Universidad de las Ciencias Informáticas

Facultad 4

Trabajo de Diploma para optar el título de Ingeniero en Ciencias Informáticas

Editor de metadatos basados en el estándar LOM para Plataforma Educativa Xauce ZERA 2.1.

Autor:

Alexnurín Columbié Méndez.

Tutores:

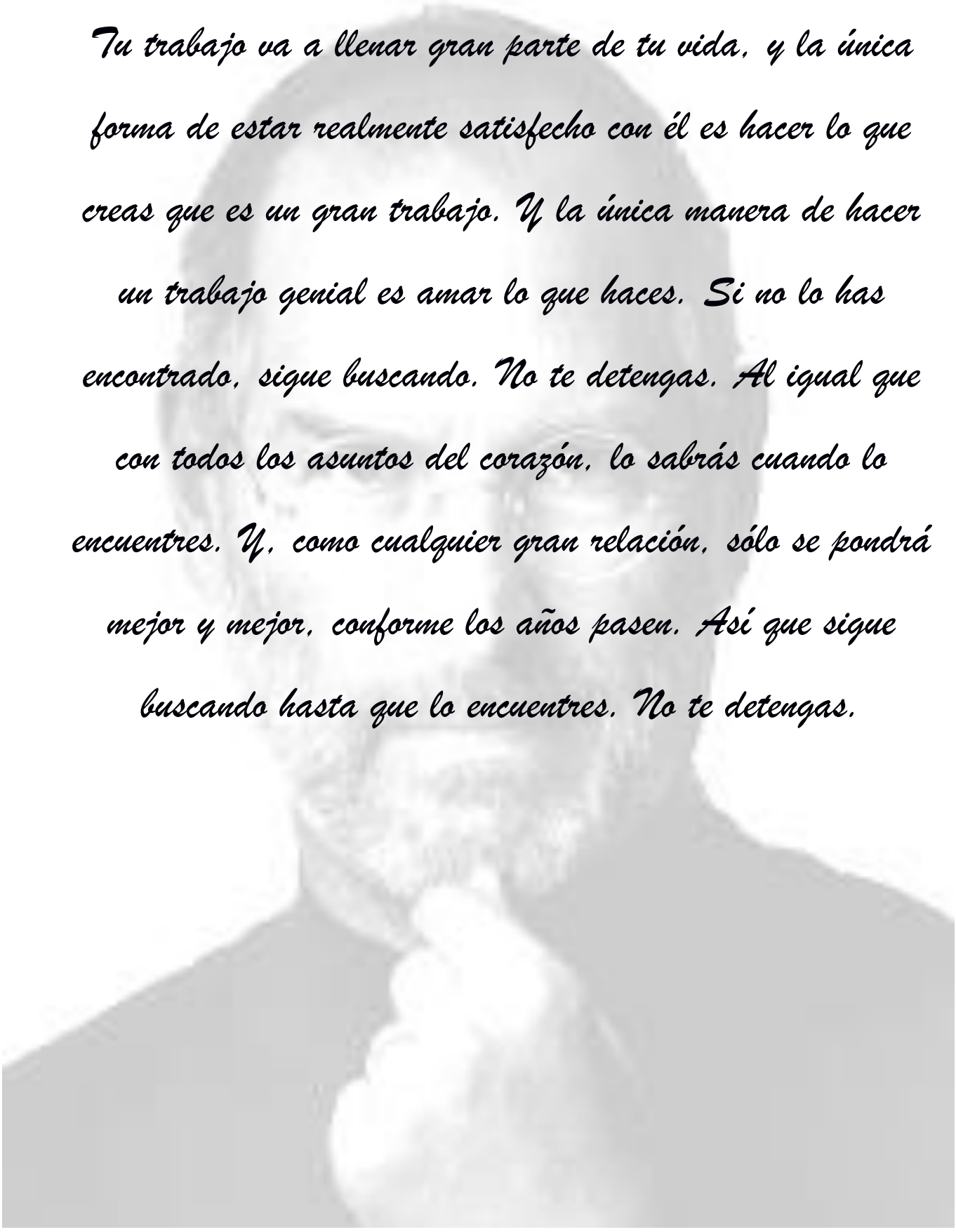
Ing. Jorge Delgado Rúa.

Ing. Agustín Castillo Cordero.

La Habana, mes de 2018

Año 60 de la Revolución.

Tu trabajo va a llenar gran parte de tu vida, y la única forma de estar realmente satisfecho con él es hacer lo que creas que es un gran trabajo. Y la única manera de hacer un trabajo genial es amar lo que haces. Si no lo has encontrado, sigue buscando. No te detengas. Al igual que con todos los asuntos del corazón, lo sabrás cuando lo encuentres. Y, como cualquier gran relación, sólo se pondrá mejor y mejor, conforme los años pasen. Así que sigue buscando hasta que lo encuentres. No te detengas.



Declaración de Autoría

Declaro ser el único autor del presente Trabajo de Diploma y reconozco al Centro de Tecnologías para la Formación (FORTES) de la Facultad 4 de la Universidad de las Ciencias Informáticas, los derechos patrimoniales del mismo, con carácter exclusivo, para que hagan el uso que estimen pertinente con el mismo.

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

_____

Firma del Autor

Alexnurín Columbié Méndez

_____ _____

Firma del Tutor **Firma del Tutor**

Ing. Agustín Castillo Cordero. Ing. Jorge Delgado Rúa.

Dedicatoria

Agradecimiento

Resumen

El Centro de Tecnologías para la Formación, de la Universidad de las Ciencias Informáticas, se especializa en la producción de aplicaciones y servicios informáticos orientados al sector educacional. El Centro, cuenta con varios proyectos de desarrollo bajo la arquitectura Xalix entre ellos La Plataforma Educativa Xauce ZERA 2.1. En esta se generan recursos educativos en los cuales no se permite su edición siguiendo un estándar específico. Con el propósito de facilitar el proceso, el presente trabajo propone un editor de metadatos basado en el estándar *Metadatos de Objetos Educativos* para contribuir a la caracterización de los recursos educativos y permitir su reutilización en distintas plataformas educativas. El editor incluye un conjunto de categorías que están relacionadas entre sí y que tienen como objetivo describir a los recursos. La validez del componente fue comprobada con la aplicación de las pruebas software, entre ellas, las pruebas de sistemas y de aceptación. De la aplicación de las pruebas de software a la propuesta de solución, se comprobó la correcta implementación de las funcionalidades y un alto grado de satisfacción por parte del cliente.

Palabras Claves: herramientas, información, metadatos.

Índice

Introducción:

El desarrollo de la tecnología y la ciencia ha provocado cambios en todas las esferas de la sociedad y una de ellas es la relacionada con el proceso de enseñanza-aprendizaje, en esta se han realizado cambios en la manera en que los estudiantes adquieren el contenido que el profesor les imparte mediante diversos medios, ya sea cursos o clases.

El desarrollo de las modalidades educativas ha permitido en el ámbito educacional que surjan nuevas propuestas como materiales didácticos que apoyen al proceso de enseñanza-aprendizaje, todo esto, unido al avance de las Tecnologías de la Información y las Comunicaciones (TIC) requiere cambios en las metodologías educativas para satisfacer la necesidad de conocimiento. Con la utilización de las tecnologías en el proceso educativo surge como consecuencia el llamado e-Learning (aprendizaje electrónico).

Este presenta un conjunto de estándares y recursos que permiten el aprendizaje desde cualquier parte sin la necesidad de la presencia física de un profesor. El e-Learning no supera el método tradicional de enseñanza, pero es de gran ayuda, ya que, apoyándose en las TIC, el estudiante puede interactuar con sus compañeros de clases y con el profesor, lo cual les brinda comodidad en sus horarios y brinda mayor participación de los estudiantes en las actividades asignadas.

Cuba ha hecho grandes avances en lo que hoy le llamamos “Proceso de Informatización de la Sociedad “que tuvo sus inicios a partir de los años ochenta y en vista de este nuevo milenio ha estado desarrollando transformaciones tecnológicas principalmente en el ámbito educacional, donde se han incrementado las posibilidades de desarrollo de modalidades educativas permitiendo, surjan propuestas o modelos como materiales didácticos que apoyen el proceso de enseñanza y aprendizaje.

Dentro de este marco se han realizado diversas acciones para proporcionar recursos educativos que apoyen al quehacer diario de los estudiantes**.** Uno de estos materiales son los Objetos de Aprendizaje (OA) los cuales posibilitan la compartición de contenidos entre instituciones con el propósito de maximizar el uso y rehuso de un mismo material en distintas situaciones educativas pudiendo ser actualizados sin modificar el curso o programa completo (1).

Al hablar de objetos de aprendizaje es natural también mencionar los repositorios de los mismos. Estas colecciones de recursos digitales constan de dos partes: los contenidos

(objetos digitales) y la meta información asociada a los contenidos denominada “Metadato”, que permiten la catalogación digital de la información que contiene un OA y su reutilización en diversos contextos, donde los metadatos se clasifican con determinados criterios de estandarización como las que se integran al modelo SCORM (del inglés Sharable Content Object Reference Model) (1).

El concepto de metadatos antecede a Internet y a la Web, pero, como es de suponer, el interés mundial por las normas y prácticas de metadatos ha estallado con el crecimiento de la publicación electrónica, las bibliotecas digitales, y la concurrente “sobrecarga de información” que resulta de las grandes cantidades de datos digitales disponibles en línea. Actualmente hay un gran interés, y constituye una parte fundamental del desarrollo de la Web Semántica por adoptar a gran escala estándares y prácticas descriptivas para los recursos electrónicos porque ello contribuirá a mejorar la recuperación de recursos relevantes en cualquier contexto.

Se han definido estándares para las diferentes áreas de aplicación de las tecnologías y uno de ellas sería el estándar Metadatos de Objetos Educativos (LOM).

Este estándar y sus ramificaciones representan el cuerpo de metadatos para materiales educativos con mayor reconocimiento y dedicación de esfuerzo en la comunidad internacional de e-Learning (2) y constituye el estándar por excelencia para Metadatos de Objetos Educativos que se rige por el Comité de Estandarización de Tecnologías Educativas del IEEE el cual define y especifica un esquema de metadatos que permite múltiples implementaciones, los atributos, sus definiciones, una estructura jerárquica que los relaciona entre ellos, y por tanto los aspectos teóricos del estándar (3).

A pesar de que el estándar no plantea ningún metadato obligatorio, se definió que los recursos debían tener al menos el título como obligatorio que se encuentra en la categoría general del estándar. El principal objetivo de este estándar es facilitar la búsqueda, la evaluación, la adquisición y el uso de recursos educativos, tanto por parte de los profesores como de los alumnos. Igualmente se pretende facilitar el intercambio de los objetos educacionales, permitiendo el desarrollo de catálogos e inventarios y teniendo en cuenta la diversidad cultural de los entornos donde estos recursos y los metadatos asociados pueden ser usados.

La aplicación del estándar permitirá tener una estructura de información externa (metadatos) que facilite el almacenamiento, identificación y recuperación de los recursos educativos con fines descriptivos, administrativos y de evaluación en el campo educativo (1).

Introducción

Actualmente en la Universidad de las Ciencias Informáticas (UCI) en el Centro de las Tecnologías para la Formación (FORTES) se desarrolla La Plataforma Educativa Xauce ZERA 2.1 que cuenta con aspectos importantes para el proceso de creación de un OA y su posterior reutilización.

Sin embargo, la plataforma cuenta con las siguientes deficiencias para la gestión de metadatos:

- No permite definir elementos obligatorios según el interés de las instituciones que lo utilicen para la edición de recursos.
- Por parte del sistema pudiera existir un mayor número de elementos para completar en las instancias de metadatos, facilitando la actividad de catalogación para el autor.
- No presenta las funcionalidades de exportación/importación imposibilitando su reutilización.

A partir de la situación problemática antes descrita se presenta el siguiente **problema de investigación:** ¿Cómo incorporar funcionalidades en la plataforma educativa Xauce ZERA 2.1 que faciliten la edición de los metadatos en los Objetos de Aprendizaje con el propósito de favorecer su reutilización?

Del problema a resolver expresado anteriormente se deriva como **objeto de estudio** los editores de metadatos para los Objetos de Aprendizaje.

El **campo de acción** está enmarcado en el desarrollo de funcionalidades que permitan la incorporación de metadatos a los Objetos de Aprendizaje en la Plataforma Educativa Xauce ZERA 2.1.

Definiendo como **objetivo general:** Desarrollar un editor de metadatos basado en el estándar LOM que contribuya a elevar la interoperabilidad de los Objetos de Aprendizaje incorporados a la Plataforma Educativa Xauce ZERA 2.1.

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

- Elaborar un marco teórico de la investigación mediante el estudio del estado del arte acerca de las tendencias, las herramientas y las estrategias actuales en el proceso de desarrollo de editores de metadatos.

- Realizar el análisis, diseño e implementación de un editor de metadatos para la Plataforma educativa Xauce ZERA 2.1 que contenga características de sistemas similares.
- Realizar pruebas de calidad de software a la propuesta de solución para verificar el correcto funcionamiento de la misma.

Posibles resultados:

- Propuesta de un conjunto de requisitos de referencia para el desarrollo de un editor de metadatos para la plataforma web educativa que permitan facilitar la búsqueda, la evaluación, la adquisición y el uso de recursos educativos.
- Editor de metadatos que permita gestionar metadatos aplicados a la descripción de recursos educativos. Definir propiedades que permitan que los objetos educacionales sean editados, gestionados, ubicados y evaluados en el estándar LOM.

Los métodos de investigación que soportan el desarrollo del presente trabajo son la combinación dialéctica de los **métodos Teóricos y Empíricos**. Entre los métodos empleados se encuentran:

Métodos Teóricos:

- **Histórico-Lógico**: permitió el estudio de la evolución y desarrollo de las plataformas y estándares educativos para la gestión del aprendizaje.
- **Modelación**: se utiliza en la confección de los diagramas que permitirán representar la propuesta de solución.
- **Analítico-Sintético** se utiliza para el análisis de las funcionalidades y los distintos componentes que brinda la Plataforma Educativa Xauce ZERA 2.1 que permiten el trabajo con metadatos, la información que es generada por el estándar LOM y para sintetizar la investigación realizada sobre las soluciones similares y tecnologías que se utilizarán para el desarrollo de un módulo o componente que contribuya a describir los recursos educativos en la Plataforma Educativa Xauce ZERA 2.1.

Métodos Empíricos:

- **Entrevista:** Se utilizó el proceso de comunicación verbal con el líder del centro y con otras personas que colaboran con el desarrollo e investigación del proyecto, para el manejo de información. Se identificaron las necesidades reales de la línea de

desarrollo, permitiendo fijar el objetivo por el que se trabajó en la propuesta de solución.

El presente trabajo está estructurado por tres capítulos:

- **Capítulo 1:** Es planteada la fundamentación teórica de la investigación realizada, donde se destacan los conceptos que están relacionados con los recursos, estándares educativos y el e-Learning (aprendizaje electrónico). Se describen las metodologías y las tecnologías de desarrollo de software que se utilizarán durante la implementación del módulo o componente.
- **Capítulo 2:** Es realizado el análisis y el diseño de la propuesta de solución, son descritos los requisitos no funcionales y funcionales que se utilizarán en la implementación del módulo o componente. Son reflejados los patrones de diseño, los patrones arquitectónicos y las prácticas de programación que garantizarán que el módulo o componente cumpla con los requerimientos deseados.
- **Capítulo 3:** Implementación y pruebas de la propuesta. Se implementan los requisitos antes definidos y se le realizan todas las pruebas que debe superar la herramienta para verificar la calidad del trabajo.

Capítulo 1: Fundamentación teórica.

Introducción

En este capítulo se tratarán los principales conceptos referentes al problema y se abordarán temas de relevancia como los términos asociados al dominio del problema. Se realizará un estudio para la buena comprensión de la investigación relacionada con los Editores de Metadatos. Además, se describe un estudio del estado del arte de las diferentes metodologías a utilizar en el ciclo de vida de software, herramientas y tecnologías existentes para la edición de metadatos de manera que se logre dar solución al problema planteado.

1.2 Aprendizaje electrónico (e-Learning)

Dentro de la Educación a distancia, es una de las opciones más utilizada para atender la necesidad de educación continua o permanente. Dadas sus características y el soporte tecnológico que lo respalda, el aprendizaje electrónico ofrece la oportunidad de que el estudiante elija sus horarios de estudio convirtiéndose así en una muy buena opción para aquellas personas autónomas que trabajen y quieran estudiar en sus momentos libres posibilitando no solo aprender conceptos nuevos sino también afianzar conocimientos y habilidades, aumentado así la autonomía y la motivación de los estudiantes por diferentes temas (4).

Este no sólo se produce a través de los estándares tecnológicos de Internet, pues también los materiales offline o descargables son un componente primordial de apoyo.

