Facultad 4

Módulo para contribuir a la interoperabilidad de los ejercicios de la plataforma ZERA

Trabajo de Diploma para optar por el Título de Ingeniero en Ciencias Informáticas

Autor:

Daimel Noda Cabrera

Tutores:

Ing. Yirka Céspedes Boch

Ing. Tania Rodríguez Valera

Ing. Arley Enrique Cera Rojas

La Habana, 21 de junio de 2017

"Año 59 de la Revolución"

Declaración de autoría

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas (UCI) a que haga el uso que estime pertinente con el mismo.

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

Autor:

_____

Daimel Noda Cabrera

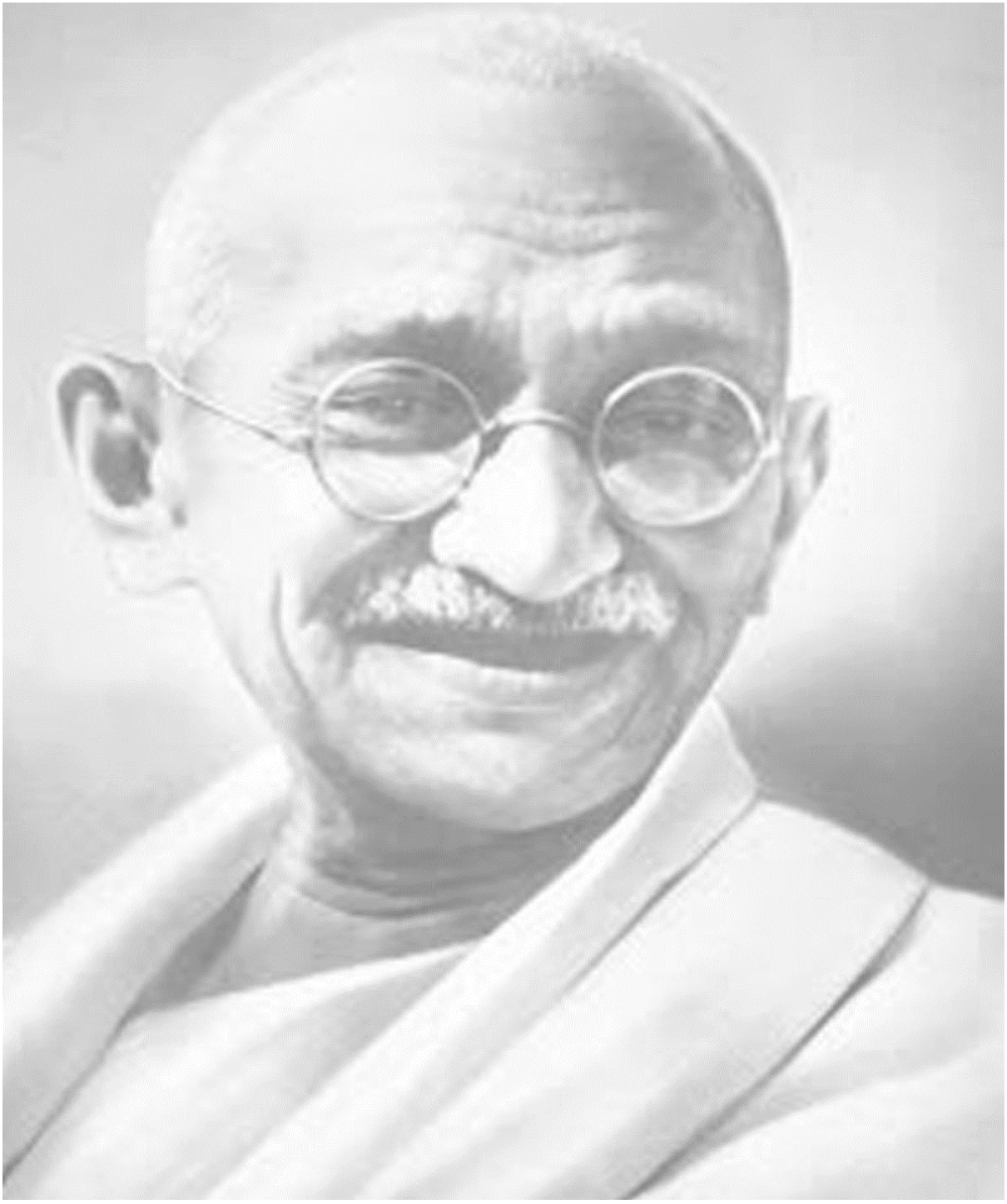
Tutor(es):

_____ _____

Ing. Arley Enrique Cera Rojas Ing. Yirka Céspedes Boch

_____

Ing. Tania Rodríguez Valera



Nuestra recompensa se encuentra en el esfuerzo y no en el resultado. Un esfuerzo total es una victoria completa.

Mahatma Gandhi

Dedico el presente trabajo:

A mis padres: Tere y Tony: Porque gracias a ellos es que todo esto hoy es posible, por su dedicación y entrega porque nunca se rindieron conmigo por muy duro que fuera el camino.

A mis abuelos Hortensia y Lorenzo: por todo su amor y cariño y que gracias a ellos tuve la fuerza para poder seguir adelante con todas las dificultades que se me presentaron en la vida y sobre todo en esta universidad.

A mi hermano Osniel: por siempre estar pendiente de como yo estaba y darme su apoyo.

A mi tío Alexis: por ser esa persona siempre incondicional y ser para mi igual que un padre.

A mi tía Yamila: por ser mi ejemplo a seguir y mi inspiración en toda mi vida como estudiante.

A mi abuela Marta: por siempre darme su amor y cariño

A la memoria de mi abuelo José Antonio que, aunque la vida no le dio la oportunidad de estar hoy presente sé que estaría hoy muy orgulloso de todos mis logros.

A toda mi familia y amigos que siempre han estado a mi lado en los momentos buenos y malos y me han ayudado en todo el transcurso de mi vida.

Agradezco:

A mis padres agradecerle por todo ese amor infinito que siempre me han dado, por su comprensión y apoyo en todos los momentos y porque sé que se han preocupado igual que yo porque este trabajo se pudiese terminar.

A mis abuelos Niña y Yoyo que siempre estuvieron pendiente de todo lo que me hacía falta, por su amor incondicional, por preocuparse tanto por mí. Por esos consejos tan buenos que siempre me dieron y me facilitaron el camino de la vida, en general por todo lo que han hecho por mí y por cómo han sido conmigo.

A mi tío Alexis que siempre ha sido esa persona incondicional en todo momento, por alegrarme el día por muy malo que fuese y siempre estar pendiente de mis cosas y por ser más que un tío para mí.

A mi tía Yami que siempre ha sido esa fuente de inspiración para poder seguir adelante y por darme tan buenos consejos.

A mi abuela Marta que siempre se ha preocupado por mí y ha sabido brindarme su amor y cariño.

A mi hermano que, aunque es el menor de los dos siempre ha estado pendiente de mí.

A mi novia Gleidys que ha sido esa persona incondicional que siempre ha estado mi lado en todo momento y que desde que la conocí hemos compartido muy lindos momentos. Te agradezco por ser como eres conmigo.

Agradecerle a todos mis familiares y amigos, que algunos son como de la familia por su preocupación y apoyo siempre en todo momento.

A mi gran amiga Susana, que yo también te considero como una hermana, que desde que la conocí en esta universidad siempre me ha ayudado en los buenos y malos momentos y me ha brindado su apoyo incondicional.

Agradezco a mis tutores que me ayudaron muchísimo y siempre me prestaron la mayor atención en todo momento y por la paciencia que tuvieron conmigo cuando los molestaba más de la cuenta.

A mi amigo el guille y a Jaquelin que siempre que tenía una duda nunca dudaron en brindarme su ayuda a la hora que fuese.

Agradecerles a mis amigos del edificio y del cuarto que no los menciono porque son muchos, pero saben que estoy agradecidos de ellos por todos los momentos que pasamos que no son pocos.

Agradecerles a todas aquellas personas que en un momento u otro se preocuparon por mí y preguntaron cómo iba la tesis.

En la Universidad de las Ciencias Informáticas se encuentra la facultad 4, en esta se desarrolla la Plataforma Educativa ZERA v2.0. La misma presenta la necesidad de que los ejercicios creados en sus cursos sean interoperables. Por esta razón fue necesario desarrollar un módulo que contribuyera a la interoperabilidad de los ejercicios en la plataforma. Este módulo les proporciona a los profesores la posibilidad de intercambiar ejercicios con otras plataformas o entre los cursos de la misma. Además, le permite al usuario exportar e importar los distintos ejercicios creados en los cursos siguiendo el estándar IMS Question and Test Interoperability (IMS QTI) v2.0. Este estándar presenta como ventaja la reutilización de los ejercicios entre plataformas del mismo tipo, además de contar con la mayor popularidad para la estandarización de evaluaciones en estos sistemas. Para guiar el desarrollo del trabajo se utilizó la metodología Proceso Unificado Ágil (AUP) en su variante UCI, generándose los artefactos fundamentales. Se utilizaron los lenguajes de programación HTML v5, CCS v3, JavaScript y PHP v7; los framework de desarrollo Symfony, Bootstrap y JQuery, y el entorno de desarrollo Netbeans v8.0. Como resultado, los usuarios podrán exportar e importar los ejercicios de los cursos siguiendo el estándar IMS QTI v2.0. Esto permite lograr la interoperabilidad de los ejercicios entre las distintas plataformas que sigan dicho estándar. Se realizaron pruebas a la propuesta de solución con el objetivo de detectar y erradicar las no conformidades. Una vez concluidas se obtuvo un módulo con la calidad y funcionalidad requerida por el cliente.

Palabras clave: exportar, importar, IMS QTI, interoperabilidad, plataforma educativa

Índice

Introducción

Las Tecnologías de la Información y las Comunicaciones (TIC) son un fenómeno que ha invadido todos los sectores de la vida siendo utilizadas en distintas esferas de la sociedad como son la salud, deporte, economía, militar y educación. Se vive en una sociedad que avanza rápidamente gracias al desarrollo tecnológico, donde las TIC han cambiado completamente el paradigma de la educación, dejando una huella en muchos campos del saber aportándole a los estudiantes elementos visuales y auditivos, conocimientos más actuales y distintos beneficios que permiten enriquecer aún más el proceso de enseñanza-aprendizaje.

En Cuba, la Universidad de las Ciencias Informáticas (UCI), fue creada por nuestro Comandante en Jefe Fidel Castro Ruz como un “Proyecto del Futuro”. Su misión es formar profesionales comprometidos con su patria, calificados en la rama de la informática y la producción de software y servicios informáticos, vinculando el estudio y el trabajo como un modelo de formación.

En la UCI, el Centro de Tecnologías para la Formación (FORTES) se encuentra desarrollando la plataforma educativa ZERA $_{v2.0}$. La misma es considerada una plataforma virtual de teleformación, comúnmente conocida como Sistemas de Gestión del Aprendizaje (LMS por sus siglas en inglés)

Un LMS (*Learning Management System*) es un sistema de gestión de aprendizaje online, que permite administrar, distribuir, monitorear, evaluar y apoyar las diferentes actividades previamente diseñadas y programadas dentro de un proceso de formación completamente virtual, o de formación semi-presencial. (Mayor, 2014)

Actualmente para los usuarios finales, es más difícil adaptarse a los sistemas por las diferencias que existen entre sí en cuanto a interfaz, tanto como en los formatos de fichero que manejan. De igual modo, para los desarrolladores es muy costoso lograr una compatibilidad con otro sistema por las variaciones de implementación que existen entre software con características similares. Es por ello que se afecta la interoperabilidad entre las distintas plataformas educativas y los ejercicios creados en esta.

La interoperabilidad es la capacidad que tiene un producto o sistema, cuyas interfaces son totalmente conocidas, para funcionar con otros productos o sistemas existentes o futuros, sin restricción de acceso o de implementación. (Albentia, 2008). O sea que las distintas plataformas educativas puedan intercambiar información sin restricción alguna.

Para dar solución a este problema se crean los estándares, que de manera general logran una terminología común, permitiendo a los diseñadores discutir los mismos conceptos y hacer valoraciones comparativas, mantenimiento y evolución, dado que todos los programas tienen la misma estructura. Una de las principales instituciones vinculadas a la estandarización de los sistemas de educación a distancia es IMS, consorcio donde se agrupan vendedores, productores, implementadores y consumidores de estos sistemas, que se enfoca completamente a desarrollar especificaciones en formato (XML "Lenguaje de Marcas Extensible"). Esta institución presenta una serie de especificaciones de las cuales IMS QTI $_{v2.0}$ es la solución apropiada para la estandarización de las evaluaciones, teniendo como mayor ventaja la reducción del costo y tiempo de desarrollo de estos objetos, gracias a la reutilización e intercambio de ejercicios y pruebas entre las distintas plataformas educativas.

Actualmente la plataforma ZERA $_{v2.0}$ no les brinda a los profesores la posibilidad de intercambiar ejercicios con otras plataformas de su mismo tipo, así como tampoco entre cursos de la misma. Por esta razón a los usuarios se les hace engorroso el manejo de los ejercicios debido a que tienen que introducir los datos de forma manual. Además, los ejercicios encontrados en otros LMS que les resulten de utilidad tienen que ser introducidos al sistema de forma manual. Esto trae consigo que se afecte en gran medida la interoperabilidad de los ejercicios en la plataforma.

Debido a esto se plantea como **problema a resolver**: ¿Cómo contribuir a la interoperabilidad de los ejercicios en la Plataforma Educativa ZERA $_{v2.0}$?

Definido el problema se establece como **objeto de estudio** interoperabilidad en las plataformas educativas.

Estableciendo como **campo de acción** de la investigación la interoperabilidad de los ejercicios en la Plataforma Educativa ZERA $_{v2.0}$

Se plantea como **objetivo general** desarrollar un módulo para contribuir con la interoperabilidad de los ejercicios en la Plataforma ZERA $_{v2.0}$.

Los **objetivos específicos** a cumplir son:

- Sistematizar los elementos teóricos relacionados con las plataformas educativas que permiten la interoperabilidad entre sus ejercicios.
- Realizar el análisis y diseño del módulo para contribuir con la interoperabilidad de los ejercicios en la plataforma educativa ZERA $_{v2.0}$.

- Implementar el módulo para contribuir con la interoperabilidad de los ejercicios en la plataforma educativa ZERA v2.0.
- Aplicar las pruebas necesarias para garantizar el correcto funcionamiento del módulo.

Para dar cumplimiento a los objetivos específicos se definieron las siguientes **tareas de investigación**.

- Revisión bibliográfica relacionada con el problema de la investigación.
- Elaboración del diseño teórico de la investigación.
- Análisis de la propuesta de solución.
- Diseño de la propuesta de solución.
- Implementación de un módulo para contribuir con la interoperabilidad de los ejercicios de la Plataforma Educativa ZERA v2.0.
- Realización de las pruebas y documentación de los resultados.

La investigación está sustentada por la siguiente **idea a defender**: Si se pueden exportar e importar los ejercicios de la Plataforma Educativa ZERA v2.0 hacia y desde otras plataformas similares que sigan el estándar IMS QTI v2.0, se contribuirá con la interoperabilidad de los mismos en la plataforma.

El **resultado** de esta investigación sería: un módulo que contribuya a la interoperabilidad de los ejercicios en la Plataforma Educativa ZERA v2.0, posibilitándole a los profesores exportar e importar los ejercicios siguiendo el estándar IMS QTI v2.0.

