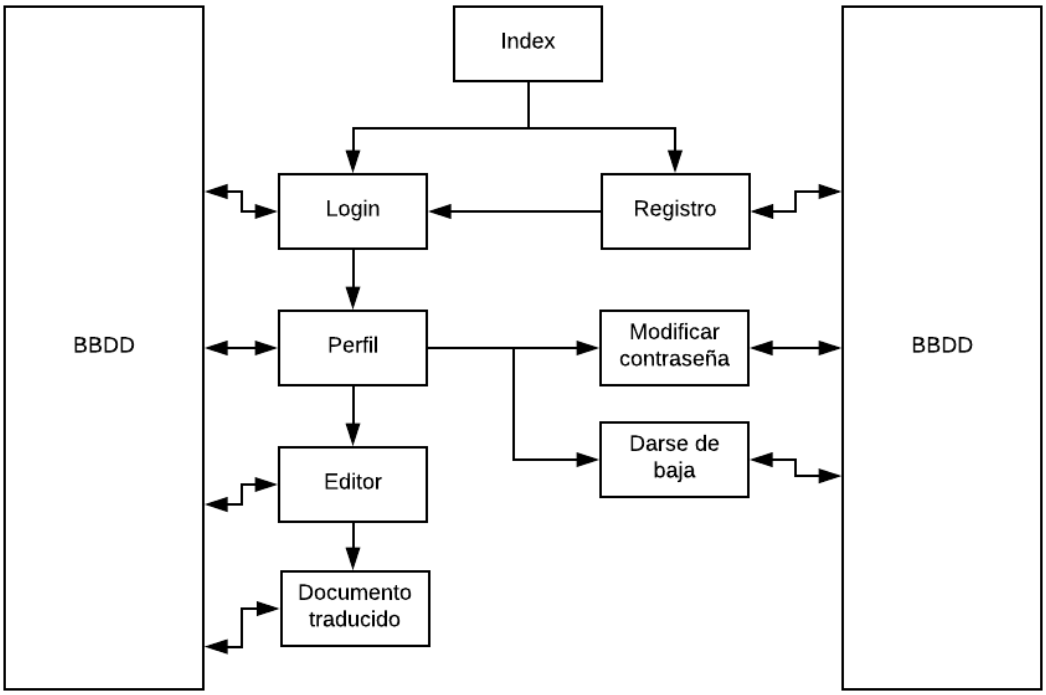


Figura 4.3: Prototipo de la capa de negocio

4.1.3. Capa de acceso a datos

Para el diseño de la base de datos se plantearon diversos diseños, aunque al final nos decidimos por el que, a juicio nuestro, otorga un mayor rendimiento y evita la redundancia de los datos. El diseño elegido responde en cuestión de segundos, incluso cuando fuerzas al máximo a la aplicación (se han realizado pruebas creando una gran cantidad de usuario y subiendo muchos archivos cada uno de ellos y la respuesta sigue siendo inmediata).
El diseño final es una base de datos relacional que consta de dos tablas: *users* y *files*.
La primera almacena los datos básicos de los usuarios y se compone de 4 columnas: *id*, *user*, *pass* y *email*. En la primera columna se introduce un identificador

único de cada usuario, el cual es autoincremental, por lo que no hace falta control por parte del administrador. En la segunda columna, el nombre de usuario escogido por el cliente a la hora de registrarse (esta columna tiene las propiedades de ser única por lo que no se podrán tener dos nombres de usuario idénticos, aunque esta propiedad también la controla la aplicación antes de envíar el *query* a la base de datos). La tercera columna contiene el hash de la contraseña que el usuario haya introducido. Y, por último, se introduce el email del usuario. Esta última columna tiene las mismas propiedades de no duplicidad que la columna *user*.
La segunda tabla contiene una relación de los archivos que ha subido cada usuario. En un principio se planteó la posibilidad de crear una tabla individual para cada usuario, pero tanto el registro como el borrado se hacían demasiado pesados, por lo que se decidió introducir toda la relación de archivos en una misma tabla.
Esta tabla contiene tres columnas: *id*, *user* y *file*. En la primera columna se introduce un identificador único de cada usuario, el cual es autoincremental, por lo que no hace falta control por parte del administrador. En la segunda se introduce el usuario que ha subido el archivo que se escribe en la tercera columna, file. La columna *user* es clave foránea de la columna con el mismo nombre en la tabla users, por lo que a la hora de introducir un archivo debe estar correctamente vinculado a un usuario existente. También se introdujo la característica de borrado en cascada, por lo que el borrado del usuario, borra todos los datos relacionados con él en la tabla *files*.

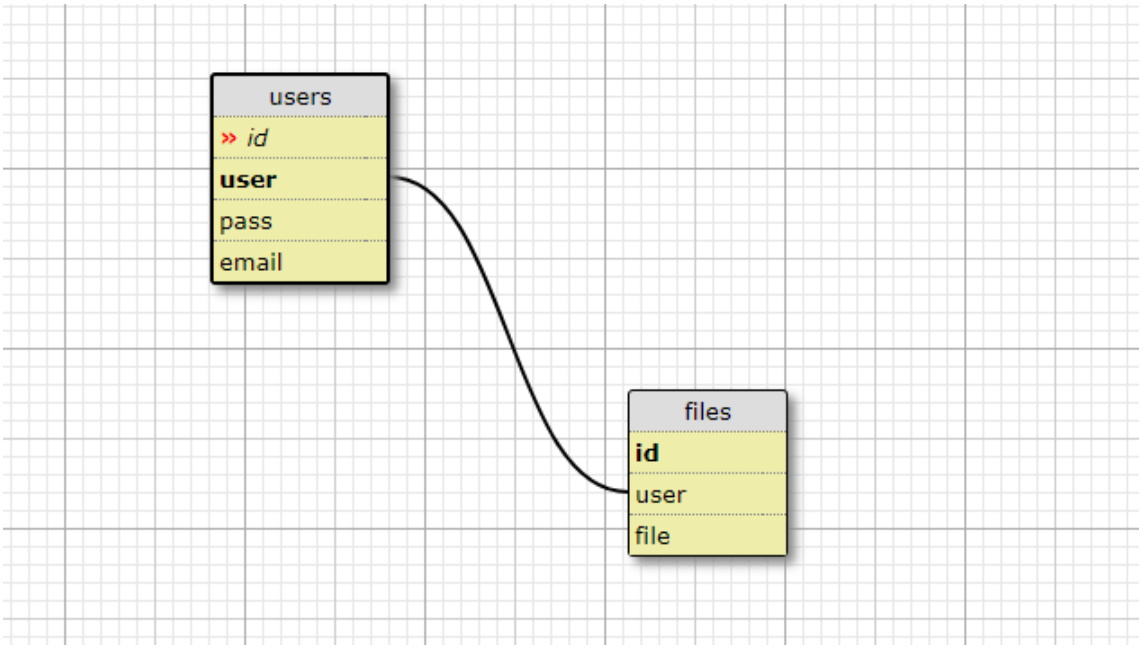


Figura 4.4: Diseño base de datos

Como se puede observar en la figura 4.4, el resultado del diseño es intuitivo y fácil de interpretar, además, la sencillez del mismo, hace que no sea necesario un administrador especializado, pues alguien con unos conocimientos mínimos de SQL puede interpretar y realizar modificaciones en el mismo si se desea.

4.2 Lenguaje de marcas

Pese a que el lenguaje de marcas se hereda de un proyecto anterior, es conveniente explicar su funcionamiento, ya que es una parte esencial para el uso de nuestra aplicación, cuya funcionalidad principal es traducir documentos desde este lenguaje de marcas al formato estándar QTI/IMS.
El lenguaje utilizado minimiza el número de marcas necesarias para crear preguntas identificables por el traductor. Las marcas de inicio y fin de pregunta contienen la letra Q, seguida del tipo de pregunta. La aplicación es capaz de manejar tres tipos de preguntas:

- **<*QN*>**: Preguntas de tipo numérico.
- **<*QF*>**: Preguntas de tipo rellenar huecos.
- **<*QM*>**: Preguntas de selección múltiple

Obligatoriamente las preguntas comenzarán con esta etiqueta y se cerrarán con la etiqueta **</*Q*>**, independientemente del tipo de pregunta que se maneje.
Además, se permite añadir la puntuación asociada a la pregunta haciendo uso de la marca **<M*x*>**, donde x es la puntuación. Esta marca ha de situarse inmediatamente después de la marca que indica el tipo de pregunta. Ademas, si se ha indicado una puntuación asociada a la pregunta, se puede indicar también la puntuación a restar si se falla la pregunta, haciendo uso de la marca **<MN*y*>**, donde y es la puntuación a restar.

4.2.1. Respuesta numérica

Además de las marcas especificadas en el apartado anterior, que son comunes para cualquier tipo de respuesta, en las respuestas de tipo numérico se indica el resultado buscado entre corchetes. En la figura 4.5 se muestra un ejemplo de cómo sería una pregunta de tipo numérico válida en la aplicación.

```
1 <QN><M1,0><MN0,33>Responde a la siguiente cuestion: 1+1={2} </Q>
```

Figura 4.5: Ejemplo pregunta numérica en el lenguaje de marcas

Esta pregunta pide al usuario que realiza el examen que introduzca el resultado correcto, en este caso, el 2. También tiene una puntuación de 1 punto en el caso de que aciertes y resta 0,33 puntos en el caso de que la respuesta sea errónea. En Poliformat se visualizaría de la forma que se ve en la figura 4.6:

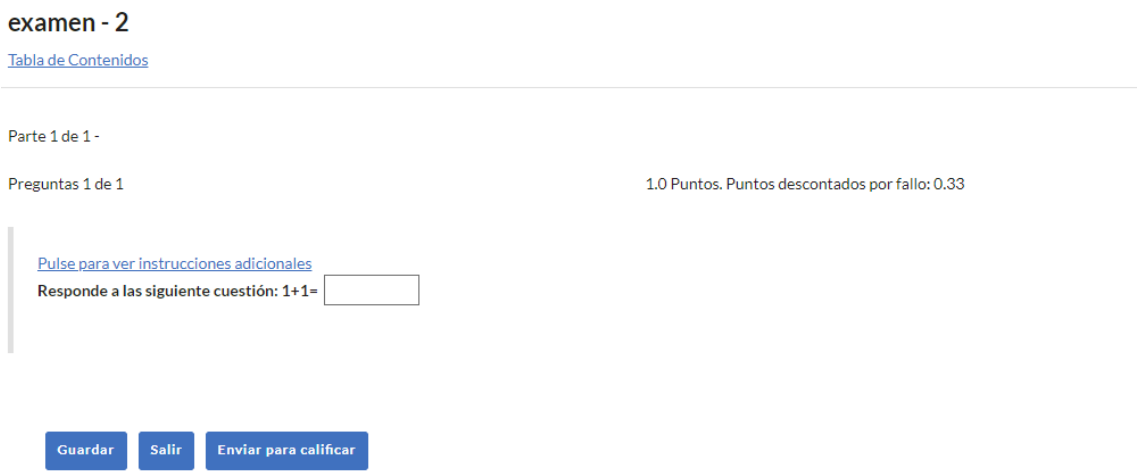


Figura 4.6: Ejemplo de pregunta tipo numérico

Además, en el lenguaje de marcas creado, a diferencia del estándar que usa Poliformat, se permite la posibilidad de añadir varios huecos de respuesta en la misma pregunta.

4.2.2. Rellenar huecos

Esta pregunta sigue exactamente la estructura de la pregunta de tipo numérico, donde la respuesta correcta se introduce entre corchetes, como se puede ver en la figura 4.7:

```
1 <QF><M1,0><MN0,33>Sigue la serie: uno {dos} tres cuatro </Q>
```

Figura 4.7: Ejemplo pregunta tipo rellenar huecos en el lenguaje de marcas

Esta pregunta pide al usuario que realiza el examen que introduzca el resultado correcto, en este caso, *dos*. También tiene una puntuación de 1 punto en el caso de que aciertes y resta 0,33 puntos en el caso de que la respuesta sea errónea. En Poliformat se visualizaría de la forma que se ve en la figura 4.8:

Figura 4.8: Ejemplo de pregunta tipo rellenar huecos

Además, en el lenguaje de marcas creado, a diferencia del estándar que usa Poliformat, se permite la posibilidad de añadir varios huecos de respuesta en la misma pregunta.