1.2.1 Proceso de enseñanza y aprendizaje

La vinculación entre investigación y docencia podría verse favorecida si la producción de objetos de aprendizaje se orienta a la difusión de resultados de investigación, visiones sobre problemáticas, instrumentos para producir más información sobre problemas y campos de problemas (5).

1.2.2 Objetos de aprendizaje (OA)

Los objetos de aprendizaje son solamente una herramienta educativa que puede insertarse en propuestas curriculares y metodologías de enseñanza y aprendizaje de muy diversa índole. Sin embargo, y considerando que no hay ciencia ni tecnología sin posicionamiento ideológico detrás, en la apropiación de una herramienta educativa como los objetos de aprendizaje, se da

la adhesión a formas de ver y producir conocimiento, a formas de ver y promover el aprendizaje (5).

Abordar una definición de Objeto de Aprendizaje es una tarea un tanto complicada, dado que existe una amplia discusión respecto del término y más, si tomamos en cuenta que éste ha ido evolucionando y adaptándose a las necesidades educativas y tecnológicas. Los objetos de aprendizaje en el ámbito educativo se introducen sin considerar necesariamente a las Tecnologías de la Información y Comunicación (TIC), sin embargo, es a partir de éstas cuando cobra fuerza la idea de tener unidades de aprendizaje auto contenidas, interoperables –capacidad de integrarse en estructuras y plataformas diferentes–, reutilizables, durables y actualizables (1).

1.2.3 Estándares en el aprendizaje electrónico

Una de los principales objetivos de los estándares es facilitar la reutilización y la durabilidad, así como también el intercambio de contenidos entre diferentes sistemas y plataformas.

También han logrado la escalabilidad haciendo posible integrar nuevos recursos según las necesidades o los avances de la materia abordada y, sobre todo, posibilitan la interoperabilidad. Los recursos sirven para cualquier plataforma que soporte el estándar en el que están creados (6).

IMS-LD

IMS Global Learning Consortium Inc. publica en el año 2003 la especificación IMS – Learning Desing desarrollada por la Open University of Netherlands, con una amplia gama de especificaciones IMS se enfocó en desarrollar una especificación que estuviera centrada en el proceso de aprendizaje sin importar el modelo pedagógico, lo que significa, que no tiene modelo pedagógico asociado y puede ser utilizado como meta-modelo ya que cuenta con la capacidad de asimilar cualquier escenario de aprendizaje, intentando asegurar la interoperabilidad de los módulos que genera.

Esta especificación, pedagógicamente neutra como se explicaba anteriormente tiene su basamento en un lenguaje de modelado educativo en el que prima como objetivo definir formalmente una estructura semántica para constituir el proceso de enseñanza-aprendizaje. Específicamente, “describe un método que va a permitir a alumnos y profesores alcanzar ciertos objetivos de aprendizaje previamente planteados mediante la realización de distintas

actividades de aprendizaje, utilizando determinados recursos didácticos, en un cierto orden, bajo ciertas condiciones, en un cierto ambiente de aprendizaje."

IMS-LD refuerza la asociación entre objeto didáctico reutilizable, actividades y roles de personas que intervienen en el proceso educativo, asemejándose más al modelo apreciable en las clases presenciales (7).

Estándares para el empaquetamiento de recursos

Estos estándares son los que permiten el intercambio de materiales entre los sistemas y herramientas como un LMS o una biblioteca digital. Son capaces de interpretar los paquetes, independientemente de la forma, el lugar y el origen de dichos paquetes. Existen varios estándares creados para el empaquetamiento uno de los cuales es SCORM.

SCORM

Los metadatos se clasifican con determinados criterios de estandarización como las que se integran al modelo SCORM (del inglés Sharable Content Object Reference Model). Este modelo sistematiza la información de tal manera que ésta pueda ser indexada y clasificada eficientemente, facilitando su integración a estructuras y plataformas diferentes. SCORM es un estándar que empaqueta y publica los objetos de aprendizaje en soporte digital. Se caracteriza por la facilidad de ser interpretado por diferentes entornos virtuales de enseñanza y de aprendizaje, como por ejemplo Moodle.

De este modo los repositorios como SCORM, "aportan servicios de búsqueda y clasificación de recursos digitales, de acuerdo a una jerarquía y categorización, estableciendo jerarquías automáticamente (1).

SCORM es, como sabemos, un modelo de referencia, un conjunto de estándares para la creación y distribución de contenido e-Learning. El modelo consta, en realidad, de tres sub-especificaciones:

- La especificación sobre cómo empaquetar el contenido formativo, basada principalmente en XML y declarada en el manifiesto del paquete SCORM.
- El entorno de ejecución (Run-Time) especifica cómo se debe presentar o "lanzar" el contenido, así como la forma de comunicación con el gestor de aprendizaje o LMS. La lógica del entorno de ejecución está programada fundamentalmente en JavaScript.

- La secuenciación indica la forma en la que el alumno o alumna pueden navegar por las diversas secciones y subsecciones del contenido. Se especifica también en el manifiesto del paquete (8).

1.3. IEEE Learning Object Metadata (LOM)

El contenido y el propósito de un OA se pueden saber sin acceder directamente a este, gracias a las características, información y propiedades que nos brindan los metadatos de forma tal que se reduce su gestión y uso. Por ende, los metadatos brindan información con el objetivo de hacer más eficiente la utilización de recursos y la búsqueda.

En la actualidad LOM y sus ramificaciones constituyen el cuerpo de metadatos para materiales educativos con mayor reconocimiento y dedicación de esfuerzo en la comunidad internacional de e-Learning (2).

Las especificaciones del estándar están agrupadas en 9 categorías de metadatos distintas:

1. Categoría general: Los metadatos en esta categoría representan información general sobre el material educativo que describe el mismo como un todo.
2. Categoría LifeCycle (ciclo de vida): Esta categoría agrupa metadatos referidos a la historia y estado actual del proceso de producción y mantenimiento del material educativo por parte de los autores.
3. Categoría meta-metadatos: Esta categoría agrupa información relativa a los metadatos en sí.
4. Categoría técnica: Categoría que agrupa metadatos relativos a las características y requisitos técnicos del material en sí.
5. Categoría educativa: Categoría que agrupa metadatos relativos a los usos educativos del material.
6. Categoría derechos: Categoría que agrupa metadatos relativos a los derechos de propiedad e intelectuales del material.
7. Categoría relación: Categoría de metadatos utilizados para establecer relaciones entre el material y otros materiales.
8. Categoría anotación: Anotaciones y comentarios sobre el material educativo.
9. Categoría clasificación: Metadatos para la clasificación del material en taxonomías (9).

1.3.1 Obligatoriedad de los elementos en el estándar LOM

Investigaciones afirman que los sesenta y cuatro elementos para los metadatos que proporciona LTSC (Comité de Estándares para Tecnología del Aprendizaje) son más de lo que se necesita. Sin embargo, el LTSC plantea que todos los elementos de información de LOM son opcionales. Esto implica que, al construir una instancia de metadatos en XML, el desarrollador puede escoger y elegir qué elementos utilizar (10).

Los elementos de obligatoriedad son los que se deben describir obligatoriamente para cada componente de un OA, estos se definen para reducir la instancia de metadatos y enfocar las búsquedas en un rango mínimo de elementos y así obtener mayor éxito en los resultados, además posibilita al usuario una guía para saber con qué elementos debe describir un OA en general y con qué elementos debe describir por ejemplo una imagen o ejercicio incluido dentro del OA (10).

1.4 Interoperabilidad

La norma estadounidense ANSI/NISO Z39.19:2005 y la norma británica BS8723-4:2007, coinciden en definir la interoperabilidad como la capacidad que tienen dos o más sistemas o componentes de intercambiar información y usar esa información que se ha intercambiado (11).

1.5 Metodología de desarrollo de software

Actualmente las metodologías de ingeniería de software pueden considerarse como una base necesaria para la ejecución de cualquier proyecto de desarrollo de software que se considere serio, y que necesite sustentarse en algo más que la experiencia y capacidades de sus programadores y equipo. Estas metodologías son necesarias para poder realizar un proyecto profesional, tanto para poder desarrollar efectiva y eficientemente el software, como para que sirvan de documentación y se puedan rendir cuentas de los resultados obtenidos (12).

En la actualidad existen una gran cantidad de metodologías para el desarrollo de software, separadas en dos grandes grupos; las metodologías tradicionales o pesadas y las metodologías agiles (12).

Las metodologías tradicionales se basan en las buenas prácticas dentro de la ingeniería del software, siguiendo un marco de disciplina estricto y un riguroso proceso de aplicación. Las metodologías ágiles, en cambio, representan una solución a los problemas que requieren

una respuesta rápida en un ambiente flexible y con cambios constantes, haciendo caso omiso de la documentación rigurosa y los métodos formales (12).

A continuación, se presenta una tabla comparativa entre ambas metodologías.

Tabla 1: Comparación entre los metodologias tradicionales y ágiles.

Metodologías Ágiles	Metodologías Tradicionales
Basadas en heurísticas provenientes de prácticas de producción de código.	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo.
Especialmente preparados para cambios de durante el proyecto.	Cierta resistencia a los cambios.
Impuestas internamente. (por el equipo)	Impuestas externamente.
Proceso menos controlado, con pocos principios.	Proceso mucho más controlado, con numerosas políticas/normas.
No existe contrato tradicional o al menos es bastante flexible.	Existe un contrato prefijado.
El cliente es parte del equipo de desarrollo.	El cliente interactúa con el equipo de desarrollo mediante reuniones.
Grupos pequeños (< 10 integrantes) y trabajando en el mismo sitio.	Grupos grandes y posiblemente distribuidos.
Pocos artefactos.	Más artefactos.
Pocos roles.	Más roles.
Menos énfasis en la arquitectura de software.	La arquitectura de software es esencial y se expresa mediante modelos.

Después de haber conocido, brevemente, ambos tipos de metodologías, se decide realizar la propuesta de solución sobre la base de una metodología ágil. En este sentido, porque están orientadas a proyectos pequeños y de poca duración, son sencillas tanto en su aprendizaje como en su aplicación y genera pocos artefactos.

1.5.1 Metodología seleccionada

Para el desarrollo de la solución propuesta, en la disciplina de requisitos, el autor de la presente investigación decide utilizar el escenario número 4. Esta selección se realiza, teniendo en cuenta, que el proyecto está bien definido, el cliente en todo momento estará junto al equipo de desarrollo para convenir los detalles de los requisitos. Además, a pesar de la complejidad mostrada por el proyecto, este no es extenso, posibilitando el uso de las Historia de Usuario.

La Metodología AUP aplica técnicas incluyendo:

1. Desarrollo Dirigido por Pruebas (test driven development - TDD en inglés).
2. Modelado ágil.
3. Gestión de Cambios ágil.

Fases de AUP:

1. Inicio.
2. Elaboración.
3. Construcción.
4. Transición.
5. Gestión de configuración.
6. Gestión de proyectos.
7. Entorno.

Al igual que en RUP, en AUP se establecen cuatro fases que transcurren de manera consecutiva.

Variación de AUP en su variante UCI

Al no existir una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos, entre otras) exigiéndose así que el proceso sea configurable, se decide hacer una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI. Una metodología de desarrollo de software tiene entre sus objetivos aumentar la calidad del software que se produce, de ahí la importancia de aplicar buenas

prácticas, la cual constituye una guía para aplicarla en una entidad desarrolladora. Estas prácticas se centran en el desarrollo de productos y servicios de calidad (13).

Características por escenarios:

Escenario No 1: Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que puedan modelar una serie de interacciones entre los trabajadores del negocio/actores del sistema (usuario), similar a una llamada y respuesta respectivamente, donde la atención se centra en cómo el usuario va a utilizar el sistema. Es necesario que se tenga claro por el proyecto que los CUN muestran como los procesos son llevados a cabo por personas y los activos de la organización.

Escenario No 2: Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que no es necesario incluir las responsabilidades de las personas que ejecutan las actividades, de esta forma modelarían exclusivamente los conceptos fundamentales del negocio. Se recomienda este escenario para proyectos donde el objetivo primario es la gestión y presentación de información.

Escenario No 3: Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio con procesos muy complejos, independientes de las personas que los manejan y ejecutan, proporcionando objetividad, solidez, y su continuidad. Se debe tener presente que este escenario es muy conveniente si se desea representar una gran cantidad de niveles de detalles y la relaciones entre los procesos identificados.

Escenario No 4: Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido. El cliente estará siempre acompañando al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos. Se recomienda en proyectos no muy extensos, ya que una HU no debe poseer demasiada información. Todas las disciplinas antes definidas (desde Modelado de negocio hasta Pruebas de Aceptación) se desarrollan en la Fase de Ejecución, de ahí que en la misma se realicen iteraciones y se obtengan resultados incrementales. En una iteración se repite el flujo de trabajo de las disciplinas: requisitos, análisis y diseño, implementación y pruebas internas. De esta forma se brinda un resultado más completo para un producto final de manera creciente. Para llegar a lograr esto, cada requisito debe tener un completo desarrollo en una única iteración *(13)*.

1.6 Marco de trabajo Xalix

Como parte de la arquitectura de referencia para el desarrollo de aplicaciones en el Centro FORTES y con el propósito de disminuir la diversidad tecnológica de soluciones. Se ha comenzado la formalización de un marco de trabajo en las Líneas de Producción de Software (LPS).

Las tecnologías utilizadas por este marco de trabajo son:

- Gestor de bases de datos: PostgreSQL.
- Framework de desarrollo: Symfony 2.7.16.
- Lenguaje de programación para el servidor: php 7.0.
- Lenguaje de programación para el cliente: html5.
- Librería Css: Boostrap 3.0.0
- Componentes nativos de Xalix: Gestión de autenticación, Gestión de servicios web, Temas y Panel de administración.

1.7 Herramientas y tecnologías a utilizar para el desarrollo de la aplicación.

El centro FORTES se ha especializado en el desarrollo de tecnologías que permitan ofrecer productos y servicios para la implementación de soluciones de formación, aplicando las TICs a todo tipo de instituciones con modelos de formación y condiciones tecnológicas diferentes. En el módulo a implementar se seleccionarán un conjunto de herramientas y tecnologías atendiendo a sus características y a las políticas de desarrollo del centro FORTES.

1.7.1 Lenguaje de modelado

UML

El Lenguaje de Unificado de Modelado (UML) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. En este sentido, se usa para entender, diseñar, hojear, configurar, mantener y controlar la información sobre tales sistemas. UML no es un lenguaje de programación, más bien es un lenguaje de propósito general, para sistemas discretos, tales como los compuestos por software, firmware o lógica digital (14).

Herramienta Case

Una Herramienta CASE es aquella que permite la creación de los artefactos necesarios, siguiendo una metodología en la construcción de un software. En la utilización de la metodología AUP, se hace necesario el uso de la misma en las fases de Análisis, Diseño e

Implementación, donde se generan la mayor cantidad de artefactos, a los cuales, es muy fácil realizarles los cambios que ocurren en el diseño de un software, precisamente por ser modelados con estas herramientas. Permite mayor calidad y rapidez del software.

Visual Paradigm 8.0 para UML

Visual Paradigm es una Herramienta Case que soporta todo el ciclo del desarrollo de un software: Análisis y Diseño, Construcción, Pruebas y Despliegue. Permite elaborar todos los diagramas casos de uso, clases y diagramas de actividades. Genera documentación y código desde los diagramas y posibilita el diseño de prototipos de interfaz de usuario.
Visual Paradigm ofrece distintas funcionalidades como:

- Diseño centrado en casos de uso y enfocado al negocio generando un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa en su versión profesional e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo (15).

Entre sus ventajas:

- Apoya todo lo básico en cuanto a artefactos generados en las etapas de definición de requisitos y de especificación de componentes.
- Tiene disponibilidad en múltiples plataformas: Microsoft Windows (98, 2000, XP, o Vista), Linux, Mac OS X, Solaris o Java.
- Brinda la posibilidad de intercambiar información mediante la importación y exportación de ficheros con aplicaciones como por ejemplo Visio y Rational Rose.
- Brinda la posibilidad de generar código a partir de los diagramas, para plataformas como .Net, Java y PHP, así como obtener diagramas a partir del código.

1.7.2 Lenguajes de programación

Lenguaje artificial que puede ser usado para controlar el comportamiento de una máquina, especialmente una computadora. Hay que distinguir dos partes en el lenguaje de programación, lo que se denomina sintaxis del lenguaje y la semántica. Por sintaxis entendemos el conjunto de las construcciones del lenguaje que consideramos correctas en

cuanto a su forma, mientras que la semántica es ese mismo conjunto de construcciones que consideramos correctas en cuanto al significado (16).

PHP 7.0

Para el desarrollo del sistema se utiliza como lenguaje de programación por parte del servidor PHP ya que es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML. Lo que distingue a PHP de algo como JavaScript del lado del cliente, es que el código es ejecutado en el servidor, generando HTML y enviándolo al cliente. El servidor web puede ser incluso configurado para que procese todos los ficheros HTML con PHP. Es extremadamente simple para el principiante, pero a su vez ofrece muchas características avanzadas para los programadores profesionales, también es un lenguaje libre e incluye gran cantidad de funciones (17).

