



**ESCUELA POLITÉCNICA
SUPERIOR DE CÓRDOBA**
Universidad de Córdoba



TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Aplicación Conversor de Cuestionarios entre XML Test de Moodle y QTI/IMS de CANVAS

Autor

Diego Alejandro Higuera Grisales

Director(es)

Prof. D. Cristóbal Romero Morales

Junio, 2023



UNIVERSIDAD DE CÓRDOBA



Índice general

1. Introducción	1
1.1. MoodleXML	2
1.2. QTI/IMS	2
1.3. Partes del proyecto	3
2. Definición del problema	5
2.1. Identificación del problema real	5
2.2. Identificación del problema técnico	6
2.2.1. Funcionamiento	6
2.2.2. Entorno	6
2.2.3. Vida esperada	7
2.2.4. Ciclo de mantenimiento	7
2.2.5. Competencia	8
2.2.6. Aspecto externo	8
2.2.7. Estandarización	8
2.2.8. Calidad y fiabilidad	9
2.2.9. Programa de tareas	9
2.2.10. Seguridad	10
3. Objetivos	11
4. Antecedentes	13
4.1. Respondus	13
4.2. Plugins Moodle	13
4.3. QTIViewer/CCReader	14
4.4. GetMarked.ia	14
4.5. Código abierto	15
5. Restricciones	17
5.1. Factores Dato	17
5.2. Factores Estratégicos	17
5.2.1. Lenguaje de programación	18
5.2.2. Entorno de desarrollo	18
5.2.3. Desarrollo de la documentación	19
6. Recursos	21
6.1. Humanos	21
6.2. Hardware	21
6.3. Software	21

7. Análisis del sistema	23
7.1. Descripción funcional	23
7.1.1. Diagrama de casos de uso	23
7.1.2. Análisis del caso de uso 1: Realizar la conversión del archivo	25
7.1.3. Análisis del caso de uso 2: Seleccionar archivo origen	26
7.1.4. Análisis del caso de uso 3: Seleccionar formato	27
7.2. Descripción de la información	28
7.2.1. Estructura de la información	28
7.2.2. Estructura de los datos de IMS/QTÍ	36
7.3. Descripción de la interfaz (Plataformas)	52
7.3.1. Plataforma Moodle	52
7.3.2. Plataforma Canvas	53
7.3.3. Representación preguntas en ambas plataformas	55
7.4. Especificación de Requisitos	57
7.4.1. Requisitos Funcionales	57
7.4.2. Requisitos de Información	58
7.4.3. Requisitos de Interfaz	58
7.4.4. Requisitos No Funcionales	58
8. Diseño del sistema	59
8.1. Modelo de datos	59
8.2. Modelo arquitectónico	61
8.2.1. Clases	61
8.2.2. Estructuración del código	66
8.2.3. Flujos de datos	68
8.3. Diseño de Interfaz de Usuario	69
9. Pruebas	71
9.1. Pruebas de caja blanca	71
9.2. Pruebas de caja negra	71
9.3. Pruebas de integración	71
9.4. Pruebas de representación	72
9.4.1. Caso de prueba 01	73
9.4.2. Caso de prueba 02	75
9.4.3. Caso de prueba 03	77
9.4.4. Caso de prueba 04	79
9.4.5. Caso de prueba 05	81
9.4.6. Caso de prueba 06	83
9.4.7. Caso de prueba 07	85
9.4.8. Caso de prueba 08	87
9.4.9. Caso de prueba 09	89
9.4.10. Caso de prueba 10	91
9.4.11. Caso de prueba 11	93
9.4.12. Caso de prueba 12	95
9.4.13. Caso de prueba 13	97
9.4.14. Caso de prueba 14	98
9.5. Matriz de cumplimiento	99
10. Conclusiones	101

11.Futuras mejoras	103
12.Planificación del proyecto	105
A. Manual de usuario	111
A.1. Introducción	111
A.2. Instalación	111
A.2.1. Descargar aplicación	111
A.2.2. Lanzar el proyecto	112
A.2.3. Posibles errores	112
A.3. Uso de la aplicación web	113
A.3.1. Pantalla de Inicio	113
A.3.2. Ampliación	114
A.3.3. Conversión/Ejecución	115
A.3.4. Errores	116
A.4. Desinstalación	117
B. Manual de código	119
B.1. Introducción	119
B.2. Organización del código	120
B.3. Ficheros	122
B.3.1. Ficheros nuevos	122

Índice de figuras

7.1. Diagrama de casos de uso	24
7.2. Marca Moodle XML - raíz	28
7.3. Marcas question	29
7.4. Marcas en común	30
7.5. Marcas multichoice	32
7.6. Marcas true/false	32
7.7. Marcas shortanswer	33
7.8. Marcas numerical	34
7.9. Marcas matching	35
7.10. Modelo de objetos de datos de QTI	36
7.11. Marcas QTI/IMS - raíz	36
7.12. Marca assessment	37
7.13. Marca qtimetadata	37
7.14. Marca section	38
7.15. Árbol de esquema XML de ítem	38
7.16. Marcas item	39
7.17. Marcas itemmetadata	40
7.18. Árbol de esquema XML del elemento Itemmetadata	40
7.19. Árbol de esquema XML del elemento Presentation	41
7.20. Árbol de esquema XML del elemento Response_lid	41
7.21. Marcas presentation - Opción múltiple	42
7.22. Marcas presentation - Falso/verdadero	43
7.23. Marcas presentation - respuesta corta y numérica	43
7.24. Marcas presentation - emparejamiento	44
7.25. Marcas presentation - emparejamiento - columnas	45
7.26. Árbol de esquema XML del elemento Resprocessing	46
7.27. Marcas resprocessing - Opción múltiple	47
7.28. Marcas resprocessing - Opción respuesta múltiple	48
7.29. Marcas resprocessing - Verdadero/Falso	49
7.30. Marcas resprocessing - Respuesta corta	50
7.31. Marcas resprocessing - Respuesta numérica	50
7.32. Marcas resprocessing - Empaieramiento	51
7.33. Importación en Moodle	52
7.34. Importación en Moodle - Correcta	52
7.35. Importación en Moodle - Error	53
7.36. Listado preguntas importadas - Moodle	53
7.37. Importación en Canvas	53
7.38. Importación en Canvas - Correcta	54

7.39. Importación en Canvas - Error	54
7.40. Listado cuestionarios - Canvas	54
7.41. Representación - Opción Múltiple	55
7.42. Representación - Respuesta Múltiple	55
7.43. Representación - Verdadero/Falso	56
7.44. Representación - Respuesta Corta	56
7.45. Representación - Respuesta Numérica	56
7.46. Representación - Emparejamiento	57
8.1. Clase QTI2moodle	61
8.2. Clase moodle2QTI	63
8.3. Diagrama de clases	65
8.4. Esquema de estructuración del código	66
8.5. Diagrama de secuencia Convertir cuestionario	68
8.6. Prototipo - parte 1	69
8.7. Prototipo - parte 2	69
9.1. Representación Canvas - prueba 1	73
9.2. Conversión - prueba 1	74
9.3. Representación Moodle - prueba 1	74
9.4. Representación Canvas - prueba 2	75
9.5. Conversión - prueba 2	76
9.6. Representación Moodle - prueba 2	76
9.7. Representación Canvas - prueba 3	77
9.8. Conversión - prueba 3	78
9.9. Representación Moodle - prueba 3	78
9.10. Representación Canvas - prueba 4	79
9.11. Conversión - prueba 4	80
9.12. Representación Moodle - prueba 4	80
9.13. Representación Canvas - prueba 5	81
9.14. Conversión - prueba 5	82
9.15. Representación Moodle - prueba 5	82
9.16. Representación Canvas - prueba 6	83
9.17. Conversión - prueba 6	84
9.18. Representación Moodle - prueba 6	84
9.19. Representación Moodle - prueba 7	85
9.20. Conversión - prueba 7	85
9.21. Representación Canvas - prueba 7	86
9.22. Representación Moodle - prueba 8	87
9.23. Conversión - prueba 8	87
9.24. Representación Canvas - prueba 8	88
9.25. Representación Moodle - prueba 9	89
9.26. Conversión - prueba 9	89
9.27. Representación Canvas - prueba 9	90
9.28. Representación Moodle - prueba 10	91
9.29. Conversión - prueba 10	91
9.30. Representación Canvas - prueba 10	92
9.31. Representación Moodle - prueba 10	93
9.32. Conversión - prueba 11	93

9.33. Representación Canvas - prueba 11	94
9.34. Representación Moodle - prueba 10	95
9.35. Conversión - prueba 12	95
9.36. Representación Canvas - prueba 12	96
9.37. Validación estructura origen - prueba 13	97
9.38. Tipos de pregunta - prueba 14	98
12.1. Diagrama de Gantt	106
A.1. Manual de usuario - Inicio	113
A.2. Manual de usuario - Seleccionar Cuestionario	113
A.3. Manual de usuario - Elección de formato Origen-Fin	114
A.4. Manual de usuario - Tipo de preguntas soportadas	114
A.5. Manual de usuario - Ampliación	115
A.6. Manual de usuario - Informe de ejecución	115
B.1. Organización del código	120

Índice de tablas

7.1. Plantilla de Definición de Casos de Uso.	24
7.2. Caso de Uso 1: Realizar la conversión del archivo	25
7.3. Caso de Uso 2: Seleccionar archivo origen	26
7.4. Caso de Uso 3: Seleccionar formato	27
8.1. Datos - Equivalencias	60
9.1. Fases de Proyecto	99
12.1. Fases de Proyecto	105

Capítulo 1

Introducción

Los grandes cambios que hemos vivido en la sociedad y las personas en los últimos tiempos se han visto reflejados tanto en nuestros hábitos, como formas de consumir formación. En concreto el impulso del e-learning y la formación online, superando los obstáculos de tiempo y espacio, donde se utilizan diferentes plataformas como herramientas de aprendizaje, las cuales han revolucionado la forma en que educadores y estudiantes interactúan y colaboran en entornos educativos virtuales. Entre las plataformas online para el aprendizaje más extendidas a nivel mundial se encuentran Moodle y Canvas LMS [1]. En este contexto, aunque la tecnología ha evolucionado rápidamente facilitándonos hacer varias cosas de nuestro día a día, existen desafíos como (*la evaluación para el aprendizaje*) y cómo implementarla en un ambiente online. Las evaluaciones son usadas para determinar el conocimiento de los estudiantes y su dominio en un tema, así como también para identificar áreas de mejora y así poder adaptar la clase o hacer cursos personalizados.

Una de las prácticas mas usadas para la evaluación online son *los cuestionarios*, esta evaluación se realiza de forma dinámica gracias a los ordenadores que pueden realizar la tarea de selección de preguntas y la posterior evaluación en un instante. Una característica única de los cuestionarios en línea es que, el orden de las preguntas y las opciones de respuestas se pueden presentar de forma aleatoria, para que los estudiantes no reciban el mismo cuestionario. Cualquier cuestionario usado en una clase tradicional puede ser fácilmente adaptado en un cuestionario en línea. Las actividades de calificar los cuestionarios y comparar el desempeño de los estudiantes son muy fáciles de hacer, ya que se llevan a cabo de forma automática por el sistema. Además, los educadores tienen la opción de proporcionar aclaraciones adicionales, instrucciones o explicaciones detalladas en forma de comentarios junto a cada pregunta del cuestionario o cada opción de respuesta. Estos comentarios personalizados pueden ayudar a los estudiantes a comprender mejor el propósito de la pregunta, brindar orientación adicional sobre cómo abordarla y ofrecer explicaciones claras sobre las opciones de respuesta. [2, 3]

Las preguntas de tipo cuestionario pueden tener diferentes estructuras atendiendo a la forma de solicitar la respuesta del usuario y a como e muestra la información o datos de la pregunta. Existen diferentes tipos de cuestionarios, entre los mas utilizados son, verdadero o falso, identificar una o varias respuestas correctas, rellenar espacios en blanco , unión de columnas (emparejamiento), respuestas numéricas entre otros tipos. Los usuarios pueden organizar estas preguntas en bancos de preguntas y utilizarlos para crear pruebas personalizadas. Estos cuestionarios son creados para ser usados en una plataforma o sistema educativo de forma que en muchas ocasiones no pueden ser reutilizados en otras plataformas, ya que no existe un único estándar. Es este el caso vamos hablar de *MoodleXML* de Moodle y *QTI/IMS*, un formato más estandarizado usado en Canvas LMS, aunque esto puede ser utilizado no sólo en Canvas, sino también en otros LMS's a nivel mundial.[4]

1.1. MoodleXML

Moodle XML es un formato de archivo utilizado en el sistema de gestión del aprendizaje Moodle. Moodle es un sistema de gestión del aprendizaje de código abierto que brinda una amplia gama de herramientas para facilitar la enseñanza y el aprendizaje en línea, principalmente en entornos educativos. El formato Moodle XML se utiliza para importar y exportar datos de Moodle, lo que permite a los usuarios compartir contenido, actividades y configuraciones entre diferentes instalaciones de Moodle o con otros sistemas compatibles con el formato XML.

El archivo Moodle XML contiene una estructura jerárquica de datos que representa el contenido de un curso o elementos específicos dentro de un curso, como actividades, cuestionarios, recursos y configuraciones. Puede incluir información sobre el nombre, descripción, valoraciones y otros atributos relacionados con los elementos del curso. Los archivos Moodle XML se pueden crear y editar utilizando herramientas específicas de Moodle, como el editor de actividades de Moodle o mediante herramientas de creación de contenido compatible con Moodle XML. [5]

1.2. QTI/IMS

El formato QTI/IMS es un estándar de intercambio de datos utilizados en sistemas de evaluación educativa en línea. QTI significa *Interoperabilidad de Pruebas y Cuestionarios* en inglés, e IMS (IMS Global Learning Consortium) es una organización sin fines de lucro que desarrolla estándares abiertos para la educación y la tecnología. Canvas es compatible con el formato QTI/IMS (IMS QTI), lo que significa que importa y exporta preguntas y

pruebas en este formato. Canvas es un sistema de gestión del aprendizaje (LMS) utilizado por muchas instituciones educativas y organizaciones para la creación, entrega y gestión de cursos en línea.

El formato QTI/IMS utiliza XML (Extensible Markup Language) como lenguaje de marcado para describir la estructura y los atributos de las preguntas y pruebas. Esto permite que el contenido se presente de manera clara y estructurada, lo que facilita su comprensión y reutilización en otras herramientas compatibles con el formato QTI/IMS sin perder su estructura y características. Canvas cuenta con su herramienta de banco de preguntas (Question Bank), que permite a los usuarios crear preguntas y pruebas. [6]

1.3. Partes del proyecto

El proyecto en su totalidad estará formado por 3 manuales: el manual técnico, un manual de usuario y el manual de código.

Este, en concreto, es el manual técnico, documento encargado de explicar detalladamente desde el problema en cuestión, hasta la solución que se le va a dar, pasando por los recursos que se van a utilizar para ello. Su estructura se puede dividir en cinco partes, que a su vez se segmentarán en diversos capítulos:

- **Parte I: Introducción al proyecto.** Aspectos generales del proyecto.
 - *Capítulo 1: Introducción.* Presentación breve del problema que se pretende solucionar y su origen, mostrando las definiciones de aquellos términos que se utilizan frecuentemente a lo largo del manual.
 - *Capítulo 2: Definición del problema.* Definición de los problemas real y técnico de forma clara y concisa.
 - *Capítulo 3: Objetivos.* Relación de los objetivos que se pretenden alcanzar en la aplicación, explicando las funciones que debe cumplir lo que se diseña y apuntando las posibles vías de solución.
 - *Capítulo 4: Antecedentes.* Información relativa a aquellos métodos que tratan de resolver el problema de distribución en planta y los proyectos realizados anteriormente.
 - *Capítulo 5: Restricciones.* Limitaciones en el ámbito del diseño que condicionan la elección de una u otra alternativa, distinguiendo entre los factores dato y los estratégicos.
 - *Capítulo 6: Recursos.* Recursos humanos que intervienen en la elaboración del proyecto, y recursos materiales de hardware y software disponibles que se utilizan para llevarlo a cabo.

- **Parte II: Análisis.** Especificación de lo que debe hacer la aplicación desarrollada.
 - *Capítulo 7: Análisis del sistema.* Análisis del funcionamiento que deberá tener el sistema, la información a manejar, lo que deberá ofrecer la interfaz y los requisitos a satisfacer.
- **Parte III: Diseño.**
 - *Capítulo 8: Diseño del sistema*
- **Parte IV: Pruebas y resultados**
 - *Capítulo 9: Pruebas.* Pruebas a las que ha sido sometido el software durante su desarrollo y pruebas de rendimiento cuando se ha finalizado.
- **Parte V: Conclusiones y Futuras Mejoras**
 - *Capítulo 10: Conclusiones.* Mirada hacia atrás para determinar qué objetivos se han cumplido, cuáles no, y el porqué.
 - *Capítulo 11: Futuras mejoras.* A pesar de que inicialmente todo proyecto trata de ser ambicioso y conseguir un software inmejorable, esto casi nunca se cumple, por lo que este capítulo intentará ser una crítica constructiva para mejorar lo realizado y ampliarlo en el futuro.
 - *Capítulo 12: Distribución temporal.* Plan de trabajo y calendario seguido para el desarrollo del proyecto.

Capítulo 2

Definición del problema

En este apartado se va definir con claridad el problema a resolver. En primer lugar, se identificará el *Problema Real* al que nos enfrentamos desde un punto de vista externo (de un usuario) y de manera superficial. Y, en segundo lugar, se identificará el *Problema Técnico* desde una perspectiva más centrada en su desarrollo mediante la técnica conocida como PDS (*Product Desing Specification*).

2.1. Identificación del problema real

Muchos componentes de preguntas pueden ser exportados en el formato XML de Moodle, porque muchas universidades ya han implementado el conocido LMS Moodle y tienen muchos componentes de preguntas en él. Es natural que esos componentes de preguntas se compartan entre las universidades [7]. Los usuarios de la próxima generación de LMS (por ejemplo, Canvas) también deben desear compartir los componentes de las preguntas. Por un lado Canvas exporta sus cuestionarios en formatos IMS/QTI, pero Moodle no acepta la importación de este formato. Moodle por su lado, recomienda el Formato Moodle XML para importar preguntas, ya que éste permite que se importen la mayor cantidad de datos de la pregunta [8]. Por lo que se necesita una herramienta de transformación entre estos formatos, para que los LMS puedan ser mas colaborativos e interactivos, mejorando la calidad de los contenidos y las evaluaciones en el proceso de aprendizaje utilizando cualquier LMS. Así éste Trabajo de Fin de Grado se centrará en el desarrollo de una *aplicación conversora de formatos*, que permita a cualquier profesor exportar e importar los cuestionarios online entre ambas plataformas de aprendizaje y así poder llevar a cabo la reutilización de los bancos de preguntas de otros formatos, así como las preguntas generadas en diferentes formatos (y por lo consiguiente en diferentes plataformas), de forma que se almacene toda la información necesaria para su reutilización.

2.2. Identificación del problema técnico

Siguiendo el esquema de la PDS, a continuación, se expondrán los aspectos técnicos del proyecto con la intención de obtener una descripción formal del problema a resolver.

2.2.1. Funcionamiento

Nos vamos a centrar únicamente en el funcionamiento del módulo de representación. Así pues, se comportará de la siguiente manera:

1. Una vez la aplicación haya generado la conversión, podrá importar directamente el cuestionario en la plataforma correspondiente.
2. Cada cuestionario tendrá su estructura distinta según el tipo, con sus respectivos nombres, marcas y flujo de respuestas.
3. Se guardará las conversiones en una carpeta que estará en la misma localización donde se encuentra el fichero a convertir.
4. Se dará al usuario la opción de elegir el cuestionario a convertir, y cual será el formato final, de QTI/IMS a MoodleXML ó de MoodleXML a QTI/IMS.
5. Debe realizar un tratamiento de errores que evite salidas inesperadas del sistema, evitando la generación de ficheros de salida incoherentes.
6. Y debe ser fácil de usar.

2.2.2. Entorno

En el análisis del entorno de la aplicación que se pretende desarrollar se tendrá en cuenta los siguientes puntos de vista principalmente: entorno de programación, entorno software, entorno de usuario y entorno físico o de trabajo.

- *Entorno de programación:* La aplicación será accesible desde el escritorio, es decir, no hará falta el acceso a Internet ni a navegador. El lenguaje de programación que se utilizará en la implementación es Python haciendo uso de librerías simples y eficientes para analizar y crear datos XML, como lo son `xml.etree.ElementTree` ó `lxml.objectify`.
- *Entorno Software:* Para el correcto funcionamiento de la aplicación serán necesarios los siguientes componentes software:
 - Python 3.7

- Se recomienda la utilización como sistema operativo cualquier distribución estable del sistema operativo Linux ya que es donde se ha desarrollado el programa, aunque se podrá también ejecutar en Windows y Mac.
- *Entorno de Usuario:* La aplicación que se desarrollará deberá ser lo más intuitiva posible, de modo que pueda ser utilizada por todo tipo de usuarios, aunque no posean conocimientos informáticos.
- *Entorno físico o de trabajo:* Para ejecutarlas, los usuarios deben descargar sus archivos e instalarlos en su máquina. Su instalación no requiere grandes requisitos ya que no ocupa demasiado espacio, pero para su ejecución debe de tenerse en cuenta que debe tener espacio suficiente para guardar los ficheros a convertir y los ficheros finales, convertidos.

2.2.3. Vida esperada

Como se ha comentado, al tratarse de formatos de cuestionarios dependientes de las plataformas que los usan, la vida esperada estará ligada a la aceptación de la estructuras de dichos formatos en las plataformas o a futuras versiones de estos. Por tanto hemos de tener en cuenta que mientras se siga avanzando el tema en cuestión, nos puede llevar a cambiar la metodología que ha sido empleada en el desarrollo de este proyecto, en un plazo de tiempo que puede ser breve o extenso.

2.2.4. Ciclo de mantenimiento

En caso de que el módulo se considere de utilidad durante un período largo de tiempo este deberá ser llevado a cabo por programadores informáticos y se hará atendiendo a las necesidades generadas, pudiendo ser de tres tipos:

- *Perfeccionamiento:* Se basará en mejorar los aspectos que se consideren oportunos o crear nuevas funcionalidades a la aplicación, requeridas por los usuarios del sistema o por los autores de cursos. Estas mejoras pueden consistir en aumentar tipos de cuestionarios a convertir, facilitar el mantenimiento del sistema para posibles cambios futuros, etc.
- *Adaptación:* Conjunto de actividades que se realizarán para adaptar la aplicación al entorno tecnológico, como se comentaba en el apartado anterior, adaptarlo a posibles cambios en el formato de las estructuras que usa para generar las conversiones.
- *Corrección:* Corregir errores que puedan aparecer en su uso diario, no descubiertos hasta el momento, como pueden ser fallos de procesamiento o de implementación.

2.2.5. Competencia

El problema que abordamos es conocido por aquellos usuarios de ambas plataformas que han tenido que rehacer los cuestionarios ya que existen pocas soluciones efectivas ó de pago. Una máxima en el desarrollo de software es no reinventar la rueda, por eso nuestra idea no yace en aportar algo que ya existe; nuestra idea es ofrecer algo que destaque y cumplir con los requisitos y expectativas del usuario de manera efectiva y eficiente. Pensamos que los cuestionarios online son un elemento muy potente para la evaluación de un alumno, eso unido a la posibilidad de crear varios tipos de preguntas y poderlas usar en diferentes plataforma se reflejará en una experiencia de usuario que marcará la diferencia.

En el punto de antecedentes se mencionarán algunas herramientas similares que hayan servido para llevar a cabo este proyecto.

2.2.6. Aspecto externo

En cuanto a la interfaz gráfica de usuario (GUI) se pretende que tenga un diseño atractivo, intuitivo y sencillo de manejar, de forma que el usuario no tenga que leer el manual de usuario ya que los botones seguirán un patrón coherente en toda la aplicación de escritorio, expresando así cada uno su función clara y concisamente, para poder obtener las conversiones sin muchas complicaciones, que es donde radica la importancia de este proyecto.

2.2.7. Estandarización

El visualizador está programado en Python haciendo uso de la librería *tkinter*. Librería bastante extendida y documentada, que no debe plantear, en principio, ningún problema. El código fuente será lo suficientemente claro y legible, estando correctamente documentado para que en un futuro cualquier programador pueda entenderlo y realizar los cambios y/o mejoras que crea convenientes. Los formatos que vamos a convertir, ambos están estandarizados, por un lado MoodleXML, este formato se basa en XML (eXtensible Markup Language) este lenguaje está completamente estandarizado por el W3C (World Wide Web Consortium), por lo que tampoco deberá suponer ningún problema para la ejecución. Además se aplican otros como: SCORM (Sharable Content Object Reference Model) y IMS Common Cartridge. Por otro lado, QTI es un estándar de IMS (Interoperability Standards for Educational Technology) que se utiliza para describir preguntas y pruebas en formato electrónico. QTI permite la creación y entrega de evaluaciones en línea y su posterior evaluación automática. Se aplican otros como IMS Content Packaging o IMS Metadata.

2.2.8. Calidad y fiabilidad

El objetivo es garantizar un alto nivel de excelencia y confiabilidad llevando a cabo exhaustivas pruebas, con diferentes tipos de preguntas, para identificar posibles puntos o componentes críticos, con el fin de minimizar la probabilidad de errores durante el funcionamiento del software. En caso de que los usuarios introduzcan errores (por ejemplo, con ficheros no validos), se busca gestionarlos adecuadamente.

2.2.9. Programa de tareas

Fase Inicial

A lo largo de esta fase se llevarán a cabo las actividades relacionadas con el estudio para la elaboración del proyecto, que son:

1. Estudio del lenguaje de desarrollo python.
2. Estudio del estándar XML.
3. Estudio del estándar QTI
4. Estudio de las librerías xml.etree.ElementTree y objectify para el manejo de ficheros de lenguajes de marca.
5. Estudio de sistemas educativos y aplicaciones gestoras de cuestionarios.

Fase de Diseño y Desarrollo

En esta fase de ingeniería se procederá al diseño y desarrollo de la solución software, y entre las tareas se encuentran:

1. Estudio de las variables que participan en el sistema:
 - a) Variedad de estructuras XML.
 - b) Variedad de tipos de preguntas.
 - c) Parámetros modificables.
 - d) Verificación y conservación de la información de los cuestionarios.
2. Diseño y desarrollo del sistema.
3. Programación de las funcionalidades.
4. Integración de las funcionalidades en la interfaz
5. Comprobar que el formato resultante es válido

Fase de Prueba y Documentación

En esta fase se llevarán a cabo las actividades correspondientes a las pruebas de la solución desarrollada. Estas pruebas se pueden clasificar en dos categorías.

1. Prueba de Unidad o de Caja blanca: Este tipo de pruebas se realizará durante el periodo que se dedique a la codificación de la aplicación, en la que irán surgiendo diversos errores y se deberán ir corrigiendo mediante la aplicación de pruebas de este tipo.
2. Prueba Funcional o de Caja negra: Este tipo de pruebas se centran en el estudio de la especificación del software, análisis de las funciones que debe realizar, de las entradas y de las salidas.

El resultado de las pruebas nos permitirá la depuración de la solución para obtener una versión final y depurada del prototipo. Finalmente, se desarrollará la documentación correspondiente a la Memoria del Proyecto.

2.2.10. Seguridad

Para el apartado de seguridad se aplican los siguientes criterios

- El programa se podrá ejecutar desde cualquier ordenador con sistema operativos como Windows, Linux y Mac.
- Al ser una aplicación de escritorio, los ficheros a convertir y los convertidos, estarán almacenados localmente en el dispositivo que ha ejecutado el programa, por lo que el nivel de privacidad de los datos será establecido por el propio usuario, quien se encargará de proteger el acceso a ficheros o directorios bajo su juicio.
- Incluye una validación de ficheros de entradas de datos, de extensión xml y verificación de permisos de lectura.
- Actualizaciones y parches, es decir, la seguridad de la aplicación también implica mantenerla actualizada con las últimas correcciones de seguridad y parches. Esto implica seguir buenas prácticas de gestión de versiones y poder permitir futuras actualizaciones periódicas para corregir vulnerabilidades conocidas.
- Se obviarán los medios de seguridad contra copias, ya que se trata de software de libre distribución.

Capítulo 3

Objetivos

De acuerdo a la identificación real y técnica del problema, que se ha realizado en el capítulo anterior, a continuación se expondrán todos los objetivos funcionales que se pretenden alcanzar con el desarrollo de este proyecto. El objetivo principal de este proyecto es realizar una aplicación de escritorio que convierta un tipo de cuestionario de un formato XML, el mismo tipo de cuestionario pero en otro formato XML, (MoodleXML a QTI/IMS, ó viceversa) favoreciendo así el uso de cuestionarios en las plataformas que usan estas estructuras. A continuación, vamos a especificar dicho objetivo descomponiéndolo en una serie de subobjetivos que será necesario cumplir para llevar a cabo el proyecto:

- Estudio del estándar MoodleXML y QTI/IMS.
- Estudio de la librería *xml.etree.ElementTree* y *objectify*.
- Estudio de otras herramientas software para el desarrollo del prototipo. Esto incluye para el desarrollo de la interfaz.
- Desarrollar una aplicación sencilla que permita cubrir la necesidad de conversión de cuestionarios en los formatos mencionados.
- La aplicación deberá generar y entender los cuestionarios bien formados de las diferentes estructuras a los que convierta.
- Permitir que el usuario pueda elegir el cuestionario a convertir, cual será el formato final y que el archivo se guarde en el dispositivo.
- Se busca crear una aplicación con una estructura modular que facilite la incorporación de nuevas funcionalidades en el futuro. La aplicación debe ser diseñada de manera escalable, de modo que sea posible añadir módulos adicionales que resulten útiles para el proceso de conversión, y que no hayan sido contemplados en este proyecto.

- Creación de una documentación completa y clara, en la cual se editará el manual de usuario del software desarrollado, así como las memorias y documentación técnica necesaria.

Capítulo 4

Antecedentes

Aunque existen en el mercado multitud de aplicaciones relacionadas con la conversión de formatos, en este apartado, se mencionarán diferentes aplicaciones comerciales relacionadas con el tema a tratar, que han servido de ayuda para concebir una idea más clara del resultado final que se quiere obtener, si nos centramos principalmente en la característica de conversión de los formatos MoodleXML y QTI/IMS, en ambos sentidos. A continuación, se mostrará un breve resumen de su funcionalidad:

4.1. Respondus

Respondus es una compañía que ofrece software y soluciones para la creación, administración y seguridad en exámenes en línea. Es un programa de pago con 30 días gratuitos, pero cuando se probó, solo se obtuvieron resultados insuficientes, ya que convertía un formato QTI a un formato *.doc*, y no especificaba cual era la respuesta correcta. [9]

4.2. Plugins Moodle

Existen algunos plugins en la comunidad Moodle que realizan la importación de QTI, que desgraciadamente están ahora obsoletos y ya no se mantienen [10], como son los siguientes:

- *Questionmark QML Importer (Alpha)*: Es una extensión que permite importar cuestionarios y evaluaciones creados en dicho formato, se mantuvo hasta 2018. En nuestras pruebas con varias preguntas de opción múltiple en el estándar QTI 1.2, ninguna de estas preguntas pudo ser importada sin errores con la ayuda de este plugin. Es importante destacar que el término *Alpha* indica que el plugin se encuentra en una fase inicial de desarrollo y es posible que no cuente con todas las características completas o pueda presentar algunos errores.

- *moodle-qformat_imsqti21plugin*: Es una extensión diseñada para el sistema de gestión de aprendizaje Moodle. Este plugin permite la importación y exportación de cuestionarios y evaluaciones en el formato IMS QTI 2.1. Cuando probamos este plugin con las preguntas del estándar QTI 2.1, ni la importación ni la exportación de Moodle XML a QTI 2.1 funcionaron.

4.3. QTIViewer/CCReader

Es una herramienta de software que proporciona una interfaz la cual permite visualizar y leer contenido en formato QTI, lo que facilita la revisión y la interacción con el contenido evaluativo, pero no permite hacer una conversión entre formatos. [11]

4.4. GetMarked.ia

Es una plataforma en línea que se centra en varios objetivos relacionados con el e-learning. Ofrece funcionalidades para la creación, gestión y evaluación de exámenes. Mediante el uso de tecnología avanzada como la Inteligencia Artificial y el análisis de datos, GetMarked.ai automatiza y optimiza el proceso de evaluación, lo que permite ofrecer una solución eficiente y precisa. Esto se traduce en un ahorro de tiempo significativo y en la generación de resultados confiables para los usuarios.

- La creación y diseño de exámenes personalizados.
- La administración de exámenes en línea.
- La corrección y calificación automática de respuestas.
- La generación de informes y análisis detallados.
- La integración con sistemas de gestión del aprendizaje (LMS) y otras herramientas educativas.

GetMarked.ai se enfoca en brindar a educadores, instituciones académicas y organizaciones una forma eficiente de evaluar el aprendizaje, reduciendo la carga administrativa y mejorando la calidad y la eficacia de las evaluaciones. Esta también permite la conversión entre formatos, pero es de pago. [12]

4.5. Código abierto

Durante la investigación de herramientas nos encontramos con códigos en el repositorio online GitHub, y otros en páginas web, los cuales sirvieron de guía para realizar nuestro código.

- Generador de preguntas XML de Moodle: Es un módulo para Python. Con este módulo, podemos generar fácilmente un conjunto de preguntas de opción múltiple e importarlas a Moodle. [13]
- text2qti: Convierte archivos de texto sin formato basados en Markdown en cuestionarios en formato QTI (versión 1.2), que pueden ser importados por Canvas y otro software educativo, usando el lenguaje Python. [14]
- moodle2qti [15]
- Procesamiento de datos QTI en Python [16]

Capítulo 5

Restricciones

En este capítulo se expondrán todas las restricciones, o factores limitativos, existentes en el ámbito del diseño que condicionarán el desarrollo de nuestro proyecto. Estos factores limitativos, según su tipo, se pueden clasificar en dos grupos: *Factores Dato* y *Factores Estratégicos*.

5.1. Factores Dato

El problema que se ha planteado apenas presenta barreras que limiten las posibles soluciones, como pudieran ser las relativas al hardware, el software, los plazos, etc. Esto no quiere decir que sean inexistentes:

- Limitaciones en el plazo de entrega. El presente proyecto tiene el plazo máximo de entrega el 10 de Junio de 2023. Para esa fecha, se espera que el proyecto esté totalmente finalizado.
- Limitaciones técnicas. Estas restricciones pueden estar relacionadas con las capacidades técnicas del equipo o las especificaciones, debido a que anteriormente desconocía totalmente el problema presentado.
- Limitaciones económicas. El coste para llevar a cabo el proyecto es mínimo; debido a la utilización de software estándar disponible gratuitamente y hardware propio.

