Universidad de las Ciencias Informáticas

Facultad 4

Integración del motor AQTIEngine a la plataforma educativa ZERA 2.0

Trabajo de Diploma para optar por el título de

INGENIERO EN CIENCIAS INFORMÁTICAS

Autor: Luis Miguel González Ravelo

Tutores: Ing. Adrián García Sánchez

Ing. Arley Enrique Cera Rojas

Ciudad de La Habana, 2016

"Año 58 de la Revolución"

Declaración de autoría.

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al Centro de Tecnologías para la Formación (FORTES) de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Luis Miguel González Ravelo

Ing. Adrián García Sánchez

Ing. Arley Enrique Cera Rojas

_____ _____ _____

RESUMEN

En la actualidad el proceso de enseñanza y aprendizaje ha alcanzado un alto desarrollo gracias a las nuevas tecnologías web, pues con el uso de los Sistemas de Gestión del Aprendizaje se puede garantizar un mejor control sobre dicho proceso en torno a los estudiantes. Con ellos surgieron los cursos masivos abiertos en línea, los cuales se han hecho muy populares, estos sistemas suelen incluir una herramienta de evaluación, la cual permite que muchos ejercicios se evalúen de forma automática, facilitando la tarea de calificación. En la Universidad de las Ciencias Informáticas, específicamente en el Centro de Tecnologías para la Formación de la Facultad 4 se desarrolla una plataforma educativa nombrada ZERA en su versión 2.0 la cual no cuenta con un sistema de evaluación para los ejercicios, este inconveniente se pudiera resolver con un algoritmo interno pero no evitaría que la calificación de sus ejercicios fuera interoperable, por lo que el presente trabajo de diploma tiene como objetivo contribuir al proceso de evaluación, y a la interoperabilidad de la calificación de los ejercicios en dicha plataforma. Como solución se llevará a cabo la integración del motor de evaluaciones AQTIEngine, el cual cuenta con características adaptables a la plataforma y que además está implementado bajo el estándar IMSQTI en su versión 2.0, con el fin de alcanzar la interoperabilidad de las calificaciones de los ejercicios en la plataforma. Para la validación de la integración con el sistema se realizaron pruebas unitarias y de integración.

PALABRAS CLAVE

Assessment, calificación, evaluación, interoperabilidad, motor de evaluación, QTI, rpTemplates.

ÍNDICE GENERAL

Índice

Índice

Índice

Índice

Índice

ÍNDICE DE FIGURAS

Índice

ÍNDICE DE TABLAS

INTRODUCCIÓN

A lo largo de estos últimos años el hablar sobre educación a distancia y la incorporación de las Tecnologías de la Información y las Comunicaciones (TIC) al proceso de enseñanza aprendizaje ha crecido exponencialmente. Sin duda las TIC están cambiando en gran medida la forma en que las personas se interrelacionan en cualquier ámbito.

La educación no está ajena a estos cambios, gracias a esta incorporación se han abierto nuevos caminos en la forma en que se aprende. Uno de estos caminos es el surgimiento del término e-learning (1) y con este también el nacimiento de las plataformas virtuales para la formación (2) más conocidas como Sistemas de Gestión del Aprendizaje (LMS, por sus siglas en inglés); estas son herramientas para la creación e impartición de cursos, con propósitos educativos y formativos, usando como medio principal Internet.

Los LMS están orientados al seguimiento del proceso de aprendizaje del estudiante, lo que posibilita la distribución de cursos, el trabajo colaborativo y realizar evaluaciones, entre otras funcionalidades. (3) Entre las capacidades básicas de los LMS se encuentra la evaluación del aprendizaje (4); la cual constituye un elemento fundamental en el proceso de enseñanza, tanto en la educación presencial como en la educación a distancia. (5) Además, sirve para identificar las fortalezas y debilidades del proceso de educación, con el fin de reajustar la intervención educativa y optimizarla.

Con los LMS también surgieron los cursos masivos abiertos en línea (MOOC por sus siglas en ingles). El concepto de MOOC fue manejado por primera vez en el año 2008 por Dave Cormier y Bryan Alexander (6) pero no es hasta unos años más tarde que comienzan su ascendente popularidad. Los MOOC, aunque el concepto fue enunciado, la definición formal aún es cambiante debido a la novedad del mismo, los MOOC son un fenómeno que ha alcanzado fuerza en los últimos años y que integra la conectividad de las redes sociales, el acceso de un reconocido experto en un campo de estudio y una colección de recursos en línea de libre acceso, siendo su cualidad más importante la posibilidad de participación activa

de varios cientos a varios miles de "estudiantes" que se auto-organizan de acuerdo con los objetivos, conocimientos y habilidades previas y los intereses comunes de aprendizaje. (7)

Una de las características que no debe faltar en un MOOC es la interoperabilidad, debido a la necesidad que existe en el proceso de enseñanza-aprendizaje de intercambiar información entre sistemas heterogéneos en ambientes homogéneos. Con el objetivo de lograr que estos sistemas sean interoperables, se nesecita un medio y un canal de comunicación. La Arquitectura Orientada a Servicios (SOA, por sus siglas en inglés) ofrece innumerables beneficios por lo que podría constituir el canal de comunicación (8). SOA es una arquitectura con acoplamiento débil, esto quiere decir que el cliente de un servicio no necesita conocer los detalles de la implementación del servicio, solamente requiere conocer y tener acceso a la interfaz del mismo. La forma más habitual de implementar dicha arquitectura es mediante servicios web (9).

Los estándares podrían constituir el medio de comunicación, ya que permiten crear una estructura común que facilita que los LMS puedan intercambiar información y crear módulos que funcionen correctamente en cualquier plataforma sin importar en cual estén. (10)

Los estándares son emitidos por organismos internacionales como IMS Global Learning Consortium (IMS) y Institute of Electrical and Electronics Engineers (IEEE) por citar algunos ejemplos. Mientras que las especificaciones son emitidas por grupos y consorcios. (11)

IMS desarrolla y mantiene más de veinte especificaciones distintas centradas en e-learning, dentro de ellas se pueden mencionar: IMS Content Packaging (IMS CP), IMS Learner Information Package (IMS LIP) y IMS Question and Test Intropera bility (IMS QTI).

La especificación IMS QTI, describe un modelo de datos para la representación de preguntas y exámenes y los resultados que los alumnos obtienen en éstos, también permite el intercambio de preguntas, exámenes y resultados entre herramientas de creación de preguntas y/o exámenes, bancos de preguntas,

sistemas de aprendizaje. El modelo de datos está descrito usando UML para que sea adaptable a diferentes lenguajes de programación, para el intercambio entre sistemas facilita un XML.

En la actualidad se han incorporado las especificaciones a los LMS, el consumo de servicios web y la utilización de los motores de evaluación. Estas últimas se encuentran entre las alternativas más utilizadas, ejemplo de estas aplicaciones son: e-QTI (12), newApi (13), Rendering and Reponses Processing (R2Q2) (14), PersonFit (15), The MathQTI Engine (16), entre otros. Estos motores son herramientas que permiten garantizar la calificación automática de los ejercicios y que además esta sea interoperable.

Después de un estudio previo realizado por los Ingenieros Arley Enrique Cera Rojas y Nayara Leal Bauta en el año 2015 sobre los motores de evaluación, estos llegaron a la conclusión de que la mayoría de estas herramientas se encontraban encaminadas a un área en específico además de tener otras desventajas, por lo que implementaron un nuevo motor de evaluación nombrado AQTIEngine. (17)

El motor de evaluación AQTIEngine fue implementado bajo la especificación IMS QTI en su versión 2.0, para evitar la necesidad de que las plataformas tengan que utilizar un algoritmo interno de evaluación. El mismo, permite que estas puedan consumir de un servicio web que brinda, el cual solo necesita las respuestas de los estudiantes y los rpTemplate, siendo este último las respuestas correctas que va a tener el ejercicio, junto con el algoritmo definido por la especificación, todo esto en una estructura de archivo XML. Luego el motor ejecuta su procedimiento interno y retorna la calificación para ese ejercicio. Agregar que el motor soporta todas las interacciones de ejercicio definidas por el estándar antes mencionado. (17)

En la Universidad de las Ciencias Informáticas (UCI) se está desarrollando una plataforma educativa nombrada ZERA en su versión 2.0. Entre sus funcionalidades cuenta con un módulo destinado al manejo de ejercicios, los cuales para garantizar su interoperabilidad están definidos bajo la especificación IMS QTI. Esta no tiene integrado un motor de evaluaciones ni un algoritmo interno que cumpla con estos estándares, lo cual no garantiza que la calificación que se otorga al estudiante sea interoperable.

De lo planteado anteriormente surge la siguiente interrogante la cual define el **problema a resolver**:

Introducción

¿Cómo contribuir a la interoperabilidad de la calificación de los ejercicios en la plataforma educativa ZERA 2.0?

Tomando como **objeto de estudio**: los procesos de evaluación automatizada en los sistemas e-learning, donde **el campo de acción** estará enmarcado en la interoperabilidad de la calificación de ejercicios en la plataforma educativa ZERA 2.0

Para dar solución al problema existente se plantea como **objetivo general**:

Integrar el motor de evaluación AQTIEngine desarrollado bajo la especificación IMS QTI v2.0 en la plataforma educativa ZERA 2.0 para contribuir a la interoperabilidad de la calificación de los ejercicios.

Para cumplir con el objetivo general se derivan los siguientes **objetivos específicos**:

-Investigar los aspectos teóricos fundamentales que sustentan la investigación, mediante consultas, extracción y recopilación de información relevante sobre el problema a resolver.

-Definir las tecnologías, las herramientas y la metodología para la integración del Motor de evaluaciones AQTIEngine a la plataforma educativa ZERA 2.0.

-Implementar la integración del motor AQTIEngine.

-Comprobar el funcionamiento del motor en la plataforma.

Para dar cumplimiento a los objetivos específicos se planifican las siguientes **tareas de investigación**:

-Estudio de la bibliografía actualizada para la elaboración del marco teórico-conceptual referente a los sistemas para la evaluación en plataformas e-learning.

-Identificación de los requisitos con los que debe cumplir la integración.

-Generación de los artefactos correspondientes con la metodología de desarrollo de software seleccionada.

-Implementación de un sistema de control de la calidad: que garantice que todos los componentes se encuentran funcionando correctamente y que la adición de un nuevo componente no afecte los demás, utilizando para ello pruebas unitarias.

Para dar solución a la interrogante planteada en el problema a resolver, la investigación se sustenta en la siguiente **Hipótesis**:

La integración del motor de evaluación AQTIEngine contribuye a la interoperabilidad de la calificación de los ejercicios en la plataforma educativa ZERA 2.0.

Los **métodos investigativos** que se utilizan para la presente investigación son:

Métodos Teóricos

Analítico-sintético, para identificar los elementos más importantes para dar solución al problema planteado tras haber realizado un estudio de las fuentes bibliográficas existentes referentes al tema.

Hipotético-deductivo, para la elaboración de la hipótesis de la presente investigación y en la obtención de nuevas líneas de trabajo a partir de los resultados adquiridos.

Análisis documental, en la consulta de la bibliografía especializada en los tópicos a fines de la investigación.

Métodos Empíricos

Observación: permitiendo estudiar más de cerca el objeto de la investigación, las acciones, causas, consecuencias, etc. Se pudo observar como las plataformas educativas son interoperables.

Para una mejor comprensión de la investigación, se decidió definir una estructura capitular que aporte cierto grado de organización y facilite el estudio del documento. Los capítulos que lo conforman son los siguientes:

Capítulo 1: Fundamentación teórica.

Se analiza y se exponen los enfoques teóricos, las principales investigaciones que anteceden a la presente, con el objetivo de generar el marco teórico y describir los principales elementos y tecnologías utilizadas para garantizar que en la plataforma educativa ZERA 2.0 las calificaciones de los ejercicios sean interoperables.

Capítulo 2: Características de la solución, Análisis y Diseño.

Se presenta la descripción de la propuesta de solución, se presenta el modelo del dominio y se especifican los requisitos que debe cumplir el sistema. Además, se generan las Historias de Usuario (HU) teniendo en cuenta el escenario adoptado y las descripciones textuales de cada requisito funcional. Se realiza el diseño del sistema, presentando los diagramas de clases de análisis y de diseño con estereotipos web. Se diagrama además el modelo de bases de datos, presentando la descripción de cada una de las tablas del modelo.

Capítulo 3: Implementación y prueba

En este capítulo se describe el proceso de implementación para la integración del motor de evaluación AQTIEngine a la plataforma educativa ZERA 2.0, de definiendo los tipos y niveles de pruebas a utilizar para garantizar una solución con calidad.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En el presente capítulo se realizará una breve descripción sobre los conceptos y aspectos teóricos más relevantes que aborda el campo de acción y que sustentan el desarrollo de la solución. Por último, a partir de la arquitectura definida en la plataforma ZERA 2.0 se hace una breve descripción de las metodologías, lenguajes de programación, tecnologías y herramientas definidas para dar respuesta a la problemática planteada.

1.2 El e-learning:

El e-learning o aprendizaje electrónico consiste en la educación y capacitación a través de Internet. Este tipo de enseñanza online permite la interacción del usuario con el material mediante la utilización de diversas herramientas informáticas.

Este nuevo concepto educativo es una revolucionaria modalidad de capacitación que posibilitó Internet, y que hoy se posiciona como la forma de capacitación predominante en el futuro. Este sistema ha transformado la educación, abriendo puertas al aprendizaje individual y organizacional. Es por ello que hoy en día está ocupando un lugar cada vez más destacado y reconocido dentro de las organizaciones empresariales y educativas.

La autora Maricarmen González Videgaray presenta el concepto de e-learning como “aprendizaje basado en tecnologías de la información y de la comunicación con interacciones pedagógicas entre estudiante y contenidos, estudiante y estudiante, y estudiante e instructor”. (1)

Roberto Baelo Álvarez en su artículo El e-learning, una respuesta educativa a las demandas de las sociedades del siglo XXI cita al autor Martín Hernández, el cual plantea que “el e-learning engloba aquellas

aplicaciones y servicios que, tomando como base las TIC, se orientan a facilitar el proceso de enseñanza aprendizaje". (18)

Podemos observar que entre los diferentes autores existe variedad de criterios con respecto a la definición de e-learning, algunos autores lo tratan como una modalidad de enseñanza-aprendizaje y otros lo consideran un proceso de aprendizaje a distancia, pero todos concuerdan en que se realiza haciendo uso de las nuevas tecnologías. Un grupo de científicos de la Universidad Abierta de Cataluña realizaron una investigación con el objetivo de llegar a un consenso referente a la definición del e-learning, y finalmente llegaron a la siguiente definición (19):

El e-learning es "una modalidad de enseñanza y aprendizaje que puede representar todo o una parte del modelo educativo en el que se aplica, que explota los medios y dispositivos electrónicos para facilitar el acceso, la evolución y la mejora de la calidad de la educación y la formación".

En la presente investigación se defiende precisamente el presentado por el grupo de científicos de la Universidad Abierta de Cataluña.

El uso del e-learning en la educación ha provocado una serie de mejoras significativas en la enseñanza. Facilita la colaboración entre docentes y estudiantes más allá de los límites físicos y académicos del centro educativo al que pertenecen. Las personas que por diferentes razones no puedan acudir a las aulas, pueden cursar estudios utilizando las bondades de Internet, pues las instituciones pueden ofertar cursos y programas de estudio. La enseñanza ha empezado a adoptar un carácter semi-presencial, causando una flexibilización del horario escolar. Permite que el estudiante adquiera nuevos conocimientos, además de los proporcionados por su profesor. El tiempo y el espacio no constituyen barreras en la enseñanza pues las nuevas tecnologías permiten incrementar la interacción entre el profesor y sus alumnos. (20)

Con el desarrollo del e-learning surgen las plataformas virtuales para la formación más conocidas como Sistemas de Gestión del Aprendizaje los cuales se describen a continuación.

1.2.1 Sistemas de Gestión del Aprendizaje:

LMS (Learning Management System) o sistema para la gestión del aprendizaje es una aplicación instalada en un servidor web que posibilita la distribución de cursos en línea de instituciones u organizaciones permitiendo así el aprendizaje electrónico.