El desarrollo de PHP se centra en la programación de scripts del lado del servidor, pero también se puede utilizar para realizar otras funcionalidades. Esta característica nos permite recopilar información de formularios, enviar o recibir cookies y generar paginas dinámicas (18). Este lenguaje posee un conjunto de ventajas:

- Este lenguaje permite crear ficheros PDF y crear imágenes sobre la marcha.
- Genera ficheros XML y cualquier tipo de texto como XHTML y crea un cache del lado del servidor para contenidos dinámicos.
- Se puede usar en cualquier tipo de sistema operativo como es Windows, Linux o Mac OS.

1.7.3 Lenguaje del lado del cliente

HTML 5

HTML5 es un lenguaje markup (de hecho, las siglas de HTML significan Hyper Text Markup Language) usado para estructurar y presentar el contenido para la web. Es uno de los aspectos fundamentales para el funcionamiento de los sitios. Con este, tenemos otras posibilidades para explotar usando menos recursos.

También entra en desuso el formato XHTML, dado que ya no sería necesaria su implementación. Se trata de un sistema para formatear el layout de nuestras páginas, así como hacer algunos ajustes a su aspecto. Con HTML5, los navegadores como Firefox, Chrome, Explorer, Safari y más pueden saber cómo mostrar una determinada página web, saber dónde están los elementos, dónde poner las imágenes y dónde ubicar el texto (19).

Desde su creación, el lenguaje HTML ha pasado de ser un lenguaje utilizado exclusivamente para crear documentos electrónicos a ser un lenguaje que se utiliza en muchas aplicaciones electrónicas como buscadores, tiendas online y banca electrónica.

1.7.3 Entorno de Desarrollo Integrado (IDE)

NetBeans

NetBeans es un proyecto exitoso de código abierto con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios (¡y creciendo!) en todo el mundo. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio 2000 y continúa siendo el patrocinador principal de los proyectos. Al día de hoy hay disponibles dos productos: el NetBeans IDE y NetBeans Platform. NetBeans IDE es un entorno de desarrollo - una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas (20).

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. NetBeans IDE es un producto libre y gratuito sin restricciones de uso.

También está disponible NetBeans Platform; una base modular y extensible usada como estructura de integración para crear grandes aplicaciones de escritorio. Empresas independientes asociadas, especializadas en desarrollo de software, proporcionan extensiones adicionales que se integran fácilmente en la plataforma y que pueden también utilizarse para desarrollar sus propias herramientas y soluciones (20).

1.7.4 Framework y Librerías

Symfony 2.7

Framework desarrollado para optimizar el desarrollo de las aplicaciones web, simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además, proporciona una estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. Por último, es un framework facilitador de la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja.

Symfony está implementado completamente en PHP. Ha sido puesto a prueba en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Este es

compatible con gran cantidad de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft (21).

Boostrap 3.3.0

Boostrap es un framework HTML, CSS y JavaScript que permite crear interfaces web, cuya particularidad es la de adaptar la interfaz del sitio web al tamaño del dispositivo en que se visualice. El mismo está diseñado pensando en ofrecer la mejor experiencia de usuario tanto a usuarios de computadora, como a Smartphone y Tablet (22).
A continuación, se presentan sus principales ventajas:

- Utiliza componentes y servicios creados por la comunidad web.
- Utiliza HTML y CSS3.
- Tiene una gran comunidad que soporta este desarrollo y cuenta con implementaciones externas como WordPress, Drupal y jQuery.
- Maneja un conjunto de buenas prácticas que perdurarán en el tiempo (22).

CSS 3.0 (Hojas de Estilo en Cascada)

Es un lenguaje de hojas de estilo creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML, es decir páginas web. CSS es la mejor forma de separar los contenidos y su presentación, es imprescindible para crear páginas web complejas. Separar la definición de los contenidos y la definición de su aspecto presenta numerosas ventajas, ya que obliga a crear documentos HTML bien definidos y con significado completo. Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes.

Cuando se crea una página web con HTML se enmarcan los contenidos como son: párrafo, titular, texto destacado, tabla, lista de elementos, entre otros. Una vez creados los contenidos, se utiliza el lenguaje CSS para definir el aspecto de cada elemento: color, tamaño y tipo de letra del texto, posición de cada elemento dentro de la página, entre otros.

1.7.5 Sistema Gestor de Bases de Datos

Los Sistemas de Gestión de Bases de Datos (SGBD), básicamente son programas orientados a la gestión y diseño de bases de datos, lo que permite su creación y modificación. Además del desarrollo y construcción de las bases de datos, opera directamente en las tablas, lo cual

hace posible la navegación y visualización de los registros almacenados en las tablas de la misma, su edición, búsqueda, inserción y eliminación (23).

EL SGBD PostgreSQL es clase empresarial, gratuito y libre**;** además de que hoy nos ofrece una gran cantidad de opciones avanzadas tales como Multi-Versión Control de concurrencia (MVCC), puntos en tiempo de recuperación, tablespaces, replicación asincrónica, transacciones anidadas (savepoints), respaldos online/hot, un sofisticado query planner/optimizer. Soporta el conjunto de caracteres internacional, codificaciones de caracteres multibyte, Unicode, mayúsculas y minúsculas. En este sentido, es altamente escalable, tanto en la enorme cantidad de datos que puede manejar y en el número de usuarios concurrentes que puede administrar (24).

Es multiplataforma, pues corre en los sistemas operativos más populares, incluyendo GNU/Linux, UNIX y Windows. Posee claves foráneas (foreign keys), uniones (joins), vistas (views), disparadores (triggers). Incluye la mayoría de tipos de datos de SQL como Integer, Numeric, Boolean, Char, Varchar, Date, Interval. Puede almacenar objetos binarios grandes, incluyendo imágenes, sonido y video (24).

1.8 Análisis de soluciones similares

Como parte del trabajo que se ha desarrollado a nivel mundial sobre la edición de metadatos, se han desarrollado muchas herramientas con este fin. Muchos de los sistemas existentes se especializan en la visualización de estos metadatos, mientras otros son capaces de exportar e importar los metadatos. Se realizó una investigación acerca de los sistemas informáticos existentes que se encargan de la edición y caracterización de metadatos, en el ámbito educacional y otras esferas. Mediante una valoración de las funcionalidades principales que realizan, se especifica si pueden ser utilizados dichos sistemas para darle solución a la problemática existente.

ShameEditor

ShameEditor es una biblioteca de redactores, presentaciones e interfaces de pregunta para el recurso RDF céntrico de metadatos. La idea central de Shame es de trabajar con los Perfiles de Anotación que abarcan:

- El uso de la multiplicidad y vocabularios definidos.
- Presenta aspectos como el orden, la agrupación y la etiqueta.

Estos perfiles de anotación entonces son usados para generar interfaces de usuario para edición, presentación o para interrogatorio de objetivos (25).

No permite la edición de objetos de aprendizajes y no es compatible con el framework sobre el que está desarrollado la plataforma Xauce ZERA ya que este está desarrollado sobre la versión 2.7.16 de Symfony y el SHAME Editor está desarrollado sobre la versión 3.0.

Reload

Reload es un proyecto financiado en el Cambio de JISC para el Estudio del Programa (X4L). El proyecto se enfoca en el desarrollo de los instrumentos que están basados en la aparición de estudios de datos específicos de interoperabilidad. Es manejado por la Universidad de Bolton con el personal localizado en esa institución y la Universidad de Strathclyde.

Los objetivos primarios de este proyecto son:

- Facilitar la creación, compartiendo y reutilizando objetos de aprendizaje.
- Los servicios realzan(mejoran) la gama de accesos pedagógicos realizables por el empleo de proyectos de lección.

Estos objetivos serán alcanzados por la producción de una suite de instrumentos de software para la redacción(creación) y la entrega de objetos de estudio estándar dóciles que incorporan guías de usuario comprensivas y recursos de ejemplo (26). Una desventaja que presenta es que no permite la edición de recursos educativos.

LOM Editor

El proyecto LOMEditor 2006 pretende dar un mejor soporte a la creación y edición de objetos de aprendizaje. El punto de partida es el proyecto LOMEditor 2005, fruto del cual surgió la herramienta LOMEditor 2005, capaz de abrir y editar objetos de aprendizaje IMS Content Package, así como de crearlos desde cero, pero no exporta/importa archivos. Esta herramienta ofrece también la posibilidad de evaluar la calidad y de componer objetos de aprendizaje.

El proyecto LOMEditor 2006 cumple los objetivos:

- Modularizar la herramienta LOMEditor 2005 dotándola de una estructura de piezas software enchufables o plugins (mecanismo de extensión que permite añadir nuevas funcionalidades o modificar las existentes).
- Añadir las funcionalidades de edición y creación de:

- Objetos de aprendizaje IMS Metadata.
- Objetos de aprendizaje con secuenciación (IMS Simple Sequencing).
- Añadir el acceso a un repositorio remoto de OA (27).

	Herramientas		
Características	**ShameEditor**	**Reload**	**LOMEditor**
Edición de metadatos	No	No	Si
Visualización de los metadatos	No	Si	Si
Exportar/Importar	Si	No	No

Luego del estudio realizado de los softwares similares se concluye estos carecen de características esenciales para la edición de recursos educativos y son herramientas de escritorio.

El módulo o componente que se propone construir será un software libre, que debe cumplir al menos con las principales funcionalidades que se listan a continuación:

- La edición de recursos educativos.
- La exportación/importación de recursos educativos.
- Visualizar los metadatos y a su vez eliminarlos.

1.9 Conclusiones parciales

Luego del estudio de la fundamentación teórica de la investigación se arribó a las siguientes conclusiones:

- El estudio del estado del arte permitió identificar los conceptos asociados a la propuesta de solución, así como las soluciones similares que determinaron las funcionalidades genéricas de los componentes básicos que serán utilizadas como base para el desarrollo de la herramienta.
- La selección de la metodología de desarrollo de software AUP en su variante UCI definida por el proyecto permitirá establecer un marco de trabajo para el desarrollo de la propuesta de solución.

- La selección del lenguaje de modelado Lenguaje de Modelado Unificado (UML) y Visual Paradigm como herramienta CASE permitirá la elaboración de artefactos de diseño y análisis de la propuesta de solución.

Capítulo 2: Modelación de la propuesta de solución.

Introducción

En el presente capítulo se describen los procesos de negocios, así como también los requisitos funcionales y no funcionales, así como el modelo de diseño del mismo. Para la descripción del trabajo se presenta las Historias de Usuario donde se encapsulan detalladamente los requisitos.

2.1 Historias de usuario

Las historias de usuario (HU) son una forma rápida de administrar los requisitos de los usuarios sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos. Las historias de usuario permiten responder rápidamente a los requisitos cambiantes (28).

Las HU tienen un orden que le permite al cliente organizar sus ideas y facilitan al equipo de trabajo identificar cuales tienen más prioridades en el momento del desarrollo de la solución. Son ideas agrupadas de acuerdo con su funcionalidad.

A continuación, se muestra la historia de usuario Guardar cambios efectuados en los metadatos, los restantes se describen en el Anexo 1 del presente documento.

<table>
<tr><td colspan="3">Historia de usuario 2 Guardar cambios efectuados en los metadatos</td></tr>
<tr><td>Número: 2</td><td colspan="2">Nombre del requisito: Guardar cambios efectuados en los metadatos</td></tr>
<tr><td colspan="2">Programador: Alexnurín Columbié Méndez</td><td>Iteración Asignada: 1era</td></tr>
<tr><td colspan="2">Prioridad: Alta</td><td>Tiempo Estimado: 3 días</td></tr>
<tr><td colspan="2">Riesgo en Desarrollo: N/A</td><td>Tiempo Real: 2 días</td></tr>
<tr><td colspan="3">Descripción:
1- Objetivo:
Permitir guardar los cambios efectuados a los metadatos en el sistema.</td></tr>
</table>

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

Para ver detalles de un evento hay que:

- Estar autenticado en el sistema con el rol X.

- Debe llenar los campos necesarios para la creación del Objeto de Aprendizaje.

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

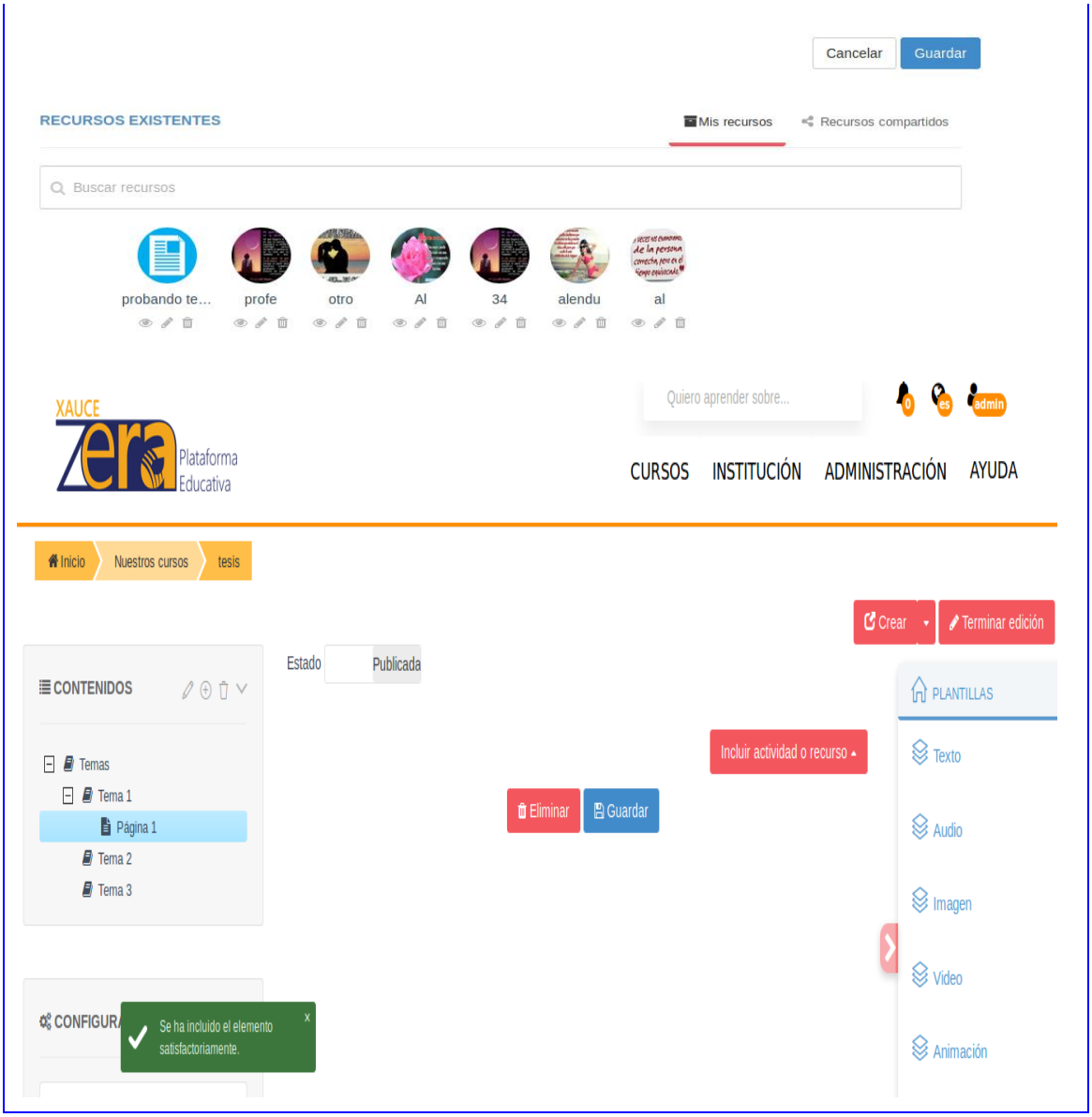
Deben haberse llenado los campos necesarios para la creación del Objeto de Aprendizaje y que no presenten errores.

4- Flujo de la acción a realizar:

El sistema mostrará los campos llenados y si no existe error entonces dará la opción de guardar y mostrará una ventana confirmando que guardo el Objeto de Aprendizaje exitosamente.

Observaciones: Estar autenticado en el sistema con el rol X.

Prototipo de interfaz:



2.2 Requerimientos del sistema

La captura de requisitos o requerimientos, es la actividad mediante la que el equipo de desarrollo de un sistema de software extrae, de cualquier fuente de información disponible, las necesidades que debe cubrir dicho sistema (29).

Requisitos funcionales del sistema

Los requisitos o requerimientos funcionales (RF) son capacidades o condiciones que el sistema debe cumplir. Definición de los servicios que el sistema debe proporcionar, cómo debe reaccionar y comportarse ante situaciones particulares. Definen las funciones que el sistema

será capaz de realizar. Describen las transformaciones que el sistema realiza sobre las entradas para producir salidas (30).