5.2. Factores Estratégicos

Siguiendo los objetivos descritos en el Capítulo 3, a continuación, identificaremos los factores estratégicos que, si bien condicionarán las distintas propuestas alternativas, pueden ser objeto de modificación o elección en uno u otro sentido.

5.2.1. Lenguaje de programación

Se ha optado por usar el lenguaje de programación Python, que a día de hoy, es un lenguaje de programación versátil y potente ya que cuenta con una gran cantidad de bibliotecas y módulos que facilitan la tarea de convertir formatos XML, por ejemplo, se hará uso de las siguientes bibliotecas estándares:

- `xml.etree.ElementTree` ya que proporciona una forma fácil de analizar y manipular documentos XML. [17]
- Biblioteca `lxml` la cual contiene el módulo `objectify`, el cual es altamente eficiente en el procesamiento de XML y XHTML.[18]

Además, Python es multiplataforma, escalable y tiene una comunidad activa [19]. Todo ello irá integrado en una aplicación programada en Python haciendo uso del módulo `tkinter`.

5.2.2. Entorno de desarrollo

El entorno de trabajo será un PC con Ubuntu 20.04.5 LTS, la versión de este sistema operativo cuenta con soporte a largo plazo, lo que la convierte en una opción estable y confiable. [20]

Para creación y modificación del código fuente se ha utilizado la herramienta *Visual Studio Code* para desarrollar la aplicación de escritorio, debido a que ofrece una amplia gama de características, herramientas y extensiones que pueden mejorar la eficiencia y la productividad en el proceso de desarrollo. Además VS Code tiene una excelente integración con Python, lo que proporciona soporte para la depuración de código Python directamente desde el editor, lo que facilita la identificación y solución de problemas en la aplicación de escritorio. [21]

Por último, para la aplicación de escritorio se ha usado *Tk*, una biblioteca de código abierto escrita en C, la cual Python incluye en su librería estándar, en el módulo `tkinter`, que permite interactuar con Tk para desarrollar aplicaciones de escritorio en Python de forma rápida, de modo que cualquier programador con una mínima base de Python puede comenzar rápidamente a crear aplicaciones gráficas profesionales y luego distribuirlas vía herramientas como `cx_Freeze` o `PyInstaller`, que se integran muy bien con Tk. [22]

En su momento, se decidió usar Python debido a que es el mismo lenguaje en el que estábamos programando las funcionalidades del conversor, y facilitaba enlazar la interfaz gráfica.

5.2.3. Desarrollo de la documentación

La documentación será creada utilizando LaTeX a través de la herramienta en línea Overleaf. Inicialmente, se consideró utilizar un editor de texto convencional como Microsoft Word, pero esta opción fue descartada al darnos cuenta de que no se podría alcanzar un nivel de control tan avanzado sobre el texto ni obtener resultados de la misma calidad que con el sistema elegido.

Para crear los diagramas necesarios que respalden el análisis y diseño de la aplicación bajo UML (Lenguaje Unificado de Modelado), utilizaremos la herramienta Draw.io en su versión web. Esta herramienta fue elegida debido a que es gratuita y proporciona la mayoría de los tipos de diagramas necesarios. Con ella, podremos representar gráficamente la estructura y las relaciones de la aplicación de manera efectiva. [23]

Capítulo 6

Recursos

Para llevar a cabo el desarrollo del proyecto se utilizarán una serie de recursos humanos, hardware y software los cuáles se detallarán a continuación

6.1. Humanos

El proyecto será realizado por el alumno de Ingeniería Informática Diego Alejandro Higuita Grisales, dirigido y coordinado por: Cristóbal Romero Morales, profesor perteneciente al departamento de Informática y Análisis Numérico de la Universidad de Córdoba.

Dado que este proyecto es un trabajo de fin de carrera, el estudiante encargado del desarrollo asumirá los roles de analista y programador. El director del proyecto tendrá la responsabilidad de orientar al estudiante durante su desarrollo y revisar periódicamente su trabajo para asegurar que avance de manera adecuada y correcta.

6.2. Hardware

El proyecto será desarrollado en los equipos propiedad del alumno.

- **Portátil:** Intel Core i7 8550U, up to 2000 MHz, 8 GB de RAM, Intel(R) UHD Graphics 620, SSD 256 GB.
- **Impresora:** Hp DeskJet 2720.

6.3. Software

En cuanto a los recursos software que se utilizarán para la realización del proyecto:

- Sistemas operativos usados:

Windows 10.

Linux (Ubuntu 20.04.4 LTS).

- Herramientas de desarrollo de la aplicación: Python 3, API de objectify y API de ElementTree.
- Herramienta para la edición de código: editor de código fuente Visual Studio Code (VS Code).
- Herramientas para el desarrollo de la documentación y diagramas:
 - \LaTeX . Sistema de composición de textos usado para realizar la documentación, bajo la herramienta de edición en línea Overleaf.
 - Draw.io. Aplicación web para la realización de diagramas (UML, ER, etc.).
 - Pencil. Software para crear el prototipo de la interfaz de usuario.

Capítulo 7

Análisis del sistema

En los capítulos anteriores se ha brindado una visión general del problema que se busca resolver, se ha descrito la aplicación que se va a desarrollar y se han establecido tanto el tipo de información que se manejará como la funcionalidad que debe proporcionar.

El objetivo de este apartado es realizar un análisis global del sistema presentando todos los requisitos que debe cumplir, es decir, se especificará técnicamente “Qué debe hacer” la aplicación. Los requisitos serán determinados en base al enunciado del problema, tal y como ha sido establecido en el Capítulo 2 de este documento. Esta etapa es de las más importantes en el desarrollo de un producto software, ya que tendrá un impacto significativo en la calidad del producto final.

El análisis de la aplicación se va a llevar a cabo mediante la utilización de una metodología orientada a objetos, ajustándose dentro de lo posible a la metodología UML en las especificaciones.

7.1. Descripción funcional

A continuación, se presentan las funcionalidades necesarias que el sistema debe cumplir para alcanzar los objetivos establecidos, mediante un diagrama de casos de uso y su especificación.

7.1.1. Diagrama de casos de uso

Los diagramas de casos de uso representan la interacción entre los usuarios (actores) y el sistema en desarrollo. Estos diagramas describen cómo los usuarios operan con el sistema y cómo los distintos elementos del sistema interactúan entre sí. Los casos de uso capturan el comportamiento deseado del sistema sin entrar en detalles sobre cómo se implementa ese comportamiento.

Aunque UML no proporciona una plantilla específica para definir casos de uso, es común utilizar una estructura estándar para documentarlos, a continuación se definirá la plantilla que será utilizada luego en el proceso de Análisis de los Casos de Uso.[Tabla 7.1]

Caso de Uso [ID]	Nombre del caso de uso
Actores	Actores que intervienen en el caso de uso(principales y secundarios)
Descripción	Descripción informal de los objetivos de uso.
Precondiciones	Condiciones que deben de cumplirse previamente para la correcta realización del caso de uso.
Escenario de éxito	Secuencia de pasos necesarios que deben seguir los actores para realizar la operación con éxito.

Tabla 7.1: Plantilla de Definición de Casos de Uso.

Por nuestra parte, nos ha parecido más interesante reflejar el comportamiento esperado del sistema en un único diagrama, ciñéndonos así al estándar UML. Esto permitirá tener una visión global de la aplicación. [Figura 7.1]

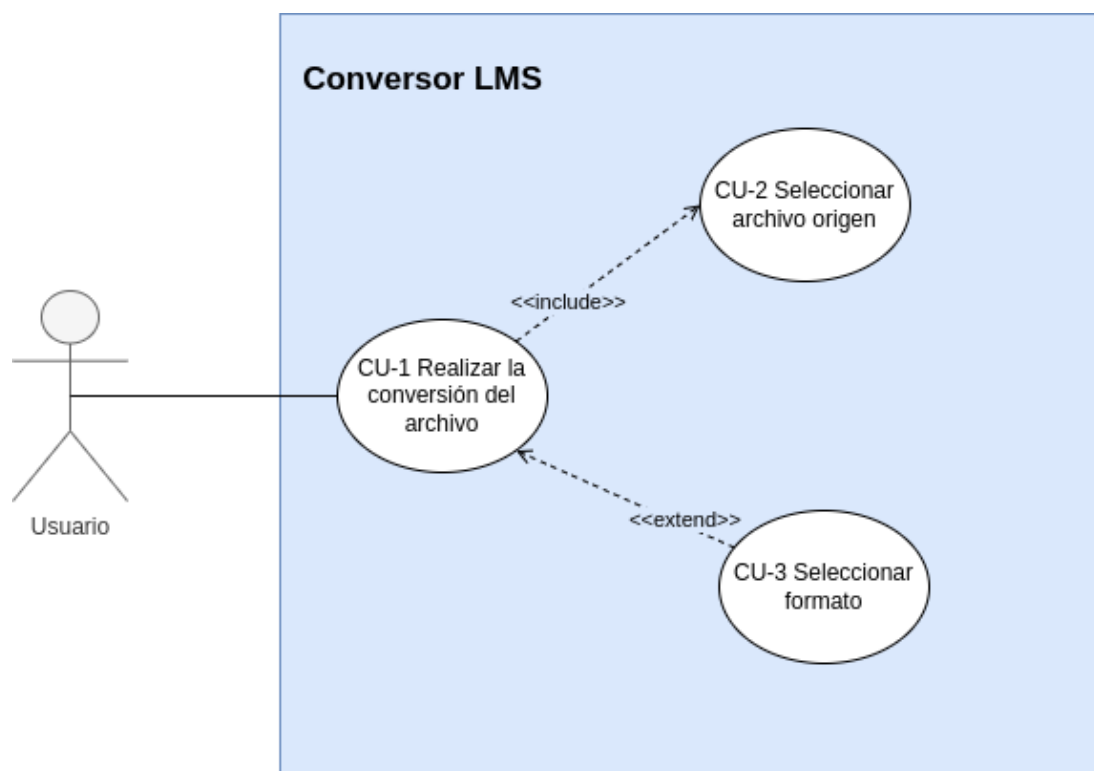


Figura 7.1: Diagrama de casos de uso

A continuación, procedemos con el análisis de los casos de uso que intervienen directamente en nuestro sistema.

- CU-1. Realizar la conversión del archivo.
- CU-2. Seleccionar archivo origen.

- CU-3. Seleccionar formato.

7.1.2. Análisis del caso de uso 1: Realizar la conversión del archivo

Caso de uso principal. Tabla 7.2

Caso de Uso [1]	Realizar la conversión del archivo
Actores	Usuario
Descripción	El usuario obtiene la conversión del fichero MoodleXML o QTI/IMS a partir otro fichero QTI/IMS o MoodleXML, correspondientemente.
Precondiciones	Debe tener guardado en el dispositivo el archivo que desea convertir con la extensión .xml.
Escenario de éxito	Flujo principal <ol style="list-style-type: none"> 1. Introducir el archivo que desea convertir 2. El usuario pulsa el botón <i>CONVERTIR</i>. 3. Obtener el documento XML con el formato destino.

Tabla 7.2: Caso de Uso 1: Realizar la conversión del archivo

7.1.3. Análisis del caso de uso 2: Seleccionar archivo origen

Cuando relacionamos los casos de uso [1] y [2] con un *include*, estamos diciendo que el primero (el caso de uso [1]) incluye al [2] (el caso de uso incluido). Es decir, sin el caso *Seleccionar archivo origen*, el caso *Realizar la conversión del archivo* no podría funcionar bien; pues no podría cumplir su objetivo. Tabla 7.3

Caso de Uso [2]	Seleccionar archivo origen
Actores	Usuario
Descripción	El usuario selecciona el fichero que pasará por el proceso de conversión
Precondiciones	Debe tener guardado en el dispositivo el archivo que desea seleccionar con la extensión .xml.
Escenario de éxito	Flujo principal 1. El usuario debe pulsar el botón <i>Selecione cuestionario</i> 2. Navegar por los ficheros del dispositivo 3. Seleccionar el fichero XML con el formato origen.

Tabla 7.3: Caso de Uso 2: Seleccionar archivo origen

7.1.4. Análisis del caso de uso 3: Seleccionar formato

Cuando relacionamos los casos de uso [1] y [3] con un *extend*, es debido a que hay situaciones en que el caso de uso de extensión (*Seleccionar formato*) no es indispensable que ocurra, y cuando lo hace ofrece un valor extra (extiende) al objetivo original del caso de uso base, y esto se debe a que, en cada conversión que se realice, no se debe estar eligiendo el formato, pues si en la conversión última coincide con los formatos origen y fin de la conversión del momento, no tendrá que ocurrir este caso de uso. Tabla 7.4

Caso de Uso [3]	Seleccionar formato
Actores	Usuario
Descripción	El usuario elige cual es formato origen y cual es el formato final
Precondiciones	No hay precondiciones
Escenario de éxito	Flujo principal 1. El usuario podrá pulsar el botón para establecer cual será el formato origen y el formato final

Tabla 7.4: Caso de Uso 3: Seleccionar formato

7.2. Descripción de la información

En este capítulo se procederá a describir la información manipulada por la aplicación, es decir, se analizará la estructura con la que será convertida y/o almacenada dicha información. Debemos recordar que los datos usados en el sistema utilizan el lenguaje de marcado XML, lo que significa que la información se estructurará en forma de árbol.

7.2.1. Estructura de la información

En este apartado se describirá las estructuras de los ficheros origen y final, es decir, la estructura desde la cual se obtendrá la información que será convertida y la estructura que posteriormente será almacenada en el dispositivo por la aplicación que se desea desarrollar. Se corresponde con las siguientes:

- Estructura de los datos de MoodleXML.
- Estructura de los datos de IMS/QTI 1.2.

A continuación se describen las estructuras de cada uno de ellos de forma detallada, la mayoría de información se obtendrá de las páginas oficiales de los formatos [5] [24], correspondientemente.

Estructura de los datos de MoodleXML

El archivo está rodeado por marcas (tags). Es importante asegurarse de que la marca xml solamente sea la primera línea del archivo. Una primera línea vacía, o marcas adicionales en la primera línea, confundirán al revisor XML de Moodle. [5] [Figura 7.2]

```
<?xml version="1.0" ?> <quiz>
```

```
.  
.   
.
```

```
</quiz>
```

Figura 7.2: Marca Moodle XML - raiz

- **<quiz></quiz>**: Es el elemento raíz del cuestionario y contiene todas las preguntas que forman parte de él, es decir, envuelven todo el cuestionario y sirven como contenedor principal.

Dentro de las marcas **<quiz>** hay varias marcas **<question>**. [Figura 7.3]

```
▼<quiz>
  ▶<question type="category">
    ...
  </question>
  ▶<question type="truefalse">
    ...
  </question>
  ▶<question type="shortanswer">
    ...
  </question>
  ▶<question type="multichoice">
    ...
  </question>
</quiz>
```

Figura 7.3: Marcas question

- **<question></question>**: Estas etiquetas encierran una pregunta individual y especifican el tipo de pregunta que se está utilizando. Por ejemplo, puede ser 'multichoice' para preguntas de opción múltiple. Soporta otros tipos como:
 - truefalse = verdadero/falso
 - shortanswer = respuesta corta
 - matching = emparejamiento
 - numerical = numérica
 - Entre otros.

Una de estas marcas **<question>** no representa una pregunta real (con un tipo de *category*), sino que se utiliza para asignar una *categoría* específica al grupo de preguntas relacionadas. Esta categoría puede ser útil al importar o exportar el cuestionario, ya que permite clasificar y organizar las preguntas en categorías predefinidas.

Cada marca **<question>**, tiene comúnmente las siguientes marcas como nodos hijos: [Figura 7.4]

```
▼<quiz>
  ▼<question type="[tipo_de_pregunta]">
    ▼<name>
      <text>Nombre de la pregunta</text>
    </name>
    ▼<questiontext format="html">
      <text>¿Cual es la respuesta a esta pregunta?</text>
    </questiontext>
    <!-- Otros elementos de la pregunta -->
    ▼<generalfeedback format="html">
      <!-- Retroalimentación general -->
    </generalfeedback>
    <defaultgrade>1.0</defaultgrade>
  </question>
</quiz>
```

Figura 7.4: Marcas en común

- **<name></name>**: Estas etiquetas contienen el nombre de la pregunta.
- **<questiontext></questiontext>**: Estas etiquetas encierran el enunciado de la pregunta
- **<generalfeedback></generalfeedback>**: Puede estar en formato HTML y Se utiliza para proporcionar una retroalimentación general que se muestra independientemente de si la respuesta del estudiante es correcta o incorrecta. Esto permite ofrecer información adicional al estudiante sobre la pregunta o el concepto evaluado, independientemente de su respuesta.
- **<defaultgrade></defaultgrade>**: Esta etiqueta define el valor de puntuación predeterminada para la pregunta.

Además, incluyen al menos una marca **<answer>** y/o dependiendo del tipo de pregunta, usarán otros tipos diferentes etiquetas que representan funciones o atributos de cada tipo de pregunta.

1. *Opción múltiple*: Las preguntas de Opción Múltiple tienen una marca **<answer>** para cada opción, la cual puede contener retroalimentación y ponderación del puntaje utilizando el atributo **fraction**. Además, una pregunta de Opción Múltiple tiene las siguientes marcas: [Figura 7.5]

- **<single></single>** (values: true/false): Esta etiqueta especifica si solo se puede seleccionar una respuesta. El valor "true" indica que solo se permite una respuesta.

- **<shuffleanswers></shuffleanswers>** (values: 1/0): Esta etiqueta indica si las opciones de respuesta deben mezclarse al mostrar la pregunta. El valor "1" significa que sí deben mezclarse.
- **<correctfeedback></correctfeedback>**: Este elemento se utiliza para proporcionar comentarios específicos cuando la respuesta del estudiante es correcta en una pregunta de opción múltiple. Puede incluir información adicional, detalles o comentarios positivos relacionados con la respuesta correcta.
- **<partiallycorrectfeedback></partiallycorrectfeedback>**: Representa el comentario que se muestra al estudiante cuando seleccione una respuesta que es parcialmente correcta. Este comentario de retroalimentación puede proporcionar información adicional o aclaraciones sobre la respuesta seleccionada.
- **<incorrectfeedback></incorrectfeedback>**: Representa el comentario que se muestra al estudiante cuando selecciona una respuesta incorrecta.
- **<answernumbering></answernumbering>** (valores permitidos: 'none', 'abc', 'ABCD' o '123'): Se utiliza para especificar el tipo de numeración que se utiliza a las opciones de respuesta en un cuestionario.

```

<single>false</single>
<shuffleanswers>true</shuffleanswers>
<answernumbering>abc</answernumbering>
▼<correctfeedback format="html">
  <text><p>Respuesta correcta.</p></text>
</correctfeedback>
▼<partiallycorrectfeedback format="html">
  <text><p>Respuesta parcialmente correcta.</p></text>
</partiallycorrectfeedback>
▼<incorrectfeedback format="html">
  <text><p>Respuesta incorrecta.</p></text>
</incorrectfeedback>
▼<answer fraction="50.0" format="html">
  <text><p>uno</p></text>
  ▼<feedback format="html">
    <text/>
  </feedback>
</answer>
▼<answer fraction="50.0" format="html">
  <text>dos</text>
  ▼<feedback format="html">
    <text/>
  </feedback>
</answer>
▼<answer fraction="0" format="html">
  <text>tres</text>
  ▼<feedback format="html">
    <text/>
  </feedback>
</answer>

```

Figura 7.5: Marcas multichoice

2. *Falso/verdadero*: Se dan dos marcas `<answer>` para la respuesta, una es verdadera y la otra es falsa. El atributo `<fraction>` de la marca de respuesta identifica cual opción es la correcta (100) y cual es la falsa (0). Ésta soportada la retroalimentación. El ejemplo siguiente muestra el formato cuando la respuesta correcta es verdadera y la falsa es errónea. [Figura 7.6]

```

▼<answer fraction="100" format="moodle_auto_format">
  <text>true</text>
  ▼<feedback format="html">
    <text><p>Aquí se escribe el feedback cuando responde verdad</p></text>
  </feedback>
</answer>
▼<answer fraction="0" format="moodle_auto_format">
  <text>false</text>
  ▼<feedback format="html">
    <text><p>Aquí se escribe el feedback cuando responde falso</p></text>
  </feedback>
</answer>

```

Figura 7.6: Marcas true/false

Este tipo de pregunta no contiene más tipos de etiquetas especiales.

3. *Respuesta corta*: El tipo de pregunta de respuesta corta soporta respuestas correctas alternas, cada una de ellas con su propia ponderación y retroalimentación. El formato Moodle XML usa solamente una marca `<answer>` para cada una de las respuestas correctas alternas. [Figura 7.7]. Además, una pregunta de respuesta corta, se añade la siguiente marca:

- `<usecase></usecase>` (values: true/false ó 1/0): Cuando se establece en true, indica que se debe hacer una coincidencia exacta, teniendo en cuenta tanto las letras mayúsculas como las minúsculas en las respuestas esperadas. Esto significa que las respuestas *Casa* y *casa* serán tratadas como respuestas diferentes. Por otro lado, cuando `<usecase>` se establece en false, se ignoran las diferencias de mayúsculas y minúsculas en las respuestas esperadas. Esto significa que las respuestas *Casa* y *casa* se consideraron como respuestas iguales.

```

<usecase>0</usecase>
▼<answer fraction="100" format="moodle_auto_format">
  <text>uno</text>
  ▼<feedback format="html">
    <text/>
  </feedback>
</answer>
▼<answer fraction="100" format="moodle_auto_format">
  <text>dos</text>
  ▼<feedback format="html">
    <text/>
  </feedback>
</answer>
▼<answer fraction="100" format="moodle_auto_format">
  <text>tres</text>
  ▼<feedback format="html">
    <text/>
  </feedback>
</answer>

```

Figura 7.7: Marcas shortanswer

4. *Respuesta numérica*: En este tipo de pregunta, se establece una o varias respuestas `<answer>` esperadas con un rango numérico y, en algunos casos, se puede especificar una tolerancia para aceptar respuestas cercanas al valor exacto. La evaluación de la respuesta del estudiante se basa en la comparación con la(s) respuesta(s) esperada(s) y las reglas de tolerancia configuradas. [Figura 7.8]. A continuación, veremos las etiquetas que regulan lo anteriormente dicho.

- `<tolerance></tolerance>`: ¿qué tan exacto debe ser el número?. Ésta eti-

queta irá dentro de `<answer>`, ya que dependerá de cada respuesta la tolerancia que tendrá.

- `<unitgradingtype></unitgradingtype>`: Cómo se introducen las unidades (0 input, 1 radio, 2 select).
- `<unitpenalty></unitpenalty>`: Penalización por unidad incorrecta
- `<showunits></showunits>`: Calificación de unidades (3 none, 1 graded, 0 optional)
- `<unitsleft></unitsleft>`: En qué posición se ponen las unidades

```
▼<answer fraction="100" format="moodle_auto_format">
  <text>15</text>
  ▼<feedback format="html">
    <text/>
  </feedback>
  <tolerance>0.5</tolerance>
</answer>
▼<answer fraction="100" format="moodle_auto_format">
  <text>20</text>
  ▼<feedback format="html">
    <text/>
  </feedback>
  <tolerance>0.1</tolerance>
</answer>
▼<answer fraction="100" format="moodle_auto_format">
  <text>5</text>
  ▼<feedback format="html">
    <text/>
  </feedback>
  <tolerance>0</tolerance>
</answer>
<unitgradingtype>0</unitgradingtype>
<unitpenalty>0.1000000</unitpenalty>
<showunits>3</showunits>
<unitsleft>0</unitsleft>
```

Figura 7.8: Marcas numerical

5. *Emparejamiento*: En Moodle XML, se utilizan etiquetas específicas para definir las preguntas de emparejamiento. Cada par está contenido adentro de una marca `<subquestion></subquestion>`. El primer ítem de cada par está contenido con una marca `<text></text>`, mientras que el segundo tiene una marca `<answer></answer>` [Figura 7.9]. Se podrá añadir la siguiente marcas.

- `<shuffleanswers></shuffleanswers>` (values true/false): Valor true para determinar el orden de los ítems debe de alatorizarse (barajearse).

- Se podrá añadir también etiquetas de feedback como:

`<correctfeedback></correctfeedback>`. Esta retroalimentación saldrá cuando se emparejen bien todas las parejas.

`<partiallycorrectfeedback></partiallycorrectfeedback>`. Esta retroalimentación saldrá cuando se emparejen bien algunas de las parejas.

`<incorrectfeedback></incorrectfeedback>`. Esta retroalimentación saldrá cuando se emparejen mal todas las parejas.

```

    <shuffleanswers>true</shuffleanswers>
  ▶ <correctfeedback format="html">
    ...
  </correctfeedback>
  ▶ <partiallycorrectfeedback format="html">
    ...
  </partiallycorrectfeedback>
  ▶ <incorrectfeedback format="html">
    ...
  </incorrectfeedback>
  <shownumcorrect/>
  ▼ <subquestion format="html">
    ▼ <text>
      <![CDATA[ <p>perro</p> ]]>
    </text>
    ▼ <answer>
      <text>mascota</text>
    </answer>
  </subquestion>
  ▶ <subquestion format="html">
    ...
  </subquestion>
  ▶ <subquestion format="html">
    ...
  </subquestion>
  ▶ <subquestion format="html">
    ...
  </subquestion>

```

Figura 7.9: Marcas matching

7.2.2. Estructura de los datos de IMS/QTI

La estructura de un cuestionario en formato QTI/IMS (IMS QTI) sigue una jerarquía específica y utiliza etiquetas XML para representar los diferentes elementos del cuestionario. La terminología adoptada para el QTI es que un Ítem se define como el bloque fundamental que contiene una o más preguntas y respuestas [24]. Podremos observar el modelo de datos QTI. [Figura 7.10]

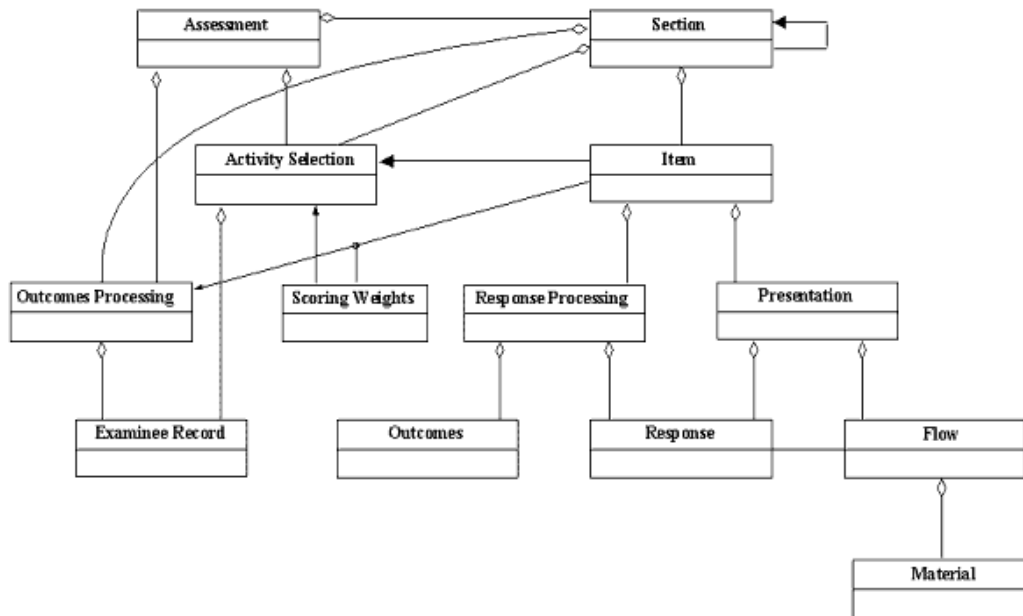


Figura 7.10: Modelo de objetos de datos de QTI

A continuación, se describe la estructura básica de un cuestionario en formato QTI/IMS. [Figura 7.11]

```

<?xml version="1.0" encoding="UTF-8"?>
<questestinterop xmlns="http://www.imsglobal.org/xsd/imsqti_v2p1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.imsglobal.org/xsd/imsqti_v2p1">

</questestinterop>

```

Figura 7.11: Marcas QTI/IMS - raíz

- **<questestinterop></questestinterop>**: Es el elemento raíz, engloba toda la información relacionada con las preguntas y pruebas, como los metadatos, las preguntas individuales, los ítems de respuesta, las opciones de respuesta, los tipos de

preguntas , la retroalimentación, entre otros elementos necesarios para describir y definir las preguntas y pruebas.

El cuestionario cuenta con un identificador y un nombre que podemos encontrar en la etiqueta:

- **<assessment></assessment>**: El término *assessment* se refiere a la evaluación o prueba de conocimientos y habilidades que se realiza utilizando el estándar QTI. Esta se divide en dos etiquetas hijo. **<qtimetadadata>** y **<section>**. [Figura 7.12]

```
▼<assessment ident="gd47885be55a08fbe26265b9d0b83ebbb" title="prueba">
  ►<qtimetadadata>
    ...
    </qtimetadadata>
  ►<section ident="root_section">
    ...
    </section>
  </assessment>
```

Figura 7.12: Marca assessment

1. **<qtimetadadata></qtimetadadata>**: Se utiliza para proporcionar metadatos relacionados con un cuestionario o una pregunta en el QTI estándar, es decir, esta etiqueta no solo nos la encontraremos como hija de **<assessment>** sino que también estará dentro de una pregunta, las cuales se identifican con la etiqueta **<item>**. Los metadatos son información adicional que describe y proporciona contexto sobre el cuestionario o la pregunta. Estos metadatos pueden incluir detalles como el título del cuestionario, el autor, la descripción, las palabras clave, la duración estimada, el nivel de dificultad, la competencia evaluada, entre otros.[Figura 7.13]

```
▼<qtimetadadata>
  ▼<qtimetadadatafield>
    <fieldlabel>qmd_timelimit</fieldlabel>
    <fieldentry>90</fieldentry>
  </qtimetadadatafield>
  ▼<qtimetadadatafield>
    <fieldlabel>cc_maxattempts</fieldlabel>
    <fieldentry>1</fieldentry>
  </qtimetadadatafield>
</qtimetadadata>
```

Figura 7.13: Marca qtimetadadata

2. **<section></section>**: Se refiere a una sección o grupo de preguntas dentro de un cuestionario o evaluación. Una sección en IMS/QTI se utiliza para organizar y estructurar las preguntas (**<item>**) relacionadas en un cuestionario. Puede contener una o varias preguntas, y se utiliza para agrupar preguntas que comparten

una temática común o se enfocan en una habilidad o competencia específica.[Figura 7.14]

```
<section id="root_section">
  <item id="gbe34ac6e19206b83e71f663d1feff9a4" title="Pregunta">
    ...
  </item>
  <item id="g522a62aaeeb93857bc6aa07f6d7cad96" title="Pregunta">
    ...
  </item>
</section>
```

Figura 7.14: Marca section

- **<item></item>**: Contiene toda la información relevante relacionada con la pregunta. La siguiente figura muestra el árbol de esquema XML para las estructuras de datos ítem: [Figura 7.15]

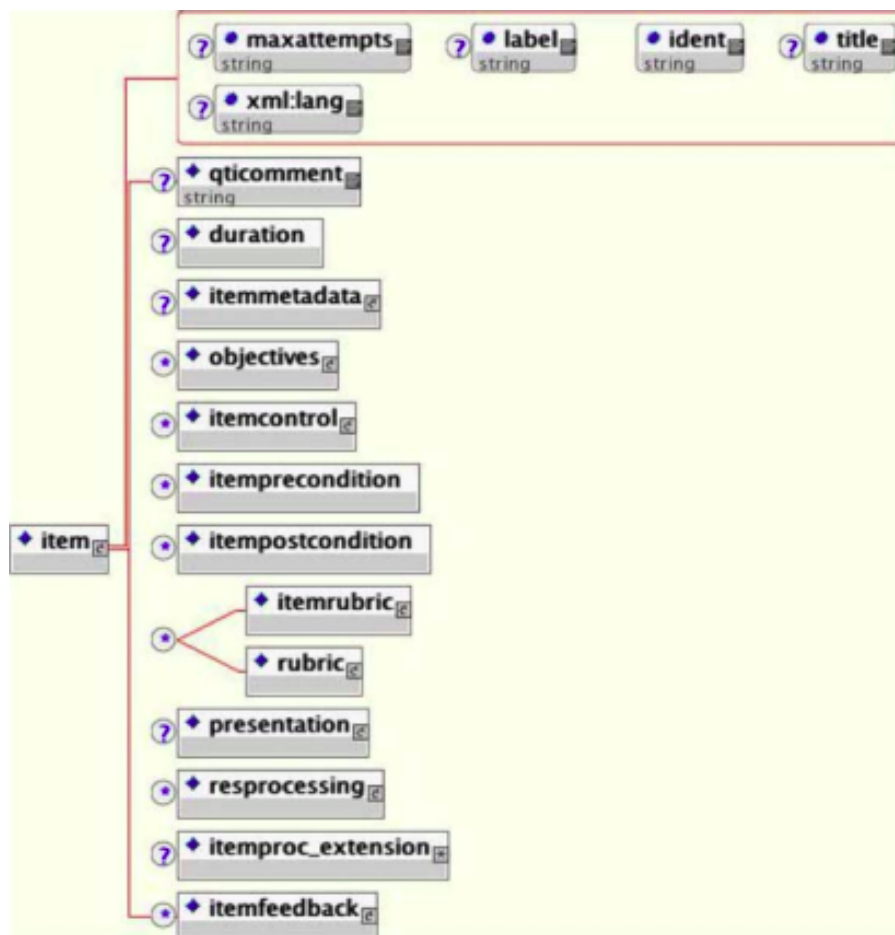


Figura 7.15: Árbol de esquema XML de ítem

Se utilizan diferentes subelementos y atributos para definir y especificar las características de la pregunta, como el tipo de pregunta (opción múltiple, verdadero/falso, respuesta corta, ensayo, etc.), el texto del enunciado, las opciones de respuesta, la

retroalimentación para cada opción, entre otros detalles específicos de la pregunta. Estos detalles mencionados se encuentran repartidos en tres etiquetas: [Figura 7.16]

```
▼<item ident="gbe34ac6e19206b83e71f663d1feff9a4" title="Pregunta">
  ▶<itemmetadata>
    ...
  </itemmetadata>
  ▶<presentation>
    ...
  </presentation>
  ▶<resprocessing>
    ...
  </resprocessing>
</item>
```

Figura 7.16: Marcas item

<itemmetadata> mantendrá la misma estructura sin importar el tipo de pregunta, mientras que **<presentation>** y **<resprocessing>** serán diferentes, y varían.