Los **métodos científicos** utilizados quedaron determinados por el objetivo general y las tareas de investigación previstas. Los mismos son:

A nivel empírico:

Observación: posibilitó la toma de las prácticas positivas de sistemas que logren la interoperabilidad siguiendo el estándar IMS QTI v2.0, así como la identificación de vulnerabilidades comunes en la puesta en funcionamiento de las mismas para convertirlas en ventajas durante el desarrollo de la investigación.

A nivel teórico:

Análisis-síntesis: fue utilizado durante la investigación para realizar un análisis de la documentación relacionada con los estándares de codificación, y los conceptos vinculados con la interoperabilidad de ejercicios en plataformas educativas. Facilitó el análisis de las características de las herramientas y estándares empleados. Esto permitió realizar posteriormente la síntesis de los aspectos más importantes para el desarrollo del trabajo.

Análisis histórico-lógico: permitió realizar un análisis sobre las tendencias en el empleo de estándares de codificación que se utilizan para lograr interoperabilidad de ejercicios entre plataformas educativas. Además, sirvió para identificar soluciones similares que aportaron elementos importantes para la solución propuesta.

Modelación: permitió representar los conceptos vinculados con la propuesta de solución y las relaciones entre ellos permitiendo un mejor entendimiento de los mismos.

Para lograr una mejor comprensión de la presente investigación se define la siguiente **estructura capitular:**

Capítulo 1: Fundamentación teórica.
En este capítulo se realiza un estudio de las soluciones similares y se describen las metodologías, herramientas y tecnologías que serán empleadas para la creación del módulo que contribuya a la interoperabilidad de los ejercicios en la plataforma educativa ZERA $_{v2.0}$.

Capítulo 2: Análisis y Diseño.
Este capítulo contiene una descripción de la propuesta de solución. Agrupa el diagrama de modelo del dominio, diagrama de análisis, de colaboración, de secuencia y los de clases del diseño generados a partir del problema planteado. Además de especificar los requisitos funcionales y no funcionales que deberá cumplir la solución de la investigación.

Capítulo 3: Implementación y pruebas.
Este capítulo contiene las clases a utilizar en la implementación de la solución. Describe las pruebas de software realizadas para lograr la calidad del producto y la satisfacción del cliente.

Capítulo 1: Fundamentación teórica

1.1 Introducción

En este capítulo se aborda el marco teórico conceptual asociado al objeto de la investigación. Se analizan las soluciones similares para obtener las funcionalidades que servirán como base para el desarrollo de la aplicación. Se describe la metodología de desarrollo de software, las herramientas y tecnologías que serán empleadas en la propuesta de solución.

1.2 Conceptos asociados al dominio del problema

A continuación, se definen los términos más significativos relacionados con el objeto de estudio, para un mejor entendimiento de la propuesta de solución.

1.2.1 Software educativo

Los software educativos, se definen de forma genérica como aplicaciones o programas computacionales que faciliten el proceso de enseñanza-aprendizaje. Algunos autores lo conceptualizan como cualquier programa computacional cuyas características estructurales y funcionales sirvan de apoyo al proceso de enseñar, aprender y administrar, o el que está destinado a la enseñanza y el autoaprendizaje y además permite el desarrollo de ciertas habilidades cognitivas; términos que seguramente se replantearán en la medida que se introduzcan nuevos desarrollos tecnológicos para el trabajo en red en Internet. (María Vidal Ledol, 2010)

Las características más generalizadas en los software educativos son:

- Finalidad: orientados a la enseñanza-aprendizaje en todas sus formas.
- Utilización del computador: el medio utilizado como soporte es el computador.
- Facilidad de uso: son intuitivos y aplica reglas generales de uso y de fácil comprensión para su navegabilidad o desplazamiento y recursividad o posibilidad de regreso a temáticas de interés desde cualquier punto en el ambiente virtual.

Interactividad: permite un intercambio efectivo de información con el estudiante. (María Vidal Ledol, 2010)

1.2.2 Interoperabilidad

La revolución de las nuevas tecnologías y el crecimiento acelerado de Internet han permitido la creación de un gran número de plataformas educativas y a su vez la necesidad de regular e interoperar estas plataformas. La interoperabilidad es un factor clave en los sistemas ya que permite el intercambio y la reutilización de recursos educativos que han sido desarrollados en distintas plataformas.

Una condición que tiene que cumplirse para que pueda darse la interoperabilidad, y que por sí sola no es suficiente, es que exista un documento o documentos que especifiquen totalmente la forma en que los sistemas tienen que interactuar.

Según, Ruiz Palacios se define como: “la habilidad de dos o más sistemas o componentes para intercambiar información y utilizar la información intercambiada.” (Ruiz, 2013)

La interoperabilidad tiene cuatro dimensiones: (Martínez Usero, 2004)

Interoperabilidad organizativa. Es aquella dimensión de la interoperabilidad relativa a la capacidad de las entidades y de los procesos a través de los cuales llevan a cabo sus actividades para colaborar con el objeto de alcanzar logros mutuamente acordados relativos a los servicios que prestan.

Interoperabilidad semántica. Es aquella dimensión de la interoperabilidad relativa a que la información intercambiada pueda ser interpretable de forma automática y reutilizable por aplicaciones que no intervinieron en su creación.

Interoperabilidad técnica. Es aquella dimensión de la interoperabilidad relativa a la relación entre sistemas y servicios de tecnologías de la información, incluyendo aspectos tales como las interfaces, la interconexión, la integración de datos y servicios, la presentación de la información, la accesibilidad y la seguridad, u otros de naturaleza análoga.

Interoperabilidad en el tiempo. Es aquella dimensión de la interoperabilidad relativa a la interacción entre elementos que corresponden a diversas oleadas tecnológicas; se manifiesta especialmente en la conservación de la información en soporte electrónico.

Teniendo en cuenta las características de cada una de las dimensiones las más apropiadas para la investigación son la Interoperabilidad semántica y la Interoperabilidad en el tiempo.

1.2.3 Formato estándar

En tecnología y en general, un estándar es una especificación que regula la realización de ciertos procesos o la fabricación de componentes para garantizar la interoperabilidad. En informática, un formato de almacenamiento es la estructura usada para grabar datos en un archivo. Un formato de documentos es una especificación que determina cómo debe almacenarse en un archivo la información contenida en un documento. Los formatos pueden ser propietarios o abiertos. Un formato propietario es propiedad de su fabricante. Su uso por parte de terceros está condicionado a los permisos que el propietario otorgue. Estos permisos dependen generalmente de decisiones comerciales. Las especificaciones de estos formatos generalmente no son públicas y restringe el acceso a archivos que han sido almacenados con dichos formatos. Un formato abierto es de uso libre. Las especificaciones se encuentran disponibles en forma gratuita y no se requiere un permiso del fabricante para desarrollar software que las utilice para almacenar o abrir documentos.

1.2.4 IMS QTI

La especificación IMS Question and Test Interoperability (IMS QTI) permite crear preguntas individuales y evaluaciones completas. El objetivo principal de esta especificación es permitir el intercambio de preguntas, evaluaciones y resultados entre distintas herramientas. Con este propósito IMS QTI plantea un modelo en el que se definen los componentes principales que intervienen en el proceso de evaluación y, adicionalmente a este modelo, se proporciona un formato de contenido para almacenar las preguntas de manera independiente del sistema o herramienta de autoría utilizada para crearlas. (Baltasar Fernández Manjón, 2010)

1.3 Especificación IMS QTI

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

Con este propósito se plantea y se documenta un formato de contenido para almacenar las preguntas o ítems independientemente del sistema o herramienta de autoría utilizada para

crearlas. Esto permite, por ejemplo, el uso de las mismas preguntas en diversos LMS o en sistemas de evaluación electrónica, o la integración en un único LMS de preguntas o exámenes desarrollados con distintas herramientas. (Baltasar Fernández Manjón, 2010)

1.3.1 Versión 2.0

La versión 2.0 surge en 2005 y 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 se 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.

Mientras que las versiones anteriores de la especificación se centraban principalmente en cómo se presentaba finalmente la pregunta, ahora se definen los posibles tipos de interacciones por parte del usuario (ejemplo, seleccionar uno o más elementos de una lista, crear asociaciones entre elementos de dos listas, introducir texto, seleccionar un fragmento de texto de un texto más largo, entre otros). Además de todas las interacciones contempladas introduce un tipo de interacción propio para poder extender el modelo y crear nuevas formas de interacción para poder introducir nuevos tipos de preguntas. Es decir, QTI no es un modelo completo y cerrado si no que permite que si una empresa lo considera oportuno pueda extenderlo y ampliarlo (eso sí, esos nuevos tipos de preguntas creados por ampliación no serían reconocidos necesariamente por otro sistema QTI).

También tiene plantillas de preguntas para crear familias de preguntas similares, en la cual hay partes variables que se seleccionan de manera aleatoria entre un conjunto de valores predefinidos.

Otra de las novedades que introduce son las preguntas adaptativas que permiten al alumno realizar varios intentos sobre dicha pregunta. Esto permite, por ejemplo, que el alumno pueda alterar su respuesta debido a la realimentación, por ejemplo, la presentación de alguna pista en caso de que la respuesta sea incorrecta o parcialmente incorrecta, o que se le planteen preguntas adicionales en función de su respuesta actual. (IMS QUESTION & TEST INTEROPERABILITY SPECIFICATION, 2010)

1.3.2 Item

Un item incluye la pregunta que se presenta al usuario y puede incluir otra información necesaria para el procesamiento de la respuesta o puntuación, retroalimentación instantánea o consejos para su realización, y otros mecanismos para mejorar el examen y/o la evaluación QTI trata de ser pedagógicamente neutral y proporciona un gran conjunto de preguntas que habitualmente se utilizan en las evaluaciones, tales como elección verdadero/falso, elección múltiple con respuesta única, elección múltiple con varias respuestas válidas, rellenar campos en blanco, ordenar objetos, relacionar objetos, entre otros. Además, permite definir nuevos tipos de preguntas si fuera necesario. (Baltasar Fernández Manjón, 2010)

1.3.3 Las preguntas

Las preguntas individuales (assessmentItem) en QTI son auto-contenidas, es decir, incluyen toda la información necesaria para su presentación al alumno y su corrección automática. Toda la información relativa a la presentación ha sido agrupada en el cuerpo (itemBody) de las preguntas. En la presentación de la pregunta están involucrados dos aspectos:

- El enunciado de la pregunta. Obviamente, la pregunta debe contener el enunciado de la misma y, de manera adicional, puede contener material explicativo complementario que permita al docente indicar el contexto en el que se realiza la pregunta
- La construcción de la respuesta. De manera adicional al enunciado de la pregunta, se debe dotar al alumno del equivalente del lápiz y papel para poder construir la respuesta. En el caso de IMS QTI v 2 se ha introducido el concepto de interacción (interaction). Dependiendo del tipo la herramienta informática generará una presentación acorde. (Baltasar Fernández Manjón, 2010)

1.3.4 Interacciones

En el caso de IMS QTI no está contemplado el concepto de tipo de pregunta, existiendo en su lugar el concepto de interacción. Las interacciones permiten al profesor especificar las herramientas que tendrá el alumno disponible para poder construir la respuesta.

Al igual que existen múltiples tipos de pregunta, también existen múltiples tipos de interacción. A continuación, se describirán los tipos de interacciones posibles que pueden utilizarse dentro de una pregunta.

Hay que remarcar que existen dos grupos de interacciones, las interacciones en línea y las interacciones en bloque. Las interacciones en línea son un tipo de interacción que pueden incluirse en medio del enunciado de la pregunta. Por otra parte, las interacciones de tipo bloque están pensadas para ser presentadas de manera independiente al enunciado de la pregunta. (Baltasar Fernández Manjón, 2010) La especificación IMS QTI define más interacciones de ejercicios, pero solo se profundizó en aquellas que resultaron de interés para la investigación.

interacciones simples

Las interacciones simples con aquellas interacciones en las que la corrección de las mismas se realiza en base a la selección de una opción o varias opciones disponibles. Las interacciones que pertenecen a esta categoría son:

choiceInteraction. Esta interacción muestra al alumno un conjunto de posibles opciones. El alumno podrá seleccionar una o varias posibles opciones como respuesta. Es posible indicar que el conjunto de posibles opciones sea barajado entre distintos intentos del alumno.

orderInteraction. En esta interacción el objetivo del alumno es reordenar el conjunto de soluciones proporcionada. Además, es posible un número mínimo y un número máximo de opciones que conforman la solución, de manera que se realizaría una selección sobre las opciones disponibles y posteriormente se realizaría una ordenación de los elementos de dicha selección.

associateInteraction. Esta interacción presenta al alumno un conjunto de opciones y permite crear asociaciones por parejas entre dichas opciones. Es posible indicar el número mínimo y máximo de asociaciones que deben crearse como parte de la respuesta. Además, también es posible indicar el número mínimo y máximo de veces que una de las opciones puede aparecer dentro de una asociación.

Interacciones de texto

En esta categoría se encuentran las interacciones en las que la respuesta que construirá el alumno puede ser una única palabra, una frase corta o un párrafo de texto completo. Estas interacciones permiten que durante el proceso de corrección se tenga en cuenta la respuesta en forma de texto que ha construido el alumno. Las interacciones que pertenecen a esta categoría son:

inlineChoiceInteraction (interacción en línea). Esta interacción está pensada para definir un hueco donde se permitirá al alumno escoger entre un conjunto de opciones, donde cada una de estas opciones una palabra o frase corta. A diferencia de gapMatchInteraction, esta interacción está ideada para que cada uno de los huecos pueda tener un conjunto de opciones independiente. Es posible definir que las respuestas sean barajadas entre distintos intentos del alumno.

textEntryInteraction (interacción en línea). Al igual que la interacción anterior, esta interacción tiene como objetivo crear un hueco donde se permitirá teclear una palabra o frase corta para poder construir la respuesta. Cuando se define una pregunta con esta interacción es posible especificar la longitud del texto que se espera que el alumno introduzca.

1.4 Estudio de soluciones similares

A continuación, se realiza un estudio de las soluciones similares con el propósito de identificar los estándares para la interoperabilidad de los ejercicios.

Moodle

Moodle es un software diseñado para ayudar a los educadores a crear cursos en línea de alta calidad y entornos de aprendizaje virtuales. Tales sistemas de aprendizaje en línea son algunas veces llamados VLEs (Virtual Learning Environments) o entornos virtuales de aprendizaje.

La palabra Moodle originalmente es un acrónimo de Modular Object-Oriented Dynamic Learning Environment (Entorno de Aprendizaje Dinámico Orientado a Objetos y Modular).

Una de las principales características de Moodle sobre otros sistemas es que está hecho en base a la pedagogía social constructivista, donde la comunicación tiene un espacio relevante en el camino de la construcción del conocimiento. Siendo el objetivo generar una experiencia de aprendizaje enriquecedora.