Requisito 1 Insertar metadatos: el sistema debe permitir al usuario insertar los elementos de información.

Requisito 2 Guardar cambios efectuados en los metadatos: el sistema debe brindar la posibilidad al usuario de guardar los cambios efectuados en los metadatos.

Requisito 3 Deshacer cambios efectuados en los metadatos: el sistema debe brindar la posibilidad al usuario de deshacer los cambios efectuados en los metadatos, borrando los cambios introducidos por el usuario.

Requisito 4 Mostrar todos los metadatos: el sistema debe mostrar al usuario todos los elementos del estándar LOM.

Requisito 5 Mostrar ayuda para especificar los elementos de información: el sistema debe ser capaz de mostrar al usuario una ayuda sobre los elementos de información que brinda el estándar de metadatos LOM. Al lado de cada elemento de información aparecerá un ícono, cada vez que el usuario de un clic sobre el mismo, el sistema debe mostrar el significado del elemento y un pequeño ejemplo.

Requisito 6 Exportar metadatos: el sistema debe brindar al usuario la posibilidad de exportar los metadatos, a una dirección especificada.

Requisito 7 Importar metadatos: el sistema debe brindar al usuario la posibilidad de importar los metadatos desde una dirección especificada.

Requisito 8 Visualizar metadatos publicados: el sistema debe permitir al usuario visualizar cada instancia de metadatos que ha sido publicada, mostrando cada elemento de información con su valor correspondiente.

Requisito 9 Buscar metadatos: el sistema debe permitir al usuario realizar una búsqueda de metadatos y/o los elementos de información que el usuario desee encontrar.
Requisito 10 Eliminar metadatos publicados: el sistema debe permitir al usuario eliminar la instancia de metadatos que elija una vez que esté publicada.

2.3 Requisitos no funcionales

Los requisitos no funcionales(RNF) se definen como las cualidades o propiedades que el *software* debe tener. Son aquellos que imponen restricciones en el diseño, la implementación,

diseño y estándares de Calidad (31). Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido y confiable.

Requisitos de seguridad y confiabilidad:

RNF1: Seguridad de acceso y administración de usuarios: otorgamiento de privilegios y roles, asignación de perfiles. Los niveles de acceso están determinados por los diferentes roles válidos dentro de la misma.

RNF2: Garantizar que la información sea editada únicamente por quien está autorizado y posea permisos para ello.

Requisitos de apariencia e interfaz externa

RNF3: El diseño de las interfaces deben ser amigables y sencillo.

RNF4: Claridad y buena organización de la información.

Requisitos de usabilidad

RNF5: El sistema podrá ser usado por cualquier persona que posea conocimientos básicos en el manejo de la computadora y de un ambiente web en sentido general.

2.4 Modelo de Diseño

El modelo del diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación (14).

Sin importar su forma, debe contener suficiente información para reflejar como habrán de traducirse los requisitos de los participantes en contenido y código ejecutable. Pero el diseño también debe ser específico (32).

2.5 Patrones de Diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular. Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno y describe también el núcleo de la solución al problema, de forma que puede utilizarse un millón de veces sin tener que hacer dos veces lo mismo (33).

2.5.1 Patrones GRASP

Los Patrones Generales de Software para Asignar Responsabilidades (GRASP, por sus siglas en inglés), son parejas de problema y solución, que codifican buenos principios y sugerencias relacionados frecuentemente con la asignación de responsabilidades. Describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones.

Creador: Permite crear objetos de una clase determinada. Es utilizado en la mayoría de las clases controladoras para crear instancias de formularios y entidades, para la vista del usuario, se evidencia en la clase MetadataController.php (34).

Controlador: Se basa en asignar la responsabilidad de todos los eventos realizados a una clase específica que constituye el único punto de entrada para cada evento. En la solución propuesta este patrón está evidenciado en la clase controladora MetadataController.php (34).

Alta Cohesión: En la perspectiva del diseño orientado a objetos, la cohesión es una medida de cuan relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. En la solución propuesta este patrón está evidenciado en las clases que extienden de la clase Repository (34).

Bajo Acoplamiento: El acoplamiento mide el grado en que una clase está conectada, tiene conocimiento o de alguna manera depende de otra. Este patrón consiste en asignar la responsabilidad de manera que el acoplamiento permanezca bajo. El bajo acoplamiento permite crear clases más independientes, más reutilizables, lo que implica mayor productividad. En la solución del componente se evidencia en las clases que extienden de Repository (34).

Experto: Consiste en asignar una responsabilidad al experto en información, se asigna la responsabilidad a la clase que cuenta con la información necesaria para cumplirla. Se evidencia en las clases que extienden de la clase Entidad, las cuales son expertas en su propia información (34).

En la propuesta de solución se evidencia en la Clase MetadataController.php, la cual se especializa exclusivamente en modelar el negocio de la edición de metadatos basados en el estándar LOM.

2.5.2 Patrones GoF

Estos patrones se dividen en tres categorías diferentes los **patrones de comportamiento** que describen la comunicación entre clases y objetos, los **patrones de creación** que permiten la configuración de objetos y su inicialización y los **patrones estructurales** que separan la interfaz de la implementación.

Patrón de creación.

Builder (Constructor). Separa la construcción de un objeto complejo de su representación, de forma que el mismo proceso de construcción pueda crear diferentes representaciones. Una clase superior Builder se aplica en la creación de clases menores cuando sea necesario crear y agregar instancias de clases, cuando sea necesario contener múltiples instancias de clases y cuando la clase superior dispone de los datos necesarios para la clase a instanciar. Reduce el acoplamiento. Permite variar la representación interna de estructuras complejas, respetando la interfaz común de la clase Builder (35). Se evidencia en la clase MetadaType.php.

2.6 Patrones Arquitectónicos

Los patrones arquitectónicos se utilizan para expresar una estructura de organización base o esquema para un software. Proporcionando un conjunto de subsistemas predefinidos, especificando sus responsabilidades, reglas, directrices que determinan la organización, comunicación, interacción y relaciones entre ellos. Heredan mucha de la terminología y conceptos de patrones de diseño, pero se centran en proporcionar modelos y métodos reutilizables específicamente para la arquitectura general de los sistemas de información. Dentro de los patrones arquitectónicos podemos encontrar Inyección de dependencias, Arquitectura dirigida por eventos (Event-driven architecture o EDA), Arquitectura orientada a servicios o Modelo Vista Controlador (36).

El patrón arquitectónico seleccionado es el Modelo Vista Controlador pues es el utilizado por el framework Symfony. Además, este patrón separa los datos de una aplicación la interfaz de usuario y la lógica de control en tres componentes distintos de forma que las modificaciones al componente de la vista pueden ser hechas con un mínimo impacto en el componente del modelo de datos (37).

2.6.1 Patrón arquitectónico Modelo Vista Controlador

El patrón Modelo Vista Controlador (MVC) 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.

EL patrón MVC es un paradigma que divide las partes que conforman una aplicación en el Modelo, las Vistas y los Controladores, permitiendo la implementación por separado de cada elemento, garantizando así la actualización y mantenimiento del software de forma sencilla y en reducido espacio de tiempo. A partir del uso de frameworks basados en este patrón se puede lograr una mejor organización del trabajo y mayor especialización de los desarrolladores y diseñadores (37).

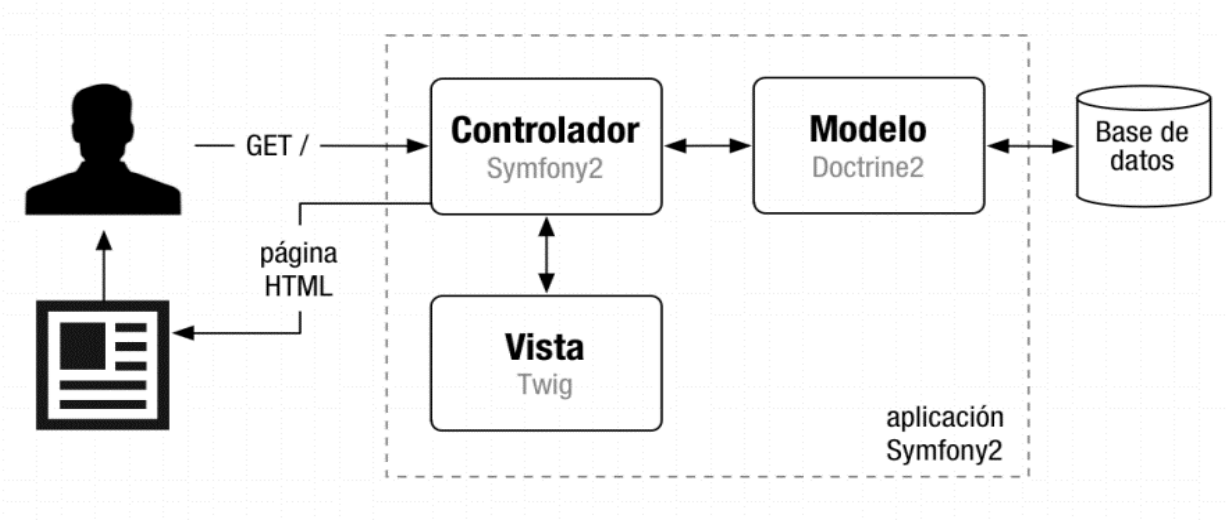


Figura 2: Patrón Arquitectónico modelo-vista –controlador.

Este patrón de arquitectura presenta varias ventajas:

- Separación clara entre los componentes de un programa; lo cual permite su implementación por separado.
- Interfaz de Programación de Aplicaciones (API) muy bien definida; cualquiera que use el API, podrá reemplazar el Modelo, la Vista o el Controlador, sin aparente dificultad.
- Conexión entre el Modelo y sus Vistas dinámica; se produce en tiempo de ejecución, no en tiempo de compilación (37).

Definiendo este patrón por partes el Modelo es el objeto que representa los datos del programa, maneja los datos y controla todas sus transformaciones. El Modelo no tiene conocimiento específico de los Controladores o de las Vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el Modelo y sus Vistas, y notificar a las Vistas cuando cambia el Modelo.

La Vista es el objeto que maneja la presentación visual de los datos representados por el Modelo. Genera una representación visual del Modelo y muestra los datos al usuario. Interactúa preferentemente con el Controlador, pero es posible que trate directamente con el Modelo a través de una referencia al propio Modelo.

El Controlador es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo, centra toda la interacción entre la Vista y el Modelo. Cuando se realiza algún cambio, entra en acción, bien sea por cambios en la información del Modelo o por alteraciones de la Vista. Interactúa con el Modelo a través de una referencia al propio Modelo (37).

2.7 Modelo de diseño

Los diagramas de clases del diseño son una representación más concreta y detallada que los diagramas de clases del análisis, aunque también representan la parte estática del sistema conteniendo las clases y sus relaciones. Son empleados para representar las relaciones que se establecen entre las clases (43).

2.8 Diagrama de Diseño

Los diagramas de clases del diseño son una representación más concreta y detallada que los diagramas de clases del análisis, aunque también representan la parte estática del sistema conteniendo las clases y sus relaciones. Son empleados para representar las relaciones que se establecen entre las clases (14).

Se presenta a continuación el diagrama de clase del diseño de la historia de usuario Insertar metadatos, guardar formulario y buscar formulario. Para el estudio de los demás diagramas del diseño remitirse al Anexo 3.

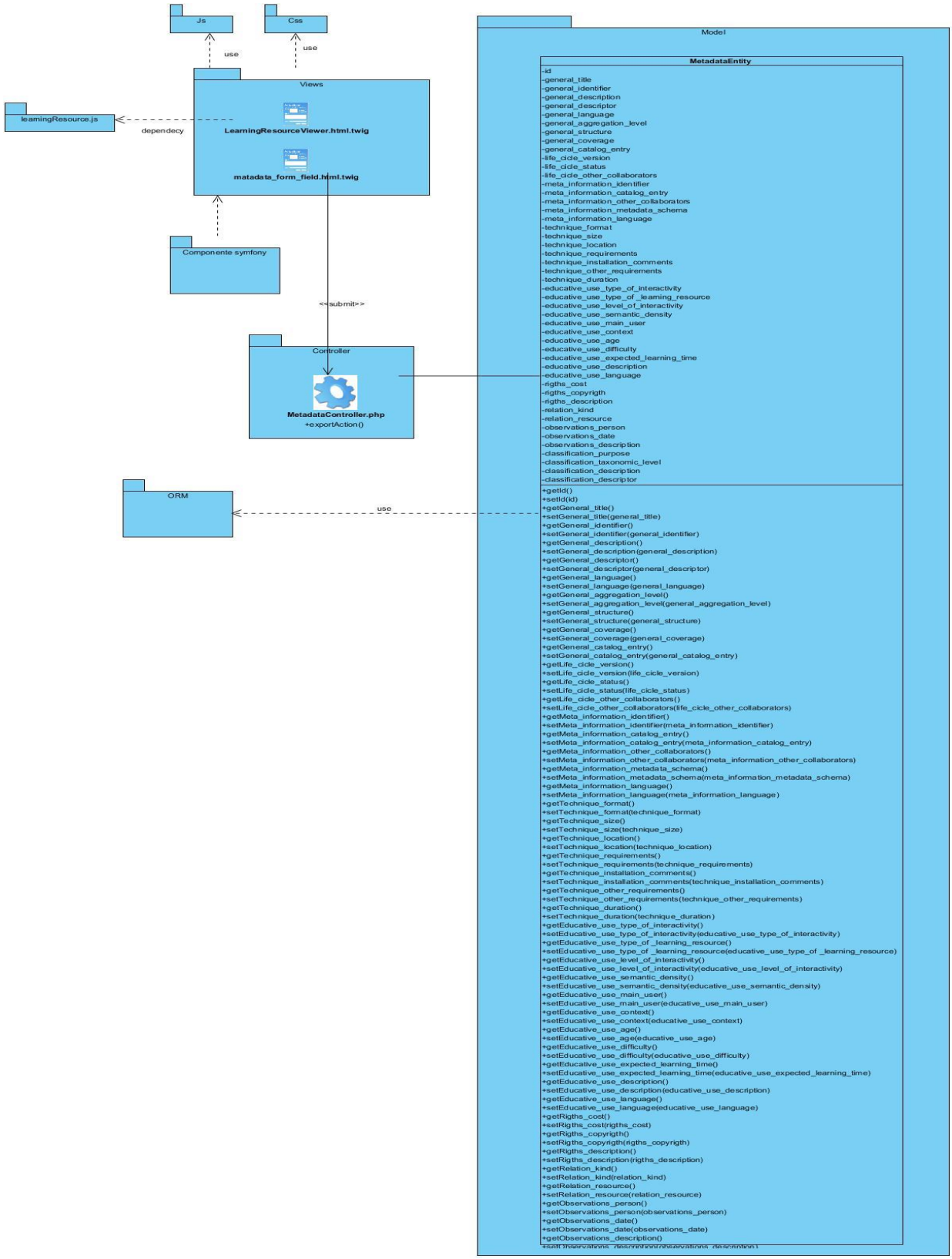


Figura 2. Diagrama de clase de diseño.

2.10 Modelo de datos

Un modelo de datos es un conjunto de herramientas conceptuales para describir datos, sus relaciones, su significado y sus restricciones de consistencia. Es el proceso de analizar los aspectos de interés para una organización y la relación que tienen unos con otros. Tiene como meta registrar los requerimientos de datos de un proceso de negocio. Está compuesto por las entidades que conformarán las tablas de la base de datos que serán utilizadas por las funcionalidades a implementar (38).