1. **<itemmetadata></itemmetadata>**: Dentro de esta etiqueta se almacena los metadatos de la pregunta, usando para ello la etiqueta anteriormente mencionada **<qtimetadata>**. Estos metadatos se almacenan por parejas de etiquetas **<field-label>** y **fieldentry** de la siguiente forma: [Figura 7.17]

- **<fieldlabel>** Se indica el nombre del atributo predefinido en QTI/IMS, que se va almacenar. Algunos valores:
 - **question_type**: Tipo de pregunta.
 - multiple_choice_question = opción múltiple con respuesta única.
 - multiple_answers_question = opción múltiple con múltiples respuestas.
 - true_false_question = verdadero/falso
 - short_answer_question = respuesta corta
 - matching_question = emparejamiento
 - numerical_question = numérica
 - Entra otros.
 - **points_possible**: Puntos posibles para la pregunta.
 - **original_answer_ids**: Son los identificadores de las respuestas.
 - **assessment_question_identifierref**: Identificador de la pregunta.
- **<fieldentry>**: Almacena los valores del nombre anteriormente definido.

```

▼<itemmetadata>
  ▼<qtimetadata>
    ▼<qtimetadadatafield>
      <fieldlabel>question_type</fieldlabel>
      <fieldentry>short_answer_question</fieldentry>
    </qtimetadadatafield>
    ▼<qtimetadadatafield>
      <fieldlabel>points_possible</fieldlabel>
      <fieldentry>1.0</fieldentry>
    </qtimetadadatafield>
    ▼<qtimetadadatafield>
      <fieldlabel>original_answer_ids</fieldlabel>
      <fieldentry>9718,8661,4255</fieldentry>
    </qtimetadadatafield>
    ▼<qtimetadadatafield>
      <fieldlabel>assessment_question_identifierrref</fieldlabel>
      <fieldentry>ge4e4a55d35858a55da6469949902c9b6</fieldentry>
    </qtimetadadatafield>
  </qtimetadata>
</itemmetadata>

```

Figura 7.17: Marcas itemmetadata

Además de **<qtimetadata>**, puede contener otras etiquetas como se verá en el árbol XML. [Figura 7.18]

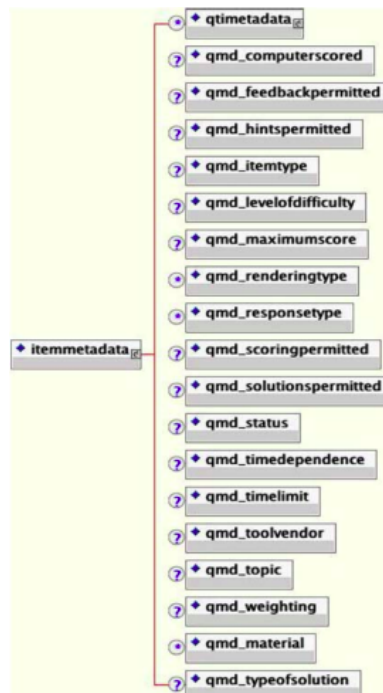


Figura 7.18: Árbol de esquema XML del elemento Itemmetadata

2. **<presentation></presentation>**: Se utiliza para encapsular y estructurar el contenido de presentación de una pregunta. La sección de presentación proporciona el enunciado y cualquier otro material de apoyo necesario para que el estudiante comprenda la pregunta y pueda responder correctamente. [Figura 7.19]

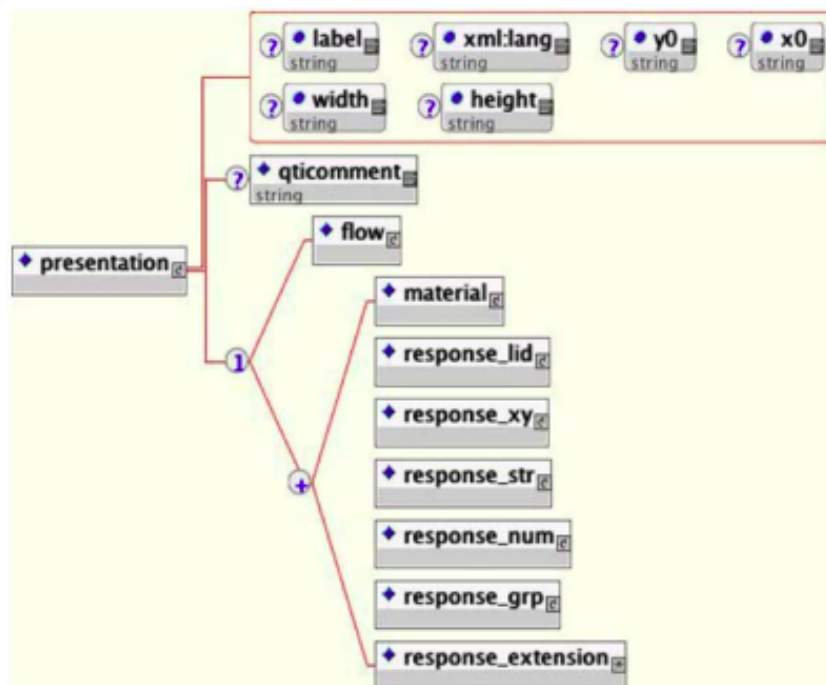


Figura 7.19: Árbol de esquema XML del elemento Presentation

La estructura almacenada dependerá del tipo de pregunta que se representa, ya que la etiqueta **<response_lid>**, cambiará.[Figura 7.20]

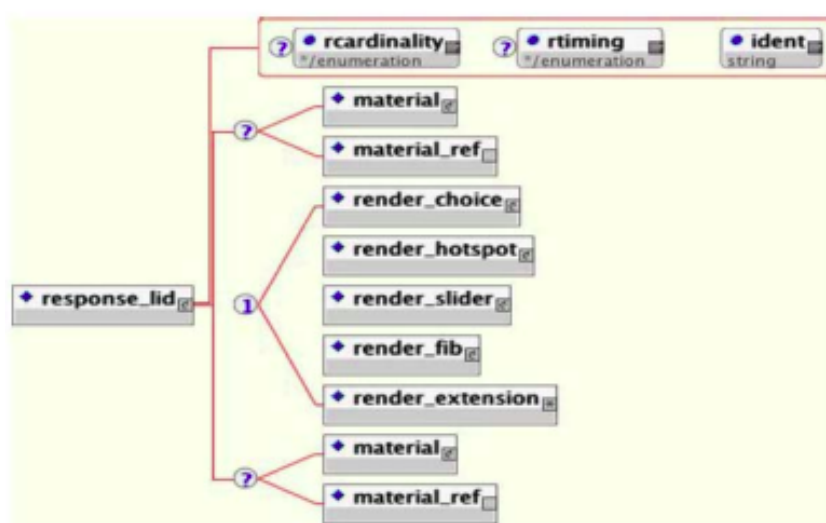


Figura 7.20: Árbol de esquema XML del elemento Response_lid

a) *Opción múltiple y Respuesta múltiple*: Hemos decidido agrupar la explicación

de estos dos tipos de preguntas ya que la representación de la estructura es igual, solo varía el valor en la etiqueta `<response_lid>`, más concretamente en el atributo **rcardinality**, ya que este restringe el número de opciones que el estudiante debe seleccionar o el número de elementos que se deben incluir en su respuesta, el cual será:

- Single: Indica que solo se permite seleccionar o ingresar una única respuesta correcta.
- Múltiple: Indica que se puede seleccionar más de una respuesta correcta.

```

▼<presentation>
  ▼<material>
    <mattext texttype="text/html"><div><p>respuestas multiples</p></div></mattext>
  </material>
  ▼<response_lid ident="response1" rcardinality="Multiple">
    ▼<render_choice>
      ▼<response_label ident="7835">
        ▼<material>
          <mattext texttype="text/html"><p>uno</p></mattext>
        </material>
      </response_label>
      ▼<response_label ident="1865">
        ▼<material>
          <mattext texttype="text/plain">dos</mattext>
        </material>
      </response_label>
      ▼<response_label ident="5081">
        ▼<material>
          <mattext texttype="text/plain">tres</mattext>
        </material>
      </response_label>
    </render_choice>
  </response_lid>
</presentation>

```

Figura 7.21: Marcas presentation - Opción múltiple

Como se puede observar en la [Figura 7.21], para estos tipos de preguntas la presentación es muy clara, ya que se divide en dos partes, por un lado el enunciado de la pregunta almacenado en la etiqueta `<material>`, además también se utiliza para representar y proporcionar contenido adicional relacionado con una pregunta o elemento de evaluación. Y por otro las respuestas están agrupadas dentro de la etiqueta `<response_lid>`, cada pregunta tiene su identificador especificado con la etiqueta `<response_label>`, esta etiqueta es la que permite al estudiante seleccionar una o varias opciones de respuesta. Luego, el contenido de la respuesta, de nuevo con la etiqueta `<material>`.

- b) *Falso/verdadero*: La presentación de este tipo de pregunta es igual a la anterior. [Figura 7.22]

```

▼<presentation>
  ▼<material>
    <mattext texttype="text/html"><div><p>Cual es verdadera</p></div></mattext>
  </material>
  ▼<response_lid ident="response1" rcardinality="Single">
    ▼<render_choice>
      ▼<response_label ident="2099">
        ▼<material>
          <mattext texttype="text/plain">Verdadero</mattext>
        </material>
      </response_label>
      ▼<response_label ident="1341">
        ▼<material>
          <mattext texttype="text/plain">Falso</mattext>
        </material>
      </response_label>
    </render_choice>
  </response_lid>
</presentation>

```

Figura 7.22: Marcas presentation - Falso/verdadero

- c) *Respuesta corta y Respuesta numérica*: Al igual que los anteriores tipos, el enunciado de la pregunta se almacenar en la etiqueta **<material>**, pero ahora el contenido de las respuesta irá en la etiqueta **<response_str>**, ya que ésta se utiliza en preguntas que requieren que el estudiante ingrese una respuesta en forma de texto o número. Dentro de esta etiqueta, se pueden establecer diferentes atributos y configuraciones para controlar el comportamiento de la respuesta y las restricciones que se aplican a la entrada del estudiante. [Figura 7.23]

```

▼<presentation>
  ▼<material>
    <mattext texttype="text/html"><div><p>es una respuesta corta</p></div></mattext>
  </material>
  ▼<response_str ident="response1" rcardinality="Single">
    ▼<render_fib>
      <response_label ident="answer1" rshuffle="No"/>
    </render_fib>
  </response_str>
</presentation>

```

Figura 7.23: Marcas presentation - respuesta corta y numérica

En la figura podemos observar el atributo **rshuffle**, el cual especifica si las opciones de respuesta en una pregunta de opción múltiple deben ser mezcladas (shuffle) o estabilizar en el orden original.

- Si: Indica que las opciones de respuesta deben ser mezcladas en un orden aleatorio cada vez que se presente la pregunta. Esto significa que las opciones se completarán en diferentes órdenes para diferentes estudiantes o en diferentes intentos de evaluación.
- No: Indica que las opciones de respuesta deben mantenerse en el orden original especificado en el archivo de evaluación. Las opciones se completan

en el mismo orden para todos los estudiantes y en todos los intentos de la evaluación.

- d) *Emparejamiento*: La presentación de este tipo de pregunta es muy diferente a la de las anteriores, aunque el título de la pregunta si se sigue representando igual con la etiqueta `<material>`. Luego cuenta con una lista de etiquetas `<response_lid>`, cada una por cada pareja que se relacionará. [Figura 7.24]

```
<presentation>
  <material>
    <mattext texttype="text/html"><div><p>una los valores</p></div></mattext>
  </material>
  <response_lid ident="response_6544">
    ...
  </response_lid>
  <response_lid ident="response_5685">
    ...
  </response_lid>
  <response_lid ident="response_7973">
    ...
  </response_lid>
  <response_lid ident="response_9086">
    ...
  </response_lid>
</presentation>
```

Figura 7.24: Marcas presentation - emparejamiento

Cada `<response_lid>`, esta dividido en dos partes: `<material>` y `<render_choice>`. [Figura 7.25]

- `<material>`: En esta etiqueta irá el valor de la columna izquierda.
- `<render_choice>`: Aquí se establecerá cada uno los valores posibles de la columna derecha con los que podrá emparejar el valor que aparece en `<material>`. Se utiliza la etiqueta `response_label` para definir cada valor.


```
▼<response_lid ident="response_6544">
  ▼<material>
    <mattext texttype="text/plain">perro</mattext>
  </material>
  ▼<render_choice>
    ▼<response_label ident="9615">
      ▼<material>
        <mattext>mascota</mattext>
      </material>
    </response_label>
    ▼<response_label ident="3430">
      ▼<material>
        <mattext>rey de la selva</mattext>
      </material>
    </response_label>
    ▼<response_label ident="2999">
      ▼<material>
        <mattext>mar</mattext>
      </material>
    </response_label>
    ▼<response_label ident="1219">
      ▼<material>
        <mattext>comida</mattext>
      </material>
    </response_label>
  </render_choice>
</response_lid>
```

Figura 7.25: Marcas presentation - emparejamiento - columnas

3. **<resprocessing></resprocessing>**: Se utiliza para definir la lógica de procesamiento de respuestas en una pregunta o actividad. Especifica cómo se deben evaluar y puntuar las respuestas proporcionadas por el estudiante. [Figura 7.26]

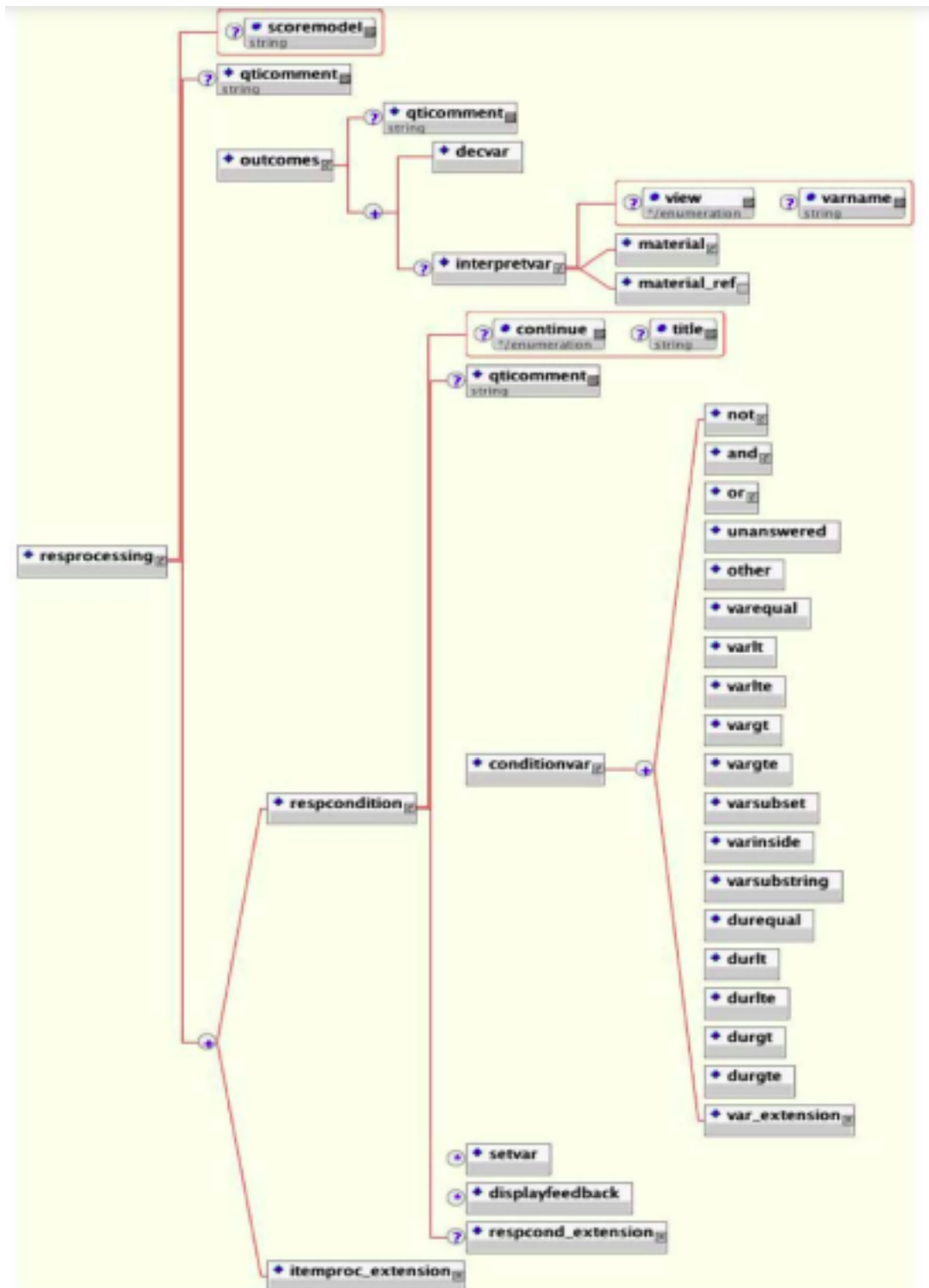


Figura 7.26: Árbol de esquema XML del elemento Resprocessing

El procesamiento de respuestas implica analizar las respuestas de un candidato y

determinar la corrección de las mismas. Los resultados obtenidos se utilizan para brindar retroalimentación al candidato, ya sea de forma inmediata o en un informe posterior. Este proceso permite evaluar las respuestas de manera precisa y proporcionar información valiosa al candidato sobre su desempeño. Además, se pueden utilizar diversas etiquetas y atributos para definir la lógica de procesamiento de respuestas, como `<setvar>` para asignar valores a variables, `<varequal>` para comparar valores, `<answer>` para especificar las respuestas correctas, `<conditionvar>` para definir condiciones, entre otros. Al igual que la etiqueta `<presentation>`, la estructura dependerá del tipo de pregunta.

a) *Opción múltiple*: Si observamos la [Figura 7.27], el procesamiento de este tipo de pregunta se divide en dos etiquetas

- `<outcomes></outcomes>`: Define los posibles resultados o puntajes que se pueden asignar a las respuestas del estudiante.
- `<respcndition></respcndition>`: Especifica condiciones y acciones que se deben tomar según las respuestas del estudiante. Además, se pueden utilizar diversas etiquetas y atributos para definir la lógica de procesamiento de respuestas, como `<conditionvar>` para definir condiciones, `<setvar>` para asignar valores a variables, `<varequal>` para comparar valores de los identificadores de las respuestas, y así especificar las respuestas correctas, entre otros.

```

▼<resprocessing>
  ▼<outcomes>
    <decvar maxvalue="100" minvalue="0" varname="SCORE" vartype="Decimal"/>
  </outcomes>
  ▼<respcndition continue="No">
    ▼<conditionvar>
      <varequal respident="respuesta1">3869</varequal>
    </conditionvar>
    <setvar action="Set" varname="SCORE">100</setvar>
  </respcndition>
</resprocessing>

```

Figura 7.27: Marcas resprocessing - Opción múltiple

b) *Respuesta múltiple*: Sigue una estructura similar a la anterior, pero aquí se usa variables lógicas, con etiquetas como `<and>` o `<not>` y así definir cuales son las respuestas correctas y cuales no lo son. Teniéndose que cumplir dicha sentencia establecida para poder establecer como correcta la respuesta. Se utiliza el elemento `<setvar>` para asignar *SCORE*. El atributo **action** se establece en *Set*, lo que indica que se debe asignar un nuevo valor a la variable. El atributo **varname** se establece que será aplicado a la variable *SCORE*. El valor *100* se establece como nuevo valor para dicha variable. [Figura 7.28]

```
▼<respcondition continue="No">
  ▼<conditionvar>
    ▼<and>
      <varequal respident="response1">7835</varequal>
      <varequal respident="response1">1865</varequal>
    ▼<not>
      <varequal respident="response1">5081</varequal>
    </not>
  </and>
</conditionvar>
<setvar action="Set" varname="SCORE">100</setvar>
</respcondition>
</resprocessing>
```

Figura 7.28: Marcas resprocessing - Opción respuesta múltiple

- c) *Falso/verdadero*: Podemos ver como esta estructura varía un poco. Observamos tres etiquetas `<respcondition>`, las dos primeras se establecen para añadir un comentario de retroalimentación cuando sean respondidas, es decir, la propiedad **feedbacktype** se establece en *Response*, lo que indica que se muestra un comentario asociado a la respuesta del estudiante. El atributo **linkrefid** especifica la referencia a la respuesta correspondiente al ID asociado. Y la última etiqueta `<respcondition>` se usa para establecer cual es la respuesta correcta. Más abajo podremos ver como se define los comentarios asociados a las preguntas a través de la etiqueta `<itemfeedback>` y el identificador correspondiente. [Figura 7.29]

```

▼<resprocessing>
  ▼<outcomes>
    <decvar maxvalue="100" minvalue="0" varname="SCORE" vartype="Decimal"/>
  </outcomes>
  ▼<respcondition continue="Yes">
    ▼<conditionvar>
      <varequal respident="response1">2099</varequal>
    </conditionvar>
    <displayfeedback feedbacktype="Response" linkrefid="2099_fb"/>
  </respcondition>
  ▼<respcondition continue="Yes">
    ▼<conditionvar>
      <varequal respident="response1">1341</varequal>
    </conditionvar>
    <displayfeedback feedbacktype="Response" linkrefid="1341_fb"/>
  </respcondition>
  ▼<respcondition continue="No">
    ▼<conditionvar>
      <varequal respident="response1">2099</varequal>
    </conditionvar>
    <setvar action="Set" varname="SCORE">100</setvar>
  </respcondition>
</resprocessing>
▼<itemfeedback ident="2099_fb">
  ▼<flow_mat>
    ▼<material>
      <mattext texttype="text/html"><p>Verdad</p></mattext>
    </material>
  </flow_mat>
</itemfeedback>
▼<itemfeedback ident="1341_fb">
  ▼<flow_mat>
    ▼<material>
      <mattext texttype="text/html"><p>Falso</p></mattext>
    </material>
  </flow_mat>
</itemfeedback>

```

Figura 7.29: Marcas resprocessing - Verdadero/Falso

- d) *Respuesta corta* : El procesamiento de este tipo de pregunta es muy sencillo, pues que solo basta con definir las posibles respuestas. Si la respuesta del estudiante coincide con alguna de las tres opciones mencionadas, se ejecuta la acción especificada dentro de `<respcondition>`, es decir, asignar un nuevo valor a la variable **SCORE**. [Figura 7.30]

```

▼<resprocessing>
  ▼<outcomes>
    <decvar maxvalue="100" minvalue="0" varname="SCORE" vartype="Decimal"/>
  </outcomes>
  ▼<respcondition continue="No">
    ▼<conditionvar>
      <varequal respident="response1">uno</varequal>
      <varequal respident="response1">dos</varequal>
      <varequal respident="response1">tres</varequal>
    </conditionvar>
    <setvar action="Set" varname="SCORE">100</setvar>
  </respcondition>
</resprocessing>

```

Figura 7.30: Marcas resprocessing - Respuesta corta

- e) *Respuesta numérica*: Como se puede observar en la [Figura 7.31], podemos definir varias formas de procesar diferentes respuestas, todo dependerá de las condiciones que establezcamos en la etiqueta **<conditionvar>**.

```

▼<resprocessing>
  ▼<outcomes>
    <decvar maxvalue="100" minvalue="0" varname="SCORE" vartype="Decimal"/>
  </outcomes>
  ▼<respcondition continue="No">
    ▼<conditionvar>
      ▼<or>
        <varequal respident="response1">21.09</varequal>
        ▼<and>
          <vargte respident="response1">21.07</vargte>
          <varlte respident="response1">21.11</varlte>
        </and>
      </or>
    </conditionvar>
    <setvar action="Set" varname="SCORE">100</setvar>
  </respcondition>
  ▼<respcondition continue="No">
    ▼<conditionvar>
      <vargte respident="response1">10.0</vargte>
      <varlte respident="response1">15.0</varlte>
    </conditionvar>
    <setvar action="Set" varname="SCORE">100</setvar>
  </respcondition>
  ▶<respcondition continue="No">
    ...
  </respcondition>
  ▶<respcondition continue="No">
    ...
  </respcondition>
</resprocessing>

```

Figura 7.31: Marcas resprocessing - Respuesta numérica

- f) *Emparejamiento*: Se verifica si la respuesta identificada por el atributo **respident**, en la etiqueta **<varequal>** es igual a un valor determinado, este valor corresponde con el identificador de la respuesta emparejada. Si se cumple esta condición, quiere decir que ha emparejado correctamente, por lo que se ejecuta la acción especificada dentro de **<respcondition>**. [Figura 7.32]

```

▼<resprocessing>
  ▼<outcomes>
    <decvar maxvalue="100" minvalue="0" varname="SCORE" vartype="Decimal"/>
  </outcomes>
  ▼<respcondition>
    ▼<conditionvar>
      <varequal respident="response_6544">9615</varequal>
    </conditionvar>
    <setvar varname="SCORE" action="Add">25.00</setvar>
  </respcondition>
  ▼<respcondition>
    ▼<conditionvar>
      <varequal respident="response_5685">3430</varequal>
    </conditionvar>
    <setvar varname="SCORE" action="Add">25.00</setvar>
  </respcondition>
  ▼<respcondition>
    ▼<conditionvar>
      <varequal respident="response_7973">2999</varequal>
    </conditionvar>
    <setvar varname="SCORE" action="Add">25.00</setvar>
  </respcondition>
  ▼<respcondition>
    ▼<conditionvar>
      <varequal respident="response_9086">1219</varequal>
    </conditionvar>
    <setvar varname="SCORE" action="Add">25.00</setvar>
  </respcondition>
</resprocessing>

```

Figura 7.32: Marcas resprocessing - Empajeramiento

En resumen, podemos ver que QTI/IMS hace uso de estructuras de datos interoperables.

7.3. Descripción de la interfaz (Plataformas)

A continuación, veremos la representación de los cuestionarios convertidos, es decir, como se deberían ver en las plataformas Moodle y Canvas, una vez se hayan convertido.

7.3.1. Plataforma Moodle

Una vez seleccionado el fichero convertido, se podrá se importar en la plataforma [Figura 7.33].

Importar preguntas de un archivo ?

► Expandir todo

▼ Formato de archivo

- ☐ Blackboard ?
- ☐ Examview ?
- ☐ Formato Aiken ?
- ☐ Formato de palabra ausente ?
- ☐ Formato GIFT ?
- ☒ Formato Moodle XML ?
- ☐ Formato WebCT ?
- ☐ Respuestas incrustadas (Cloze) ?

► General

▼ Importar preguntas de un archivo

Importar ! Seleccione un archivo... Tamaño máximo para archivos nuevos: 1000MB

MULTI__tema5_i.xml

Importar

En este formulario hay campos obligatorios !.

Figura 7.33: Importación en Moodle

En la siguiente pantalla, ésta misma nos notificará que la importación ha sido correcta en el caso de que el fichero cumpla con el formato establecido y tenga una estructura correcta. En la caso de que esto no ocurra, la importación se detendrá y notificará el posible error del fichero. [Figuras 7.34 y 7.35] correspondientemente.

Procesando las preguntas del archivo importado. x

Importando 7 preguntas desde archivo x

Figura 7.34: Importación en Moodle - Correcta

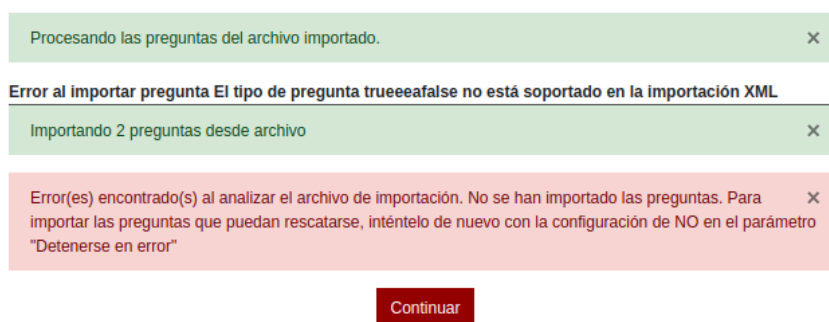


Figura 7.35: Importación en Moodle - Error

En el caso de que la importación haya sido correcta, podremos visualizar el listado de las preguntas. [Figura 7.36].

Pregunta	Acciones	Creado por	Última modificación por
Nombre de la pregunta / ID number		Nombre / Apellido(s) / Fecha	Nombre / Apellido(s) / Fecha
<input type="checkbox"/> 05-09-COMBINACION 01	Editar	Diego Alejandro Higueta Grisales 21 de May de 2023, 10:10	Diego Alejandro Higueta Grisales 21 de May de 2023, 10:10
<input type="checkbox"/> 05-09-COMBINACION 02	Editar	Diego Alejandro Higueta Grisales 21 de May de 2023, 10:10	Diego Alejandro Higueta Grisales 21 de May de 2023, 10:10
<input type="checkbox"/> 05-09-COMBINACION 03	Editar	Diego Alejandro Higueta Grisales 21 de May de 2023, 10:10	Diego Alejandro Higueta Grisales 21 de May de 2023, 10:10
<input type="checkbox"/> 05-09-COMBINACION 04	Editar	Diego Alejandro Higueta Grisales 21 de May de 2023, 10:10	Diego Alejandro Higueta Grisales 21 de May de 2023, 10:10
<input type="checkbox"/> 05-09-COMBINACION 05	Editar	Diego Alejandro Higueta Grisales 21 de May de 2023, 10:10	Diego Alejandro Higueta Grisales 21 de May de 2023, 10:10
<input type="checkbox"/> 05-09-COMBINACION 06	Editar	Diego Alejandro Higueta Grisales 21 de May de 2023, 10:10	Diego Alejandro Higueta Grisales 21 de May de 2023, 10:10
<input type="checkbox"/> 05-09-COMBINACION 07	Editar	Diego Alejandro Higueta Grisales 21 de May de 2023, 10:10	Diego Alejandro Higueta Grisales 21 de May de 2023, 10:10

Figura 7.36: Listado preguntas importadas - Moodle

7.3.2. Plataforma Canvas

Para importar un fichero en formato QTI/IMS, además de cumplir la estructura, este debe estar comprimido en formato zip. [Figura 7.37].

Importar contenido

Tipo de contenido:

Original: tema5_i.zip

Banco de preguntas predeterminado:

Opciones: ☐ Sobrescribir el contenido de la actividad con identificaciones parecidas

Al importar el mismo contenido de una asignatura más de una vez, reemplazará cualquier contenido existente en la asignatura.

Figura 7.37: Importación en Canvas

Al igual que la plataforma anterior, recibiremos una notificación por parte de Canvas, tanto si la importación ha sido correcta, como si no lo ha sido. [Figuras 7.38 y 7.39] correspondientemente.



Figura 7.38: Importación en Canvas - Correcta



Figura 7.39: Importación en Canvas - Error

Si la importación ha sido correcta, podremos encontrar el cuestionario en el menú **pruebas** ubicado en la columna izquierda. En este menú encontraremos todos los cuestionarios subidos. Al frente de cada cuestionario aparece un check verde si el cuestionario ha sido publicado para los estudiantes. [Figura 7.40]



Figura 7.40: Listado cuestionarios - Canvas

7.3.3. Representación preguntas en ambas plataformas

La representación de cada tipo de pregunta serán similares en ambas plataformas, aunque el procesamiento puede variar un poco, como lo vimos en el apartado 7.2 *Descripción de la información*. A continuación enseñaremos la visualización de los tipos de preguntas (los que soporta nuestro sistema conversor) en las plataformas *Moodle* y *Canvas*.