Una de las fortalezas de Moodle es que es Software Libre. Esto significa que su creador inicial, al momento de publicarlo en Internet, decidió utilizar la Licencia Pública GNU (GPL) y por lo tanto puede ser utilizado sin pagar “licencias”. La institución que lo instale está autorizada a copiar, usar y modificar Moodle. En consecuencia, la plataforma Moodle conforma un sistema permanentemente activo, seguro y en constante evolución. (Moodle en la formación inicial del profesorado, 2005)

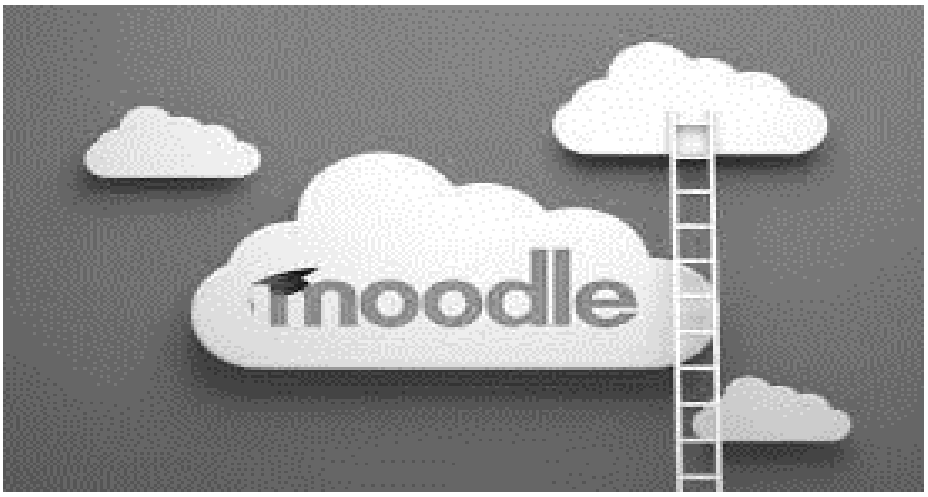


Figura 1. Plataforma Educativa Moodle

Estudiar la plataforma educativa Moodle resultó de gran apoyo para la visualización de los ejercicios a la hora de ser importados, pues esta antes de importarlos al sistema los muestra al usuario. Esta opción le brinda la posibilidad de hacerle algún cambio o simplemente aceptarlo garantizando que no se importen ejercicios con errores.

Sakai

Sakai es una plataforma en línea desarrollada por una comunidad de educadores para facilitar la enseñanza y el aprendizaje colaborativo en instituciones educativas. El proyecto originado en la Universidad de Michigan, fue anunciado oficialmente en EDUCAUSE en noviembre de 2003. La fundación Sakai se formalizó en octubre del 2005 y se le concedió la propiedad intelectual de la plataforma por medio de un acuerdo de licencia de derecho de autor. La plataforma es distribuida gratuitamente como un programa de código abierto bajo la licencia de comunidad educativa.

La plataforma es utilizada para el manejo de cursos, como hace Blackboard y Moodle, pero también es un ambiente de aprendizaje y colaboración que añade herramientas para el desarrollo de contenidos. Permite colocar el prontuario y presentaciones dentro de la plataforma, y ofrece plantillas para diseñar lecciones de manera secuencial, y para crear asignaciones y exámenes en línea. Además, tiene un ‘drop box’ para compartir documentos, facilidades de archivo de emails enviados, y herramientas para crear un wiki y un blog, Podcasts, calendario, chat y foros de discusión, glosario, encuestas, una sección de páginas web externas y noticias vía RSS.

Otra de las capacidades de Sakai es la construcción de e-portafolios, que está integrado en la plataforma y se actualiza según la actividad del usuario. El usuario puede diseñar, publicar,

compartir y ver portafolios de sus trabajos, además de un sistema de matrices para documentar su desarrollo y aprendizaje.

Sakai permite hacer una prueba de la plataforma en servidores gratuitos para experimentar todas las herramientas. La aplicación ha sido desarrollada en Java, lo cual permite su uso en múltiples sistemas operativos. La última versión estable es la 2.7.1 lanzada el 25 de agosto de 2010. (Sakai – Plataforma educativa de código abierto, 2010)

Figura 2. Plataforma Educativa Sakai

El estudio de esta plataforma proporcionó la forma de identificar las tipologías a la hora de ser importadas.

Zera 1.0

La plataforma educativa ZERA en su versión 1.0, es un LMS creado en la Universidad de las Ciencias Informáticas, entre las características que hacen de esta plataforma única se tiene: que es basada en híper-entornos de aprendizajes; permite la creación de cursos con una estructura capitular donde el contenido se muestra con la estructura de un libro: avance del contenido (marcador de libro), resaltado, apuntes al contenido; creación de 30 tipos de recursos y 11 tipologías de ejercicios; soporte para las especificaciones IMS QTI y SCORM; incorpora las sugerencias de uso y registro de avance; la evaluación por rúbricas y por competencias; atención diferenciada (recorridos dirigidos, softwares, orientaciones de trabajos); sistema distribuido ideal para instituciones con problemas de conectividad y los procesos comunes de la gestión

académica y herramientas de comunicación fórum, chat, entre otras funcionalidades. (Plataforma educativa ZERA: modelo de adaptación de contenidos sensible al contexto, 2015)

Figura 3. Plataforma Educativa ZERA v1.0

El estudio de esta plataforma resultó de gran ayuda para el estudio de todo lo referente al estándar IMS QTIv2.0 y su representación en una planilla XML.

1.5 Metodologías, tecnologías y herramientas

A continuación, se describen la metodología, tecnologías y herramientas utilizadas para el desarrollo del módulo.

1.5.1 Metodología de desarrollo de software

El desarrollo de software no es una tarea fácil. Prueba de ello es que existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo. Por una parte, se tienen aquellas propuestas más tradicionales que se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir y las herramientas y notaciones que se usarán. Estas propuestas han demostrado ser efectivas y necesarias en un gran número de proyectos, pero también han presentado problemas en muchos otros.

Otra aproximación es centrarse en otras dimensiones, como por ejemplo el factor humano o el producto software. Esta es la filosofía de las metodologías ágiles, las cuales dan mayor valor al

individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas. Este enfoque muestra su efectividad en proyectos con requisitos muy cambiantes y cuando se exige reducir drásticamente los tiempos de desarrollo, pero manteniendo una alta calidad. (Metodologías ágiles para el desarrollo de software: extreme Programming (XP), 2006)

Proceso Unificado Ágil (AUP)

Constituye una versión simplificada del Proceso Unificado Racional (*Rational Unified Process*, RUP), desarrollada por Scott Ambler. Esta metodología combina procesos propios del concepto unificado tradicional con técnicas ágiles, con el objetivo de mejorar la productividad. Permitiendo así describir de manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio. AUP aplica técnicas ágiles, entre las que se incluyen (Sánchez, 2014):

- El desarrollo dirigido por pruebas.
- El modelado ágil.
- La gestión de cambios ágil.
- La refactorización de bases de datos para mejorar la productividad.

Al igual que RUP, en AUP se definen cuatro fases que se desarrollan de manera consecutiva:

1. Inicio.
2. Elaboración.
3. Construcción.
4. Transición.

AUP define siete disciplinas, de ellas cuatro son dirigidos para la ingeniería y tres para la gestión de proyectos. La UCI realiza una variación definiendo nuevas disciplinas, establecidas de la siguiente forma:

1. Modelado de negocio.
2. Requisitos.
3. Análisis y diseño.
4. Implementación.

5. Pruebas internas.
6. Pruebas de liberación.
7. Pruebas de aceptación.
8. Despliegue.

Las disciplinas propuestas por AUP: gestión de configuración, gestión de proyectos y entorno, se cubrirán de acuerdo a las áreas de procesos que define CMMI-DEV en su versión 1.3. Estas quedan de la siguiente forma: gestión de la configuración, la planeación del proyecto, el monitoreo y control de proyecto.

AUP propone 9 roles en el desarrollo de un proyecto, pero debido a la adaptación realizada por la Universidad, se decide para el ciclo de vida de los proyectos tener 11 roles, de los cuales se mantienen algunos de los propuestos por AUP, unificando o agregando otros para una mejor organización del equipo de desarrollo. A continuación, se muestran los roles resultantes:

1. Jefe de proyecto.
2. Planificador.
3. Analista.
4. Arquitecto de información.
5. Desarrollador.
6. Administrador de la configuración.
7. *Stakeholder* (Cliente/Proveedor de requisitos).
8. Administrador de calidad.
9. Probador.
10. Arquitecto de software.
11. Administrador de base de datos.

Extreme Programming (XP)

XP es una metodología ágil para el desarrollo de software y consiste básicamente en ajustarse estrictamente a una serie de reglas que se centran en las necesidades del cliente para lograr un

producto de buena calidad en poco tiempo, centrada en potenciar las relaciones interpersonales como clave para el éxito del desarrollo de software.

La filosofía de XP es satisfacer por completo las necesidades del cliente, por eso lo integra como una parte más del equipo de desarrollo. Promueve el trabajo en equipo, preocupándose en todo momento del aprendizaje de los desarrolladores y estableciendo un buen clima de trabajo.

Este tipo de programación es la adecuada para los proyectos con requisitos imprecisos, muy cambiantes y donde existe un alto riesgo técnico. XP está diseñada para el desarrollo de aplicaciones que requieran un grupo de programadores pequeño, dónde la comunicación sea más factible que en grupos de desarrollo grandes. La comunicación es un punto importante y debe realizarse entre los programadores es, los jefes de proyecto y los clientes. (Yolanda, 2013)

Selección de la metodología de desarrollo de software para solucionar el problema

Debido a las características del proyecto es necesario combinar elementos de las metodologías ágiles tales como: el desarrollo dirigido por pruebas y la gestión de cambios ágil con un enfoque más tradicional en el que se lleve a cabo una documentación más detallada y se generen un mayor número de artefactos que puedan facilitar la comprensión del sistema. Por estas razones se decide utilizar la metodología AUP en su variante UCI, utilizando el escenario 4 “Historias de usuario” y los artefactos definidos por el cliente, siguiendo las líneas de desarrollo del proyecto ZERA$_{v2.0}$ y rigiéndose por el documento Metodología de desarrollo para la actividad productiva de la UCI en su versión 1.2.

1.6 Tecnologías y herramientas

A continuación, se mencionan las tecnologías y herramientas utilizadas para la realización del proyecto siguiendo las líneas de desarrollo del proyecto.

1.6.1 Lenguajes de desarrollo

Un lenguaje de programación es aquella estructura que, con una cierta base sintáctica y semántica, imparte distintas instrucciones a un programa de computadora. El lenguaje de programación tiene la capacidad de especificar, de forma precisa, cuáles son los datos que debe trabajar un equipo informático, de qué modo deben ser conservados o transferidos dichos datos y qué instrucciones debe poner en marcha la computadora ante ciertas circunstancias. (Merino,

2012) Para el desarrollo de la presente investigación se utilizaron los lenguajes del lado del cliente y del servidor.

Lenguajes del lado del cliente: son aquellos que el programa reside junto a la página web en el servidor, pero es transferido al cliente para que este los ejecute. Es decir, el servidor no interviene en el proceso de crear la página web solicitada por el usuario.

Lenguajes del lado del servidor: los lenguajes son ejecutados por el servidor y lo que se envía al cliente es la respuesta o el resultado de dicha ejecución.

A continuación, se describen los lenguajes que se utilizaron:

HTML v5: HTML es el lenguaje con el que se define el contenido de las páginas web. Básicamente se trata de un conjunto de etiquetas que sirven para definir el texto y otros elementos que compondrán una página web, como imágenes, listas y vídeos. (Álvarez, 2001). HTML5 es la quinta revisión de este estándar. Las principales novedades que trae son nuevas etiquetas para conseguir la Web Semántica (que los elementos o etiquetas aporten significado y no solo contenido) y nuevas Apis para permitir funcionalidades avanzadas de JavaScript. Esta versión permite una mayor interacción entre las páginas web y contenido media, así como una mayor facilidad a la hora de codificar los diseños básicos (Martínez, 2010).

CCS v3: Significa Cascade Style Sheets, también llamado Hojas de Estilo en Cascada. CSS es un lenguaje de marcado que se emplea para dar formato a un sitio web. Es decir, funciona en conjunto con los archivos HTML.

CSS3 se puede aplicar en la misma hoja en la que estás desarrollando un documento HTML, pero por motivos de productividad se suele realizar en un documento aparte con la extensión .css. Este documento se puede vincular a cada página HTML que conforme el sitio web, es por ello que es más útil realizar los estilos por separado.

CSS3 sirve para cambiar el aspecto de un sitio web, desde las medidas para los márgenes hasta las especificaciones para las imágenes y el texto. Funciona mediante módulos, algunos de los más comunes son “colors”, “fonts”, “backgrounds”, entre otros. Los módulos son sólo categorías en las que se pueden dividir las modificaciones que hacemos al aspecto de nuestro sitio web. (Damian, 2010)

JavaScript: (a veces abreviado como JS) es un lenguaje con muchas posibilidades, utilizado para crear pequeños programas que luego son insertados en una página web y en programas más grandes, orientados a objetos mucho más complejos. Con JavaScript se puede crear diferentes efectos e interactuar con los usuarios. Este lenguaje posee varias características, entre ellas se puede mencionar que es un lenguaje basado en acciones que posee menos restricciones. Además, gran parte de la programación en este lenguaje está centrada en describir objetos, escribir funciones que respondan a movimientos del mouse, aperturas, utilización de teclas, cargas de páginas entre otros. Es necesario resaltar que hay dos tipos de JavaScript: por un lado, está el que se ejecuta en el cliente, este es el JavaScript propiamente dicho, aunque técnicamente se denomina Navigator JavaScript. Pero también existe un JavaScript que se ejecuta en el servidor, es más reciente y se denomina LiveWire JavaScript. (Valdés, 2007)

PHP v7: son las siglas en inglés de “Hypertext Pre-Processor” que al traducirlo al español pierde un poco el sentido, lo mejor es analizar que significa “Lenguaje de Programación Interpretado”. Este lenguaje permite la visualización del contenido dinámico en las páginas web. Todo el código PHP es invisible para el usuario, porque todas las interacciones que se desarrollan en este lenguaje son por completo transformadas para que se puedan ver imágenes, variedad de multimedia y los formatos con los que son capaces de interactuar añadiendo o descargando información de ellos. (Venemedia, 2014)

Algunas de las características presentadas por PHP 7 son:

- Está dirigido al desarrollo de aplicaciones web dinámicas que permitan el acceso a información la cual se encuentre almacenada en una base de datos.
- Se puede acceder fácilmente a él, pues se presenta como una libre alternativa.
- Su lenguaje es fácil de aprender, puede ser utilizado por expertos y no expertos.
- Se puede aplicar técnicas de programación orientada a objetos.
- Tiene mucha flexibilidad lo que ha permitido que sea muy utilizado como lenguaje base para las aplicaciones web de manejo de contenido.

1.6.2 Framework de programación

En el desarrollo de software, un framework o infraestructura digital, es una estructura conceptual y tecnológica de soporte definido, normalmente, con artefactos o módulos concretos de software, que puede servir de base para la organización y desarrollo de software. Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto (Riehle, 2000)

En los sistemas informáticos, un framework es a menudo una estructura en capas que indica qué tipo de programas pueden o deben ser construidos y cómo se interrelacionan. Algunos marcos de trabajo de sistemas informáticos también incluyen programas reales, especifican interfaces de programación u ofrecen herramientas de programación para usar los marcos. Un framework puede servir para un conjunto de funciones dentro de un sistema y cómo se interrelacionan; las capas de un sistema operativo; las capas de un subsistema de aplicación; cómo debería normalizarse la comunicación en algún nivel de una red, entre otros. (definición de framework, 2016)

Symfony: Symfony es un proyecto PHP de software libre que permite crear aplicaciones y sitios web rápidos y seguros de forma profesional. Es un completo framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones.