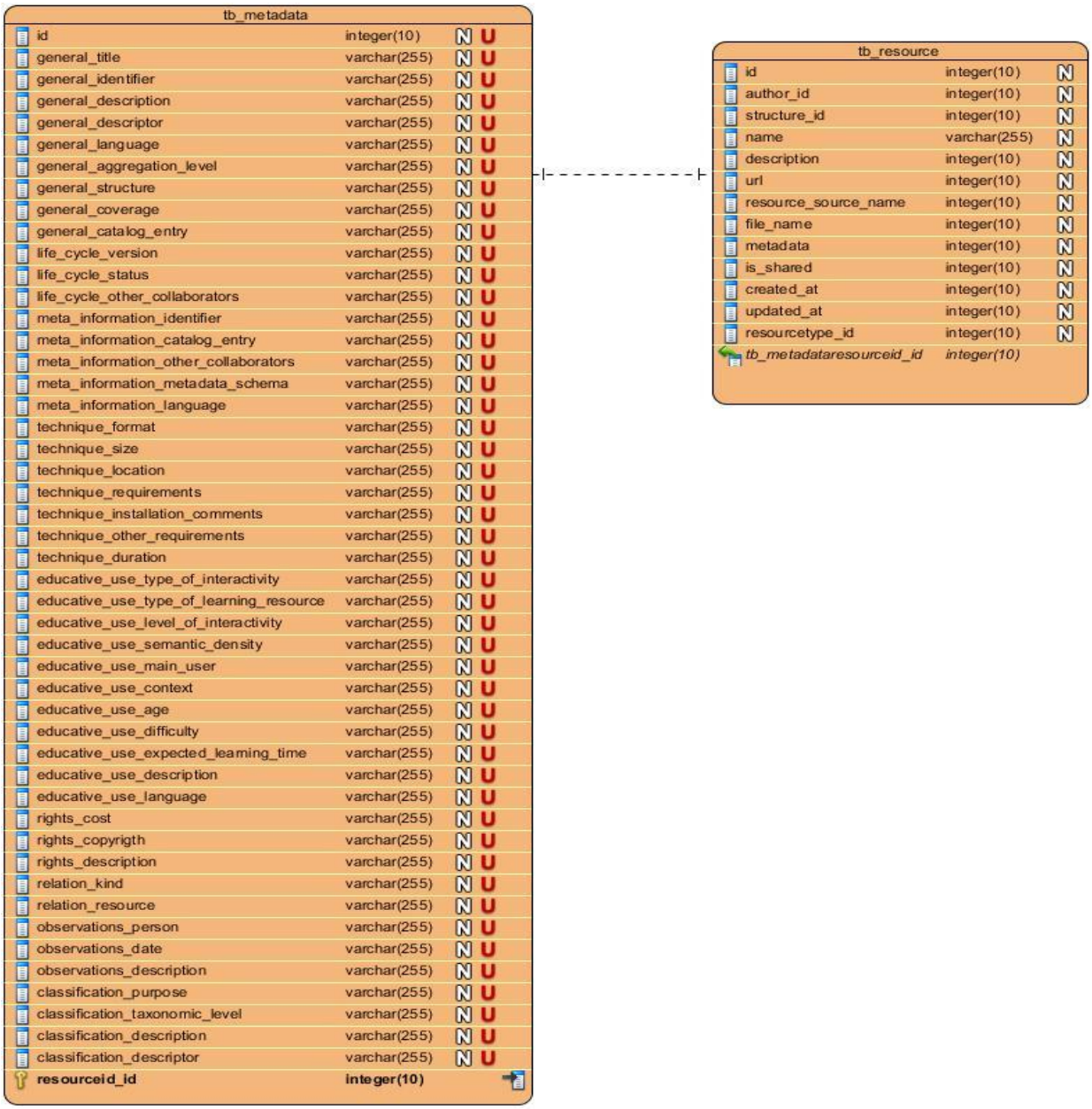


Figura 3: Modelo de datos del sistema.

Descripción de las tablas de la base de datos

La recolección de los datos se realiza utilizando la base de datos de la Plataforma Educativa ZERA. Se realiza un estudio del modelo Entidad Relación, el cual cuenta con 223 tablas. De este modelo se seleccionaron las tablas que almacenaban información de interés para la investigación, trabajando finalmente con 2. Las tablas a describir son tb_metadata y tb_resource.

Tabla 2 Descripción de la tabla tb_metadata de la base de datos

tb_metadata		
Descripción: en la siguiente tabla se agrupa la información correspondiente a los datos ingresados por el usuario a la hora de crear un recurso.		
Atributo	**Tipo**	**Descripción**
id	Integer	Etiqueta única que identifica al objeto en la tabla.
general_title	varchar (255)	Almacena el título del recurso.
general_identifier	varchar (255)	Almacena el identificador del recurso.
general_description	varchar (255)	Almacena una breve descripción del recurso.
general_descriptor	varchar (255)	Almacena el nombre del descriptor del recurso.
general_language	varchar (255)	Almacena el idioma del recurso.

general_aggregation_level	varchar (255)	Almacena los niveles de agregación (1,2,3,4).
general_structure	varchar (255)	Almacena las diferentes estructuras que puede tener el recurso ya sea atómica, colección, en red, jerárquica o lineal.
general_coverage	varchar (255)	Almacena la cobertura que tendrá el recurso.
general_catalog_entry	varchar (255)	Almacena la entrada del catálogo del recurso.
life_cicle_version	varchar (255)	Almacena la versión del recurso
life_cicle_status	varchar (255)	Almacena el estado del recurso ya sea si está publicado o no
life_cicle_other_collaborators	varchar (255)	Almacena los nombres de otros colaboradores que estén relacionados con el recurso.
meta_information_identifier	varchar (255)	Almacena el identificador del recurso.
meta_information_catalog_entry	varchar (255)	Almacena la entrada al catálogo del recurso
meta_information_other_collaborators	varchar (255)	Almacena los nombres de otros colaboradores relacionados con el

		recurso.
meta_information_metadata_schema	varchar (255)	Almacena el esquema de metadatos del recurso.
meta_information_language	varchar (255)	Almacena el idioma del recurso.
technique_format	varchar (255)	Almacena el formato del recurso.
technique_size	varchar (255)	Almacena el tamaño del recurso.
technique_location	varchar (255)	Almacena la ubicación del recurso.
technique_requirements	varchar (255)	Almacena los requerimientos del recurso.
technique_installation_comments	varchar (255)	Almacena los comentarios de instalación del recurso.
technique_other_requirements	varchar (255)	Almacena otros requerimientos que el autor del recurso quiera tomar en cuenta.
technique_duration	varchar (255)	Almacena la duración del recurso.
educative_use_type_of_interactivity	varchar (255)	Almacena el tipo de interactividad (activo,

		expositivo, combinado).
educative_use_type_of _learning_resource	varchar (255)	Almacena el tipo de recurso(audio, documento, imagen).
educative_use_level_of_interactivity	varchar (255)	Almacena el nivel de interactividad (bajo, medio, alto).
educative_use_semantic_density	varchar (255)	Almacena la densidad semántica (bajo, medio, alto).
educative_use_main_user	varchar (255)	Almacena a quién estará dirigido el recurso.
educative_use_context	varchar (255)	Almacena el contexto del recurso (aula, laboratorio).
educative_use_age	varchar (255)	Almacena la edad del destinatario del recurso.
educative_use_difficulty	varchar (255)	Almacena la dificultad del recurso (bajo, medio, alto).
educative_use_expected_learning_time	varchar (255)	Almacena el tiempo esperado de aprendizaje.
educative_use_description	varchar (255)	Almacena breve descripción del recurso.
educative_use_language	varchar (255)	Almacena el idioma del

		recurso.
rigths_cost	varchar (255)	Almacena el coste del recurso (sí, no).
rigths_copyrigth	varchar (255)	Almacena los derechos de autor.
rigths_description	varchar (255)	Almacena breve descripción del recuso.
relation_kind	varchar (255)	Almacena el tipo de relación que tiene con el recurso principal.
relation_resource	varchar (255)	Almacena el recuso principal al que se refiere esta relación.
observations_person	varchar (255)	Almacena la persona que describe el recurso.
observations_date	varchar (255)	Almacena la fecha de creación del recurso.
observations_description	varchar (255)	Almacena una breve descripción del recurso.
classification_purpose	varchar (255)	Almacena la finalidad del recurso (nivel educativo, competencia, accesibilidad, disciplina).
classification_taxonomic_level	varchar (255)	Almacena el nivel taxonómico.
classification_description	varchar (255)	Almacena una breve

		descripción del recurso.
classification_descriptor	varchar (255)	Almacena el nombre del descriptor del recurso

tb_resource		
Descripción: en la siguiente tabla se agrupa la información correspondiente a la tabla resource donde se almacenan los principales parámetros del recurso.		
Atributo	***Tipo***	***Descripción***
id	Integer	Etiqueta única que identifica al objeto en la tabla.
author_id	Integer	Almacena el identificador del autor.
structure_id	Integer	Almacena el identificador de la estructura del recurso.
name	varchar(255)	Almacena el nombre del recurso.
description	Integer	Almacena una breve descripción del recurso.
url	Integer	Almacena la dirección desde donde se carga el recurso
resource_source_name	Integer	Almacena la fuente del recurso.
file_name	Integer	Almacena el nombre del archivo.

metadata	Integer	Almacena los metadatos asociados al recurso.
is_shared	Integer	Almacena la información de esta compartido o no.
created_at	Integer	Almacena el nombre del creador del recurso.
updated_at	Integer	Almacena la información de quién actualizó el recurso.
resourcetype_id	Integer	Almacena el identificador del tipo de recurso.

2.11 Conclusiones parciales

La propuesta de solución presentada en este capítulo está sustentada por 10 requisitos funcionales lo cual brindó un enfoque del módulo a realizar, el desarrollo del modelo del diseño permitió definir las bases necesarias para la implementación del módulo y posibilitó que se obtuvieran de una forma concreta las relaciones que se establecen entre las clases y el esclarecimiento de la arquitectura y los patrones de diseño a utilizar lo cual contribuyó a lograr la implementación centrada en el diseño realizado.

Capítulo 3: Implementación y validación de la solución propuesta.

Introducción

En este capítulo se describen los elementos establecidos por el proceso de desarrollo empleado, que responden a la implementación de la solución. Además, se muestran los diagramas de despliegue y de componentes. Se incluirán los resultados de las pruebas realizadas al componente con el propósito de valorar si el sistema cumple con los requisitos establecidos. En este proceso de pruebas se definen varios métodos, técnicas y tipos de pruebas, las cuales se abordarán durante el desarrollo del presente capítulo.

3.1 Modelo de implementación.

El modelo de implementación describe cómo los elementos del diseño se implementan en términos de componentes. Describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y los lenguajes de programación utilizados, y cómo dependen los componentes uno de otros (14).

3.1.1 Descripción de la propuesta de solucion

El componente desarrollado permite agregarles metadatos a los recursos siguiendo las especificaciones del estándar LOM, estos podrán ser guardados después de haber completado el formulario a la hora de insertarlos. Estos metadatos podrán ser visualizados en caso de que el usuario lo requiera en el visor de recurso, donde podrá también buscar el metadato que desee y visualizarlo en caso de que exista. Los metadatos podrán ser exportados hacia una dirección especificada por el usuario y a su vez podrán importarse permitiendo así su reutilización en diversos contextos.

3.1.2 Diagrama de componentes.

Los diagramas de componentes permiten modelar la vista de implementación del sistema a partir del cual se construye la aplicación. Muestran la organización y las dependencias lógicas entre un conjunto de componentes de *software* y organiza los subsistemas de implementación en capas. Un componente es el empaquetamiento físico de los elementos de un modelo, como son las clases en el modelo de diseño (14).

A continuación, se muestra el diagrama de componente correspondiente al sistema.

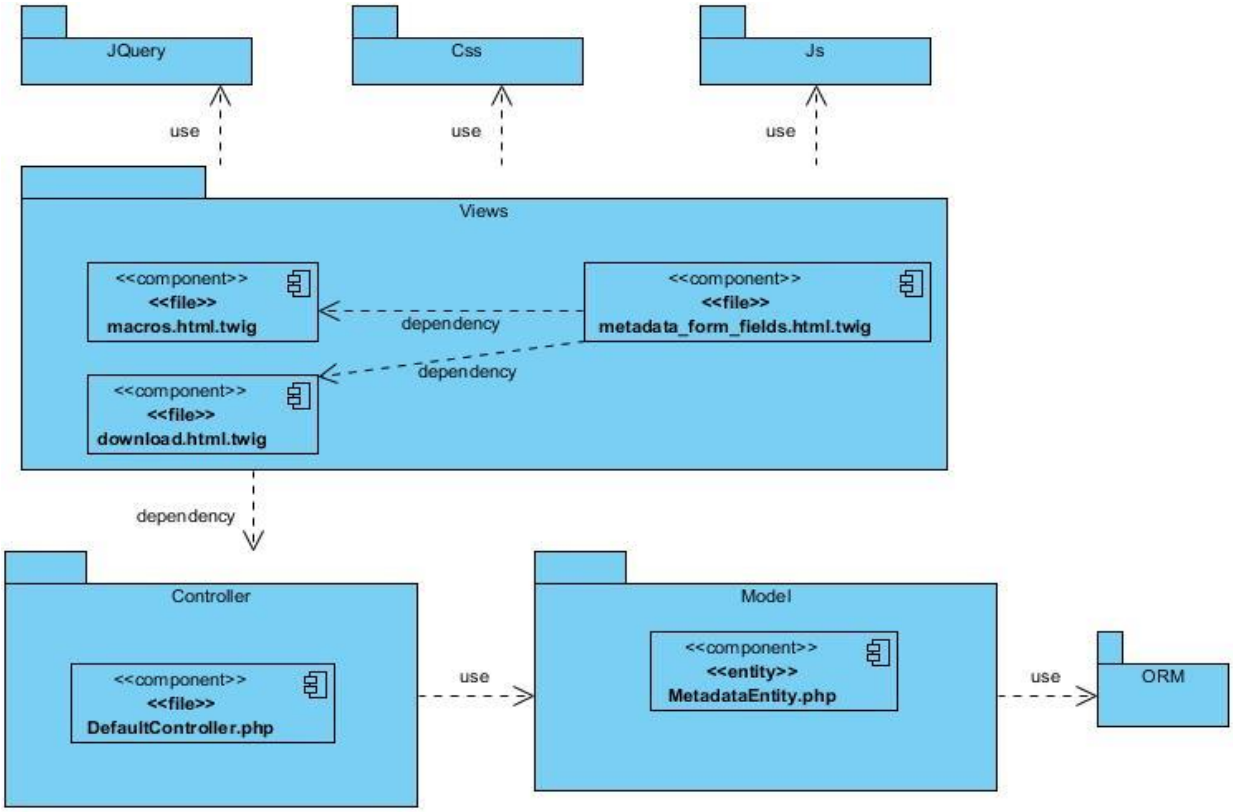


Figura 4: Diagrama de componentes del sistema.

3.1.3 Diagrama de despliegue

El Diagrama de Despliegue es un tipo de diagrama del lenguaje unificado de modelado que se utiliza para modelar la disposición física de los artefactos software en nodos.

Es un diagrama estructurado que muestra la arquitectura del sistema desde el punto de vista del despliegue de los artefactos del software en los destinos de distribución (44).

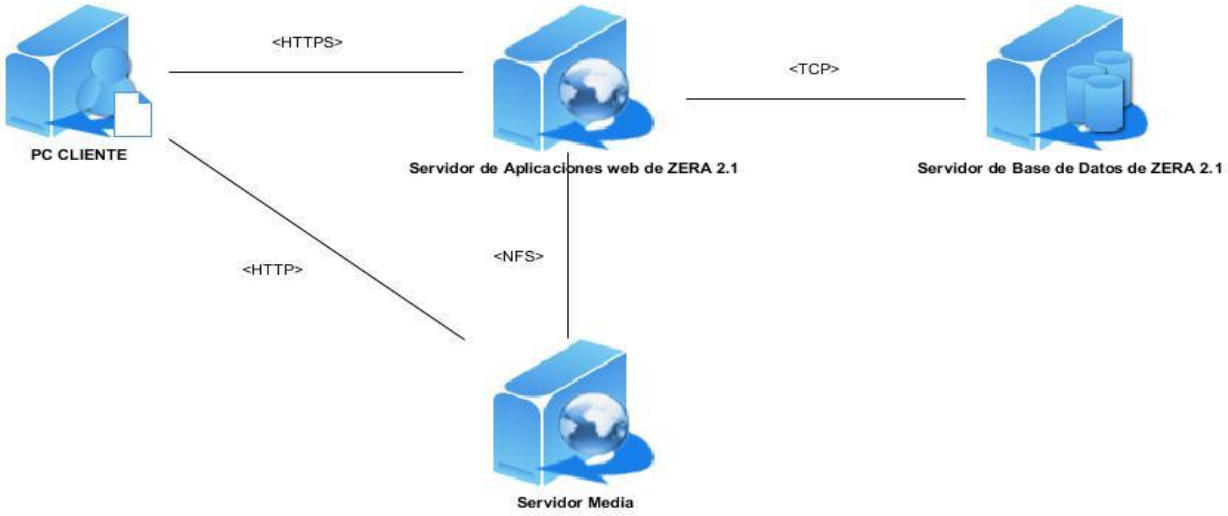


Figura 5: Diagrama de despliegue del sistema.

3.1.4 Estándares de codificación

JavaScript

Descripción: se utiliza la notación camelCase para los nombres de funciones. Todos los nombres comienzan con una letra y se utiliza la palabra reservada "var" para declarar variables.

Ejemplo:

```
var metadataSearch = function(where, what, filters) {
    var paramsQuery = "";
```

Figura 6: Ejemplo de código de JavaScript

HTML

Descripción: no se debe colocar espacios entre la relación atributo-valor. Se debe utilizar lowercase para el nombre de los atributos de las etiquetas.

Ejemplo:

```
<form id="form learningResource id" class="form-horizontal" name="fortes learning resource sources file type"
```

Figura 7: Ejemplo de código de HTML

CSS

Descripción: una declaración CSS siempre termina en punto y coma y los conjuntos de declaraciones se colocan entre llaves. Para hacer un código CSS legible, ponga una declaración en cada línea. Coloque la llave que cierra en una línea nueva.