- **Opción Múltiple:** Éste es el tipo de pregunta más utilizado en los cuestionarios, donde hay un enunciado y deben aparecer múltiples opciones para elegir pero solo habrá una única opción correcta. [Figuras 7.41a y 7.41b]

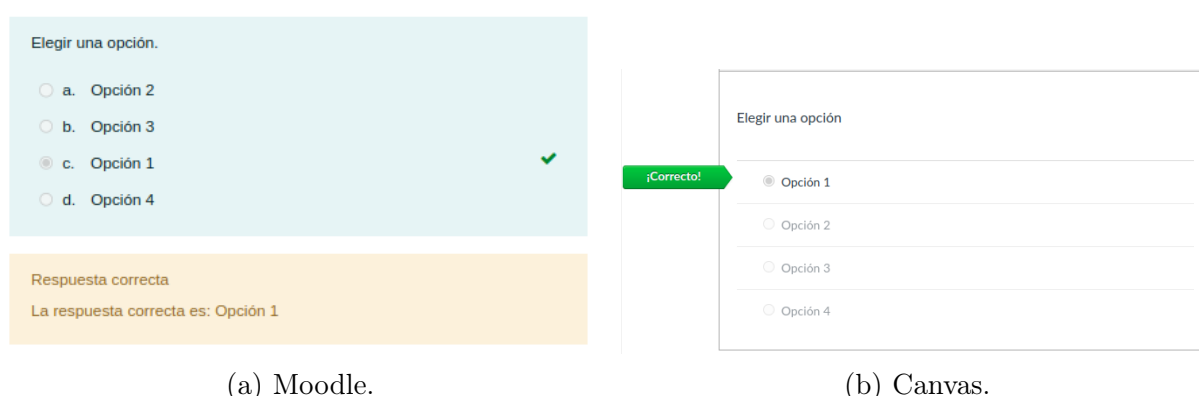


Figura 7.41: Representación - Opción Múltiple

- **Respuesta Múltiple:** Al igual que el tipo de pregunta anterior, debe aparecer varias opciones pero con la diferencia de que hay más de una respuesta correcta. En *Canvas* debe seleccionar todas las respuestas correctas, pues de lo contrario, si selecciona una respuesta correcta y otra incorrecta, el sistema dará puntaje 0. Mientras que en *Moodle* si dará puntajes parciales por cada respuesta correcta. [Figuras 7.42a y 7.42b]

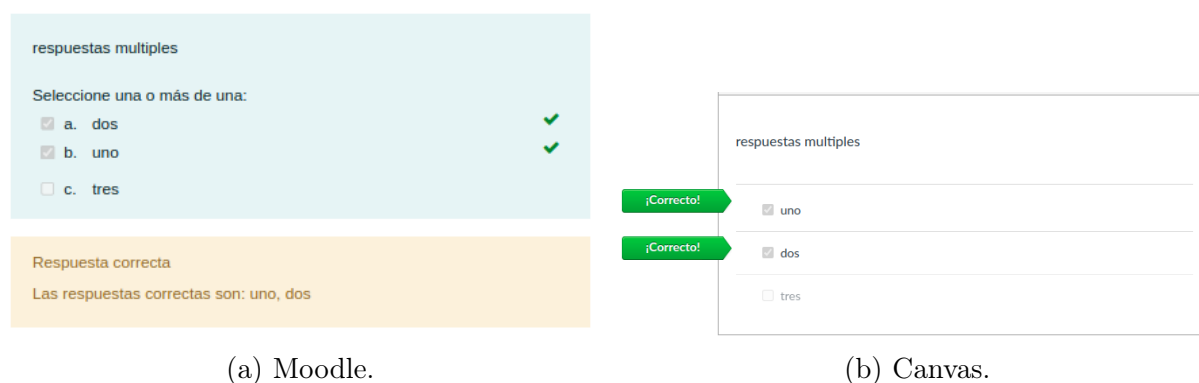


Figura 7.42: Representación - Respuesta Múltiple

- **Verdadero/Falso:** Estas preguntas presentan una declaración y se dan dos opciones para respuesta, una es verdadera y la otra es falsa. Debe seleccionar la opción correspondiente. [Figuras 7.43a y 7.43b]

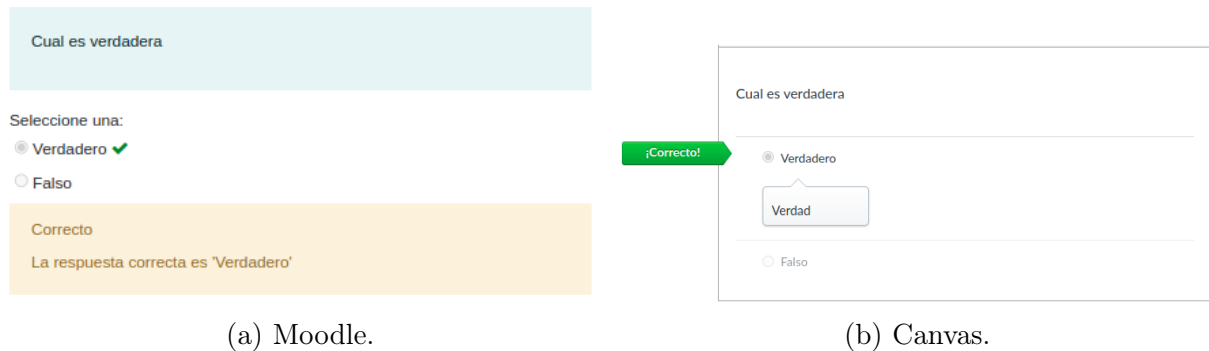


Figura 7.43: Representación - Verdadero/Falso

- **Respuesta Corta:** Aparecerá un espacio en blanco para escribir la respuesta, esta suele componerse de una sola palabra, es decir, respuestas breves o completar oraciones utilizando un texto corto [Figuras 7.44a y 7.44b]

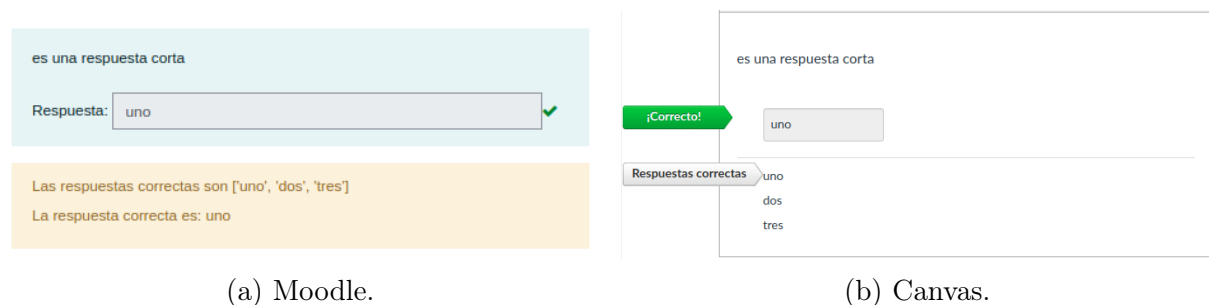


Figura 7.44: Representación - Respuesta Corta

- **Respuesta Numérica:** Al igual que el tipo anterior, aparecerá un espacio en blanco pero en este caso es para introducir un valor numérico. [Figuras 7.45a y 7.45b]

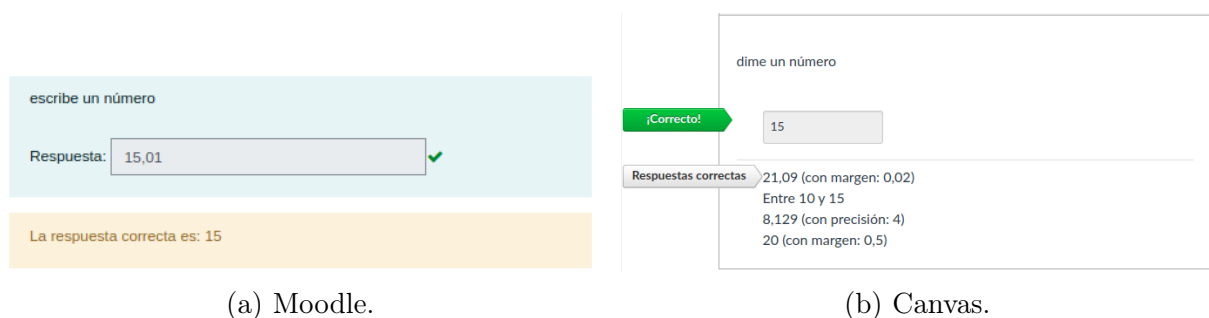


Figura 7.45: Representación - Respuesta Numérica

- **Emparejamiento:** La representación de una pregunta de emparejamiento incluye dos columnas, una para el primer conjunto de elementos y otra para el segundo conjunto de elementos. Cada elemento en la columna de la izquierda se debe emparejar con un elemento correspondiente en la columna de la derecha, los cuales aparecen en forma de desplegables con todas las opciones posibles de emparejar. [Figuras 7.46a y 7.46b]

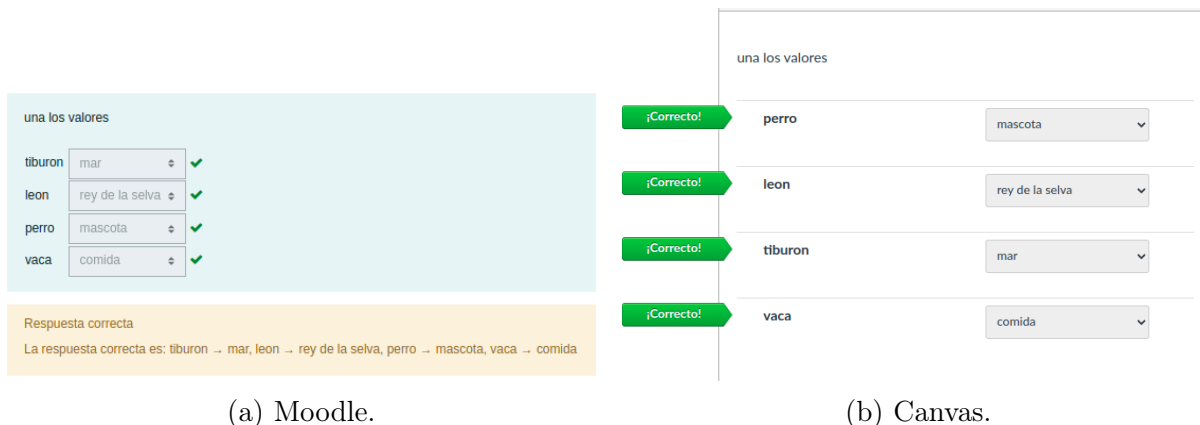


Figura 7.46: Representación - Emparejamiento

7.4. Especificación de Requisitos

En esta sección, se detallan los requisitos técnicos necesarios para proporcionar una descripción completa de los servicios que ofrece el sistema. Para facilitar su trazabilidad, estos requisitos se cumplen de acuerdo con los casos de uso y las descripciones de funcionalidad, información e interfaz.

7.4.1. Requisitos Funcionales

- RF-1. El sistema debe permitir cargar archivos XML.
- RF-2. El sistema debe realizar la conversión del formato QTI/IMS al formato MoodleXML.
- RF-3. El sistema debe realizar la conversión del formato MoodleXML al formato QTI/IMS
- RF-4. El sistema debe permitir escoger cual será el formato origen y cual será el formato final.
- RF-5. El sistema debe guardar el fichero convertido al formato final.
- RF-6. El sistema debe mostrar donde ha guardado el fichero convertido.

- RF-7. El sistema debe validar la estructura del archivo XML en su formato origen.
- RF-8. El sistema debe informar de los tipos de preguntas que puede convertir.

7.4.2. Requisitos de Información

- RINF-1. El fichero a convertir solo puede tener preguntas que soporte el sistema.
- RINF-2. El fichero a convertir debe ser de extensión .XML.
- RINF-3. Los formatos XML de los ficheros deben ser:
 - MoodleXML.
 - QTI/IMS 1.2, pero comprimido en un archivo zip.
- RINF-4. El sistema notificará al usuario en caso de que no pueda convertir el fichero y/o ocurra un error en la conversión.

7.4.3. Requisitos de Interfaz

- RI-1. El sistema deberá ofrecer una interfaz usable e intuitiva, la cual no requiera una gran curva de aprendizaje ni grandes conocimientos.

7.4.4. Requisitos No Funcionales

- RNF-1. El sistema deberá realizar de manera instantánea la conversión entre los formatos.
- RNF-2. Las conversiones deben ser compatibles en las plataformas moodle y Canvas.

Capítulo 8

Diseño del sistema

En los capítulos anteriores, se han establecido las características, restricciones y requisitos que deben cumplirse en el desarrollo de la aplicación. Ahora, en los próximos capítulos, se explicará detalladamente cómo se diseñará la aplicación para satisfacer esos requisitos.

El diseño de la aplicación se basa en la información recopilada durante la fase de especificación de requisitos. En esta etapa, se analizará y planificará la implementación del modelo de datos de la aplicación, se determinará su arquitectura, se diseñará la interfaz de usuario y se describirán los procedimientos para llevar a cabo las operaciones más complejas que la aplicación pueda realizar.

8.1. Modelo de datos

Los datos serán extraídos del fichero origen, es decir, el que fichero que se va a convertir. Se deben tener en cuenta las estructuras de los formatos, ambas analizados en el punto 7.2 de este documento. No obstante, no toda la información contenida en ellos nos resulta relevante; pues es posible que se pierda información al pasar de un formato XML a otro, especialmente cuando se trata de formatos con estructuras y atributos diferentes, como en el caso de MoodleXML y QTI/IMS 1.2. Esto se debe a que cada formato puede tener sus propias convenciones, elementos y atributos específicos para representar los datos, es decir, pueden no tener una representación directa en el otro formato, lo que puede conducir a una pérdida de información o a una representación menos precisa de los datos. En la tabla 8.1, se mostrarán los elementos y/o datos principales de los cuales se harán uso, junto con su equivalencia en ambos formatos. En el caso que la equivalencia corresponda a más de una etiqueta, se intentará poner la jerarquía entera, ó al menos la etiqueta padre.

Por otra parte, con el objetivo de dotar al sistema final con la mejor calidad, se van a aplicar algunas buenas praxis de la programación orientada a objetos. [25]

Concretamente, los datos que maneja cada estructura se gestionarán a través de un módulo de encapsulamiento para cada formato, y dependiendo cual será el formato origen, se usará un módulo u otro, que hará las veces de intermediario entre los datos y la interfaz. Esto facilitará su mantenimiento a lo largo del tiempo, ya que en caso de que se efectúe cualquier modificación en el formato de los datos, solo será necesario realizar una adaptación en el módulo correspondiente para que el sistema siga funcionando correctamente.

Elemento	MoodleXML	QTI/IMS
Elemento raíz	<quiz>	<questestinterop>
Representar una pregunta	<question>	<item>
Tipo de pregunta	atributo <i>type</i>	<itemmetadata>→< <i>fieldlabel</i> > <i>question_type</i>
Valor de la pregunta	<defaultgrade>	<itemmetadata>→< <i>fieldlabel</i> > <i>points_possible</i>
Proporcionar el título o nombre de la pregunta	<name>	<title>
Definir el texto o contenido de la pregunta	<questiontext>	<presentation>→< <i>material</i> >
Representar una respuesta posible de opción múltiple	<answer>	<response_label>
Representar una respuesta posible de rellenar espacios	<answer>	<response_str>
Representar columna izquierda en emparejar	<subquestion>	<response_lid>
Representar columna derecha en emparejar	<answer>	<response_label>
Porcentaje aportado de la respuesta	atributo <i>fraction</i>	<setvar>
Saber la respuesta correcta	valor de <i>fraction</i>	Identificador de la respuesta
Indicar si las respuestas deben ser mezcladas	<shuffleanswers>	atributo <i>rshuffle</i>
Indicar si hay más de una respuesta correcta	<single>	atributo <i>rcardinality</i>
Retroalimentación asociados a una opción	<feedback>	<itemfeedback>

Tabla 8.1: Datos - Equivalencias

Los elementos y/o atributos que no tienen equivalencia, estableceremos valores por defecto, que luego se podrán modificar en las plataformas correspondientes.

8.2. Modelo arquitectónico

El principal objetivo de este apartado es desarrollar la estructura del programa en módulos y representar las relaciones de control entre estos, permitiendo así un desarrollo más ordenado.

8.2.1. Clases

1. **QTI2moodle**: La clase *QTI2moodle* se utiliza para leer/convertir aquellos ficheros que tienen una estructura en formato *QTI*, y través de las equivalencias expuestas en el capítulo anterior crea el fichero en formato *moodleXML*. [Figura 8.1]

QTI2moodle
+ file_input + out + path_out
- readQTI() - makeCategoria() - getPrefix() - addText() - defaultMarks() - feedbackMarks() - produceTFQuestion() - produceMCQuestion() - produceSAQuestion() - produceNUMQuestion() - produceMATCHQuestion() - writequestionfile() - fixHtmlText() - convertQTI() - m_conv()

Figura 8.1: Clase QTI2moodle

Dicha clase consta de los siguientes atributos:

- **file_Input**. Fichero de entrada.

- **out**. Nombre del fichero de salida.
- **path_out**. Ruta que almacenará el fichero de salida.

Dicha clase consta de los siguientes métodos:

- **readQTI()**. Función que lee el fichero de entrada en formato QTI, y a través de las equivalencias y las siguientes funciones que vamos a describir, crea el fichero de salida en formato moodleXML.
- **makeCategoria()**. Función que crea la etiqueta `<category>` explicada en el capítulo anterior.
- **getPrefix()**. Función que obtiene un prefijo según el tipo de pregunta.
- **addText()**. Función que añade la etiqueta texto a otra etiqueta padre.
- **defaultMarks()**. Función que añade algunas marcas opcionales de moodleXML con valores por defecto.
- **feedbackMarks()**. Función que añade las marcas de retroalimentación.
- **produceTFQuestion()**. Función que crea una pregunta *Verdadero/Falso* en formato MoodleXML.
- **produceMCQuestion()**. Función que crea una pregunta *Múltiple elección o Múltiple respuesta* en formato MoodleXML.
- **produceSAQuestion()**. Función que crea una pregunta *Respuesta corta* en formato MoodleXML.
- **produceNUMQuestion()**. Función que crea una pregunta *Respuesta numérica* en formato MoodleXML.
- **produceMATCHQuestion()**. Función que crea una pregunta *Empajamiento* en formato MoodleXML.
- **writequestionfile()**. Función que añade al fichero de salida las marcas de un fichero XML y la estructura del árbol.
- **fixHtmlText()**. Función que añade algunas marcas HTML para una mejor presentación.
- **convertQTI()**. Función que convierte el fichero QTI.
- **m_conv()**. Función principal, con la que la interfaz se comunicará.

2. **moodle2QTI**: La clase *moodle2QTI* se utiliza para leer/convertir aquellos ficheros que tienen una estructura en formato *moodleXML*, y través de las equivalencias expuestas en el capítulo anterior crea el fichero en formato *QTI/IMS*. [Figura 8.2]

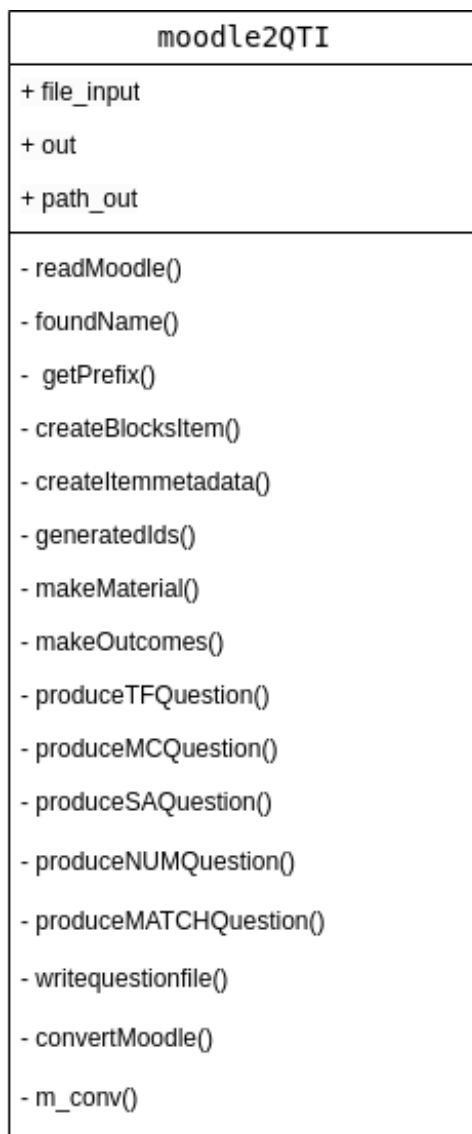


Figura 8.2: Clase moodle2QTI

Dicha clase consta de los siguientes atributos:

- **file_Input**. Fichero de entrada.
- **out**. Nombre del fichero de salida.
- **path_out**. Ruta que almacenará el fichero de salida.

Dicha clase consta de los siguientes métodos:

- **readMoodle()**. Función que lee el fichero de entrada en formato MoodleXML, y a través de las equivalencias y las siguientes funciones que vamos a describir, crea el fichero de salida en formato QTI/IMS.
- **foundName()**. Función que busca el nombre del cuestionario.
- **getPrefix()**. Función que obtiene un prefijo según el tipo de pregunta.
- **createBlocksItem()**. Función que crea los tres bloques que contiene todos los items de QTI.
- **createItemmetadata()**. Función que crea los metadatos correspondientes a un item.
- **generateIds()**. Función que crea identificadores para las respuestas
- **makeMaterial()**. Función que crea la etiqueta **<material>** que almacena las cadenas de texto.
- **makeOutcomes()**. Función que crea la etiqueta **<outcomes>**.
- **produceTFQuestion()**. Función que crea una pregunta *Verdadero/Falso* en formato QTI/IMS.
- **produceMCQuestion()**. Función que crea una pregunta *Múltiple elección o Múltiple respuesta* en formato QTI/IMS.
- **produceSAQuestion()**. Función que crea una pregunta *Respuesta corta* en formato QTI/IMS.
- **produceNUMQuestion()**. Función que crea una pregunta *Respuesta numérica* en formato QTI/IMS.
- **produceMATCHQuestion()**. Función que crea una pregunta *Empajamiento* en formato QTI/IMS.
- **writequestionfile()**. Función que añade al fichero de salida las marcas de un fichero XML y la estructura del árbol.
- **convertMoodle()**. Función que convierte el fichero MoodleXML. .
- **m_conv()**. Función principal, con la que la interfaz se comunicará .

3. **Fichero Principal:** Aunque no corresponda a una de las clases principales del sistema, sino la función principal de entrada., el módulo **gui.py** tendrá el entorno de código de nivel máximo, es decir, el `__main__`, el cual se encarga de la interfaz de usuario, la gestión de eventos y la coordinación de las diferentes funcionalidades de la aplicación.

En la siguiente imagen, se muestra como quedará el diagrama de clases.[Figura [8.3]]

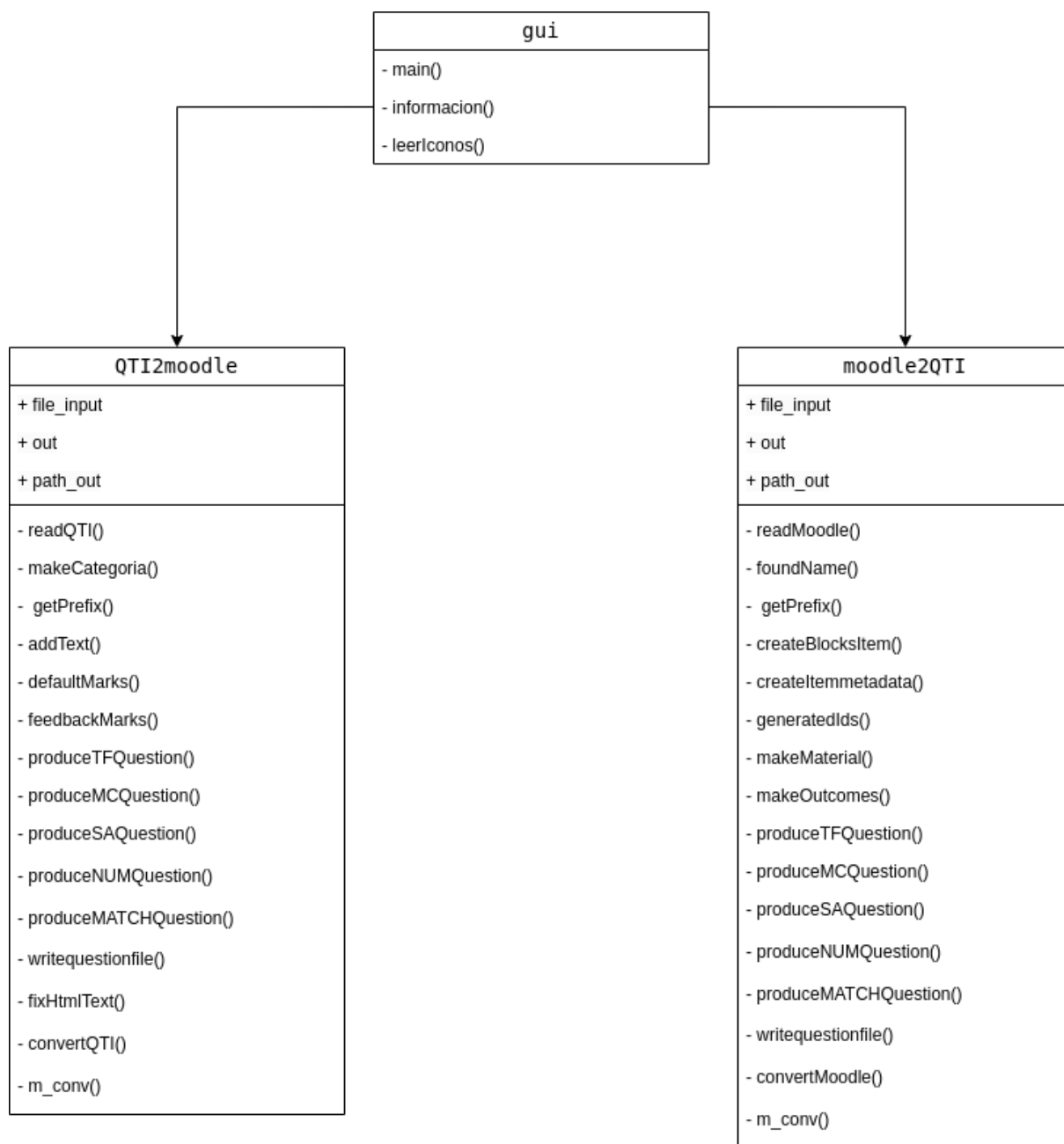


Figura 8.3: Diagrama de clases

8.2.2. Estructuración del código

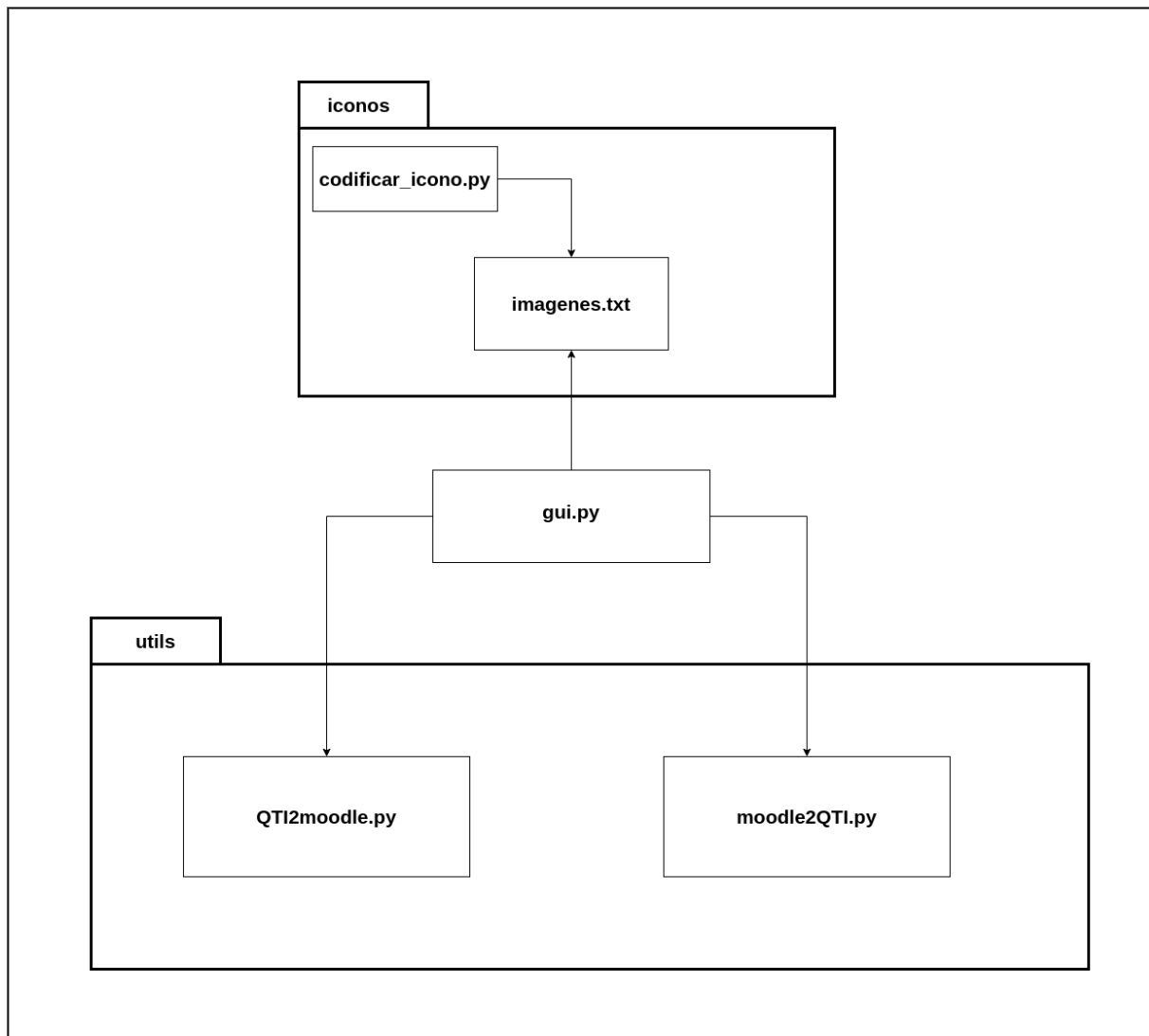


Figura 8.4: Esquema de estructuración del código

Como se puede observar en la [Figura 8.4], los archivos que componen nuestro módulo se encuentran repartidos en dos carpetas distintas, y el archivo **gui.py**.

Carpeta iconos

Contiene todos los iconos de la interfaz de la aplicación.

- **codificar_icono.py**. Es una pequeña función, que codifica los iconos que se muestran en la interfaz [26]. Esta codificación se realiza ya que nuestra aplicación no estará solo enfocada en Windows, el cual si permitiría cargar los iconos desde el ejecutable. Además cuando convirtamos nuestra aplicación de escritorio a un archivo ejecutable, para que no tenga ninguna dependencia de pasar cada icono, uno por

uno, a la carpeta donde se encuentre el ejecutable , tan solo tendremos que pasar el siguiente archivo a describir (**imagenes.txt**).

- **imagenes.txt**. Almacena los iconos que se muestran en la interfaz en Base64, luego el main de gui.py lee el fichero y usa los elementos.

Carpeta utils

Contiene información relevante para el funcionamiento de la aplicación.

- **QTI2moodle.py** Archivo que contiene la clase *QTI2moodle*. Es llamado desde *gui.py*.
- **moodle2QTI.py** Archivo que contiene la clase *moodle2QTI*. Es llamado desde *gui.py*.

Archivo gui.py

Como se explicó anteriormente, es el encargado de enseñar la interfaz, gestionar los eventos de clicks y/o ratón, para así coordinar las funciones de conversión, y contiene funciones como las siguientes:

- **main()**. Es el primer módulo de Python especificado por el usuario que empieza a ejecutarse. Contiene el código que genera la visualización de la interfaz, inicialización de variables y finaliza llamando al archivo correspondiente para la conversión.
- **informacion()**. Es una función que representa un modal, el cual enseña un mensaje de información sobre los tipos de cuestionarios que puede convertir.
- **leerIconos()**. Obtiene los iconos almacenados en el fichero **imagenes.txt**.

8.2.3. Flujos de datos

A partir del análisis de los casos de uso y su diagrama correspondiente, se definirá el flujo de datos que deberá seguir el sistema, más tarde, durante la etapa de implementación.

Diagrama de secuencia Convertir cuestionario

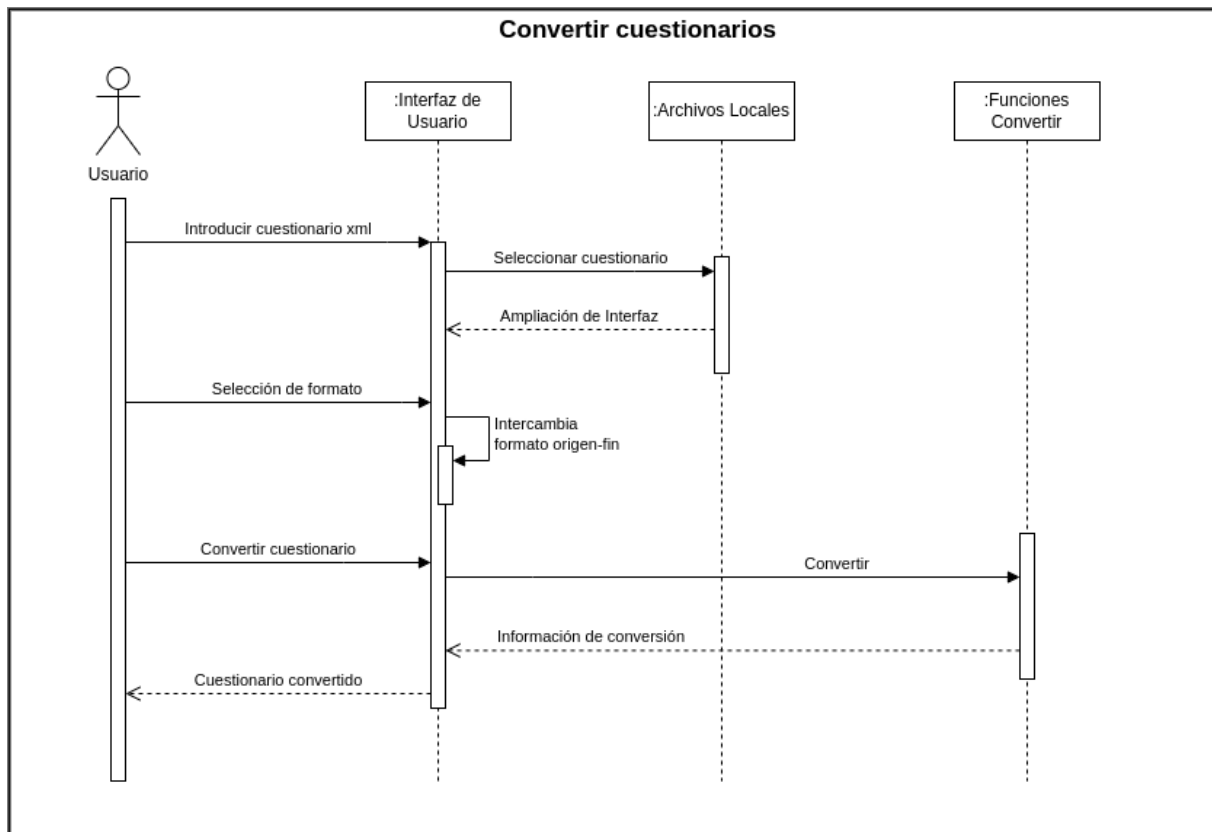


Figura 8.5: Diagrama de secuencia Convertir cuestionario

8.3. Diseño de Interfaz de Usuario

La interfaz gráfica de la aplicación, consta de dos partes bien diferenciadas: una en la que permite elegir el fichero a convertir y cual será el formato origen y final [Figura 8.6]. Y la otra parte, aparece cuando se ha seleccionado el cuestionario, que corresponde a la conversión del fichero [Figura 8.7]. A continuación el prototipo.



Figura 8.6: Prototipo - parte 1

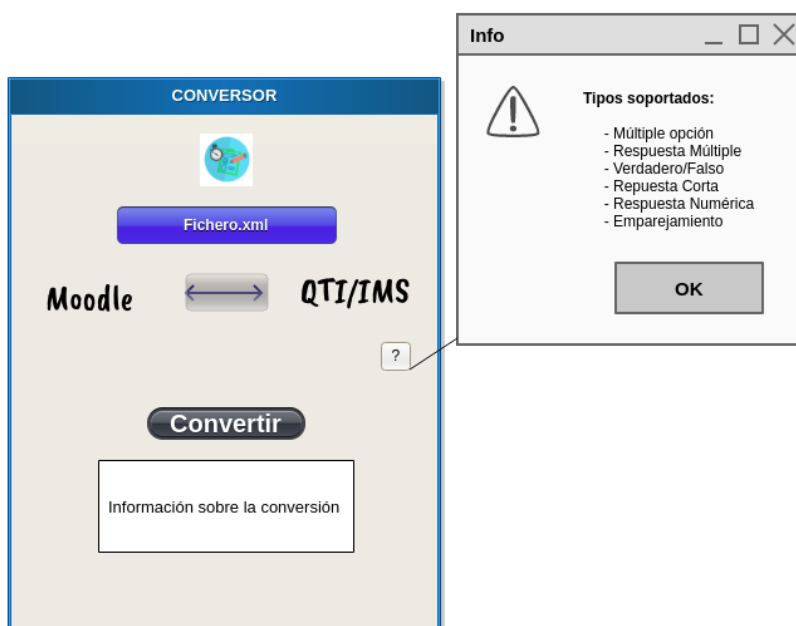


Figura 8.7: Prototipo - parte 2

Capítulo 9

Pruebas

La realización de las pruebas de funcionamiento es una parte esencial en el ciclo de vida de cualquier desarrollo de software. En este capítulo, se proporcionará una descripción del plan de pruebas que se ha seguido desde las etapas iniciales de desarrollo hasta las versiones finales del sistema.

