Universidad de las Ciencias Informáticas

Facultad 4

Título: “Componente de revisión de recursos educativos para la plataforma educativa Zera 2.0”.

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

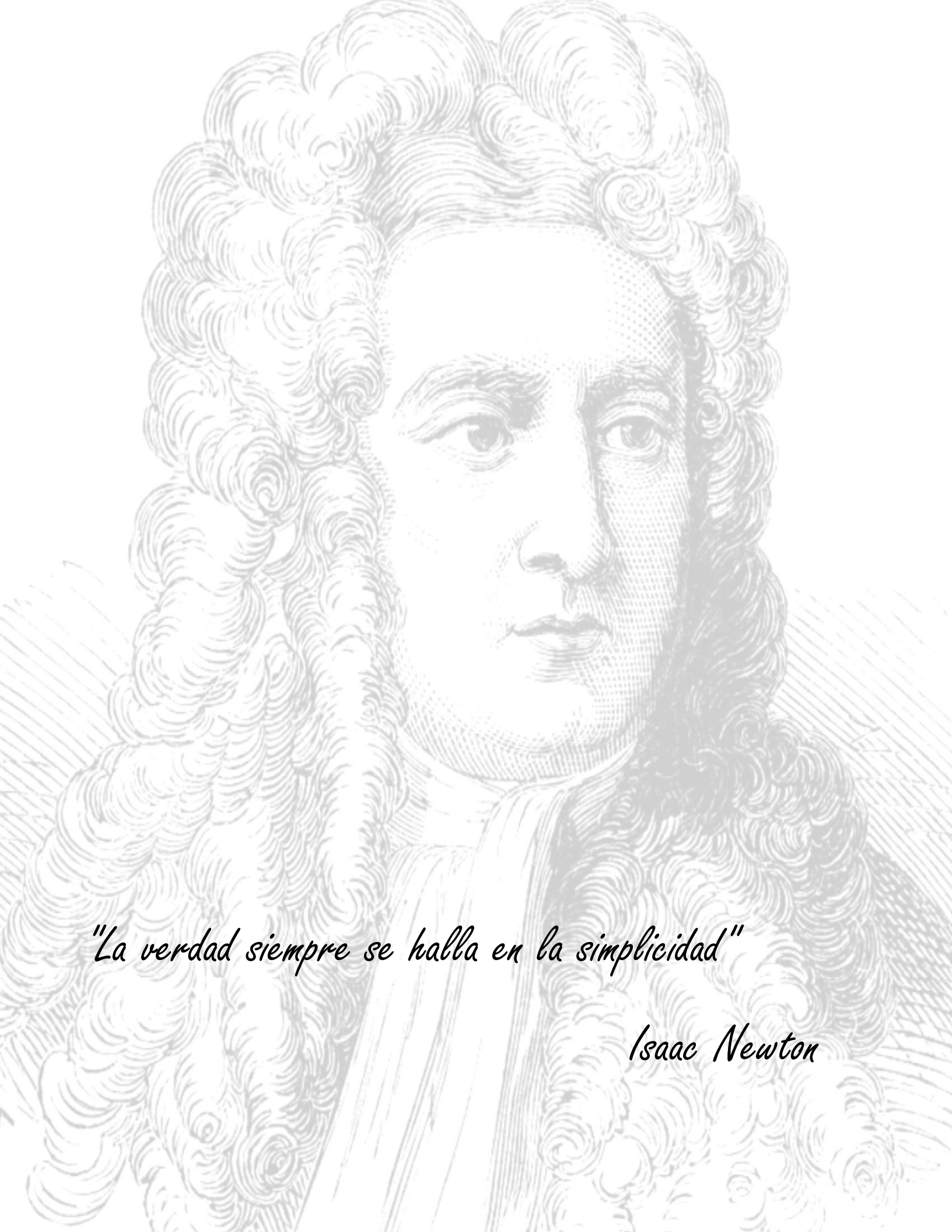
Autor: Pedro Pablo Suárez Govea.

Tutor: Ing. Nadia Sánchez Batista

Ing. Yandris Mata Cabrera

La Habana, junio de 2016

‘‘Año 58 de la Revolución’’



"La verdad siempre se halla en la simplicidad"

Isaac Newton

Declaración de Autoría

Declaramos ser los autores del presente trabajo de diploma y otorgamos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

_____

Firma del Autor

Pedro Pablo Suárez Govea

_____ _____

Firma del Tutor Firma del Tutor

Ing. Nadia Sánchez Batista Ing. Yandris Mata Cabrera

Datos de contacto

Pedro Pablo Suárez Govea

Universidad de las Ciencias Informáticas, La Habana, Cuba

Correo electrónico: ppsuarez@estudiantes.uci.cu

Ing. Nadia Sánchez Batista

Universidad de las Ciencias Informáticas, La Habana, Cuba

Correo electrónico: nbatista@uci.cu

Ing. Yandris Mata Cabrera

Universidad de las Ciencias Informáticas, La Habana, Cuba

Correo electrónico: ymata@uci.cu

Agradecimientos

A todos mis amigos, familiares y compañeros.

A todas aquellas persona que me ayudaron en el desarrollo de este trabajo o se preocuparon por el mismo.

Resumen

Los procesos de revisión constituyen un aspecto fundamental para garantizar la calidad del contenido educativo. Existen diversos tipos de procesos de revisión, dentro de los que se encuentran revisión simple, por pares, por roles, colaborativa, automática, entre otros. En la Universidad de las Ciencias Informáticas (UCI), se desarrolla la plataforma educativa Zera2, este sistema gestiona contenido educativo dirigido al proceso de enseñanza aprendizaje. Dicha información necesita contar con un nivel de calidad para evidenciar la veracidad de los cursos que se gestionan en la plataforma.

La presente investigación tiene como objetivo desarrollar un componente de revisión que realice varios procesos, apoyado por criterios de evaluación que puedan configurarse en dependencia del tipo de recurso a revisar. Como resultado de la investigación, se generaron los principales artefactos del flujo de trabajo que establece la metodología de desarrollo *Agile Unified Process* versión UCI. Se desarrolló un componente de revisión compuesto por revisiones simple, por pares, por roles y colaborativa. Además se realizaron pruebas que permitieron identificar y erradicar los errores presentes en el componente, para posteriormente corregirlos.

Palabras calves: componente**,** plataforma educativa, revisión.

Índice

Introducción:

Los procesos de revisión constituyen un aspecto importante en las diferentes áreas del saber, los cuales garantizan la calidad en dichas áreas. El diccionario Manual de la Lengua Española Vox define la revisión como "*prueba o examen que se hace para comprobar que algo funciona correctamente*" (1) y proceso como el "*conjunto de las diferentes fases o etapas sucesivas que tiene una acción o un fenómeno complejos*" (1). Por otra parte, *K Dictionaries Ltd* define revisión como "*examen de una cosa hecha con detenimiento*" (2). Basándose en los conceptos antes mencionados, el autor de esta investigación define como procesos de revisión el conjunto fases y acciones realizadas para probar el correcto funcionamiento y/o veracidad de algo.

En el mundo contemporáneo, la calidad de la comunicación científica resulta un tema de interés para los académicos e investigadores de las diferentes áreas del saber. En tal sentido, la publicación científica es un tema muy debatido, por considerarse que todo lo publicado en una revista científica es garantía de su validez. Pero existe una gran diversidad de procesos de revisión, los cuales son aplicados indistintamente por las editoriales de las revistas científicas. Algunas de ellas poseen consejos editoriales que utilizan como sistema de revisión a la revisión por pares (RP) para la selección de publicaciones.

Los procesos de revisión se apropian del desarrollo alcanzado por las Tecnologías de la Información y las Comunicaciones (TIC) con el objetivo de dar respuestas a las múltiples demandas de la sociedad contemporánea. Dichos procesos son altamente necesitados en el proceso de aprendizaje electrónico (e-learning), el cual se define como la "*modalidad del proceso enseñanza aprendizaje que consiste en el diseño, puesta en práctica y evaluación de un curso o plan formativo desarrollado a través de redes de ordenadores y puede definirse como una educación o formación ofrecida a individuos que están geográficamente dispersos o separados o que interactúan en tiempos diferidos del docente empleando los recursos informáticos y de telecomunicaciones*" (3). El e-learning es una opción factible para el proceso de enseñanza aprendizaje, al superar las barreras de espacio y tiempo, lo que ha abierto el camino hacia una mejor comunicación entre los diferentes actores, al permitir la construcción y autogestión del conocimiento.

El uso de los sistemas e-learning ha propiciado la creación de contenido educativo, el que para ser utilizados de manera adecuada, necesita ser estandarizado para su posterior reutilización y recuperación. En respuesta a este problema surgen los objetos de aprendizajes (OA), "*son cualquier entidad, digital o no digital, la cual puede ser usada, re-usada o referenciada durante*

el aprendizaje apoyado por tecnología" (4). La característica más relevante de los OA es la posibilidad de reutilización y adaptabilidad que estos brindan.

En la actualidad existe gran cantidad de plataformas educativas, entre ellas destacan Edmodo[1], que es "*una plataforma social que facilita la comunicación y la interacción virtual como complemento de la presencialidad, un ambiente de aprendizaje donde los involucrados pueden ser Directivos, Docentes, Estudiantes y hasta padres de familia*" y Moodle[2], que es "*aula virtual por excelencia utilizada en múltiples ámbitos. Es un paquete de software para la creación de cursos y sitios Web basados en Internet para dar soporte a un marco de educación social constructivista*". Estas plataformas comprueban la calidad de su contenido educativo a través de procesos que revisan y evalúan los recursos educativos, haciendo uso de varios métodos y criterios.

En el Centro de Tecnologías para la Formación (FORTES) de la Universidad de la Ciencias Informáticas (UCI) se desarrolla la plataforma para cursos masivos abiertos en línea (Zera 2.0), esta plataforma gestiona su contenido en forma de recursos educativos (RE), estos "*son todos aquellos medios empleados por el docente para apoyar, complementar, acompañar o evaluar el proceso educativo que dirige u orienta*" (5), dicho concepto es muy similar a el de un OA. El contenido que se gestiona en Zera 2.0 debe contar con una calidad que respalde su información, debido a que si los recursos no contienen informacion confiable o correcta, el curso perdería confiabilidad y por consecuencia no cumpliría con el objetivo del mismo. Para mejorar este servicio son imprescindibles los procesos de revisión indistintamente del formato en que sea contenida dicha informacion.

Tomando en consideración lo anteriormente expresado y el cambio de modelo de desarrollo del centro FORTES a líneas de productos de software se plantea como **problema a resolver**:¿Cómo contribuir a elevar la calidad de los recursos educativos gestionados en la plataforma educativa Zera 2.0?

Se establece como **objeto de estudio:** los procesos de revisión en herramientas que gestionan contenido.