El conjunto de herramientas de un LMS permite realizar cinco funciones principales: la administración del espacio de aprendizaje; la comunicación de los participantes; la gestión de contenidos; la gestión del trabajo en grupos, y la evaluación; constituyendo esta última una de las más importantes debido a que permiten la creación, edición y realización de ciertos tipos de test, anónimos o nominales, de trabajos, la autocorrección o la corrección (con realimentación), la calificación y publicación de calificaciones y la visualización de información estadística sobre los resultados, y también, el progreso de cada alumno. (21)

Asimismo, en la evaluación radica una de las funciones pedagógicas principales del empleo de los LMS, debido a que facilitan la creación de ejercicios de evaluación y autoevaluación. La evaluación de los aprendizajes le proporcionaría al profesor información sobre el proceso en la adquisición de conocimientos y destrezas por parte del alumno y también sobre la efectividad del diseño, y sobre el desarrollo, de todo el sistema de formación. Mientras que al alumno lo acercara a su propio progreso a través de ejercicios de autoevaluación. (22)

1.2.2 MOOC

Los MOOCs (acrónimo en inglés de Massive Open Online Course) o COMA en español (Curso Online Masivo Abierto) son cursos en línea dirigidos a un amplio número de participantes a través de Internet según el principio de educación abierta y masiva. El término fue acuñado en 2008 por Dave Cormier y Brian Alexander. (23)

De forma más simplificada se dice que es una clase apoyada por la Web, diseñada para aceptar un gran número de participantes. (24)

1.2.3 Análisis de sistemas similares.

En la actualidad existen un gran número de MOOCs alrededor de todo el mundo, entre los más conocidos están Coursera, Miríada X y Udacity. A continuación se realiza un análisis de las principales características y datos de estos sistemas.

1.2.3.1 Coursera

Es una plataforma de educación virtual nacida en octubre de 2011 y desarrollada por académicos de la Universidad de Stanford con el fin de brindar oferta de educación masiva a la, con cursos en inglés y otros idiomas como el español, francés, italiano y chino. Coursera ofrece cursos, tanto gratuitos como de pago, sobre temas variados a niveles universitarios, pero abiertos a todos los sectores de la población. Fue fundada por los profesores en ciencias computacionales Andrew Ng y Daphne Koller en octubre de 2011 con el lanzamiento de dos cursos gratuitos, "Aprendizaje automático" e "Introducción a las bases de datos". En septiembre de 2012, Coursera ya contaba con 1.2 millones de estudiantes, 121 cursos y 33 Universidades asociada (25), en diciembre de 2012 llegó a 1.9 millones de estudiantes. En enero de 2013, anunció que el American Council on Education aprobó cinco cursos para otorgar créditos universitarios. (25) En febrero de 2015 se informó que Coursera firmó una alianza con Google e Instagram para emitir "microdegrees", que incluye una práctica en el proyecto final que serán diseñadas en conjunto con las principales universidades y las principales empresas de alta tecnología. (26)

1.2.3.2 Miríada X

Es un proyecto de formación en línea que tiene su origen a principios del año 2013 por el Banco Santander y Telefónica, a través de la Red Universia y Telefónica Educación Digital y basado en la plataforma de software libre WEMOOC. Ofrece cursos en línea masivos y en abierto de forma gratuita y aptos para cualquier usuario interesado en el contenido del mismo. En 2014, Miríada X contó con la participación de 45 universidades de nueve países: España, Colombia, Chile, Argentina, Perú, México, Brasil, Puerto Rico, República Dominicana y El Salvador; más de 1.000 profesores y 195 cursos impartidos. (27)

Su éxito la ha llevado a convertirse en una plataforma de formación online de referencia no solo a nivel español, sino también europeo, en el que más de un 40% de los MOOCs provienen de universidades españolas según el portal Open Education Europa, siendo Miríada X factor clave en la evolución educativa española según muestra el informe de la Sociedad de la información en España en el año 2013. (28)

1.2.3.3 Udacity

Es es una organización educativa con ánimo de lucro fundada por Sebastian Thrun, David Stavens y Mike Sokolsky que ofrece cursos online masivos y abiertos. Según Thrun, el origen del nombre Udacity proviene del deseo de la compañía de ser "audaz para ti, el estudiante". (29)

Udacity es el resultado de las clases de informática gratuitas ofrecidas en el año 2011 a través de la Universidad de Stanford. (30) Desde el 1 de agosto de 2013, Udacity tiene 25 cursos activos. (31) Thrun ha declarado que espera se matriculen medio millón de estudiantes después de una inscripción de 160.000 alumnos en el curso predecesor en Stanford, Introducción a la Inteligencia Artificial, y 90.000 estudiantes se inscribieron en las dos clases iniciales a partir de marzo de 2012. Udacity se anunció en la conferencia Digital Life Design 2012. Udacity es financiado por la firma de capital de riesgo, Charles River Ventures y $300.000 de dinero personal de Thrun (29), en octubre de 2012, la firma de capital riesgo Andreessen Horowitz llevó otra inversión de $15 millones a Udacity. (32)

Una de las características fundamentales que tienen estos MOOCs es la interoperabilidad, puesto que existe una gran necesidad de puedan intercambiar información entre ellos, este tema será abordado con mayor profundidad en el siguiente epígrafe.

1.3 Interoperabilidad

La interoperabilidad es la condición que hace posible que las diferencias entre dos o más sistemas de información no sean una barrera para que estos puedan comunicarse y utilizar los contenidos y servicios respectivos, con el fin de desarrollar una tarea determinada. (33)

La Real Academia de Ingeniería define la interoperabilidad como la “condición que permite a equipos o municiones ser usados por varios sistemas, aunque con menos prestaciones que los completamente intercambiables”. (34)

La red temática IST IDEAS ha definido la interoperabilidad como: “la habilidad de un sistema o producto de trabajar con otros sistemas o productos sin esfuerzo especial de la parte del usuario ". (35)

Según Tamayo y otros (36), la interoperabilidad es la capacidad que deben tener dos o más sistemas o componentes de intercambiar información y poder utilizar esa información”.

En el marco de la investigación se utilizaría como referencia la definición planteada por varios autores en el artículo Interoperabilidad de sistemas de organización del conocimiento.

La interoperabilidad entre plataformas tecnológicas hace posible que los contenidos y las herramientas de aprendizaje se puedan intercambiar y utilizar desde distintos entornos (33). La interoperabilidad en los LMS se podría garantizar utilizando un medio y un canal de comunicación, SOA por sus grandes ventajas podría establecer el canal de comunicación mientras que la utilización de estándares podría constituir el medio de comunicación.

1.3.1 Arquitectura Orientada a Servicios

La arquitectura SOA establece un marco de diseño para la integración de aplicaciones independientes, de manera que desde la red pueda accederse a sus funcionalidades, las cuales se ofrecen como servicios. La forma más habitual de implementarla es mediante servicios web. Estos pueden ser catalogados como aplicaciones localizadas en algún lugar de la red y que puede ser accedido mediante protocolos de Internet (como Hypertext Transfer Protocol (HTTP)) utilizando interfaces bien definidas, descritas usando un lenguaje de descripción funcional estándar. Los servicios web se basan en un conjunto de estándares de comunicación, como son XML para la representación de datos, SOAP, el lenguaje Web Services Description Language (WSDL) y Universal Description, Discovery, and Integration (UDDI).

La arquitectura SOA es una instancia del modelo de desarrollo de software basado en componentes, donde los servicios web son considerados los componentes (9). Un diseño basado en componentes permite que cada elemento pueda ser reutilizado, actualizado y reemplazado. Es aquí donde las arquitecturas SOA juegan un papel fundamental, ofreciendo una manera de interconectar sistemas utilizando tecnologías independientes de la plataforma como SOAP y Representational State Transfer(REST) . (37)

SOAP: es un protocolo pensado para el intercambio de información en entornos descentralizados y distribuidos. Usa las tecnologías relacionadas con XML a fin de definir un marco de trabajo extensible para los mensajes y se apoya en WSDL y UDDI. Los dos objetivos de diseño principales de SOAP son la simplicidad y la extensibilidad. Para alcanzar estos objetivos, SOAP simplemente elimina de su arquitectura aquellos aspectos que con más frecuencia se encuentra en los sistemas distribuidos. (38)

REST: es una técnica de arquitectura software para sistemas web más restringido y fiable. Se sustenta sobre los estándares de HTTP y Uniform Resource Identifier(URI), que sirve para identificar recursos en Internet. Un concepto importante en REST es la existencia de recursos; para manipular estos recursos, los componentes de la red (clientes y servidores) se comunican a través de una interfaz estándar (HTTP) e intercambian representaciones de estos recursos.

En síntesis, los servicios basados en REST han surgido como una alternativa prometedora diferente de los servicios basados en SOAP, gracias a su capacidad de trasmitir datos directamente sobre HTTP y además a su simplicidad y naturaleza liviana. REST constituye una arquitectura simple, escalable, eficiente, segura, confiable y extensible. La arquitectura multinivel tanto para servicios web como para aplicaciones web dinámicas conlleva a la reutilización, simpleza, extensibilidad y a una clara separación de las responsabilidades de los componentes. Por lo antes expuesto, para garantizar la interoperabilidad de la calificación de los ejercicios en la plataforma educativa ZERA 2.0 el desarrollador utilizará los servicios basados en REST.

1.3.2 Estándares

Los estándares son acuerdos internacionales documentados o normas establecidas por consenso mundial. Contienen las especificaciones técnicas y de calidad que deben reunir todos los productos y servicios para cumplir satisfactoriamente con las necesidades para las que han sido creadas y para poder competir internacionalmente en condiciones de igualdad. (39)

Los estándares en el ámbito del e-learning especifican cómo los fabricantes pueden construir cursos online y las plataformas sobre las cuales son impartidos estos cursos, de tal manera que puedan interactuar unas con otras.

Además, permiten crear tecnologías de aprendizaje más poderosas, personalizar el aprendizaje basándose en las necesidades individuales de los alumnos.

Entre los principales objetivos que persigue la aplicación de un estándar en el e-learning se encuentran los siguientes (40):

Durabilidad: que la tecnología desarrollada con el estándar evite la obsolescencia de los cursos.

Interoperabilidad: que se pueda intercambiar información a través de una amplia variedad de LMS.

Accesibilidad: que un usuario pueda acceder al contenido apropiado en el momento justo y en el dispositivo correcto.

Reusabilidad: que los distintos cursos y objetos de aprendizaje puedan ser reutilizados con diferentes herramientas y en distintas plataformas.

Adaptabilidad: que se facilite la adaptación o personalización del entorno de aprendizaje.

Productividad: si los proveedores de tecnología e-learning desarrollan sus productos siguiendo estándares comúnmente aceptados, la efectividad de e-learning se incrementa significativamente y el tiempo y costos se reducen.

El empleo de los estándares en e-learning trae consigo múltiples beneficios por diversas razones. Los estándares garantizan la viabilidad futura de la inversión en productos del e-learning, impidiendo que sean dependientes de una única tecnología. Producto a esto la oferta de cursos disponibles en el mercado aumenta considerablemente, disminuyendo de esta manera los costos de adquisición y evitando costosos desarrollos. Además, posibilita el intercambio y compraventa de cursos, proporcionando incluso que las organizaciones adquieran rendimientos extraordinarios sobre sus inversiones. Por otra parte, facilita la aparición de herramientas estándar para la creación de contenidos, permitiendo que las organizaciones puedan desarrollar contenidos sin la necesidad de acudir a especialistas en e-learning. (40)

1.2.3.1 Organizaciones encargadas de la estandarización

La creación de estándares globales es una tarea compleja. Sería muy difícil llegar a un consenso que cubra las necesidades de todos o de una gran mayoría para ser adoptados de forma genérica, sin embargo, diferentes grupos están trabajando en el desarrollo tanto de especificaciones como de estándares en los diferentes niveles que se requieren, para poder establecer entornos e-learning integrados e interoperables. (11)

Los esfuerzos de los cuerpos que desarrollan especificaciones y estándares están orientados hacia una forma común de identificar, definir y comunicar a todos los recursos involucrados en un entorno e-learning (contenidos, docentes, estudiantes, aplicaciones, proveedores, etcétera). Estos trabajos tuvieron sus inicios en grupos que comenzaron a trabajar diferentes áreas de los estándares. A continuación se mencionan los cuerpos que están trabajando en el desarrollo de propuestas para la estandarización del e-learning:

Aviation Industry Computer Based Training Committee (AICC), Advanced Distributed Learning (ADL), Alliance of Remote Instructional Authoring and Distribution Networks for Europe (ARIADNE), IEEE/Learning Technology Standards Committee (LTSC) y IMS. De estos grupos de desarrollo, una de las propuestas más ampliamente adoptadas en un gran número de herramientas, ha sido la propuesta de IMS Learning Consortium que cuenta con miembros de organizaciones comerciales, educativas y gubernamentales dedicadas a definir y distribuir arquitecturas abiertas para actividades de educación en línea. Uno de sus resultados es lo que se conoce como el estándar IMS. (11)

1.4 IMS Learning Consortium

IMS es una iniciativa ambiciosa, que propone el uso de un lenguaje común basado en XML, para la identificación homogénea de recursos entre los sistemas de aprendizaje. A través de detalladas especificaciones cubre las necesidades para la interoperabilidad entre los sistemas de diferente naturaleza. Las especificaciones que tiene abarca, entre otros puntos, accesibilidad y adaptación del estudiante, la definición de competencias como requisito o como resultado de un aprendizaje, el empaquetamiento de contenidos, la formación de repositorios de contenidos digitales, información de agentes del proceso educativo, así como un lenguaje para expresar diferentes modelos pedagógicos. (41)

Entre las especificaciones disponibles de IMS se pueden mencionar: IMS Reusable Definition of Competency or Educational Objective (IMS RDCEO), IMS CP, IMS Digital Repositories Interoperability (IMS DRI), IMS LIP, IMS Shareable State Persistente (IMS SSP), IMS Simple Sequencing (IMS SS) y IMS QTI, entre otras.

Esta última especificación propone la descripción de preguntas y tests basándose en el lenguaje estándar XML. Además, está orientada a permitir la interoperabilidad de contenido entre los sistemas de evaluación. Siendo útil para editores, autoridades de certificación y maestros quiénes podrán importar y exportar sus datos entre sistemas compatibles. (11)

1.4.1 IMS, Interoperabilidad de Preguntas y Pruebas

Historia

IMS QTI es una de las especificaciones en las que el consorcio IMS trabajó más tempranamente (los primeros trabajos se realizaron en 1999). Sin embargo, el potencial de la especificación IMS QTI no se ha explotado completamente debido en parte a su complejidad y a la escasa existencia de herramientas informáticas que pusieran en práctica (habitualmente de manera parcial) la especificación. En la historia de IMS QTI hay 3 versiones de la misma que han tenido una gran repercusión: IMS QTI versión 1.2, IMS QTI versión 2.0 y finalmente IMS QTI versión 2.1. (42)

IMS QTI versión 1.2 (aparece en 2002) es la última versión finalizada completamente en la que se abarcan tanto preguntas individuales como exámenes completos. Una vez publicada la especificación, surgieron diversos problemas a la hora de implementarla. Aunque se publicó un suplemento (IMS QTI 1.2.1), parte de los problemas que habían surgido requerían de grandes cambios de manera que dichas modificaciones provocarían que se perdiera la compatibilidad con las versiones anteriores. Además, algunas otras partes de la especificación necesitaban clarificarse o extenderse para resolver los problemas que habían surgido durante su puesta en práctica. (42)

Desde la aparición de IMS QTI 1.2 y con el desarrollo de nuevas especificaciones como el IMS Content Packaging, IMS Simple Sequencing e IMS Learning Design surgió la necesidad de hacer compatible la especificación IMS QTI a las nuevas iniciativas. Así apareció la versión 2.0 (2005) de la especificación donde se comenzó con la armonización con las otras especificaciones y el comienzo de la resolución de los problemas de la antigua especificación. Sin embargo, para simplificar el proceso de adopción y permitir un trabajo razonable, esta especificación ser concentró sólo en las preguntas individuales y no actualiza aquellas partes que tienen que ver con la composición de dichas preguntas, es decir, la creación de exámenes completos. (42)

En el desarrollo de la propuesta de solución se trabajará con la versión 2.0 de IMS QTI, pues esta es la utilizada por la plataforma ZERA 2.0 y el motor de evaluaciones AQTIEngine.

Descripción

La especificación IMS QTI describe un modelo de datos completo que permite una representación mediante el lenguaje XML de preguntas y exámenes completos. Además, incluye la interacción disponible para el tipo de pregunta, las reglas completas para calificar la pregunta, las normas para ordenar y seleccionar qué preguntas del examen se van a representar y como evaluar el examen. (43)

El objetivo fundamental de esta especificación es permitir el intercambio de preguntas, evaluaciones y resultados entre distintas herramientas. Además, ha sido diseñada para mantener la interoperabilidad e innovación a través de puntos de extensión y escalabilidad bien definidos. Los puntos de extensión pueden ser utilizados para representar ítems directamente sin importar la procedencia de los datos. (44)

La especificación contiene multitud de tipos de preguntas diferentes, incluyendo preguntas simples, de texto y gráficas, pudiendo incluso añadir a las preguntas archivos de multimedia, como pueden ser archivos de sonido y video. Además, las preguntas contienen en sí mismas la definición de cuál es la respuesta o las respuestas correctas, y sobre cómo puntuarlas. (43)

IMS QTI en ZERA 2.0 y AQTIEngine

En la plataforma educativa ZERA 2.0 actualmente hay implementadas siete tipos de interacciones de ejercicios, creadas bajo el estándar IMS QTI v2.0, las mismas son soportadas por el motor de evaluaciones AQTIEngine, ellas son:

-Selección simple

-Selección múltiple

-Ordenamiento cronológico

-Verdadero y falso

-Enlazar columnas

-Completar por desplazamiento

-Completar por escritura.