9.1. Pruebas de caja blanca

Como parte del proceso de desarrollo, se realizaron pruebas exhaustivas en cada uno de los métodos que forman los módulos durante su fase de codificación.

9.2. Pruebas de caja negra

En las últimas etapas del proyecto, comenzaron a realizarse pruebas para comprobar:

- Correcto funcionamiento de la interfaz.
- Coherencia entre los datos introducidos y la salida generada, es decir, fichero en formato origen y el fichero en formato salida.
- Tiempo de respuesta óptimo.
- Respuesta del sistema ante la falta de datos.
- Comunicación con el back-end.

9.3. Pruebas de integración

Las pruebas de integración han sido parte del propio desarrollo, ya que todos los módulos interactúan entre si.

9.4. Pruebas de representación

Una vez el sistema se encuentra completamente integrado y en su versión final, se procede a supervisar de manera detallada que la conversión y/o representación de cada uno de los tipos de pregunta es la esperada. Para poder realizar este apartado, tuvimos que crear una cuenta con un espacio de estudio en cada plataforma correspondiente a usar los formatos, es decir, *Moodle* y *Canvas*. Por un lado en Canvas es relativamente fácil crear un espacio con un rol de profesor que nos permitiera publicar cuestionarios, pero por otro lado, Moodle, tuvimos que solicitar un curso de prueba en ENOA a infomoodle. Por tanto, los cuestionarios a convertir serán creados en la propia plataforma, exportados, convertidos con nuestro sistema, y luego importado en la otra plataforma.

9.4.1. Caso de prueba 01

Representación de preguntas del tipo **opción múltiple**. De *Canvas* a *Moodle*.

1. Representación *Canvas*: [Figura 9.1]

Pregunta 1 1 / 1 puntos

¿Cuál es la capital de Colombia?

¡Correcto!

☒ Bogotá

☐ Medellín

☐ Caracas

☐ París

Pregunta 2 0 / 1 puntos

¿Quién pintó *Las meninas*?

Respondido

☒ Francisco de Goya

☐ Salvador Dalí

Respuesta correcta ☐ Diego Velázquez

☐ Benito Antonio Martínez

Figura 9.1: Representación Canvas - prueba 1

2. Conversión: [Figura 9.2]

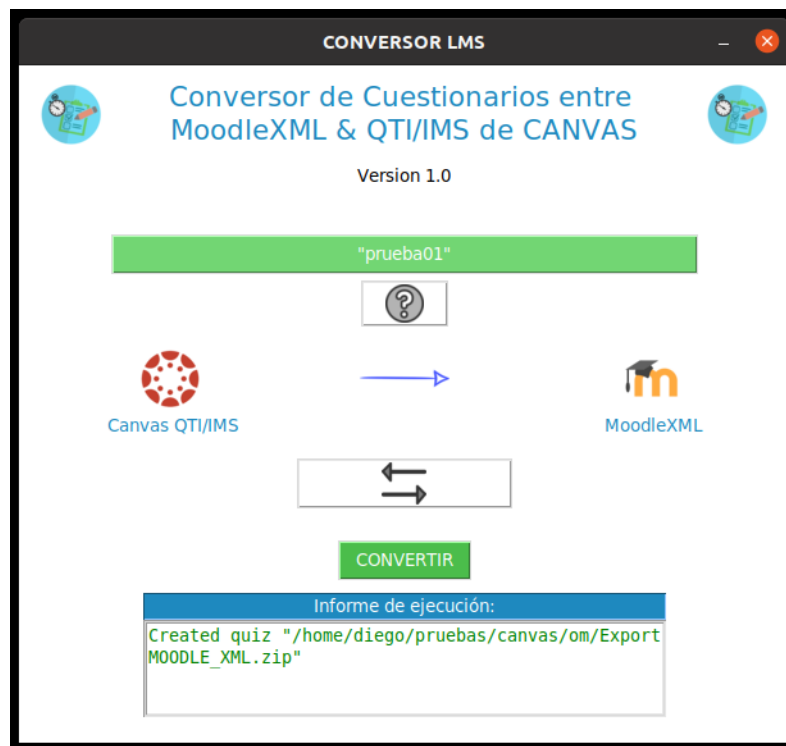


Figura 9.2: Conversión - prueba 1

3. Representación Moodle: [Figura 9.3]



Figura 9.3: Representación Moodle - prueba 1

9.4.2. Caso de prueba 02

Representación de preguntas del tipo **respuesta múltiple**. De *Canvas* a *Moodle*.

1. Representación *Canvas*: [Figura 9.4]

Pregunta 1 0 / 1 puntos

¿Cuál de los siguientes alimentos te gusta comer?

Respuesta correcta

Respondido

¡Correcto!

☐ Pasta

☒ Hamburguesa

☒ Ensaladas

☐ Guisos

Pregunta 2 1 / 1 puntos

¿Cuál de las siguientes categorías de aplicaciones usas en tu móvil?

☐ Juegos

☒ Música

☒ Redes sociales

☒ Aprendizaje

¡Correcto!

¡Correcto!

¡Correcto!

Figura 9.4: Representación Canvas - prueba 2

2. Conversión: [Figura 9.5]

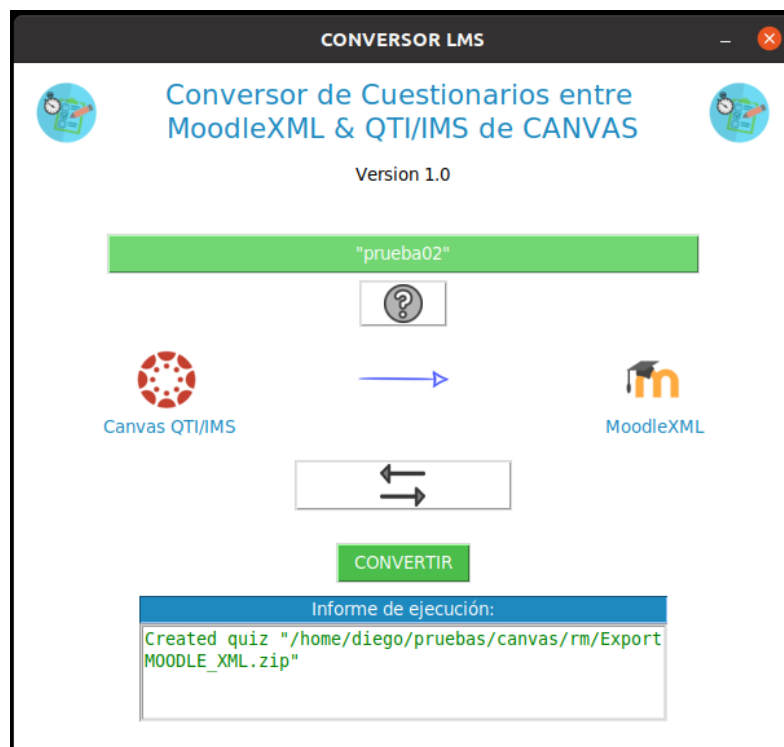


Figura 9.5: Conversión - prueba 2

3. Representación Moodle: [Figura 9.6]

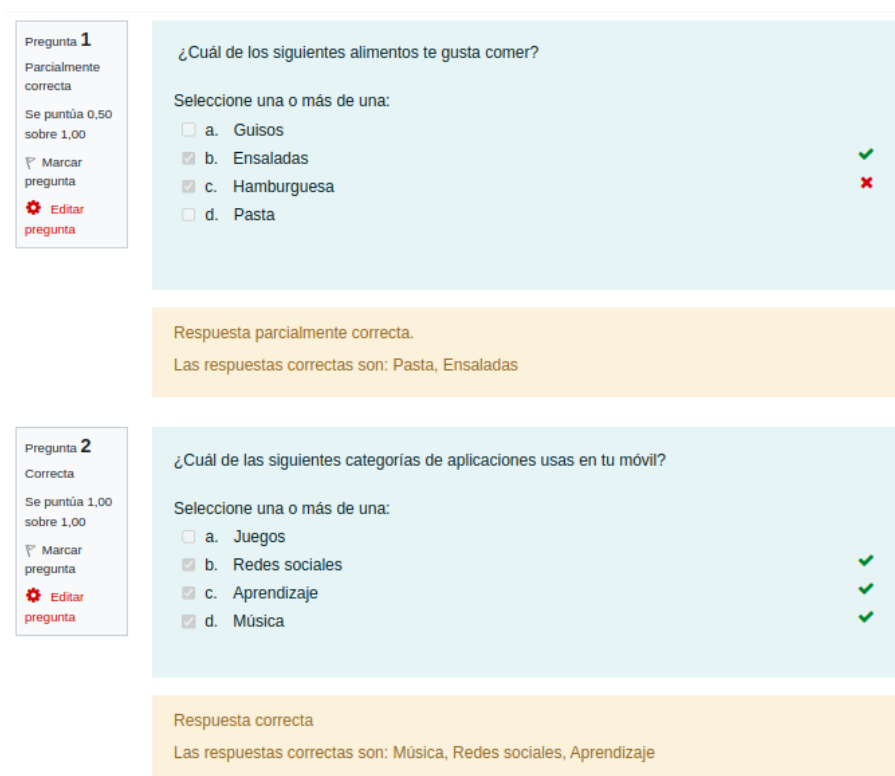


Figura 9.6: Representación Moodle - prueba 2

9.4.3. Caso de prueba 03

Representación de preguntas del tipo **verdadero/falso**. De *Canvas* a *Moodle*.

1. Representación *Canvas*: [Figura 9.7]

The image shows two question boxes from the Canvas LMS interface. The first box, 'Pregunta 1', has a score of '1 / 1 puntos' and contains the statement 'La caja negra de un avión es negra'. It has two radio button options: 'Verdadero' and 'Falso'. A green arrow labeled '¡Correcto!' points to the 'Falso' option. The second box, 'Pregunta 2', has a score of '0 / 1 puntos' and contains the statement 'El unicornio es el animal nacional de Escocia.'. It also has two radio button options: 'Verdadero' and 'Falso'. A grey arrow labeled 'Respuesta correcta' points to the 'Verdadero' option, and a red arrow labeled 'Respondido' points to the 'Falso' option, which is enclosed in a red rectangular box.

Figura 9.7: Representación Canvas - prueba 3

2. Conversión: [Figura 9.8]

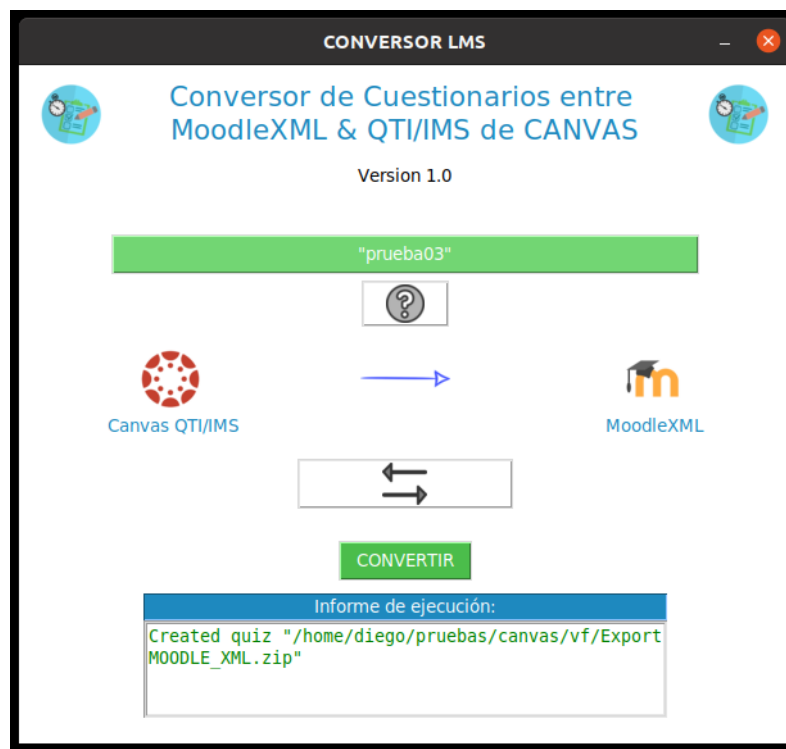


Figura 9.8: Conversión - prueba 3

3. Representación Moodle: [Figura 9.9]

Pregunta 1
Correcta
Se puntúa 1,00 sobre 1,00
🚩 Marcar pregunta
⚙ Editar pregunta

La caja negra de un avión es negra

Seleccione una:

☐ Verdadero

☒ Falso ✓

Son naranjas.
La respuesta correcta es 'Falso'

Pregunta 2
Incorrecta
Se puntúa 0,00 sobre 1,00
🚩 Marcar pregunta
⚙ Editar pregunta

El unicornio es el animal nacional de Escocia.

Seleccione una:

☐ Verdadero

☒ Falso ✗

La respuesta correcta es 'Verdadero'

Figura 9.9: Representación Moodle - prueba 3

9.4.4. Caso de prueba 04

Representación de preguntas del tipo **respuesta corta**. De *Canvas* a *Moodle*.

1. Representación *Canvas*: [Figura 9.10]

The image displays two examples of short-answer questions from the Canvas LMS interface. Each question is presented in a box with a header indicating the question number and the score (1 / 1 puntos). The first question, 'Pregunta 1', asks '¿Cuál es fórmula del agua?' and shows a correct answer 'H2o' with a green '¡Correcto!' banner. Below the answer, a 'Respuestas correctas' section lists 'h2o', 'H2O', 'H2o', and 'h2O'. The second question, 'Pregunta 2', asks '¿Cómo se escribe gato en inglés?' and shows a correct answer 'cat' with a green '¡Correcto!' banner. Below the answer, a 'Respuestas correctas' section lists 'cat', 'Cat', and 'CAT'.

Pregunta 1 1 / 1 puntos

¿Cuál es fórmula del agua?

¡Correcto!

H2o

Respuestas correctas

- h2o
- H2O
- H2o
- h2O

Pregunta 2 1 / 1 puntos

¿Cómo se escribe gato en inglés?

¡Correcto!

cat

Respuestas correctas

- cat
- Cat
- CAT

Figura 9.10: Representación Canvas - prueba 4

2. Conversión: [Figura 9.11]

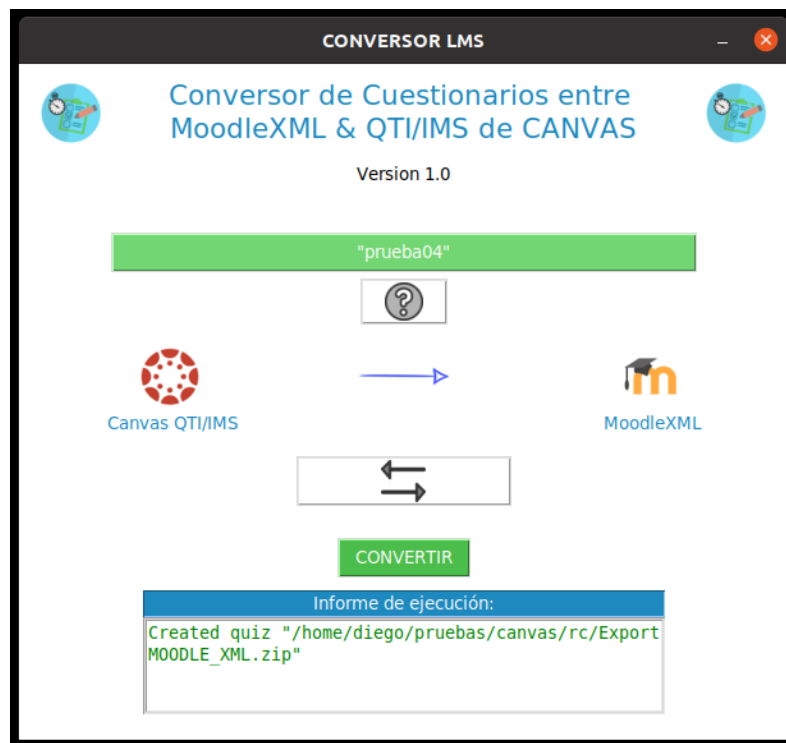


Figura 9.11: Conversión - prueba 4

3. Representación Moodle: [Figura 9.12]

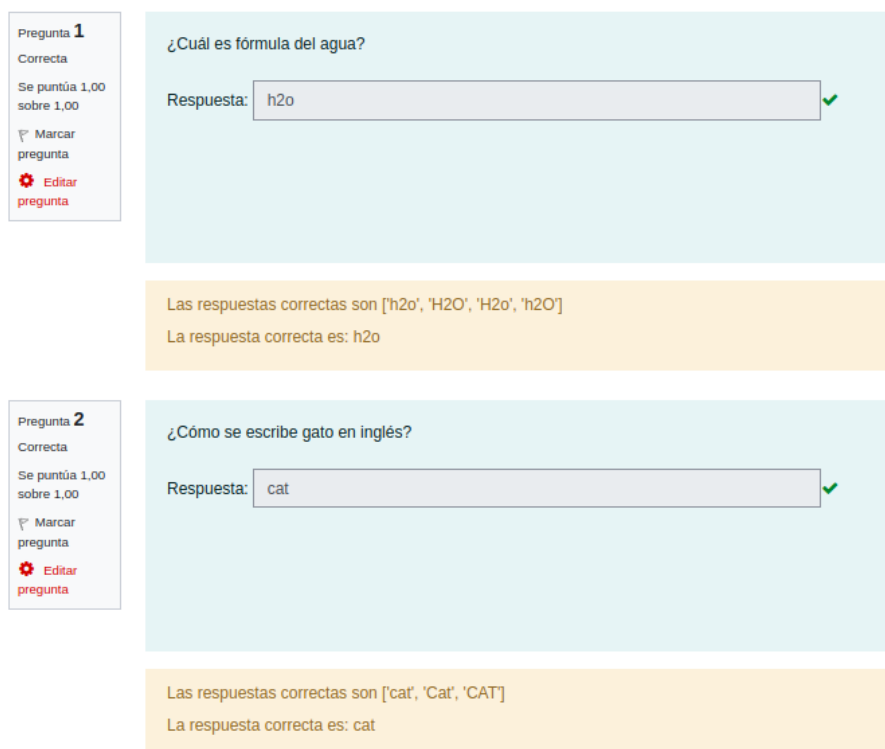


Figura 9.12: Representación Moodle - prueba 4

9.4.5. Caso de prueba 05

Representación de preguntas del tipo **respeusta numérica**. De *Canvas* a *Moodle*.

1. Representación *Canvas*: [Figura 9.13]

Pregunta 1 1 / 1 puntos

¿Cuanto vale el número PI

¡Correcto!

3,1416

Respuestas correctas 3,1416 (con margen: 0,01)

Pregunta 2 0 / 1 puntos

¿Cuál es la raíz de dos?

Respondido

1,39

Respuestas correctas 1,4142 (con margen: 0,02)

Figura 9.13: Representación Canvas - prueba 5

2. Conversión: [Figura 9.14]

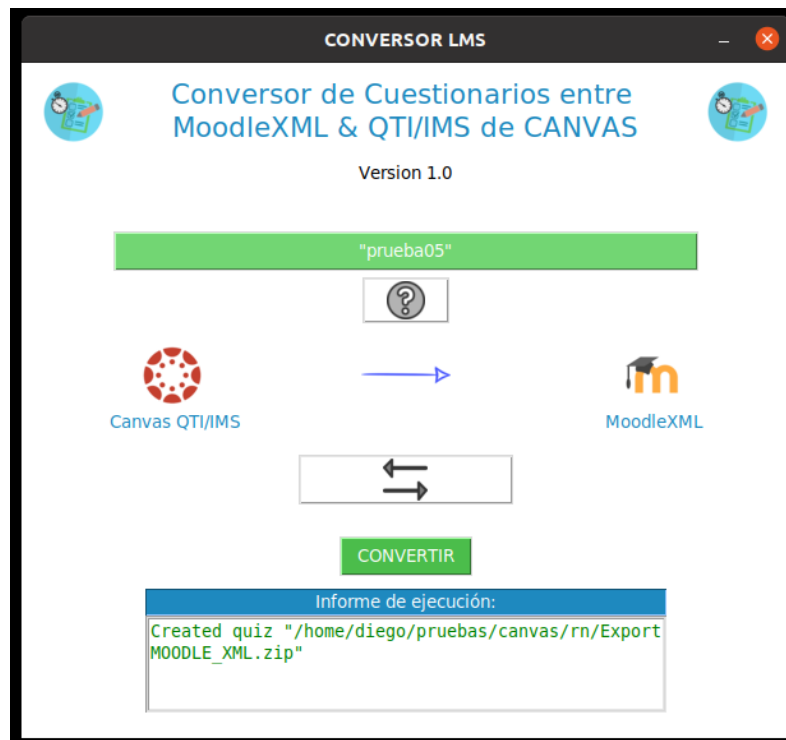


Figura 9.14: Conversión - prueba 5

3. Representación Moodle: [Figura 9.15]

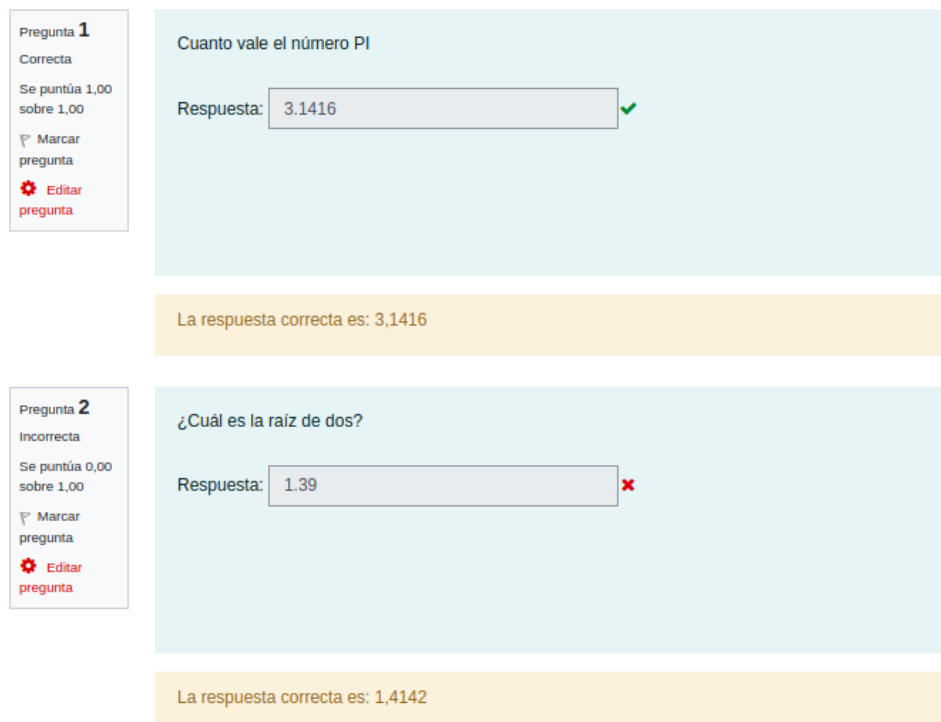


Figura 9.15: Representación Moodle - prueba 5

9.4.6. Caso de prueba 06

Representación de preguntas del tipo **emparejamiento**. De *Canvas* a *Moodle*.

1. Representación *Canvas*: [Figura 9.16]

Pregunta 1 0,5 / 1 puntos

Relacionar el animal con su tipo correspondiente.

¡Correcto!	Ballena	mamíferos
¡Correcto!	Tiburón	peces
Respondido	Cocodrilo	anhbio
		Respuesta correcta reptiles
Respondido	Rana	reptiles
		Respuesta correcta anfibio

Figura 9.16: Representación Canvas - prueba 6

2. Conversión: [Figura 9.17]



Figura 9.17: Conversión - prueba 6

3. Representación Moodle: [Figura 9.18]

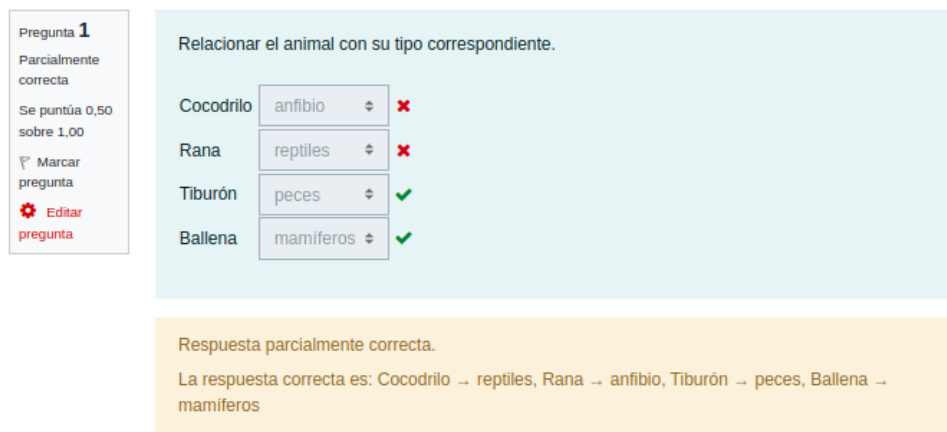


Figura 9.18: Representación Moodle - prueba 6

9.4.7. Caso de prueba 07

Representación de preguntas del tipo **opción múltiple**. De *Moodle* a *Canvas*.

1. Representación *Moodle*: [Figura 9.19]

Pregunta 1
Correcta
Se puntúa 1,00 sobre 1,00
⚙ Marcar pregunta
⚙ Editar pregunta

¿Quién fue Albert Einstein?

- ☐ a. Programador
- ☐ b. Químico
- ☐ c. Cantante
- ☒ d. Físico

Respuesta correcta
La respuesta correcta es: Físico

Pregunta 2
Incorrecta
Se puntúa 0,00 sobre 1,00
⚙ Marcar pregunta
⚙ Editar pregunta

¿Cómo se dice perro en Inglés?

- ☐ a. Rabitt
- ☐ b. Cat
- ☒ c. Perrow
- ☐ d. Dog

Respuesta incorrecta.
La respuesta correcta es: Dog

Figura 9.19: Representación Moodle - prueba 7

2. Conversión: [Figura 9.20]

CONVERSION LMS

Convertor de Cuestionarios entre MoodleXML & QTI/IMS de CANVAS

Version 1.0

"prueba07"

?

MoodleXML → Canvas QTI/IMS

↔

CONVERTIR

Informe de ejecución:

```
Created quiz "/home/diego/pruebas/moodle/ExportM00DLE_XML.zip"
```

Figura 9.20: Conversión - prueba 7

3. Representación *Canvas*: [Figura 9.21]

The image shows a screenshot of a Canvas LMS quiz interface. It contains two questions, each in a separate box. Question 1 is titled 'Pregunta 1' and is worth '1 / 1 puntos'. The question is '¿Quién fue Albert Einstein?'. The options are 'Físico', 'Químico', 'Cantante', and 'Programador'. A green arrow labeled '¡Correcto!' points to the 'Físico' option. Question 2 is titled 'Pregunta 2' and is worth '0 / 1 puntos'. The question is '¿Cómo se dice perro en Inglés?'. The options are 'Dog', 'Cat', 'Perrow', and 'Rabitt'. A grey arrow labeled 'Respuesta correcta' points to the 'Dog' option, and a red arrow labeled 'Respondido' points to the 'Perrow' option, which is highlighted with a red border.

Pregunta 1	1 / 1 puntos
¿Quién fue Albert Einstein?	
¡Correcto!	<input checked="" type="radio"/> Físico
	<input type="radio"/> Químico
	<input type="radio"/> Cantante
	<input type="radio"/> Programador

Pregunta 2	0 / 1 puntos
¿Cómo se dice perro en Inglés?	
Respuesta correcta	<input type="radio"/> Dog
	<input type="radio"/> Cat
Respondido	<input checked="" type="radio"/> Perrow
	<input type="radio"/> Rabitt

Figura 9.21: Representación Canvas - prueba 7

9.4.8. Caso de prueba 08

Representación de preguntas del tipo **respuesta múltiple**. De *Moodle* a *Canvas*.

1. Representación *Moodle*: [Figura 9.22]

The screenshot shows the Moodle question interface. On the left, a sidebar for 'Pregunta 1' indicates it is correct and worth 1.00 points. The main area displays the question '¿Cuáles son nombres de sistemas operativos?' with five options: a. Microsoft (checked, marked incorrect with a red X), b. Google (unchecked), c. Apple (unchecked), d. Windows XP (checked, marked correct with a green check), and e. Linux (checked, marked correct with a green check). Below the question, a yellow box states 'Respuesta correcta' and 'Las respuestas correctas son: Windows XP, Linux'.

The second question, 'Pregunta 2', asks '¿Cuáles de las siguientes palabras lleva tilde?' with five options: a. egoista (checked, correct), b. Aqui (checked, correct), c. fue (unchecked), d. Cuestionario (unchecked), and e. alegría (checked, correct). A yellow box below indicates 'Respuesta correcta' and 'Las respuestas correctas son: alegría, egoista, Aqui'.

Figura 9.22: Representación Moodle - prueba 8

2. Conversión: [Figura 9.23]

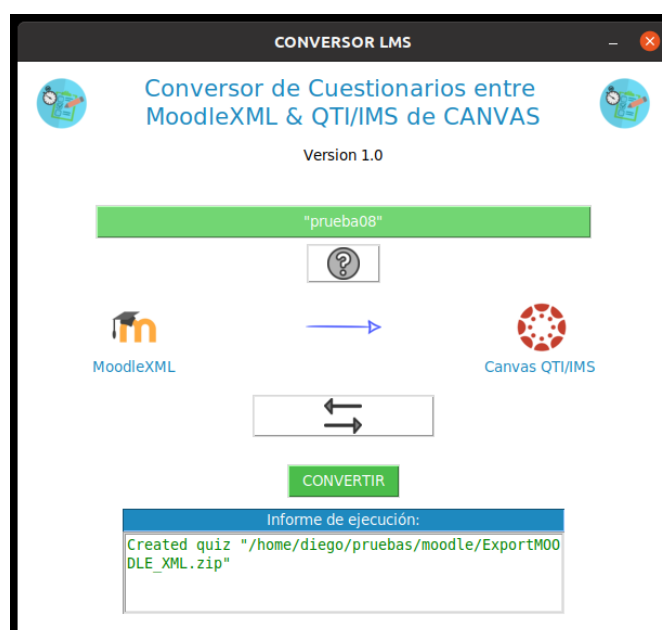


Figura 9.23: Conversión - prueba 8

3. Representación *Canvas*: [Figura 9.24]

Pregunta 1 0 / 1 puntos

¿Cuáles son nombres de sistemas operativos?

Respondido ☒ Apple

¡Correcto! ☒ Windows XP

Respuesta correcta ☐ Linux

☐ Google

Respondido ☒ Microsoft

Pregunta 2 1 / 1 puntos

¿Cuáles de las siguientes palabras lleva tilde?

¡Correcto! ☒ alegría

☐ fue

¡Correcto! ☒ egoista

☐ Cuestionario

¡Correcto! ☒ Aquí

Figura 9.24: Representación Canvas - prueba 8

9.4.9. Caso de prueba 09

Representación de preguntas del tipo **verdadero/falso**. De *Moodle* a *Canvas*.

1. Representación *Moodle*: [Figura 9.25]



The screenshot displays two quiz questions from Moodle. The first question, 'Pregunta 1', is marked 'Correcta' (Correct) with a score of 1.00. The question text is 'H&M son las siglas de 'Hennes & Mauritz''. The options are 'Verdadero' (selected with a green checkmark) and 'Falso'. The feedback below states 'Correcto' and 'La respuesta correcta es 'Verdadero''. The second question, 'Pregunta 2', is marked 'Incorrecta' (Incorrect) with a score of 0.00. The question text is 'En una baraja de cartas, el rey tiene bigote'. The options are 'Verdadero' (selected with a red X) and 'Falso'. The feedback below states 'La respuesta correcta es 'Falso''.

Figura 9.25: Representación Moodle - prueba 9

2. Conversión: [Figura 9.26]

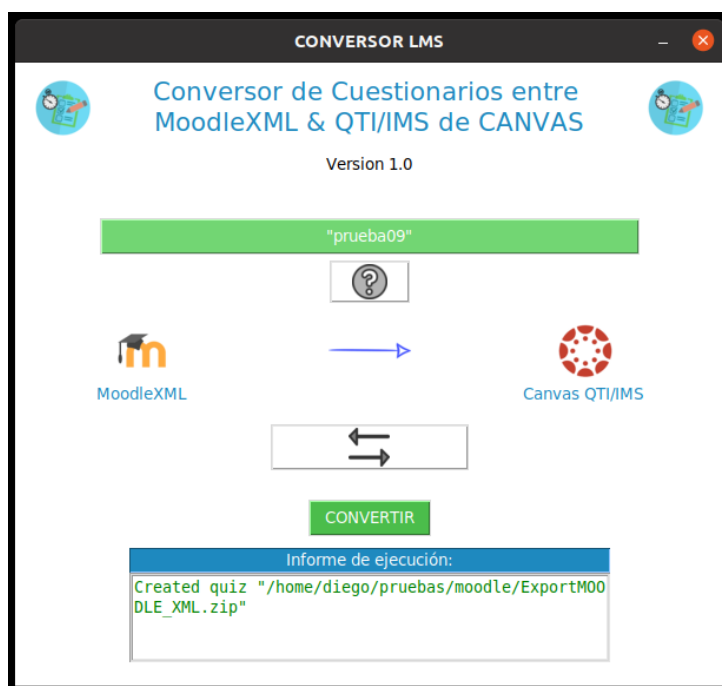


Figura 9.26: Conversión - prueba 9

3. Representación *Canvas*: [Figura 9.27]

The image shows a screenshot of a Canvas LMS quiz interface. It contains two questions, each in a separate box.

Pregunta 1 (0 / 1 puntos):
H&M son las siglas de 'Hennes & Mauritz'
The question has two radio button options: 'Verdadero' and 'Falso'. The 'Falso' option is selected. A red box highlights the 'Falso' option and a feedback message: 'Te pensabas que era 'Hombre y Mujer'?'. To the left of the question box, there are two labels: 'Respuesta correcta' (in a grey arrow) and 'Respondido' (in a red arrow).