4.2.3. Selección múltiple

Para este tipo de preguntas se introducen dos marcas nueva, <op>y <rc>. La primera se utilizará antes de cada opción de respuesta. Si se introduce la marca <rc>justo después de la marca <op>, indicará que esa respuesta es la correcta. Al ser una pregunta de selección múltiple es posible introducir más de una respuesta correcta.
Como se puede ver en la figura 4.9:

```
<QM><M1,0><MN0,33>
2x6/3
<op>3
<op><rc>4
<op>5
</Q>
```

Figura 4.9: Ejemplo pregunta de selección múltiple en el lenguaje de marcas

Esta pregunta pide al usuario que realiza el examen que seleccione la respuesta correcta, en este caso, 4. También tiene una puntuación de 1 punto en el caso de que aciertes y resta 0,33 puntos en el caso de que la respuesta sea errónea. En Poliformat se visualizaría de la forma que se ve en la figura 4.10:

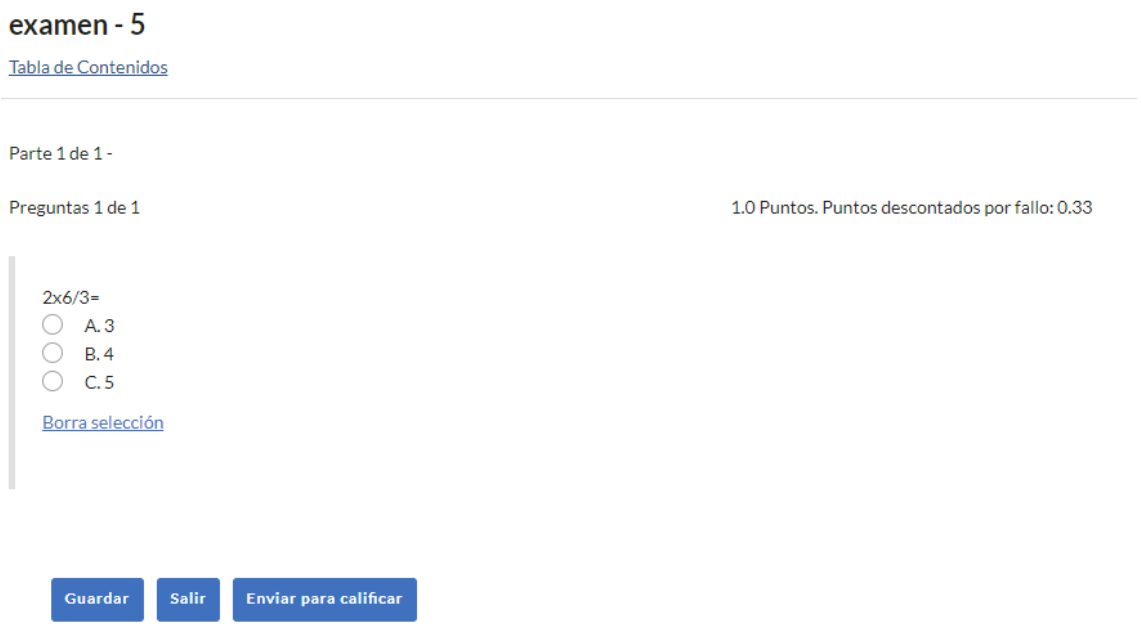


Figura 4.10: Ejemplo de pregunta tipo selección múltiple

4.2.4. Formato

Aparte de las marcas para indicar las preguntas y las respuestas, se permite formatear el texto haciendo uso de las siguientes marcas:

- ***Salto de línea:***
- ***Texto en negrita:*** <B></B>
- ***Texto en cursiva:*** <I></I>
- ***Texto subrayado:*** <U></U>
- ***Subíndice:***
- ***Superíndice:***
- ***Enlaces:***

4.2.5. Opciones de generación automática

Una de las motivaciones principales del proyecto es facilitar la tarea de los docentes a la hora de crear baterías de preguntas para exámenes en las plataformas docentes. Por ello, además de la aplicación web, pensamos que es una buena idea dar opciones de generación automática de baterías de preguntas. Hemos desarrollado tres tipos de solución: un asistente para introducir preguntas una a una; un ejemplo de programa de consola para generar múltiples preguntas de forma automática; y un ejemplo de cómo utilizar herramientas ofimáticas (hojas de cálculo y procesadores de texto) para generar múltiples preguntas de forma automática.

Asistente en Java

Con el programa de la figura 4.11, escrito en lenguaje JAVA, se pueden generar fácilmente las preguntas escribiendo los datos básicos. Una de las ventajas de este asistente es que se introducen de forma más rápida que en el asistente de Poliformat y, además, minimizamos la interacción con Poliformat, al tener que subir las preguntas solo una vez.

```
import java.io.*;
import java.util.*;
public class Asistente
{
  public static void main (String args[]){
    String op1="",op2="",op3="",op4="",respuesta="";
    String res="";
    boolean sigue=true;
    while(sigue){
      System.out.println("Tipo de pregunta: (M,F,N)");
      Scanner scanner = new Scanner(System.in);
      String tipo = "<Q"+scanner.nextLine()+">";
      System.out.println("Puntuacion:");
      scanner = new Scanner(System.in);
      String puntuacion ="<M"+ scanner.nextLine()+">";
      System.out.println("Puntos a restar en fallo:");
      scanner = new Scanner(System.in);
      String resta = "<MN"+scanner.nextLine()+">";
      System.out.println("Enunciado:");
      scanner = new Scanner(System.in);
      String enunciado = scanner.nextLine();
      if (tipo.equals("<QN>")){
        System.out.println("Respuesta: (solo tipo numerico)");
        scanner = new Scanner(System.in);
        respuesta = " {"+scanner.nextLine()+"}";
        res=res+tipo+puntuacion+resta+enunciado+respuesta+"</Q>";
      }
      else if (tipo.equals("<QF>")){
        System.out.println("Respuesta: (texto)");
        scanner = new Scanner(System.in);
        respuesta = " {"+scanner.nextLine()+"}";
        res=res+tipo+puntuacion+resta+enunciado+respuesta+"</Q>";
      }
      else if (tipo.equals("<QM>")){
        System.out.println("Primera opcion:");
        scanner = new Scanner(System.in);
        op1 = scanner.nextLine();
        System.out.println("Segunda opcion:");
        scanner = new Scanner(System.in);
        op2 = scanner.nextLine();
        System.out.println("Tercera opcion:");
        scanner = new Scanner(System.in);
        op3 = scanner.nextLine();
        System.out.println("Cuarta opcion:");
        scanner = new Scanner(System.in);
        op4 = scanner.nextLine();
        System.out.println("Cual es la correcta?(1,2,3 o 4):");
        scanner = new Scanner(System.in);
        int correcta = scanner.nextInt();
        switch(correcta){
```

```
51              case 1:op1="<op><rc>"+op1;break;
52              case 2:op2="<op><rc>"+op2;break;
53              case 3:op3="<op><rc>"+op3;break;
54              case 4:op4="<op><rc>"+op4;break;
55            }
56            if(!op1.startsWith("<"))op1="<op>"+op1;
57            if(!op2.startsWith("<"))op2="<op>"+op2;
58            if(!op3.startsWith("<"))op3="<op>"+op3;
59            if(!op4.startsWith("<"))op4="<op>"+op4;
60            res=res+tipo+puntuacion+resta+enunciado+op1+op2+op3+op4+"</Q>";
61          }
62          System.out.println("Quieres generar otra pregunta? (S, N)");
63          scanner = new Scanner(System.in);
64          String otra = scanner.nextLine();
65          if(otra.equals("N"))sigue=false;
66        }
67        System.out.println("Aqui tienes el codigo para las preguntas:");
68        System.out.println(res);
69      }
70  }
```

Como se puede observar en el código, la aplicación pide al usuario, en este orden, los siguientes datos: Tipo de pregunta, puntuación, puntuación a restar si hay fallo, enunciado y, dependiendo del tipo, la respuesta directa o las opciones de respuesta (para selección múltiple). A continuación se le pregunta si desea introducir otra pregunta o obtener el código.
Con el código que se genera, el usuario puede introducirlo directamente en el editor online de la aplicación web y obtener el xml en formato estándar QTI/IMS, o bien, puede copiarlo a un archivo .txt y subirlo a la aplicación para su traducción.
Como se puede observar en la siguiente imagen, el usuario que haga uso del programa, no debe tener ningún tipo de conocimientos en programación. Incluso, aunque no es recomendable, ya que conocer los tipos de preguntas es importante para saber qué se está introduciendo exactamente, ni siquiera debe entender como funciona el lenguaje de marcas que se ha creado para el proyecto.
Por supuesto, un usuario que conozca mejor el lenguaje de marcas, puede hacer uso de más opciones dentro de este, como las etiquetas para dar formato al texto, pero esto nos sigue pareciendo una buena opción y, sobretodo, cómoda, incluso para usuarios expertos.
En la siguiente imagen se ejemplifica la sencillez del programa mediante una captura de una ejecución en la que se introducen una pregunta de selección múltiple y una pregunta de rellenar huecos.

```
Tipo de pregunta: (M,F,N)
M
Puntuación:
1
Puntos a restar en fallo:
0,33
Enunciado:
1+1?
Primera opción:
1
Segunda opción:
2
Tercera opción:
3
Cuarta opción:
4
¿Cual es la correcta?(1,2,3 o 4):
2
Quieres generar otra pregunta? (S, N)
S
Tipo de pregunta: (M,F,N)
F
Puntuación:
1
Puntos a restar en fallo:
0,33
Enunciado:
¿En qué lenguaje trabaja ASP.NET?
Respuesta: (texto)
C
Quieres generar otra pregunta? (S, N)
N
Aquí tienes el código para las preguntas:

<QM><M1><MN0,33>1+1?<op>1<op><rc>2<op>3<op>4</Q><QF><M1><MN0,33>¿En qué lenguaje trabaja ASP.NET? {C}</Q>
```

Figura 4.11: Ejemplo del asistente de creación de preguntas

Baterías de preguntas en JAVA

Esta opción está pensada sobretodo para preguntas que sean fáciles de generar automáticamente, por ejemplo, una traducción de un número a binario, o una multiplicación con distintos operandos, etc. Para ejemplificar el funcionamiento de este generador, vamos a suponer que queremos examinar a todos los alumnos sobre cuantos bytes ocupa un programa en memoria. Para ello se define el tamaño de cada instrucción en memoria y se aleatorizan las instrucciones para cada examen. Después se suman las respuestas. Este ejemplo se puede hacer para cualquier tipo de pregunta que se nos ocurra, y que cumpla las condiciones que cumple esta, es decir, que para cada valor de la pregunta haya una respuesta fija, por ejemplo, conversiones entre unidades, entre sistemas numéricos, etc.

```
1  import java.io.*;
2  import java.util.*;
3  public class BateriaDePreguntas
4  {
5    public static void main (String args[]){
6      String[] instrucciones = {"INC RA", "DEC RA", "MOVEI RA, 0x10", "
         MOVE RA, 0x24", "MOVE 0x24, RA", "ADDI RA, 0xFF", "ADD RA, 0xA1
         ", "PUSH RA", "POP RA", "COMPAREI RA, 0x00"};
7      int[] bytesocupados=  {1,1,2,2,2,2,2,1,1,2};
8      int respuesta=0;
9      int aux=1;
10     int w=0;
11
12     //VARIABLES DE CONTROL
13     int NUMPREGUNTAS=100;
14     String codigo="";
```

```
15      String tipopregunta="N";
16      double puntuacion=1;
17      double puntuacionrestar=0.33;
18
19      for(int numpregunta=0;numpregunta<NUMPREGUNTAS;numpregunta++){
20        respuesta=0;
21        codigo+="<Q"+tipopregunta+"><M"+puntuacion+">"+"<MN"+
              puntuacionrestar+">";
22        codigo+="Cuantos bytes ocupa el siguiente programa?<BR>";
23        codigo+="IN 0<BR>";
24        respuesta+=2;
25        aux= ((int)(1+Math.random()*10));
26          for(int j=0;j<aux;j++){
27            w=((int)(Math.random()*10));
28             codigo+=instrucciones[w]+"<BR>";
29             respuesta+=bytesocupados[w];
30          }
31        codigo+="OUT 1<BR>STOP<BR>";
32        respuesta+=3;
33        codigo+="Escribe el valor en decimal: {"+respuesta+"}";
34      }
35    System.out.println(codigo);
36    }
37  }
```