Dicho motor soporta todas las restantes interacciones definidas bajo este estándar, por lo que agregar una de ellas a la plataforma no traería consecuencias a la hora de calificar los ejercicios.

1.5 Motores de evaluación

Algunas plataformas e-learning como: Moodle, Sakai e ILIAS ofrecen una metodología de aprendizaje centralizado, basado en contenidos distribuidos por los profesores. Además, brindan solamente funcionalidades clásicas, y los métodos de evaluación están centrados en los contenidos teóricos, donde los estudiantes pueden verificar de inmediato su conocimiento realizando exámenes de opción múltiple o ejercicios del tipo de asociación conceptual. (45) Asimismo, presentan la imposibilidad de efectuar retroalimentaciones individuales al trabajo de un alumno con una herramienta en particular (46).

Para poder superar estas limitaciones en las plataformas de la segunda generación, una solución es el diseño de sistemas de enseñanza-aprendizaje orientados a servicios, entendido como un conjunto de servicios basados en la web. Estos sistemas conforman lo que se denomina la tercera generación de plataformas y contienen un elemento fundamental para el proceso de evaluación, conocido como motores de evaluación. (45)

El motor de evaluación constituye el núcleo del sistema y es el encargado de realizar la evaluación automática. Para realizar esta tarea el motor de evaluación debe acceder a la información proporcionada

por el entorno de trabajo del estudiante, y obtener los parámetros de entrada necesarios para poder funcionar, acorde con la evaluación configurada en él. (45)

Algunos ejemplos de ellos son:

1.5.1 e-QTI:

Es un motor altamente modular y extensible que facilita el ciclo de evaluación en términos de generación, ejecución, presentación, clasificación y archivado. Se basa en la especificación IMS QTI pero es capaz de realizar la exportación e importación de los ejercicios de evaluación en una amplia gama de formatos, también puede mantener una reserva de preguntas que pueden ser reutilizados en evaluaciones futuras. El motor es especialmente adecuado para ser integrado con plataformas web utilizando interfaces de aprendizaje bien definidas, orientado a Java y orientado a servicios. Es compatible con los siguientes tipos de preguntas: verdadero/falso, de opción múltiple, llenar los espacios vacíos, entre otras. Fue desarrollado utilizando tecnología Java con el fin de maximizar la flexibilidad y extensibilidad de la aplicación resultante. Se puede integrar fácilmente con sistemas e-learning a través de los servicios web, utilizando Simple Object Access Protocol (SOAP). Fue utilizado como parte de un LMS experimental llamado e-Aula, que constituye una colaboración entre el Centro de Estudios Superiores Felipe II y la Universidad Complutense de Madrid. (12)

1.5.2 NewAPIS

Forma parte de un mundo virtual en 3D, es un motor de IMS QTI que permite el intercambio de información con el mundo virtual para ver y guardar el progreso de los estudiantes. NewAPIS cumple con la especificación IMS QTI 2.1 y apoya la gestión de las pruebas completas. Es un motor de representación de elemento modular que está a cargo de interpretar archivos XML. Estos archivos tienen que estar previamente creados por una herramienta de creación compatible. Gestiona los XMLs QTI y crea el Extensible HyperText Markup Language (XHTML) de la prueba que se visualiza por los usuarios a través de un reproductor web. El motor calcula las interacciones de los usuarios al responder las preguntas.

Almacena todos los datos de la evaluación, comprueba las respuestas de los estudiantes y calcula sus puntuaciones utilizando la información contenida en los archivos Question and Test Introperability (QTI). (13)

1.5.3 R2Q2

Posee una arquitectura débilmente acoplada que consta de tres servicios interoperables. Las interacciones tanto internas como externas del motor son gestionadas utilizando un componente interno denominado router. El router es el responsable de analizar y trasmitir los diversos componentes a los servicios web responsables. Además, se encarga de la gestión de las interacciones de software externos con el sistema. Esto posibilita que los servicios sean más simples y que también puedan mantener una interfaz de acoplamiento flexible, pero sin la necesidad de intercambiar grandes volúmenes de Extensible Markup Language (XML). El procesador es el encargado de procesar las respuestas dadas por los usuarios y además de generar una retroalimentación. El procesador compara la respuesta de los usuarios con un conjunto de reglas y a su vez genera variables de respuesta basados en dichas reglas. Es un proyecto creado por la Universidad de Southampton, nace por la necesidad de que exista un punto inicial para probar la versión 2.0 de la especificación IMS QTI. (14)

1.5.4 PersonFit

Es un módulo que permite desarrollar evaluaciones adaptativas en el entorno de aprendizaje Moodle. El sistema tiene dos componentes principales: la parte cliente (PersonFit Client), que permite a los profesores definir los ítems, y la parte integrada con Moodle, donde se concentra el componente inteligente para el desarrollo de las evaluaciones adaptativas. Almacena todos los ítems en formato IMS QTI, y ofrece la posibilidad de importar material para evaluación, en este formato, desde otros módulos y aplicaciones. Ofrece los siguientes tipos de ítems: opción múltiple, verdadero/falso, respuesta múltiple, rellenar huecos, numérico, ordenación y emparejamiento. (15)

1.5.5 The MathQTI Engine

Es un módulo del Ambiente de Aprendizaje Wallis que interpreta los ejercicios de MathQTI, creado por la Universidad de Edinburgh en honor al matemático Jhon Wallis del siglo XVII. Emplea ejercicios interactivos que en cierta manera tienen un contexto predeterminado, son designados para una situación de aprendizaje específico. Además, son adaptativos en dependencia del tipo de retroalimentación que proveen. Los ejercicios interpretados son separados del resto del sistema, con la esperanza de reutilizar este motor en otros sistemas. MathQTI para la interpretación de fórmulas de OpenMath y para el procesamiento de respuesta con el sistema Yaca utiliza las librerías de RIACA JSP. Con extensiones simples y algunas librerías de código adicional el motor de evaluación MathQTI, provee un sistema de conceptos para el futuro desarrollo de Wallis y otros sistemas. (16)

En la presente investigación no se utilizarán ninguno de estos motores, se usará el motor AQTIEngine antes mencionado en la introducción de dicha investigación.

Una vez aclarado todos los conceptos y aspectos teóricos más relevantes que aborda el campo de acción y que sustentan el desarrollo de la solución, se pasará a seleccionar la metodología, las herramientas tecnologías y lenguajes que se usarán para dar cumplimiento a la implementación de dicha solución.

1.6 Herramientas, tecnologías, lenguajes y metodología.

Para la realización de la presente investigación el desarrollador realizó un estudio de varias herramientas, tecnologías, lenguajes y metodologías, con el objeto de poder realizar la integración el motor de Evaluación AQTIEngine a la plataforma educativa ZERA 2.0.

Para la selección de las herramientas a ser utilizadas posteriormente, se tuvo en cuenta las que más repercusión han tenido a nivel mundial, que presentan mayor bibliografía, que el equipo presenta un mayor dominio y por último y no menos importante que sean de software libre o código abierto. Además, la UCI ha determinado una estrategia marcaria para sus productos, la cual define los componentes del diseño

utilizando un conjunto de tecnologías. Dado que el motor AQTIEngine y la plataforma educativa ZERA 2.0 está definida bajo la estrategia antes mencionada el desarrollador seleccionó las tecnologías, lenguajes y framework del lado del cliente que también se ajustara a esta estrategia y dicho sea de paso a la línea base del proyecto.

1.6.1 Metodología de desarrollo de software

El éxito de cualquier software depende de muchos factores, pero uno de los más importantes es la selección de la metodología que se adecue a tu proyecto.

Una metodología es un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un nuevo software. Las metodologías de desarrollo de software pueden clasificarse en tradicionales/pesadas o ágiles. Las primeras centran su atención en llevar una documentación exhaustiva de todo el proyecto y en cumplir con un plan de proyecto, definido todo esto, en la fase inicial del desarrollo. Mientras que la segunda se basa en dos aspectos puntuales, el retrasar las decisiones y la planificación adaptativa; permitiendo potenciar aún más el desarrollo de software a gran escala. (47)

Rational Unified Process(RUP): es una metodología tradicional, además, es un proceso formal que provee un acercamiento disciplinado para asignar tareas y responsabilidades dentro de una organización de desarrollo. Su objetivo es asegurar la producción de software de alta calidad que satisfaga los requerimientos de los usuarios finales. Puede ser adaptado y extendido para satisfacer las necesidades de la organización que lo adopte. Es guiado por CU y centrado en la arquitectura, y utiliza Unified Modeling Language (UML) como lenguaje de notación (47)

RUP divide el proceso en cuatro fases, dentro de las cuales se realizan varias iteraciones en un número variable según el proyecto y en las que se hace un mayor o menor hincapié en las distintas actividades. Las cuatro fases del ciclo de vida son: inicio, elaboración, construcción y transición.

Microsoft Solution Framework(MSF): es una metodología tradicional, un modelo de proceso que describe una secuencia de alto nivel de las actividades para la construcción e implementación de soluciones. En lugar de prescribir una serie específica de los procedimientos, es lo suficientemente flexible para dar cabida a una amplia gama de proyectos. Combina dos modelos estándar de la industria: la cascada y la espiral. Es un proceso impulsado por hitos, estos son puntos en el proyecto cuando los entregables importantes se han terminado y pueden ser revisados. Está diseñado para adaptarse a las necesidades cambiantes de los proyectos, moviéndose a través de iteraciones de ciclos de desarrollo cortos y versiones incrementales de la solución. (48)

Scrum: es una metodología ágil de desarrollo de proyectos que toma su nombre y principios de los estudios realizados sobre nuevas prácticas de producción por Hirotaka Takeuchi e Ikujijo Nonaka a mediados de los 80. Es una metodología de desarrollo muy simple, que requiere trabajo duro porque no se basa en el seguimiento de un plan, sino en la adaptación continua a las circunstancias de la evolución del proyecto. Es orientado a las personas más que a los procesos. Emplea la estructura de desarrollo ágil: incremental basada en iteraciones y revisiones. Es un modo de desarrollo de carácter adaptable más que predictivo. Scrum denomina sprint a cada iteración de desarrollo y recomienda realizarlas con duraciones de 30 días. El sprint es por tanto el núcleo central que proporciona la base de desarrollo iterativo e incremental. (49)

Extreme Programming(XP): es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. (50)

Agile Unified Process(AUP): es una versión simplificada de RUP que utiliza técnicas y conceptos de este. Se basa en la gestión de riesgos proponiendo que aquellos componentes con alto riesgo tengan más prioridad que los demás y sean desarrollados en etapas tempranas del proyecto. Desarrolla prototipos ejecutables durante la fase de elaboración del producto, demostrando la validez de la arquitectura para los requisitos clave del producto y determinando los riesgos técnicos. Al igual que en RUP, en AUP se establecen cuatro fases que transcurren de manera consecutiva y que acaban con hitos claros alcanzados. El proceso AUP establece un modelo más simple que el que aparece en RUP por lo que reúne en una única disciplina las disciplinas de Modelado de Negocio, Requisitos y Análisis y Diseño. El resto de disciplinas (Implementación, Pruebas, Despliegue, Gestión de Configuración, Gestión y Entorno) coinciden con las restantes de RUP. (51)

Metodología de desarrollo para la Actividad productiva de la UCI (AUP_UCI): constituye una variación del Proceso Unificado Ágil, en unión con el modelo de desarrollo Capability Maturity Model Integration (CMMI-DEV) versión 1.3. Esta metodología se elaboró teniendo en cuenta las características del proceso productivo en la UCI y es aplicable a todos sus proyectos de desarrollo. De las 4 fases que propone AUP (Inicio, Elaboración, Construcción, Transición), la Metodología de desarrollo para la Actividad productiva de la UCI, mantiene la fase de Inicio, pero modificando el objetivo de la misma. Las restantes 3 fases de AUP se unifican en una sola, que recibe el nombre de Ejecución y se agrega la fase de Cierre. (52)

AUP propone 7 disciplinas (Modelo, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyecto y Entorno), la metodología definida en la UCI tiene 8 disciplinas, pero a un nivel más atómico que el definido en AUP. Los flujos de trabajos: Modelado de negocio, Requisitos y Análisis y diseño en AUP están unidos en la disciplina Modelo, en la variación para la UCI se consideran cada uno de ellos disciplinas. Se mantiene la disciplina Implementación, en el caso de Prueba se desagrega en 3 disciplinas: Pruebas Internas, de Liberación y Aceptación y la disciplina Despliegue se considera opcional. Las restantes 3 disciplinas de AUP asociadas a la parte de gestión, para la variación UCI, se cubren con

las áreas de procesos que define CMMI-DEV v1.3 para el nivel 2, serían CM (Gestión de la configuración), PP (Planeación de proyecto) y PMC (Monitoreo y control de proyecto). (52)

AUP propone 9 roles: Administrador de proyecto, Ingeniero de procesos, Desarrollador, Administrador de BD, Modelador ágil, Administrador de la configuración, Stakeholder, Administrador de pruebas, y Probador. La Metodología de desarrollo para la Actividad productiva de la UCI establece 11 roles: Jefe de Proyecto, Planificador, Analista, Arquitecto de información (opcional), Desarrollador, Administrador de la Configuración, Stakeholder (Cliente/Proveedor, de requisitos), Administrador de calidad, Probador, Arquitecto de software (Sistema), y Administrador de BD. (52)

Una vez concluida la investigación de las metodologías y observado sus características, se decidió que el desarrollo de la solución esté guiado por la metodología AUP_UCI establecida para el desarrollo de la actividad productiva de la UCI. Para la selección se tuvo en cuenta que el equipo de desarrollo es pequeño (solo una persona), el cliente pertenece a la Universidad y puede sostener reuniones periódicas con el desarrollador, características que se corresponden con los principios ágiles de AUP y la variante UCI. Además de que es la metodología que se usa en la línea base de la plataforma ZERA 2.0.

1.6.2 Lenguaje de Modelado

UML: permite a los creadores de sistemas generar diseños que capturen sus ideas en una forma convencional y fácil de entender para comunicárselas a otras personas. Está compuesto por diferentes elementos gráficos que se combinan para conformar diagramas. Debido a que es un lenguaje cuenta además con reglas para combinar tales elementos (53). La versión a utilizar sería la 2.1.

1.6.3 Tecnologías y lenguajes del lado del cliente

HyperText Markup Language(HTML): es lenguaje que se utiliza para crear documentos que muestren una estructura hipertexto. Además, permite crear documentos de tipo multimedia, es decir, que contengan información más allá de la simplemente textual como por ejemplo: imágenes, videos, sonido, entre otros.

Los documentos HTML se conforman como documentos de texto planos, en los que todo el formato del texto se especifica mediante marcas de texto (nombradas etiquetas), que delimitan los contenidos a los que afecta la etiqueta (54)

En el desarrollo de la integración del motor de evaluación a la plataforma el lenguaje HTML se utilizó para enmaquetar todo el contenido que se le mostraría al cliente en las páginas estáticas de la aplicación. La versión a utilizar sería la 5.0.

Cascading Style Sheets(CSS): es un lenguaje de hojas de estilo creado para controlar el aspecto o presentación de los documentos electrónicos HTML o XHTML. Es utilizado para definir el aspecto de cada elemento: color, tamaño, tipo de letra, separación vertical y horizontal entre elementos, posición de cada elemento dentro de la página (55). El lenguaje de hojas de estilo CSS se encargó de controlar el aspecto visual de las páginas estáticas de la aplicación, permitiendo construir un diseño más agradable a la vista del cliente. La versión a utilizar sería la 3.0.

JavaScript (JS): es el lenguaje de programación común más utilizado para añadir interactividad a una página web. Un programa en JavaScript se integra en una página web y es el navegador el que lo interpreta. Es decir, es un lenguaje interpretado no compilado pues no genera ningún tipo de fichero objeto. Es un lenguaje moderno, sencillo, muy útil, barato pues solo se necesita una bloc de notas y un navegador, además, es visual debido a que permite la moderna programación visual (56). El uso del lenguaje JavaScript facilitó al desarrollador la tarea de validar y ejecutar las acciones que se activan al pulsar botones o seleccionar elementos. La versión a utilizar sería la 1.8.