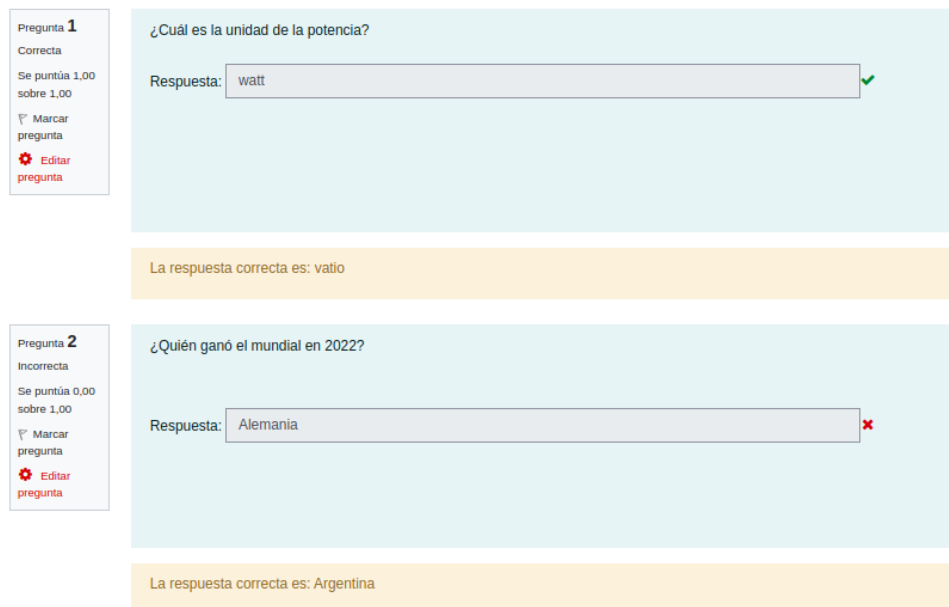
Pregunta 2 (1 / 1 puntos):
En una baraja de cartas, el rey tiene bigote
The question has two radio button options: 'Verdadero' and 'Falso'. The 'Falso' option is selected. To the left of the question box, there is a green arrow label: '¡Correcto!'.

Figura 9.27: Representación Canvas - prueba 9

9.4.10. Caso de prueba 10

Representación de preguntas del tipo **respuesta corta**. De *Moodle* a *Canvas*.

1. Representación *Moodle*: [Figura 9.28]



Pregunta 1
Correcta
Se puntúa 1,00 sobre 1,00
🚩 Marcar pregunta
⚙️ Editar pregunta

¿Cuál es la unidad de la potencia?

Respuesta: ✓

La respuesta correcta es: vatio

Pregunta 2
Incorrecta
Se puntúa 0,00 sobre 1,00
🚩 Marcar pregunta
⚙️ Editar pregunta

¿Quién ganó el mundial en 2022?

Respuesta: ✗

La respuesta correcta es: Argentina

Figura 9.28: Representación Moodle - prueba 10

2. Conversión: [Figura 9.29]



Figura 9.29: Conversión - prueba 10

3. Representación *Canvas*: [Figura 9.30]

Pregunta 1 1 / 1 puntos

¿Cuál es la unidad de la potencia?

¡Correcto!

watt

Respuestas correctas

- vatio
- watt
- J/s

Pregunta 2 0 / 1 puntos

¿Quien ganó el mundial en 2022?

Respondido

Alemania

Respuestas correctas

- Argentina
- Messi
- Perdió Francia

Figura 9.30: Representación Canvas - prueba 10

9.4.11. Caso de prueba 11

Representación de preguntas del tipo **repuesta numérica**. De *Moodle* a *Canvas*.

1. Representación *Moodle*: [Figura 9.31]

The screenshot displays two questions from a Moodle quiz. Each question is shown in a light blue box with a sidebar on the left containing metadata and action buttons.

Pregunta 1: Correcta. Se puntúa 1,00 sobre 1,00. ¿Cuántos huesos hay en total en el cuerpo? Respuesta: 205. La respuesta correcta es: 206.

Pregunta 2: Incorrecta. Se puntúa 0,00 sobre 1,00. ¿Cuál es el número atómico del oxígeno? Respuesta: 9. La respuesta correcta es: 8.

Figura 9.31: Representación Moodle - prueba 10

2. Conversión: [Figura 9.32]

The screenshot shows the 'CONVERSOR LMS' application window. The title bar reads 'CONVERSOR LMS'. The main content area has a header 'Convertor de Cuestionarios entre MoodleXML & QTI/IMS de CANVAS' and 'Version 1.0'. Below this, there is a green bar with the text '"prueba11"'. A question mark icon is in the center. Below the icon, there are logos for 'MoodleXML' and 'Canvas QTI/IMS' connected by a blue arrow. A box with two horizontal arrows (one pointing left, one pointing right) is below the logos. A green button labeled 'CONVERTIR' is below the arrows. At the bottom, there is a box labeled 'Informe de ejecución:' containing the text: 'Created quiz "/home/diego/pruebas/moodle/ExportMoodle_XML.zip"'. The window has standard OS controls (minimize, maximize, close) in the top right corner.

Figura 9.32: Conversión - prueba 11

3. Representación *Canvas*: [Figura 9.33]

Pregunta 1		1 / 1 puntos
¿Cuántos huesos hay en total en el cuerpo?		
¡Correcto!	<input type="text" value="205"/>	
Respuestas correctas 206 (con margen: 1)		

Pregunta 2		0 / 1 puntos
¿Cuál es el número atómico del oxígeno?		
Respondido	<input type="text" value="9"/>	
Respuestas correctas 8 (con margen: 0)		

Figura 9.33: Representación Canvas - prueba 11

9.4.12. Caso de prueba 12

Representación de preguntas del tipo **emparejamiento**. De *Moodle* a *Canvas*.

1. Representación *Moodle*: [Figura 9.34]

Pregunta **1**

Parcialmente correcta

Se puntúa 0,33 sobre 1,00

Una cada lugar con su país correspondiente

El Cristo Redentor	Brasil	✓
Torre Eiffel	EEUU	✗
Estatua de la Liberta	Francia	✗

Respuesta parcialmente correcta.

Ha seleccionado correctamente 1.

La respuesta correcta es: El Cristo Redentor → Brasil, Torre Eiffel → Francia, Estatua de la Liberta → EEUU

Figura 9.34: Representación Moodle - prueba 10

2. Conversión: [Figura 9.35]



Figura 9.35: Conversión - prueba 12

3. Representación *Canvas*: [Figura 9.36]

Pregunta 1		0,33 / 1 puntos
Una cada lugar con su país correspondiente		
Respondido	Estatua de la Liberta	Francia
		Respuesta correcta: EEUU
Respondido	Torre Eiffel	EEUU
		Respuesta correcta: Francia
¡Correcto!	El Cristo Redentor	Brasil

Figura 9.36: Representación Canvas - prueba 12

9.4.13. Caso de prueba 13

El programa indica que no se ha convertido el cuestionario, debido al formato del archivo origen. [Figura 9.37]. Es decir, el fichero seleccionado en esta prueba tenia un formato *moodleXML* y si nos fijamos en el formato origen-fin establecido es, de **QTI/IMS** a **moodleXML**, por ello, el programa ha indicado que no se ha podido crear la conversión.

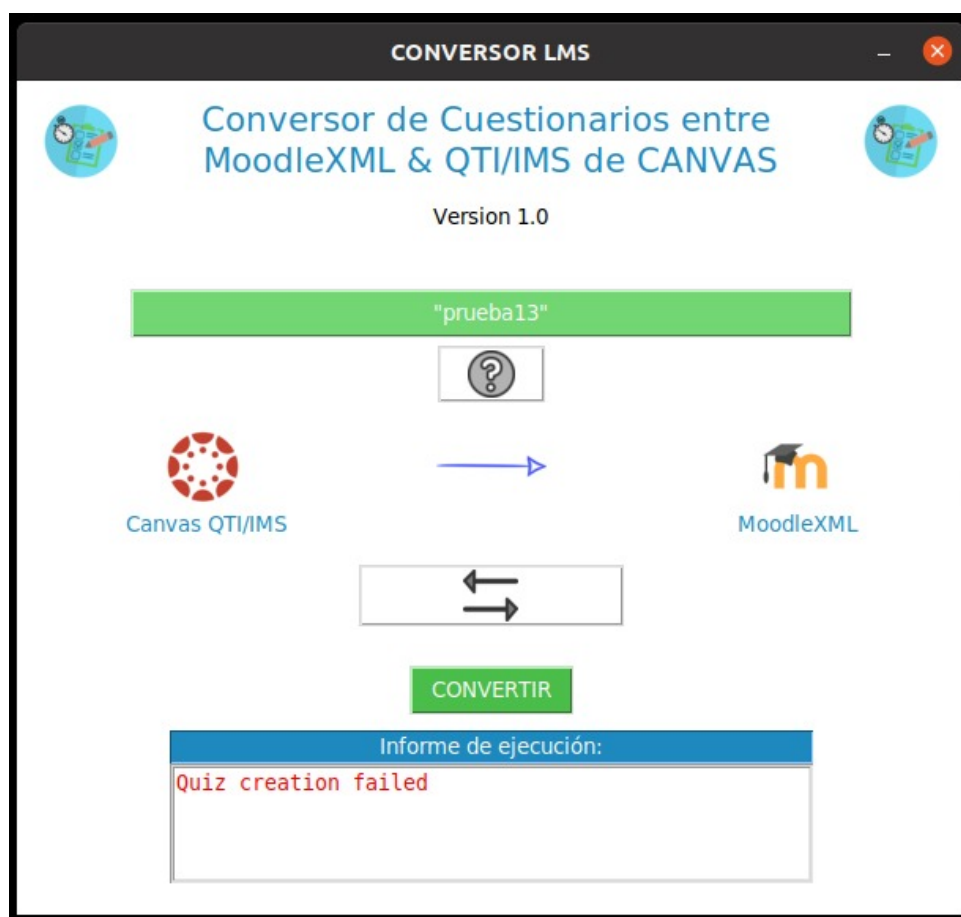


Figura 9.37: Validación estructura origen - prueba 13

9.4.14. Caso de prueba 14

El programa indica los tipos de preguntas que soporta [Figura 9.38]

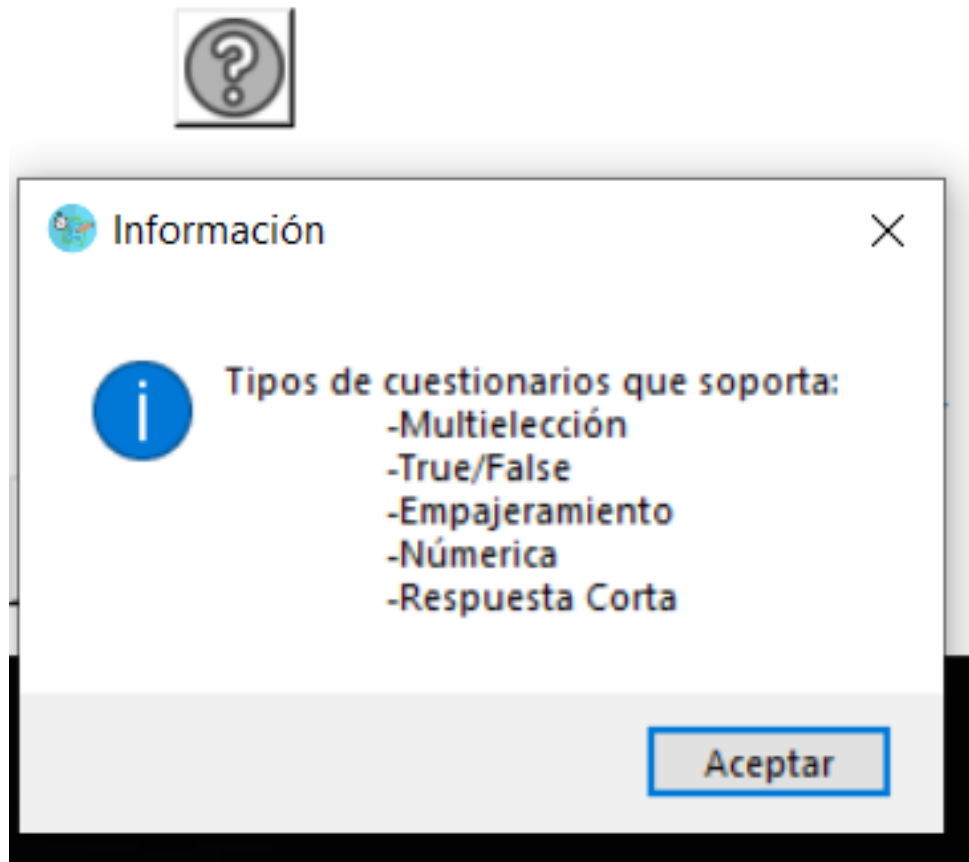


Figura 9.38: Tipos de pregunta - prueba 14

9.5. Matriz de cumplimiento

En la [Tabla 12.1] se comprobará que las pruebas realizadas cubren todos los requisitos funcionales definidos en el capítulo 7 de este documento.

Caso de prueba	Requisitos
P1	RF-1, RF-2, RF-4, RF-5, RF-6
P2	RF-1, RF-2, RF-5, RF-6
P3	RF-1, RF-2, RF-5, RF-6
P4	RF-1, RF-2, RF-5, RF-6
P5	RF-1, RF-2, RF-5, RF-6
P6	RF-1, RF-2, RF-5, RF-6
P7	RF-1, RF-3, RF-4, RF-5, RF-6
P8	RF-1, RF-3, RF-5, RF-6
P9	RF-1, RF-3, RF-5, RF-6
P10	RF-1, RF-3, RF-5, RF-6
P11	RF-1, RF-3, RF-5, RF-6
P12	RF-1, RF-3, RF-5, RF-6
P13	RF-7
P14	RF-8

Tabla 9.1: Fases de Proyecto

Capítulo 10

Conclusiones

Una vez concluidas todas las fases de desarrollo de la aplicación, se realizará en este capítulo una exposición de las conclusiones a las que se ha llegado después de la finalización de las mismas.

Durante la ejecución de este proyecto, he adquirido un mayor conocimiento sobre el proceso de desarrollo de software y he confirmado la importancia de una planificación temporal adecuada para abordar de manera organizada y coherente las etapas de estudio, análisis, diseño, implementación y pruebas. Siempre manteniendo en mente los objetivos establecidos. Por otro lado pude comprobar como la conversión de los cuestionarios XML simplifica el intercambio de datos entre las plataformas Moodle y Canvas. Permitiendo así una mejor gestión, ofreciendo una mayor reutilización e integración de los cuestionarios entre dichas plataformas, lo que se traduce a una mejora la eficiencia y la productividad. Es cierto que me he enfrentado desafíos adicionales a los previstos, especialmente en relación con la puesta en marcha de la aplicación. Sin embargo, los objetivos se han logrado de manera exitosa y el resultado ha sido satisfactorio. Destacando sobretodo que nuestro sistema proporciona una interfaz fácil de usar, esto permite a los usuarios, incluso aquellos sin conocimientos avanzados de programación o XML, realizar la conversión de manera rápida y eficiente.

Capítulo 11

Futuras mejoras

Una vez finalizado, y con el objetivo de facilitar posibles futuras expansiones de este proyecto, se presentarán algunas ideas que podrían desarrollarse para agregar mayor valor a la aplicación:

- Migrar la aplicación a una versión más reciente de python para prevenir errores de compatibilidades.
- Actualizar los formatos de las estructuras MoodleXML y QTI/IMS.
- Añadir más tipos de preguntas permitidas para convertir, como por ejemplo ensayos, selección de palabras, etc.
- Conversión de mayor cantidad de formatos de cuestionarios computacionales, es decir, ampliar el espectro de formatos a los que poder convertir y desde los que poder convertir.
- Sería conveniente realizar una interfaz web ya que puede ser encontrada más fácil.

Capítulo 12

Planificación del proyecto

La distribución temporal para el plan de trabajo y calendario que se ha seguido durante el desarrollo del proyecto es el que se desglosa a continuación en la [Tabla 12.1].

Fases	Inicio	Fin	Duración(días)	Horas Empleadas
Estudio y aprendizaje de las tecnologías a usar	20/02/2023	02/03/2023	9	40
Análisis y Especificación de requisitos	04/03/2023	15/03/2023	11	40
Diseño	16/03/2023	24/03/2023	8	40
Desarrollo e implementación de la solución	25/03/2023	26/04/2023	32	100
Evaluación y Pruebas	23/04/2023	02/05/2023	9	40
Documentación	01/05/2023	01/06/2023	31	40

Tabla 12.1: Fases de Proyecto

Además, se ha utilizado un diagrama de Gantt para distribuir temporalmente las fases del proyecto. Se ha decidido dividir la relación temporal por semanas para una mejor comprensión de la planificación.

Diagrama de Gantt

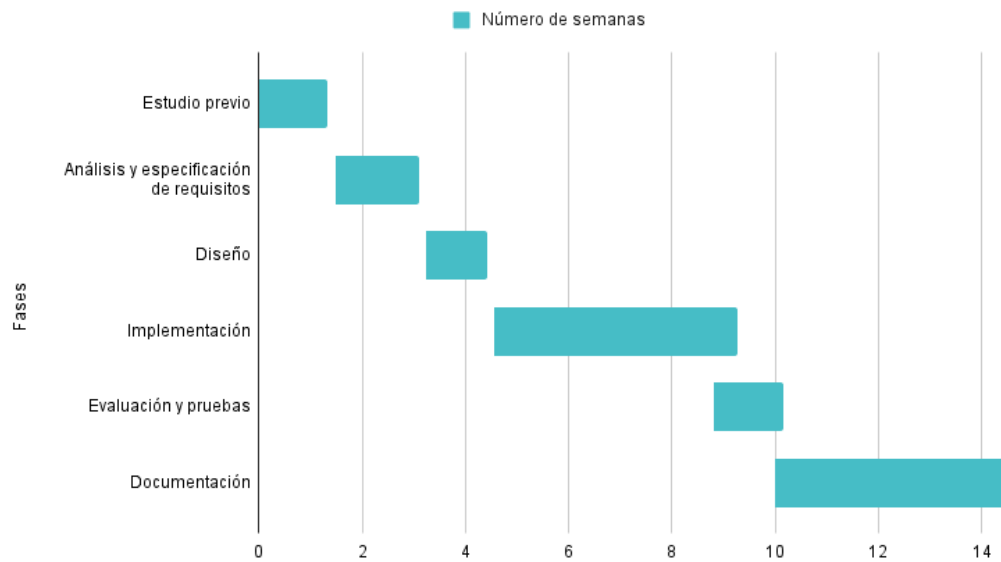


Figura 12.1: Diagrama de Gantt

Bibliografía

- [1] La evolución del e-learning: Pasado, presente y futuro. URL <https://editorialelearning.com/blog/evolucion-del-e-learning/>.
- [2] 5 maneras de evaluar el aprendizaje en línea - Observatorio / Instituto para el Futuro de la Educación. URL <https://observatorio.tec.mx/edu-bits-blog/cinco-maneras-de-evaluar-el-aprendizaje-en-linea/>.
- [3] Importancia del feedback en el aprendizaje online. URL <https://robertojasinski.com/importancia-del-feedback-en-el-aprendizaje-online/>.
- [4] Irwan Kautsar. MOODLE XML TO IMS QTI ASSESSMENT TEST PORTABILITY ON LEARNING MANAGEMENT SYSTEM. URL <https://www.academia.edu/5255704>.
- [5] Formato Moodle XML - MoodleDocs, . URL https://docs.moodle.org/all/es/Formato_Moodle_XML.
- [6] 1EdTech Question & Test Interoperability Specification Overview | IMS Global Learning Consortium, . URL <https://imglobal.org/question/index.html>.
- [7] Royyana M. Ijtihadie, Bekti C. Hidayanto, Achmad Affandi, Yoshifumi Chisaki, and Tsuyoshi Usagawa. Dynamic content synchronization between learning management systems over limited bandwidth network. *Human-centric Computing and Information Sciences*, 2(1):1–16, dec 2012. ISSN 21921962. doi: 10.1186/2192-1962-2-17/TABLES/2. URL <https://hcis-journal.springeropen.com/articles/10.1186/2192-1962-2-17>.
- [8] Importar preguntas - MoodleDocs. URL <https://docs.moodle.org/all/es/Importar{ }preguntas>.
- [9] Inicio - Respondus. URL <https://web.respondus.com/>.
- [10] Moodle plugins directory, . URL <https://moodle.org/plugins/>.

- [11] NB Software, hogar de HTML Meta-Data Editor y AppLaunch., . URL <https://www.nbsoftware.com/quizauthor/qtiviewer/index.html>.
- [12] Import any Word document quiz into Canvas, Blackboard or Moodle. URL <https://digitaliser.getmarked.ai/>.
- [13] Generador de Preguntas XML de Moodle. URL https://wwwmain.h.kobe-u.ac.jp/{~}nagasaka/research/xml{_{}}quiz/index.phtml.en.
- [14] text2qti/README.md en maestro · gpoore/text2qti · GitHub. URL <https://github.com/gpoore/text2qti/blob/master/README.md>.
- [15] GitHub - jderriks/moodle2qti: Convierta el archivo XML del banco de preguntas de Moodle a la estructura de carpetas QTI. Comprímalo y pruebe la importación a otro LMS o Testtool, . URL <https://github.com/jderriks/moodle2qti>.
- [16] procesamiento de datos QTI en Python; ejemplos usando pyslet, beautifulsoup4 y lxml. · GitHub. URL <https://gist.github.com/lsloan/1ba7539d097f9c622054c8e83a241297>.
- [17] xml.etree.ElementTree — La API XML de ElementTree — documentación de Python - 3.11.3. URL <https://docs.python.org/3/library/xml.etree.elementtree.html>.
- [18] lxml.objectify. URL <https://lxml.de/objectify.html>.
- [19] Python in Visual Studio Code. URL <https://code.visualstudio.com/docs/languages/python>.
- [20] Disponible Ubuntu 20.04.5 LTS con Linux 5.15 y más novedades. URL <https://www.muylinux.com/2022/09/02/ubuntu-20-04-5-lts/>.
- [21] Qué es Visual Studio Code y qué ventajas ofrece | OpenWebinars. URL <https://openwebinars.net/blog/que-es-visual-studio-code-y-que-ventajas-ofrece/>.
- [22] Introducción a Tcl/Tk (tkinter) - Recursos Python. URL <https://recursospython.com/guias-y-manuales/introduccion-a-tkinter/>.
- [23] J. Rumbaugh, G. Booch, and I. Jacobson. Unified modeling language, 2015. URL https://es.wikipedia.org/wiki/Lenguaje_unificado_de_modelado.

- [24] 1Interoperabilidad de preguntas y pruebas de EdTech v1.2: descripción general | Consorcio de Aprendizaje Global IMS, . URL <https://www.imsglobal.org/question/quiv1p2/imsqti{ }oviewv1p2.html{#}1399143>.
- [25] POO en Javascript - David Poza. URL <https://davidinformatico.com/poo-en-javascript>.
- [26] Íconos en ventanas de Tk (tkinter) - Recursos Python. URL <https://recursospython.com/guias-y-manuales/iconos-en-ventanas-de-tk-tkinter/>.
- [27] Lourdes Costero, Ranz Susana, Oliva Pérez, Miguel Ángel Sánchez, Fernández Dirigido, Baltasar Fernández, Manjón Iván, Martínez Ortiz, and Dpto S I P _____ . Editor y motor de QTI. 2005.
- [28] 1Descripción general de la especificación de interoperabilidad de preguntas y pruebas de EdTech | Consorcio de Aprendizaje Global IMS, . URL <https://www.imsglobal.org/question/index.html>.
- [29] Casos a incluir y casos a extender. URL <https://www.abiztar.com.mx/articulos/casos-a-incluir-casos-a-extender.html>.

Apéndice A

Manual de usuario

A.1. Introducción

Este manual ha sido creado para ayudar al usuario final a instalar, utilizar y desinstalar la aplicación. Si se necesita información más detallada sobre el funcionamiento o diseño, te recomendamos consultar el manual técnico.

A.2. Instalación

A continuación, se detallarán los pasos a seguir para la instalación del producto.

A.2.1. Descargar aplicación

Descargar la carpeta con la aplicación web.

- https://github.com/DiegoHigueta/ConversorLMS_TFG.git

En este repositorio podremos encontrar lo siguiente:

- Código fuente de la aplicación.
- Ejecutables de la aplicación para los sistemas operativos:
 - Linux.
 - Windows.
 - Mac.
- Manuales/Documentación.

A.2.2. Lanzar el proyecto

Dependiendo en el sistema operativo que se encuentre, deberá hacer uso del ejecutable correspondiente tan solo pulsando sobre:

- Linux → gui
- Windows → gui.exe
- Mac → gui.app

A.2.3. Posibles errores

El proceso que se va a describir a continuación no tiene que realizarlo siempre, solamente si durante el lanzamiento del ejecutable ha surgido algún tipo de error, en este caso, podremos ejecutar el código fuente, para ello deberemos tener instalado en su dispositivo:

- python (<https://www.python.org>)
- La biblioteca **lxml** de python (**pip install lxml**)

Luego, tendrá que situarse en carpeta del código fuente descargada anteriormente, abrir una terminal (Línea de comandos) y ejecutar la siguiente sentencia:

- **Linux**

```
python3 gui.py
```

- **Windows**

```
python .\gui.py
```

- **Mac**

```
python gui.py
```

A.3. Uso de la aplicación web

Una vez la aplicación ha sido lanzada el funcionamiento es sencillo e intuitivo.

A.3.1. Pantalla de Inicio



Figura A.1: Manual de usuario - Inicio

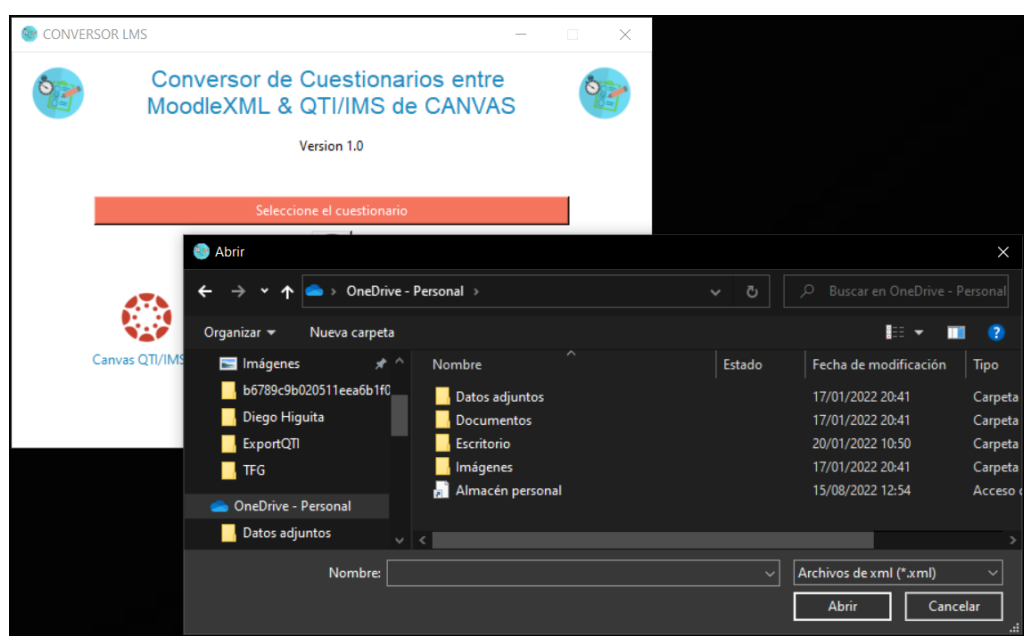
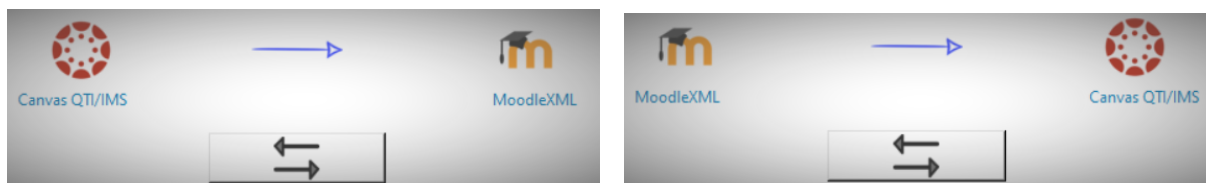


Figura A.2: Manual de usuario - Seleccionar Cuestionario

Como podemos observar, nada más lanzar la aplicación nos encontramos con la pantalla principal [Figura A.1] , donde podremos seleccionar el cuestionario [Figura A.2] con el botón **Seleccionar**, este cambiará de color *rojo* a *verde*. También podremos intercambiar(elegir) entre los formatos origen y fin. [Figuras A.3a y A.3b]



(a) de Canvas a Moodle .

(b) de Moodle a Canvas

Figura A.3: Manual de usuario - Elección de formato Origen-Fin

Puedes saber que tipo de preguntas podrás obtener conversión. [Figura A.4]

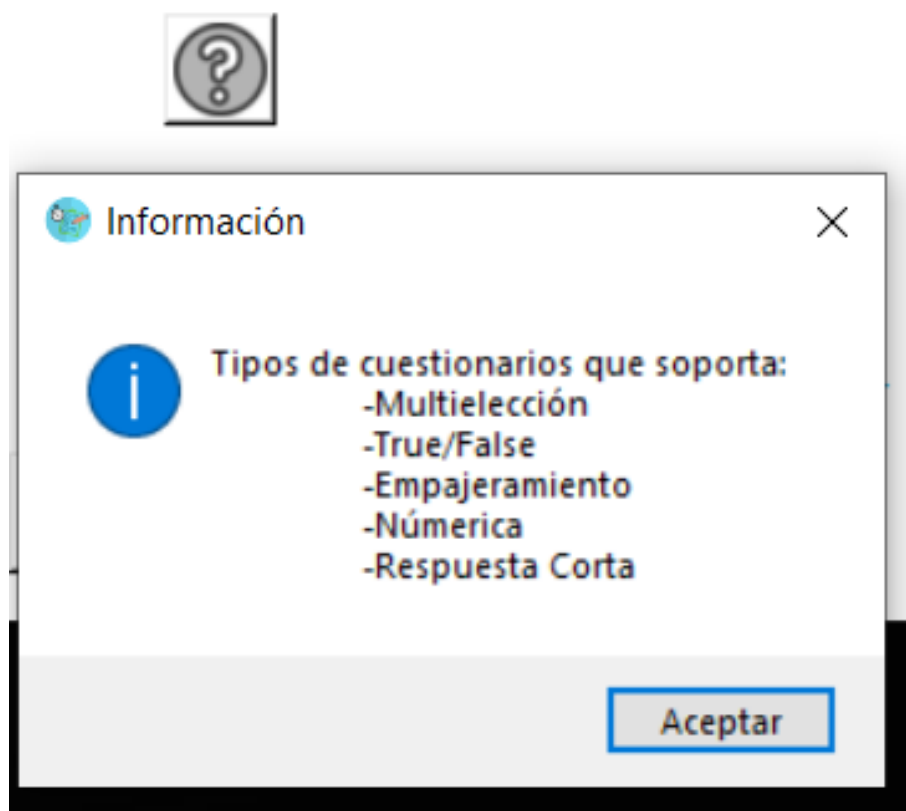


Figura A.4: Manual de usuario - Tipo de preguntas soportadas

A.3.2. Ampliación

Una vez hemos seleccionado el cuestionario a convertir y establecer cual será el formato de origen-fin, se nos abrirá/ampliará la ventana de inicio.

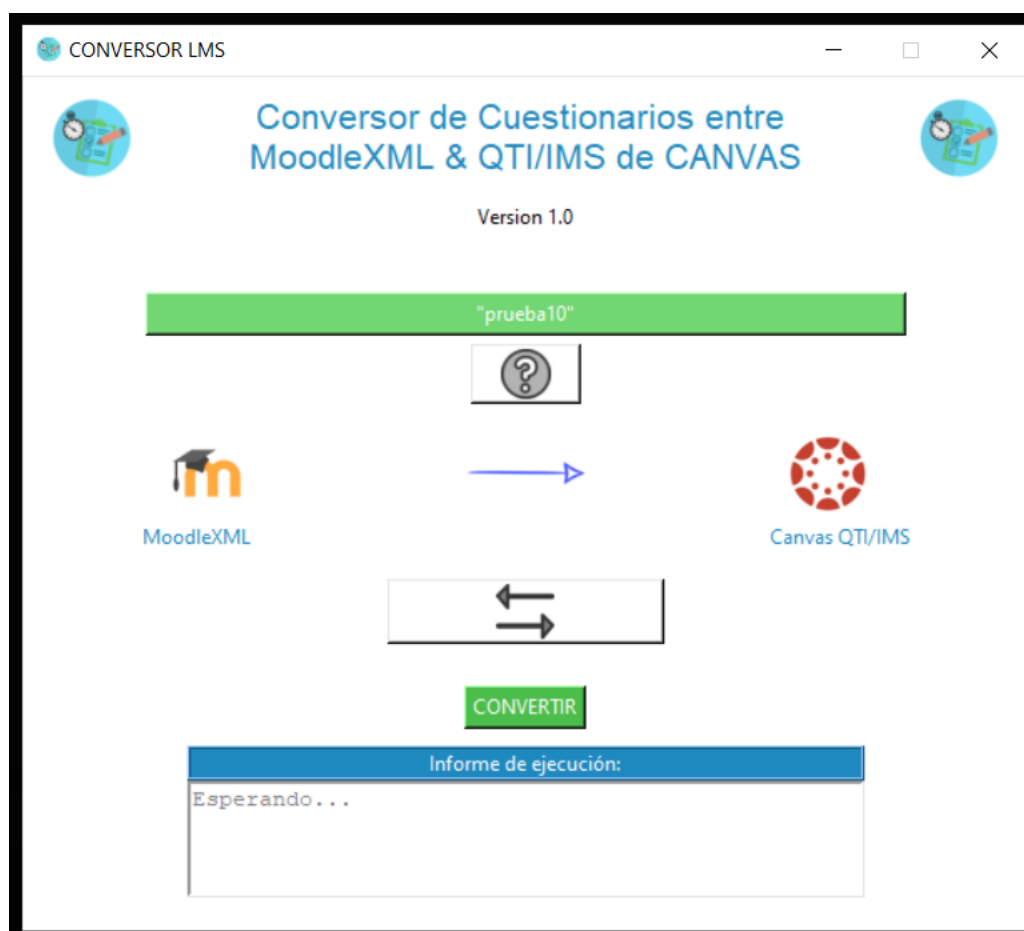
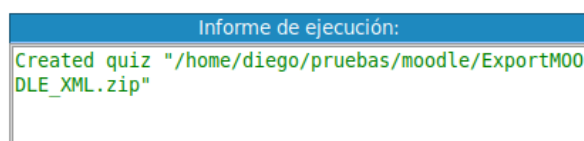


Figura A.5: Manual de usuario - Ampliación

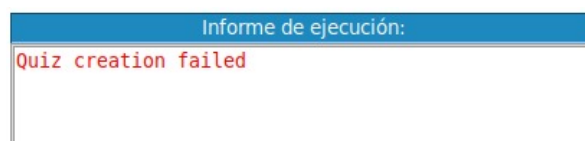
Como se observa en la [Figura A.5], esta ampliación añade dos elementos más a la pantalla, por un lado el botón de [**CONVERTIR**] y un cuadro de información [**Informe de Ejecución**].