Presenta ciertas ventajas como:

- Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web.
- Su código, y el de todos los componentes y librerías que incluye, se publican bajo la licencia MIT de software libre.
- La documentación del proyecto también es libre e incluye varios libros y decenas de tutoriales específicos.
- Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación compleja.
- Automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.

- Los componentes de Symfony son tan útiles y están tan probados, que proyectos tan gigantescos como Drupal 8 están construidos con ellos.

El resultado de todas estas ventajas es que no se debe reinventar nuevas soluciones cada vez que se crea una nueva aplicación web. (Fabien Potencier, 2008)

Bootstrap: Es un framework desarrollado y liberado por Twitter que tiene como objetivo facilitar el diseño web. Permite crear de forma sencilla webs de diseño adaptable, es decir, que se ajusten a cualquier dispositivo y tamaño de pantalla y siempre se vean igual de bien. Es Open Source o código abierto, por lo que se puede usar de forma gratuita y sin restricciones. (María, 2016)

Ofrece ciertas ventajas donde la más genérica es que permite simplificar el proceso de maquetación, sirviendo de guía para aplicar las buenas prácticas y los diferentes estándares.

- Se puede tener una web bien organizada de forma visual rápidamente: la curva de aprendizaje hace que su manejo sea asequible y rápido si se sabe maquetar.
- Permite utilizar muchos elementos web: desde iconos a desplegables, combinando HTML5, CSS y JavaScript.
- El diseño siempre será adaptable, no importa el dispositivo, la escala o resolución.
- El grid system: maquetar por columnas nunca fue tan fácil. Además, son muy configurables.
- Se integra muy bien con las principales librerías JavaScript.
- Cuenta con implementaciones externas para WordPress, Drupal, y otros.
- Permite el uso Less, para enriquecer aún más los estilos de la web.

JQuery: jQuery es una librería JavaScript (open-source), fuente abierta que funciona en múltiples navegadores, y que es compatible con CSS3. Su objetivo principal es hacer la programación “scripting” mucho más fácil y rápida del lado del cliente. Con jQuery se pueden producir páginas dinámicas, así como animaciones parecidas a Flash en un tiempo relativamente corto.

JQuery fue publicado por primera vez en enero del 2006 en “BarCamp NYC” por John Resign. Soporte para AJAX fue agregado un mes después, y el modelo de licenciamiento open source

del MIT fue adoptado en mayo de ese mismo año. (jQuery: Qué es, Orígenes, Ventajas y Desventajas, 2013)

- La ventaja principal de jQuery es que es mucho más fácil que sus competidores. Se pueden agregar plugins fácilmente, traduciéndose esto en un ahorro substancial de tiempo y esfuerzo.

- La licencia open source permite que la librería siempre cuente con soporte constante y rápido, publicándose actualizaciones de manera constante. La comunidad jQuery es activa y sumamente trabajadora.

- Otra ventaja sobre sus competidores como Flash y puro CSS es su excelente integración con AJAX

1.6.3 Entorno de desarrollo integrado

NetBeans: es un entorno de desarrollo, una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el NetBeans IDE. Es un producto libre y gratuito sin restricciones de uso. El editor de NetBeans PHP proporciona plantillas de código y herramientas de generación de código, como la generación de "getter y setter", refactorización, como "renombrar instantáneamente", sugerencias de parámetros, sugerencias y arreglos rápidos y finalización de código inteligente. (Gimeno, 2011)

La elección de este entorno de desarrollo se sustenta en que es el utilizado en el proyecto al cual va dirigido la presente investigación, además de que los miembros del equipo de desarrollo tienen experiencia en el trabajo con este IDE y también se cuenta con la licencia para su uso.

1.6.4 Herramienta CASE

Las herramientas CASE (Computer Aided Software Engineering), Ingeniería de Software Asistida por Computadora son diversas aplicaciones informáticas o programas informáticos destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y de dinero. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de diseño del proyecto, cálculo

de costos, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras. (Flores, 2013)

Visual Paradigm

Es una herramienta CASE para desarrollo de aplicaciones utilizando modelado UML ideal para Ingenieros de Software, Analistas de Sistemas y Arquitectos de sistemas que están interesados en construcción de sistemas a gran escala y necesitan confiabilidad y estabilidad en el desarrollo orientado a objetos. Ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Permite producir la documentación del sistema con diseñador de plantilla, dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Los analistas del sistema pueden estimar las consecuencias de los cambios en los diagramas de análisis de impacto, tales como la matriz y el diagrama de análisis. La versión a utilizar en la presente investigación es la 8.0. Se llega a esta conclusión por las ventajas que esta presenta, ya que permitirá realizar los diagramas y artefactos que se generan durante el desarrollo del software, con una interfaz de usuario fácil de usar. (Paradigm, 2015)

1.7 Conclusiones del capítulo

- Con el estudio de los conceptos asociados a la propuesta de solución se logró una mejor comprensión de interoperabilidad y del estándar IMS QTI.
- A partir de las soluciones similares se identificaron como funcionalidades bases para lograr la interoperabilidad de los ejercicios las de importar y exportar.
- Se describe la metodología de desarrollo de software AUP para generar los artefactos ingenieriles de la propuesta de solución.
- Mediante los nuevos paradigmas de desarrollo de software se identificaron los lenguajes, herramientas y tecnologías HTML v5, CCS v3, JavaScript y PHP v7; los framework de desarrollo Symfony, Bootstrap y JQuery, y el entorno de desarrollo Netbeans v8.0 para el desarrollo del mismo que facilitarán la creación del módulo.

Capítulo 2: Descripción de la propuesta de solución

2.1 Introducción

Para realizar el diseño de un sistema informático es necesario aclarar todos los conceptos y sus relaciones utilizando un lenguaje perceptible, mediante el modelo de dominio se relacionan los conceptos asociados a la investigación. Se realiza el levantamiento y especificación de los requisitos funcionales y no funcionales. Se plantea además la propuesta de solución al problema principal de la presente investigación. Se describe la arquitectura a utilizar y los patrones de diseños empleados para el desarrollo de los componentes.

2.2 Modelo de dominio

Un modelo del dominio es una representación visual de las clases conceptuales u objetos del mundo real en un dominio de interés. Este modelo muestra las clases conceptuales significativas en un dominio del problema. Es el artefacto más importante que se crea durante el análisis orientado a objetos. Un modelo del dominio se representa con un conjunto de diagramas de clases en los que no se define ninguna operación. (Craig Larman, 2003)

A continuación, se mencionan los conceptos que serán empleados en el modelo de dominio.

Curso: es donde se encuentra toda la información referente a un tema dado en la plataforma administrado por el profesor del curso. Uno de los componentes que integran el curso son los ejercicios.

Ejercicios: es la herramienta fundamental que emplea el profesor para evaluar lo aprendido por los estudiantes en un curso. Algunas de las tipologías de ejercicios son:

- Selección simple: le permite al estudiante seleccionar una de las respuestas que considere correcta.
- Selección múltiple: le permite al estudiante seleccionar varias respuestas de una misma pregunta.
- Verdadero o falso: le permite seleccionar al estudiante si la pregunta es verdadera o falsa.
- Ordenamiento: le permite ordenar a los estudiantes las preguntas.

- Enlazar: le permite enlazar al estudiante la pregunta con la respuesta correcta.
- Completamiento por escritura: permite al estudiante escribir en un cuadro de texto la palabra que considere correcta para completar el texto dado.
- Completamiento por desplazamiento: permite al estudiante seleccionar en una lista de palabras dada cual es la que considera correcta para completar el texto dado.

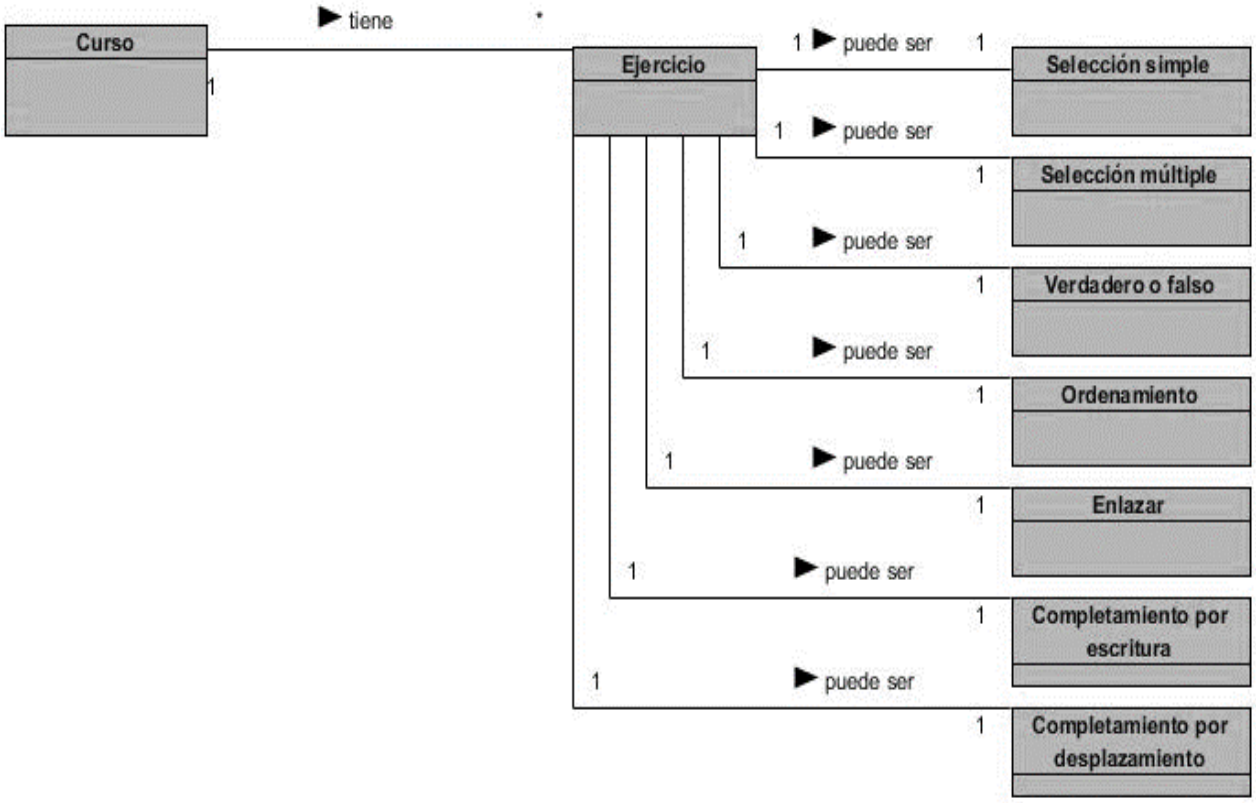


Figura 4. Modelo de dominio

2.3 Descripción de la propuesta de solución

La siguiente propuesta de solución consiste en desarrollar el módulo que permite exportar e importar las siguientes tipologías de ejercicios de la plataforma educativa Zera$_{v2.0}$: selección simple, selección múltiple, verdadero o falso, ordenamiento, enlazar columnas, completamiento por escritura, completamiento por desplazamiento.

Este módulo le permitirá al usuario contar con la opción de importar los ejercicios hacia la plataforma de otras que sigan el estándar IMS QTI $_{v2.0}$., posibilitando no tener que introducir los datos manualmente. De igual modo lograrán exportar los ejercicios facilitando almacenarlos y

compartirlos con usuarios de otras plataformas que utilicen el mismo estándar. Obteniendo como resultado un módulo que se ajuste a sus requerimientos y necesidades para lograr la interoperabilidad de los ejercicios de la Plataforma Educativa ZERA $_{v2.0}$.

2.4 Requisitos de software

Los requisitos para un sistema son la descripción de los servicios proporcionados por el sistema y sus restricciones operativas. Estos requisitos reflejan la necesidad de los clientes de un sistema que ayude a resolver algún problema como el control de un dispositivo, hacer un pedido o encontrar información. Es el proceso de descubrir, analizar, documentar y verificar estos servicios y restricciones. (Sommerville, 2007)

2.4.1 Requisitos funcionales

Lo requisitos funcionales de un sistema describen lo que el sistema debe hacer. Estos dependen del tipo de software que se desarrolle, de los posibles usuarios y del enfoque general tomado por la organización al tomar requisitos. Estos requisitos funcionales definen los resultados que el sistema debe proporcionar. (Sommerville, 2007)

La propuesta de solución de la presente investigación debe cumplir con los requisitos funcionales siguientes:

RF1. Importar ejercicios. El sistema permite importar los ejercicios hacia un curso determinado.

RF2. Exportar ejercicios. El sistema permite exportar todos los ejercicios antes creados en un curso.

RF3. Exportar ejercicios en lote. El sistema permite exportar varios ejercicios en una sola acción.

2.4.2 Requisitos no funcionales

Los requisitos no funcionales, como su nombre sugieren, son aquellos requisitos que no se refieren directamente a las funciones específicas que proporcione el sistema, sino a las propiedades emergentes de este como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento. De forma alternativa, define las restricciones del sistema como la capacidad de los dispositivos de entrada/salida y las representaciones de datos que se utilizan en las interfaces

Seguridad

- Establecer permisos por roles para el acceso al sistema. Autenticación basada en las bondades del dominio UCI. Validación de datos en el cliente, para evitar estados inconsistentes en la información y posibles ataques al sistema.

Usabilidad

- Cumplir con las pautas de diseño establecidas en la Estrategia Marcaria de la Universidad. Cuando se crea/actualiza/elimina un elemento, así como el cancelar, se muestra un mensaje con el resultado de la acción.

Hardware

- Servidor de bases de datos relacional PostgreSQL 9.4 con memoria RAM: 16 GB, Disco Duro: 100 GB y microprocesador: 6 x 800 GHz.

- Servidor de aplicaciones NGINX: 2.7 con memoria RAM: 16 GB, Disco Duro: 500 GB y microprocesador: 6 x 800 GHz.

Portabilidad

- El sistema será accesible desde estaciones de trabajo de escritorio, laptop, Tablet y Smartphone. Estos deberán contar con un navegador web moderno (Navegadores web: Firefox (v10 en adelante) y Chrome (v20. en adelante). Historias de usuario

Una historia de usuario es una representación de un requisito escrito en una o dos frases utilizando el lenguaje común del usuario. Las historias de usuario bien escritas son esenciales para el Desarrollo Ágil. Deben ser independientes entre sí; se debe poder negociar los detalles entre los usuarios y los desarrolladores; las historias deben tener valor para el cliente; deben ser lo suficientemente claras para que los desarrolladores puedan estimarlas; deben ser pequeñas; y deben poder probarse usando los casos de prueba pre-definidos. (Cohn, 2004) A continuación se muestran las de exportar e importar ejercicios y la de exportar ejercicios en lote. Para la realización de las historias se tuvo en cuenta un plan de riesgo relacionado con el proyecto. El plan se puede consultar en el Anexo 1: Plan de riesgo.

Tabla 1. Historia de usuario. Exportar ejercicios

Número:1	**Nombre del requisito:** Exportar ejercicios
Programador: Daimel Noda Cabrera	**Iteración Asignada:** 2da
Prioridad: Alta	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: 1, 2, 3,4, 5, 6,7, 8, 9	**Tiempo Real:** 2 días

Descripción:

1. **Objetivo:**

Permitir exportar un ejercicio.