Esta opción es algo más complicada que el asistente, pero si estamos algo familiarizados con la programación puede servirnos para generar una batería de preguntas con diferentes enunciados y que todas se hallen en el formato correcto. Como hemos dicho esto es útil para generar diferentes exámenes para evaluar el mismo conocimiento. Al ejecutarlo la salida quedaría de esta forma:

```
<QN><M1.0><MN0.33>
¿Cuantos bytes ocupa el siguiente programa?<BR>
IN 0<BR>
POP RA<BR>
POP RA<BR>
DEC RA<BR>
DEC RA<BR>
COMPAREI RA, 0x00<BR>
COMPAREI RA, 0x00<BR>
OUT 1<BR>
STOP<BR>
Escribe el valor en decimal: {13}
</Q>

<QN><M1.0><MN0.33>
¿Cuantos bytes ocupa el siguiente programa?<BR>
IN 0<BR>
POP RA<BR>
POP RA<BR>
MOVE 0x24, RA<BR>
PUSH RA<BR>
COMPAREI RA, 0x00<BR>
ADD RA, 0xA1<BR>
OUT 1<BR>
STOP<BR>
Escribe el valor en decimal: {14}
</Q>
```

Figura 4.12: Ejemplo de ejecución del generador automático de baterías de preguntas

Combinar correspondencia

Esta opción está pensada sobretodo para preguntas que sean fáciles de generar automáticamente, por ejemplo, una traducción de un número a binario, o una multiplicación con distintos operandos, etc. Para ejemplificar el funcionamiento de este generador, vamos a suponer que queremos examinar a todos los alumnos sobre conversiones de números decimales a binario.
Para nuestro ejemplo hemos utilizado el programa de libre distribución LibreOffice, pero otros procesadores de texto como Microsoft Word o OpenOffice también disponen de esta opción. Primero, se prepara la base de datos donde almacenamos los pares de numero y la respuesta indicada. Esta tabla puede generarse de forma manual o utilizando fórmulas de la hoja de cálculo e incluso las herramientas de programación incorporadas, todo depende de la pericia del profesor:

	A	B
1	Numero	Respuesta
2	345	101011001
3	753	1011110001
4	593	1001010001
5	849	1101010001
6	583	1001000111
7	573	1000111101
8	532	1000010100
9	239	1010001110

Figura 4.13: Base de datos preparada para las referencias cruzadas

Después, se prepara el documento de texto con las marcas para las referencias cruzadas:

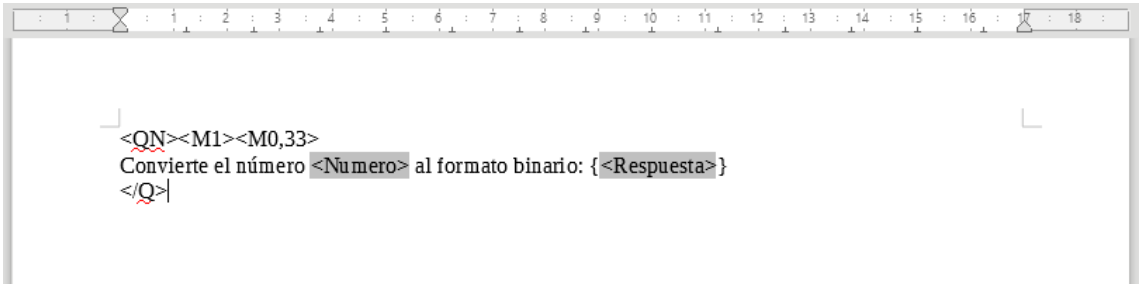
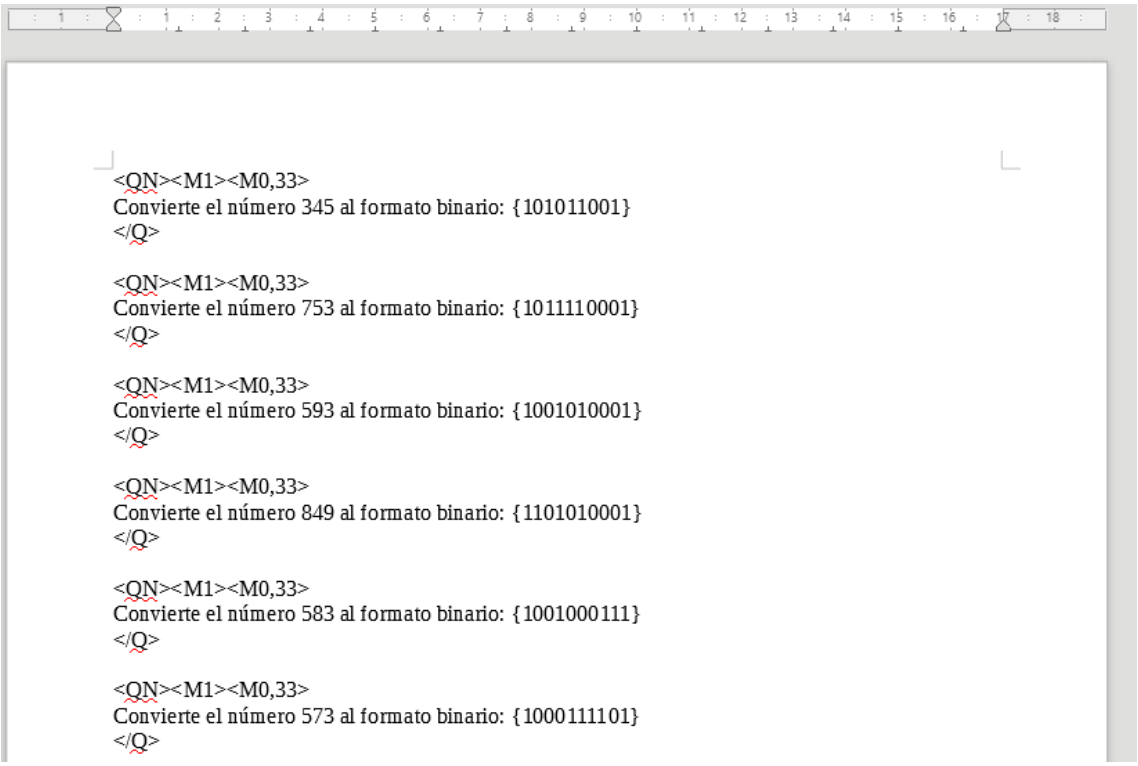


Figura 4.14: Documento de texto preparado para las referencias cruzadas

Por último, al ejecutar el asistente de combinación de correspondencia, se rellenan las preguntas automáticamente:



<QN><M1><M0,33>
Convierte el número 345 al formato binario: {101011001}
</Q>

<QN><M1><M0,33>
Convierte el número 753 al formato binario: {1011110001}
</Q>

<QN><M1><M0,33>
Convierte el número 593 al formato binario: {1001010001}
</Q>

<QN><M1><M0,33>
Convierte el número 849 al formato binario: {1101010001}
</Q>

<QN><M1><M0,33>
Convierte el número 583 al formato binario: {1001000111}
</Q>

<QN><M1><M0,33>
Convierte el número 573 al formato binario: {1000111101}
</Q>

Figura 4.15: Resultado tras la combinación de correspondencia

Este método, como el anterior, sirve para generar baterías grandes de la misma pregunta con diferentes valores y resultados y puede ser útil en muchos casos, como la evaluación de alumnos en los casos que se quiera que tengan exámenes diferentes sobre la misma rama de conocimiento.
Las dos últimas herramientas son sólo ejemplos de posibles modos de generar preguntas de forma automática, los conocimiento y la imaginación del profesor le permitirán encontrar otros modos.

CAPÍTULO 5
Tecnologías

En los siguientes apartados se definen las distintas tecnologías utilizadas para el desarrollo del proyecto. Esta breve introducción ayuda a entender el contexto de las tecnologías y el por qué han sido elegidas estas y no otras.
Este apartado no pretende ser un tutorial exhaustivo sobre las tecnologías, sino un simple acercamiento, por si el lector desconoce alguna de ellas.

5.1 Capa de presentación

5.1.1. HTML

HTML[1], cuyas siglas en castellano vienen a significar *lenguaje de marcas de hipertexto*, es un lenguaje sencillo y accesible para la construcción de páginas web. Hoy en día, su conocimiento, aunque sea a nivel básico, es vital para la distribución de contenidos en Internet. Hablando de su accesibilidad, uno de los puntos fuertes es que los programas para su desarrollo y funcionamiento vienen preinstalados en cualquier sistema operativo de casi cualquier computadora del mundo. Además, desde hace una década, se ofrecen diversas soluciones para que la construcción de páginas web sea tan sencilla como arrastrar elementos y rellenarlos con texto. Muchas de estas alternativas le suenan a casi todo el mundo, son constructores como 1&1, Wix, Wordpress y demás alternativas.
Su importancia es tal que en casi todas las escuelas del mundo, dentro de las asignaturas de informática, se estudian los conceptos básicos. Y es que, en el siglo XXI, casi cualquier persona necesita de la publicación en Internet. Parafraseando a uno de los padres de la informática moderna, Bill Gates: *"Si su negocio no está en Internet, su negocio no existe"*.
Si bien es cierto que con el lenguaje **HTML** *per se* podemos tener una página operativa y funcional en Internet, la mayoría de programadores web conocen también CSS (*Cascading Style Sheets*, por sus siglas en inglés). La finalidad de estas hojas de estilo es complementar al **HTML** y dotarlo de una mejor apariencia en cuanto a fuentes, márgenes, colores, etc.
Desde hace un par de revisiones de **HTML**, el estilo se puede aplicar directamente sobre cualquier elemento con la etiqueta <*style*> y los atributos que deseemos, por lo que, pese a que el estándar está establecido mundialmente como el principal lenguaje web, se sigue intentando mejorar la accesibilidad del mismo.

Para resaltar la importancia de las hojas de estilo, en la siguiente figura se puede ver una comparativa entre nuestra aplicación web con y sin hojas de estilo aplicadas.

(a) Con CSS (b) Sin CSS

Figura 5.1: Comparativa aplicación web con y sin hojas de estilo

Se puede observar que la funcionalidad es básicamente la misma, tanto el *menú*, como los botones de *iniciar sesión*, *registrarse* y *usar sin registro* siguen siendo accesibles y funcionales. Pero, como también puede observarse a simple vista, en la figura de 5.1a, con las hojas de estilo aplicadas, la interfaz de la aplicación web es más clara e intuitiva. El diseño no es solo cuestión de hacer algo más agradable a la vista, también trata de reordenar la información de forma accesible.

5.2 Capa de negocio

5.2.1. ASP.NET

ASP.NET, *Active Server Pages*, por sus siglas en inglés, es una tecnología propiedad de Microsoft, que se utiliza para la creación de aplicaciones web generadas dinámicamente del lado del servidor.
Las páginas de **ASP.NET** son conocidas como *web forms* y están contenidas en archivos con extensión *.aspx*. Se sirven de su propio constructor para la introducción de los elementos web típicos como etiquetas, formularios, etc. Aunque también pueden utilizarse etiquetas HTML, combinándolo con la programación *code-behind* en C#. **ASP.NET** también ofrece la posibilidad de programar en los lenguajes *Visual Basic* y *JavaScript*.
ASP.NET se apoya en el servidor *IIS*, por lo que su funcionalidad está limitada a Windows, por lo tanto no será posible alojar la página web en un servidor Linux. Valoramos la posibilidad de desarrollar la aplicación en otras tecnologías para el soporte de alojamiento en Linux, pero, estudiando el mercado, nos dimos cuenta que el precio de los distintos servidores era más o menos el mismo. No merecía la pena renunciar a una tecnología como **ASP.NET**. (En el momento de redactar esta memoria se puede hacer correr la aplicación desarrollada en **ASP.NET** bajo el servidor *Apache* en Linux, pero su uso está desaconsejado. Al no haber soporte oficial pueden surgir problemas en el transcurso normal de la aplicación web.)
ASP.NET fue liberado en enero de 2002, hasta entonces se llamaba ASP y ahora a estas versiones anteriores se las conoce comúnmente como *ASP clásico*. Una de las características nuevas que introducía **ASP.NET** era la posibilidad de utilizar el modelo *code-behind*.