1.6.4 Tecnologías y lenguajes del lado del servidor

XML: Lenguaje de Marcado Extensible, puede utilizarse para crear documentos de texto que contienen datos con un formato estructurado. Además de los datos, puede incluirse un conjunto de reglas que definen de forma detallada la estructura de dichos datos. Las reglas se encuentran almacenadas en un

lugar centralizado y pueden ser utilizadas para construir mensajes XML estandarizados, los cuales pueden ser intercambiados entre diversas aplicaciones (57). Es el lenguaje empleado por IMS QTI para definir los algoritmos de evaluación de los ejercicios.

1.6.4.1 Lenguajes de procesamiento de datos:

Java: es un lenguaje de programación creado por James Gosling, Patrick Naughton, Chris Warth,

Ed Frank y Mike Sheridan en Sun Microsystems en 1991 basado en el lenguaje C++. Es orientado a objetos, interpretado por una máquina virtual, multiplataforma y fuertemente tipado. (58)

Proporciona numerosas comprobaciones en compilación y en tiempo de ejecución. Está diseñado para estaciones de trabajo, sobre arquitecturas distintas y con sistemas operativos diversos.

Python: es un lenguaje de alto nivel simple, muy expresivo y legible. En tiempo de ejecución detecta muchos errores y proporciona abundante información para detectarlos y corregirlos. Puede ser usado tanto como lenguaje imperativo procedimental o como lenguaje orientado a objetos. Posee un rico juego de estructuras de datos que se pueden manejar con facilidad. (59)

PHP: es un lenguaje de programación sencillo, de sintaxis cómoda y similar a otros lenguajes. Es rápido, interpretado, orientado a objetos y multiplataforma. Para este lenguaje se encuentran disponibles una multitud de librerías. El intérprete de PHP, diversos módulos y gran cantidad de librerías desarrolladas para PHP, son de código libre (54). Posee una alta capacidad de conexión con la mayoría de los Sistemas Gestores de bases de datos. No requiere de definición de tipos de datos para las variables.

Para el desarrollo de la propuesta de solución, teniendo en cuenta las ventajas que ofrece, que es de software libre, el gran número de librerías que hay desarrolladas, así como la bibliografía disponible, seleccionando como lenguaje de programación del lado del servidor PHP en su versión 5.5.

1.6.5 Framework del lado del cliente

jQuery: es un framework JavaScript, implementa una serie de clases (Programación Orientada a Objeto(POO)) que permite programar sin preocuparse por el navegador con que está visitando el usuario.

Ofrece una infraestructura con la que se obtendría mayor facilidad para la creación de aplicaciones complejas del lado del cliente. Es un producto serio, estable, bien documentado y con un gran equipo de desarrolladores a cargo de la mejora y actualización del framework (60). El empleo de este framework facilitó el proceso de construcción de contenidos. La versión a utilizar sería la 1.10.2.

Boostrap: mediante la combinación de CSS y JavaScript, este framework simplifica el proceso de creación de diseños web. Es un framework que posee numerosos componentes web. Además, permite un ahorro significativo de esfuerzo y tiempo debido a que facilita la creación de interfaces que se adapten a cualquier navegador. Ofrece un conjunto de plantillas CSS y ficheros JavaScritp que permiten la integración del framework de forma sencilla con las aplicaciones web. Por otra parte es capaz de integrarse con las principales librerías JavaScript, por ejemplo jQuery (61). Su uso en el desarrollo del producto simplifica el proceso de creación de las interfaces de usuario, permitiendo crear con facilidad un ambiente acogedor y agradable a la vista, además, que garantiza que las páginas se puedan mostrar en cualquier navegador web. La versión a utilizar sería la 3.0.

1.6.6 Framework del lado del servidor

Yii: es un framework de código abierto, de alto rendimiento del lenguaje PHP, basado en componentes.

Ofrece excelente integración con la mayoría de los Sistemas Gestores de Base de Datos(SGBD) haciendo uso de los patrones de diseño Data Access Objects y Active Record. Permite desarrollar aplicaciones web de manera fácil, eficiente y extensible, haciendo uso de la POO. Acelera significativamente el proceso de desarrollo de software y facilita la reutilización de código. Emplea como patrón arquitectónico el Modelo Vista Controlador(MVC). (62)

CodeIgniter: es un framework de código abierto, construido para programadores del lenguaje PHP que necesitan un conjunto de herramientas simple y elegante para crear aplicaciones web con todas las funciones. CodeIgniter es un framework que utiliza el patrón arquitectónico MVC, es orientado a objetos. (63)

Es capaz de trabajar en la mayoría de los entornos o servidores, incluso en sistemas de alojamiento compartido. Es bastante flexible y su núcleo es ligero lo que evita una sobrecarga del servidor.

Symfony: es un completo framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Fue diseñado para que se ajustara a los siguientes requisitos: fácil de instalar y configurar en la mayoría de las plataformas, independiente del sistema gestor de base de datos, sencillo de usar en la mayoría de los casos, pero flexible como para adaptarse a los casos más complejos, sigue la mayoría de buenas prácticas y patrones de diseño para la web, código fácil de leer que incluye cometarios de phpDocumentor, fácil de entender lo que permite su integración con librerías desarrolladas por terceros (64). Este framework de lenguaje PHP brinda la posibilidad de optimizar gracias a sus características el proceso de desarrollo del producto.

Para la integración del motor de evaluación AQTIEngine, teniendo en cuenta las ventajas que ofrece, su facilidad de uso y de instalación, que es de código abierto y que posee abundante bibliografía, así como de librerías desarrolladas por terceros, el equipo de trabajo seleccionó como framework de desarrollo del lado del servidor Symfony en su versión 2.7.13

1.6.7 Servidor web

Apache: es una plataforma de servidores web de código fuente. El equipo de desarrollo creó originalmente un servidor web altamente configurable, lo que le permitió alcanzar gran popularidad. En la versión 2 de este servidor, el grupo Apache se ha concentrado en la escalabilidad, en la seguridad y en el rendimiento. Entre las características de Apache que lo hacen tan popular se pueden mencionar que:

es un servidor altamente configurable de diseño modular, es una tecnología gratuita de código fuente abierto, trabaja con gran cantidad de Perl, PHP y otros lenguajes script, funciona en Linux y otros sistemas Unix, además también funciona en Windows (65). El desarrollador usaría este servidor web porque además sus características este es usado en la línea base del proyecto que desarrolla la plataforma educativa ZERA 2.0. La versión a utilizar sería la 2.4.7.

1.6.8 Entorno de desarrollo integrado

Netbeans: es un programa compuesto por un conjunto de herramientas de programación que permite elaborar aplicaciones de escritorio, para la web y para dispositivos portátiles sin que cambie la forma de programar. La programación mediante este se realiza a través de componentes de software modulares, también llamados módulos, que le aportan gran funcionalidad y versatilidad posibilitando la integración con PHP y la librería jQuery de JavaScript. Ofrece todas las funciones de los entornos de desarrollo integrado avanzados como diseño de interfaces, asistentes para la conexión con bases de datos, creación automática de propiedades y clases, etc (66).

Eclipse: dispone de un editor de texto con resaltador de sintaxis, la compilación es en tiempo real, además tiene pruebas unitarias con JUnit, posee control de versiones (CVS), posee asistentes para la creación de proyectos, clases, etcétera. Su principal inconveniente, común a otros IDEs en mayor a menor medida, es el consumo de recursos, además carece de mucho soporte para webapps (.war, jsp y servlets) tal y como lo hace Netbeans. Los plugins por lo general no son ni tan potentes ni tan sencillos como el módulo que en Netbeans viene preinstalado. (67)

Después de analizar las principales ventajas, desventajas y características principales de cada uno de estos entornos de desarrollo se llegó a la conclusión de que se va a utilizar Netbeans para el desarrollo de la solución, este en su versión 8.0.

1.6.9 Herramientas de modelado

Visual Paradigm for UML: es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Ayuda a una rápida construcción de aplicaciones con calidad y a un menor costo. Permite crear todos los tipos de diagramas de clases, ingeniería inversa, generar código desde diagramas y generar documentación (68). En el desarrollo de la integración del motor de evaluación se utiliza solo para crear todos los diagramas. La versión a utilizar sería la 8.0.

1.6.10 Conclusiones parciales

En este capítulo se realizó un estudio referente a los principales conceptos asociados al dominio del problema. Se analizaron algunos motores de evaluación para el logro de la interoperabilidad entre plataformas educativas. Por otra parte, se estudiaron las iniciativas fundamentales respecto a los estándares más utilizados, para garantizar la interoperabilidad en e-learning, donde quedó demostrado que IMS QTI es una de las especificaciones más utilizadas en este campo. Además, para la comunicación entre el motor de evaluación y la plataforma educativa se utilizaría REST para consumir de los servicios web que brinda el motor.

Se decide integrar el motor de evaluación AQTIEngine utilizando como lenguaje de programación PHP v5.5 y JavaScript v1.8, usando como framework Symfony v2.3. Además, se utilizará como servidor web Apache en su versión v2.4.7, como SGBD PostgreSQL v9.3.3 y Doctrine v2.0 como técnica para mapear objetos relacionales. Las herramientas Netbeans y Visual Paradigm, ambas en su versión 8.0 serán utilizadas como entorno de desarrollo y para el modelado respectivamente. Para garantizar la calidad del software se utilizaría PHPUnit en su versión 3.7.28 para la realización de pruebas unitarias. Todo el proceso de desarrollo del software será guiado y controlado por la metodología de desarrollo AUP_UCI.

CAPÍTULO 2: CARACTERÍSTICAS DE LA SOLUCIÓN, ANÁLISIS Y DISEÑO.

2.1 Introducción

En el presente capítulo se enuncian y describen las características del sistema a desarrollar como propuesta de solución a la problemática planteada. Se identifican los requisitos funcionales y no funcionales con los que debe cumplir el sistema a desarrollar. Se muestran los diagramas de clases del diseño, diagramas de despliegue, modelo de dominio y patrones de diseño utilizando la metodología AUP_UCI.

2.2 Características de la solución

La integración del motor AQTIEngine contribuye a la interoperabilidad de la calificaciones de los ejercicios en la plataforma educativa ZERA 2.0, pues dicho motor usa el estándar IMS-QTI, además permite que esta plataforma pueda consumir el servicio web que brinda el mismo para evaluar los ejercicios realizados por los estudiantes. Al concluir la evaluación provee de una retroalimentación a la plataforma, quién será la encargada de mostrársela al estudiante. Todo el proceso es realizado de forma transparente al usuario.

2.3 Modelo de dominio

Un modelo de dominio facilita la comprensión de los sistemas ya que permite agrupar y clasificar objetos, disminuyendo la complejidad y el número de elementos en el modelado. Además, se utiliza para describir los objetos y sus relaciones con los sistemas, facilitando una mejor comunicación entre usuarios, desarrolladores y clientes al establecer un lenguaje común para el entendimiento del mismo. (69)

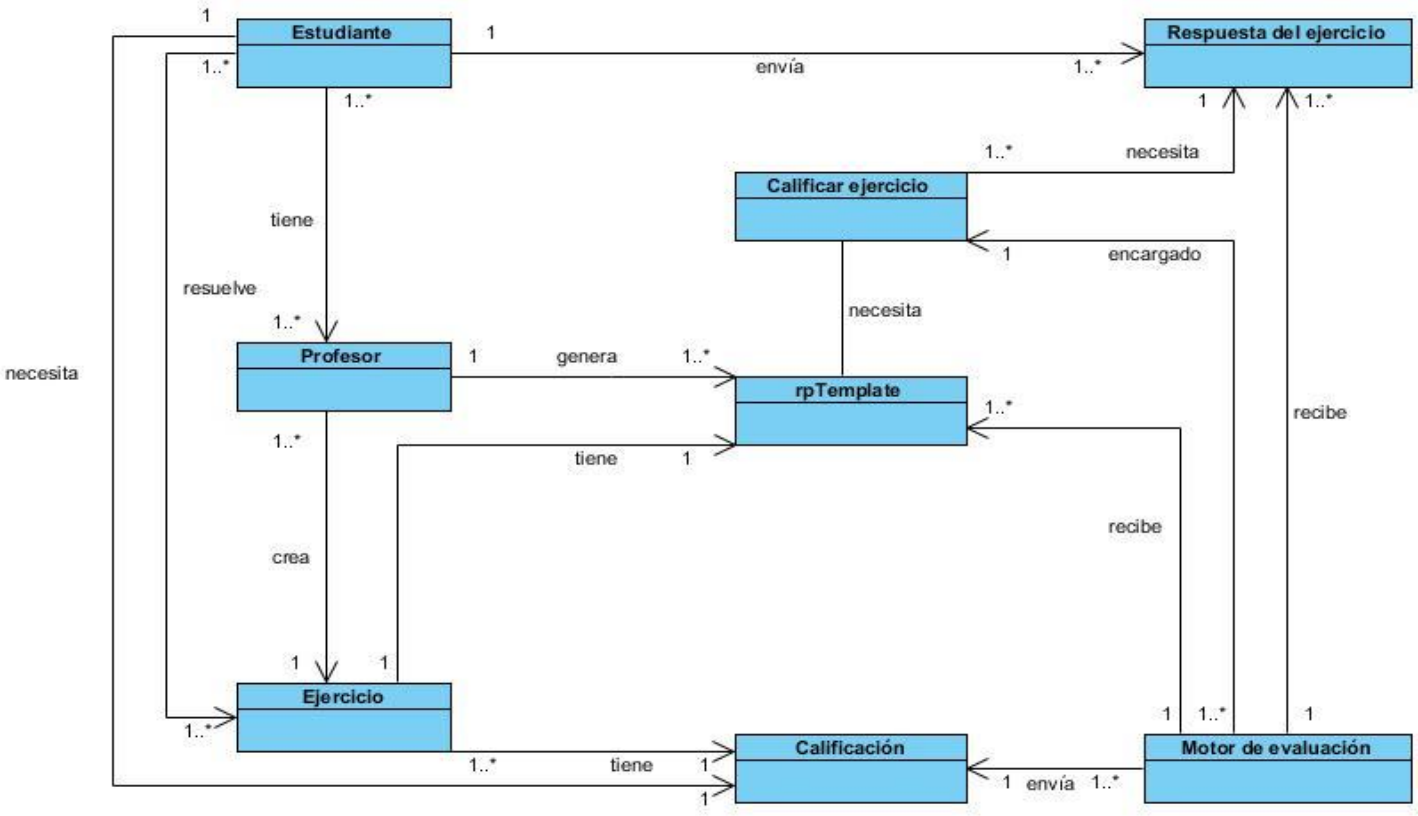


Figura 2.1: Modelo de dominio

2.3.1 Definición de las clases del Modelo de dominio.

Estudiante: es el usuario que responde el cuestionario o ejercicio.

Profesor: es el usuario que crea los ejercicios y cuestionarios.

Ejercicio: son preguntas formuladas, las cuales se generan en las plataformas educativas por un profesor.

Calificación: evaluación obtenida por el estudiante después de ser revisado el ejercicio.

Respuesta del ejercicio: respuesta del ejercicio dada por el estudiante.

Calificar el ejercicio: acción que es realizada por el motor de evaluación para poder evaluar un ejercicio que es enviado por el estudiante, haciendo uso del algoritmo de evaluación creado por el profesor.

Motor de evaluación: es utilizado por la plataforma y se encarga de generar el resultado de las pruebas, emitiendo una calificación y/o retroalimentación de cada ejercicio.

2.4 Descripción de la solución

En la actualidad existen especificaciones que regulan la forma en que se crean y se autoevalúan los ejercicios digitales en los LMS, el motor AQTIEngine usa la especificación IMS-QTI, por lo que la integración de dicho motor a la plataforma educativa ZERA 2.0 es de gran importancia para que la misma pueda ser interoperable con otros LMS. La integración consiste en crear un cliente web para que consuma los servicios que brinda la API del motor, para realizar la calificación de los ejercicios resueltos por los estudiantes. Para llevar a cabo la calificación, al motor se le proporciona la información de las respuestas de los estudiantes además de los rpTemplate de cada ejercicio, mediante el cliente web, luego el motor ejecuta su algoritmo interno, construye la calificación del estudiante y la devuelve al cliente web, después la plataforma se encarga de mostrarle las calificaciones a dicho estudiante.

La utilización de este motor en la plataforma trae consigo muchos beneficios a la misma y por ende al proceso de enseñanza y aprendizaje en la plataforma educativa ZERA 2.0, pues permite:

-Que se puedan calificar los ejercicios de forma automática en la plataforma.

-Que las calificaciones de los ejercicios que resuelven los estudiantes sea la misma en cualquier plataforma para el mismo ejercicio.

-Agilizar el proceso de calificación de los ejercicios.

-Evitar que la plataforma tenga que crear un algoritmo interno propio para la evaluación, pues este puede que no sea interoperable.