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

Para exportar el ejercicio hay que:

En la lista de ejercicios creados seleccionar el botón exportar de las acciones del ejercicio.

3. **Comportamientos válidos y no válidos (flujo central y alternos):**

4. **Flujo de la acción a realizar:**

El sistema debe permitirle al usuario seleccionar el botón exportar.

El sistema permite seleccionar la ruta donde se exportará el ejercicio.

El sistema debe permitirle al usuario cancelar la opción de exportar el ejercicio.

Observaciones: Exportar ejercicios se cumple tanto para los ejercicios de selección simple como para los de selección múltiple, de verdadero o falso, de ordenamiento, de enlazar, de completamiento por escritura y de completamiento por desplazamiento.

Prototipo elemental de interfaz gráfica de usuario:

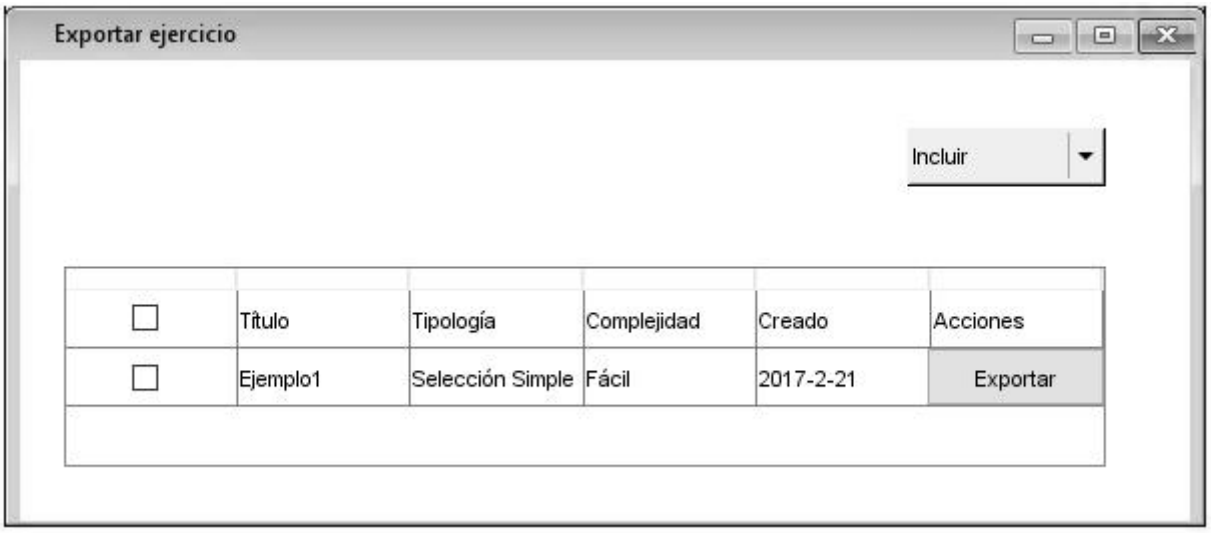


Tabla 2. Historia de usuario. Importar ejercicios

Número:2	**Nombre del requisito:** Importar ejercicios
Programador: Daimel Noda Cabrera	**Iteración Asignada:** 1ra
Prioridad: Alta	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: 1, 2, 3,4, 5, 6,7, 8, 9	**Tiempo Real:** 2 días

Descripción:

1. **Objetivo:**

Permitir importar un ejercicio.

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

Para importar el ejercicio hay que:

Seleccionar el botón importar.

3. **Comportamientos válidos y no válidos (flujo central y alternos):**

4. **Flujo de la acción a realizar:**

El sistema debe permitirle al usuario seleccionar el botón Importar.

El sistema permite seleccionar el ejercicio a importar.

El sistema muestra un formulario para que el usuario compruebe el ejercicio importado.

El sistema debe permitir guardar el ejercicio un vez comprobado.

Observaciones: Importar ejercicios se cumple tanto para los ejercicios de selección simple como para los de selección múltiple, de verdadero o falso, de ordenamiento, de enlazar, de completamiento por escritura y de completamiento por desplazamiento.

Prototipo elemental de interfaz gráfica de usuario:

2.5 Descripción de la arquitectura

El concepto de arquitectura de software se refiere a la estructuración del sistema que se crea en etapas tempranas del desarrollo. Esta estructuración representa un diseño de alto nivel del sistema que tiene dos propósitos primarios: satisfacer los atributos de calidad (desempeño, seguridad, modificabilidad), y servir como guía en el desarrollo. El no crear este diseño desde etapas tempranas del desarrollo puede limitar severamente que el producto final satisfaga las necesidades de los clientes. Además, el costo de las correcciones relacionadas con problemas en la arquitectura es muy elevado. (Arquitectura de Software, 2015)

La arquitectura de la propuesta de solución está determinada por el uso de la tecnología Symfony 2.0 permitiendo el desarrollo de aplicaciones web siguiendo la arquitectura Modelo Vista Controlador (MVC).

El patrón Modelo-Vista-Controlador surge con el objetivo de reducir el esfuerzo de programación, necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos, a partir de estandarizar el diseño de las aplicaciones. Es un paradigma que divide las partes que

conforman una aplicación en el Modelo, las Vistas y los Controladores. Lo cual permite la implementación por separado de cada elemento, garantizando así la actualización y mantenimiento del software de forma sencilla y en un reducido espacio de tiempo. A partir del uso de *framework* basados en el MVC se puede lograr una mejor organización del trabajo y mayor especialización de los desarrolladores y diseñadores. (Telemática, 2012)

2.6 Patrones de diseño

Son principios generales de soluciones que aplican ciertos estilos que ayudan a la creación de software. Es una descripción de un problema y la solución a la que le da el nombre y que se puede aplicar en nuevos contextos. Muchos patrones ayudan a asignar responsabilidades a los objetos. Un patrón es un par/problema solución con nombre que se puede aplicar en nuevos contextos, con consejos acerca de cómo aplicarlo en nuevas situaciones y discusiones sobre sus compromisos. (Rojas, 2007)

Aunque una aplicación sea única, tendrá partes comunes con otras aplicaciones: acceso a datos, creación de objetos, operaciones entre sistemas, entre otras. En lugar de reinventar la rueda, podemos solucionar problemas utilizando algún patrón, ya que son soluciones probadas y documentadas por multitud de programadores. Programación SOLID, control de cohesión y acoplamiento o reutilización de código son algunos de los beneficios que podemos conseguir al utilizar patrones. (Patrones de diseño: qué son y por qué debes usarlos, 2014)

2.6.1 Patrones GRASP

Describen los principios fundamentales para asignar responsabilidades a los objetos. (Larman, 1999) Para el desarrollo de la presente investigación solo se utilizarán los siguientes patrones GRASP.

- **Experto.** Consiste en asignar una responsabilidad al experto en información, se le asigna la responsabilidad a la clase que tiene la información necesaria para llevar a cabo la responsabilidad. Este patrón se evidencia en la clase SimpleSelectExercice.

- **Bajo acoplamiento.** Consiste en asignar responsabilidades de manera que el
- acoplamiento (innecesario) se mantenga bajo. Este patrón se evidencia en la clase controladora que hereda de una única clase controladora generar.

- **Alta cohesión:** Consiste en asignar una responsabilidad de modo que la cohesión siga siendo alta. Este patrón se evidencia en la clase controladora, la cual tiene funcionalidades que se relacionan entre sí.

- **Controlador.** Consiste en asignar la responsabilidad de gestionar un mensaje de un evento del sistema a una clase que represente el sistema global, dispositivo o un subsistema (controlador de fachada), o represente un escenario de caso de uso en el que tiene lugar el evento del sistema (controlador de caso de uso o sesión). Este patrón se evidencia en la clase controladora: ejemplo ExerciceController

- **Creador:** permite crear objetos de una clase determinada. Es utilizado en la mayoría de las clases controladoras para crear instancias de formularios y entidades. Este patrón se evidencia en la clase ExerciceController.

2.6.2 Patrones Gof

Los patrones de diseño GoF, sus siglas significan Gang of Four (grupo de los cuatro) debido a sus creadores: Erich Gamma, Richard Helm, Ralph Johnson y John Vlisides. Estos patrones de diseño tuvieron un gran éxito en el mundo de la informática a partir de la publicación del libro Design Patterns (patrones de diseño) escrito a principios de los 90 y en el cual se recogen 23 patrones de diseño comunes.

Estos se dividen en 3 grupos de acuerdo a la naturaleza de cada uno:

- **De creación:** conciernen al proceso de creación de objetos.

- **Estructurales:** tratan la composición de clases y objetos.

- **Comportamiento:** caracterizan las formas en las que interactúan y reparten responsabilidades las distintas clases u objetos.

- **Singleton** (Instancia única): entra dentro del grupo de los de creación, garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Se evidencia en la creación de la clase controladora.
- **Observer** (Observador): entra dentro del grupo de los de comportamiento define una relación de uno a muchos entre objetos, de manera que cuando un objeto cambie de estado se notifique y actualicen automáticamente todos los objetos que dependen de él.

2.7 Modelo de clases del diseño

El modelo de diseño es un refinamiento y formalización adicional del modelo del análisis, donde se toman en cuenta las consecuencias del ambiente de implementación. El resultado del modelo de diseño son especificaciones muy detalladas de todos los objetos, incluyendo sus operaciones y atributos. El modelo de diseño se basa en el diseño por responsabilidades. (Areli, 2011)

2.7.1 Diagrama de clases del diseño

Un diagrama de clases de diseño (DCD) representa las especificaciones de las clases e interfaces de software en una aplicación. Entre la información general se encuentra:

- Clases, asociaciones y atributos.
- Interfaces, con sus operaciones y constantes.
- Métodos.
- Información acerca del tipo de los atributos.
- Navegabilidad.
- Dependencias.

A diferencia de las clases conceptuales del Modelo del Dominio, las clases de diseño de los DCD muestran las definiciones de las clases de software en lugar de los conceptos del mundo real.
A continuación, se muestran los diagramas de clases del diseño del exportar e importar ejercicios y el de exportar en lote.