Code-Behind

Code-Behind es la separación del código correspondiente a diferentes eventos del contenido de la web. Por lo tanto, al final queda un entorno más limpio, accesible y portable para el desarrollador, pues identifica fácilmente qué elementos pertenecen a la interfaz y qué elementos controlan las interacciones de la aplicación. Por un lado tenemos el contenido de la página, como pueden ser los textos, los iconos, formularios etc. escritos en HTML y en un archivo separado el código C#
Este modelo es el que hemos seguido para el desarrollo de nuestra aplicación web Para controlar esta interacción se utilizan etiquetas colocadas al inicio del documento ASPX. En esta etiqueta se especifica el archivo que controla la construcción de la página web.

```
1 <%@ Page Language="C#" AutoEventWireup="true"
2 CodeFile="Default.aspx.cs" Inherits="_Default" %>
```

Con esta etiqueta y la directiva Page, podemos indicarle a la página cómo debe controlar el código de la aplicación, por partes:

- ***Language='C#'***: este atributo indica al compilador el lenguaje en el que está escrito el código. C# en este caso.
- ***AutoEventWireup='true'***: aquí indicamos si vamos a seguir el modelo *inline* (típico del ASP clásico) o el modelo ***code-behind***. Al darle valor true indicamos esto último.
- ***CodeFile='Default.aspx.cs'***: Aquí le indicamos el nombre del archivo donde incluiremos el código de la aplicación.
- ***Inherits='_Default'***: indicamos la clase que se va a heredar, esta clase va a ir a buscarla al archivo que indicamos en el atributo anterior.

La página que indicamos en esta directiva tiene una estructura básica similar a esta:

```
1  using System;
2
3
4  public partial class Default2 : System.Web.UI.Page
5  {
6  protected void Page_Load(object sender, EventArgs e)
7  {
8
9  }
10 }
```

En el método existente por defecto se dan directivas sobre lo que debe hacer la página mientras se carga. Desde este método se controlan acciones como la gestión de usuarios (ocultar determinadas partes de la página o redireccionar si no tienes un usuario con permisos necesarios) y también se llaman a los distintos métodos que se van a usar.
Un ejemplo de esto último es nuestro convertidor. El formulario llama a uno de estos métodos al pulsar sobre el botón enviar.

5.2.2. Visual Studio

Visual Studio es un entorno de desarrollo integrado para sistemas operativos Windows. Soporta múltiples lenguajes de programación como Visual Basic, C#, C++, Java, PHP, Python... y diversos entornos de desarrollo como MVC, Django o ASP.NET, entre otros. Gracias a este amplio abanico de opciones se pueden desarrollar prácticamente todos los tipos de aplicaciones imaginables, como páginas web, dispositivos móviles, estaciones de trabajo, etc.
Para nuestro proyecto hemos utilizado la versión Community, que tiene casi todas las funcionalidades de la versión completa, pero Microsoft otorga licencia para usarlas en proyectos pequeños o educativos.

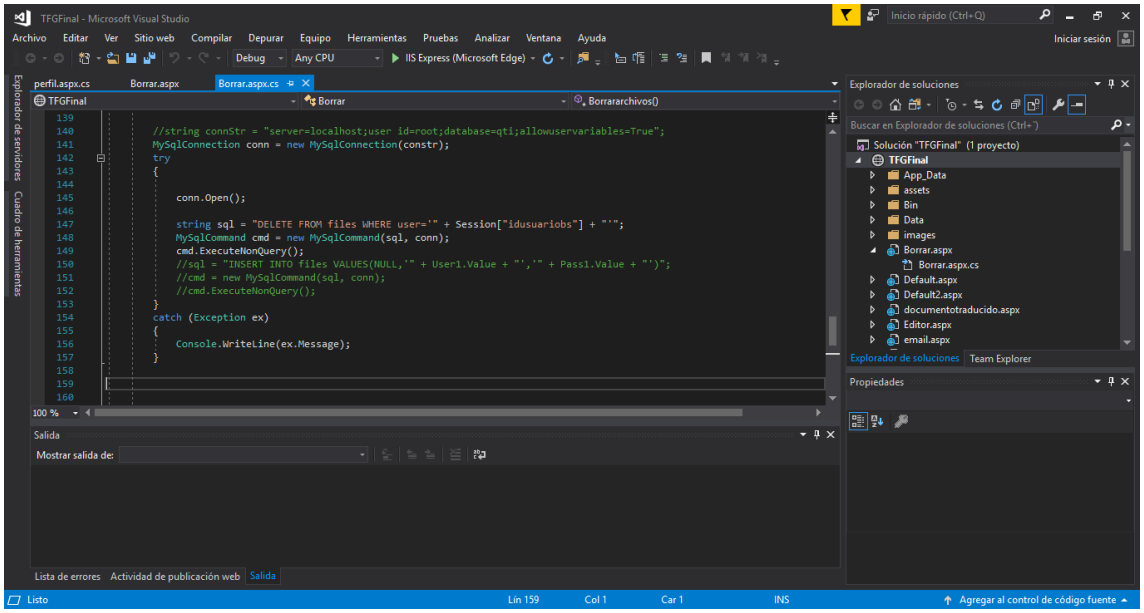


Figura 5.2: Interfaz Visual Studio

5.2.3. C#

C# o C Sharp[4] es un lenguaje de programación orientado a objetos desarrollado por Microsoft. Forma parte de su plataforma .NET, aunque ya existe un compilador implementado para generar programas para distintas plataformas, como Unix, Android, iOS...
Su sintaxis básica viene influenciada por C y C++, como evolución de los mismos. Por lo general, es un lenguaje bastante asequible si el programador está familiarizado con otros lenguajes orientados a objetos como podría ser JAVA, como se puede observar en la siguiente sección de código, que muestra la estructura básica de un programa en **C#**, con el archiconocido ejemplo Hello World!

```
using System;
namespace HelloWorld
{
class Hello
{
static void Main()
{
Console.WriteLine("Hello World!");
Console.WriteLine("Press any key to exit.");
Console.ReadKey();
}
}
}
```

5.3 Capa de acceso a datos

5.3.1. Bases de datos

Una **base de datos**[3] es un conjunto de datos ordenados que pertenecen a un mismo contexto. Los datos se almacenan para ser utilizados posteriormente.

En la mayoría de aplicaciones que utilizan una base de datos se utiliza una **base de datos** relacional. La idea fundamental de este tipo de **bases de datos** es el uso de relaciones, las tablas se enlazan o conectan mediante claves compartidas en varias tablas. Estas relaciones se organizan en forma lógica como conjuntos de datos llamados tuplas. El diseño de esta **base de datos** se fundamenta en representar los problemas reales y administrar los datos dinámicamente. A la hora de diseñar la **base de datos** debemos tener en cuenta el proceso de normalización, para:

- Evitar la redundancia de los datos.
- Disminuir problemas de actualización de los datos en las tablas.
- Proteger la integridad de los datos.

Para que una tabla se encuentre normalizada tiene que cumplir las siguientes restricciones:

- Cada tabla debe tener su nombre único.
- No puede haber dos filas iguales. No se permiten los duplicados.
- Todos los datos en una columna deben ser del mismo tipo.

Más adelante, se entrará en detalle sobre la implementación de nuestra **base de datos** en concreto.
Para la gestión de las **bases de datos** se utiliza un SGBD (Sistema Gestor de Bases de Datos). En nuestro caso hemos escogido MySQL, por las razones que se detallan en el siguiente punto.

MySQL

MySQL es un sistema de gestión de bases de datos relacional. Existen muchos otros, pero estas son algunas de las características que hacen interesante su uso:

- Es multiplataforma, se puede utilizar tanto en Windows, Linux o Mac.
- Gran cantidad de documentación, al ser un SGBD de amplio uso.
- Es fácil de aprender si conocemos el estándar SQL.
- Éxito probado. Al ser una base de datos ampliamente utilizada en grandes proyectos y empresas, nos ofrece una fiabilidad muy alta.

- Sencillez de gestión. A diferencia de otras bases de datos como Oracle, esta, al ofrecer unas características mínimas para el funcionamiento normal de una aplicación, hace que para el mantenimiento de la base de datos no sea necesario recurrir a un DBA (*DataBase Administrator*, por sus siglas en inglés).
- Escalabilidad. Permite escalar la base de datos en caso de que lo necesitemos, aunque en principio no va a ser necesario.
- Licencia GPL, por lo que el coste inicial es cero.
- Las transacciones cumplen el principio ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad), esto es interesante para mantener la integridad de los datos almacenados.

CAPÍTULO 6
Implementación

6.1 Capa de presentación

En esta sección se va a hacer un repaso breve de la interfaz gráfica de la aplicación y las distintas funcionalidades a las que se tiene acceso mediante ella. Como ya hemos explicado anteriormente, vamos a hacer uso de webforms de ASP (con extensión .aspx), pero el lenguaje utilizado para la construcción de la interfaz es HTML, combinado con CSS y scripts de JS.

- Default.aspx: Default es la nomenclatura que otorga ASP.NET al archiconocido index.html. Desde este documento se puede acceder a la mayoría de las funciones de la aplicación. Tenemos la posibilidad de registrarnos (llamada a registro.aspx), iniciar sesión (llamada a login.aspx), o usar la aplicación sin registrarnos mediante el formulario de subida de archivos de la parte inferior, en el footer.

Figura 6.1: Index

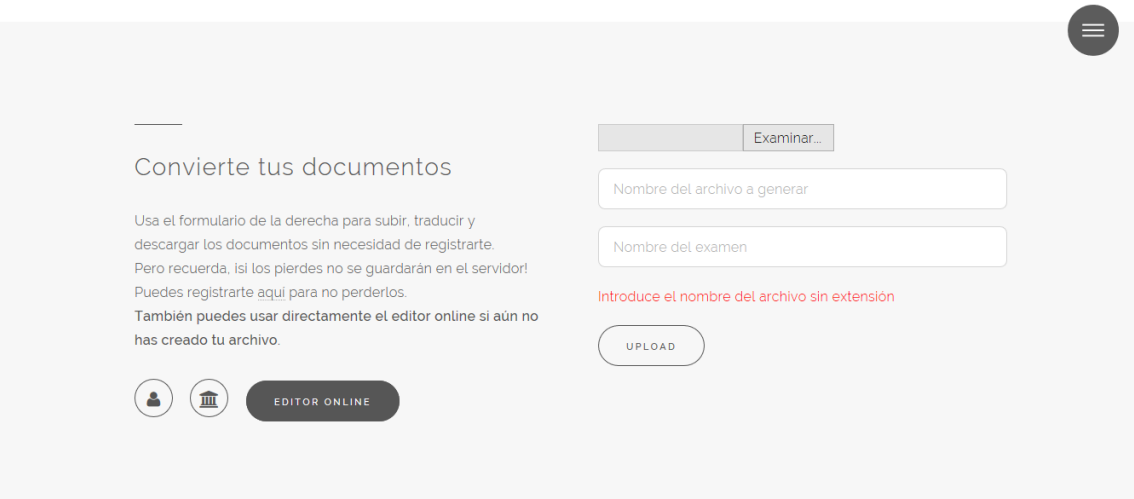


Figura 6.2: Footer index

- Editor.aspx: Desde la interfaz de este documento se puede introducir directamente el lenguaje de marcas a transformar en un archivo en el estándar QTI/IMS. El funcionamiento del lenguaje de marcas es exactamente idéntico que en la subida de archivos.

Figura 6.3: Editor online

- DocumentoTraducido.aspx: Esta pantalla sirve como enlace entre la subida y traducción del archivo y la descarga del mismo. Al principio pensamos en la posibilidad de que la descarga del archivo sucediera de forma automática sin tener que aceptar una confirmación adicional, pero nos dimos cuenta de que a veces el usuario no querría descargar todos los archivos que tradujera, y podía ser algo pesado de cara a una utilización ágil de la aplicación.

Figura 6.4: Documento generado

- Registro.aspx: Desde aquí un usuario puede registrarse para hacer uso de las funciones para usuarios registrados. Se pide un nombre de usuario, un email y una contraseña (con 6 caracteres mínimo). Mediante un script de js se comprueba que la contraseña cumple la política. En caso afirmativo, se activa el botón de registro.

Figura 6.5: Registrarse

- Login.aspx: Introduciendo el email o usuario y la contraseña correctas el usuario puede acceder a su perfil y hacer uso de las funciones reservadas para usuarios registrados como el historial de archivos.

Figura 6.6: Login

- Perfil.aspx: Esta es la ventana principal para los usuarios registrados. Desde aquí pueden controlar todas las opciones de su cuenta, como el cambio de contraseña o la posibilidad de darse de baja en la aplicación. En la parte inferior se encuentra el historial de archivos, en el que están disponibles para descargar tanto el .txt del examen, así como el .xml en el estándar QTI/IMS.