Como **campo de acción:** los procesos de revisión en herramientas que gestionan contenido en

[1] Disponible en: https://www.edmodo.com/?language=es

[2] Disponible en: http://moodle.org/

la plataforma educativa Zera 2.0.

Para dar solución al problema de investigación, se define como **objetivo general:** desarrollar un componente de revisión para contribuir a elevar la calidad de los recursos educativos gestionados en la plataforma educativa Zera2.0.

Se plantean como **objetivos específicos:**

- Realizar un estudio del estado del arte de los procesos de revisión de recursos educativos.
- Generar los artefactos principales de los flujos de trabajo según la metodología AUP-UCI.
- Implementar el componente de revisión que pueda ser integrado al marco de trabajo que utiliza la plataforma educativa Zera 2.0.
- Validar las funcionalidades de los procesos de revisión implementados en el componente a través de pruebas unitarias y de aceptación.

Se plantea como **preguntas científicas**:

- ¿Cómo se presentan en la literatura científica los procesos de revisión de recursos educativos?
- ¿Cuáles son los procesos de revisión más utilizados y cómo son aplicados en las plataformas educativas?
- ¿Qué procesos de revisión existentes pueden ser empleados en la plataforma educativa Zera 20 para contribuir a elevar la calidad de los RE?

Con el fin de resolver y dar cumplimiento a los objetivos se emplearon los siguientes métodos de investigación:

Método teórico:

- Histórico-Lógico: permitió identificar tecnologías, lenguajes y metodologías de desarrollo a utilizar en la solución propuesta.

Método empírico:

- Observación: Posibilitó tomar las mejores prácticas de sistemas similares con el mismo fin de la herramienta a desarrollar. Además, posibilitó realizar las pruebas al producto

desarrollado.

El documento está organizado en tres capítulos, de la siguiente forma:

- Capítulo I: Fundamentación teórica:

  Abarca los conceptos asociados al tema, necesarios para la comprensión de la solución del problema planteado. Se exponen los elementos teóricos utilizados en la investigación, describiendo además las tecnologías, metodología, herramientas y lenguajes de programación utilizados en el desarrollo de la solución.

- Capítulo II: Propuesta de solución:

  En este capítulo son descritos los principales conceptos relacionados con la investigación a través de un modelo de dominio. Se identifican los requisitos funcionales y no funcionales. Se obtienen los diagramas de clases de diseño, además es construido el modelo de datos y son realizadas las descripciones correspondientes.

- Capítulo III: Implementación y prueba:

  Se describen los principales aspectos del desarrollo, se definen los tipos de pruebas y los casos de prueba realizados al software.

Capítulo 1: Fundamentación Teórica

En el presente capítulo se describen los procesos de revisión y se dan a conocer conceptos relacionados con este tema. Se destacan las principales características de los elementos asociados a la temática, así como las herramientas empleadas para la solución de la problemática.

1.1 Procesos de revisión

Las revisiones en los ámbitos académicos se destacan por ser un método fiable para garantizar la calidad, originalidad y rigor científico en una publicación. Si se confía en una revista académica, es en cierto punto por el proceso de revisión de los artículos que se publican en ellas, sistemas con altos estándares de calidad prestigian a las publicaciones que revisan, creando una retroalimentación entre el proceso y los objetos que analiza. El volumen de información científica que es publicada en la actualidad ha hecho necesario el desarrollo de cada vez más complejos procesos de revisión que respalden su veracidad. (6)

La evolución de las revistas científicas ha permitido desarrollar diversos modelos del proceso de revisión existiendo diferentes tipos.

1.1.1 Revisión por pares

La revisión por pares (RP) puede ser vista como un proceso complejo y riguroso de gran importancia que se usa para validar contenido de diferentes tipos. El propósito de la revisión por pares es medir la calidad, factibilidad y credibilidad de las investigaciones con miras a ser publicadas.

Según el Lic. Ernesto G. Rodríguez "*la RP no es más que la revisión del manuscrito por especialistas en la temática con grado científico equivalente al de los autores (pares, a la par; llamados comúnmente revisores, árbitros, evaluadores o expertos), en número que puede variar según las características del proceso de revisión y tipo de manuscrito. Los revisores evalúan generalmente de forma independiente el manuscrito y recomiendan una decisión a tomar: la posible aceptación del manuscrito, tras atender las exigencias de la revista, o su rechazo, por no satisfacerlas. Una vez que ellos han emitido su evaluación, un editor a cargo colecta las observaciones y dictamina basado en ellas, y les comunica a los autores las acciones a tomar para que perfeccionen los puntos débiles del manuscrito si el dictamen es favorable a la publicación. Los autores tienen la opción de satisfacer los requerimientos de los revisores y*

obtener retroalimentación de estos en una serie de reenvíos (comúnmente hasta dos). Así se verifica y complementa la calidad del manuscrito, antes de su procesamiento documental y publicación." (7)

La bibliografía especializada define tres modalidades fundamentales de RP, que se diferencian en su forma solo por el conocimiento de la identidad de los autores y revisores. (6)

- Simple Ciego: el revisor conoce la identidad del autor, pero el autor no conoce la del revisor. Este método puede ser vulnerable al nepotismo, sus fundamentos éticos han venido bajo crítica.
- Abierta: revela las identidades de autores y revisores, y los autores tienen la capacidad de identificar los comentarios de los revisores.
- Doble Ciego: tanto los revisores como los autores son anónimos. Actualmente es la alternativa más significativa, debido a que elimina cualquier pista o señal que ayude a identificar a los autores o revisores. Busca preservar el anonimato, para asegurar así que la revisión se haga de forma justa. En ocasiones es difícil ocultar la identidad de un autor, particularmente si se empeña en darse a conocer, mediante autocitas en trabajos previos.

Varios estudios han sugerido que los artículos que antes de ser publicados pasaron por una revisión por pares Doble Ciego, fueron más citados que los artículos publicados por otro tipo de revisión. (6)

1.1.2 Revisión simple

Es un proceso a realizar por una sola persona. Se centra mayormente en un área o tema en específico. La revisión se realiza de manera similar a la revisión por pares, a diferencia de que solo interactúa una persona. Se debe hacer uso igualmente de las metodologías para la evaluación de los recursos. Su objetivo es comprobar y evaluar los recursos educativos a publicar. Al intervenir una sola persona el proceso es más ágil e interactivo. De su correcta aplicación depende la calidad de un recurso que será usado en un proceso de enseñanza aprendizaje. (8)

1.1.3 Revisión por roles

Debe ser ejecutada o llevada a cabo por equipos conformados por roles, los cuales centrarán la revisión en aspectos específicos. El revisor general sería la persona encargada de confeccionar los equipos de revisión. El editor será la persona encargada de asignar los roles y las revisiones a realizar en el sistema. El proceso descrito se lleva a cabo mediante un equipo de trabajo, cuyo

resultado final será dado cuando se obtenga el criterio de cada revisor. De igual forma se hace uso de los instrumentos o metodologías de evaluación de los recursos. (8)

1.1.4 Revisión colaborativa

Este método somete a los usuarios en la revisión de los recursos con el fin de lograr la calidad requerida en los mismos. El objetivo consiste en que una vez se publica el recurso, ya sea en una plataforma o un repositorio, los destinatarios finales del mismo detecten irregularidades y deficiencias en el mismo, participando de forma directa en el proceso de revisión. (8)

1.1.5 Revisión automática

Es un proceso donde no existe la intervención humana. A través de un sistema o aplicación se realizan las comprobaciones necesarias al recurso antes de ser publicado. El software garantiza la no repetición del elemento, así como también la verificación de otros parámetros como el tamaño del recurso y cumplimiento de estándares que permitan la obtención de modelos comunes de información. Por otra parte es permitido mencionar que este método no es capaz de detectar el plagio. (8)

En la actualidad el tema de las revisiones de publicaciones científicas y contenido educativo es muy controversial, no quedando exentas de este debate las plataformas de aprendizaje electrónico (e-learning) las cuales necesitan que la información que en ellas se muestra sea veraz y esté correctamente enfocada a sus consumidores. (6)

1.2 Plataforma e-learning (LMS Learning Management System)

En la actualidad el concepto de *e-learning* como modalidad educativa es más usado y reconocido dentro del mundo de la educación. Dicho término puede ser definido como "*educación y capacitación a través de Internet".* (9). Este tipo de enseñanza en línea permite la interacción del usuario con el material mediante la utilización de diversas herramientas informáticas o plataformas e-learning las cuales pueden ser definidas como *"un espacio virtual de aprendizaje orientado a facilitar la experiencia de capacitación a distancia, tanto para empresas como para instituciones educativas*" (10). Estos sistemas permiten la creación de "aulas virtuales"; en ellas se produce la interacción entre tutores y alumnos, y entre los mismos alumnos; como también la realización de evaluaciones, el intercambio de archivos y una amplia gama de herramientas adicionales. Entre los beneficios de las plataformas e-learning destacan que estas: (10)

- Brindan capacitación flexible y económica.
- Combinan el poder de Internet con el de las herramientas tecnológicas.

- Anulan las distancias geográficas y temporales.
- Permiten utilizar la plataforma con mínimos conocimientos.
- Posibilitan un aprendizaje constante y nutrido a través de la interacción entre tutores y alumnos.
- Ofrecen libertad en cuanto al tiempo y ritmo de aprendizaje.

1.3 Recurso educativo (RE)

Los recursos educativos pueden ser clasificados como objetos o recursos digitales y según P. Mocoso en ellos: "*el valor de la información es impredecible, depende de aspectos muy diversos, y, sobre todo, de quien la utilice*..." (11). La intención de estos desde un primer momento es transmitir una información y generar nuevos conocimientos a través de elementos de la didáctica y la pedagogía. Según Ariño (12) y Cañizares (8) estos pueden ser cursos completos o materiales, multimedias, OA o cualquier elemento con fines educativos, siempre y cuando cuenten con una alta pertinencia y certificación de la calidad.

1.4 Objeto de aprendizaje (OA)

Como parte del contenido educativo gestionado en cualquier plataforma e-learning se encuentran los OA, que permiten la organización de contenidos digitales que se encuentren en distintas localizaciones y se requiera su utilización en diferentes contextos. Además, su uso ofrece innumerables ventajas para estudiantes, educadores e investigadores. Un OA según el Ministerio de Educación Nacional Colombiano "*es un conjunto de recursos digitales, autocontenible y reutilizable, con un propósito educativo y constituido por al menos tres componentes internos: contenidos, actividades de aprendizaje y elementos de contextualización. El objeto de aprendizaje debe tener una estructura de información externa (metadatos) que facilite su almacenamiento, identificación y recuperación*". (13)

Todas las características de los OA, son compactadas por el Modelo de Referencia de Objetos Compartibles de Contenido (*SCORM, Sharable Content Object Reference Model*).El modelo SCORM, representa un medio pedagógicamente neutro para los diseñadores y desarrolladores de OA, es decir, trata específicamente los aspectos concernientes a la dimensión tecnológica, a través de un conjunto de especificaciones para el desarrollo, empaquetamiento y distribución de material educativo, que permite a los OA ser reutilizables, accesibles, interoperables y durables. Con el objetivo de elevar la calidad de los mismos, se le realizan revisiones. (14)

1.5 Las revisiones de objetos de aprendizaje y recursos educativos

Las revisiones ayudan a los editores y administradores de las plataformas a decidir qué puede ser conveniente publicar y a su vez respalda a los autores y editores en sus esfuerzos por mejorar la calidad de la comunicación. Los revisores desempeñan un papel vital en los procesos de revisión, al examinar los recursos con mayor profundidad, para realizar la selección de cuáles serán publicados.