2.5 Levantamiento de requisitos.

La obtención y análisis de los requisitos constituye una de las etapas fundamentales en el proceso de ingeniería de requisitos. En esta actividad los ingenieros de software trabajan en conjunto con los clientes y usuarios finales, con el fin de determinar el dominio de la aplicación, los servicios que debe proporcionar, el rendimiento requerido para el sistema, así como las restricciones del hardware (70). Es decir, son definidos los requisitos que la solución debe cumplir tanto funcionales como no funcionales.

2.5.1 Requisitos funcionales.

Los requisitos funcionales describen lo que el sistema o software debe hacer. Una función es una capacidad útil proporcionada por uno o más componentes de un sistema. (71)

-RF 1 Calificar ejercicio

-RF 2 Recalificar Ejercicio

-RF 3 Establecer comunicación con el motor

-RF 4 Guardar calificaciones

-RF 5 Mostrar calificaciones

-RF 6 Gestionar rpTemplate

- RF 6.1 Generar rpTemplate
- RF 6.2 Guardar rpTemplate
- RF 6.3 Actualizar rpTemplate

2.5.2. Requisitos no funcionales

Los requisitos no funcionales especifican las propiedades del sistema, tales como la seguridad. (71)

Seguridad

RNF 1: Se debe garantizar la protección de accesos no autorizados.

RNF 2: Garantizar el acceso a las funcionalidades definidas para los usuarios de acuerdo a los roles que posean.

RNF 3: Mantener el sistema disponible evitando que los mecanismos de seguridad impidan el acceso a la información requerida por los usuarios autorizados.

2.6 Descripción de actores

Un actor es una entidad externa al sistema que realiza algún tipo de interacción con el mismo. Esta representación sirve tanto para actores que son personas como para otro tipo de actores (otros sistemas, sensores, etc.). (72)

Tabla 2.1: descripción de actores

Actor	Descripciones
Estudiante	Actor que interactúa con la plataforma resolviendo los ejercicios, y generando una respuesta que se le envía al motor, quien es el encargado de devolverle los resultados de las mismas.
Profesor	Actor que interactúa con la plataforma creando los ejercicios, y a la vez generando los rpTemplate asociados a cada uno de ellos.

2.7 Descripción de requisitos.

A continuación se muestra la descripción de la HU Calificar ejercicio. Para el estudio de las demás HU, remitirse a los Anexos.

Tabla 2.2: Descripción de la HU: Calificar ejercicio.

2 Descripción de Requisitos de Software

Número: 1	**Nombre del requisito:** Calificar Ejercicio
Programador: Luis Miguel González Ravelo	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 90 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** 75 días

Descripción:

1- Objetivo:

Permitir que un ejercicio pueda ser calificado a partir de la respuesta de los estudiantes.

2- Acciones para lograr el objetivo (precondiciones y datos):

Para que se pueda calificar un ejercicio :

- El estudiante tiene que tener los permisos necesarios para resolver los cuestionarios o ejercicios individuales.
- Debe existir en el sistema al menos un algún ejercicio y además esté asociado a algún cuestionario.
-El estudiante deberá responder el cuestionario y enviar sus respuestas.

3- Comportamientos válidos y no válidos (flujo central y alternos):

Debe haberse respondido el o los ejercicios del cuestionario con anterioridad.

4- Flujo de la acción a realizar:

Inicialmente el estudiante selecciona el cuestionario que quiere resolver para su posterior calificación. Una vez seleccionado y respondido el cuestionario, podrá ver las calificaciones de cada ejercicio demás de las del cuestionario en conjunto pulsando en la opción “Calificar”.

Observaciones: los cuestionarios son para agrupar varios ejercicios.

2.8 Modelo de análisis

El propósito fundamental del análisis es analizar los requisitos con mayor profundidad que los que se tienen como resultado de la captura de requisitos. Un modelo de análisis se describe utilizando el lenguaje

de los desarrolladores y puede, por tanto introducir un mayor formalismo y ser utilizado para razonar sobre los aspectos internos del sistema. Los requisitos pueden ser estructurados de manera que facilite su comprensión, su preparación, su modificación, y, en general, su mantenimiento. Se encuentra estructurado por clases y paquetes estereotipados, de manera que proporciona la estructura a la vista interna. Es utilizado fundamentalmente por los desarrolladores para comprender como debería ser diseñado e implementado el sistema. Esboza como se debería llevar a cabo la funcionalidad dentro del sistema, sirve como una primera aproximación al diseño. (73)

2.8.1. Diagramas de clases del análisis

Una clase de análisis representa una abstracción de una o varias clases y/o subsistemas del diseño del sistema. Según Jacobson (73) las clases del análisis siempre encajan en uno de los tres estereotipos básicos: clase de interfaz, de control y de entidad. A continuación se muestra el diagrama de clase del análisis correspondiente a la HU Calificar ejercicio. Para el estudio de los demás diagramas de clases del análisis, remitirse a los Anexos.

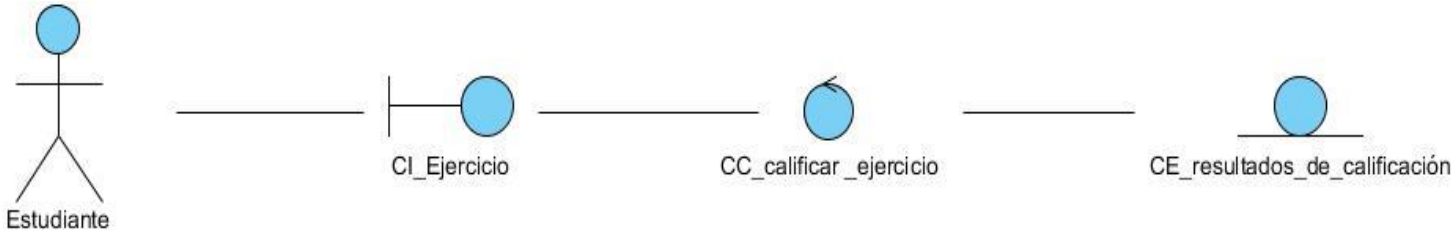


Figura 2.2: Diagrama de clases del análisis: HU Calificar ejercicio, HU Guardar calificaciones, HU Establecer comunicación con el motor.

2.8.2 Diagramas de colaboración del análisis

Los diagramas de colaboración muestran las interacciones entre objetos creando enlaces entre ellos y añadiendo mensajes a esos enlaces. El nombre de un mensaje debería denotar el propósito del objeto invocante en la interacción con el objeto invocado. (73)

A continuación se muestra el diagrama de colaboración del análisis correspondiente al CU Evaluar ejercicio. Para el estudio de los demás diagramas de colaboración del análisis, remitirse a los Anexos.

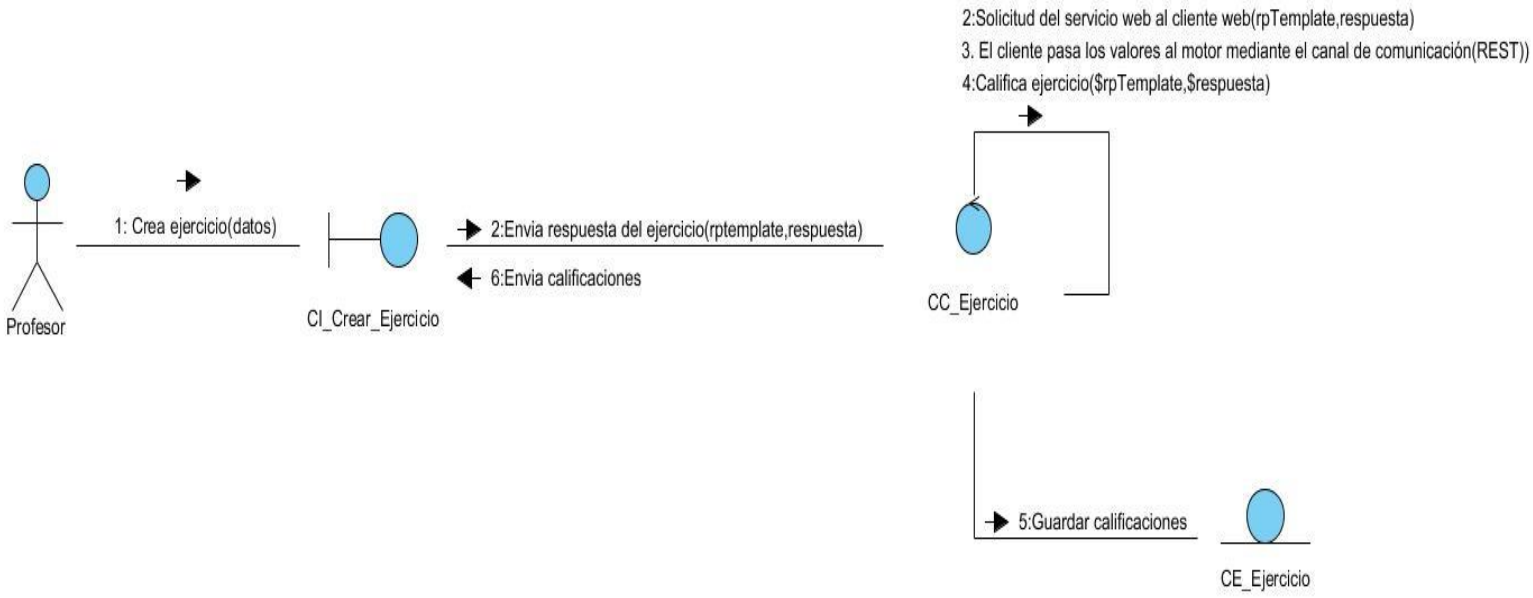


Figura 2.7: Diagrama de colaboración del análisis: HU Calificar ejercicio, HU Guardar calificaciones, HU Establecer comunicación con el motor.

2.9 Patrón arquitectónico Modelo Vista Controlador en Symfony

Symfony está basado en un patrón clásico del diseño web conocido como arquitectura MVC. El objetivo que se busca al utilizar este patrón es separar el código responsable de la representación de los datos en la pantalla, del código encargado de la ejecución de la lógica del negocio (74). Para poder lograr dicho

objetivo el patrón divide la capa de presentación en tres tipos de objetos básicos: modelos, vistas y controladores:

El Modelo: representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.

La Vista: transforma el modelo en una página web que permite al usuario interactuar con ella (64). Además, representa los datos del modelo e invoca acciones de un controlador en respuesta a las acciones de un usuario (74).

El Controlador: se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones (HTTP, consola de comandos, email, etc.). El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes. (64) En el desarrollo de la solución se usó este patrón, pues el mismo ya viene implementado en la arquitectura de Symfony 2, que fue el framework seleccionado para desarrollar la integración del motor de avaluaciones a la plataforma.

2.10 Aplicación de los patrones de diseño en Symfony.

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular. Identifica clases, instancias, roles, colaboraciones y la distribución de responsabilidades. Un framework simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. (75)

Patrones General Responsibility Asignment Software Patterns(GRASP):

Experto: se aplica en el modelo a las clases encargadas de la abstracción de datos. Estas son las clases responsables de hacer las consultas a la base de datos utilizando Doctrine, pues cuentan con los atributos necesarios para ejecutar esa función, por lo que tienen las responsabilidad de realizar directamente las acciones sobre la base de datos (76). Un ejemplo de esto se ve en las clases ExerciseManager y QuestionnaireManager.

Creador: es el encargado de identificar quién debe ser el responsable de la creación o instanciación de nuevos objetos o clases (77). Un ejemplo de la utilización de este patrón se muestra en la clase BasicEngine.

Bajo acoplamiento: se evidencia en la capa modelo. Las clases de acceso a los datos tienen bastante independencia de las clases de abstracción de datos. La poca dependencia entre las clases permite una mayor reutilización (76).

Alta cohesión: debido a la estructura de los proyectos en Symfony que facilita la organización del trabajo es posible crear y trabajar con clases con una alta cohesión. Esto hace posible que el software sea flexible a cambios sustanciales con efecto mínimo, garantizando la alta cohesión (76). El uso de este patrón se evidencia en la clase AQTIEngine.

Controlador: sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es el controlador quien recibe los datos del usuario y quien los envía a las distintas clases según el método llamado (77). Un ejemplo de la utilización de este patrón se muestra en la clase ExerciseControler.

Patrones Gang of Four (GoF):

Patrón singleton (The Singleton Pattern): asegura que una clase tiene solo una instancia, y proporciona un punto de acceso global a la misma (78). Un ejemplo de la utilización de este patrón se evidencia en la clase AQTIEngine.

Patrón la fábrica (The Factory Pattern): define una interfaz para crear un objeto, pero permite a las subclases decidir qué clase instanciar. Además, permite aplazar una instanciación de clases a las subclases (78). Un ejemplo de la utilización de este patrón se evidencia en la clase AQTIEngine.

Patrón observador (The Observer Pattern): define de una a muchas dependencias entre objetos de forma que cuando un objeto cambia de estado, todos sus dependientes son notificados y actualizadas de forma automática (78). . Un ejemplo de la utilización de este patrón se evidencia en la clase AttemptsSubscriber.

Patrón fachada (The Facade Pattern): proporciona una interfaz unificada para un conjunto de interfaces en un subsistema. Define una interfaz nivel superior que hace el subsistema más fácil de usar (78). Este ejemplo de patrón se ve en las clases EvaluatorInterfaces, EvaluationsServiceInterface y EvaluatorFormInterface.

Patrón inyección de dependencia (Dependency Injection Pattern): es donde los componentes dan sus dependencias a través de sus constructores, métodos, o directamente en los campos. Además, la mayoría de los frameworks PHP modernos utilizan inyección de dependencias para proporcionar un conjunto de componentes desacoplados pero cohesionados (79). Un ejemplo del uso de este patrón se evidencia en la clase Evaluation.

2.11 Modelo de diseño

En el diseño se modela el sistema y se encuentra la forma de que soporte todos los requisitos, incluyendo los requisitos no funcionales y cualquier otra restricción. El modelo de diseño crea un punto de partida

para las actividades de implementación subsiguientes. Permite descomponer los trabajos de implementación en partes más manejables que puedan ser llevadas a cabo por diferentes equipos de desarrollo. Da una forma al sistema mientras que intenta preservar la estructura definida por el modelo de análisis. (73)

2.11.1 Diagramas de secuencia del diseño

Los diagramas de secuencia muestran las interacciones entre objetos mediante transferencia de mensajes entre objetos o subsistemas. El nombre del mensaje debería indicar una operación del objeto que recibe la invocación o de una interfaz que el objeto proporciona. (73)

A continuación se presenta el diagrama de secuencia del diseño correspondiente a la HU Calificar ejercicio. Para el estudio de los demás diagramas de secuencia del diseño remitirse a los Anexos.

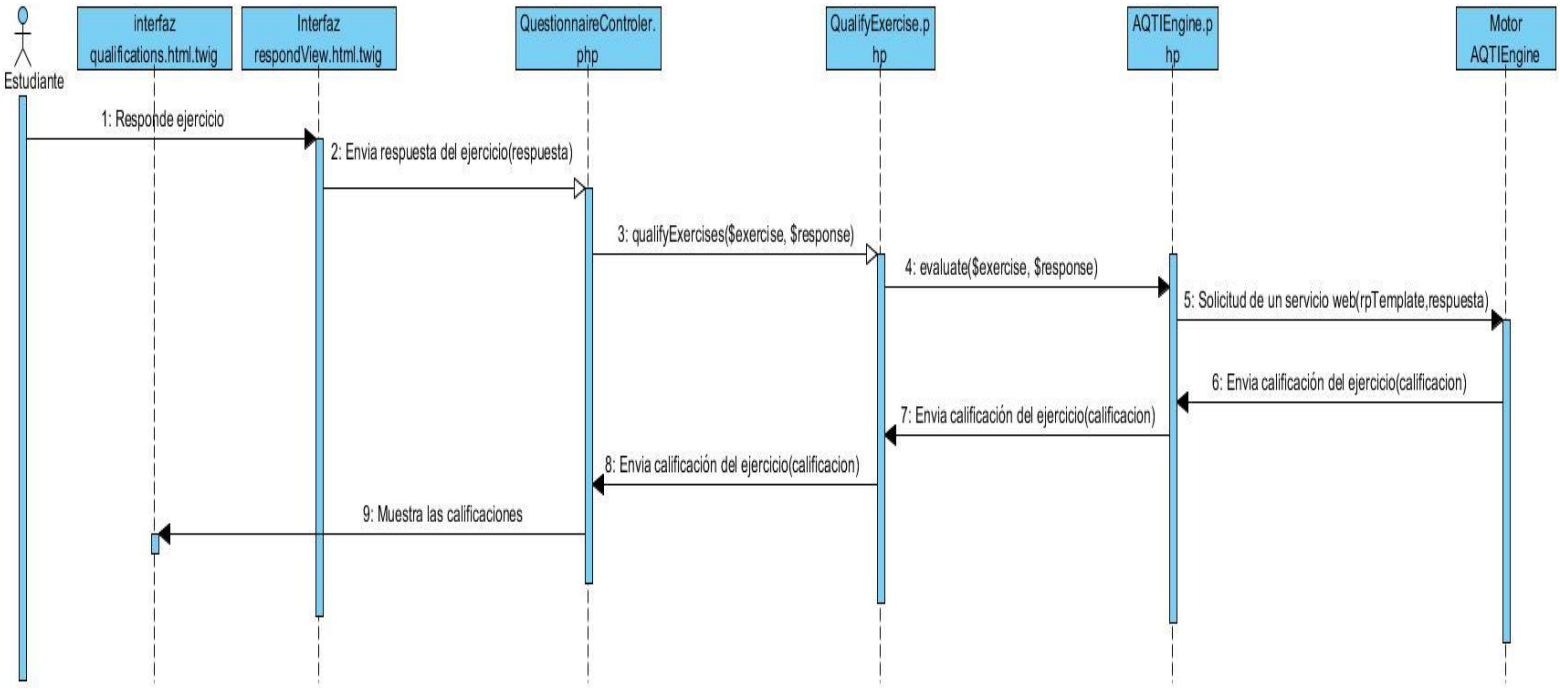


Figura 2.12: Diagrama de secuencia del diseño: HU Calificar ejercicio.