Figura 6.7: Perfil personal

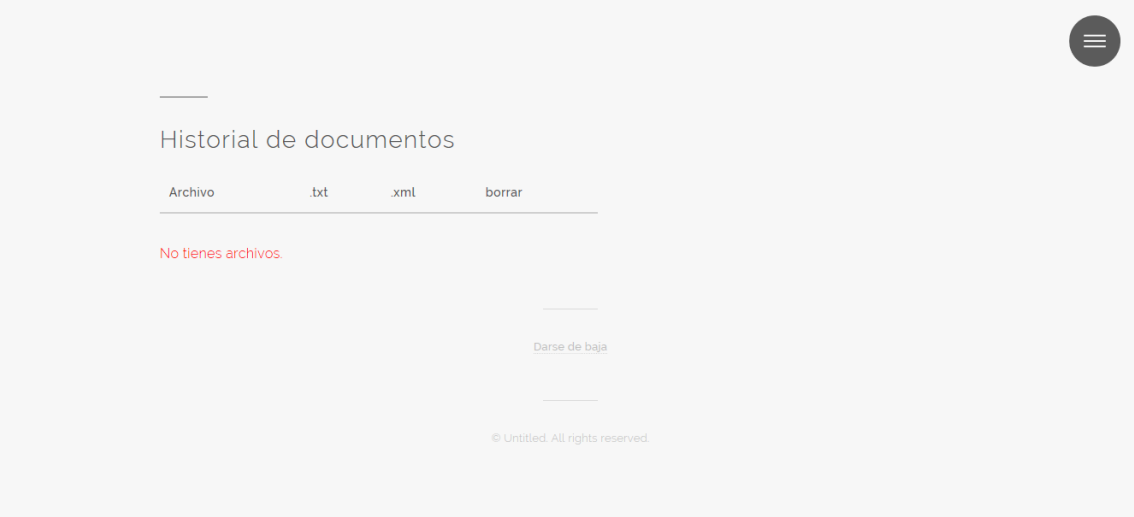


Figura 6.8: Historial de archivos

- ModificarContraseña.aspx: Aquí el usuario puede cambiar su contraseña, introduciendo la vieja y confirmando la nueva que haya elegido. El mismo script que estaba presente en el registro se utiliza aquí para controlar que la nueva contraseña también cumpla con la política establecida.

Figura 6.9: Modificar contraseña

- Email.aspx: Si un usuario no recuerda su contraseña puede restablecerla mediante el envío de un email.

Figura 6.10: Recuperar contraseña

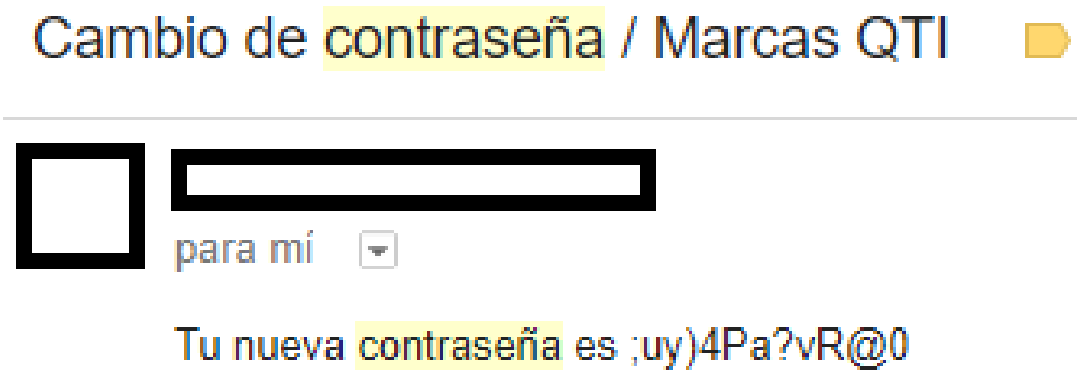


Figura 6.11: Email de recuperación

- Borrar.aspx: Desde aquí el usuario puede darse de baja en la aplicación.

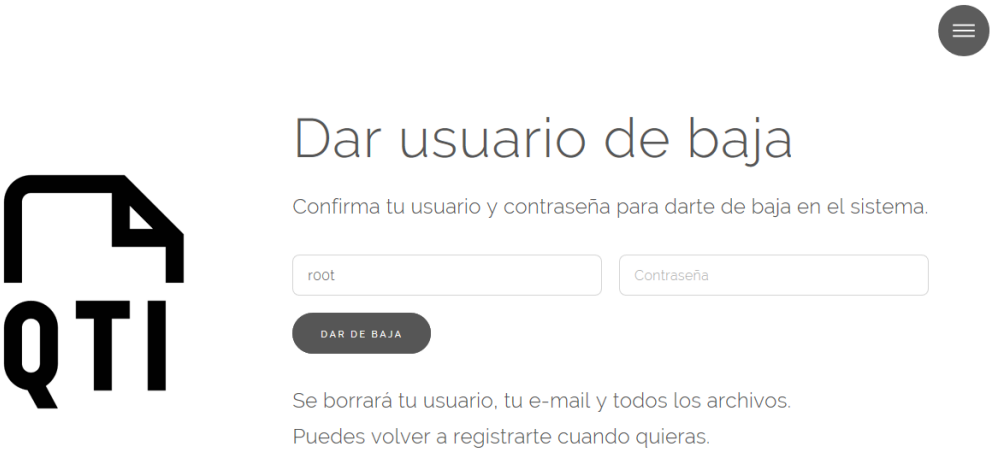


Figura 6.12: Darse de baja

6.2 Capa de negocio

En esta sección se detalla las funciones que realizan los documentos adjuntos a los archivos que muestran la interfaz. Conviene recordar de nuevo que ASP.NET hace uso de code-behind, por lo tanto cada documento de interfaz tiene adjunto un documento de código que controla sus funciones.

- Default.aspx.cs: El código de este documento es el que transforma el lenguaje de marcas al estándar QTI/IMS (en la sección 6.2.1 se comenta con más detalle) y almacena los archivos en el directorio personal del usuario, en el caso de que esté logueado en el sistema. Al finalizar la traducción se produce una redirección a documentotraducido.aspx con la ruta del archivo a descargar.

- Editor.aspx.cs: Este archivo contiene el código que transforma el texto que se introduce a texto plano, por lo que es una buena opción para copiar y pegar preguntas desde procesadores de textos y evitar la mayoría de problemas con caracteres ocultos. Para generar la traducción se hacen uso de los métodos del archivo comentado anteriormente.

- DocumentoTraducido.aspx.cs: Este documento es llamado con una ruta en el parámetro que indica el archivo que ha sido traducido para que la aplicación pueda dirigirse al directorio indicado y descargar el archivo correcto.

- Registro.aspx.cs: El código incluído en este documento realiza las consultas a la base de datos para controlar el registro de un usuario (primero comprueba que no exista un usuario previo con los mismos datos y luego almacena los datos en las tablas).

- Login.aspx.cs: El código busca al usuario en la base de datos y compara el hash de las passwords. Si lo encuentra, salta a la pantalla perfil.aspx
- Perfil.aspx.cs: El código de este documento recupera desde la BBDD todos los documentos asociados al usuario que está navegando por su perfil personal. También controla el borrado de archivos.
- ModificarContraseña.aspx.cs: Este documento modifica la contraseña del usuario en la BBDD, tras comprobar que es él realmente el que está haciendo la petición (Esto se controla mediante la consulta de la contraseña actual).
- Email.aspx: El código sustituye la contraseña actual del usuario por otra nueva generada aleatoriamente y controla el envío por email de la misma.
- Borrar.aspx.cs: Mediante el código que controla este webforms, se borra todo tipo de relación del usuario tras pedirle varias veces su confirmación. Serán eliminados tanto sus datos como su directorio personal de archivos.
- Logout.aspx.cs: Como su nombre indica, sirve para cerrar la sesión del usuario. Esta pantalla solo se compone de código y se controla su uso desde el menú. No hay una interfaz gráfica para ello.

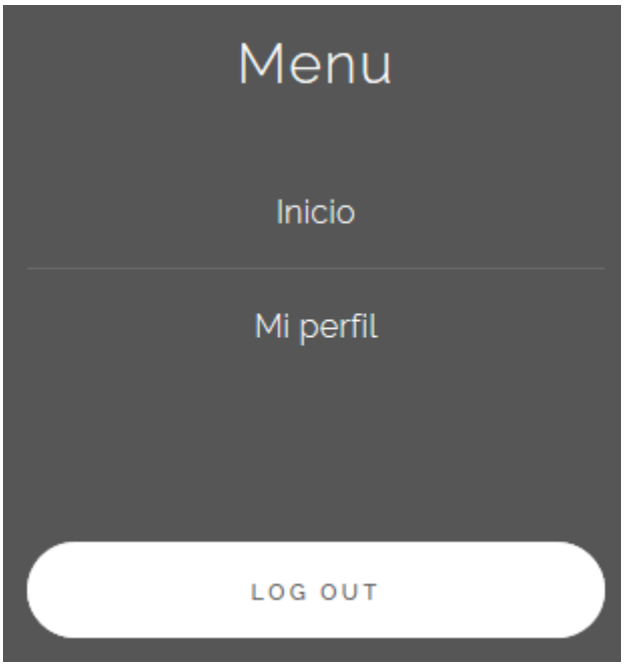


Figura 6.13: Logout

6.2.1. Traductor de lenguaje de marcas a QTI/IMS

El traductor de lenguaje de marcas al estándar QTI/IMS cuyo código se encuentra en el documento *Default.aspx.cs*, ya comentado en la sección 6.2, ya había sido implementado en el TFG citado en la introducción en forma de aplicación de escritorio. Para la realización de este proyecto teníamos acceso al código del mismo, por lo que era innecesario tener que realizarlo de nuevo desde cero. Bien

es cierto que ha habido un trabajo de adaptación del mismo. Podríamos resumir el trabajo en dos objetivos: Adaptar el código para que funcione en la nueva interfaz gráfica de la aplicación web y añadir una gestión de errores.
Para el primer objetivo se han añadido dos nuevos métodos al código original. El primero se encarga de la gestión del fichero (o del texto si el usuario utiliza el editor), leerlo, convertirlo a texto y analizar el tipo de preguntas que el usuario ha introducido. Una vez se obtiene la relación de preguntas a traducir, se llama al método original que se encarga de esto (y del que más adelante comentaremos las modificaciones realizadas). El segundo método que se ha añadido era esencial, ya que uno de los requisitos de nuestra aplicación web es que gestione usuarios y un historial de archivos para cada uno. Este método se encarga de identificar al usuario que ha pedido la traducción y almacenar en su correspondiente carpeta personal el fichero introducido (y, más adelante, de guardar el fichero traducido en el mismo sitio). Además, este método también comprueba que no haya dos archivos con el mismo nombre. El formato elegido si hay coincidencias de nombres ha sido introducir el nombre original seguido de la fecha actual.
Para el segundo objetivo, se ha añadido dentro del método original de conversión una comprobación de que las etiquetas introducidas siguen el formato correcto y, sobre todo, que la pregunta se abre y se cierra correctamente. Si hay algún error, la aplicación web se encarga de notificar en qué línea ha encontrado algo que no puede reconocer (Esto se puede comprobar más adelante en el Anexo 1, donde viene detallado y se muestra una captura del error).
A su vez, se han reordenado y modificado algunas partes del código para hacer más comprensible la lectura del mismo en el futuro, tanto para nosotros, si entráramos a mejorar la aplicación o corregir algún error inesperado, como para otras personas que puedan necesitar este proyecto como trabajo previo para el suyo propio.

6.3 Capa de acceso a datos

Para la implementación de la base de datos se ha elegido MySQL como el SGBD que controla la persistencia de los datos de los usuarios. La implementación ha seguido fielmente el diseño que se ha propuesto antes. En las siguientes imágenes se puede observar con más detalle la estructura de las tablas.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	**id** 🔑	int(255)			No	*Ninguna*		AUTO_INCREMENT
2	**user**	varchar(300)	latin1_swedish_ci		No	*Ninguna*		
3	**file**	varchar(300)	latin1_swedish_ci		No	*Ninguna*		

Figura 6.14: Estructura de la tabla files

Para la primera tabla, files, se ha optado por implementar el campo *id* como un campo autoincremental que numera los registros introducidos. Este campo es interesante sobretodo a nivel estadístico para poder controlar cuantos archivos ha

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	id	int(255)			No	*Ninguna*		AUTO_INCREMENT
2	user	varchar(300)	latin1_swedish_ci		No	*Ninguna*		
3	pass	varchar(300)	latin1_swedish_ci		No	*Ninguna*		
4	email	varchar(300)	latin1_swedish_ci		No	*Ninguna*		

Figura 6.15: Estructura de la tabla users

manejado históricamente la aplicación. Los atributos *user* y *file* se han implementado como varchar, para que el usuario y el nombre del archivo se introduzcan como texto plano. Esta forma de implementación es interesante, ya que con una consulta simple se pueden recuperar todos los archivos que ha subido cada usuario por separado.
Para la segunda tabla, se ha introducido también un campo *id*, por los mismos motivos que en la tabla anterior. Los campos restantes se han implementado también pensando en que el usuario va a introducir texto plano, tanto para el nombre de usuario, el email y el hash de la contraseña.