Ejemplo:

```css
.label-metadata{
    font-family: cursive;
}
```

Figura 8: Ejemplo de código de CSS

Descripción: adicionar una línea en blanco antes de cada sentencia return, a menos que se encuentre como única sentencia (como en un if o un else).

Ejemplo:

```php
    if ($metadata->handleReques($request)) {
        $this->container->get('metadata.form.metadata_type')->save($metadata);
    }
    return new JsonResponse(JsonResponse::HTTP_OK);
else {
    return new JsonResponse(JsonResponse::HTTP_BAD_REQUEST);

}
```

Figura 9: Ejemplo de código de PHP

Descripción: declarar los atributos de las clases antes de los métodos.

Ejemplo:

```php
/**
 * @var string $classificationDescriptor
 *
 * @ORM\Column(name="classification_descriptor", type="string", nullable=true)
 * @Assert\Regex("/^([\w áéíóúÁÉÍÓÚñÑ.,\-_;:'])*$/")
 *
 */
private $classificationDescriptor;

/**
 * @return int
 */
public function getId()
{
    return $this->id;
}
```

Figura 10: Ejemplo de código de PHP

Convención de nombres

Descripción: la declaración de funciones o métodos siempre comenzará con letra inicial minúscula. En caso de ser un nombre compuesto se regirá por la normativa camelCase.

Ejemplo:

```
public function exportAction()
{
```

Figura 11: Ejemplo de código de PHP

Descripción: se utiliza namespaces para todas las clases.

Ejemplo:

```
namespace FORTES\MetadataEditorBundle\Controller;
```

Figura 12: Ejemplo de código de PHP

Descripción: para definir cada uno de los servicios hay que tener en presente los siguientes indicadores:

- ✓ Los nombres de los servicios contienen grupos separados por puntos.
- ✓ El alias de Inyección de Dependencias del bundle es el primer grupo.
- ✓ Se utiliza minúsculas para los nombres de servicios y sus parámetros.
- ✓ Un nombre de grupo utiliza la notación guión bajo.

Ejemplo:

```
metadata.form.metadata_type:
```

Figura 14: Ejemplo de código de PHP

Comentarios en el código

Descripción: los comentarios que emplean una sola línea se definen de la siguiente manera.

Ejemplo:

```
// Write resource temp json file
```

Figura 15: Ejemplo de código comentado

Descripción: los comentarios que emplean varias líneas utilizan las anotaciones siguientes:

- ✓ @param: esta etiqueta provee el nombre, el tipo y la descripción de los parámetros de una función.
- ✓ @return: esta etiqueta es utilizada para documentar el valor que retorna una función.

Ejemplo:

```
/*
 * @param FormBuilder $builder
 * @param array       $array
 * @return mixed
 */
```

Figura 16: Ejemplo de código comentado

3.2 Pruebas de software.

El desarrollo del software implica en sí, errores que pueden empezar a ocurrir desde el primer momento del proceso en el que los requerimientos pueden estar especificados de forma errónea, así como en los posteriores pasos del diseño e implementación. Es por esto, que se hace necesario contar con una actividad que garantice la calidad. Las pruebas es la actividad en la cual un sistema o componente es ejecutado bajo unas condiciones o requerimientos especificados, los resultados son observados y registrados y una evaluación es hecha de algún aspecto del sistema o componente (32).

3.3 Niveles de pruebas

Las pruebas son aplicadas para diferentes tipos de objetivos, en diferentes escenarios o niveles de trabajo. Entre los niveles de prueba se encuentran el nivel de pruebas unitarias, nivel de pruebas de integración, nivel de pruebas del sistema y nivel de pruebas de aceptación. Una vez implementado el sistema fue sometido al nivel de prueba que a continuación se detalla (39).

Pruebas del Sistema: se prueba el sistema para comprobar que se cumplen los requisitos funcionales. El *software* debe probarse ya sea para decidir acerca de su aceptación, para analizar defectos globales o para estudiar aspectos específicos de su comportamiento, tales como seguridad o rendimiento. Este tipo de pruebas estudia el producto completo (40).

Pruebas de aceptación: comprueban el comportamiento del sistema frente a los requisitos del cliente (suele participar el mismo cliente o los usuarios). Se lleva a cabo con el objetivo de que el cliente valide los requisitos que debe poseer el sistema (40).

3.3.1 Métodos de pruebas

Existen distintas técnicas de pruebas que proporcionan criterios para generar casos de pruebas que provoquen fallos en los programas, estas se agrupan en:

Pruebas de caja blanca o estructural: la prueba de caja blanca, en ocasiones llamada prueba de caja de cristal, es un método de diseño que usa la estructura de control descrita como parte del diseño al nivel de componentes para derivar los casos de prueba. Estas comprueban los caminos lógicos del software, proponiendo casos de prueba que ejerciten conjuntos específicos de condiciones y/o bucles. Requieren del conocimiento de la estructura interna del sistema y son derivadas a partir de las especificaciones internas de diseño o el código (41).

Pruebas de caja negra o funcional: las pruebas de caja negra, también denominadas pruebas de comportamiento, se concentran en los requisitos funcionales del *software*. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Permiten encontrar funciones incorrectas o ausentes, errores de interfaz, rendimiento, inicialización y terminación, así como errores en estructuras de datos o en accesos a la base de datos externas. La prueba de caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del *software* y son realizadas sin tener conocimiento interno del sistema (41).

Dentro del método de caja negra se hace uso de la técnica Partición Equivalente por ser una de las más efectivas para examinar los valores válidos e inválidos de las entradas existentes en el sistema. La Partición Equivalente divide el dominio de entrada de un programa en clases de datos a partir de las cuales se derivan casos de prueba (41).

De las pruebas del sistema se tomará el método de caja negra, debido a que este permitirá corregir problemas en la interfaz del usuario y se comprobará que esta propuesta realiza las funciones requeridas por el usuario.

3.3.2 Partición equivalente

La partición equivalente es un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Este se dirige a una definición de casos de pruebas que descubran clases de errores, reduciendo así el número total de casos de pruebas que hay que desarrollar.

El diseño de casos de prueba para partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados válidos y no válidos para las condiciones de entrada. Por lo general, esta condición es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición booleana (42).

3.4 Diseños de caso de prueba

La intensión de los casos de prueba es probar el sistema de una forma detallada, incluyendo las entradas con las que se experimentarán, las condiciones bajo las cuales se realizan y los resultados esperados.

A continuación, se presentan los casos de prueba propuestos para la HU Insertar Metadatos. Para consultar el resto de los casos de prueba remitirse al Anexo 3.

Tabla 3:Diseño de caso de prueba Insertar Metadatos

Escenario	Descripción	Revisión	Respuesta del sistema	Flujo central
EC 1.1 Opción incluir recurso	El usuario selecciona la opción incluir recurso		El sistema muestra un listado de los recursos existentes Además brinda las siguientes opciones: - Desde Archivo - Desde Url	Inicio/Acceder/Modo Edición/Incluir actividad o recurso/incluir recurso
EC 1.2 Opción incluir actividad	El usuario selecciona la opción incluir actividad	V	El sistema muestra un listado de actividades Además brinda las siguientes opciones: -Tareas -Foros -Cuestionario -Talleres -Seminarios	Inicio/Acceder/Modo Edición/Incluir actividad o recurso/incluir actividad
EC 1.3 No hay resultado	El usuario selecciona la	N/A	El sistema no encuentra recursos en la lista regresa al EC 1.2	Inicio/Acceder/Modo Edición/Incluir

	opción incluir recurso			actividad o recurso/incluir actividad

Descripción general: El requisito comienza cuando el usuario ingresa a la interfaz Incluir recurso donde se mostrará un listado de los recursos existente y además permitirá insertar un nuevo recurso.

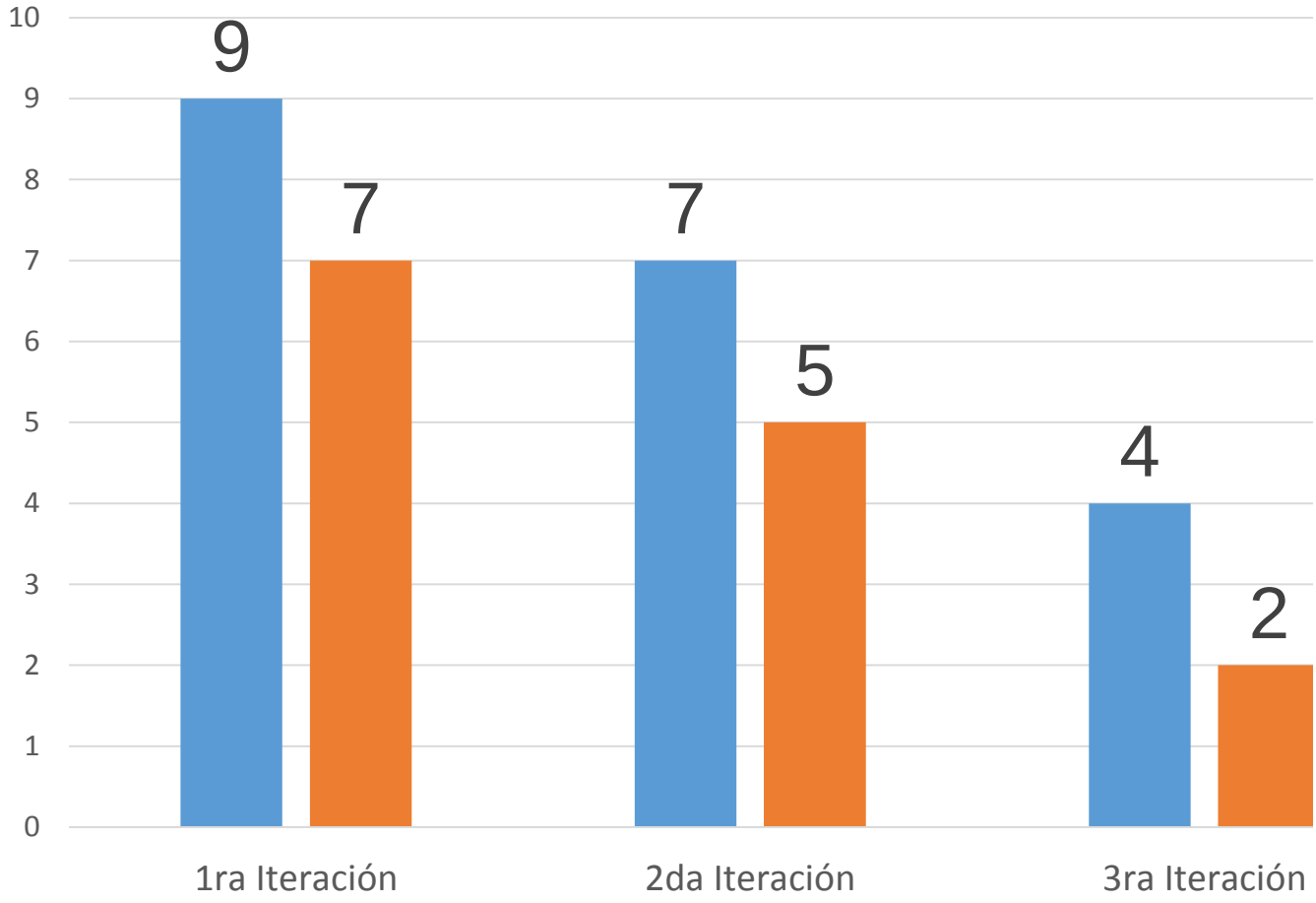
Resultado de las pruebas:

Una vez concluida la implementación del componente se detectaron varias no conformidades (NC). En la tabla siguiente se muestran los resultados obtenidos luego de aplicadas todas las pruebas:

Tabla 4: Cantidad de no conformidades por iteración

Iteraciones	Cantidad de casos de prueba	No conformidades detectadas		
		Detectadas	Resueltas	Total
1	10	9	7	9
2	10	7	5	7
3	10	4	2	4

Para un mejor entendimiento de los resultados, se muestra en el gráfico siguiente la cantidad de no conformidades detectadas por cada una de las iteraciones.



Las principales NC encontradas fueron errores ortográficos, tanto tildes como cambios de mayúsculas por minúsculas, así como también errores en las acciones de insertar y exportar. Además, la acción buscar no se realizaba correctamente. Después de concluida la tercera iteración se le realizó una regresión y se resolvieron las NC encontradas.

3.5 Conclusiones parciales

Se generó el diagrama de despliegue para entender la relación de dependencia, así como la relación entre los componentes hardware y software. Los estándares de codificación permitieron organizar el código de la propuesta de solución. Los diseños de casos de pruebas permitieron identificar los errores del sistema. También se aplicaron las pruebas de sistema y de aceptación las cuales permitieron detectar algunas no conformidades, las cuales fueron corregidas para mejorar la calidad y funcionalidad del componente.

Conclusiones generales:

El desarrollo del componente unido con el análisis de los resultados obtenidos, adecuaron la obtención de las siguientes conclusiones:

- Se logró un componente que permite la incorporación de metadatos en los Objetos de Aprendizaje bajo el estándar LOM en la Plataforma Educativa Xauce ZERA 2.1.
- Siguiendo la metodología AUP en su variante UCI y con los artefactos ingenieriles, las herramientas y tecnologías seleccionadas se logró el diseño y ejecución del componente desarrollado siguiendo las especificaciones del estándar LOM.
- El análisis de las soluciones similares estudiadas permitió determinar que ninguna de ellas satisfacía las necesidades del cliente tanto por restricciones de software como del negocio del sistema, por lo que se decidió desarrollar una solucion a la medida del cliente.
- Se aplicaron pruebas de aceptación y de sistema las cuales permitieron determinar y corregir las no conformidades encontradas.

Recomendaciones

A partir del desarrollo del presente trabajo de diploma se recomienda:

- Incluir en la propuesta de solución diversos estándares educativos con el fin de caracterizar los metadatos siguiendo diversos formatos.

Bibliografía

1. María de los Ángeles Serrano Islas. Revista e-Formadores: Objetos de Aprendizaje.

2. Yerandy MG, Adrián. GH. Plataforma educativa Zera, estándares utilizados para la gestión y distribución de recursos educativos. 2013.

3. Marks Dekkers. Dublincore: Metadatos para la interoperabilidad [Internet]. [citado 4 de mayo de 2018]. Disponible en: http://hipatia.uc3m.es/es/eventos/dcmies1/Dekkers_spa.pdf.

4. EcuRed. Conocimiento con todos y para todos EcuRed. [Internet]. Disponible en: http://www.ecured.cu/index.php/Aprendizaje_electr%C3%B3nico.

5. Chan, María Elena. OBJETOS DE APRENDIZAJE: una herramienta para la innovación educativa.

6. Recursos para formación a través de TIC. Estándares de e-learning [Internet]. 2013. Disponible en: https://elearningdocs.wordpress.com/2013/04/08/estandares-de-e-learning/

7. Judiht García Rodríguez. “Análisis y Diseño del nivel B de la especificación IMS-LD para el módulo de Diseño instruccional en CRODA 2.0”. La Habana; 2012.

8. URLEARNING. En qué consiste la especificación SCORM [Internet]. 2013. Disponible en: https://www.urlearning.eu/en-que-consiste-la-especificacion-scorm/

9. Mariano SE. CNICE Revista de Tecnologías de la Información y Comunicación Educativas.

10. Erik Duval. Modelos de agregación de contenidos. 2006.

11. Ana M. Martínez Tamayo, Julia C. Valdez, Edgardo A. Stubbs, Yanina González Terán, María Inés Kessler. Interoperabilidad de sistemas de organización del conocimiento: el estado del arte [Internet]. [citado 15 de mayo de 2018]. Disponible en: http://www.scielo.org.ar/scielo.php?script=sci_arttext&pid=S1851-17402011000100002

12. Estaban Gabriel Maida, Julian Pacienzia. Metodología de Desarrollo de Software [Internet]. 2015 [citado 27 de noviembre de 2017]. Disponible en: http://bibliotecadigital.uca.edu.ar/repositorio/tesis/metodologias-desarrollo-software.pdf

13. Betty Marlene Pachucho Hernández. Scribd. 5 de 2009 [Internet]. [citado 6 de abril de 2018]. Disponible en: http://es.scribd.com/doc/55136485/47/Ventajas-de-utilizar-un

14. Grady Booch, Ivar Jacobson, James Rumbaugh. El Lenguaje Unificado de Modelado. Manual de referencia. Madrid: Peason Education; 2000.