2.11.2 Diagrama de despliegue

El diagrama de despliegue es un modelo de objetos que describe la distribución física del sistema. Se utiliza como entrada principal en las actividades de diseño e implementación, debido a que la distribución del sistema tiene una influencia principal en su diseño. (73)

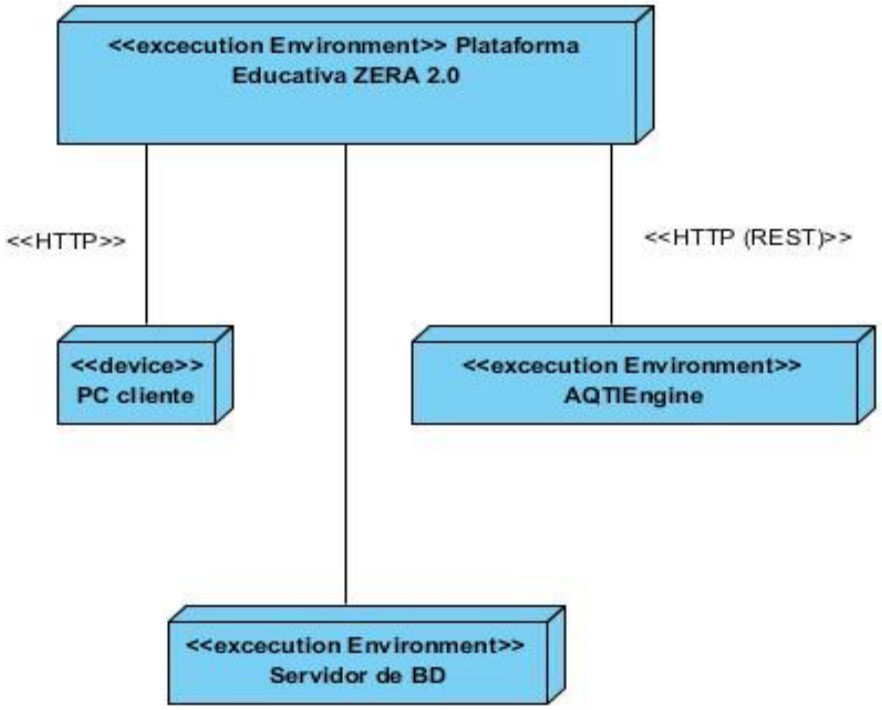


Figura 3.4: Diagrama de despliegue.

2.12 Diseño de la base de datos

El diseño de la base de datos es un proceso en el cual se deben tener en cuenta varios aspectos, el principal a tener en cuenta para la creación de las tablas de una base de datos es la definición correcta de cada uno de sus atributos. Uno de los modelos más utilizados para diseñar bases de datos es el modelo entidad-relación, ya que permite una definición clara y concisa de los esquemas conceptuales y de su

visión. Este modelo se encuentra basado en dos conceptos: las entidades, que son objetos sobre los cuales se desea guardar información y las relaciones, que constituyen asociaciones entre entidades. (80)

A continuación se presenta el diseño del modelo de base de datos propuesto para la investigación:

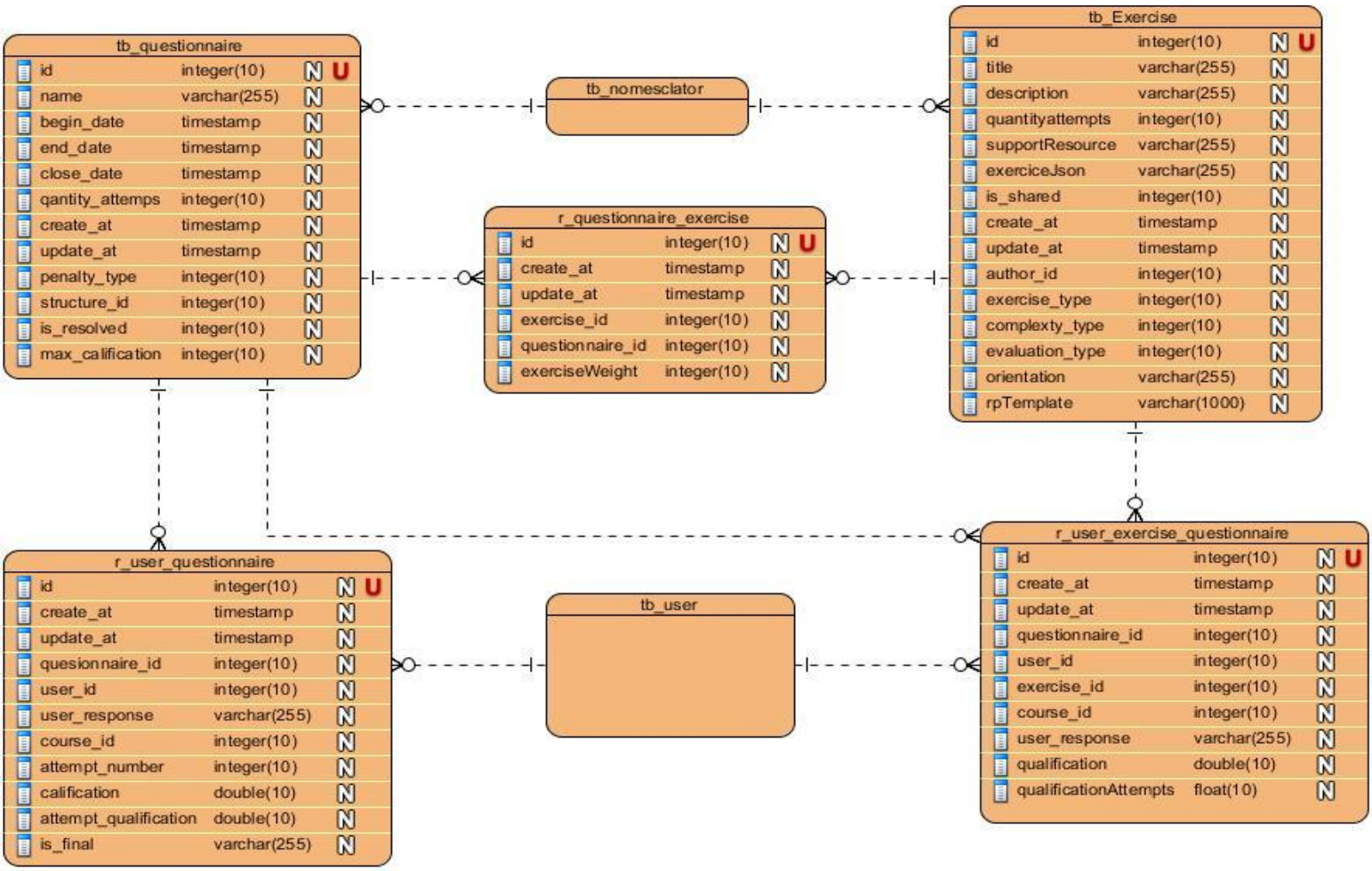


Figura 2.6: Modelo de base de datos.2.12.1 Descripción de las tablas

2.12.1 Descripción de las tablas.

En esta sección se presenta una breve descripción de cada uno de los atributos de la tabla tb_questionnaire, el resto de las descripciones se encuentran en los Anexos.

Tabla 2.14: Tabla tb_questionnaire

tb_questionnaire		
Descripción: En la siguiente tabla se agrupa la información correspondiente a cada uno de los cuestionarios existentes en el sistema.		
Atributo	Tipo	Descripción
id	integer (10)	Etiqueta única que identifica el objeto en la tabla.
name	varchar(255)	Almacena el nombre del cuestionario.
begin_date	timestamp	Almacena la fecha de inicio del cuestionario.
end_date	timestamp	Almacena la fecha final del cuestionario
close_date	timestamp	Almacena la fecha de cierre del cuestionario.
quantity_attempts	integer (10)	Almacena la cantidad de intentos que permite el cuestionario por cada usuario.
create_at	timestamp	Almacena la fecha en la que se creó el cuestionario.
update_at	timestamp	Almacena la fecha en la que se actualizó el cuestionario.
penaltity_type	integer (10)	Almacena el tipo de penalización a utilizar en las calificaciones.
is_resolved	integer (10)	Almacena si el cuestionario ya ha sido resuelto o no.
max_calification	integer (10)	Almacena la máxima calificación para el cuestionario.

2.13 Conclusiones parciales

Durante el flujo de trabajo de Análisis y Diseño propuesto por la metodología de desarrollo AUP_UCI, llevado a cabo en la integración del motor de evaluación AQTIEngine a la plataforma educativa ZERA 2.0, se analizaron como principal artefacto de entrada las HU descritas en el capítulo anterior. Como resultado se obtuvieron los diagramas correspondientes al modelo del análisis y del diseño, se define la arquitectura MVC como arquitectura a aplicar por las facilidades que proporciona, se utilizaron varios patrones para el diseño que facilitan la reutilización y mejor organización de la librearía desarrollada para dicha integración. Además, se diseñó el diagrama entidad relación que define la estructura de la base de datos.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

3.1 Introducción:

Uno de los objetivos más importante de este capítulo es implementar la integración y con ello darle cumplimiento a los requisitos funcionales, para que se realice esta actividad el desarrollador deben tener en cuenta toda la documentación reflejada anteriormente. Posteriormente se deben realizar las pruebas que permitan detectar los errores cometidos en la implementación de la herramienta, para darle solución lo más pronto posible. Finalmente, se debe validar la propuesta mediante los métodos definidos en la investigación.

3.2 Diagramas de componentes

Los diagramas de componentes se utilizan para modelar la vida estática de un sistema. Muestra las organizaciones y las dependencias entre un conjunto de componentes de software y además, organiza los subsistemas de implementación en capas. Los componentes constituyen su elemento central, un componente es el empaquetamiento físico de los elementos de un modelo, como son las clases en el modelo de diseño. (73)

A continuación se muestra el diagrama de componente correspondiente a la HU Calificar ejercicio. Para el estudio de los demás diagramas de componentes, remitirse a los Anexos.

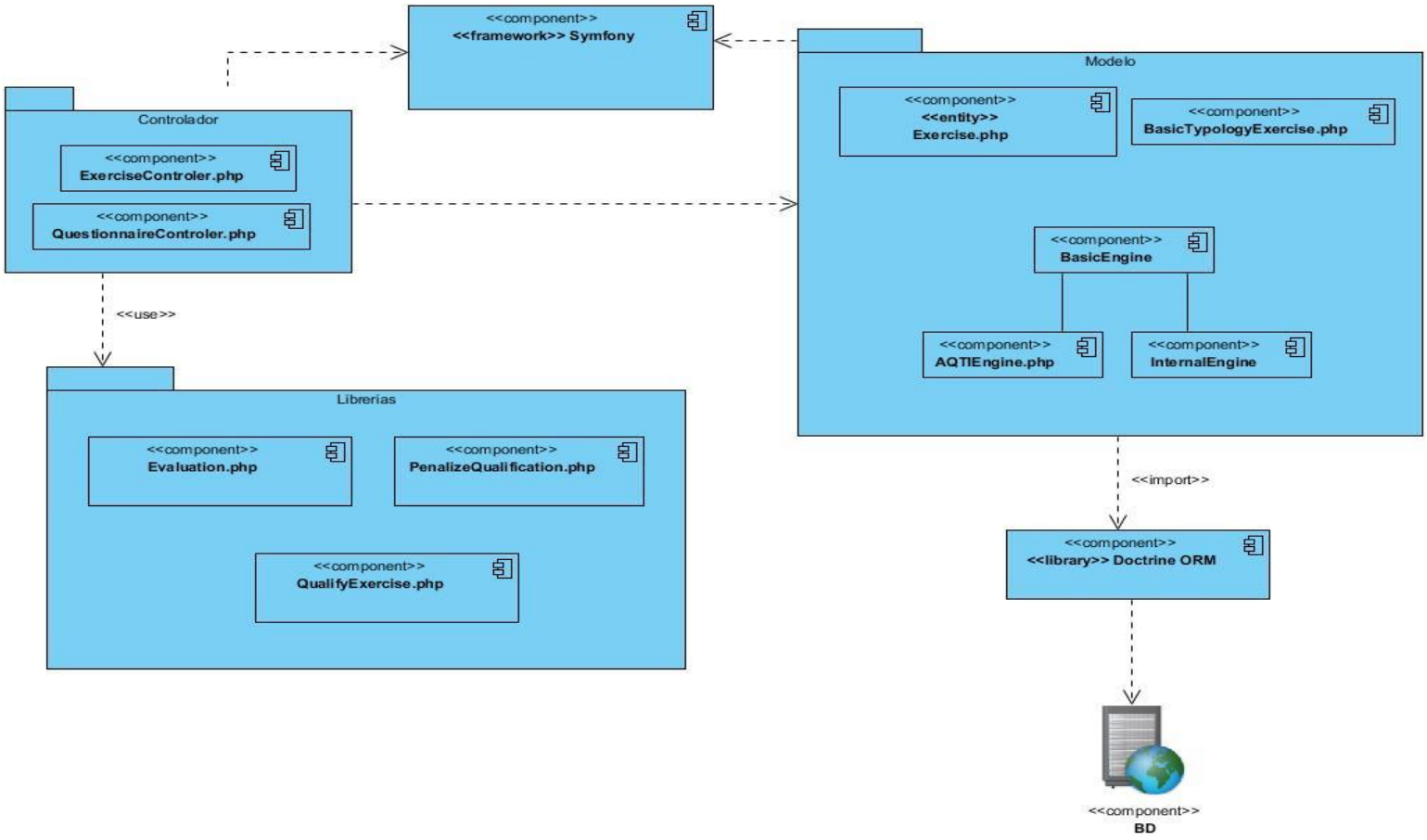


Figura 3.1: Diagrama de componente: HU Calificar ejercicio.

3.3 Modelo de Prueba

El modelo de prueba describe principalmente cómo se prueban los componentes ejecutables en el modelo de implementación con pruebas de integración y de sistema. Además, puede describir cómo han de ser privados aspectos específicos del sistema. Constituye una colección de casos de prueba, procedimientos de prueba y componentes de prueba (73).

3.3.1 Niveles de Prueba

La estrategia que se ha de seguir a la hora de evaluar dinámicamente un sistema de software debe permitir comenzar por los componentes más simples y más pequeños e ir avanzando progresivamente hasta

probar todo el software en su conjunto. Más concretamente, los pasos a seguir son: pruebas unitarias, pruebas de integración, pruebas del sistema y pruebas de aceptación.

Pruebas Unitarias: centra el proceso de verificación en la menor unidad del diseño del software: el componente de software o módulo. Se prueba la interfaz del módulo para asegurar que la información fluye de forma adecuada hacia y desde la unidad de programa que está siendo probada (81)

Pruebas de Integración: es una técnica sistemática para construir la estructura del programa mientras que, al mismo tiempo, se llevan a cabo pruebas para detectar errores asociados con la interacción. El objetivo es tomar los módulos probados mediante la prueba de unidad y construir una estructura de programa que esté de acuerdo con lo que dicta el diseño. (81)

Pruebas del Sistema: está constituida por una serie de pruebas diferentes cuyo propósito primordial es ejercitar profundamente el sistema basado en computadora. Aunque cada prueba tiene un propósito diferente, todas trabajan para verificar que se han integrado adecuadamente todos los elementos del sistema y que realizan las funciones apropiadas. (81)

Pruebas de Aceptación: son realizadas por el usuario final en lugar del responsable del desarrollo del sistema. Una prueba de aceptación puede ir desde un informal paso de prueba hasta la ejecución sistemática de una serie de pruebas bien planificadas. Además, se llevan a cabo con el objetivo de que el cliente valide todos los requisitos que debe poseer el sistema. (81)

3.3.2 Métodos de Prueba

Las técnicas de prueba proporcionan distintos criterios para generar CP que provoquen fallos en los programas. Estas técnicas se agrupan en (82):

Pruebas de caja blanca o estructural, también conocidas como pruebas estructurales o pruebas de caja transparente se basan en un minucioso examen de los detalles procedimentales del código a evaluar.