6.4 Seguridad

Para cumplir los requisitos de seguridad especificados en el capítulo correspondiente, se han realizado las siguientes acciones. No solo para garantizar en la medida de lo posible la integridad de la aplicación, sino también para garantizar la confidencialidad de los datos de los usuarios, haciendo especial atención a la contraseña personal.

En primer lugar, los usuarios al registrarse en la aplicación introducen su contraseña personal en texto plano. Pese a que para el almacenamiento de las mismas utilizamos MySQL, uno de los sistemas más fiables en lo que a bases de datos se refiere, y nadie externo al administrador del sistema puede consultar las mismas, tampoco nos parecía ético el hecho de que los administradores pudieran conocerlas. Para resolver este problema, hemos decidido encriptar las contraseñas antes de almacenarlas en la base de datos. En nuestro caso, nos hemos decidido por el algoritmo SHA-256, que es un hash de 64 dígitos hexadecimales. El hash solo se calcula en una dirección y no se puede decodificar de vuelta, por lo que aunque alguien consiguiera recuperarlos, tardaría años, en el mejor de los casos, en decodificarlos.

En segundo lugar, la utilización de code-behind, explicado anteriormente, nos permite ocultar la información y la estructura de la aplicación al usuario. Todas las peticiones que se envían desde el usuario se validan y ASP.NET las rechaza si contienen alguna estructura inválida. Esto evita en gran medida los ataques de cross-site scripting.

En tercer lugar, se da la opción al usuario de cambiar la contraseña siempre que lo desee sin ningún tipo de restricciones, por si existiera la sospecha de que ha habido algún acceso no autorizado a su cuenta personal. Y, aún en el caso de que

alguien consiguiera entrar en su cuenta personal y modificar la contraseña, podría recuperar el acceso a su cuenta haciendo uso del email de recuperación. Se crea una contraseña aleatoria de 12 caracteres, siendo mínimo uno de ellos un carácter no alfanumérico.

Por último, pero no menos importante, se han realizado pruebas de *SQL Injection*. La inyección SQL se basa en insertar código invasor dentro del código SQL programado, a fin de alterar el funcionamiento normal del programa. Si este ataque resulta exitoso se pueden borrar tablas, recuperar datos ocultos, etc.
En el prototipo inicial de la aplicación, este tipo de ataque era bastante sencillo de realizar, incluso sin tener grandes conocimientos informáticos. Para ejemplificar esto, tenemos que situarnos en la pantalla de registro. Si un usuario introduce los datos normalmente, la consulta queda de la siguiente forma:

```
1 INSERT INTO users VALUES(NULL,'"+nombreUsuario+"','"+hash+"','"+email+"
    ')"
```

En la consulta, *nombreUsuario* hace referencia al nombre introducido por el usuario, *hash* a la contraseña encriptada y *email*, al email introducido por el usuario. Si el usuario introduce correctamente los valores, el sistema hará la consulta y todo funcionará como está previsto. El problema viene cuando el usuario introduce en el campo email el código malicioso, por ejemplo:

```
1 ejemplo@email.com');DROP TABLE users;INSERT INTO files VALUES('ejemplo'
    ,'ejemplo
```

Al introducir esto en el campo de email, el usuario se introducía correctamente en la base de datos, a continuación se ejecutaba la consulta **DROP TABLE users**, que era el objetivo real, que borra la tabla de usuarios, y la última sentencia de **INSERT** se utiliza para mantener la coherencia de las consultas y que la comilla y el paréntesis final que inserta la aplicación desde el código tengan coherencia y las consultas se ejecuten al no haber ningún error semántico.
Para subsanar este problema de seguridad se han tomado dos medidas. La primera pasa por eliminar los caracteres especiales de las consultas, en especial el punto y coma, que hace que dos consultas se puedan anidar en un solo query. La segunda solución implementada pasa por comprobar, antes de enviar la consulta a la base de datos, que el string que se pasa al servidor cumple la estructura esperable de la misma. En este caso, que tenga una sola consulta tipo INSERT, y que se introduzcan 4 valores únicamente.

CAPÍTULO 7
Implantación del sistema y pruebas

En este capítulo se va a hacer una breve descripción de como se ha implantado el sistema. En las siguientes secciones se van a ofrecer también los resultados de las pruebas realizadas.

7.1 Despliegue local

El despliegue de la aplicación web en un servidor local ha sido un valor muy útil para poder probar en el momento todos los cambios que se iban realizando. Este despliegue ha sido utilizado sobretodo en el primer prototipo, hasta que hemos conseguido implantar la base de datos MySQL, un diseño de la interfaz que nos agradara y las funcionalidades básicas de la aplicación. A su vez también nos ha permitido realizar pruebas de rendimiento y validación de código en un entorno real.
El despliegue se ha dividido en dos fases. Primero, se ha implantado un servidor de bases de datos que gestione MySQL. Gracias al IDE de Visual Studio, explicado anteriormente, el despliegue ha sido sencillo, pues este tiene integración nativa con MySQL. Después de crear las tablas y realizar la configuración necesaria, se ha desplegado la aplicación web. Para esto, necesitábamos un servidor Windows. Existen varias opciones disponibles, pero nos hemos decantado por la opción nativa de Windows, IIS. Al venir lista para instalar en cualquier sistema operativo Windows (7 o superior), nos ha parecido la opción más recomendable. Además, como en el caso de la base de datos, se integra nativamente con el IDE. En cuanto al resto del proceso, el propio Visual Studio se encarga de compilar la aplicación web e interpretar el funcionamiento de los comandos de la misma y mostrártela en el navegador web elegido. Ya que nuestro proyecto no versa sobre esto, no entraremos en más detalle.

7.2 Pruebas

7.2.1. Pruebas de funcionamiento en navegadores

Para comprobar el funcionamiento de la aplicación en los distintos navegadores objetivos, se ha desplegado la web en todos ellos y se ha hecho un trabajo de

rutina básica para probar todas las funcionalidades. La rutina seguida en todos ellos ha sido:

1. Subir un archivo sin registro.
2. Descargar el archivo.
3. Registrar un usuario.
4. Iniciar sesión.
5. Subir un archivo desde el usuario.
6. Recuperarlo desde el historial de archivos.
7. Modificar la contraseña.
8. Logout.
9. Recuperar la contraseña.
10. Darse de baja en la aplicación.

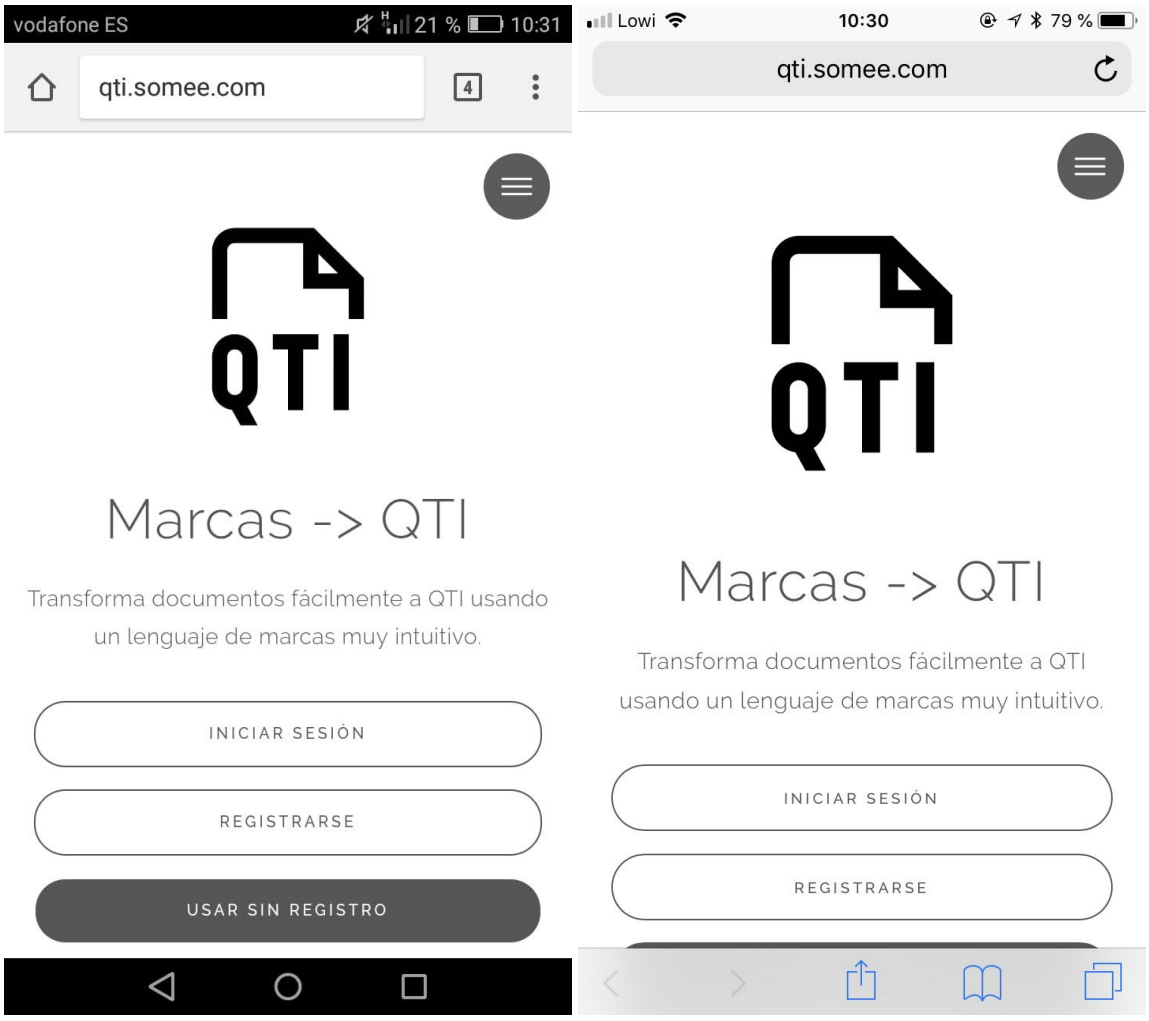
Con estos 10 pasos, se pueden comprobar las funcionalidades básica de la aplicación. Los resultados han sido satisfactorios en todos ellos. Además, mientras navegábamos por la aplicación, íbamos comprobando que no había errores de visualización. A continuación se ofrecen unas capturas del index para mostrar que la interfaz mantiene la coherencia entre navegadores:

Figura 7.1: Interfaz en Windows y Opera Browser

Figura 7.2: Interfaz en Windows y Microsoft Edge

Figura 7.3: Interfaz en Windows y Google Chrome

Figura 7.4: Interfaz en Windows y Mozilla Firefox



(a) Android y Google Chrome (b) Iphone y Safari

Figura 7.5: Interfaz en dispositivos móviles

7.2.2. Pruebas de rendimiento

En esta sección se describen las pruebas realizadas para evaluar el funcionamiento de la aplicación web en términos de potencia.
Para ello, hemos realizado pruebas de rendimiento en la traducción de archivos, obteniendo los siguientes datos:

n Preguntas	Tiempo de respuesta
5	00.01s
50	00.04s
100	00.10s
500	00.60s
1000	01.50s
2000	03.50s
3000	07.13s
4000	10.87s

Tabla 7.1: Tabla de tiempos de respuesta

La conclusión que sacamos tras analizar estos datos es que el tiempo de respuesta medio de la aplicación es muy bueno. En la práctica casi nunca se van a utilizar baterías de más de mil preguntas y el tiempo de respuesta para estos casos no llega a los dos segundos.