La presente investigación pretende analizar algunas alternativas que contribuyan a garantizar la calidad de los RE, proveer a los revisores de algunos instrumentos o guías que permitan cierta uniformidad en las revisiones, así como a automatizar indicadores en las revisiones para aligerar la carga para el revisor.

1.6 Instrumentos de evaluación para la revisión de la calidad

En la actualidad la creación de RE ha aumentado, lo que ha traído consigo la necesidad de diseñar criterios, instrumentos y normas que permitan garantizar la calidad de los recursos creados. A partir de la situación anterior, surgen varios modelos de evaluación de calidad de Recursos Educativos que se focalizan en diversos aspectos constituyentes de los mismos.

1.6.1 Instrumento para la evaluación de objetos de aprendizaje LORI

Esta herramienta permite evaluar los objetos de aprendizaje en función de nueve variables (15):

- Calidad de los contenidos: veracidad, exactitud, presentación equilibrada de ideas y nivel adecuado de detalle.
- Adecuación de los objetivos de aprendizaje: coherencia entre los objetivos, actividades, evaluaciones, y el perfil del alumnado.
- *Feedback* (retroalimentación) y adaptabilidad: contenido adaptativo o *feedback* dirigido en función de la respuesta de cada alumno/a y su estilo de aprendizaje.
- Motivación: capacidad de motivar y generar interés en un grupo concreto de alumno/as.
- Diseño y presentación: el diseño de la información audiovisual favorece el adecuado procesamiento de la información.
- Usabilidad: facilidad de navegación, interfaz predictiva para el usuario y calidad de los recursos de ayuda de la interfaz.
- Accesibilidad: el diseño de los controles y la presentación de la información está adaptada

para discapacitados y dispositivos móviles.

- Reusabilidad: capacidad para usarse en distintos escenarios de aprendizaje y con alumno/as de distintos bagajes.
- Cumplimiento de estándares: Adecuación a los estándares y especificaciones internacionales.

En LORI las variables se puntuarán utilizando una escala del 1 al 5. Si la variable no es relevante para la evaluación del objeto de aprendizaje o si el evaluador no se siente capacitado para juzgar una variable concreta, entonces se puede marcar NA (No Aplica). (15) El instrumento LORI puede ser adaptado según las necesidades de la organización.

1.6.2 Instrumento para la evaluación de objetos de aprendizaje de MERLOT

Multimedia Educational Resource for Learning and Online Teaching (MERLOT) quien es un repositorio de recursos libre, ofrece enlaces a materiales de formación en línea clasificados dentro de siete categorías: Artes, Negocios, Educación, Humanidades, Matemáticas, Ciencias y Tecnología. (16)

Este repositorio presenta un instrumento de evaluación propio para garantizar la calidad de la información que brinda. El cual funciona de la siguiente manera: una vez que un recurso es asignado a un evaluador aparece en MERLOT como “bajo revisión”. Esta evaluación es realizada por al menos dos personas altamente competentes quienes desde una perspectiva individual crean una “revisión compuesta”. Este material permanece catalogado como “bajo revisión” hasta que el resultado de la revisión por pares es publicado en la web de MERLOT. (16)

Los criterios de evaluación de los recursos se basan en tres dimensiones: calidad del contenido, facilidad de uso y potencial efectividad como herramienta de enseñanza. A continuación, se explicarán las dimensiones y los criterios de calidad empleados en cada una de ellas (16):

- Calidad del contenido: comprende tanto el significado educativo del contenido como su exactitud o validez. La evaluación general sobre los recursos de calidad se basa sobre preguntas como: ¿Presenta el software conceptos, modelos y habilidades válidas (correctas)?, ¿Presenta el software educacionalmente significativos conceptos, modelos y habilidades para la disciplina? Para responder a estas preguntas los evaluadores utilizan las siguientes pautas: el contenido es parte importante del currículo dentro de la disciplina. Los principales tópicos del currículo son cubiertos en algún grado en las clases introductorias dentro de la disciplina y/o “cada uno lo enseña” y/o es identificado como

una de las principales áreas por las disciplinas de organizaciones profesionales. El contenido es difícil de enseñar y aprender. El contenido es un prerrequisito para entender materiales más avanzados en la disciplina.

- Potencial efectividad como herramienta de enseñanza aprendizaje: determinar la actual efectividad de una herramienta en el proceso de enseñanza aprendizaje es lo más difícil de evaluar porque se requiere información en el momento en que el recurso está siendo utilizado por los estudiantes. Sin embargo, la efectividad de la herramienta se puede evaluar de forma potencial por los expertos definiendo según su criterio si el recurso ayudará a mejorar el proceso de enseñanza aprendizaje.
- Facilidad de uso: la cuestión básica que persigue esta dimensión es la facilidad de uso de la herramienta tanto por parte de los profesores como de los alumnos especialmente durante la primera vez que se utiliza. Los planteamientos que ayudarían a evaluar esta dimensión están relacionados al valor estético y la provisión de realimentación a las respuestas de los usuarios.

1.6.3 Resultado del estudio de los instrumentos de evaluación.

Después de realizar un análisis de los diferentes modelos de evaluación de Objetos de aprendizaje, se concluye que algunos incluyen aspectos significativos para la evaluación de los RE, pero debido a la diversidad de criterios no existe un consenso específico que pueda ser adaptado a las instituciones, ni a todos los recursos educativos por igual. Por lo antes expuesto, el autor propone la implementación de un componente de revisión que pueda desarrollar diferentes criterios de revisión adaptables a los intereses de la institución, así como a las especificidades de los contenidos a gestionar.

1.7 Sistemas similares

La versión 1.0 de Zera desarrollada por FORTES, cuenta un módulo que realiza de forma integral la comprobación de los recursos educativos, gestionados y compartidos por los docentes, mediante la selección de un procedimiento de revisión. Como apoyo se permite la gestión y configuración de los indicadores y metodologías evaluativas. Dicho modulo realiza los procesos de revisión simple, automática y por pares. La diferencia de arquitectura entre las plataformas y el cambio de modelo de desarrollo del centro, a línea de producto de software imposibilita la utilización de dicho modulo. Además en la nueva versión se desean implementar otros procesos de revisión (revisión por roles y colaborativa).

1.8 Líneas de productos de software

En la actualidad como respuesta a la necesidad de optimizar el proceso de creación de software han surgido como nuevos paradigmas de desarrollo de software las líneas de productos de software (LPS)**,** que son el ensamblaje de partes de software previamente elaboradas, inspiradas en los procesos de producción de sistemas físicos, se fundamentan en la reutilización de software y asumen la existencia de una industria de partes. Diferentes autores definen que "*se refieren a técnicas de ingeniería para crear un portafolio de sistemas de software similares, a partir de un conjunto compartido de activos de software, usando un medio común de producción*" (17)

Modelo básico de una LPS (17):

La entrada: activos de software.

- Una colección de partes de software (requisitos, diseños, componentes, casos de prueba, etc.) que se configuran y componen de una manera prescrita para producir los productos de la línea.

El Control: Modelos de Decisión y Decisiones de Productos.

- Los Modelos de Decisiones describen los aspectos variables y opcionales de los productos de la línea.
- Cada producto de la línea es definido por un conjunto de decisiones (decisiones del producto).

El proceso de producción:

- Establece los mecanismos o pasos para componer y configurar productos a partir de los activos de entrada.
- Las decisiones del producto se usan para determinar qué activos de entrada utilizar y cómo configurar los puntos de variación de esos activos.

La salida: productos de software.

- Conjunto de todos los productos que pueden o son producidos por la línea de productos.

Beneficios:

- La entrega de productos de software de una manera más rápida, económica y con una mejor calidad.

- Las LPS producen mejoras en tiempo de entrega del producto, y en su calidad.
- Reducen las tasas de defectos, los costos de ingeniería y el tamaño del portafolio de productos.

1.9 Tecnologías y Herramientas

Actualmente existe una amplia variedad de metodologías, tecnologías y herramientas que permiten la creación y el desarrollo de aplicaciones web. Las herramientas seleccionadas se encuentran dentro del marco tecnológico del centro FORTES, siguiendo así con un conjunto de políticas establecidas y definidas por la arquitectura del marco de trabajo de la plataforma Zera 2.0. Por lo anteriormente expuesto se decidió hacer uso de las siguientes herramientas, tecnologías y metodología de desarrollo de software.

1.9.1 Metodología de desarrollo de software

El desarrollo de software debe ser estructurado, planificado y controlado por una metodología que documente y controle toda la información que en este se genere. Una gran variedad de metodologías se ha desarrollado a lo largo de los años, que pueden ser agrupadas en dos grandes grupos: (18)

- Metodologías tradicionales (Pesadas): orientadas al control de los procesos, estableciendo rigurosamente las actividades a desarrollar, herramientas a utilizar y notaciones que se usarán.
- Metodologías ágiles (Ligeras): las orientadas a la interacción con el cliente y el desarrollo incremental del software, mostrando versiones parcialmente funcionales del software al cliente en determinados intervalos de tiempo, para que pueda evaluar y sugerir cambios en el producto.

El Proceso Unificado Ágil de *Scott Ambler* o *Agile Unified Process* (AUP por sus siglas en inglés) es una versión simplificada del *Rational Unified Process* (RUP). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. El AUP aplica técnicas ágiles incluyendo: (19)

- Desarrollo dirigido por pruebas (test driven development TDD en inglés).
- Modelado ágil.
- Gestión de cambios ágil.

- Refactorización de base de datos para mejorar la productividad.

Al igual que en RUP, en AUP se establecen cuatro fases que transcurren de manera consecutiva: (19)

- Inicio: el objetivo de esta fase es obtener una comprensión común cliente-equipo de desarrollo del alcance del nuevo sistema y definir una o varias arquitecturas candidatas para el mismo.
- Elaboración: el objetivo es que el equipo de desarrollo profundice en la comprensión de los requisitos del sistema y en validar la arquitectura.
- Construcción: durante la fase de construcción el sistema es desarrollado y probado al completo en el ambiente de desarrollo.
- Transición: el sistema se lleva a los entornos de preproducción donde se somete a pruebas de validación y aceptación y finalmente se despliega en los sistemas de producción.