Pruebas de caja negra o funcionales, también conocidas como pruebas de caja opaca, funcionales, de entrada/salida o inducidas por los datos están especialmente indicadas en aquellos módulos que van a ser interfaz con el usuario.

3.4 Pruebas aplicadas

En este epígrafe se evidencian las pruebas aplicadas para garantizar que la solución fuese correcta y además se integrara con el sistema. A continuación se muestran cuales fueron.

3.4.1 Pruebas Unitarias

Para la validación de la integración del motor AQTIEngine a la plataforma educativa ZERA 2.0, se realizaron pruebas unitarias las cuales se ejecutaron a través del comando (phpunit -c app/) haciendo uso de la librería PHPUnit, estas fueron ejecutadas por el desarrollador a medida que se fue implementando la solución, por lo que las deficiencias detectadas se fueron corrigiendo durante la ejecución de las mismas.

3.4.2 Pruebas de integración

Después de haber realizado las pruebas unitarias, obteniendo resultados satisfactorios, podemos asegurar que todas las funcionalidades probadas se ejecutan correctamente de manera individual, dado esto, se pasa el siguiente nivel, que serían las pruebas de integración, donde se verificó la correcta interacción de la plataforma educativa ZERA 2.0 con la solución propuesta en la presente investigación. Para ello se realizaron las pruebas de integración, las mismas se harán mediante casos de prueba que garanticen que el flujo del sistema no se vea afectado por la incorporación del motor y de las funcionalidades antes mencionadas.

3.4.2.1 Casos de Prueba

Los casos de prueba utilizados fueron diseñados por integrantes del proyecto que desarrolla la plataforma educativa ZERA 2.0, pues al no tener la implementación de la integración una interfaz visual, solo se verificará que el flujo central del sistema siga siendo correcto después de agregar las nuevas

funcionalidades. A continuación se explicara el flujo Calificar cuestionario, para los demás casos remitirse a los anexos.

Tabla 3.1 Caso de Prueba Calificar cuestionario

Diseño de Casos de Prueba: HU Calificar Cuestionario			
Descripción General: El sistema deberá calificar un cuestionario una vez que el estudiante concluya el mismo. **Condiciones de ejecución:** El cuestionario debe estar resuelto y la cantidad de intentos disponibles no debe haberse agotado. **SC 1 Calificar cuestionario**			
Escenario	**Descripción**	**Respuestas de sistema**	**Flujo Central**
EC 1.1 Opción calificar cuestionario	Selecciona la acción de calificar cuestionario.	El sistema califica automáticamente las respuestas dadas a los cuestionarios, teniendo en cuenta la ponderación realizada de los ejercicios que lo componen y las penalizaciones definidas para los mismos. Permite realizar: - Nuevo intento(Siempre que no se hayan agotados la cantidad disponible) - Finalizar	Página principal/Todos los cursos/clic en el nombre del curso/Acceder/clic en el nombre del cuestionario/Resuelve el cuestionario/Calificar

Una vez ejecutado el flujo que muestra el caso de prueba anterior y de forma transparente al usuario intervienen algunas de las funcionalidades añadidas en el transcurso del desarrollo de la solución, entre ellas están:

- isRunning() de la clase AQTIEngine
- evaluate($rpTemplate, $userResponse) de la clase AQTIEngine
- evaluateAction($rpTemplate, $responseJSON) de la clase Evaluation
- viewQualificationAction($course_id, $questionnaire_id) de la clase QuestionnaireController
- qualifyAction($id) de la clase QuestionnaireController
- calculateCalification(Questionary $questionary, array $responseUser, $course) de la clase QuestionnaireController

3.4.3 Resultados obtenidos

La realización de las pruebas unitarias utilizando la librería PHPUnit, devolvió un resumen que contiene la cantidad de tests ejecutados y el número de aserciones. La siguiente imagen muestra los resultados obtenidos. Para el estudio del resto de las iteraciones remitirse a los anexos.

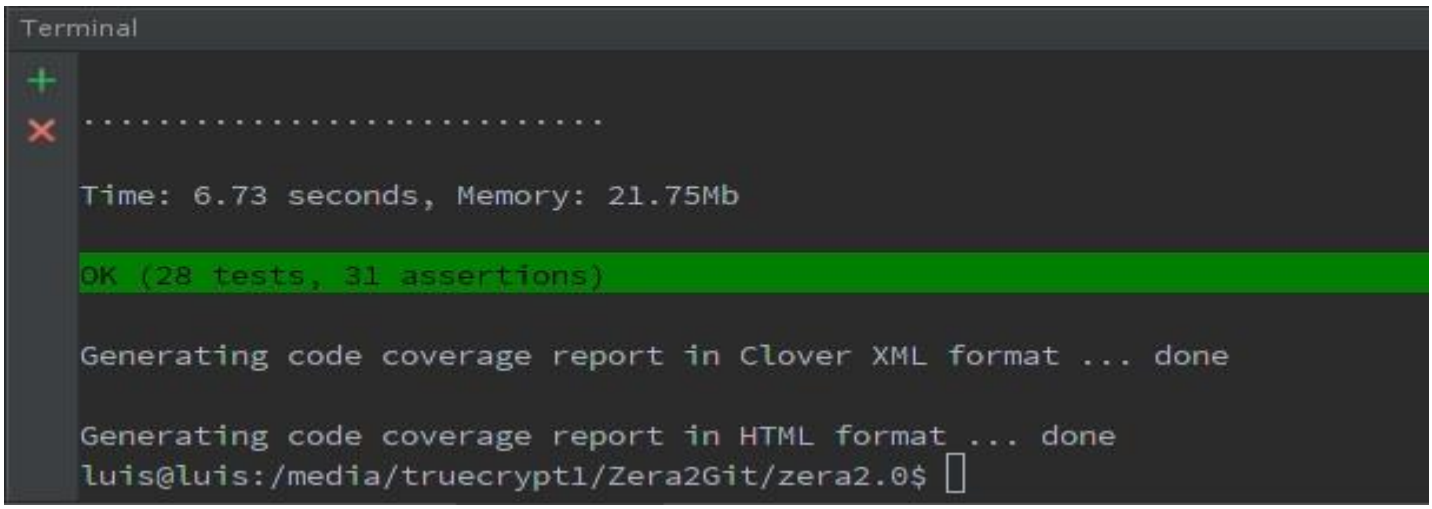
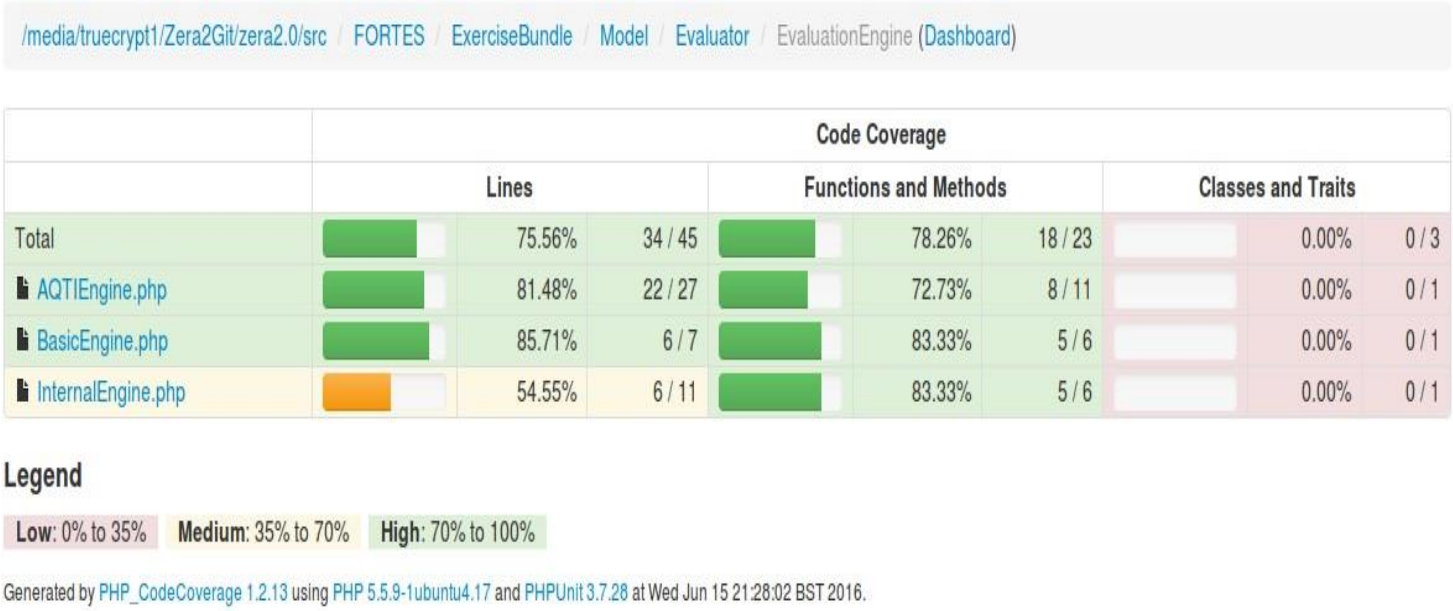


Figura 3.6: Resultados de pruebas unitarias

Además se generó un reporte donde se muestra la cantidad de líneas y métodos que se ejecutaron en cada prueba.



/media/truecrypt1/Zera2Git/zera2.0/src / FORTES / ExerciseBundle / Model / Evaluator / EvaluationEngine (Dashboard)

	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
Total		75.56%	34 / 45		78.26%	18 / 23		0.00%	0 / 3
AQTIEngine.php		81.48%	22 / 27		72.73%	8 / 11		0.00%	0 / 1
BasicEngine.php		85.71%	6 / 7		83.33%	5 / 6		0.00%	0 / 1
InternalEngine.php		54.55%	6 / 11		83.33%	5 / 6		0.00%	0 / 1

Legend

Low: 0% to 35% Medium: 35% to 70% High: 70% to 100%

Generated by PHP_CodeCoverage 1.2.13 using PHP 5.5.9-1ubuntu4.17 and PHPUnit 3.7.28 at Wed Jun 15 21:28:02 BST 2016.

Figura 3.7: Códigos y métodos ejecutados.

La realización de las pruebas de integración permitió verificar que todas las funcionalidades agregadas al sistema se integraran con el mismo, permitiendo que los flujos de las acciones del usuario ya sea como profesor o estudiante no se vieran afectadas y tuvieran un correcto funcionamiento. A continuación se presentan las imágenes donde se puede observar que el flujo se ejecutó satisfactoriamente obteniendo una correcta y esperada respuesta del sistema.

En la siguiente imagen se muestra que al añadir ejercicio este se añade satisfactoriamente y sin errores. Para los demás casos remitirse a los anexos.

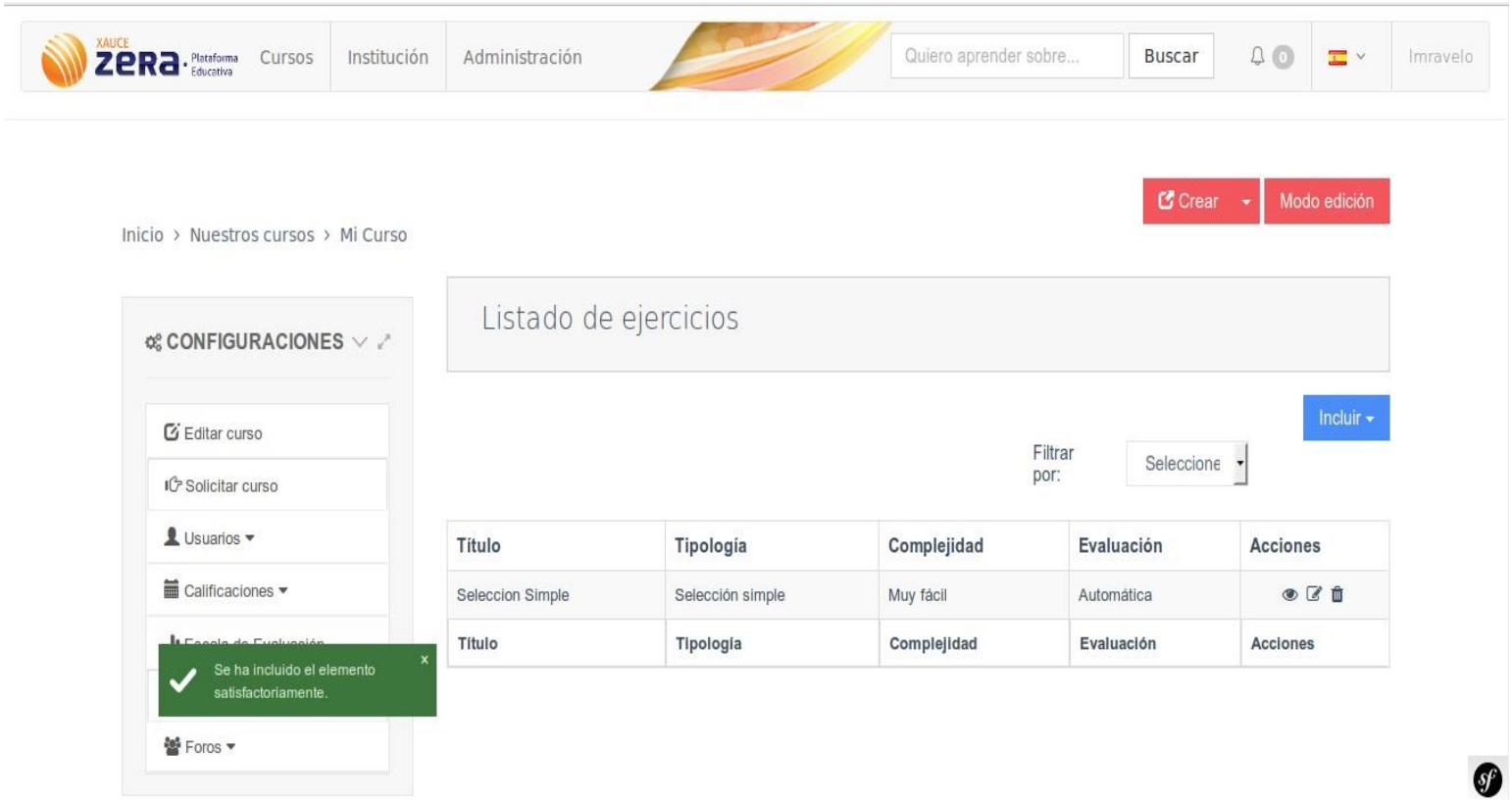


Figura 3.8 Prueba para el flujo incluir ejercicio

3.5 Conclusiones parciales

Mediante la construcción del diagrama de componente se incrementó el entendimiento de la descomposición física de la solución, en cuanto a construcción y funcionamiento. El proceso de implementación respondió a las necesidades de la plataforma lo cual fue comprobado mediante las pruebas unitarias y de integración realizadas a dicha solución.

CONCLUSIONES GENERALES

El presente trabajo ha dado cumplimiento a los objetivos trazados de la investigación científica, obteniéndose:

- El estudio del estado de arte arrojó un punto de partida a la solución definiendo características y métodos a utilizar.
- El análisis y diseño de la solución propuesta utilizando la metodología AUP_UCI generó documentación detallada para futuros cambios y centró al desarrollador en los requerimientos de esta.
- Se integró el motor AQTIEngine a la plataforma educativa ZERA 2.0 permitiendo realizar las calificaciones automáticas de los ejercicios y que además dichas calificaciones sean interoperables.
- Se desarrolló un punto de extensión el cual permite que la plataforma pueda crecer, en cuanto a formas de calificar los ejercicios.
- Se comprobó el funcionamiento del motor en la plataforma ya sea de forma externa o de interna a ella arrojando resultados satisfactorios, pues mostró su correcto funcionamiento al aplicarle las pruebas unitarias y de integración.

RECOMENDACIONES

Concluida la investigación y con el objetivo de perfeccionar y ampliar la solución, además de lograr una mayor explotación de las potencialidades que brinda el motor AQTIEngine se recomienda:

-Hacer uso de la retroalimentación en los ejercicios de la plataforma educativa ZERA 2.0.