7.2.3. Pruebas de validación HTML

Para asegurarnos de que el desarrollo de la aplicación web cumple con los estándares del World Wide Web Consortium, vamos a utilizar su propio validador para detectar errores en el código de la web. A la hora de realizar estas pruebas, nos hemos encontrado con dos dificultades. La primera pasa porque el código que se genera en un documento tipo web forms de ASP.NET, ha de ser previamente procesado por el servidor para poder validar su código HTML. Si intentas validar el archivo directamente, vas a comprobar que las etiquetas propias de ASP.NET, como las que controlan el *code-behind*, que han sido explicadas anteriormente, o *runat* (que se encarga de decirle al archivo de code behind que ese elemento debe ser procesado), son interpretadas por el validador como errores, al no reconocerlas. Estas etiquetas desaparecen al ser procesadas por el servidor que aloja la aplicación web. Para subsanar este problema, hemos decidido implantar la aplicación temporalmente en un servidor de hosting gratuito, somee.com. A continuación se muestra un ejemplo de la validación de W3C en el index de la aplicación:

Figura 7.6: Validación W3C para index de la aplicación web

7.2.4. Pruebas de validación CSS

A la hora de validar una página web, debemos considerar no solo el HTML asociado, que controla el contenido, sino también el CSS, que controla la forma en que se presenta la aplicación. En este caso, al habernos enfocado en la sencillez de diseño, la aplicación web hace uso de un único documento en lenguaje CSS que da formato a todos los elementos de la web.
Para validar el código CSS, vamos a hacer uso de la misma herramienta que en el apartado anterior, que nos ofrece gratuitamente W3C y que es el estándar utilizado mundialmente para este propósito.
A continuación se muestra la imagen que prueba que el código ha pasado las pruebas del validador con éxito:

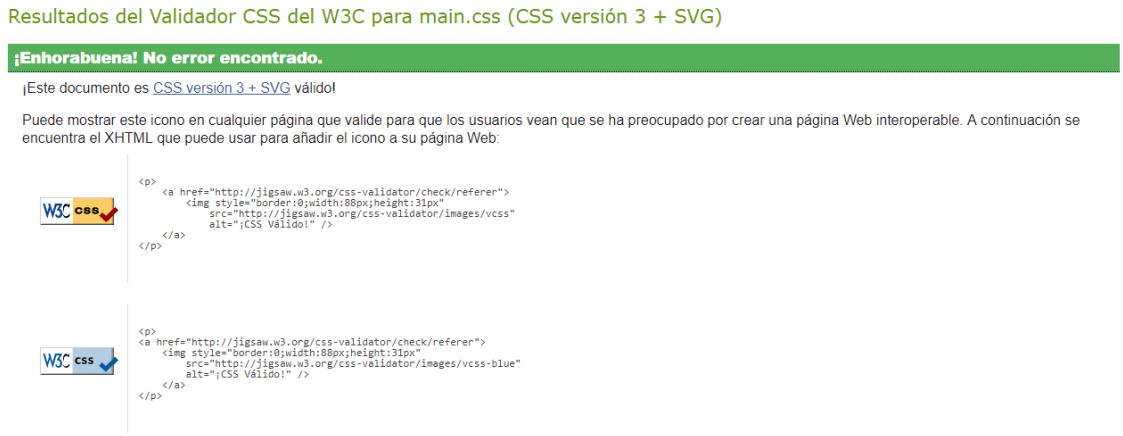


Figura 7.7: Validación W3C del CSS de la aplicación web

CAPÍTULO 8
Ampliaciones futuras

A día de hoy, la fase en la que se encuentra nuestro proyecto, nos convence a todos y es suficiente para el uso que se le pretende dar, pero, siempre hemos creído que no hay nada que no se pueda mejorar. Por esto, hemos pensado una serie de mejoras y ampliaciones que se podrían valorar en un futuro según fueran las necesidades del momento.

- Creación de una aplicación móvil para hacer nativamente la traducción de documentos. La mayor ventaja que otorgaría sería tener una versión offline de la aplicación web, por lo que no haría falta ningún tipo de conexión, como actualmente.
- Creación de una aplicación para Linux. Ya hemos hablado de la portabilidad y actualmente disponemos de una versión offline para Windows y una aplicación web para su uso con conexión en cualquier dispositivo, pero tampoco estaría de más tener una versión offline para otro de los principales sistemas operativos.
- Introducir más tipos de preguntas. Poliformat ofrece, en estos momentos, 12 tipos diferentes de preguntas, mientras que la herramienta de conversión es capaz de convertir sólo 3. Hay varias razones para que sólo se ofrezcan estos tres tipos de preguntas. En primer lugar, no todas las preguntas que actualmente se ofrecen en Poliformat estaban disponibles cuando se creó la herramienta, como por ejemplo el tipo de pregunta Respuesta calculada. En segundo lugar, no todos los tipos de preguntas son susceptibles de ser automatizadas. Y por último, las preguntas que se escogieron al desarrollar la herramienta fueron las que el tutor del TFG necesitaba. Sin embargo, un trabajo futuro interesante es estudiar las nuevas preguntas disponibles en Poliformat e incorporar aquellas que sea interesantes y puedan automatizarse a la herramienta de conversión.

CAPÍTULO 9
Conclusiones

Tras la realización y finalización del proyecto, llega la hora de valorar tanto el trabajo realizado como lo aprendido a nivel personal.

En cuanto al primer aspecto, estamos muy satisfechos con nuestro rendimiento, pues todos los requisitos que nos planteábamos al principio se han cumplido. Además, hemos continuado mejorando la aplicación y planteándonos funcionalidades nuevas, para las que hemos tenido siempre el tiempo y los recursos suficientes. Creemos que las tecnologías escogidas han sido las correctas para el desarrollo total del proyecto, si bien al principio nos planteábamos escoger otras, como desarrollar el código en PHP.

Algunos aspectos de nuestra formación han sido muy importantes para el desarrollo de este proyecto, ya que, si bien no habíamos trabajado directamente con el entorno ASP.NET, como sí que habíamos hecho con bases de datos, nuestra formación previa nos ha otorgado la base necesaria para desarrollar la aplicación web con soltura. Por citar algunas de las asignaturas que nos han facilitado esta tarea podríamos nombrar *Programación*, *Bases de datos y sistemas de información*, *Estructuras de datos y algoritmos*, *Ingeniería del software*, *Interfaces persona computador* y *Diseño y gestión de bases de datos*, entre otras.

A nivel personal, el desarrollo de la aplicación web nos ha permitido conocer más en profundidad un lenguaje de programación (C#) y un IDE (Visual Studio) que son dos de los más importantes en el mundo de desarrollo web. Además, la experiencia de trabajar con plazos de entrega y tener que ir desarrollando un informe sobre el proyecto a medida que se trabajaba en él nos puede aportar un valor añadido en el mundo laboral.

En definitiva, estamos bastante contentos en general, tanto con el desarrollo del proyecto, como con el resultado final. La aplicación web debería ayudar a alentar a los docentes a hacer uso de la misma, puesto que se ha conseguido la portabilidad y accesibilidad buscada. También creemos que este proyecto debería alentar a otros docentes a desarrollar proyectos conjuntamente con alumnos para continuar mejorando el uso de las plataformas docentes, pues con las tecnologías y conocimientos adecuados se pueden implementar soluciones adaptadas a las necesidades de los docentes.

Bibliografía

[1] Tim Berners-Lee and Dan Connolly. Hypertext markup language-2.0. Technical report, 1995.

[2] Tim Bray, Jean Paoli, C Michael Sperberg-McQueen, Eve Maler, and François Yergeau. Extensible markup language (xml). *World Wide Web Journal*, 2(4):27–66, 1997.

[3] Diego Rafael Llanos Ferraris. Fundamentos de informática y programación en c. *Paraninfo*.

[4] Anders Hejlsberg, Scott Wiltamuth, and Peter Golde. *C# language specification*. Addison-Wesley Longman Publishing Co., Inc., 2003.

[5] HÉCTOR HERRAIZ MUÑOZ. Convertidor de lenguaje de marcas a formato qti/ims, 2015.

[6] Luis Ortiz Farley. Campus virtual: la educación más allá del lms. *Revista de Universidad y Sociedad del Conocimiento (RUSC)*, 2007.

[7] IMS QTI. Ims question & test interoperability specification. *IMS Global Learning Consortium*, 2005.

[8] Introducing Sakai. https://sakaiproject.org/.

[9] Market Share. Browser market share, 2009.

[10] William R. Watson. An argument for clarity: What are learning management systems, what are they not, and what should they become? *TechTrends*, 51:2, February 2007.

APÉNDICE A
Manual de usuario

A.1 Conversión sin registro

A la hora de subir un archivo debemos hacerlo desde este formulario ubicado en la página de Inicio. El formulario es el mismo que si usamos la página registrándonos, pero si lo hacemos sin registrar, los archivos generados (tanto el .txt con el lenguaje de marca, como el .xml convertido) no se guardarán en nuestro historial de archivos (apartado A.2.8) para ser recuperados posteriormente.

Figura A.1: Formulario de conversión sin registro de la aplicación web

Por seguridad, el tamaño máximo de archivo permitido es de 8MB. Realizando pruebas hemos estimado que este tamaño límite permite generar más de 4000 preguntas por archivo, por lo que debería ser suficiente.

Con el botón *Examinar...* buscamos y seleccionamos un archivo .txt de nuestro ordenador, smartphone o tablet. En el campo de texto de abajo podemos introducir el nombre del archivo que vamos a generar y en el otro campo podemos establecer el título del examen que aparecerá una vez subido a Sakai o la plataforma docente correspondiente.

Para mayor comodidad del usuario, estos dos campos se autorrellenan con el nombre del fichero que se está subiendo, pero pueden ser cambiados a posteriori.

Nota: Tal y como avisa el formulario, no es necesario introducir la extensión .xml, la aplicación la añade automáticamente.

Una vez generado el documento la aplicación redirige a la siguiente pantalla.

Figura A.2: Descarga sin registro de la aplicación web

El documento generado puede ser descargado haciendo uso del botón *Descargar* o *volver a inicio* usando el otro botón para seguir navegando por la aplicación web.

Aviso: Una vez abandonada esta pantalla el archivo generado es deshechado por lo que no se puede volver a recuperar. Por esto, instamos a los usuarios a registrarse para mantener un historial de archivos generados.

A.2 Conversión con registro

En este apartado se recoge los distintos casos de navegación por la aplicación web de un usuario que quiere registrase para guardar su historial de archivos traducidos.

A.2.1. Inicio

Para empezar, nos enfrentamos al menú de inicio. Como se puede ver en la imagen, el diseño es bastante sencillo e intuitivo. Se encuentran tres botones en la parte central: *Iniciar sesión; Registrarse; Usar sin registro.* En los siguientes apar-

Figura A.3: Inicio de la aplicación web

tados se detallan los botones de inicio de sesión y registro. Para consultar las funcionalidades del tercero, acude a la primera parte de este anexo.

A.2.2. Registro

Tras pulsar en el menú de registro, se nos abre la siguiente página, donde debemos introducir nuestro usuario deseado (la aplicación arrojará un error si este nombre de usuario ya está registrado en el sistema), nuestro email y nuestra contraseña.

Figura A.4: Registro de la aplicación web

Aunque la aplicación te avisa si no lo haces correctamente, presta especial atención al formato del email y a la política de contraseñas, que establece que la contraseña debe contener entre 6 y 16 caracteres.

A.2.3. Inicio de sesión

En esta pantalla se introducen directamente el nombre de usuario o email y la contraseña para iniciar sesión y manejar tus archivos. Otras opciones que te ofrece

Figura A.5: Inicio de sesión en la aplicación web

la aplicación es *Registrarse*, en el caso de que aún no hayas registrado tu usuario en el sistema y la opción de *recuperar contraseña*, detallada a continuación.

A.2.4. Recuperación de contraseña

En el caso de que hayas olvidado la contraseña de inicio de sesión, hay dos opciones. La primera, la obvia, pasa por introducir el correo electrónico que hayas registrado en el sistema. Una vez enviado, la aplicación te asignará una nueva contraseña y te la enviará por email. Para hacer efectiva esta opción necesitarás, por supuesto, seguir teniendo acceso al correo electrónico usado para el registro.

Figura A.6: Recuperar contraseña en la aplicación web

La segunda opción es enviar un correo electrónico al administrador del sistema con datos identificativos que recuerdes de tu cuenta, por lo general: nombre de usuario, correo electrónico, antiguas contraseñas utilizadas, nombre de los exámenes generados... y todo lo que se te ocurra. Si el administrador considera que tu petición es legítima, se te asignará una nueva contraseña.

Una vez recuperada la contraseña, podrás iniciar sesión (apartado A.2.3) y acceder a tu perfil personal (apartado A.2.5)

Recomendación: Al asignarte una nueva contraseña, nuestro consejo es que la cambies por una que puedas recordar fácilmente y que sea única. Las contraseñas que genera la aplicación son bastante difíciles de adivinar, pero siempre va a ser más segura una no generada aleatoriamente.