Al no adaptarse la metodología perfectamente a las características del proyecto (equipo de desarrollo, recursos, etc.), se decide usar una variación de la metodología AUP desarrollada en la institución: AUP-UCI, la cual de las 4 fases que propone AUP (Inicio, Elaboración, Construcción, Transición) decide para el ciclo de vida de los proyectos de la UCI mantener la fase de Inicio, pero modificar el objetivo de la misma, se unifican las restantes 3 fases de AUP en una sola, a la que se nombrara, Ejecución y se agrega la fase de Cierre. Para una mayor comprensión se muestra la siguiente Tabla. (19)

Fases AUP	Fases Variación AUP-UCI	Objetivos de las fases (Variación AUP-UCI)
Inicio	Inicio	Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización

		cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta uno el proyecto.
Elaboración	Transición	En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del Proyecto Transición considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, se obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.
Construcción		
Transición		
	Cierre	En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Tabla 1: Fases variación AUP-UCI.

1.9.2 Herramienta de modelado

Las herramientas de modelado son diversas aplicaciones informáticas o programas informáticos destinados a aumentar la productividad en el desarrollo de software reduciendo el costo de los mismos en términos de tiempo y dinero. Estas herramientas usan lenguajes de modelado como

Lenguaje Unificado de Modelado (UML,*Unified Modeling Language*) para representar la lógica de la aplicación, negocio, etc. (20)

El UML es un lenguaje gráfico para la especificación, visualización, construcción y documentación de modelos orientados a objetos que representan sistemas intensivos en software. (20)

Entre las características que fundamentaron la elección de su versión 2.0 como lenguaje de modelado para esta investigación están que (20):

- Proporciona a los desarrolladores un lenguaje de modelado ampliamente aceptado y listo para usar.
- Integra las mejores prácticas del desarrollo de software.
- Permite el intercambio de modelos entre las diferentes herramientas de software.
- Es independiente del lenguaje de programación y de métodos y procesos particulares de desarrollo de software.
- Proporciona sus propios mecanismos de extensión.
- Agrupa los conceptos de orientación a objetos definiendo su significado.

Se seleccionó como herramienta de modelado *Visual Paradigm for UML 8.0 Enterprise Edition* que "*es una herramienta de modelado UML que proporciona un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. Es capaz de soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de diferentes tipos de diagramas.*" (21)

Esta herramienta está disponible en múltiples plataformas (Windows, GNU/Linux), su diseño se centra en casos de usos y se enfoca a la calidad del software. Es fácil de instalar y actualizar, tiene capacidad de ingeniería directa e inversa, soporta UML y sus versiones son compatibles entre sí. Brinda funcionalidades para la generación de código fuente a partir de diagramas de clases. Está diseñada para una amplia gama de usuarios interesados en la construcción de sistemas de software de forma fiable a través de la utilización de un enfoque orientado a objetos.

1.9.3 Lenguaje de desarrollo

Es un conjunto de reglas, notaciones, símbolos y/o caracteres que permiten a un programador

poder expresar el procesamiento de datos y sus estructuras en la computadora; puede ser usado para controlar el comportamiento de una máquina, o para definir una secuencia de instrucciones para su procesamiento. (22)

PHP 5.4.16

PHP (acrónimo de "PHP: *Hypertext Preprocessor*") es un lenguaje interpretado de alto nivel embebido en páginas HTML y ejecutado en el servidor. Al nivel más básico, PHP procesa la información de formularios, genera páginas con contenidos dinámicos, o manda y recibe cookies. Una de las características más potente y destacable de PHP es su soporte para una gran cantidad de bases de datos, pues escribir una interfaz vía web para una base de datos es una tarea simple con PHP. También soporta el uso de otros servicios que usen protocolos como IMAP, SNMP, NNTP, POP3, HTTP y derivados. También se pueden abrir sockets de red directos (raw sockets) e interactuar con otros protocolos. (23)

Las razones de utilizar este lenguaje se deben a su poder y sencillez. PHP está disponible para la mayoría de sistemas operativos existentes. Desde Unix, Linux, Microsoft Windows, MAC, entre otros. En cuanto a su costo, es un software libre, por lo que puede ser utilizado sin problemas. Su rendimiento es muy bueno y verdaderamente eficiente, pues con un servidor modesto se puede atender millones de peticiones al día, lo cual es más que suficiente para el desarrollo de la aplicación en cuestión. Con PHP se tiene acceso al código fuente, es decir si se desea agregar o modificar algo para obtener un funcionamiento de acuerdo a ciertas necesidades se puede hacer con total libertad.

HTML5

HTML (*Hypertext Markup Language*) es un lenguaje de marcado para describir la estructura de las páginas web. Brinda al autor los medios para: (24)

- Publicar documentos en línea con encabezados, texto, tablas, listas, fotos, entre otros elementos.
- Recuperar información en línea a través de enlaces de hipertexto.
- Formas de diseño para la realización de transacciones con servicios remotos, para su uso en la búsqueda de información, hacer reservas y pedidos de productos.
- Incluir videoclips, clips de sonido y otras aplicaciones directamente en sus documentos.

CCS3

CSS son las siglas de *Cascading Style Sheets*, Hojas de Estilo en Cascada que es un lenguaje

que describe la presentación de los documentos estructurados en hojas de estilo para diferentes métodos de interpretación, es decir, describe cómo se va a mostrar un documento en pantalla, por impresora, por voz (cuando la información es pronunciada a través de un dispositivo de lectura) o en dispositivos táctiles basados en Braille. (24)

CSS es una especificación desarrollada por el W3C (*World Wide Web Consortium*) para permitir la separación de los contenidos de los documentos escritos en HTML de la presentación del documento con las hojas de estilo, incluyendo elementos tales como los colores, fondos, márgenes, bordes, tipos de letra, entre otros, modificando la apariencia de una página web de una forma más sencilla, permitiendo a los desarrolladores controlar el estilo y formato de sus documentos. (24)

JavaScript 1.8.5

En términos generales, JavaScript permite mejorar la gestión cliente/servidor. Un guion de JavaScript puede tratar y gestionar localmente, en el cliente (navegador del usuario), eventos tales como: (25)

- Comprobar la validez de los campos cumplimentados en un formulario.
- Abrir y cerrar ventanas.
- Cambios dinámicos en una página (aspecto y contenidos).
- Tratamiento de cadenas de texto.
- Operaciones aritméticas.

1.9.4 Framework a utilizar

En el desarrollo de software, un *framework* es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, en base a la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, librerías y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio. (26)

Symfony2.7.9

Symfony es un *framework* diseñado para optimizar el desarrollo de las aplicaciones web de acuerdo a algunas de sus características. Primeramente, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Presenta varias herramientas y clases que permiten reducir el tiempo de desarrollo de las aplicaciones web complejas. Symfony está desarrollado completamente en PHP 5. Es compatible con la mayoría de los gestores de bases de datos, como MySQL, PostgreSQL, Oracle y Microsoft SQL Server. Se puede ejecutar tanto en plataformas Unix y Linux, como en plataformas Windows. Alguna de las características que presenta son (27):

- Sigue las mejores prácticas de patrones de diseño para la web.
- Código fácil de leer y permite un mantenimiento sencillo.
- Independiente del sistema gestor de bases de datos.
- Utiliza programación orientada a objetos.
- Fácil de instalar y configurar en plataformas como Windows y Linux.
- Posee una potente línea de comandos que facilitan la generación de código, lo cual permite ahorrar tiempo de trabajo.
- Fácil de extender, lo que permite su integración con las bibliotecas de otros fabricantes.
- Utiliza la arquitectura Modelo – Vista – Controlador (MVC) como la capa de abstracción de base de datos, el controlador frontal y las acciones.

Symfony automatiza la mayoría de elementos comunes de los proyectos web, como por ejemplo (27):

- La capa de internacionalización que incluye Symfony permite la traducción de los datos y de la interfaz, así como la adaptación local de los contenidos.

- La capa de presentación utiliza plantillas y diseños que pueden ser creados por diseñadores HTML sin ningún tipo de conocimiento del *framework*. Las ayudas incluidas permiten minimizar el código utilizado en la presentación, pues encapsulan grandes bloques de código en llamadas simples a funciones.

- Los formularios incluyen validación automatizada y relleno automático de datos, lo que asegura la obtención de datos correctos y mejora la experiencia de usuario.

- Los datos incluyen mecanismos de escape que permiten una mejor protección contra los ataques producidos por datos corruptos.

- La gestión de la caché reduce el ancho de banda utilizado y la carga del servidor.
- La autenticación y la gestión de credenciales simplifican la creación de secciones restringidas y la gestión de la seguridad de usuario.
- Las interacciones con AJAX son tan fáciles de implementar mediante las ayudas que permiten encapsular los efectos JavaScript compatibles con todos los navegadores en una única línea de código.

JQuery

jQuery es un *framework* Javascript, implementa una serie de clases (Programación Orientada a Objeto) que permite programar sin preocuparse por el navegador con que está visitando el usuario. Ofrece una infraestructura con la que se tiene mucha mayor facilidad para la creación de aplicaciones complejas del lado del cliente. Es un producto serio, estable, bien documentado y con un gran equipo de desarrolladores a cargo de la mejora y actualización del framework. (28)

Bootstrap v3.0:

Es un *framework* HTML, CSS *y JavaScript* que se puede utilizar como base para crear sitios o aplicaciones web, está diseñado pensando en ofrecer la mejor experiencia de usuario tanto a usuarios de computadoras, como a *smartphones* y *tablet* (29) A continuación se exponen sus principales ventajas (30):

- Permite ahorrar tiempo (no hay que empezar una página desde cero, sino que se puede trabajar sobre el código que aporta y empezar a desarrollar desde ahí.).
- Utiliza componentes y servicios creados por la comunidad web.
- Maneja un conjunto de buenas prácticas que perdurarán en el tiempo.
- Emplea HTML5 y CSS3.
- Herramienta sencilla y ágil para construir sitios web e interfaces. Tiene temas por defecto bastante optimizados y que se pueden modificar fácilmente.

1.9.5 Entorno de Desarrollo Integrado (IDE)

Un entorno de desarrollo integrado, es un área de programación que ha sido empaquetada como

un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Se decidió usar como IDE NetBeans (versión 8.0) porque posee soporte para PHP, es fácil de usar y es software libre.

NetBeans IDE 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 y es un producto libre. (29)

Ventajas (29):

- La plataforma Netbeans puede ser usada para desarrollar cualquier tipo de aplicación.
- Reutilización de módulos.
- Instalación y actualización simple.
- Incluye plantillas de desarrollo.
- Posee soporte para PHP.