-Implementar en la plataforma ZERA 2.0 otros tipos de interacciones de ejercicios que define la especificación IMS QTI v2.0 e integrarlas con el motor.

REFERENCIAS BIBLIOGRÁFICAS

1. **González, M. V.** Evaluación de la reacción de alumnos y docentes en un modelo mixto de aprendizaje para educación superior. 2007, págs. 83-103.

2. **Galán, B. M. y Mateos, D. R.** *La evaluación de la formación universitaria semipresencial y en línea en el contexto del EES mediante el uso de los informes de actividad de la plataforma Moodle. Revista Iberoamericana de Educación a Distancia.* 2012.

3. **González, R. C. n.** Repositorio de Recursos educativos para las instituciones de educación superior. Doctor, Universidad de las Ciencias Informáticas. 2012.

4. **de la Fuente, G. B.** *Modelo de repositorio institucional de contenido educativo (RICE): la gestión de materiales digitales de docencia y aprendizaje en la biblioteca universitaria.Doctoral, Universidad Carlos III.* 2010.

5. **Rodríguez, N. F.** *Fundamentos del proceso educativo a distancia : enseñanza, aprendizaje y evaluación.* 2014.

6. **[En línea] [Citado el: 24 de Enero de 2016.] http://e-aprendizaje.es/2012/11/26/mooc-y-sostenibilidad-segun-stephen-downes/.**

7. Mcauley, A., y otros. The MOOC model for digital practice. . 2010.

8. Martínez, G. C. y Rothenberg, E. ***Arquitecturas orientadas al servicio en el ámbito de las redes de la próxima generación.*** **2008.**

9. Agüero, D. N y Velazco, C. C. Interoperabilidad de componentes y servicios. ***Serie Científica de la Universidad de las Ciencias Informáticas.*** **3. 2011, Vol. 4, págs. 1-18.**

10. ***Plataformas de enseñanza virtual para entornos educativos. Revista de Medios y Educación.*** **Rodríguez, J. S. 2009, págs. 217-233.**

11. Guzmán, C. L. ***Los Repositorios de Objetos de Aprendizaje como soporte a un entorno e-learning. Doctorado, Universidad de Salamanca.*** **2005.**

12. Martínez Ortiz, I., y otros. e-QTI: A Reusable Assessment Engine. 2006, págs. 1-12.

13. Arroyo, D. M., y otros. Assessment in 3d virtual worlds: qti in wonderland. 2010, págs. 410-417.

14. Wills, G., y otros. ***Rendering and Reponses Processing for QTIv2 Question Types.*** **2006.**

15. Sanz Santamaría, S., Zorita, J. A. V. y Serrano, J. G. Una Revisión de Herramientas Asistidas por Ordenador para la Evaluación del Conocimiento. IEEE-RITA. 2008, 3, págs. 77-86.

16. Goguadze, G., Mavrikis, M. y González, A. ***Interoperability Issues between Markup formats for Mathematical Exercises.*** **2006.**

17. Rojas, A. E. C. y Bauta, N. L. ***Motor de Evaluación AQTIEngine.*** **2015.**

18. Baelo, R. A. El e-learning, una respuesta educativa a las demandas de las sociedades del siglo xxi. Revista de Medios y Educación. 2009, págs. 87-96.

19. Sangra, A. ***Hacia una definición inclusiva del e-learning. Barcelona.*** **2011.**

20. Moreira, M. A. y Segura, J. A. e-Learning: Enseñar y Aprender en Espacios Virtuales. 2009, págs. 1-29.

21. Cesteros, A. F.-P. ***Las plataformas e-learning para la enseñanza y el aprendizaje.*** **2011.**

22. Zapata, M. Sistemas de gestión del aprendizaje-Plataformas de teleformación. 2003, págs. 1-48.

23. In. Internet. Desarrollo web. [En línea] 2015. [Citado el: 10 de 12 de 2015.] http://desarrolloweb.dlsi.ua.es/cursos/2012/que-son-los-moocs/preguntas-respuestas#termino-mooc.

24. In. TIMES, F. Definition of mooc. 2014.

25. Coursera. [En línea] [Citado el: 18 de Febrero de 2016.] https://www.coursera.org/about/partners.

26. BrooKings. [En línea] [Citado el: 18 de Febrero de 2016.] http://www.brookings.edu/blogs/techtank/posts/2015/02/23-mooc-google-coursera-butler.

27. Miríada X. [En línea] [Citado el: 19 de Febrero de 2016.] https://miriadax.net/home.

28. Miríada X. [En línea] [Citado el: 19 de Febrero de 2016.] http://web.archive.org/web/20150905235643/https://www.miriadax.net/.

29. http://robots.stanford.edu/. [En línea] [Citado el: 19 de Febrero de 2016.] http://robots.stanford.edu/.

30. Innovation on NBCNEWS.com. [En línea] [Citado el: 19 de Febrero de 2016.] http://www.nbcnews.com/id/46138856/ns/technology_and_science-innovation/.

31. Udacity. [En línea] [Citado el: 19 de Febrero de 2016.] https://www.udacity.com/courses/all.

32. THE WALL SREET JOURNAL. [En línea] [Citado el: 19 de Febrero de 2016.] http://blogs.wsj.com/digits/2012/10/25/startup-udacity-builds-bankroll-for-online-learning/.

33. Landeta, A. E., y otros. *Evolución y retos de la educación Virtual Construyendo el e-learning del siglo XXI.* 2011.

34. RAI. Real Academia de Ingeniería. [En línea] 2015. [Citado el: 20 de 11 de 2015.] http://diccionario.raing.es/es/lema/interoperabilidad.

35. Poler, R. *La interoperabilidad de software y aplicaciones en redes de empresas.* 2015.

36. Tamayo, M., M, A., y otros. Interoperabilidad de sistemas de organización del conocimiento: el estado del arte. Información, cultura y sociedad. 2011, 24, págs. 15-37.

37. García, V. M. A., y otros. *Presente y futuro del desarrollo de plataformas Web de elearning en educación superior.* 2015.

38. Barbero, P. C. Plataformas de integración. servicios web basados en rest y soap. 2015, págs. 1-23.

39. Carrera, D. R. Proyecto de evaluación de plataformas de teleformación para su implantación en el ámbito universitario. 2003, págs. 0-124.

40. González, J. R. H. y Marín, R. H. *Estándares de e-learning: guía de consulta.* 2010.

41. Guzmán, C. L. y Peñalvo, F. J. G. Estándares y Especificaciones para los Entornos e-learning : Convergencia en Contenidos y Sistemas. 2010.

42. Ministerio de Educación y Ciencia. Educacion. [En línea] 2006. [Citado el: 06 de Junio de 2016.] http://ares.cnice.mec.es/informes/16/contenido/31.htm.

43. Interoperability, T.;Simple, O. IMS Question & Test Interoperability 1. 2014, 4, págs. 61-91.

44. Ortega, R. A. M. *Quiz: una funcionalidad para Aqurate.* 2008.

45. Ros, S. M. n. *Sistemas de Elearning abiertos basados en servicios.* 2012.

46. Ponce, V. M. *Plataformas virtuales y herramientas informáticas evaluativas con sentido formativo: alcances y limitaciones.* 2013.

47. Figueroa, R. G., Cabrera, A. A. y and Solís, C. J. Metodologías tradicionales vs. metodologías ágiles. 2015, págs. 1-9.

48. Corporation, M. *MSF Process Model v. 3.1.* 2002.

49. Palacio, J. ***El modelo Scrum.*** **2006.**

50. Letelier, P. and Penad, C. ***Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP).*** **2006.**

51. Rodriguez D, A. ***Descripción de la Metodología de Desarrollo de Software Agile Unified Process (AUP).*** **2011.**

52. UCI. ***Metodología de desarrollo para la Actividad productiva de la UCI.*** **[pdf] 2015.**

53. Schmuller, J. Intercambios virtuales. ***Aprendiendo UML en 24 horas.*** **. [En línea] 2001. [Citado el: 25 de 3 de 2016.] http://www.intercambiosvirtuales.org/libros-manuales/aprendiendo-uml-en-24-horas-joseph..**

54. Mateu, C. ***Desarrollo de aplicaciones web.*** **2004.**

55. Pérez, J. E. ***Introducción a CSS.*** **2008.**

56. Jav. ***JavaScript, manual práctico de informática.*** **2014.**

57. Sturm, J. ***Desarrollo de soluciones XML.*** **2001.**

58. Schildt, H. ***Manual de referencia Java2. 4ta edition.*** **2001.**

59. Marzal, A. y Garcia, I. ***Introduccion a la programacion con Python.*** **2003.**

60. Alvarez, M. A. ***Manual de jQuery.*** **2009.**

61. GENBETA. GENBETA desarrollo y software. [En línea] 2015. [Citado el: 12 de 3 de 2016.] http://www.genbetadev.com/frameworks/bootstrap.

62. Winesett, J. ***Agile Web Application Development with Yii 1.1 and PHP5. Packt Publishing.*** **2010.**

63. Argudo, J. ***CodeIgniter 1.7. Packt Publishing.*** **2009.**

64. Potencier, F. y Zaninotto, F. ***Symfony la guía definitiva.*** **2007.**

65. Mohammed, J. K. ***La biblia, Servidor Apache 2.*** **2012.**

66. Net. Netbeans. [En línea] 2011. [Citado el: 15 de 3 de 2016.] http://netbeans.org/features/ide/index.htm..

67. Prezi. [En línea] [Citado el: 08 de Marzo de 2016.] https://prezi.com/8_lkuuyre_nm/conceptos-caracteristicas-ventajas-y-desventajas-de-los-i/.

68. Vis. Visual Paradigm. [En línea] 2015. [Citado el: 25 de 11 de 2015.] http://www.visual-paradigm.com.

69. Eriksson, H.-E. and Penker, M. ***Business Modeling with UML: Business Patterns at Work.*** **2000.**

70. Sommerville, I. ***Ingeniería del Software.*** **2005.**

71. Young, R. ***The Requirements Engineering Handbook.*** **2004.**

72. Ferré, X. and Sánchez, M. ***Desarrollo Orientado a Objetos con UML.*** **2015.**

73. Jacobson, I., Booch, G. y Rumbaugh, J. ***El proceso unificado de desarrollo de software.*** **2000.**

74. de la Torre, c., y otros. ***Guía de Arquitectura N-capas orientada al dominio con .NET 4.0.*** **2010.**

75. Symfony. Symfony.es. [En línea] 2009. [Citado el: 25 de 3 de 2016.] http://symfony.es/.

76. Santiesteban, I. I. P., y otros. ***Desarrollo de funcionalidades que faciliten al docente su preparación y el control del aprendizaje de los estudiantes en la plataforma educativa Zera.*** **2011.**

77. García, J. C. ***SOLID y GRASP. Buenas prácticas hacia el éxito en el desarrollo de software.*** **2012.**

78. Freeman, E. y Freeman, E. ***Head First Design Patterns.*** **2004.**

79. Potencier, F. and Zaninotto, F. ***Symfony la guía definitiva.*** **2007.**

80. Castro, F. L. ***Modelo de datos, conceptos y clasificación.*** **2000.**

81. Ruiz, R. T. ***Las Pruebas de Software y su Importancia en las Organizaciones.*** **2010.**

82. Caisa, C. J. G. y Semblantes, L. V. C. ***Implementación de Pruebas Caja Negra y Caja Blanca aplicables al Sistema Escolástico del Colegio Nacional.*** **2010.**

83. edu. EducarChile. [En línea] 2013. [Citado el: 5 de 11 de 2015.] http://www.educarchile.cl/ech/pro/app/detalle?ID=217565.

84. Álvarez, M. A. ***Manual de jQuery.*** **2009.**

85. Beltrán, A. G., y otros. ***La autoevaluación como actividad docente en entornos virtuales de aprendizaje/enseñanza. Revista de Educación a Distancia.*** **2014.**

86. Casanova, M. A. ***La evaluación educativa.*** **1998.**

87. Centeno, G. M. y Cubo, S. D. ***Evaluación de la competencia digital y las actitudes hacia las TIC del alumnado universitario.*** **2013.**

88. Cubero, S. ***Manual de WebCT para el estudiante.*** **2015.**

89. Fernández, B. M. ***Especificaciones y estándares en e-learning. Revista de Tecnologías de la Información y Comunicación Eductivas.*** **2006.**

90. Fernández, B. O. y Bazan, C. M. ***Plataforma Ilias como herramienta para docencia. Utilización en la escuela politécnica superior de Jaén.*** **2015.**

91. Gonzalo, H. Sistemas de exámenes educativos por Internet. 2006.

92. Larman, C. ***UML y Patrones: Una introducción al análisis y diseño orientado a objetos y al proceso unificado.*** **2003.**

93. Jansch, I. ***php\|architect's Guide to Enterprise PHP Development.*** **2008.**

94. Lizárraga, R. E., Colado, A. Z. y Garzón, J. F. P. ***Análisis Comparativo de las Plataformas Educativas Virtuales Moodle y Dokeos. Revista Iberoamericana para la Investigación y el Desarrollo Educativo.*** **2013.**

95. Manjón, B. F., y otros. ***Uso de estándares aplicados a Tic en educación. Mariano Segura Escobar, España, secretaría edition.*** **2011.**

96. Placeres, D. R. y Cabrera, J. E. M. La evaluación como categoría reguladora del proceso enseñanza-aprendizaje: una experiencia desde la asignatura de Sistemas Operativos en la Universidad de las Ciencias Informáticas. Serie Científica de la Universidad de las Ciencias Informáticas. 2013, 2, págs. 1-9.

97. Peñalosa, E. C. ***Evaluación de los aprendizajes y estudio de la interactividad en entornos en línea : un modelo para la investigación.*** **2010.**

98. Navarro, R. (2007). ***REST vs Web Services.*** **2007.**

99. Mohammed, J. K. ***La biblia, Servidor Apache 2.*** **2012.**

100. Quesada, R. C. ***Evaluación del aprendizaje en la educación a distancia en línea.*** **2006.**

101. Ridao, M., Doorn, J. y Sampaio, J. ***Uso de Patrones en la Construcción de Escenarios.*** **2000.**

102. Ruiz, S., Piloto, Y. y Roselló, R. ***El peligro de un Caso de Uso muy largo. Mitos y realidades.*** **2015.**

103. Santiuste, V. B. y Arranz, M. L. ***Nuevas perspectivas en el concepto de evaluación.*** **2009.**

104. Suehring, S. ***MySQL Bible. Wiley Publishing, Inc., New York.*** **2002.**

105. Team, D. P. ***Doctrine 2 ORM Documentation.***

106. Tejada, M. d. M. R. La plataforma ilias como apoyo a la docencia presencial en ingeniería técnica industrial. Revista Electrónica Actualidades Investigativas en Educación". 2010, 10, págs. 1-21.

107. Vázquez, C., y otros. ***El proceso de retroalimentación en la evaluación. Un aporte al aprendizaje significativo de los estudiantes universitarios.*** **2010.**

108. Mcauley, A., y otros. The MOOC model for digital practice. . 2010.

109. Fowler, M. La Nueva Metodología. [En línea] 11 de 01 de 2010. [Citado el: 8 de 2 de 2016.] http://www.programacionextrema.org/articulos/newMethodology.es.html .

110. symfony. ***wdwdw.*** **2334.**

111. Brink, B y Lautenbach. ***La evaluación en el diseño pedagogicoy desarrollo de MOOC.*** **2011.**

112. Moore, M. G. ***La evaluación en el diseño pedagógico y desarrollo de mooc.*** **1989.**

113. Garrison y Anderson. ***La evaluación en el diseño pedagógico y desarrollo de MOOC.*** **2005.**

114. Gallego y Gutiérrez. ***La evaluación en el diseño pedagógico y desarrollo de MOOC.*** **2011.**

115. Herran. ***La evaluación en el diseño pedagógico y desarrollo de MOOC.*** **2014.**

116. de laHerrán, A. ***La evaluación en el diseño pedagógico y desarrollo de MOOC.*** **2014.**

117. Fowler, M. 2003.

118. Swan. ***La evaluación en el diseño pedagógico y desarrollo de MOOC.*** **2001.**

119. Community of Inquiry. ***La evaluación en el diseño pedagógico y desarrollo de MOOC.*** **2011.**

120. Sangra, A. *Hacia una definicion inclusiva del e-learning.* Barcelona : s.n., 2011.

121. EdSurge. [En línea] [Citado el: 2015 de Diciembre de 8.] https://www.edsurge.com/news/2015-09-08-udacity-coursera-and-edx-now-claim-over-24-million-students.

122. Coursera. [En línea] [Citado el: 18 de Febrero de 2016.] https://blog.coursera.org/post/44227940791/bienvenidos-a-las-nuevas-universidades-de-latinoamerica.