Este cambio se puede hacer desde la pantalla de cambiar contraseña, accediendo desde el perfil personal (apartado A.2.5)

Figura A.7: Cambiar contraseña de la aplicación web

A.2.5. Perfil personal

La estructura básica de esta página son tres botones que te permiten: *Ver archivos; Subir nuevo; Cambiar contraseña*.
El primero te permite consultar tu historial de archivos generados (apartado A.2.8).
A este apartado también se puede acceder haciendo scroll hacia abajo en la misma página del perfil.
El segundo te lleva a la subida de archivos para traducir (apartado A.2.6)
Por último, el tercer botón te permite cambiar una contraseña (apartado A.2.4)

Figura A.8: Perfil personal de la aplicación web

A.2.6. Conversión

A la hora de subir un archivo debemos hacerlo desde este formulario ubicado en la página de Inicio. El formulario es el mismo que si usamos la página sin registrarnos, pero si lo hacemos desde nuestro usuario, los archivos generados (tanto el .txt con el lenguaje de marca, como el .xml convertido) se guardarán en nuestro historial de archivos (apartado A.2.8) para ser recuperados posteriormente.

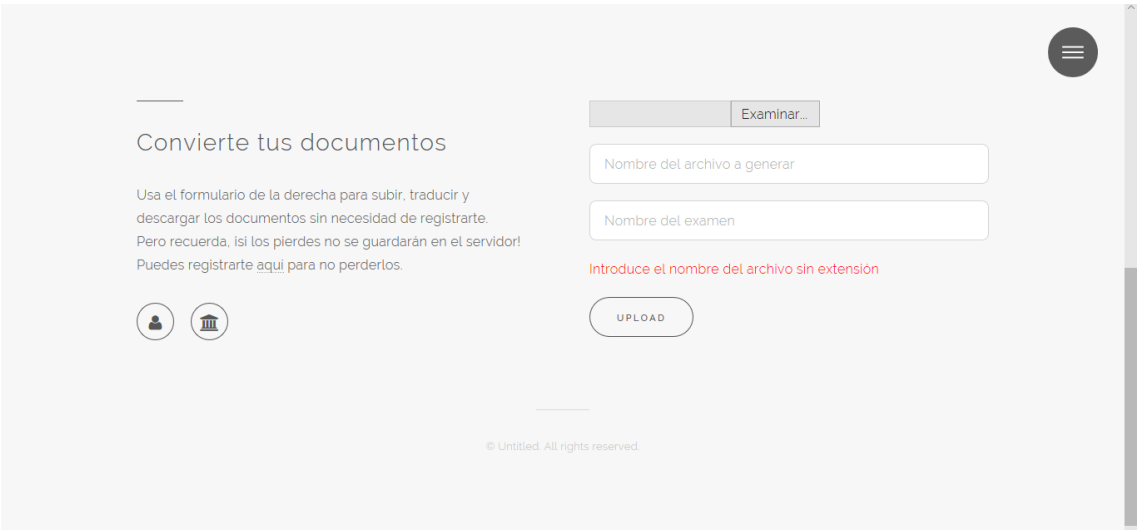


Figura A.9: Formulario de conversión de la aplicación web

Por seguridad, el tamaño máximo de archivo permitido es de 8MB. Realizando pruebas hemos estimado que este tamaño límite permite generar más de 4000 preguntas por archivo, por lo que debería ser suficiente.

Con el botón *Examinar...* buscamos y seleccionamos un archivo .txt de nuestro ordenador, smartphone o tablet. En el campo de texto de abajo podemos introducir el nombre del archivo que vamos a generar y en el otro campo podemos establecer el título del examen que aparecerá una vez subido a Sakai o la plataforma docente correspondiente.
Para mayor comodidad del usuario, estos dos campos se autorrellenan con el nombre del fichero que se está subiendo, pero pueden ser cambiados a posteriori.

Nota: Tal y como avisa el formulario, no es necesario introducir la extensión .xml, la aplicación la añade automáticamente.

A.2.7. Manejo de errores

A continuación se detallan los posibles errores que pueden aparecer durante el manejo de la aplicación web. Se detallan los más comunes y que son controlados por el desarrollador. Por supuesto pueden aparecer otros errores por factores externos, como una mala conexión a Internet, una caída esporádica del servidor que aloja la aplicación y derivados.

Usuario duplicado

Un error muy típico a la hora de crear un usuario nuevo es que nuestro nombre de usuario sea muy genérico y otro usuario ya lo tenga asignado. Por esto se hace una comprobación en la base de datos antes de crear un nuevo perfil. La aplicación web, tras darle al botón de *Registrarse*, hará la comprobación y devolverá el mensaje de error.

Figura A.10: Usuario duplicado en la aplicación web

La única solución a este problema es elegir otro nombre de usuario. Para mayor comodidad, se guardan el usuario intentado y el email introducido.

Datos de login incorrecto

Al introducir el nombre de usuario o email y la contraseña, si estos no son encontrados en la base de datos se devuelve el siguiente error. Este error puede tener varias causas, que el usuario no recuerde su contraseña o que esté introduciendo el nombre de usuario o email incorrectos.

Figura A.11: Login incorrecto en la aplicación web

En el primer caso, el usuario puede hacer uso de la opción *Recuperar contraseña* (apartado A.2.4).
En el segundo caso no es posible recuperar la cuenta si el usuario no se acuerda del email que introdujo.
Para mayor seguridad, la aplicación no especifica si el error se encuentra en la contraseña o en el usuario o email.

Archivo generado con errores

La aplicación web es capaz de reconocer el error fatal que haría que el xml generado no se mostrara correctamente. Cuando intentas subir un archivo que contiene un error de cierre de pregunta, la aplicación manda un aviso y no genera el archivo nuevo. Como se puede ver en la captura siguiente, se muestra una alerta en el formulario indicando la línea en la que se ha omitido la etiqueta de cierre.

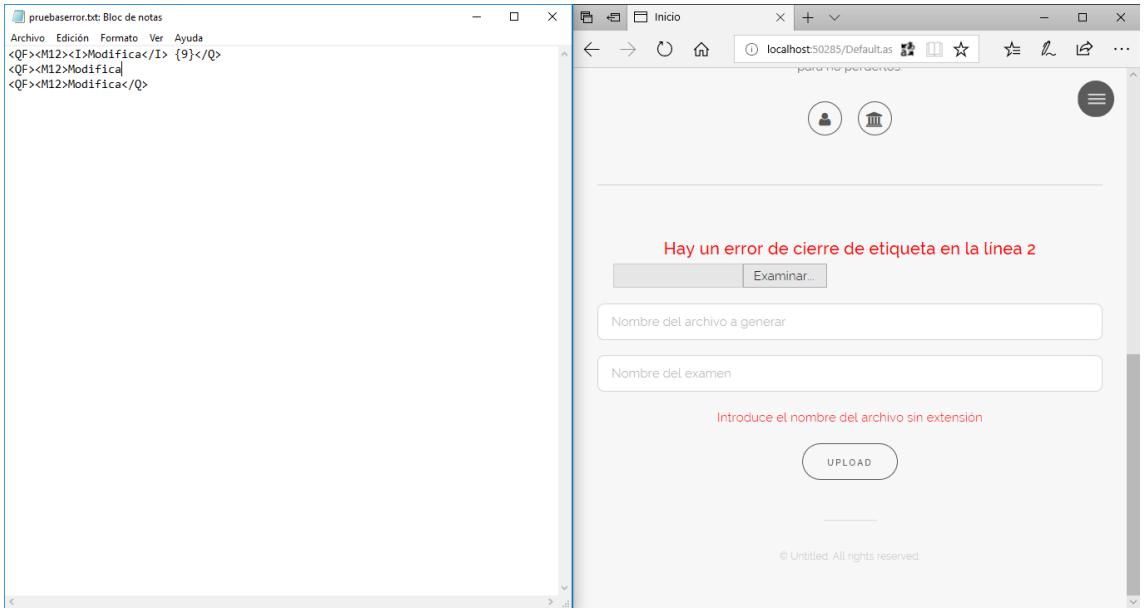


Figura A.12: Error de cierre en la aplicación web

A.2.8. Historial de archivos

Cuando un usuario registrado hace uso del convertidor de archivos, el sistema guarda un historial de sus archivos (tanto el *.txt* con lenguaje de marcas como el *.xml* generado en el estándar QTI/IMS). El funcionamiento es sencillo. El sistema guarda en una carpeta asociada todos los archivos y al consultar el historial de archivos los ordena mostrando el nombre que se le asignó y los enlaces de descarga, como se puede ver en la siguiente figura.

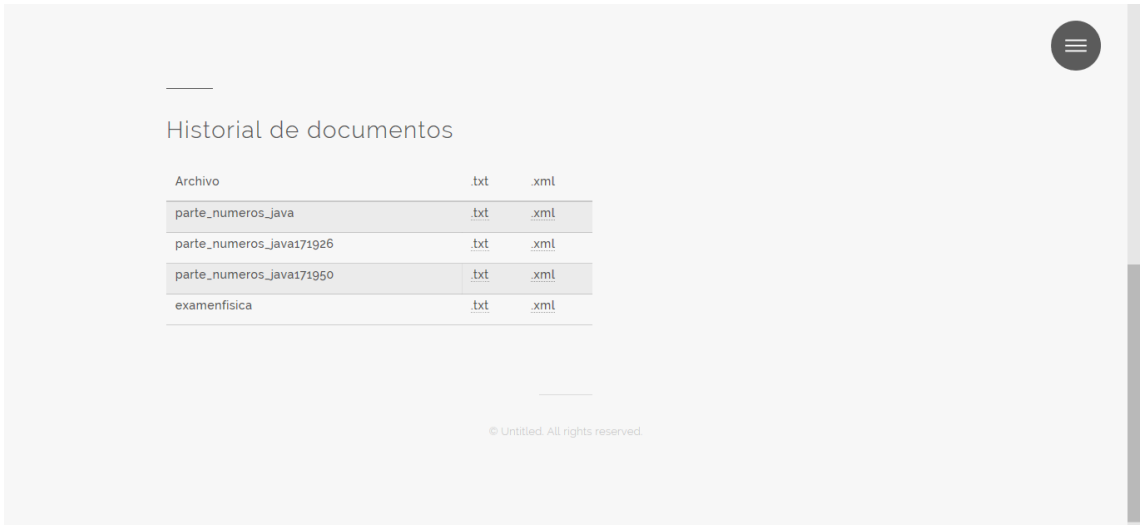


Figura A.13: Historial de archivos de la aplicación web

Los archivos se guardan en el perfil personal eternamente, sin límite de archivos subidos, siempre que el usuario no decida cancelar su cuenta.

A.2.9. Borrado de usuario

Un usuario podrá darse de baja en cualquier momento. Con esta acción se borrarán de la aplicación todos los datos personales del usuario, así como los archivos que haya generado.
Desde la parte inferior del perfil personal, mediante el hipervínculo *Darse de baja* se accederá al formulario de borrado de usuario, mostrado a continuación.

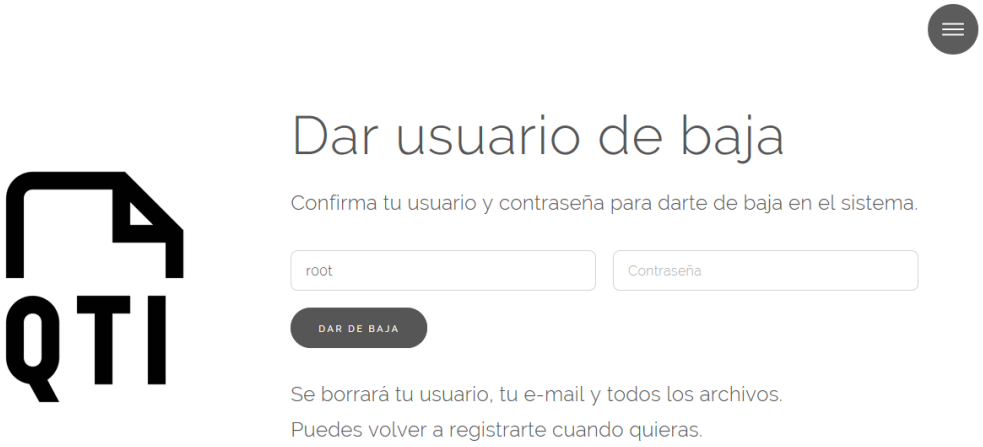


Figura A.14: Darse de baja en la aplicación web

Para confirmar el borrado, se debe introducir de nuevo la contraseña y pulsar sobre el botón *Dar de baja*, tras lo cual se volverá a pedir confirmación. Una vez aceptado en la ventana emergente, todos los datos serán borrados.
Si se desea, posteriormente, como se produce un borrado total de los datos, el usuario podrá volver a registrarse con el mismo nombre y email que tenía anteriormente.