1.9.6 Sistema Gestor de Bases de Datos 9.3

El sistema gestor de bases de datos relacionales conocido como PostgreSQL, es uno de los gestores de bases de datos de código abierto más avanzado hoy en día, ofrece el control de concurrencia multiversión, soporta casi toda la sintaxis SQL (incluyendo subconsultas, transacciones y tipos y funciones definidas por el usuario), contando también con un amplio conjunto de enlaces con lenguajes de programación (incluyendo PHP, C, C++, Java, Perl, Tcl y Python). (30)

Es un sistema de base de datos fácil de aprender. Es altamente configurable, con lo cual puede personalizarse de acuerdo al escenario donde será utilizado. Es multiplataforma (cuenta con versiones para sistemas operativos Unix-like y Windows). Es extensible pues se pueden implementar nuevos tipos de datos, funciones, operadores, y lenguajes e incorpora un sistema de recolección de estadísticas en tiempo real de transacciones. (30)

1.10 Conclusiones del capítulo

- El estudio de un conjunto de conceptos asociados a los procesos de revisión contribuyó a un mejor entendimiento de cómo se desarrollan los mismos.
- Queda evidencia de la necesidad de incorporar los procesos de revisión para contribuir a elevar la calidad de los recursos educativos.

- Se definen las herramientas y tecnologías a utilizar, teniendo en cuenta el marco de trabajo de la plataforma.

Capítulo 2: Propuesta de solución

El presente capitulo tiene como objetivo describir las características fundamentales del componente de revisión desarrollado para Zera 2.0, definiendo en cada una de sus fases un grupo de artefactos que brindan una compresión clara del sistema a desarrollar. En este capítulo se presentarán los siguientes artefactos: Modelo del dominio, la especificación de requisitos funcionales y no funcionales que tendrá la propuesta de solución. Se describen los artefactos generados durante el diseño de la solución y se definen los patrones arquitectónicos y de diseño a utilizar.

2.1 Modelo del dominio

Un modelo del dominio se utiliza con frecuencia como fuente de inspiración para el diseño de los elementos del software, y será una entrada necesaria para varios de los siguientes artefactos que se verán. El modelo del dominio muestra (a los modeladores) clases conceptuales significativas en un dominio del problema; es el artefacto más importante que se crea durante el análisis orientado a objetos. Un modelo del dominio es una representación de las clases conceptuales del mundo real, no de componentes software. (20)

2.1.1 Conceptos del dominio

- **Usuario:** persona que interactúa con el componente.
- **Administrador:** tipo de usuario encargado de gestionar criterios de evaluación, gestionar los roles, gestionar equipos y asignar los RE a revisión.
- **Revisor:** tipo de usuario encargado de revisar los RE que se le asignan a su equipo.
- **Equipo de revisión:** agrupación de revisores.
- **Rol:** categoría que define las funciones que desempeña un revisor dentro de un equipo.
- **Revisión:** revisión realizada a un elemento para medir su calidad.
- **Criterio de evaluación:** guía o conjunto de indicadores utilizados para realizar una revisión.
- **Indicador de evaluación:** elemento que mide un aspecto específico del objeto a revisar.
- **RE:** recursos digitales con un propósito educativo.
- **RE sin asignar a revisión:** RE que no se ha revisado ni asignado a revisión.

- **RE asignado a revisión:** RE que está asignado a revisión.

2.1.2 Diagrama de modelo del dominio

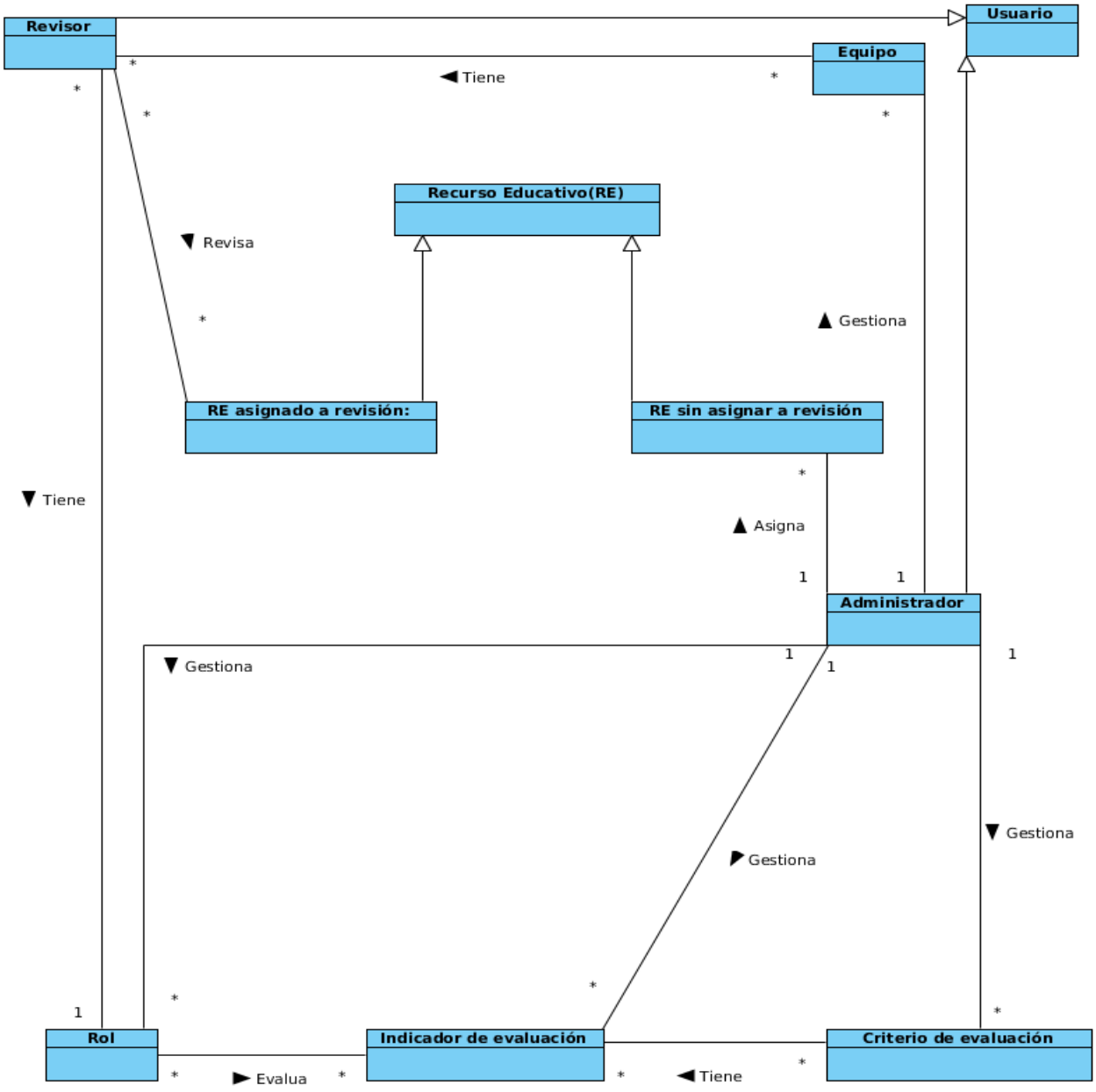


Figura 1: modelo de dominio proceso de revisión.

2.2 Propuesta de solución

Con el objetivo de contribuir a la calidad de los RE gestionados en Zera2.0 la presente investigación se propone integrar a la plataforma un componente que desarrolle varios procesos de revisión apoyado de criterios de evaluación, que guíen la revisión, permitiendo que tanto las revisiones como los criterios puedan ser configurados por el administrador del sistema.

Los procesos de revisión definidos para esta investigación son:

- El proceso de revisión simple: interviene un solo revisor, que debe ser especialista en contenido o al menos tener dominio del tema, con el objetivo de garantizar un nivel de calidad determinado para el recurso.

- El proceso de revisión por pares: es ejecutado por dos o más revisores que deben ser expertos en el tema, los cuales harán de forma individual una revisión similar a la simple; de aquí saldrá la opinión de estos revisores con las cuales el revisor general deberá evaluar para dar la conclusión de la revisión (Aceptada o Denegada). El Proceso de Revisión por Pares para esta investigación puede ser ejecutado de tres formas: Abierta, Simple Ciega, Doble Ciega.

- El proceso de revisión por roles: debe ser ejecutado o llevado a cabo por equipos conformados por roles, los cuales centrarán la revisión en aspectos específicos. El administrador es la persona encargada de confeccionar los equipos de revisión, asignándole el rol a cada revisor. El proceso descrito se lleva a cabo mediante un equipo de trabajo, cuyo resultado final será dado cuando se obtenga el criterio de cada revisor. De igual forma se hace uso de los criterios de evaluación de los recursos. El Proceso de Revisión por Roles para esta investigación puede ser ejecutado de tres formas: Abierta, Simple Ciega, Doble Ciega.

- El proceso de revisión colaborativa: permite a los usuarios comentar los recursos con el fin de conocer el criterio de los mismos. El objetivo consiste en que una vez publicado el recurso en la plataforma, los destinatarios finales del mismo detecten irregularidades y deficiencias en el mismo, participando de forma directa en el proceso de revisión.

2.3 Requerimientos del sistema

En ingeniería del software y el desarrollo de sistemas, un requerimiento es una necesidad documentada sobre el contenido, forma o funcionalidad de un producto o servicio. Los requerimientos son declaraciones que identifican atributos, capacidades, características y/o cualidades que necesita cumplir un sistema (o un sistema de software) para que tenga valor y utilidad para el usuario. En otras palabras, los requerimientos muestran qué elementos y funciones son necesarias para un proyecto. (32)

2.3.1 Requisitos funcionales

RF1: Gestionar indicador de evaluación.

RF1.1: Crear indicador de evaluación.

RF1.2: Modificar indicador de evaluación.

RF1.3: Listar indicadores de evaluación.

RF1.4: Visualizar indicador de evaluación.

RF1.5: Eliminar indicador de evaluación.

RF2: Gestionar criterio de evaluación.

RF2.1: Crear criterio de evaluación.

RF2.2: Modificar criterio de evaluación.

RF2.3: Listar criterios de evaluación.

RF2.4: Visualizar criterio de evaluación.

RF2.5: Eliminar criterio de evaluación.

RF3: Gestionar equipo de revisión.

RF3.1: Crear equipo de revisión.

RF3.2: Modificar equipo de revisión.

RF3.3: Listar equipos de revisión.

RF3.4: Visualizar equipo de revisión.

RF3.5: Eliminar equipo de revisión.

RF4: Gestionar rol.

RF4.1: Crear rol.

RF4.2: Modificar rol.

RF4.3: Listar roles.

RF4.4: Visualizar rol.

RF4.5: Eliminar rol.

RF5: Enviar RE a revisión.

RF6: Listar RE asignados.