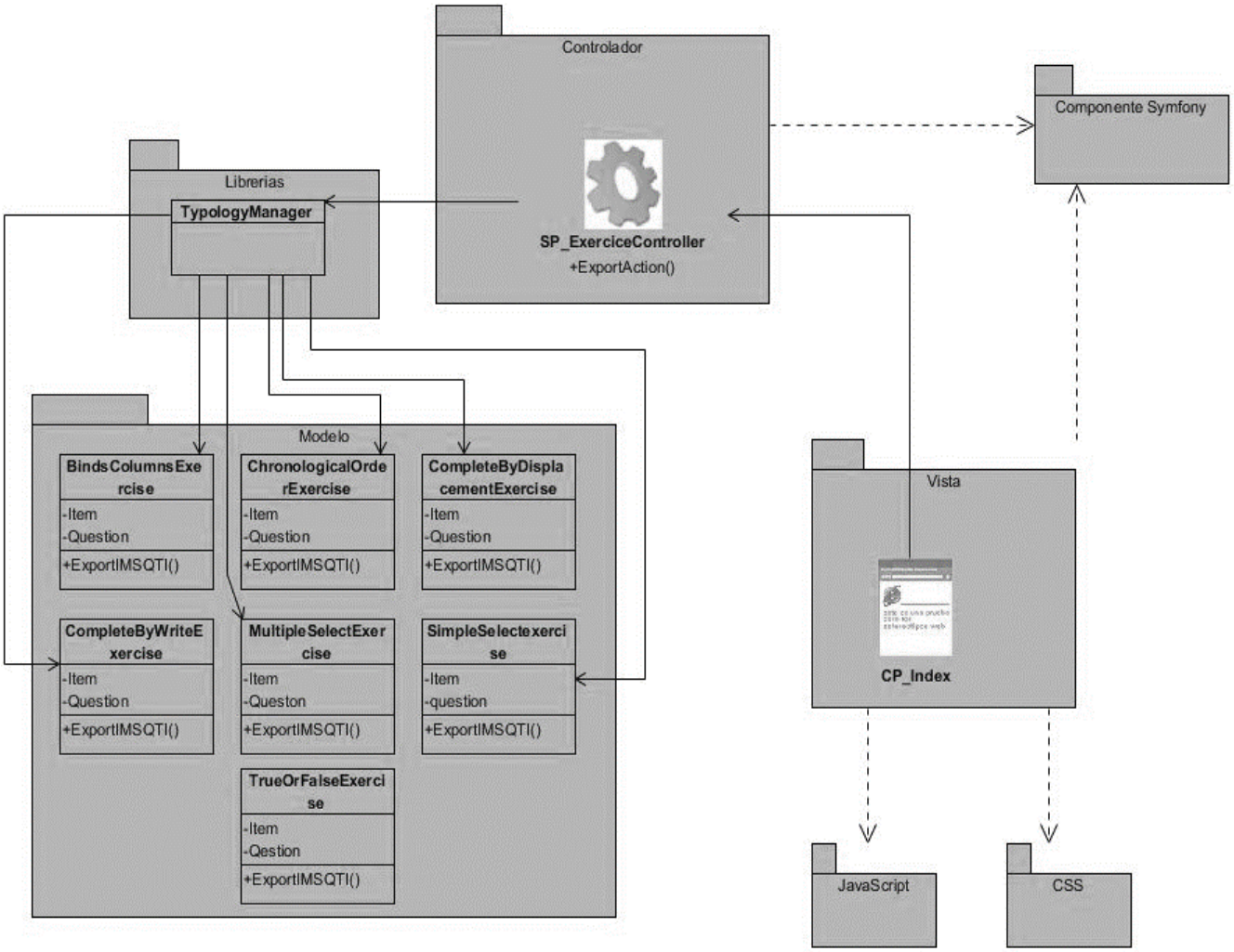


Figura 5. Diagrama de clases del diseño. Exportar ejercicio

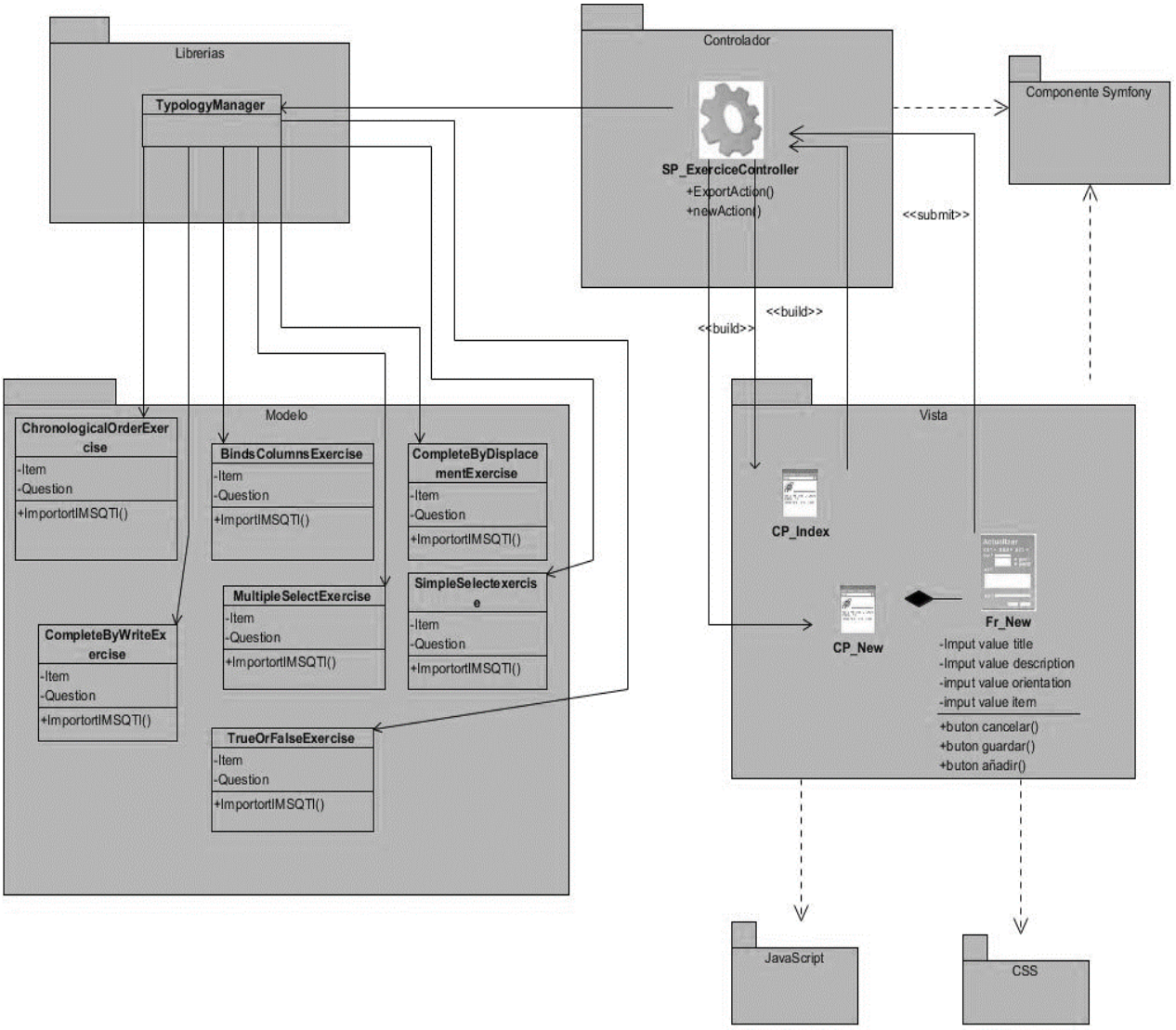


Figura 6. Diagrama de clases del diseño. Importar ejercicio

2.7.2 Diagrama de secuencia del diseño

El diagrama de secuencia del diseño detalla los modelos de análisis tomando en cuenta todas las implicaciones y restricciones técnicas. El propósito del diseño es especificar una solución que trabaje, pueda ser fácilmente convertida en código fuente y construir una arquitectura simple, así como también extensible. (Sifuentes, 2013).

A continuación, se muestran los diagramas de secuencia del exportar e importar ejercicios y el de exportar en lote.

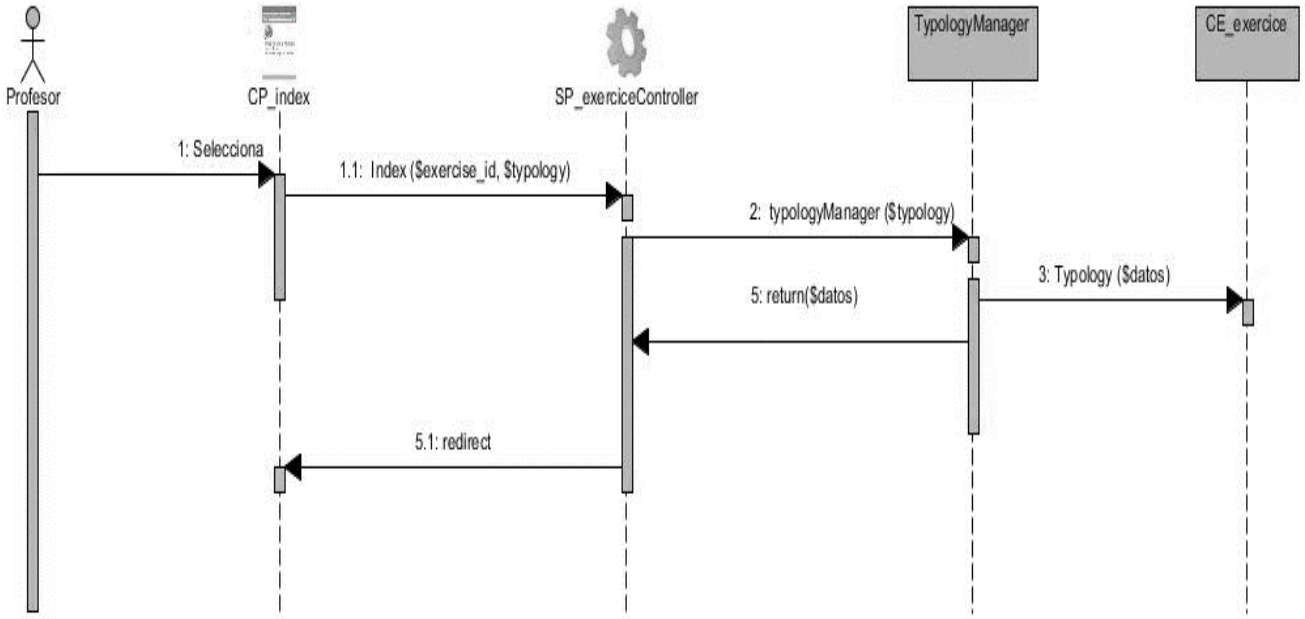


Figura 7: Diagrama de secuencia. Exportar ejercicio

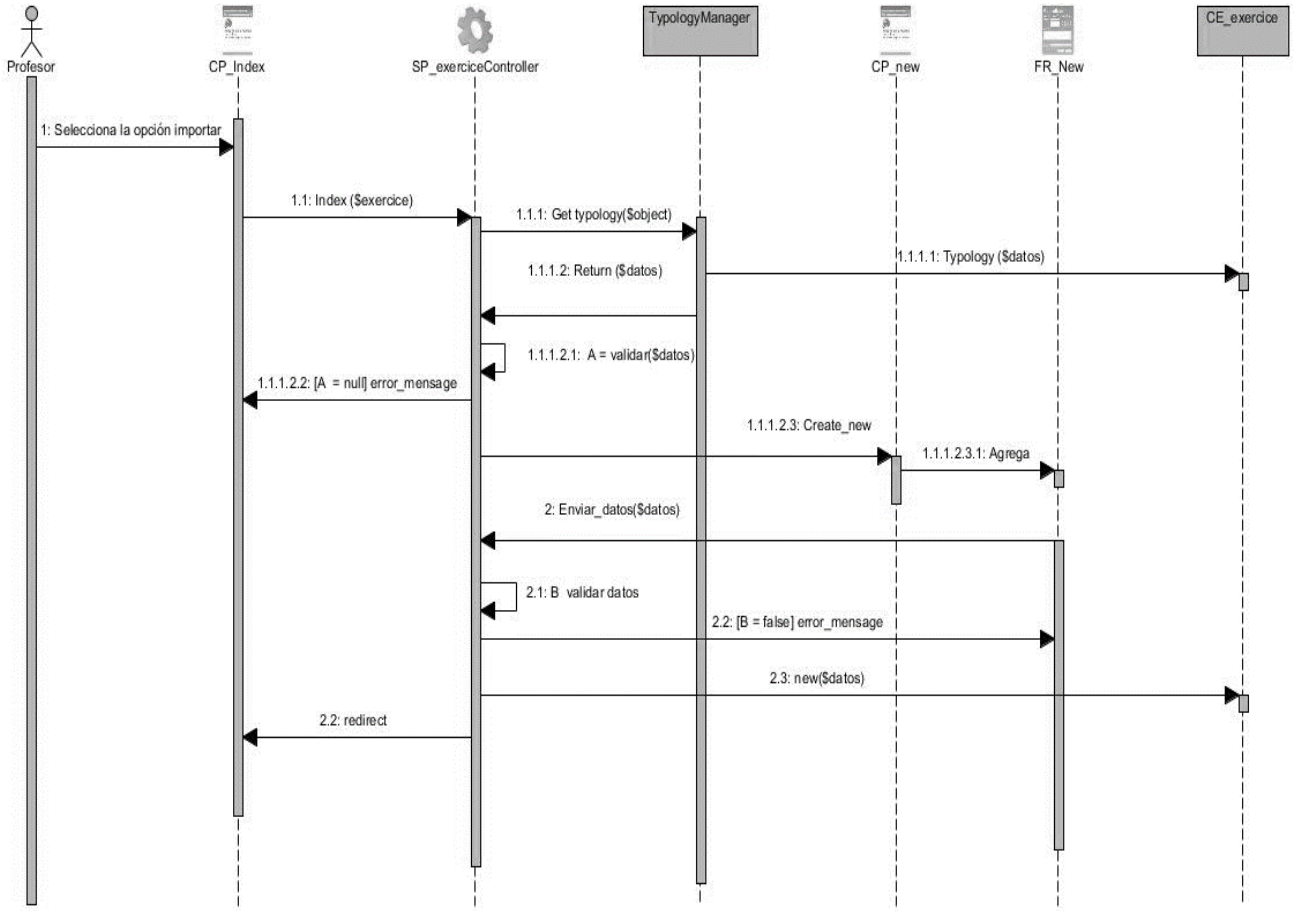


Figura 8: Diagrama de secuencia. Importar ejercicio

2.7.3 Diagrama de despliegue

El Diagrama de despliegue es un diagrama estructurado que muestra la arquitectura del sistema desde el punto de vista del despliegue, distribución de los artefactos del software en los destinos de despliegue. Los artefactos representan elementos concretos en el mundo físico que son el resultado de un proceso de desarrollo. Ejemplos de artefactos son archivos ejecutables, bibliotecas, archivos, esquemas de bases de datos, archivos de configuración, entre otros.

El destino de despliegue está generalmente representado por un nodo que es o bien de los dispositivos de hardware o bien algún entorno de ejecución de software. Los nodos pueden ser conectados a través de vías de comunicación para crear sistemas en red de complejidad arbitraria.

Los diagramas de despliegue pueden describir la arquitectura a nivel de especificación (también llamado nivel de tipo) o al nivel de instancia (de manera similar a los diagramas de clases y diagramas de objetos). (UML: Diagrama de Despliegue, 2013)

- Los diagramas de despliegue de nivel de especificación muestran una visión general del despliegue de los artefactos hacia los destinos de despliegue, sin hacer referencia a casos concretos de artefactos o nodos.

- Los diagramas de nivel de instancia muestran el despliegue de instancias de artefactos en instancias específicas de los destinos de despliegue. Se pueden utilizar por ejemplo para mostrar las diferencias existentes en nombres/identificaciones en ambientes de despliegue a desarrollo, de "staging" o de producción, entre construcciones específicas o servidores de despliegue o dispositivos.

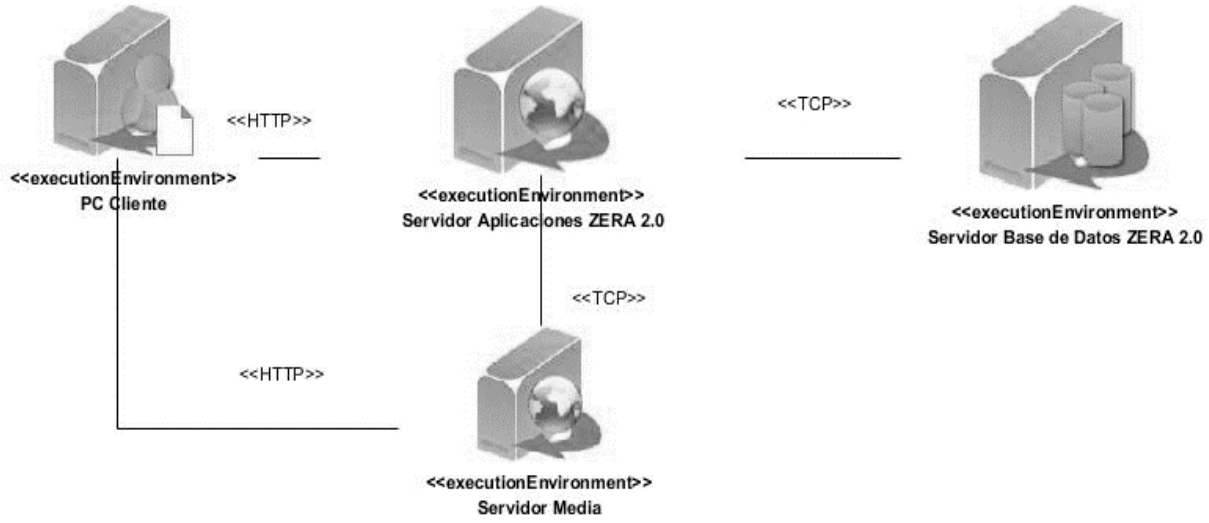


Figura 9: Diagrama de despliegue

PC Cliente: pc desde donde se podrá visualizar e interactuar con la plataforma a través de un navegador web (previamente instalado en la máquina).

Servidor Web: servidor Apache donde se encuentra desplegado el componente, al que se conectan los clientes por medio de sus estaciones de trabajo.

Servidor de BD: servidor PostgreSQL en el que se encuentran almacenados todos los datos persistentes del sistema.

Servidor media: donde se encuentran almacenados todos los datos multimedia.

Los protocolos utilizados son:

- **HTTP:** protocolo de comunicación estándar-básico que se utiliza en las arquitecturas web para la comunicación establecida entre el servidor web y las estaciones de los usuarios.
- **TCP/IP:** protocolo mediante el cual se realiza la comunicación entre el servidor web y el servidor de bases de datos; además de la comunicación entre el servidor web y el motor de evaluación.

2.7.4 Modelo de datos

El modelado de datos es el proceso de documentar un diseño de sistema de software complejo como un diagrama de fácil comprensión, usando texto y símbolos para representar la forma en que los datos necesitan fluir. El diagrama se puede utilizar como un mapa para la construcción de un nuevo software o para la reingeniería de una aplicación antigua. Es un conjunto de conceptos que pueden servir para describir la estructura de una Base de Datos, tipo de datos, y las relaciones que deben cumplirse para esos datos. Por lo general los modelos de datos contienen además un conjunto de operaciones básicas para especificar lecturas y actualizaciones de la base de datos. (Laurens, 2014)

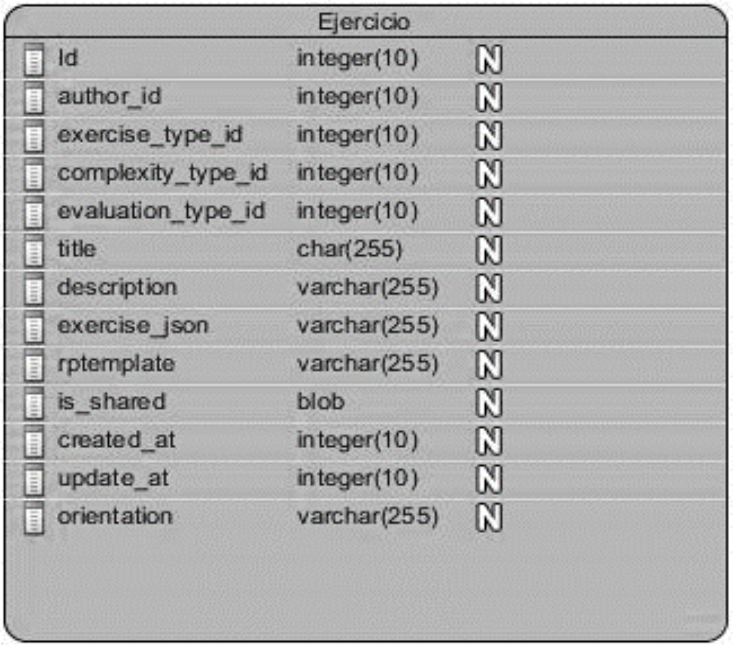


Figura 10: Modelo de datos

2.8 Conclusiones del capítulo

- En el capítulo se realizó el modelo de dominio para representar los conceptos que intervienen en el desarrollo de la propuesta de solución.
- Los requisitos funcionales y no funcionales identificados permitieron obtener una idea general de las funcionalidades que debe cumplir la aplicación.