15. Visual Paradigm. [citado 27 de noviembre de 2017]. Disponible en: http://www.visual-paradigm.com/

16. Manuel Torres Remón. FUNDAMENTOS DE PROGRAMACIÓN, LENGUAJES Y TÉCNICAS DE PROGRAMACIÓN. 2012.

17. PHP.net. PHP. [En línea] [Internet]. [citado 1 de abril de 2018]. Disponible en: http://www.php.net/manual/es/intro-whatis.php

18. PhpStorm IDE de programación web. [online]. [Internet]. 2017 [citado 1 de abril de 2018]. Disponible en: http://www.editoresdecodigo.com/2014/06/descargar-phpstorm-full-ide-para-php-y-mas.html

19. Entendiendo HTML5: guía para principiantes [Internet]. 2013 [citado 28 de noviembre de 2017]. Disponible en: https://hipertextual.com/archivo/2013/05/entendiendo-html5-guia-para-principiantes/

20. Bienvenido a NetBeans y www.netbeans.org [Internet]. [citado 28 de noviembre de 2017]. Disponible en: https://netbeans.org/index_es.html

21. F. Potencier. Symfony la guía definitiva [Internet]. [citado 27 de noviembre de 2017]. Disponible en: http://librosWeb.es/

22. José Acedo. ¿Qué es el Framework Boostrap? Ventajas y Desventajas. [Internet]. [citado 15 de abril de 2018]. Disponible en: http://programacion.jias.es/2015/05/web-%c2%bfque-es-el-framework-bootstrap-ventajas-desventajas/

23. Angel Cabo Yera. Diseño y programación de base de datos. Madrid, España: Visión Libro; 2007.

24. PostgreSQL. PostgreSQL [Internet]. PostgreSQL web oficial. [citado 27 de noviembre de 2017]. Disponible en: http://www.postgresql.org.es

25. SHAME - Standardized Hyper Adaptible Metadata Editor [Internet]. 2006. Disponible en: kmr.nada.kth.se/shame/

26. RELOAD: Reusable eLearning Object Authoring & Delivery [Internet]. 2008. Disponible en: www.reload.ac.uk

27. Abraham Alonso Calvo, Elena Cañas Caicoya, Daniel Nieto Sanz. LOMEditor 2006: Editor de objetos de aprendizaje en base a plugins. 2005.

28. JOSÉ RUBÉN LAÍNEZ FUENTES. Desarrollo de Software Ágil: Extreme Programming y Scrum. 2ª Edición [Internet]. [citado 7 de marzo de 2018]. Disponible en: https://books.google.com.cu/books?id=TxRpCwAAQBAJ&pg=PA49&dq=Histori#v=onepage&q&f=false

29. ESCALONA MJ KOCH N. Ingeniería de Requisitos en Aplicaciones para la Web [Internet]. 2002 [Internet]. [citado 6 de mayo de 2018]. Disponible en: Recuperado a partir de: https://www.powtoon.com/about/

30. Requerimientos funcionales y no funcionales [Internet]. [citado 11 de marzo de 2018]. Disponible en: http://www.slideshare.net/Lismirabal/requerimientos-funcionales-y-no-funcionales

31. J.A. Galves., J. P. Gomes Gallego. Ingeniería de requerimientos. Universidad Tecnológica de Pereira. 2010.

32. Roger S Pressman. Ingeniería del Software: Un Enfoque Práctico. 2005.

33. Ricardo Botero Tabares. Patrones Grasp y Anti-Patrones: un Enfoque Orientado a Objetos desde Lógica de Programación. Entre Ciencia e Ingeniería. 2011. No. 8, p. 161–173. USA: UR. FI-INCO; 2011.

34. L. Hernández Montenegro. Uso de patrones de diseño de software. Vol. 59. 2012.

35. David Hernández Tejada. TELEPROGRAMADORES. GUIA DE PATRONES DE DISEÑO. [Internet]. [citado 20 de mayo de 2018]. Disponible en: http://www.teleprogramadores.com.

36. Christopher Alexander, Murray Silverstein, Sara Ishikawa. El Lenguaje de Patrones [Internet]. [citado 12 de marzo de 2018]. Disponible en:

https://books.google.com.cu/books?id=WHArAAAACAAJ&dq=Patrones+de+Arquitectura.&hl=es&sa=X&redir_esc=y

37. Romero González, Yenisleydi Fernández, Yanette Díaz. Patrón modelo-vista-controlador. Revista Telem@tica. 2012; 11:47-57.

38. Ramez Elmasri, Shamkant B. Navathe. Fundamentos de Sistemas de Bases de Datos.

39. Carlos Blanco Bueno. open course ware [Internet]. [citado 15 de mayo de 2018]. Disponible en: http://ocw.unican.es/ensenanzas-tecnicas/ingenieria-del-softwareii/materiales.

40. Roberto Tenorio Ruiz. Las Pruebas de Software y su Importancia en las Organizaciones. 2018.

41. Ian SOMMERVILLE. Ingeniería del software. 7.ª ed. 2005.

42. Natalia Juristo, Ana M Moreno, Sira Vegas. TÉCNICAS DE EVALUACIÓN DE SOFTWARE. 2018.

43. Francisco Javier Martínez. Departamento de informática [Internet]. 2018. Disponible en: http://di002.edv.uniovi.es/~cueva/asignaturas/PFCOviedo/PFCpatronesJava.pdf.

44. Alex Prezi Melchor. Tutorial Diagramas de Despliegue [Internet]. [citado 19 de mayo de 2018]. Disponible en: https://prezi.com/e_gpb7xev_im/tutorial-diagramas-de-despliegue

Anexo 1

Historia de usuario 1 Insertar metadatos	
Número: 1	**Nombre del requisito:** Insertar metadatos
Programador: Alexnurín Columbié Méndez	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** 2 días

Descripción:

1- Objetivo:

Permitir insertar metadatos.

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

Para insertar un metadato es necesario:

- Estar autenticado en el sistema con el rol X.
- Acceder al botón Desde Archivo donde aparecerán los elementos del estándar
- Tener en cuenta los siguientes datos: Escoger la categoría del estándar LOM que va a llenar con los elementos de información que sean requeridos.

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

El campo título es el único que será obligatorio, los demás serán opcionales.

Título: campo de texto que admite caracteres alfabéticos y tiene un máximo de hasta 100 caracteres

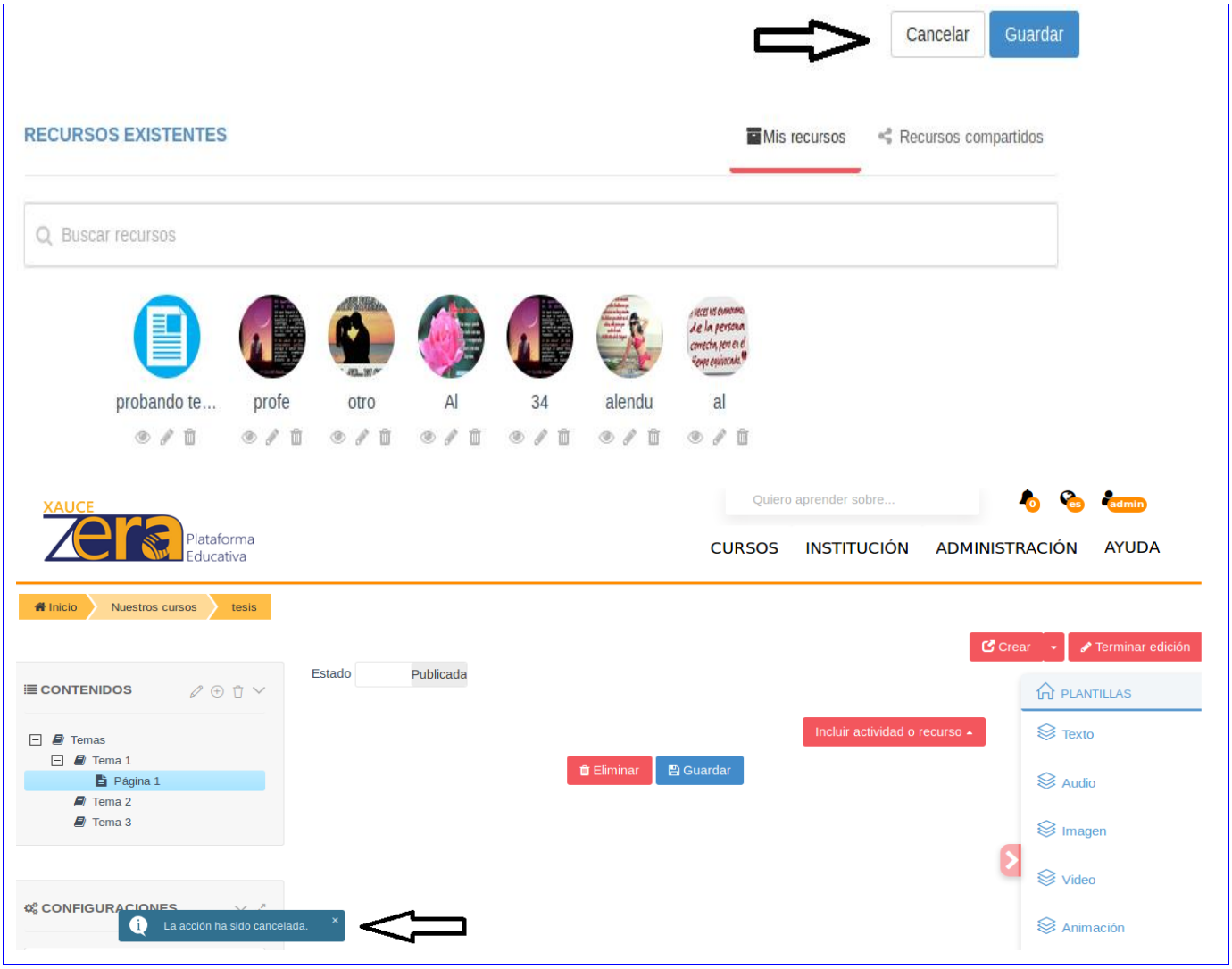
4- Flujo de la acción a realizar:

- Cuando el usuario inserta un metadato y selecciona la opción Guardar, se crea un nuevo Objeto de Aprendizaje y el sistema muestra un mensaje de información.
- Si selecciona la opción Cancelar regresará a la vista anterior.

Observaciones: Estar autenticado en el sistema con el rol X.

Prototipo de interfaz:

Historia de usuario 3 Deshacer cambios efectuados en los metadatos	
Número: 3	**Nombre del requisito:** Deshacer cambios efectuados en los metadatos
Programador: Alexnurín Columbié Méndez	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** 2 días
Descripción: **1- Objetivo:** Permitir deshacer cambios efectuados en los metadatos. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para deshacer cambios hay que: - Estar autenticado en el sistema con el rol X. - Al menos un campo se haya completado y no se haya guardado la información en el sistema. **3- Flujo de la acción a realizar:** Cuando el usuario selecciona la opción deshacer cambios se borrarán los cambios introducidos por el usuario que no hayan sido guardados.	
Observaciones: Estar autenticado en el sistema con el rol X.	
Prototipo de interfaz:	



Historia de usuario 4 Mostrar todos los metadatos	
Número: 4	**Nombre del requisito:** Mostrar todos los metadatos
Programador: Alexnurín Columbié Méndez	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** 2 días
Descripción: **1- Objetivo:** Permitir mostrar todos los elementos del estándar LOM. **2- Acciones para lograr el objetivo (precondiciones y datos):**	

Para mostrar los elementos del estándar LOM hay que:

- Estar autenticado en el sistema con el rol X.

3- Flujo de la acción a realizar:

- El usuario podrá escoger la opción de mostrar los elementos del estándar LOM.

Observaciones: Estar autenticado en el sistema con el rol X.

Prototipo de interfaz:

Historia de usuario 5 Mostrar ayuda para especificar los elementos de información.	
Número: 5	**Nombre del requisito:** Mostrar ayuda para especificar los elementos de información.
Programador: Alexnurín Columbié Méndez	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** 2 días

Descripción:

1- Objetivo:

Permitir mostrar ayuda para especificar los elementos de información.

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

Para mostrar ayuda se necesita que:

- Estar autenticado en el sistema con el rol X.
- Se debe llenar los campos requeridos para insertar un Objeto de Aprendizaje.

3- Flujo de la acción a realizar:

- Al llenar los campos de información al lado de cada elemento de información aparecerá un ícono, cada vez que el usuario de un clic sobre el mismo, el sistema mostrará el significado del elemento y un pequeño ejemplo.

Observaciones: Estar autenticado en el sistema con el rol X.
Prototipo de interfaz:

Historia de usuario 6 Exportar metadatos.	
Número: 6	**Nombre del requisito:** Exportar metadatos.
Programador: Alexnurín Columbié Méndez.	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** 2 días

Descripción:

1- Objetivo:

Permitir exportar los metadatos al sistema.

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

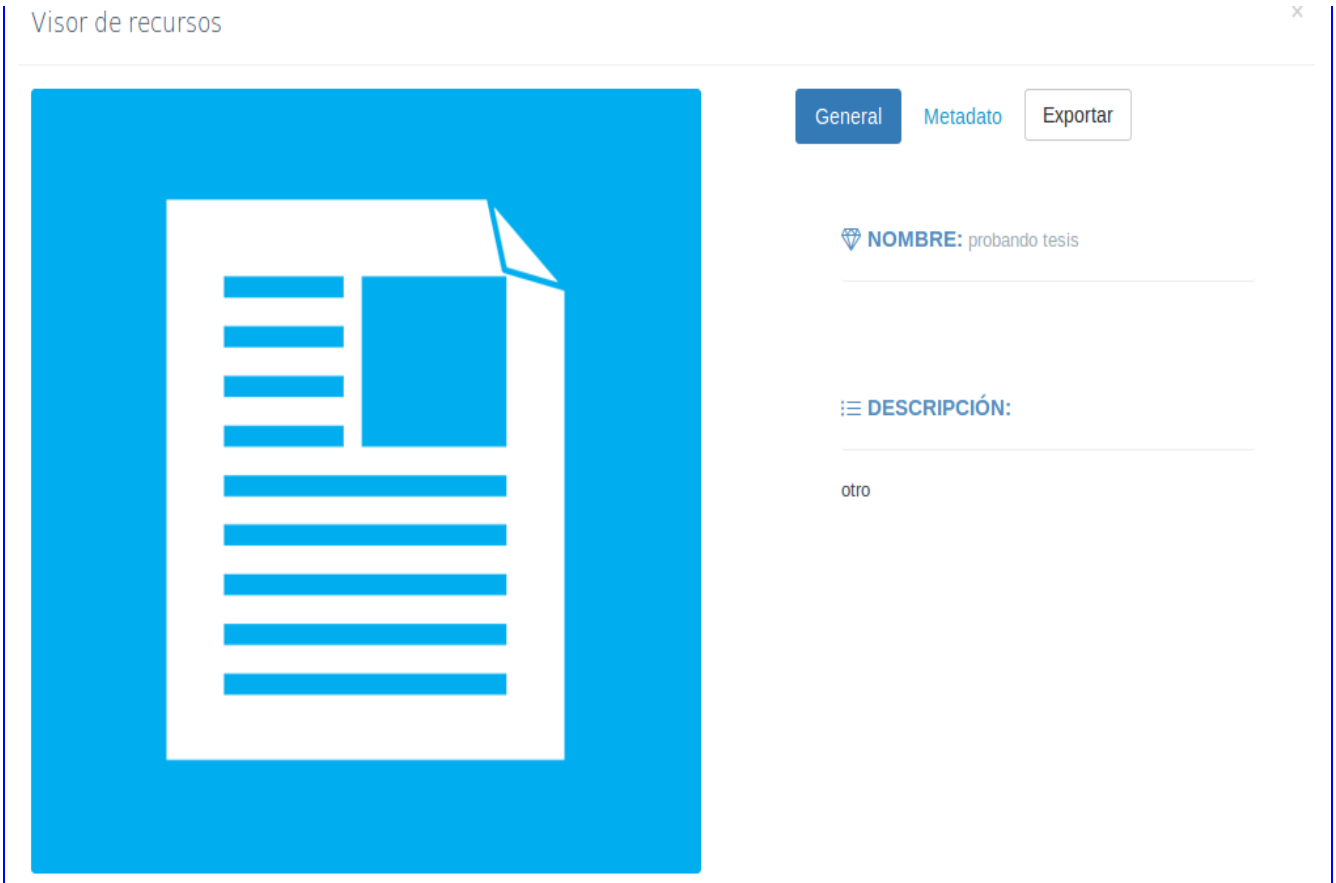
Para exportar metadatos hay que:

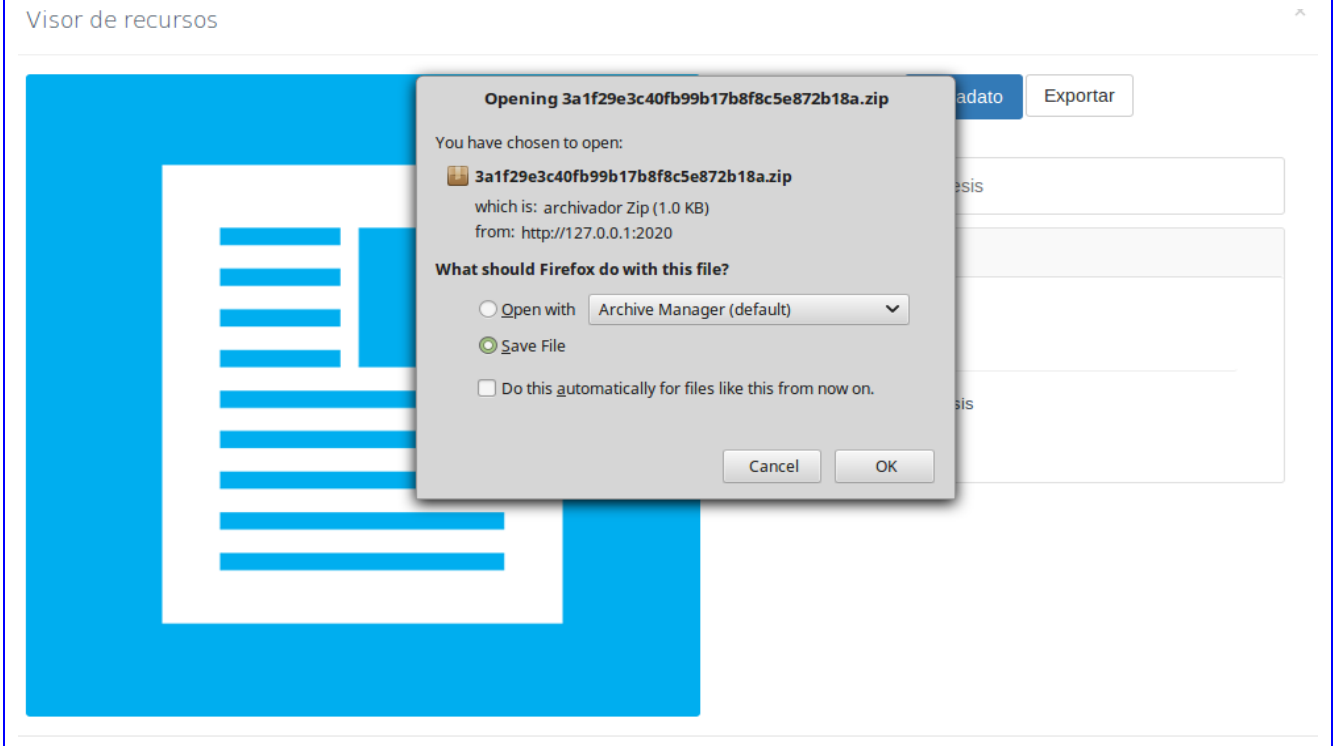
- Estar autenticado en el sistema con el rol X.
- Se debe de crear un Objetos de Aprendizaje antes de exportar un archivo.

3- Flujo de la acción a realizar:

-El usuario escoge la opción exportar archivo y se le muestra un cuadro con una dirección especifica donde seleccionara el Objeto de Aprendizaje si es que existe, en caso de no existir no se podrá exportar ningún archivo.

Observaciones: Estar autenticado en el sistema con el rol X.
Prototipo de interfaz:


Visor de recursos
General
Metadato
Exportar
NOMBRE: probando tesis
DESCRIPCIÓN:
otro


Visor de recursos
Exportar
Opening 3a1f29e3c40fb99b17b8f8c5e872b18a.zip
You have chosen to open:
3a1f29e3c40fb99b17b8f8c5e872b18a.zip
which is: archivador Zip (1.0 KB)
from: http://127.0.0.1:2020
What should Firefox do with this file?
Open with
Archive Manager (default)
Save File
Do this automatically for files like this from now on.
Cancel
OK

Historia de usuario 7 Importar metadatos.	
Número: 7	**Nombre del requisito:** Importar metadatos
Programador: Alexnurín Columbié Méndez.	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** 2 días
Descripción: **1- Objetivo:** Permitir importar metadatos en el sistema. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para exportar la agenda hay que: - Estar autenticado en el sistema con el rol X. - Debe de haberse creado y guardado antes un Objeto de Aprendizaje. **3- Flujo de la acción a realizar:** -Cuando el usuario selecciona la opción importar archivo se le mostrará un cuadro con una dirección especifica donde seleccionará el archivo a importar.	
Observaciones: Estar autenticado en el sistema con el rol X.	
Prototipo de interfaz:	

Historia de usuario 7 Visualizar metadatos.	
Número: 8	**Nombre del requisito:** Visualizar metadatos.
Programador: Alexnurín Columbié Méndez.	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** 2 días
Descripción: **1- Objetivo:** Permitir visualizar metadatos en el sistema. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para visualizar metadatos hay que:	

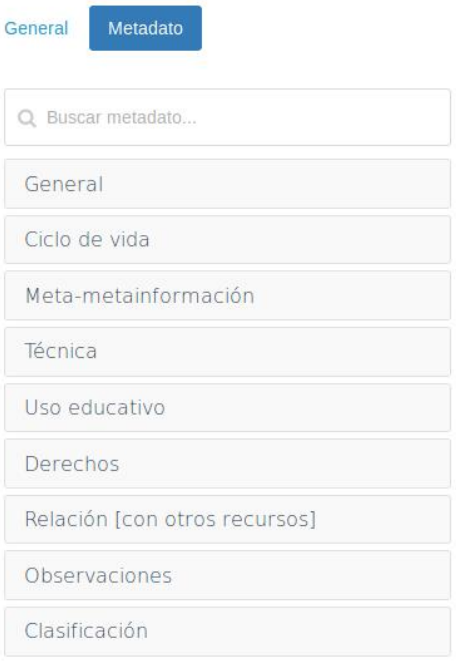
- Estar autenticado en el sistema con el rol X.
- Debe existir al menos un Objeto de Aprendizaje para visualizar.

3- Flujo de la acción a realizar:

-Cuando el usuario selecciona la opción visualizar recurso aparecerá un cuadro publicando todos los Objetos de Aprendizaje existentes.

Observaciones: Estar autenticado en el sistema con el rol X.

Prototipo de interfaz:



Historia de usuario 9 Buscar metadatos.	
Número: 9	**Nombre del requisito:** Buscar metadatos.
Programador: Alexnurín Columbié Méndez.	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** 2 días

Descripción:

1- Objetivo:

Permitir buscar todos los metadatos asociados a un recurso.

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

Para buscar metadatos hay que:

- Estar autenticado en el sistema con el rol X.
- Debe existir en el sistema al menos un recurso.

3- Flujo de la acción a realizar:

Cuando el usuario selecciona la opción buscar metadatos aparecen todos los metadatos asociados a un recurso.

Observaciones: Estar autenticado en el sistema con el rol X.

Prototipo de interfaz:

Historia de usuario 10 Eliminar metadatos.	
Número: 10	**Nombre del requisito:** Eliminar metadatos.
Programador: Alexnurín Columbié Méndez.	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** 2 días

Descripción:

1- Objetivo:

Permitir eliminar todos los metadatos asociados a un recurso.

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

Para eliminar los metadatos hay que:

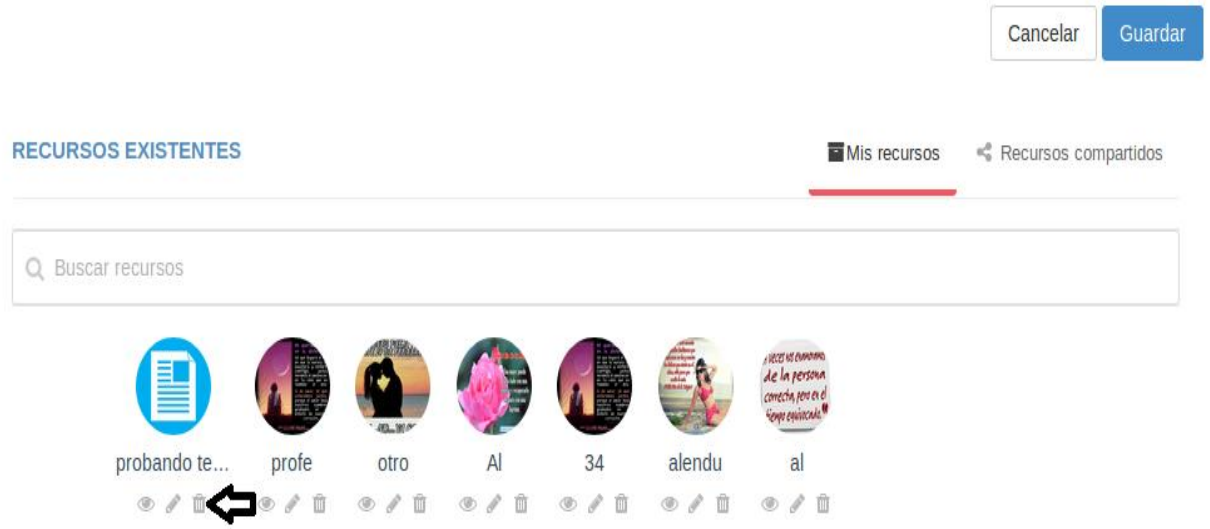
- Estar autenticado en el sistema con el rol X.
- Debe existir en el sistema al menos un recurso.

3- Flujo de la acción a realizar:

Cuando el usuario selecciona la opción eliminar metadatos estos se eliminarán conjuntamente con el recurso asociado.

Observaciones: Estar autenticado en el sistema con el rol X.

Prototipo de interfaz:



Anexo 2

Tabla 5: Diseño de caso de prueba Guardar metadatos

Escenario	Descripción	Revisión	Respuesta del sistema	Flujo central
EC 1.1 Opción Guardar	El usuario selecciona la opción guardar		El sistema guardará los metadatos	Inicio/Acceder/Modo Edición/Incluir actividad o recurso/incluir recurso/Guardar

Tabla 6: Diagrama de caso de prueba Deshacer cambios

Escenario	Descripción	Revisión	Respuesta del sistema	Flujo central

EC 1.1 Opción Cancelar	El usuario selecciona la opción cancelar		El sistema cancelará la acción	Inicio/Acceder/Modo Edición/Incluir actividad o recurso/incluir recurso/Cancelar

Tabla 7: Diseño de caso de prueba Exportar metadatos

Escenario	Descripción	Revisión	Respuesta del sistema	Flujo central
EC 1.1 Opción Visor de recurso	El usuario selecciona la opción Desde Archivo y seguido de esta selecciona la opción Visor de recurso		El sistema mostrará todos los metadatos que tiene el recurso. Además brinda la opción siguiente: -Buscar metadato	Inicio/Acceder/Modo Edición/Incluir actividad o recurso/incluir recurso/Visor de recurso/Exportar metadato

Tabla 8: Diagrama de casos de prueba Importar metadatos

Escenario	Descripción	Revisión	Respuesta del sistema	Flujo central
EC 1.1 Opción Desde Archivo	El usuario selecciona la opción Desde Archivo y seguido de esta selecciona la opción Importar metadato		El sistema mostrará una ventana donde se podrá acceder a la dirección donde está el recurso comprimido que se desea importar	Inicio/Acceder/Modo Edición/Incluir actividad o recurso/incluir recurso/Importar metadato

Tabla 9: Diagrama de caso de prueba Mostrar metadatos

Escenario	Descripción	Revisión	Respuesta del sistema	Flujo central

EC 1.1 Opción Desde Archivo	El usuario selecciona la opción Desde Archivo		El sistema mostrará todos los metadatos para crear un recurso	Inicio/Acceder/Modo Edición/Incluir actividad o recurso/incluir recurso/Desde Archivo

Tabla 10: Diagrama de caso de prueba Buscar metadato

Escenario	Descripción	Revisión	Respuesta del sistema	Flujo central
EC 1.1 Opción Visor de recurso	El usuario selecciona la opción Visor de recurso		El sistema mostrará todos los metadatos para crear un recurso Además brindará la opción siguiente: -exportar	Inicio/Acceder/Modo Edición/Incluir actividad o recurso/incluir recurso/Visor de recurso/Buscar metadato
EC 1.2 No hay metadatos	El usuario selecciona la opción Visor de recurso y no hay metadatos asociados al recurso		El sistema no encuentra los metadatos del recurso y no tendrá ningún criterio de búsqueda y mostrará solamente el nombre y la descripción del recurso Además brindará la opción siguiente: -exportar	Inicio/Acceder/Modo Edición/Incluir actividad o recurso/incluir recurso/Visor de recurso/Buscar metadato

Tabla 11: Diagrama de caso de prueba Visualizar metadato

Escenario	Descripción	Revisión	Respuesta del sistema	Flujo central
EC 1.1 Opción Visor de recurso	El usuario selecciona la opción Visor de recurso		El sistema mostrará todos los metadatos del recurso	Inicio/Acceder/Modo Edición/Incluir actividad o recurso/incluir recurso/Visor de recurso
EC 1.2 No hay metadatos	El usuario selecciona la opción Visor de recurso		El sistema no mostrará los metadatos del recurso solo el nombre y la descripción	Inicio/Acceder/Modo Edición/Incluir actividad o recurso/incluir

			recurso/Visor de recurso

Anexo 3

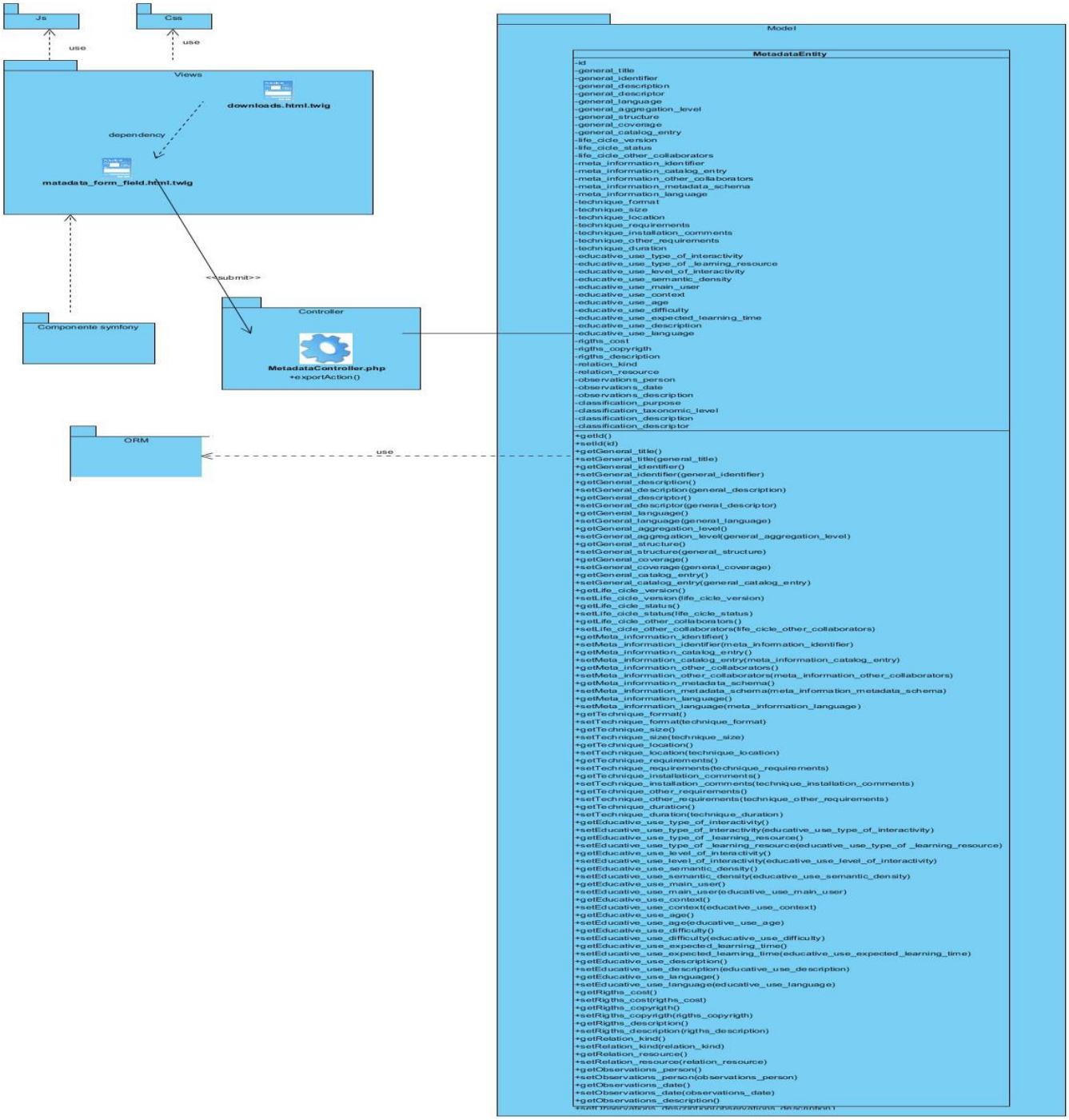


Figura 5: Diagrama de clase de diseño. Exportar metadatos.