A.3.3. Conversión/Ejecución

Teniendo seleccionado el cuestionario, y habiendo seleccionado el formato origen-fin en la pantalla inicial podremos presionar el botón **CONVERTIR**, y podremos ver donde se encuentran el nuevo cuestionario convertido al formato fin. Si no se ha logrado la conversión, se informará un mensaje de error. [Figuras A.6a y A.6b]



(a) Localización de la conversión



(b) Error

Figura A.6: Manual de usuario - Informe de ejecución

Una vez convertido, podremos subir/importar el cuestionario a la plataforma correspondiente. Allí podremos visualizarlo y realizar los cambios que consideremos oportunos.

Nota: Se sugiere encarecidamente que antes de importar el cuestionario, lo pases por un proceso de verificación con una herramienta para XML. Una opción simple es abrir el cuestionario (el fichero) utilizando su navegador, como por ejemplo **Google Chrome** o **Firefox**. El uso de estas herramientas pueden ayudar a detectar posibles errores o problemas de formato en el fichero XML antes de su importación.

A.3.4. Errores

Por último, si el cuadro **Informe de ejecución** nos indica que no se ha podido realizar la conversión puede deberse a lo siguiente:

- No ha establecido el formato origen-fin correctamente.
- El fichero seleccionado no corresponde a ningún de los formatos permitidos.
- El fichero seleccionado no tiene permisos de lectura.
- Existe la posibilidad que el programa falle debido a una posible etiqueta que no se he tenido en cuenta en el desarrollo.

Si esta usando la aplicación en el sistema operativo *Windows*, es posible que al pasar el fichero por una herramienta XML obtenga un error y no pueda observarlo. Esto se debe a que en la primera linea del fichero hay más contenido de lo que debería. **Solución:** Abrir el fichero con un editor de texto, y justo después de la siguiente sentencia:

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
```

Debe presionar la tecla **Intro** (Un salto de linea) de su teclado, dejando así en la primera linea del fichero solo la sentencia indicada. Ahora si podrá observar el fichero con una herramienta XML.

A.4. Desinstalación

Si se desea desinstalar la aplicación, debes seguir los siguientes pasos:

1. Encuentra la ubicación del ejecutable. Normalmente, se encuentra en la ubicación de la carpeta que especificaste al descargarlo.
2. Elimina el ejecutable y todos los archivos asociados a él. Esto puede incluir archivos adicionales, como bibliotecas o recursos que se hayan empaquetado con el ejecutable. Asegúrate de eliminar todos los archivos relacionados para una desinstalación completa.
3. Si creó un acceso directo o un enlace en el menú de inicio, también debes eliminarlos manualmente. Puedes hacerlo haciendo clic derecho en el acceso directo y seleccionando *Eliminar* o navegando al menú de inicio y eliminando el enlace correspondiente.

Apéndice B

Manual de código

B.1. Introducción

Este manual de código, pertenece al proyecto: *Aplicación Conversor de Cuestionarios entre XML Test de Moodle y QTI/IMS de CANVAS*.

El lenguaje principalmente utilizado ha sido Python, con él se ha diseñado toda la lógica de conversión de los elementos de los cuestionarios, haciendo uso de las librerías *objectify* y *ElementTree*, con sus métodos correspondientes. También se ha hecho uso de Python para todos los aspectos relativos a la interfaz gráfica del sistema.

B.2. Organización del código

En este apartado, se muestra a modo de esquema la forma en la que se encuentra organizado todo el código perteneciente al proyecto. [Figura B.1]

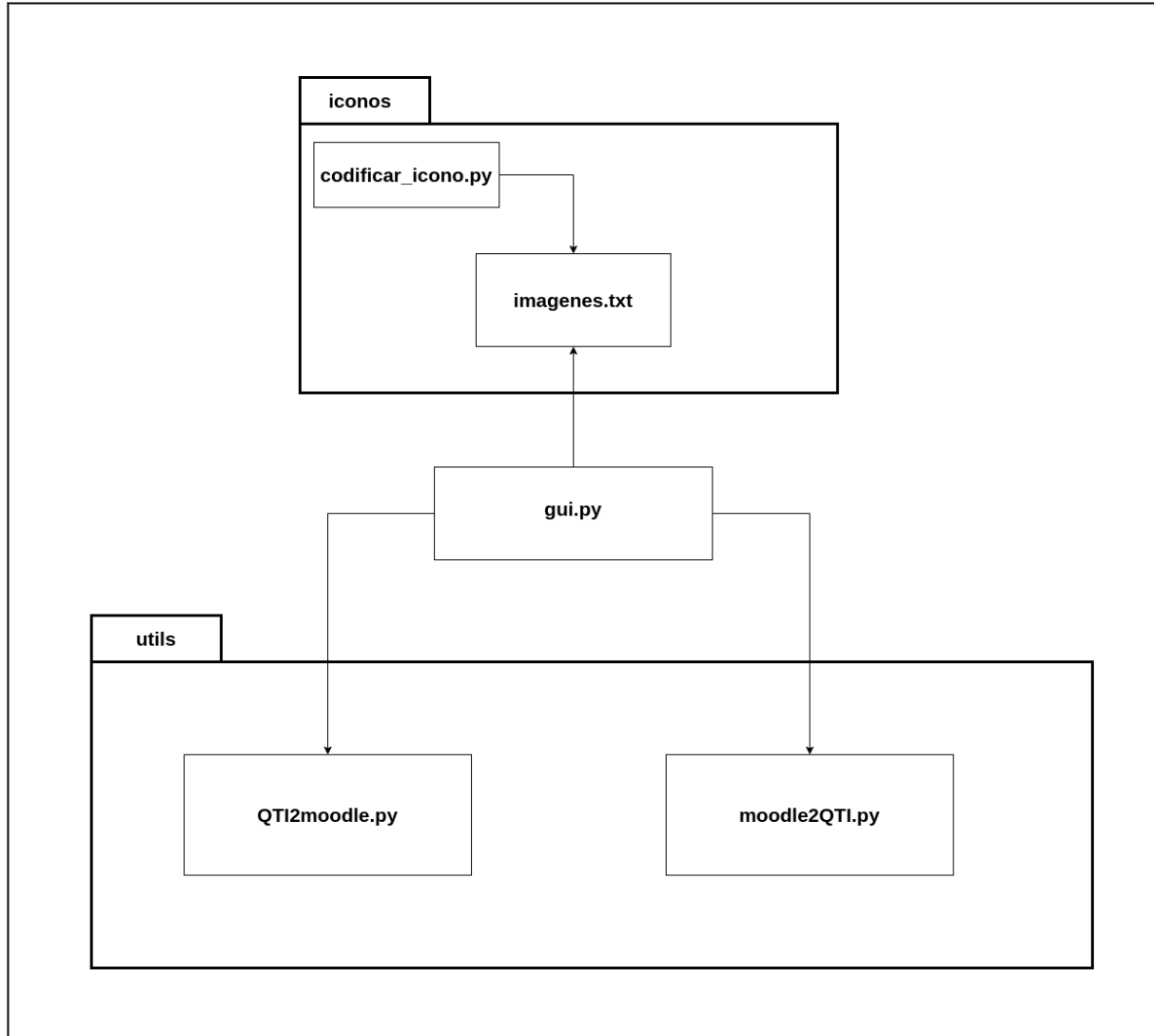


Figura B.1: Organización del código

Como se puede observar en la [Figura 8.4], los archivos que componen nuestro módulo se encuentran repartidos en dos carpetas distintas, y el archivo **gui.py**.

Carpeta iconos

Contiene todos los iconos de la interfaz de la aplicación.

- **codificar_icono.py**. Es una pequeña función, que codifica los iconos que se muestran en la interfaz [26]. Esta codificación se realiza ya que nuestra aplicación no está solo enfocada en Windows, el cual si permitiría cargar los iconos desde el ejecutable. Por lo que el ejecutable de nuestra aplicación de escritorio, deberá tener en el misma carpeta, el siguiente archivo a describir (**imagenes.txt**).
- **imagenes.txt**. Almacena los iconos que se muestran en la interfaz en Base64, luego el main de gui.py lee el fichero y usa los elementos.

Carpeta utils

Contiene información relevante para el funcionamiento de la aplicación.

- **QTI2moodle.py** Archivo que contiene la clase *QTI2moodle*. Es llamado desde *gui.py*.
- **moodle2QTI.py** Archivo que contiene la clase *moodle2QTI*. Es llamado desde *gui.py*.

Archivo gui.py

Como se explicó anteriormente, es el encargado de enseñar la interfaz, gestionar los eventos de clicks y/o ratón, para así coordinar las funciones de conversión, y contiene funciones como las siguientes:

- **main()**. Es el primer módulo de Python especificado por el usuario que empieza a ejecutarse. Contiene el código que genera la visualización de la interfaz, inicialización de variables y finaliza llamando al archivo correspondiente para la conversión.
- **informacion()**. Es una función que representa un modal, el cual enseña un mensaje de información sobre los tipos de cuestionarios que puede convertir.
- **leerIconos()**. Obtiene los iconos almacenados en el fichero **imagenes.txt**.

B.3. Ficheros

El conversor de cuestionarios está programado completamente desde cero, por lo que a continuación se mostrará todos los nuevos ficheros.

B.3.1. Ficheros nuevos

gui.py

```
1  #!/usr/bin/env python
2
3  import os
4  import pathlib
5  import shutil
6  import time
7  import tkinter as tk
8  from tkinter import ttk
9  from tkinter import PhotoImage
10 import tkinter.filedialog
11 import webbrowser
12 from tkinter import filedialog
13 from pathlib import Path
14 import utils.QTI2moodle as qti
15 import utils.moodle2QTI as moodle
16 from base64 import b64decode
17 from tkinter import messagebox as mb
18
19 """
20 Interfaz del sistema
21 """
22
23 """
24 Obtiene los iconos almacenados en el fichero "imagenes.txt"
25 """
26 def leerIconos(fichero_icono):
27
28     datos = []
29     with open(fichero_icono) as fname:
30         lineas = fname.readlines()
31         for linea in lineas:
32             datos.append(linea.strip('\n'))
33
34     return datos
35
36 """
37 Modal que enseña un mensaje de información sobre los tipos de cuestionarios que puede
38 convertir
39 """
40 def informacion():
41     mb.showinfo(
42         "Información"
43         , "Tipos de cuestionarios que soporta:\n\t-Multielección\n\t-True/False\n\t-
44           Empaquetamiento\n\t-Númerica\n\t-Respuesta Corta"
```

```

44         )
45
46
47 """
48 Función principal que se ejecuta y enseña la interfaz
49 """
50 def main():
51
52     file_name = ''
53
54     #Ventana principal
55     window = tk.Tk()
56     #Definimos ciertos valores de la ventana
57     window.title('CONVERSION LMS')
58     window.columnconfigure(0, weight=1)
59     window.rowconfigure(0, weight=1)
60     window.config(width=400, height=300, bg='white')
61     window.resizable(width=False, height=False)
62     #window.eval('tk::PlaceWindow . center') #En el caso que queramos centrarla en la
        pantalla
63
64     #Creacion de botones y elementos
65
66     #####
67     run_button = tk.Button(window, text= "CONVERTIR")
68     run_message_label = tk.Label(
69         window,
70         text='\nInforme de ejecución:\n',
71         bg = "#1E89BF",
72         fg= "white",
73         relief='ridge',
74         width=50,
75         height=1,
76     )
77     run_message_frame = tk.Frame(
78         window, width=50, height=25,
79         borderwidth=1, relief='sunken', bg='white',
80     )
81     run_message_text = tk.Text(run_message_frame, height=4, width=50)
82     #####
83
84
85     current_row = 0
86
87     iconos=leerIconos("iconos/imagenes.txt")#Obtenemos las imagenes que se usaran en
        la interfaz
88     icono = PhotoImage(data=b64decode(iconos[0]))
89     info = PhotoImage(data=b64decode(iconos[5]))
90
91     #-----
92     header_label_icono_1 = tk.Label(window, image=icono)
93     header_label_icono_1.grid(
94         row=current_row, column=0, columnspan=1, padx=(15,0), pady=10
95     )
96     header_label_icono_1['bg']="white"
97

```

```
98     #-----
99     header_label = tk.Label(
100         window,
101         text='Conversor de Cuestionarios entre \nMoodleXML & QTI/IMS de CANVAS',
102         font=(None, 16),
103     )
104
105     header_label.config(
106         bg= "white",
107         fg= "#1E89BF"
108     )
109
110     header_label.grid(
111         row=current_row, column=1, columnspan=1, padx=5,pady=10,
112         sticky='nsew',
113     )
114     #-----
115
116
117     header_label_icono_2 = tk.Label(
118         window,
119         image=icono
120     )
121     header_label_icono_2['bg']="white"
122     header_label_icono_2.grid(
123         row=current_row, column=2,columnspan=1,padx=(0,15)
124     )
125
126     #-----
127
128     # current_row += 1
129     # header_link_label = tk.Label(
130     #     window,
131     #     text='github.com/gpoore/text2qti',
132     #     font=(None, 14), fg='blue', cursor='hand2',
133     # )
134     # header_link_label.bind('<Button-1>', lambda x: webbrowser.open_new('https://
135     #     github.com/DiegoHiguita/ConversorLMS_TFG.git'))
136     # header_link_label.grid(
137     #     row=current_row, column=1, columnspan=1, padx=(30, 30),
138     #     sticky='nsew',
139     # )
140
141     current_row += 1
142     version_label = tk.Label(
143         window,
144         text=f'Version 1.0',
145         bg="white"
146     )
147
148     version_label.grid(
149         row=current_row, column=1, columnspan=1, padx=(30, 30), pady=(0, 30),
150         sticky='nsew',
151     )
152     current_row += 1
```

```

153     last_dir = None
154
155     #Función que permite navegar por las carpetas del dispositivo
156     def browse_files():
157         nonlocal file_name
158         nonlocal last_dir
159         if last_dir is None:
160             initialdir = pathlib.Path('~').expanduser()
161         else:
162             initialdir = last_dir
163         file_name = tkinter.filedialog.askopenfilename(
164             initialdir=initialdir,
165             filetypes=(
166                 ("Archivos de xml", "*.xml"),
167                 ("Todos los archivos", "*.*")
168             )
169         )
170         if file_name:
171             if last_dir is None:
172                 last_dir = pathlib.Path(file_name).parent
173             file_browser_button.config(text=f'"{pathlib.Path(file_name).stem}"', fg="
174                 white", bg="#72D673", activebackground="#67BF67")
175             run()
176         else:
177             file_browser_button.config(text=f'Seleccione el cuestionario', fg="white"
178                 , bg="#F5745D", activebackground="#DC4F36")
179             run_button.grid_remove()
180             run_message_frame.grid_remove()
181             run_message_label.grid_remove()
182
183     file_browser_button = tk.Button(
184         window,
185         text='Seleccione el cuestionario',
186         fg="white",
187         bg="#F5745D",
188         command=browse_files,
189         activebackground="#DC4F36",
190         activeforeground="white"
191     )
192     file_browser_button.grid(
193         row=current_row, column=1, columnspan=1, padx=(5, 5), pady=(5, 0),
194         sticky='nsew',
195     )
196
197     current_row += 1
198
199     infoButton = tk.Button(
200         window,
201         image=info,
202         bg="white",
203         command=informacion
204     )
205
206     infoButton.grid(

```

```
207         row=current_row, column=1, columnspan=1, padx=200, pady=(5,15),
208         sticky='nsew',
209     )
210
211     current_row += 1
212
213     # -----
214     moodle_ico = PhotoImage(data=b64decode(iconos[1]))
215     canvas_ico = PhotoImage(data=b64decode(iconos[2]))
216     flecha = PhotoImage(data=b64decode(iconos[3]))
217     cambio = PhotoImage(data=b64decode(iconos[4]))
218
219
220     flecha_label = tk.Label(
221         window,
222         image=flecha
223     )
224     flecha_label['bg']='white'
225     flecha_label.grid(
226         row=current_row, column=1, columnspan=1, padx=(0,0),
227         sticky='nsew',
228     )
229
230
231     tipo1_label = tk.Label(
232         window,
233         image=canvas_ico
234     )
235     tipo1_label['bg']='white'
236     tipo1_label.grid(
237         row=current_row, column=1, columnspan=1, padx=(25,0),
238         sticky='w',
239     )
240
241     tipo2_label = tk.Label(
242         window,
243         image=moodle_ico
244     )
245     tipo2_label['bg']='white'
246     tipo2_label.grid(
247         row=current_row, column=1, columnspan=1, padx=(0,15),
248         sticky='e',
249     )
250
251     current_row += 1
252
253     tipo1_label_text = tk.Label(
254         window,
255         text="Canvas QTI/IMS",
256         bg="white",
257         fg="#1E89BF"
258     )
259
260     tipo1_label_text.grid(
261         row=current_row, column=1, columnspan=1,
262         sticky='w',
```

```

263     )
264
265     tipo2_label_text = tk.Label(
266         window,
267         text="MoodleXML",
268         bg="white",
269         fg="#1E89BF"
270     )
271
272     tipo2_label_text.grid(
273         row=current_row, column=1, columnspan=1,
274         sticky='e',
275     )
276
277
278     current_row += 1
279
280     #Función que permite establecer cual será el formato de inicio y cual será el
281     formato fin
282     def changeType():
283         auxLabel=tk.Label(
284             image=tipo1_label['image'],
285             text=tipo1_label_text['text']
286         )
287
288         if tipo2_label_text['text']=='MoodleXML':
289             tipo1_label.grid(
290                 padx=(15,0)
291             )
292             tipo2_label.grid(
293                 padx=(0,25)
294             )
295         else:
296             tipo1_label.grid(
297                 padx=(25,0)
298             )
299             tipo2_label.grid(
300                 padx=(0,15)
301             )
302
303         tipo1_label.config(
304             image=tipo2_label['image']
305         )
306
307         tipo1_label_text.config(
308             text=tipo2_label_text['text']
309         )
310
311         tipo2_label.config(
312             image=auxLabel['image']
313         )
314
315         tipo2_label_text.config(
316             text=auxLabel['text']
317         )

```

```
318
319
320     # -----
321
322     change_button = tk.Button(
323         window,
324         image=cambio,
325         command=changeType,
326         bg="white"
327     )
328
329     change_button.grid(
330         row=current_row, column=1, padx=150, pady=15,
331         sticky='nsew',
332     )
333
334     # -----
335     current_row += 2
336
337     #Función que expande la ventana principal para poder permitir la ejecución
338     def run():
339
340         run_button.grid(
341             row=current_row, column=0, columnspan=4, padx=30, pady=10
342         )
343         run_button.config(
344             command=convertir,
345             bg="#4BBD4B", activebackground="#3FA63F",
346             fg="white"
347         )
348
349
350         run_message_label.grid(
351             row=current_row+1, column=1, columnspan=1, padx=(30, 30), pady=(0, 0),
352             sticky='nsew',
353         )
354
355         run_message_frame.grid(
356             row=current_row+2, column=1, columnspan=1, padx=30, pady=(0, 20),
357             sticky='nsew',
358         )
359
360
361         run_message_text['state']=tk.NORMAL
362         run_message_text.delete(1.0, tk.END)
363         run_message_text.config(
364             fg="gray"
365         )
366         run_message_text.insert(tk.INSERT, "Esperando...")
367         run_message_text['state']=tk.DISABLED
368         run_message_text.grid()
369         run_message_text['state']=tk.DISABLED
370
371
372     #Función que llama a la clase correspondiente para realizar la conversión
373     def convertir():
```

```

374
375     run_message_text['state']=tk.NORMAL
376
377     error_message = None
378     file_path = pathlib.Path(file_name)
379     if tipo1_label_text["text"] == "Canvas QTI/IMS":
380         qti_2moodle = qti.QTI2moodle(file_name, file_path.stem, file_path.parent.
            as_posix())
381         conv=qti_2moodle.m_conv()
382         if conv==-1:
383             error_message = f'Quiz creation failed'
384         elif conv==0:
385             error_message = f'No questions to convert'
386
387     else:
388         moodle_2qti = moodle.moodle2QTI(file_name, file_path.stem, file_path.
            parent.as_posix())
389         conv=moodle_2qti.m_conv()
390         if conv==-1:
391             error_message = f'Quiz creation failed'
392         elif conv==0:
393             error_message = f'No questions to convert'
394
395
396     if error_message:
397
398         run_message_text.delete(1.0, tk.END)
399         run_message_text.insert(tk.INSERT, error_message)
400         run_message_text['fg'] = 'red'
401     else:
402         run_message_text.delete(1.0, tk.END)
403         run_message_text.insert(tk.INSERT, f'Created quiz "{file_path.parent.
            as_posix()}/ExportMOODLE_XML.zip"')
404         run_message_text['fg'] = 'green'
405
406
407     run_message_text['state']=tk.DISABLED
408
409
410     window.iconphoto(True, icono)
411     window.mainloop()
412
413
414 if __name__ == "__main__":
415     main()

```

moodle2QTI.py

```
1  #!/usr/bin/env python
2
3  import errno
4  import xml.etree.ElementTree as ET
5  import urllib.parse
6  import re
7  import os, shutil # make dir and copy files
8  import base64
9  import sys, getopt
10 # lxml is only used for cleaning the CDATA html. Con: not part of Python default
    install
11 import lxml
12 from lxml.html import fromstring, tostring, clean
13 import uuid
14
15
16
17 answerindex = 999 #Variable global para asignar números ID a las respuestas
18
19
20 """
21 Clase para leer/convertir aquellos ficheros
22 que tienen una estructura en formato MoodleXML, y través de las equivalencias
23 crea el fichero en formato QTI/IMS.
24 """
25 class moodle2QTI():
26
27
28     """
29     Inicializar los atributos de los objetos pertenecientes a la clase moodle2QTI
30     """
31     def __init__(self, file_input, out, path_out):
32
33         self.file_input = file_input #Fichero de entrada
34         self.out = out #Nombre del fichero de salida
35         self.path_out = path_out #Ruta que almacenará el fichero de salida
36
37
38
39     """
40     Función que lee el fichero de entrada en formato MoodleXML, y a través de las
41         equivalencias
42     y las siguientes funciones de la clase, crea el fichero de salida en formato QTI/
43         IMS.
44     """
45     def readMoodle(self, inputfile, outputfolder):
46
47         #Obtenemos el arbol xml de MoodleXML
48         try:
49             tree = ET.parse(inputfile)
50         except Exception as e:
51             error_message = f'An error occurred in reading the quiz file. Technical
52                             details:\n\t{e}'
53             print(error_message)
```

```

51
52
53     root = tree.getroot()
54
55
56     #Obtenemos todas las preguntas del cuestionario
57     preguntas = root.findall("question")
58
59
60     if len(preguntas)==0:
61         print("No hay prgeuntas para convertir")
62         return 0
63
64     #Variables de control
65     have_cat=0
66     name=""
67
68     #Creamos el arbol correspondiente al fichero de salida en formato QTI
69     ET.register_namespace('', "http://www.imsglobal.org/xsd/ims_qtiasiv1p2") #
        no ns0 namespaces here
70
71     questestinterop = ET.fromstring(str(
72         '<questestinterop xmlns="http://www.imsglobal.org/xsd/ims_qtiasiv1p2"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:
            schemaLocation="http://www.imsglobal.org/xsd/ims_qtiasiv1p2 http://www
            .imsglobal.org/xsd/ims_qtiasiv1p2p1.xsd"></questestinterop>'))
73
74
75     u = uuid.uuid1() #Identificadores unicos
76     ident = u.hex
77
78     #Creamos los metadatos por defecto para cualquier cuestionario
79     assessment = ET.SubElement(questestinterop,"assessment", ident=str(ident),
        title="")
80
81     qtimetadadata = ET.SubElement(assessment, "qtimetadadata")
82     qtimetadadatafield = ET.SubElement(qtimetadadata, "qtimetadadatafield")
83     fieldlabel = ET.SubElement(qtimetadadatafield,"fieldlabel")
84     fieldlabel.text = "cc_maxattempts"
85     fieldentry = ET.SubElement(qtimetadadatafield,"fieldentry")
86     fieldentry.text = "1"
87
88     #Creamos la seccion que almacenará las preguntas
89     section = ET.SubElement(assessment,"section", ident="root_section")
90
91     # Enumeracion de preguntas a convertir
92     convertix = 0
93
94     #Recorremos los items(preguntas) del cuestionario
95     for q in preguntas:
96
97         try:
98             if q.attrib['type'] == "category":
99                 #En el caso que el fichero MoodleXML tenga categoria, podremos
100                 extraer un titulo para el cuestionario
101                 text = q.find('category/text').text
102                 name = self.foundName(text)
103                 assessment.set('title',name)
104                 have_cat=1 #Indicamos que se ha encontrado la categoria

```

```
101         else:
102             if have_cat==0:
103                 #En el caso que no contenga categoria, usaremos como titulo
104                 el nombre del fichero de entrada
105                 text = self.file_input
106                 name = str(ident)
107                 assessment.set('title',text)
108
109             if q.attrib['type'] != "category": #Corresponde con aquellas
110                 preguntas reales que no son categoria
111
112                 #Obtenemos valores para los metadatos
113                 qtype = q.attrib['type']
114                 u = uuid.uuid1()
115                 convertix = convertix + 1
116                 #Obtenemos el prefijo correspondiente para el nombre del fichero
117                 de salida
118                 prefix = self.getprefix(qtype)
119                 q.set('convertix', prefix + str(convertix))
120                 item = ET.SubElement(section,"item",ident=q.get('convertix'),
121                                     title="")
122                 title = q.find('name/text').text
123                 item.set('title', title)
124
125                 #Procedemos a convertir la pregunta dependiendo de su tipo
126
127                 if qtype == 'shortanswer':
128                     q.set('type',"short_answer_question")
129
130                     self.produceSAQuestion(q, item, u)
131                     #-----
132
133                 if qtype == 'multichoice':
134                     q.set('type',"multiple_choice_question")
135
136                     if q.find('single').text == "false":
137                         q.set('type',"multiple_answers_question")
138
139                     self.produceMCQuestion(q, item, u)
140                     #-----
141
142                 if qtype == 'truefalse':
143                     q.set('type',"true_false_question")
144                     self.produceTFQuestion(q,item,u)
145                     #-----
146
147                 if qtype == 'matching':
148                     q.set('type',"matching_question")
149                     self.produceMATCHQuestion(q,item,u)
150                     #-----
151
152                 if qtype == 'numerical':
153                     q.set('type',"numerical_question")
154                     self.produceNUMQuestion(q,item,u)
155
156         except e:
```

```

153         print("Error in produce: ", qtype)
154         print("Details: ", e)
155         return -1
156
157
158         #Creamos el fichero de salida
159         outputfolder = outputfolder+'/' + name
160         out = prefix + name + '.xml'
161         filename = outputfolder + '/' + out
162
163         try:
164             if not os.path.exists(outputfolder):
165                 os.makedirs(outputfolder)
166
167         except OSError as e:
168             if e.errno != errno.EEXIST:
169                 raise
170
171         self.writequestionfile(questestinterop, filename)
172
173         return 1, outputfolder
174
175
176     """
177     Función que busca el nombre del cuestionario
178     """
179     def foundName(self, text):
180         if 'para' in text:
181             posicion = text.index('para') + 4
182         elif 'en' in text:
183             posicion = text.index('en') + 2
184         else:
185             posicion = 0
186
187         text = text[posicion + 1:len(text)]
188         text = re.sub('\.', '', text)
189
190         print(posicion)
191
192         if posicion != 0 and '/' in text:
193             posicion = text.index('/')
194             text = text[posicion + 1:len(text)]
195
196         if posicion == 0: #EN el caso de que no encuentre un patron y podamos sacar el
197             nombre, le añadimos un identificador unico
198             u = uuid.uuid1()
199             ident = u.hex
200             text = str(ident)
201
202         return text
203
204     """
205     Función que obtiene un prefijo según el tipo de pregunta
206     """
207     def getprefix(self, qtype):

```

```
208     prefix = 'AAA_ERROR'
209     if qtype == 'shortanswer':
210         prefix = 'MSHORT_'
211     if qtype == 'numerical':
212         prefix = 'NUMERIC_'
213     if qtype == 'matching':
214         prefix = 'MATCH_'
215     if qtype == 'multichoice':
216         prefix = "MULTI_"
217     if qtype == 'truefalse':
218         prefix = "TRUEFALSE_"
219     return prefix
220
221
222     """
223     Función que crea los tres bloques que contiene todos los items de QTI
224     """
225     def createBlocksItem(self,item):
226         itemmetadata = ET.SubElement(item, "itemmetadata")
227         presentation = ET.SubElement(item, "presentation")
228         resprocessing = ET.SubElement(item, "resprocessing")
229
230         return itemmetadata,presentation,resprocessing
231
232
233     """
234     Función que crea los metadatos correspondientes a un item
235     """
236     def createItemmetadata(self,tag,itemmetadata,u):
237         possible_answer = tag.findall('answer')
238
239         #generar id de respuestas
240         original_answer_ids = self.generatedIds(possible_answer)
241
242         question_type = tag.attrib['type']
243         point_possible = tag.find('defaultgrade').text
244
245         assessment_question_identfierref = u.hex
246
247         v_fieldlabel = ['question_type','points_possible','original_answer_ids','
248                        assessment_question_identfierref']
249         v_fielentry = [question_type,point_possible,original_answer_ids,
250                       assessment_question_identfierref]
251
252         qtimetadadata = ET.SubElement(itemmetadata, "qtimetadadata")
253
254         for i in range(len(v_fieldlabel)):
255             qtimetadadatafield = ET.SubElement(qtimetadadata, "qtimetadadatafield")
256
257             fieldlabel = ET.SubElement(qtimetadadatafield,"fieldlabel")
258             fieldlabel.text = v_fieldlabel[i]
259
260             fielentry = ET.SubElement(qtimetadadatafield,"fielentry")
261             fielentry.text = v_fielentry[i]
262
263         return possible_answer,original_answer_ids
```

```

262
263
264     """
265     Función que crea identificadores para las respuestas
266     """
267     def generatedIds(self, possible_answer):
268
269         global answerindex
270
271         original_answer_ids = ""
272         for i in range(len(possible_answer)):
273             original_answer_ids += str(answerindex)
274             if i==len(possible_answer)-1:
275                 original_answer_ids += ""
276             else:
277                 original_answer_ids += ","
278
279             answerindex += 1
280
281
282         return original_answer_ids
283
284
285     """
286     Función que crea la etiqueta <material> que almacena las cadenas de texto
287     """
288     def makeMaterial(self, padre, text, texttype="text/html"):
289         material = ET.SubElement(padre, "material")
290
291         mattext = ET.SubElement(material, "mattext", texttype=texttype)
292         mattext.text = text
293
294
295     """
296     Función que crea la etiqueta <outcomes>
297     """
298     def makeOutcomes(self, padre):
299         outcomes = ET.SubElement(padre, "outcomes")
300         decvar = ET.SubElement(outcomes, "decvar", maxvalue="100", minvalue="0",
301                                varname="SCORE", vartype="Decimal")
302
303
304     """
305     Función que crea una pregunta Múltiple elección o
306     Múltiple respuesta en formato QTI/IMS.
307     """
308     def produceMCQuestion(self, tag, item, u):
309
310         #Creamos los bloques de la pregunta
311         itemmetadata, presentation, resprocessing = self.createBlocksItem(item)
312
313         #### ITEMMETADATA -> Creamos los metadatos de la pregunta
314         possible_answer, original_answer_ids=self.createItemmetadata(tag, itemmetadata,
315                                u)
316

```

```

316     #### PRESENTATION -> Creamos la visualización de la pregunta
317     m_qtext = tag.find('questiontext/text').text
318     self.makeMaterial(presentation,m_qtext,"text/html")#Contenido de la pregunta
319
320     #atributos/elementos correspondientes a la pregunta
321     identRes = "response1"
322     response_lid = ET.SubElement(presentation,"response_lid", ident=identRes,
323                                 rcardinality="")
324     if tag.find('single').text == 'true':
325         response_lid.set('rcardinality', 'Single')
326     else:
327         response_lid.set('rcardinality', 'Multiple')
328
329     render_choice = ET.SubElement(response_lid, "render_choice")
330
331     #Creamos las respuestas
332     original_answer_ids = original_answer_ids.split(',')
333     i=0
334     correctlist = []
335     badlist = []
336     scores = []
337     for ans in possible_answer:
338
339         atext = ans.find('text').text
340         response_label = ET.SubElement(render_choice,"response_label",ident=
341                                     original_answer_ids[i])
342         self.makeMaterial(response_label,atext,"text/html")
343
344         if int(float(ans.attrib['fraction'])) > 10: #Aquellas que tengan menos
345             puntaje no se consideran respuestas correctas
346             correctlist.append(original_answer_ids[i])
347             scores.append(int(float(ans.attrib['fraction'])))
348         else:
349             badlist.append(original_answer_ids[i])
350         i+=1
351
352     ####
353
354     #### RESPROCESSING -> Creamos el procesamiento de la pregunta
355     self.makeOutcomes(resprocessing)
356
357     respcondition = ET.SubElement(resprocessing, "respcondition", {'continue':"No
358                                "})
359     conditionvar = ET.SubElement(respcondition,"conditionvar")
360
361     if correctlist==1: # Multiple opción con una sola respuesta
362         varequal = ET.SubElement(conditionvar,"varequal",respident=identRes)
363         varequal.text = correctlist[0]
364
365     else: #Multiple opción con multiple respuesta
366         eti_and = ET.SubElement(conditionvar, "and")
367         for correctId in correctlist:
368             varequal = ET.SubElement(eti_and,"varequal",respident=identRes)
369             varequal.text = correctId
370
371         for badId in badlist:

```