- A partir de los requisitos funcionales se realizaron las historias de usuario, con el propósito de planificar el tiempo de desarrollo de las iteraciones y el orden en que se implementan.
- La arquitectura y los patrones de diseño seleccionados constituyeron una guía para la implementación de la propuesta de solución y se realizaron los diagramas de clase del diseño para que todas las funcionalidades se implementaran sin dificultad.

Capítulo 3: Implementación y pruebas

3.1 Introducción

Uno de los objetivos más importantes de este capítulo es implementar el módulo en forma de producto, para que se realice esta actividad los desarrolladores deben tener en cuenta toda la documentación reflejada anteriormente. El proceso se guiará por el análisis y diseño realizados, utilizándose como artefacto principal el diagrama de clases del diseño para darle al desarrollador una mejor visión del sistema. Durante este proceso se realizan todos los requerimientos funcionales teniendo en cuenta también los no funcionales para así llegar al producto final. Posteriormente se deben realizar las pruebas que permitan detectar su nivel de calidad, así como los errores cometidos en la implementación, para darle solución lo más pronto posible.

3.2 Modelo de implementación

La implementación es la actividad fundamental durante la fase de construcción, partiendo principalmente del resultado obtenido durante el flujo de trabajo de análisis y diseño. Durante esta etapa se desarrolla la arquitectura, se logra estructurar el sistema final, se organiza el código, se implementan los elementos del diseño, y se integran todos estos elementos para obtener un resultado satisfactorio que responda a lo definido durante las fases anteriores. Al llevar a cabo dicho modelo se establece la estructura de los elementos, basándose en las responsabilidades asignadas a los subsistemas de implementación y su contenido.

3.2.1 Diagrama de componentes

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Los componentes representan todos los tipos de elementos software que entran en la fabricación de aplicaciones informáticas. Pueden ser simples archivos, paquetes de Ada, bibliotecas cargadas dinámicamente, etc. Las relaciones de dependencia se utilizan en los diagramas de componentes para indicar que un componente utiliza los servicios ofrecidos por otro componente. (GARCIA SAAVEDRA MADELINE TRACY, 2015)

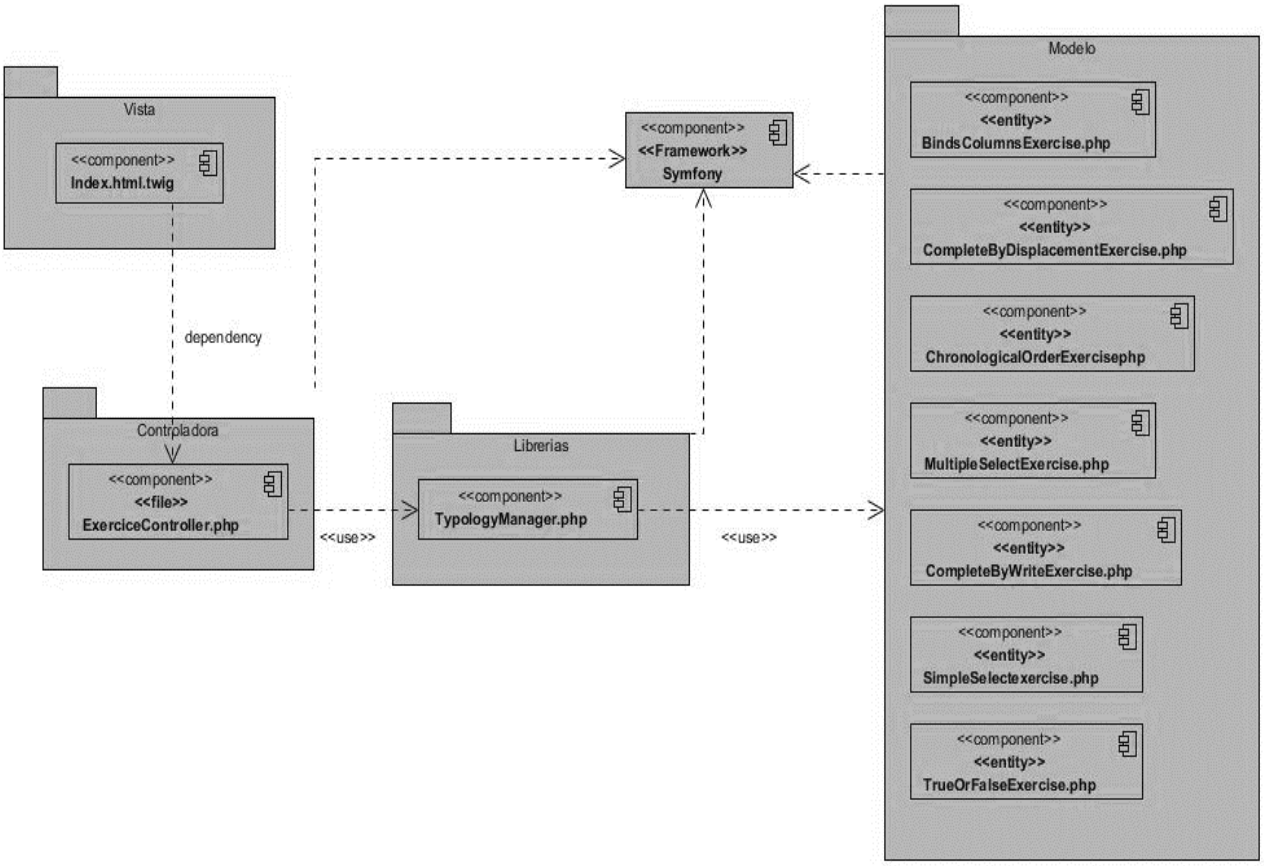


Figura 11: Diagrama de componentes

3.2.2 Estructura de la propuesta de solución

La propuesta de solución emplea la arquitectura MVC, la cual divide las clases en tres capas: los modelos, las vistas y las controladoras. La utilización de esta arquitectura le permite a los desarrolladores mantener una mayor organización del código. A continuación, se muestra la estructura de carpetas empleadas para la propuesta de solución.

El proyecto se encuentra dentro de la carpeta *ExerciseBundle*, esta almacena una serie de subcarpetas como: *Controller* que contiene las clases controladoras, ejemplo: *ExerciseController.php*. La carpeta *Model* posee subcarpetas que agrupan los modelos del proyecto, ejemplo: *Typology* con una clase por cada tipología como *SimpleSelectExercise.php.* Además, la carpeta *Resources* posee todos los recursos que utiliza el bundle incluyendo las vistas, las plantillas XML y las clases JavaScript. Las plantillas XML se pueden acceder a través del directorio *public/ xml.* Dentro de la carpeta *views* se puede acceder a las vistas, esta contiene una subcarpeta nombrada *Default*. En la misma se genera la clase *index.html.twig* que responde a la vista de la propuesta de solución.

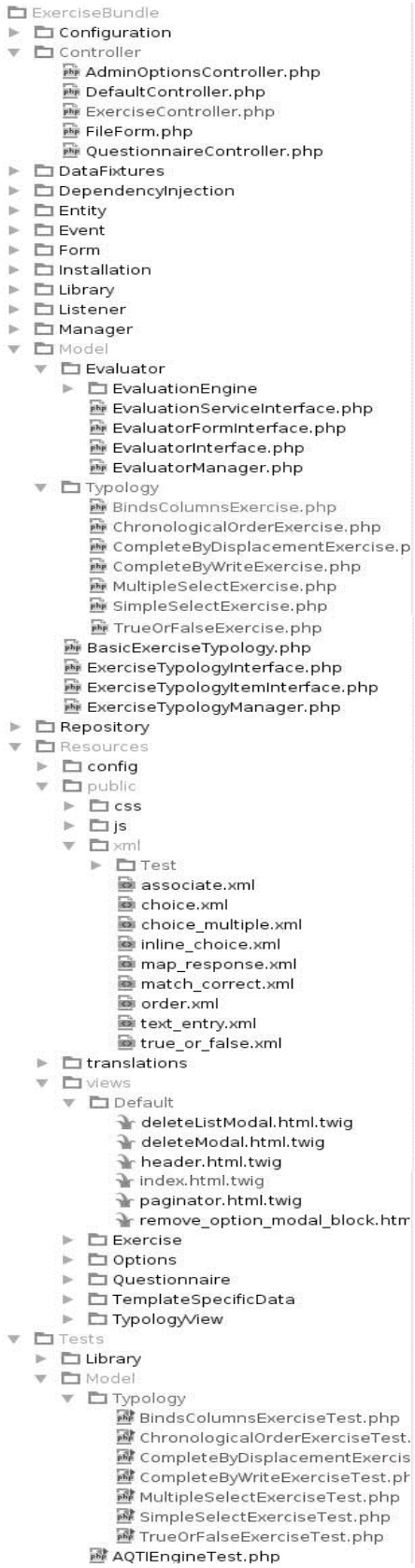


Figura 12: Estructura de carpetas de la propuesta de solución

3.3 Estilos y estándares de codificación

La forma de escribir código es propia de cada programador y completamente diferente a la forma de cualquier otro. Sin embargo, se definen estándares de codificación porque un estilo de programación homogéneo en un proyecto permite que todos los participantes lo puedan entender en menos tiempo y que el código en consecuencia sea mantenible. Para la implementación de la propuesta de solución se deciden las siguientes normas de codificación.

La propuesta de solución se rige por los estilos y estándares de codificación definidos por el proyecto Zera$_{v2.0}$. mediante los documentos PSR-0, PSR-1, PSR-2, PSR-4 y los desarrolladores del proyecto. Algunas de estas normas son las siguientes: los comentarios que emplean una sola línea, empiezan con /* y terminan en */, adicionar un espacio después de cada delimitador coma y agregar una línea en blanco antes de cada sentencia return. También se define que las variables deben estar declaradas con nombres sugerentes y cada clase tiene que estar comentariada para explicar su funcionalidad. Además, para facilitar la lectura del código se deben añadir líneas en blanco separando bloques de código grandes o lógicos.

3.4 Diseño de caso de prueba

El diseño de casos de prueba permite diseñar las entradas y las salidas esperadas para probar el sistema. El objetivo es diseñar casos de prueba que, sistemáticamente, saquen a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y de esfuerzo. Los casos de prueba se desarrollan para definir las cosas que es necesario validar a fin de asegurar que el sistema funciona correctamente y está construido con un alto nivel de calidad. Una suite de pruebas es una colección de casos de prueba que se agrupan para ejecutar pruebas. (Maricarmen Sotelo Nájera, 2013)

A continuación, se muestran los diseños de casos de pruebas de los requisitos exportar e importar y el de exportar en lote.

Tabla 3. Diseño de casos de prueba. Exportar ejercicio

Descripción general: Permitir exportar ejercicio.			
Condiciones de ejecución: para exportar el ejercicio hay que: -crear los ejercicios.			
Nombre de la sección: SC1 Exportar ejercicio.			
Escenarios	Descripción	Respuesta del sistema	Flujo central

EC1.1 Exportar ejercicio	El usuario selecciona el botón exportar.	El sistema permite exportar el ejercicio y brindar además la posibilidad de realizar una de las siguientes opciones: • Aceptar • Cancelar	Curso/ Crear/Ejercicio/ Exportar
EC1.2 Opción Aceptar	El usuario selecciona el directorio donde desea guardar el ejercicio y la opción Aceptar.	El sistema permite guardar el ejercicio.	Curso/ Crear/Ejercicio/ Exportar/Aceptar
EC1.3 Opción Cancelar	El usuario selecciona la opción Cancelar	El sistema regresa a la lista de ejercicios.	Curso/ Crear/Ejercicio/ Exportar/Cancelar

Tabla 4. Diseño de casos de prueba. Importar ejercicio

Descripción general: Permitir importar ejercicio.			
Condiciones de ejecución: para importar el ejercicio hay que: -tener el ejercicio en un xml.			
Nombre de la sección: SC1 Importar ejercicio.			
Escenarios	Descripción	Respuesta del sistema	Flujo central
EC1.1 Importar ejercicio	El usuario selecciona la opción importar.	El sistema permite importar el ejercicio y brinda además la posibilidad de realizar una de las siguientes opciones: • Aceptar • Cancelar	Curso/ Crear/Ejercicio/ Importar

		El sistema permite hacerle cambios al ejercicio importado perteneciendo a la prueba de Crear Ejercicio.	
EC1.2 Opción Aceptar	El usuario selecciona el fichero del ejercicio y la opción Aceptar.	El sistema permite importar el ejercicio.	Curso/ Crear/Ejercicio/ Importar/Aceptar
EC1.3 Opción Cancelar	El usuario selecciona la opción Cancelar.	El sistema regresa a la lista de ejercicios.	Curso/ Crear/Ejercicio/ Importar/Cancelar

3.5 Pruebas de software

Las pruebas de software comprenden el conjunto de actividades que se realizan para identificar posibles fallos de funcionamiento, configuración o usabilidad de un programa o aplicación. La fase de pruebas es una de las más costosas del ciclo de vida del software. Obviamente, se aplican diferentes técnicas de prueba a cada tipo de producto de software. Maximiza la cantidad de defectos descubiertos y permite que los desarrolladores los corrijan e incrementen la confiabilidad del sistema. Las pruebas no son determinantes y hay que realizarlas bajo restricciones de tiempo y presupuesto. (Usaola, 2009)

3.5.1 Pruebas unitarias

Para desarrollar la prueba unitaria se deben tener claros los siguientes aspectos:

- Los datos de entrada son conocidos por el Tester o Analista de Pruebas y estos deben ser preparados con minuciosidad, ya que los resultados de las pruebas dependen de estos.

- Se debe conocer qué componentes interactúan en cada caso de prueba.

- Se debe conocer de antemano qué resultados debe devolver el componente según los datos de entrada utilizados en la prueba.