RF7: Realizar revisión.

RF8: Gestionar resultado de revisión.

RF8.1: Listar resultados de revisión.

RF8.2: Visualizar resultado de revisión.

RF9: Comentar RE.

RF10: Gestionar comentario.

RF10.1: Listar comentarios.

RF10.2: Revisar comentario.

RF11: Gestionar usuarios problemáticos.

RF11.1: Configurar variables para determinar usuarios problemáticos.

RF11.2: Listar usuarios problemáticos.

RF12: Gestionar usuarios destacados.

RF12.1: Configurar variables para determinar usuarios destacados.

RF12.2: Listar usuarios destacados.

2.3.2 Requisitos no funcionales:

Usabilidad

RNF1: el componente debe poseer una interfaz fácil de utilizar para cualquier tipo de usuario con conocimientos básicos de computación en el manejo de ordenadores.
RNF2: el componente debe revisar todos los campos de datos para revisar que no se inserten datos erróneos.

RNF3: el componente debe utilizar íconos y textos de enlace sugerentes para lograr que el usuario encuentre fácilmente lo que busca.

Software

RNF4: el componente debe compilar sobre cualquier navegador, siendo compatible con: Mozilla Firefox 10.0, Opera 10.00 y Chrome 20.0 en adelante, para poder acceder a las opciones que brinda el sistema.

Seguridad

RNF5: el acceso a la información debe estar restringido al rol que la utiliza.

RNF6: en los formularios evitar las inyecciones de código JavaScript.

Hardware

RNF7: para un buen funcionamiento, la PC cliente debe contar con las siguientes características:

- ✓ Procesador Intel Celeron.
- ✓ Memoria RAM de 512MB.

RNF8: para un buen funcionamiento, el servidor donde se ejecute el sistema debe contar con las siguientes características:

- ✓ Procesador Intel Dual Core 2.8 GHz.
- ✓ Memoria RAM de 8GB.

2.4 Historias de usuario

Las historias de usuario son la técnica utilizadas para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento las historias de usuario pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas. (33)

Número:1.1	**Nombre del requisito:** Crear indicador de evaluación.
Programador: Pedro Pablo Suarez Govea	**Iteración Asignada:** 1ra
Prioridad: Alta.	**Tiempo Estimado:**4h
Riesgo en Desarrollo: Medio	**Tiempo Real:**
Descripción: **El sistema debe permitir al administrador crear indicadores de evaluación con los**	

siguientes datos:

- Nombre: nombre del indicador. Admite caracteres alfanuméricos.
- Descripción: descripción del indicador. Admite cualquier carácter.
- Roles asociados: roles que interviene en la revisión del indicador.
- Rango: Rango numérico en el que se da la evaluación cuantitativa del indicador. Campo de selección.

El sistema debe mostrar un mensaje de error en caso que los campos "nombre", "rango" y "descripción" no sean llenados o el campo contenga caracteres no válidos.

Observaciones:

Prototipo elemental de interfaz gráfica de usuario:

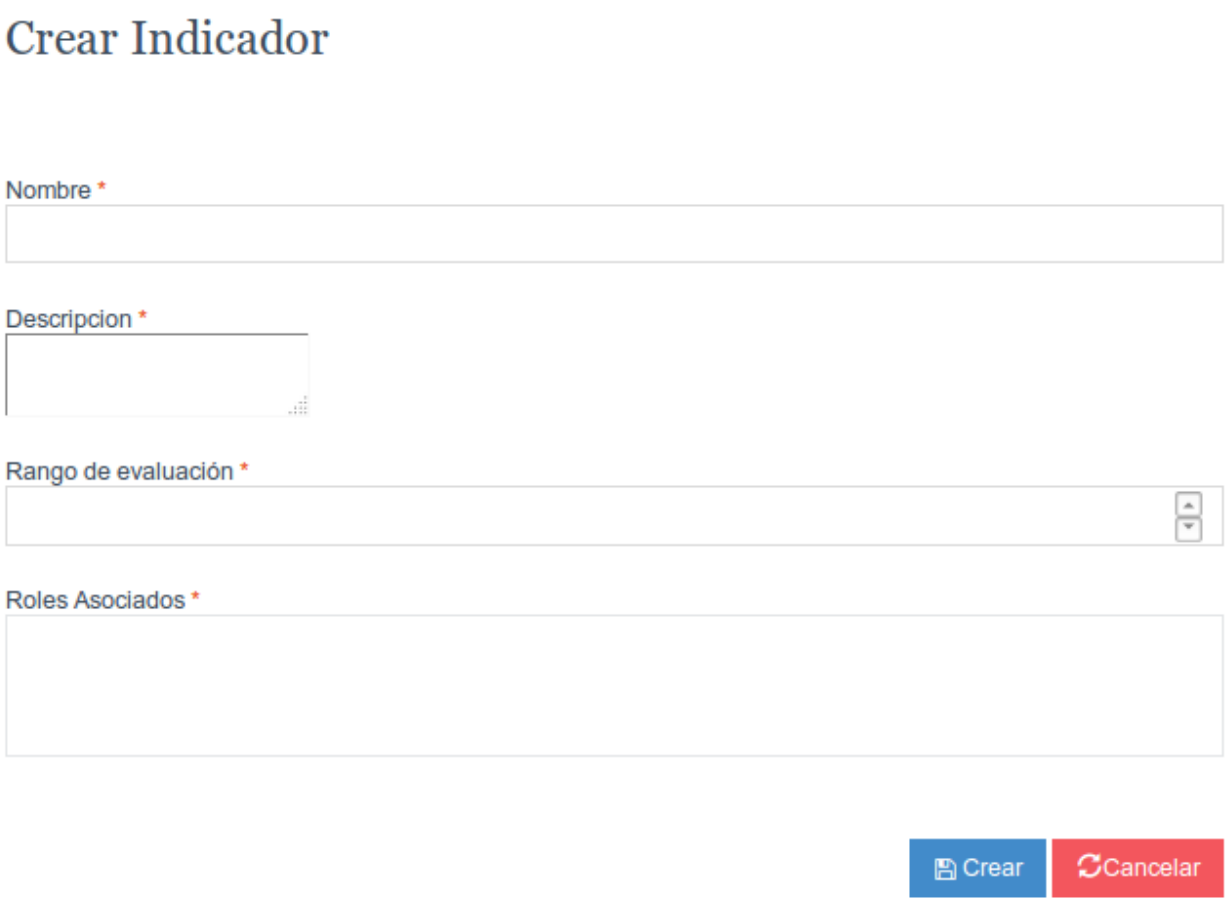


Tabla 2: Historia de usuario crear indicador de evaluación.

Número: 1.2	**Nombre del requisito:** Modificar indicador de evaluación.
Programador: Pedro Pablo Suarez Govea	**Iteración Asignada:** 1ra
Prioridad: Alta.	**Tiempo Estimado:**4h
Riesgo en Desarrollo: Medio	**Tiempo Real:**

Descripción:

El sistema debe permitir al administrador modificar indicadores de evaluación, con los siguientes datos:

- Nombre: nombre del indicador. Admite caracteres alfanuméricos.
- Descripción: descripción del indicador. Admite cualquier carácter.
- Roles asociados: roles que interviene en la revisión del indicador. Campo de selección.
- Rango: Rango numérico en el que se da la evaluación cuantitativa del indicador. Campo de selección.

El sistema debe mostrar un mensaje de error en caso que los campos "nombre" y "descripción" no sean llenados o el campo "nombre" contenga caracteres no válidos.

Observaciones:

Prototipo elemental de interfaz gráfica de usuario:

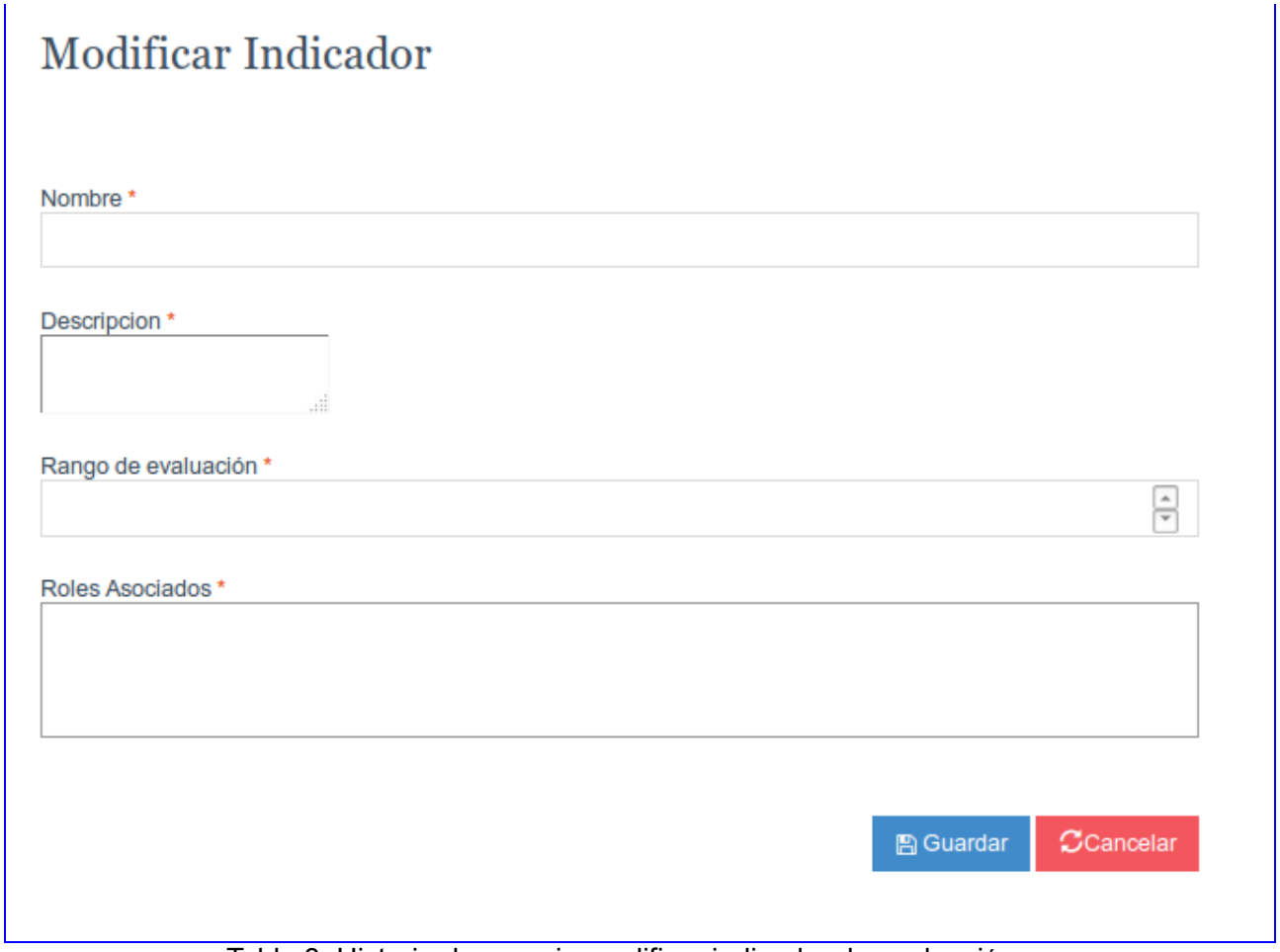


Tabla 3: Historia de usuario modificar indicador de evaluación.