```

368         eti_not = ET.SubElement(eti_and, "not")
369         varequal = ET.SubElement(eti_not, "varequal", respident=identRes)
370         varequal.text = badId
371
372
373         setvar = ET.SubElement(respcondition, "setvar", action="Set", varname="SCORE")
374         setvar.text = "100"
375
376
377     """
378     Función que crea una pregunta Verdadero/Falso en formato QTI/IMS.
379     """
380     def produceTFQuestion(self, tag, item, u):
381
382         #Creamos los bloques de la pregunta
383         itemmetadata, presentation, resprocessing = self.createBlocksItem(item)
384
385         #### ITEMMETADATA -> Creamos los metadatos de la pregunta
386         possible_answer, original_answer_ids = self.createItemmetadata(tag, itemmetadata,
387                                 u)
388
389         #### PRESENTATION -> Creamos la visualización de la pregunta
390         m_qtext = tag.find('questiontext/text').text #Contenido de la pregunta
391         self.makeMaterial(presentation, m_qtext, "text/html")
392
393         #atributos/elementos correspondientes a la pregunta
394         identRes = "response1"
395         response_lid = ET.SubElement(presentation, "response_lid", ident=identRes,
396                                     rcardinality="")
397         response_lid.set('rcardinality', 'Single')
398         render_choice = ET.SubElement(response_lid, "render_choice")
399
400         #Creamos las respuestas
401         original_answer_ids = original_answer_ids.split(',')
402         i=0
403
404         for ans in possible_answer:
405
406             atext = ans.find('text').text
407             if atext=="true":
408                 atext="Verdadero"
409             elif atext=="false":
410                 atext="Falso"
411
412             response_label = ET.SubElement(render_choice, "response_label", ident=
413                                     original_answer_ids[i])
414             self.makeMaterial(response_label, atext, "text/html")
415
416             if int(float(ans.attrib['fraction'])) > 10: #Aquellas que tengan menos
417                 puntaje no se consideran respuestas correctas
418                 original_answer_ids.append(original_answer_ids[i])
419
420             itemfeedback = ET.SubElement(item, "itemfeedback", ident=
421                                     original_answer_ids[i]+"_fb")
422             flow_mat = ET.SubElement(itemfeedback, "flow_mat")
423             fd = ans.find('feedback/text').text

```

```

419         self.makeMaterial(flow_mat,fd,"text/html")
420
421         i+=1
422     ###
423
424     #### RESPROCESSING -> Creamos el procesamiento de la pregunta
425
426     self.makeOutcomes(resprocessing)
427
428     for i in range(len(possible_answer)+1):
429
430         if i < len(possible_answer):
431             respcondition = ET.SubElement(resprocessing, "respcondition", {'
432                 continue':'Yes'})
433         else:
434             respcondition = ET.SubElement(resprocessing, "respcondition", {'
435                 continue':'No'})
436
437         conditionvar = ET.SubElement(respcondition,"conditionvar")
438         varequal = ET.SubElement(conditionvar,"varequal",respid=identRes)
439         varequal.text = original_answer_ids[i]
440
441         if i < len(possible_answer):
442             displayfeedback = ET.SubElement(respcondition,"displayfeedback",
443                 feedbacktype="Response", linkrefid=original_answer_ids[i]+"_fb")
444         else:
445             setvar = ET.SubElement(respcondition,"setvar",action="Set",varname="
446                 SCORE")
447             setvar.text = "100"
448
449
450     """
451     Función que crea una pregunta Respuesta corta en formato QTI/IMS.
452     """
453     def produceSAQuestion(self,tag, item, u):
454
455         #Creamos los bloques de la pregunta
456         itemmetadata,presentation,resprocessing = self.createBlocksItem(item)
457
458         #### ITEMMETADATA -> Creamos los metadatos de la pregunta
459         possible_answer,original_answer_ids=self.createItemmetadata(tag,itemmetadata,
460             u)
461
462         #### PRESENTATION -> Creamos la visualización de la pregunta
463         m_qtext = tag.find('questiontext/text').text #Contenido de la pregunta
464         self.makeMaterial(presentation,m_qtext,"text/html")
465
466         #atributos/elementos correspondientes a la pregunta
467         identRes = "response1"
468         response_str = ET.SubElement(presentation,"response_str", ident=identRes,
469             rcardinality="Single")
470
471         render_fib = ET.SubElement(response_str, "render_fib")
472         response_label = ET.SubElement(render_fib,"response_label", ident="answer1",
473             rshuffle="No")

```

```

468     ###
469
470     #### RESPROCESSING -> Creamos el procesamiento de la pregunta
471     original_answer_ids = original_answer_ids.split(',')
472
473     self.makeOutcomes(resprocessing)
474
475     respcondition = ET.SubElement(resprocessing, "respcondition", {'continue': 'No
476         '})
477     conditionvar = ET.SubElement(respcondition, "conditionvar")
478
479     #Creamos las respuestas
480     for ans in possible_answer:
481         varequal = ET.SubElement(conditionvar, "varequal", respident="response1")
482         atext = ans.find('text').text
483         varequal.text = atext
484
485     setvar = ET.SubElement(respcondition, "setvar", action="Set", varname="SCORE")
486     setvar.text = "100"
487
488     """
489     Función que crea una pregunta Respuesta numérica en formato QTI/IMS.
490     """
491     def produceNUMQuestion(self, tag, item, u):
492
493         #Creamos los bloques de la pregunta
494         itemmetadata, presentation, resprocessing = self.createBlocksItem(item)
495
496         #### ITEMMETADATA -> Creamos los metadatos de la pregunta
497         possible_answer, original_answer_ids = self.createItemmetadata(tag, itemmetadata,
498             u)
499
500         #### PRESENTATION -> Creamos la visualización de la pregunta
501         m_qtext = tag.find('questiontext/text').text #Contenido de la pregunta
502         self.makeMaterial(presentation, m_qtext)
503
504         #atributos/elementos correspondientes a la pregunta
505         identRes = "response1"
506         response_str = ET.SubElement(presentation, "response_str", ident=identRes,
507             rcardinality="Single")
508         render_fib = ET.SubElement(response_str, "render_fib", fibtype="Decimal")
509         response_label = ET.SubElement(render_fib, "response_label", ident="answer1")
510         ###
511
512         #### RESPROCESSING -> Creamos el procesamiento de la pregunta
513         self.makeOutcomes(resprocessing)
514
515         for ans in possible_answer: #Creamos las respuestas
516             tolerancia = ans.find('tolerance').text
517             valor = ans.find('text').text
518             score = ans.attrib['fraction']
519             respcondition = ET.SubElement(resprocessing, "respcondition", {'continue':
520                 'No'})
521             conditionvar = ET.SubElement(respcondition, "conditionvar")
522             et_or = ET.SubElement(conditionvar, "or")

```

```

520
521         varequal = ET.SubElement(et_or, "varequal", respident=identRes) #valor
522             exacto
523         varequal.text = valor
524
525         et_and = ET.SubElement(et_or, "and") #Error aceptado
526         vargte = ET.SubElement(et_and, "vargte", respident=identRes)
527         vargte.text = str(round(float(valor) - float(tolerancia),5))
528         varlte = ET.SubElement(et_and, "varlte", respident=identRes)
529         varlte.text = str(round(float(valor) + float(tolerancia),5))
530
531         setvar = ET.SubElement(respcondition, "setvar", action="Set", varname="SCORE")
532
533         setvar.text = str(score)
534
535     """
536     Función que crea una pregunta Empajeramiento en formato QTI/IMS.
537     """
538     def produceMATCHQuestion(self, tag, item, u):
539
540         #Creamos los bloques de la pregunta
541         itemmetadata, presentation, resprocessing = self.createBlocksItem(item)
542
543         #### ITEMMETADATA -> Creamos los metadatos de la pregunta
544
545         subquestion = tag.findall('subquestion')
546         possible_answer = tag.findall('subquestion/answer')
547         #en un principio se debería mantener la relacion/orden subquestion-answer que
548             viene de moodle
549
550         subquestion_ids = self.generatedIds(subquestion) #generar id de subquestions
551         original_answer_ids = self.generatedIds(possible_answer) #generar id de
552             respuestas
553
554         question_type = tag.attrib['type']
555         point_possible = tag.find('defaultgrade').text
556
557         assessment_question_identfierref = u.hex
558
559         v_fieldlabel = ['question_type', 'points_possible', 'original_answer_ids', 'assessment_question_identfierref']
560         v_fielentry = [question_type, point_possible, original_answer_ids, assessment_question_identfierref]
561         qtimetadadata = ET.SubElement(itemmetadata, "qtimetadadata")
562         for i in range(len(v_fieldlabel)):
563             qtimetadadatafield = ET.SubElement(qtimetadadata, "qtimetadadatafield")
564
565             fieldlabel = ET.SubElement(qtimetadadatafield, "fieldlabel")
566             fieldlabel.text = v_fieldlabel[i]
567
568             fielfentry = ET.SubElement(qtimetadadatafield, "fielfentry")
569             fielfentry.text = v_fielentry[i]
570
571     """

```

```

570     #### PRESENTATION -> Creamos la visualización de la pregunta
571     m_qtext = tag.find('questiontext/text').text #Contenido de la pregunta
572     self.makeMaterial(presentation,m_qtext,"text/html")
573
574     #Creamos las respuestas
575     original_answer_ids = original_answer_ids.split(',')
576     subquestion_ids = subquestion_ids.split(',')
577     i=0
578
579     for sub in subquestion: #Columna izquierda
580         identRes = "response_"+original_answer_ids[i]
581         response_lid = ET.SubElement(presentation,"response_lid", ident=identRes)
582
583         subText = sub.find('text').text
584         self.makeMaterial(response_lid,subText,"text/html")
585
586         render_choice = ET.SubElement(response_lid, "render_choice")
587
588         j=0
589         for ans in possible_answer: #Columna derecha
590             #su atributo shuffle indica si hay que reordenar aleatoriamente los í
591             tems
592             response_label = ET.SubElement(render_choice,"response_label", ident=
593                 subquestion_ids[j])
594             ansText = ans.find('text').text
595             self.makeMaterial(response_label,ansText,"text/html")
596             j+=1
597
598         i+=1
599     ####
600
601     #### RESPROCESSING -> Creamos el procesamiento de la pregunta
602     self.makeOutcomes(resprocessing)
603     subScores = float(100/len(possible_answer))
604     i=0
605     for ans in possible_answer:
606         respcondition = ET.SubElement(resprocessing, "respcondition")
607         conditionvar = ET.SubElement(respcondition,"conditionvar")
608         varequal = ET.SubElement(conditionvar,"varequal", respident="response_"+
609             original_answer_ids[i])
610         varequal.text = str(subquestion_ids[i])
611         setvar = ET.SubElement(respcondition,"setvar",action="Add",varname="SCORE
612             ")
613         setvar.text = str(subScores)
614         i+=1
615
616     """
617     Función que añade al fichero de salida las marcas de un fichero XML y la
618     estructura del arbol.
619     """
620     def writequestionfile(self,questestinterop, filename):
621         f = open(filename, "w")
622         f.write('<?xml version="1.0" encoding="utf-8" standalone="yes"?>')
623         try:
624             f.write(ET.tostring(questestinterop, encoding='utf-8', method='xml')).

```

```
        decode('utf-8'))
621     except Exception as e:
622         error_message = f'Error al añadir el arbol al fichero. Technical details
            :\n\t{e}'
623         print(error_message)
624         print("\n")
625         return -1
626
627     print(f"Wrote {filename}")
628
629
630     """
631     Función que convierte el fichero MoodleXML.
632     """
633     def convertMoodle(self, inputfile, outputfolder):
634         val, outputfolder = self.readMoodle(inputfile, outputfolder)
635         shutil.make_archive(outputfolder, 'zip', outputfolder) #El formato QTI se
            debe subir a canvas comprimido en .zip
636         return val
637
638
639     """
640     Función principal, con la que la interfaz se comunicará
641     """
642     def m_conv(self):
643
644         inputfile = self.file_input
645         outputfolder = self.path_out + '/ExportQTI'
646
647         print(f'Input file is "{inputfile}"')
648         print(f'Output FOLDER is "{outputfolder}"')
649
650         try:
651             val = self.convertMoodle(inputfile, outputfolder)
652             if val == 0:
653                 return 0
654             if val == -1:
655                 return -1
656         except:
657             print("An exception occurred in convertMoodle")
658             return -1
659         else:
660             return 1
```

QTI2moodle.py

```

1  #!/usr/bin/env python
2
3  import errno
4  import xml.etree.ElementTree as ET
5  import urllib.parse
6  import re
7  import os, shutil # make dir and copy files
8  import base64
9  import sys, getopt
10 # lxml is only used for cleaning the CDATA html. Con: not part of Python default
    install
11 import lxml
12 from lxml.html import fromstring, tostring, clean
13 from lxml import objectify
14
15 """
16 Clase para leer/convertir aquellos ficheros
17 que tienen una estructura en formato QTI/IMS, y través de las equivalencias
18 crea el fichero en formato MoodleXML.
19 """
20 class QTI2moodle():
21
22
23     """
24     Inicializar los atributos de los objetos pertenecientes a la clase QTI2moodle
25     """
26     def __init__(self, file_input, out, path_out):
27
28         self.file_input = file_input #Fichero de entrada
29         self.out = out #Nombre del fichero de salida
30         self.path_out = path_out #Ruta que almacenará el fichero de salida
31
32
33
34     """
35     Función que lee el fichero de entrada en formato QTI, y a través de las
36 equivalencias y las demás funciones de la clase, crea el fichero
37 de salida en formato moodleXML.
38     """
39     def readQTI(self, inputfile, outputfolder):
40
41
42         filename = inputfile
43         error_message = None
44
45         #Obtenemos el arbol xml de QTI
46         try:
47             tree = objectify.parse(filename)
48         except Exception as e:
49             error_message = f'An error occurred in reading the quiz file. Technical
                    details:\n\t{e}'
50             print(error_message)
51
52         root = tree.getroot()

```

```
53
54
55     #Desglosamos los elementos del arbol XML de QTI/IMS
56     titleArc=root.assessment.attrib["title"]
57
58     items = tree.findall('://{http://www.imsglobal.org/xsd/ims_qtiasiv1p2}item')
59
60
61     if len(items)==0:
62         print("No hay prgeuntas para convertir")
63         return 0
64
65
66     #Creamos el arbol correspondiente al fichero de salida en formato moodleXML
67     quiz = ET.Element("quiz")
68
69     only=0
70
71     #Recorremos los items(preguntas) del cuestionario
72     for item in items:
73
74         ##### Cada item esta dividido como minimo en tres partes
75         itemdata = item.itemmetadata.qtimetadata.getchildren()
76
77         itempresentation = item.presentation.getchildren() #tiene dos hijos (
78             material=0, response_lid=1)
79
79         itemresprocessing = item.resprocessing.getchildren()
80         #####
81
82
83         qtidata = []
84         #Obtenemos los valores de los metadatos
85         for data in itemdata:
86             qtidata.append(data.fieldentry)
87
88         #Estos son los unicos valores de [qtidata] que contienen equivalencia en
89         el formato moodleXML
89         qtype = qtidata[0]
90         point = qtidata[1]
91
92
93         #####
94
95         question=itempresentation[0].mattext #Contenido de la pregunta
96         pos_aw = []
97         correctChoiceID = []
98
99
100        #Realizamos un pre-procesamineto para obtener ciertos elementos antes de
101        convertir cada pregunta
101        if qtype == 'short_answer_question' or qtype == 'multiple_choice_question'
102            or qtype == 'multiple_answers_question' or qtype == '
            true_false_question':
            for i in range(1,len(itemresprocessing)): #Recorremos los
                procedimientos de la pregunta
```

```

103         if itemresprocessing[i].attrib['continue']=="No": #Y solo nos
               interesa aquel 'para' el procesamiento, ya que signifca que
               es la respuesta correcta
104
105         score = str(itemresprocessing[i].setvar).strip() #Obtenemos
               la puntuación de la respuesta
106
107         if len(itemresprocessing[i].conditionvar.getchildren()) == 1
               and qtype != 'multiple_answers_question':
108             #Guardamos el ID de la respuesta correcta (cuando solo
               hay UNA correcta)
109             haveAnd = itemresprocessing[i].conditionvar.getchildren()
110             if str(haveAnd[0].attrib) != "{}":
111                 correctChoiceID.append( str(itemresprocessing[i].
               conditionvar.varequal).strip())
112             else:
113                 correctChoiceID.append( str(haveAnd[0].varequal).
               strip())
114
115         elif len(itemresprocessing[i].conditionvar.getchildren()) > 1
               and qtype != 'multiple_answers_question':
116             #En el caso de que haya más de una, guardamos las
               posibles respuestas
117             for aw in itemresprocessing[i].conditionvar.getchildren()
               :
118                 pos_aw.append(aw)
119             else:
120                 #En el caso de que la pregunta sea multi respuesta,
               guardamos todas las respuestas correctas
121                 answ=itemresprocessing[i].conditionvar.getchildren()
122                 children=answ[0].getchildren()
123                 for element in children:
124                     if str(element.attrib) != "{}":
125                         correctChoiceID.append(element)
126
127
128
129         ##### Para una mejor subida del fichero a moodle, añadimos una
               categoria
130         if only==0:
131             quiz = self.makeCategoria(quiz,titleArc)
132             only = only+1
133         #####
134
135
136         prefix = self.getprefix(qtype) #Obtenemos el prefijo correspondiente para
               el nombre del fichero de salida
137
138         question= self.fixHtmlText(question) #Pasamos el valor de la pregunta a
               formato html
139
140         #Procedemos a convertir la pregunta dependiendo de su tipo
141         try:
142             if qtype == 'short_answer_question':
143                 questionMask = ET.SubElement(quiz,"question", type= "shortanswer
               ")

```

```

144         self.produceSAQuestion(questionMask,item,question ,pos_aw, point,
145                                 score)
146
147         if qtype == 'multiple_choice_question' or qtype=='
148             multiple_answers_question':
149
150             questionMask = ET.SubElement(quiz,"question", type= "multichoice
151                                     ")
152             choices = itempresentation[1].render_choice.getchildren()
153             self.produceMCQuestion(questionMask,item,question ,choices,
154                                     correctChoiceID, point,score)
155
156         # -----
157
158         if qtype == 'true_false_question':
159             choices = itempresentation[1].render_choice.getchildren()
160             questionMask = ET.SubElement(quiz,"question", type= "truefalse")
161             self.produceTFQuestion(questionMask,item,question ,choices,
162                                     correctChoiceID, point,score)
163
164         # -----
165         #
166         if qtype == 'matching_question':
167             questionMask = ET.SubElement(quiz,"question", type= "matching")
168
169             subquestion = []
170             choices = []
171
172             for i in range(1,len(itempresentation)):#Columna izquierda
173                 subquestion.append(itempresentation[i].material.mattext)
174
175             for aw in itempresentation[1].render_choice.getchildren():#
176                 Columna derecha
177                 choices.append(aw.material.mattext)
178
179             self.produceMATCHQuestion(questionMask,item,question ,subquestion
180                                     , choices, point)
181
182         # -----
183
184         if qtype == 'numerical_question':
185             scores =[]
186             choices = []
187             tolerancias = []
188             tipoUnidades = itempresentation[1].render_fib.attrib['fibtype']
189
190             for i in range(1,len(itemresprocessing)):#Empezamos desde el hijo
191                 1, ya que el 0 es outcomes
192                 info = itemresprocessing[i].getchildren()
193
194                 sc = info[1] #Score
195                 naw = info[0].getchildren() #Condiciones
196
197                 #Obtenemos el valor exacto de la respuesta y el error
198                 permitido
199                 if len(naw)==1:
200                     valores=naw[0].getchildren()

```

```

191         exacto = valores[0]
192         dif = valores[1]
193         error = round(float(dif.varlte)-float(exacto),5)
194     else:
195         exacto=sum(naw)/2
196         error = round(float(naw[1])-float(exacto),5)
197
198         scores.append(sc)
199         choices.append(exacto)
200         tolerancias.append(error)
201
202         questionMask = ET.SubElement(quiz,"question", type= "numerical")
203         self.produceNUMQuestion(questionMask,item,question ,choices,
204                                 tolerancias, point,scores,tipounidades)
205
206     except e:
207         print("Error in produce: ", qtype)
208         print("Details: ", e)
209         return -1
210
211
212     #Creamos el fichero de salida
213     self.out =prefix+'_'+self.out+'.xml'
214     filename = outputfolder+'/' + self.out
215     self.writequestionfile(quiz, filename)
216
217
218
219     """
220     Función que crea la categoría usando el título del cuestionarios, esto
221     puede ser útil al importar o exportar el cuestionario, ya que permite
222     clasificar y organizar las preguntas en categorías predefinidas.
223     """
224     def makeCategoria(self,quiz,title):
225
226         question = ET.SubElement(quiz, "question", type="category")
227
228         category = ET.SubElement(question, "category")
229         categoryText = ET.SubElement(category, "text")
230         texto = "$course$/top/Valor por defecto para " + str(title)
231         categoryText.text = texto
232
233         info = ET.SubElement(question, "info", format = "moodle_auto_format")
234         infoText = ET.SubElement ( info, "text")
235         texto = "Categoría por defecto para preguntas compartidas en el contexto [ "+
236                 str(title) + " ]"
237         infoText.text = texto
238
239         idnumber = ET.SubElement(question, "idnumber")
240
241         return quiz
242
243     """
244     Función que obtiene un prefijo según el tipo de pregunta
245     """

```

```
245     def getprefix(self, qtype):
246         prefix = 'AAA_ERROR'
247
248         if qtype == 'short_answer_question':
249             prefix = 'MSHORT_'
250
251         if qtype == 'numerical_question':
252             prefix = 'NUMERIC_'
253
254         if qtype == 'matching_question':
255             prefix = 'MARCHING_'
256
257         if qtype == 'multiple_choice_question' or qtype == 'multiple_answers_question':
258             prefix = "MULTI_"
259
260         if qtype == 'true_false_question':
261             prefix = "TR_FL_"
262
263
264         return prefix
265
266
267     """
268     Función que añade la etiqueta texto a otra etiqueta padre
269     """
270     def addText(self, padre, text):
271
272         etiquetaText = ET.SubElement(padre, "text")
273         etiquetaText.text = str(text)
274
275
276     """
277     Función que añade algunas marcas opcionales de moodleXML con valores por defecto
278     """
279     def defaultMarks(self, question, generalfb_val="", defaultgrade_val="1.0000000",
280                     penalty_val="0.0000000", hidden_val="0"):
281
282         gf = ET.SubElement(question, "generalfeedback", format="html")
283         self.addText(gf, generalfb_val)
284
285         defaultgrade = ET.SubElement(question, "defaultgrade")
286         defaultgrade.text = defaultgrade_val
287
288         penalty = ET.SubElement(question, "penalty")
289         penalty.text = penalty_val
290
291         hidden = ET.SubElement(question, "hidden")
292         hidden.text = hidden_val
293
294
295     """
296     Función que añade las marcas de retroalimentación
297     """
298     def feedbackMarks(self, question, textcorrect="", textpartially="", textincorrect="
```



```

    "):
299
300     correctfeedback = ET.SubElement(question, "correctfeedback", format="html")
301     self.addText(correctfeedback, textcorrect)
302
303
304     partiallycorrectfeedback = ET.SubElement(question, "partiallycorrectfeedback"
        , format="html")
305     self.addText(partiallycorrectfeedback, textpartially)
306
307     incorrectfeedback = ET.SubElement(question, "incorrectfeedback", format="html
        ")
308     self.addText(incorrectfeedback, textincorrect)
309
310
311
312     """
313     Función que crea una pregunta Verdadero/Falso en formato MoodleXML
314     """
315     def produceTFQuestion(self, question, tag, questionParse ,choices, correctChoiceID,
        point, score):
316
317
318         #Creamos el contenido de la pregunta
319         name = ET.SubElement(question, "name")
320         self.addText(name, str(tag.attrib["title"]))
321
322         questiontext = ET.SubElement(question, "questiontext", format="html")
323         self.addText(questiontext, str(questionParse))
324
325         #Creamos atributos/elementos correspondientes a la pregunta
326         generalfeedback = ""
327         defaultgrade = str(point)
328         penalty="1"
329         hidden="0"
330         self.defaultMarks(question, generalfeedback, defaultgrade, penalty, hidden)
331         itemfeed = []
332         for meta in tag.getchildren():
333             if str(meta.attrib) != "{}":
334                 itemfeed.append(meta.flow_mat.material.mattext)
335
336         if len(itemfeed)==1:
337             itemfeed.append(meta.flow_mat.material.mattext)
338
339         if len(itemfeed)==0:
340             itemfeed.append("")
341
342
343         #Creamos las respuestas
344         for choice in choices:
345             choiceID = str(choice.attrib['ident']).strip()
346             answer = ET.SubElement(question, "answer", fraction="", format="
                moodle_auto_format")
347
348             if str(choice.material.mattext) == "Verdadero":
349                 text = "true"

```

```
350         else:
351             text="false"
352
353         self.addText(answer, text)
354
355         awfeedback = ET.SubElement(answer, "feedback",format="html")
356         awfeedbacktext = ET.SubElement(awfeedback, "text")
357
358         print(itemfeed)
359         if len(itemfeed)>1:
360             awfeedbacktext.text = str(itemfeed[1])
361         else:
362             awfeedbacktext.text = str(itemfeed[0])
363
364         if choiceID == correctChoiceID[0]:
365             answer.set('fraction', score)
366             awfeedbacktext.text = str(itemfeed[0])
367         else:
368             answer.set('fraction', '0')
369
370
371
372
373     """
374     Función que crea una pregunta 'Múltiple elección'
375     o 'Múltiple respuesta' en formato MoodleXML.
376     """
377     def produceMCQuestion(self,question,tag, questionParse ,choices, correctChoiceID,
378         point,score):
379
380         #Creamos el contenido de la pregunta
381         name = ET.SubElement(question, "name")
382         self.addText(name, str(tag.attrib["title"]))
383
384         questiontext = ET.SubElement(question, "questiontext", format="html")
385         self.addText(questiontext, str(questionParse))
386
387         #Creamos atributos/elementos correspondientes a la pregunta
388         generalfeedback = ""
389         defaultgrade = str(point)
390         self.defaultMarks(question, generalfeedback, defaultgrade)
391
392         single = ET.SubElement(question, "single")
393         if tag.presentation.response_lid.attrib['rcardinality'] == 'Single':
394             single.text = "true"
395         else:
396             single.text = "false"
397
398         shuffleanswers = ET.SubElement(question, "shuffleanswers")
399         shuffleanswers.text = "true"
400
401         answernumbering = ET.SubElement(question, "answernumbering")
402         answernumbering.text = "abc"
403
404         ##### Feed Back ###
405         if len(correctChoiceID)>1:
```

```

405         correct = " <p>Respuesta correcta</p>"
406         partial = " <p>Respuesta parcialmente correcta.</p> "
407         incorrect = " <p>Respuesta incorrecta.</p> "
408         self.feedbackMarks(question, correct, partial, incorrect)
409     elif (len(tag.getChildren())>3):#Cuando la pregunta tiene etiquetas
retroalimentación (ademas de los tres bloques principales)
410         itemfeedback = tag.findall('.{http://www.msglobal.org/xsd/
            ims_qtiasiv1p2}itemfeedback')
411         for feed in itemfeedback:
412             if feed.attrib['ident'] == "correct_fb":
413                 correct = feed.flow_mat.material.mattext
414             elif (feed.attrib['ident'] == "general_incorrect_fb"):
415                 incorrect = feed.flow_mat.material.mattext
416
417         self.feedbackMarks(question, correct, "", incorrect)
418
419     else:#Cuando no hay etiquetas de retroalimentación
420         self.feedbackMarks(question)
421     #####
422
423
424     #Creamos las respuestas
425     scoreMax = score
426     for choice in choices:
427         choiceID = str(choice.attrib['ident']).strip()
428
429         answer = ET.SubElement(question, "answer", fraction="", format="html")
430         self.addText(answer, str(choice.material.mattext) )
431
432         awfeedback = ET.SubElement(answer, "feedback",format="html")
433         awfeedbacktext = ET.SubElement(awfeedback, "text")
434         awfeedbacktext.text = ""
435
436         if len(correctChoiceID)>1: #MultiRespuesta
437             score = str(float(scoreMax)/len(correctChoiceID))
438
439             for correct in correctChoiceID:
440
441                 if choiceID == str(correct):
442                     answer.set('fraction', score)
443                     awfeedbacktext.text = ""
444                     break
445                 else:
446                     answer.set('fraction', '0')
447         else:
448             if choiceID == correctChoiceID[0]:
449                 answer.set('fraction', score)
450                 awfeedbacktext.text = ""
451             else:
452                 answer.set('fraction', '0')
453
454
455
456     """
457     Función que crea una pregunta Respuesta corta en formato MoodleXML.
458     """

```

```

459     def produceSAQuestion(self,question,tag, questionParse ,choices, point,score):
460
461         #Creamos el contenido de la pregunta
462         name = ET.SubElement(question, "name")
463         self.addText(name, str(tag.attrib["title"]))
464
465         questiontext = ET.SubElement(question, "questiontext", format="html")
466         self.addText(questiontext, str(questionParse))
467
468         #Creamos atributos/elementos correspondientes a la pregunta
469         generalfeedback = "Las respuestas correctas son " + str(choices)
470         defaultgrade = str(point)
471         self.defaultMarks(question, generalfeedback, defaultgrade)
472         usecase = ET.SubElement(question, "usecase")
473         usecase.text = "0"
474
475         #Creamos las respuestas
476         for choice in choices:
477
478             answer = ET.SubElement(question, "answer", fraction=score, format="
479                 moodle_auto_format")
480             self.addText(answer, choice)
481
482             awfeedback = ET.SubElement(answer, "feedback",format="html")
483             awfeedbacktext = ET.SubElement(awfeedback, "text")
484             awfeedbacktext.text = ""
485
486         """
487         Función que crea una pregunta Respuesta numérica en formato MoodleXML.
488         """
489     def produceNUMQuestion(self,question,tag, questionParse ,choices,tolerancias,
490         point,scores,tipounidades):
491
492         #Creamos el contenido de la pregunta
493         name = ET.SubElement(question, "name")
494         self.addText(name, str(tag.attrib["title"]))
495
496         questiontext = ET.SubElement(question, "questiontext", format="html")
497         self.addText(questiontext, str(questionParse))
498
499         #Creamos atributos/elementos correspondientes a la pregunta
500         self.defaultMarks(question, "", str(point))
501
502         #Creamos las respuestas
503         for i in range( len(choices)):
504
505             answer = ET.SubElement(question, "answer", fraction=scores[i], format="
506                 moodle_auto_format")
507             self.addText(answer, choices[i])
508
509             awfeedback = ET.SubElement(answer, "feedback",format="html")
510             awfeedbacktext = ET.SubElement(awfeedback, "text")
511             awfeedbacktext.text = ""
512
513             tolerance = ET.SubElement(answer,"tolerance")

```

```

512         tolerance.text = str(tolerancias[i])
513
514
515         ####Creamos marcas que añaden más información a la pregunta con valores por
           defecto
516
517         #Cómo se introducen las unidades (0 input, 1 radio, 2 select)
518         unitgradingtype = ET.SubElement(question, "unitgradingtype")
519         if tipoUnidades == "Decimal":
520             unitgradingtype.text = "0"
521
522         #Penalización por unidad incorrecta
523         unitpenalty = ET.SubElement(question, "unitpenalty")
524         unitpenalty.text="0.1"
525
526         #Calificación de unidades (3 none, 1 graded, 0 optional)
527         showunits = ET.SubElement(question, "showunits")
528         showunits.text="3"
529
530         #En qué posición se ponen las unidades
531         unitsleft = ET.SubElement(question, "unitsleft")
532         unitsleft.text = "0"
533
534
535         """
536         Función que crea una pregunta Empaquetamiento en formato MoodleXML.
537         """
538         def produceMATCHQuestion(self,question,tag, questionParse , subquestions ,choices
           ,point):
539
540         #Creamos el contenido de la pregunta
541         name = ET.SubElement(question, "name")
542         self.addText(name, str(tag.attrib["title"]))
543
544         questiontext = ET.SubElement(question, "questiontext", format="html")
545         self.addText(questiontext, str(questionParse))
546
547         #Creamos atributos/elementos correspondientes a la pregunta
548         generalfeedback = ""
549         defaultgrade = str(point)
550         self.defaultMarks(question, generalfeedback, defaultgrade)
551         shuffleanswers = ET.SubElement(question, "shuffleanswers")
552         shuffleanswers.text = "true"
553         correct = " <p>Respuesta correcta</p>"
554         parcial = " <p>Respuesta parcialmente correcta.</p> "
555         incorrect = " <p>Respuesta incorrecta.</p> "
556         self.feedbackMarks(question, correct, parcial, incorrect)
557
558         #Creamos las respuestas
559         for i in range(len(subquestions)):
560             #Columna izquierda
561             subquestion = ET.SubElement(question, "subquestion", format="html")
562             self.addText(subquestion,subquestions[i])
563
564             #Columna derecha
565             ansSub = ET.SubElement(subquestion, "answer")

```

```
566         self.addText(ansSub, choices[i])
567
568
569     """
570     Función que añade al fichero de salida las marcas de un fichero XML y la
571     estructura del arbol.
572     """
573     def writequestionfile(self, quiz, filename):
574
575         f = open(filename, "w")
576         f.write('<?xml version="1.0" encoding="utf-8" standalone="yes"?>')
577         try:
578             f.write(ET.tostring(quiz, encoding='utf-8', method='xml').decode('utf-8')
579                     )
580         except Exception as e:
581             error_message = f'Error al añadir el arbol al fichero. Technical details
582                             :\n\t{e}'
583             print(error_message)
584             print("\n")
585             return -1
586
587         print(f"Wrote {filename}")
588
589     """
590     Función que añade algunas marcas HTML para una mejor presentación
591     """
592     def fixHtmlText(self, text):
593
594         text = urllib.parse.unquote(str(text))
595
596         text = re.sub('</div>', '</p></div> </div></prompt></div> ', text)
597         text = re.sub('<div>', ' <div><prompt><div> <div><p> ', text)
598
599         return text
600
601     """
602     Función que convierte el fichero QTI.
603     """
604     def convertQTI(self, inputfile, outputfolder):
605
606         if not os.path.exists(outputfolder): os.makedirs(outputfolder)
607         val = self.readQTI(inputfile, outputfolder)
608         #shutil.make_archive(outputfolder, 'zip', outputfolder)
609         return val
610
611     """
612     Función principal, con la que la interfaz se comunicará
613     """
614     def m_conv(self):
615
616         inputfile = self.file_input
617         outputfolder = self.path_out + '/ExportMOODLE_XML'
```

```
619
620     print(f'Input file is "{inputfile}"')
621     print(f'Output FOLDER is "{outputfolder}"')
622     try:
623         val = self.convertQTI(inputfile, outputfolder)
624         if val==0:
625             return 0
626         if val==-1:
627             return -1
628     except:
629         print("An exception occurred in convertQTI")
630         return -1
631     else:
632         return 1
```