- Finalmente se deben comparar los datos obtenidos en la prueba con los datos esperados, si son idénticos se puede decir que el módulo superó la prueba y se comienza con la siguiente. (Oré, 2009)

Las pruebas unitarias son realizadas con la herramienta phpunit el cual es un entorno para realizar pruebas unitarias en el lenguaje de programación PHP. Además, Symfony2 integra una librería independiente de phpunit para desarrollar test. A continuación, se evidencia la prueba unitaria del método exportar de la tipología de selección simple.

```
public function testExportIMSQTI()
{
    $this->document->loadXml(
        file_get_contents($this->path . 'bundles/exercise/xml/Test/ExportSimpleSelectTest.xml')
    );
    $rpTemplate = $this->document->saveXML();
    $imput      = $this->entity;
    $output     = $this->simpleSelectExercise->exportIMSQTI($imput);
    $this->assertEquals($rpTemplate, $output);
}
```

Figura 13: Prueba unitaria exportar selección simple

Resultados de las pruebas unitarias

Las pruebas unitarias fueron realizadas a las funcionalidades de exportar e importar sumando un total de catorce pruebas unitarias corrigiéndose los errores encontrados durante el transcurso del desarrollo del módulo.

```
Configuration read from /media/truecrypt1/zeraDeveloper/app/phpunit.xml.dist

.................

Time: 4.37 seconds, Memory: 16.50Mb

OK (17 tests, 14 assertions)
daimel@daimel-Presario-CQ57-Notebook-PC:/media/truecrypt1/zeraDeveloper$
```

Figura 14: Resultado de las pruebas unitarias

Pruebas de aceptación

Dentro del tipo de pruebas que se tiene para validar el software, una de las más importantes son las de aceptación. Son aquellas pruebas que son diseñadas por el propio equipo de desarrollo en base a los requisitos funcionales especificados en la fase de análisis para cubrir todo ese espectro, y ejecutadas por el propio usuario final, no por todos evidentemente, pero sí por una cantidad de usuarios finales significativo que den validez y conformidad al producto que se les está entregado en base a lo que se acordó inicialmente. (Pruebas de aceptación en sistemas navegables, 2010)

Resultados de las pruebas de aceptación

Las pruebas de aceptación fueron realizadas a todos los componentes de la propuesta de solución, realizándose 3 iteraciones arrojando los siguientes resultados: se encontraron un total de 25 no conformidades, dentro de las cuales nueve clasifican de complejidad alta, seis de complejidad media y diez de complejidad baja siendo corregidas todas estas no conformidades al final de cada iteración.

Las no conformidades se clasificaron según su complejidad definiéndose de la siguiente forma:

- Alta son los errores en el código o errores de funcionalidad.
- Media son los errores de ortografía o de validación.
- Baja son los errores de interfaz.

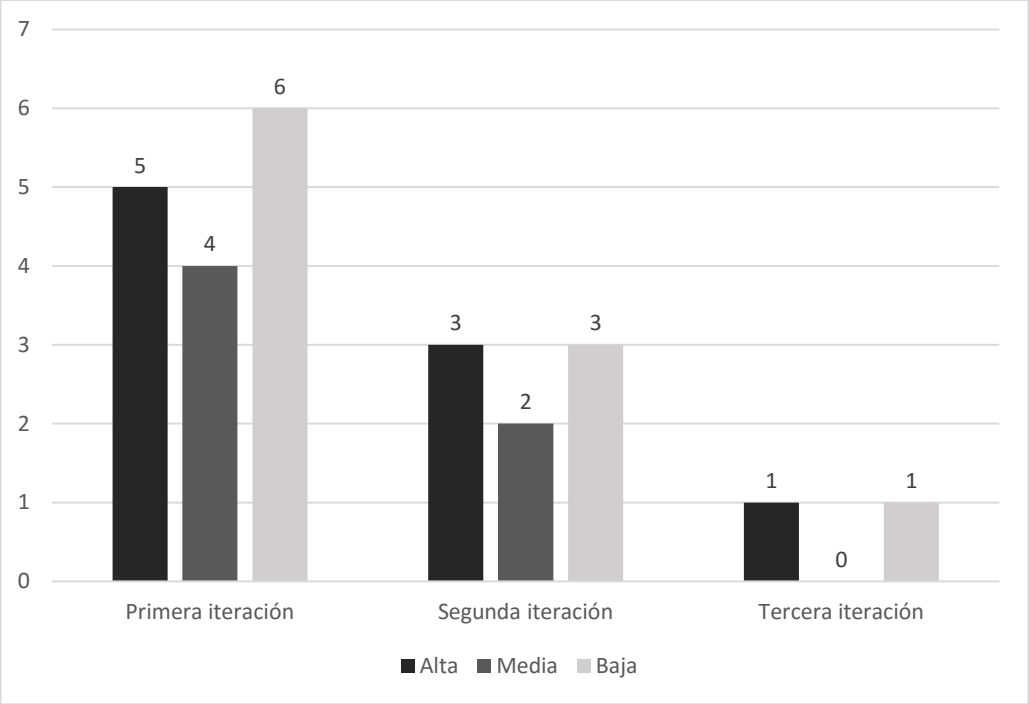


Figura 15: No conformidades encontradas en las pruebas de aceptación

3.6 Conclusiones del capítulo

- Se generó el diagrama de componentes para entender la relación de dependencia y uso de los componentes del sistema.
- Los estilos y estándares de codificación permitieron organizar el código de la propuesta de solución.
- Se realizaron diseños de casos de pruebas posibilitando identificar los errores del programa. Además, se realizaron un total de 14 pruebas unitarias a la aplicación con la herramienta Phpunit, de las cuales todos los errores encontrados por esta fueron corregidos demostrando el correcto funcionamiento del sistema.
- Las pruebas de aceptación realizadas en 3 iteraciones al módulo permitieron detectar 25 no conformidades que fueron corregidas al final de cada iteración, mejorando la funcionalidad del mismo.

Conclusiones generales

El presente trabajo ha dado cumplimiento a los objetivos trazados de la investigación, posibilitando obtener las siguientes conclusiones:

- El estudio sobre plataformas educativas que logran interoperabilidad entre sus ejercicios con otras plataformas, permitió definir las funcionalidades de exportar e importar siguiendo el estándar IMS QTI v2.0, así como evidenciar los distintos lenguajes, tecnologías y herramientas HTML v5, CCS v3, JavaScript y PHP v7; los framework de desarrollo Symfony, Bootstrap y JQuery, y el entorno de desarrollo Netbeans v8.0 para el desarrollo del mismo.

- En la fase de análisis y diseño quedó documentado cada uno de los artefactos definidos por la metodología AUP-UCI, los cuales facilitaron el desarrollo del módulo para contribuir a la interoperabilidad de los ejercicios en la Plataforma Educativa ZERA v2.0.

- Se obtuvo un módulo que contribuye a la interoperabilidad de los ejercicios de la Plataforma Educativa ZERA v2.0 permitiendo intercambiar sus ejercicios con otras plataformas que utilicen el estándar IMS QTI v2.0, ejemplo Moodle v 2.x, Zakai y ZERA v1.0

- Se aplicaron al módulo desarrollado 14 pruebas unitarias haciendo uso de la herramienta Phpunit corrigiéndose los errores encontrados durante el desarrollo de la aplicación. Se realizaron además pruebas de aceptación en 3 iteraciones corrigiéndose un total de 25 no conformidades, las cuales permitieron garantizar el correcto funcionamiento del módulo.

Recomendaciones

A partir del trabajo realizado se recomienda al proyecto Plataforma Educativa ZERA v2.0 para la continuidad de la presente investigación:

- Actualizar el motor de evaluaciones para utilizar la versión 2.1 de IMS QTI.
- Implementar soporte para la utilización de otros estándares de interoperabilidad para los ejercicios tales como SCORM.

Referencias bibliográficas

Álvarez, Miguel Ángel. 2001. desarrolloweb.com. [En línea] 2001. https://desarrolloweb.com/articulos/que-es-html.html.

—. 2009. *Manual de jQuery.* 2009.

Angelfire. 2011. geek the planet. [En línea] 2011. [Citado el: 28 de 2 de 2017.] http://geektheplanet.net/5462/patrones-gof.xhtml.

Areli, Tapia Hernández. 2011. *Fundamentos de ingeniería de software.* 2011.

Arquitectura de Software. Cervantes, Humberto. 2015. 27, 2015.

Bautista, JOSE M. 2017. Ingeniería de Software. *Programación extrema XP.* [En línea] 2017. [Citado el: 23 de enero de 2017.] http://ingenieriadesoftware.mex.tl/52753_XP---Extreme-Programing.html.

Código, Editores de. 2014. Editores de Código. [En línea] 2014. http://www.editoresdecodigo.com/2014/06/descargar-phpstorm-full-ide-para-php-y-mas.html.

Cohn, Mike. 2004. User Stories Applied. 2004.

Craig Larman, Prentice Hall. 2003. UML y Patrones. 2003.

Damian. 2010. CSS 3 HTML 5. [En línea] 23 de 11 de 2010. http://html5.dwebapps.com/que-es-css3/.

definición de framework. Rouse, Margaret. 2016. 2016.

Diagramas de colaboración. María Isabel Sánchez Segura, Arturo Mora Soto. 2012. 2012.

Fabien Potencier, François Zaninotto. 2008. *Symfony la guía definitiva.* 2008.

Flores, Araceli Soledad Domínguez. 2013. Unidad 1: UML y el proceso unificado. *Desarrollo e implementación de Sistemas de Información.* 2013.

GARCIA SAAVEDRA MADELINE TRACY, GONZALES SOTO CLAUDIA AURELIA, SIERRA ESTEVEZ PAOLA VANINA, YAPUCHURA VASQUEZ GREDTZEL MARIEL. 2015. *Diagrama de componentes definición.* 2015.

Gimeno, Juan Manuel. 2011. *Introducción a netbeans.* 2011.

Giulio, Carlo Ignacio. 2016. Sakai LMS: Abierto. [En línea] 2016. http://anisakai.es/wp-content/uploads/2014/04/Why_SAKAI-Espa%C3%B1ol.pdf.

IMS QUESTION & TEST INTEROPERABILITY SPECIFICATION. 2010. 2010.

Irina Ivis Santiesteban Pérez, Miguel Medina Ramírez. 2011. *Desarrollo de funcionalidades que faciliten al docente su preparación y el control del aprendizaje de los estudiantes en la plataforma educativa ZERA.* 2011.

jQuery: Qué es, Orígenes, Ventajas y Desventajas. Duarte, Eugenio. 2013. 2013.

kramer. 2011. MODELO DE ANÁLISIS. [En línea] 23 de 5 de 2011. [Citado el: 9 de 3 de 2017.] https://mundokramer.wordpress.com/2011/05/20/modelo-de-analisis-software/.

Larman, Craig. 1999. *UML y Patrones.* 1999.

Laurens, Yenifer. 2014. Modelo Conceptual de. [En línea] 2014. http://www.ciberesquina.una.edu.ve/tutorialdbd/documentos/modelosdedatos-enlace.pdf.

María. 2016. Puntoabierto. [En línea] 2016. puntoabierto.net/que-es-bootstrap-y-cuales-son-sus-ventajas.

Maria Vidal Ledol, Freddy Gomez Martinez, Alina M Ruiz Piedra. 2010. [En línea] 2010. http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S0864-21412010000100012.

Maricarmen Sotelo Nájera, Livingstone Asafaad Vázquez Sánchez, Angelo Marlon Montoya Guerra. 2013. Técnicas de diseño de casos de prueba. [En línea] 7 de 5 de 2013. https://prezi.com/rbzzdkq_aobm/tecnicas-de-diseno-de-casos-de-prueba/.

Martínez Usero, J. A. 2004. *La necesidad de interoperabilidad de la información en los servicios de administración electrónica: xml, una posible solución.* 2004.

Martínez, Víctor. 2010. Trazosweb. [En línea] 1 de 2 de 2010. [Citado el: 14 de 3 de 2017.] http://www.trazos-web.com/2010/02/01/html5-que-es-y-como-usarlo/.

Mayor, Alicia Cañellas. 2014. Centro de comunicación y pedagogía. [En línea] 2014. http://www.centrocp.com/lms-y-lcms-funcionalidades-y-beneficios/.

Merino, Julián Pérez Porto y María. 2012. Definiciones: Definición de lenguaje de programación. [En línea] 2012. (http://definicion.de/lenguaje-de-programacion/).

Metodologías ágiles para el desarrollo de software: extreme Programming (XP). Letelier, Patricio. 2006. 26, Buenos Aires: s.n., 2006, Vol. 5. ISSN 1666-1680.

Moodle en la formación inicial del profesorado. Gorospe, José Miguel Correa. 2005. 1, 2005, Vol. 4.

Oré, Alexander. 2009. CalidadySoftware.com. [En línea] 2009. [Citado el: 9 de 5 de 2017.] http://www.calidadysoftware.com/testing/pruebas_unitarias2.php.

Paradigm, Visual. 2015. Visual Paradigm. [En línea] 2015. [Citado el: 13 de enero de 2016.] http://www.visual-paradigm.com/aboutus/newsreleases/vpuml80.jsp.

Patrones de diseño: qué son y por qué debes usarlos. 2014. 2014.

Plataforma educativa ZERA: modelo de adaptación de contenidos sensible al contexto. Yerandy Manso Guerra, Roxana Cañizares Gonzáles, Juan Pedro Febles Rodríguez. 2015. 27, 2015.

Pruebas de aceptación en sistemas navegables. J. Ponce, F.J. Domínguez-mayo, M.J. Escalona, M. Mejías, 2010. 3, Sevilla: s.n., 2010, Vol. 6.

Riehle, Dirk. 2000. *Framework Design.* 2000.

Rojas, Juan Carlos Olivares. 2007. Tecnología nacional de México. *Patrones de diseño.* [En línea] 2007. [Citado el: 27 de 2 de 2017.] http://dsc.itmorelia.edu.mx/~jcolivares/courses/dp07b/patrones.pdf.

Ruiz, Palacios. 2013. Especificación y desarrollo de un sistema basado en 9 tecnologías semánticas para la integración e interoperabilidad de aplicaciones heterogéneas. [En línea] 2013.: http://e-archivo.uc3m.es/handle/10016/17159.

Sakai – Plataforma educativa de código abierto. Alvarez, Margie. 2010. 2010.

Sánchez, Tamara Rodríguez. 2014. *Metodología de desarrollo para la actividad productiva de la UCI.* La Habana: s.n., 2014.

Sifuentes, Luis Ivan. 2013. [En línea] 19 de 5 de 2013.

Sommerville, Ian. 2007. Ingenieria del Software Séptima Edición. 2007.

Telemática. Yanette Díaz González, Yenisleidy Fernández Romero. 2012. 1, 2012, Vol. 11.

UML, Diagrama de Despliegue. 2007. sparx. [En línea] 2007. http://www.sparxsystems.com.ar/resources/tutorial/uml2_deploymentdiagram.html.

UML: Diagrama de Despliegue. Sarmiento, Joana. 2013. 2013.

Usaola, Dr. Macario Polo. 2009. *Mantenimiento Avanzado de Sistemas de.* 2009.

Valdés, Damián Pérez. 2007. Maestros del web. [En línea] 3 de 7 de 2007. [Citado el: 14 de 3 de 201.] http://www.maestrosdelweb.com/que-es-javascript/.

Venemedia. 2014. [En línea] 16 de 11 de 2014. [Citado el: 14 de 3 de 2017.] http://conceptodefinicion.de/php/.

Yolanda, Borja López. 2013. *Metodología Ágil de Desarrollo de Software-XP.* 2013.

Albentia, Iratxo. 2008. Interoperabilidad: ¿A qué aspiramos cuando hablamos de ella? [En línea] 2008.

Baltasar Fernández Manjón, Pablo Moreno Ger, José Luis Sierra Rodríguez, Iván Martínez Ortiz. 2010. Educación. [En línea] 2010. http://ares.cnice.mec.es/informes/16/contenido/31.htm