Número:1.3	**Nombre del requisito:** Listar indicadores de evaluación.
Programador: Pedro Pablo Suarez Govea	**Iteración Asignada:** 1ra
Prioridad: Alta.	**Tiempo Estimado:**3h
Riesgo en Desarrollo: Medio	**Tiempo Real:**

Descripción:

El sistema debe permitir al administrador listar los indicadores de evaluación, especificando los siguientes datos:

- Nombre: nombre del indicador.
- Rango: Rango numérico en el que se da la evaluación cuantitativa del indicador.

Observaciones:
Prototipo elemental de interfaz gráfica de usuario: 

Tabla 4: Historia de usuario listar indicadores de evaluación.

2.5 Diseño

El diseño es definido en [IEEE610.12 - 90] como "*el proceso de definir la arquitectura, los componentes, interfaces, y las otras características de un sistema o componente*". Visto como un proceso, el diseño de software es la actividad del ciclo de vida de ingeniería de software en la que los requerimientos de software son analizados para causar una descripción de la estructura interna del software que servirá como base para su construcción. Más precisamente, un diseño de software (el resultado) debe describir la arquitectura de software, es decir, cómo el software está en estado de descomposición y organizado en los componentes y las interfaces entre esos componentes. También debe describir los componentes en un nivel de detalle que permita su construcción. (33)

2.5.1 Arquitectura y patrones

La arquitectura del software manifiesta las decisiones tempranas del diseño del software, que influirán en el desarrollo del sistema, su distribución y futura evolución. En términos generales, la arquitectura define la estructura que deberá tener el sistema para poder cumplir con los requerimientos del cliente. Por su parte, los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Un patrón de diseño resulta ser una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reutilizable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias. (34)

Las aplicaciones web pertenecen generalmente a la familia de arquitecturas denominada Llamada y Retorno: "*Esta familia de estilos enfatiza la modificabilidad y la escalabilidad. Dichos estilos son los más implementados en sistemas de gran escala. Miembros de la familia son las arquitecturas de programa principal y subrutina, los sistemas basados en llamadas a procedimientos remotos, los sistemas orientados a objeto y los sistemas jerárquicos en capas*" (35), un ejemplo de los sistemas jerárquicos en capas, es el Modelo Vista Controlador.

2.5.2 Arquitectura Modelo Vista Controlador (MVC)

El *framework* Symfony está basado en el patrón arquitectónico Modelo Vista Controlador (MVC). Este patrón separa en tres niveles las funcionalidades de una aplicación con el objetivo de aumentar la usabilidad de las mismas. Estos niveles son:

- Modelo: administra y maneja todo lo relacionado con los datos del sistema, da respuesta a peticiones de información sobre el estado de la aplicación (normalmente desde la vista), y responde con instrucciones de cambio de estado (usualmente desde el controlador) a la vista.

- Vista: gestiona lo relacionado con mostrar la información al usuario. La vista está representada por ficheros escritos en PHP que se encargan de construir la página HTML con la que interactúa el usuario.

- Controlador: interpreta los eventos que son lanzados por la entrada estándar del usuario (normalmente mouse y teclado), informando de los mismos al modelo y/o la vista para que se ejecuten los cambios apropiadamente. (36)

2.5.3 Patrones de diseño

Teniendo en cuenta las necesidades del sistema y el patrón de arquitectura previamente elegido, se decide emplear para estructurar el diseño del sistema Patrones Generales de Software para Asignar Responsabilidades, más conocidos como patrones GRASP (General *Responsibility Asignment Software Patterns,* por sus siglas en inglés).

Dentro de los patrones GRASP utilizados para el diseño del sistema y que incluye por defecto en su arquitectura el *framework* de desarrollo seleccionado se destacan los siguientes:

- **Experto:** la responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos). Una clase, contiene toda la información necesaria para realizar la labor que tiene encomendada. (37).Se manifiesta en las clases contenidas en el directorio entidad (Entity), las cuales son expertas en la información que modelan, tales

como rolEntity y indicadorEntity.

- **Creador:** permite crear objetos de una clase determinada. Es utilizado en la mayoría de las clases controladoras para crear instancias de formularios y entidades (37).Se manifiesta en revisionController y defaultController.
- **Controlador:** consiste en asignar la responsabilidad de controlar el flujo de eventos del sistema a clases específicas. Esto facilita la centralización de actividades (validaciones, seguridad, etc.). El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión. (37) Este patrón se evidencia en todos los elementos contenidos en el directorio Controller.
- **Alta cohesión:** cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable. (37)Este patrón se evidencia en las clases controladoras, en las se definieron una serie de funcionalidades que se relacionan entre sí, de modo que cada una de estas clases solo contenga los métodos correspondientes a su área funcional.
- **Bajo Acoplamiento:** debe haber pocas dependencias entre las clases. (37)

2.5.4 Diagramas de clases del diseño

Un diagrama de clases sirve para visualizar las relaciones entre las clases que involucran el sistema. Los diagramas de clases representan un conjunto de elementos del modelo que son estáticos, como las clases, sus contenidos y las relaciones que se establecen entre ellos. (38)

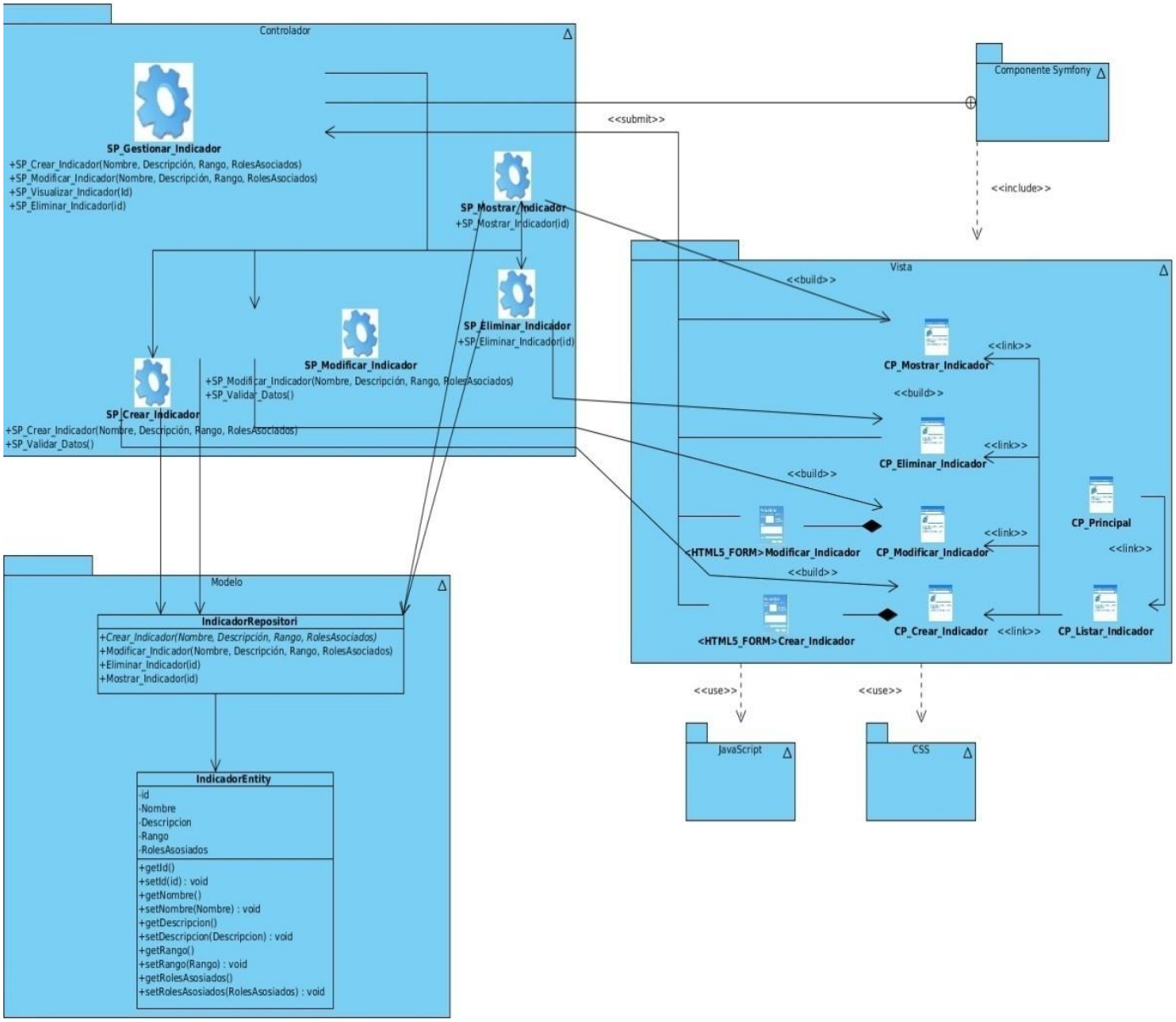


Figura 2: Diagrama de clase de diseño Gestionar Indicador.

2.5.5 Diagramas de secuencia del diseño

Los diagramas de secuencia muestran las interacciones entre objetos mediante transferencia de mensajes entre objetos o subsistemas. El nombre del mensaje debería indicar una operación del objeto que recibe la invocación o de una interfaz que el objeto proporciona (38)

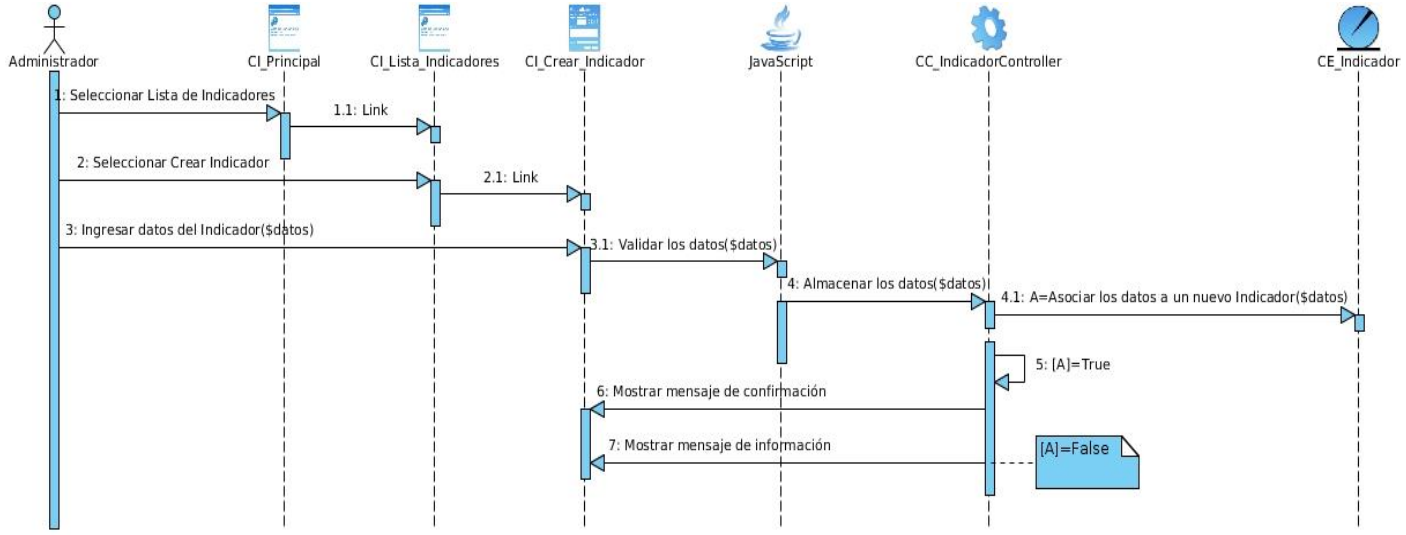


Figura 3: Diagrama de secuencia del diseño Crear Indicador.

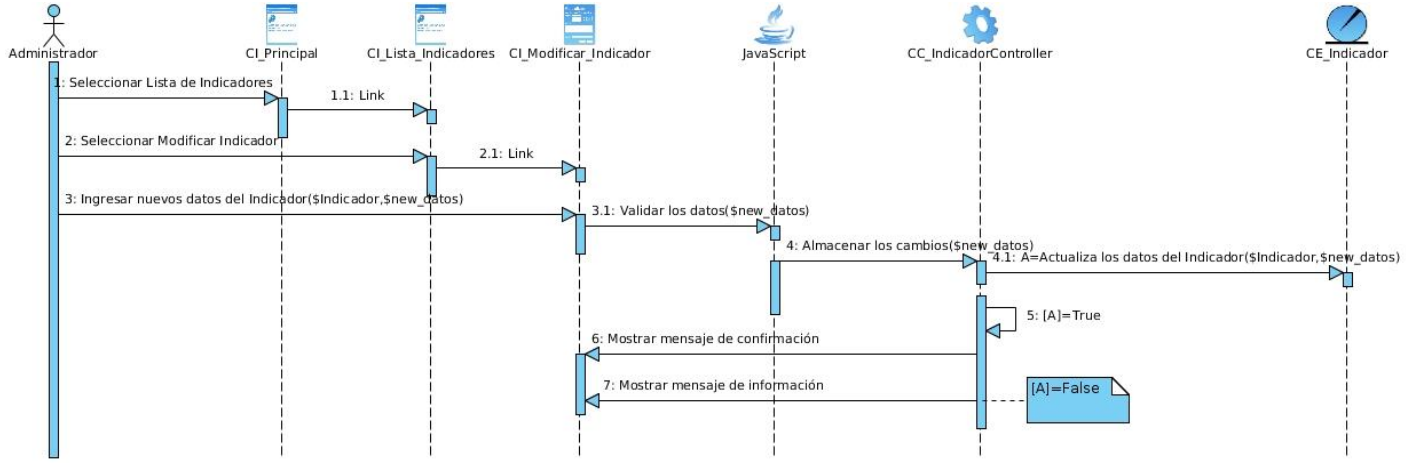


Figura 4: Diagrama de secuencia del diseño Modificar Indicador.

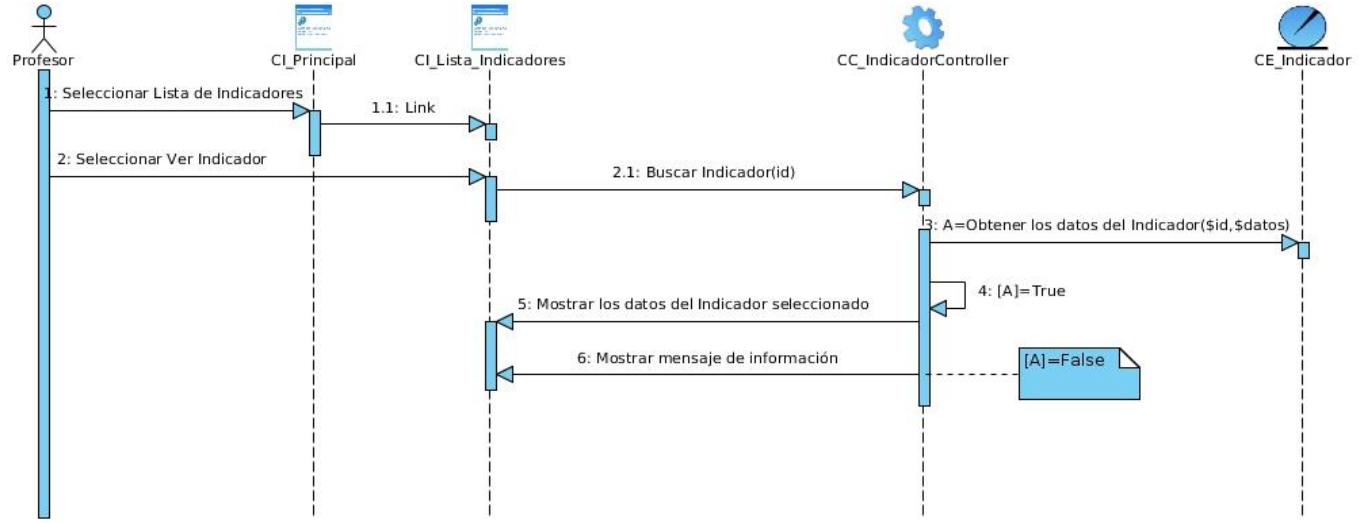


Figura 5: Diagrama de secuencia del diseño Mostrar Indicador.

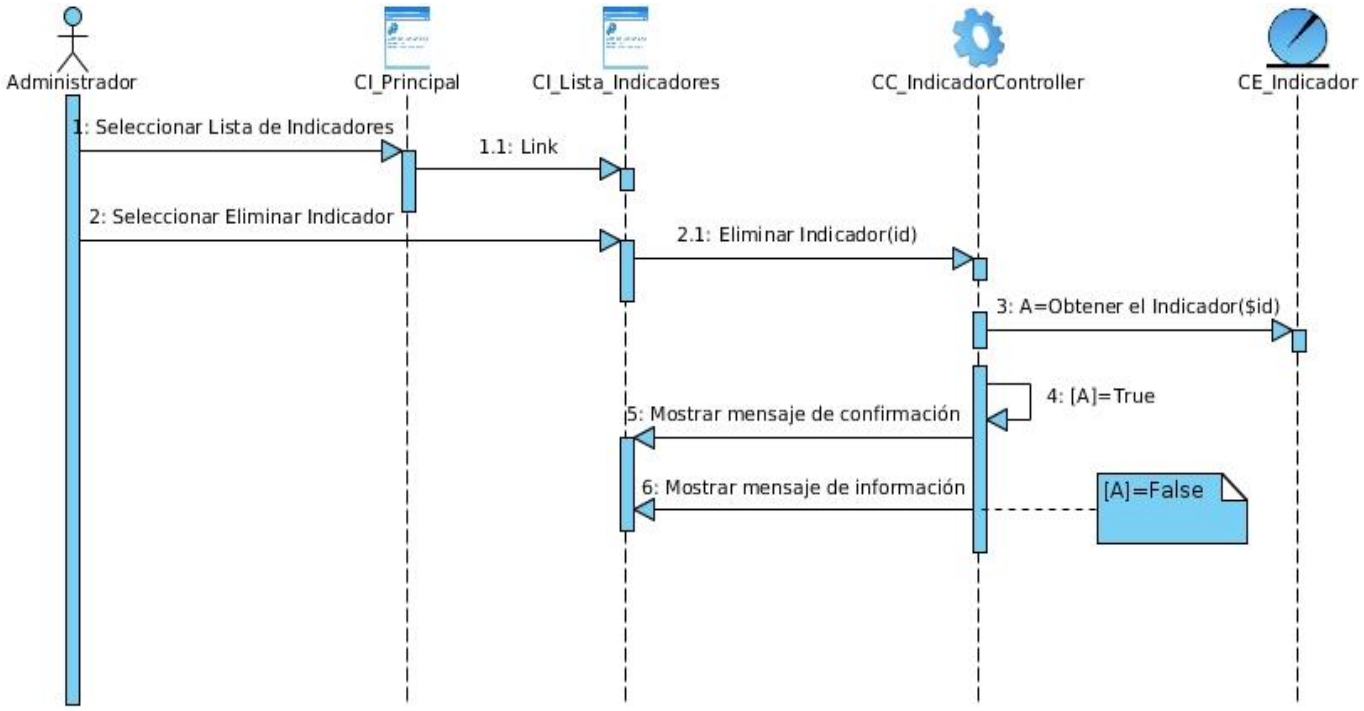


Figura 6: Diagrama de secuencia del diseño Eliminar Indicador.

2.6 Conclusiones del capítulo

- De la propuesta de solución planteada se detectaron un total de 32 requisitos funcionales y 8 no funcionales reflejados todos en 32 historias de usuario.
- Se decidió implementar un componente revisión para la plataforma Zera2.0 que desarrolle la revisión simple, por pares, por roles y colaborativa; apoyada en criterios de evaluación configurables.
- La selección de los patrones arquitectónicos y de diseño aporta sencillez al proceso de desarrollo y permite la obtención de una herramienta robusta.

Capítulo 3: Implementación y prueba

En el capítulo se realizará la implementación y las pruebas al software para dar cumplimiento a los estándares de calidad. Se crea el diagrama de componentes del sistema y se definen los estándares de codificación a usar. Son definidos los niveles y técnicas de pruebas que serán aplicadas para evaluar la calidad del sistema. Se diseñan posteriormente casos de prueba para comprobar dicha calidad.

3.1 Diagrama de componentes

Un diagrama de componente es, como su nombre lo indica, un esquema o diagrama que muestra las interacciones y relaciones de los componentes de un modelo. Entendiéndose como componente a una clase de uso específico, que puede ser implementada desde un entorno de desarrollo, ya sea de código binario, fuente o ejecutable; dichos componentes poseen tipo, que indican si pueden ser útiles en tiempo de compilación, enlace y ejecución. (20)

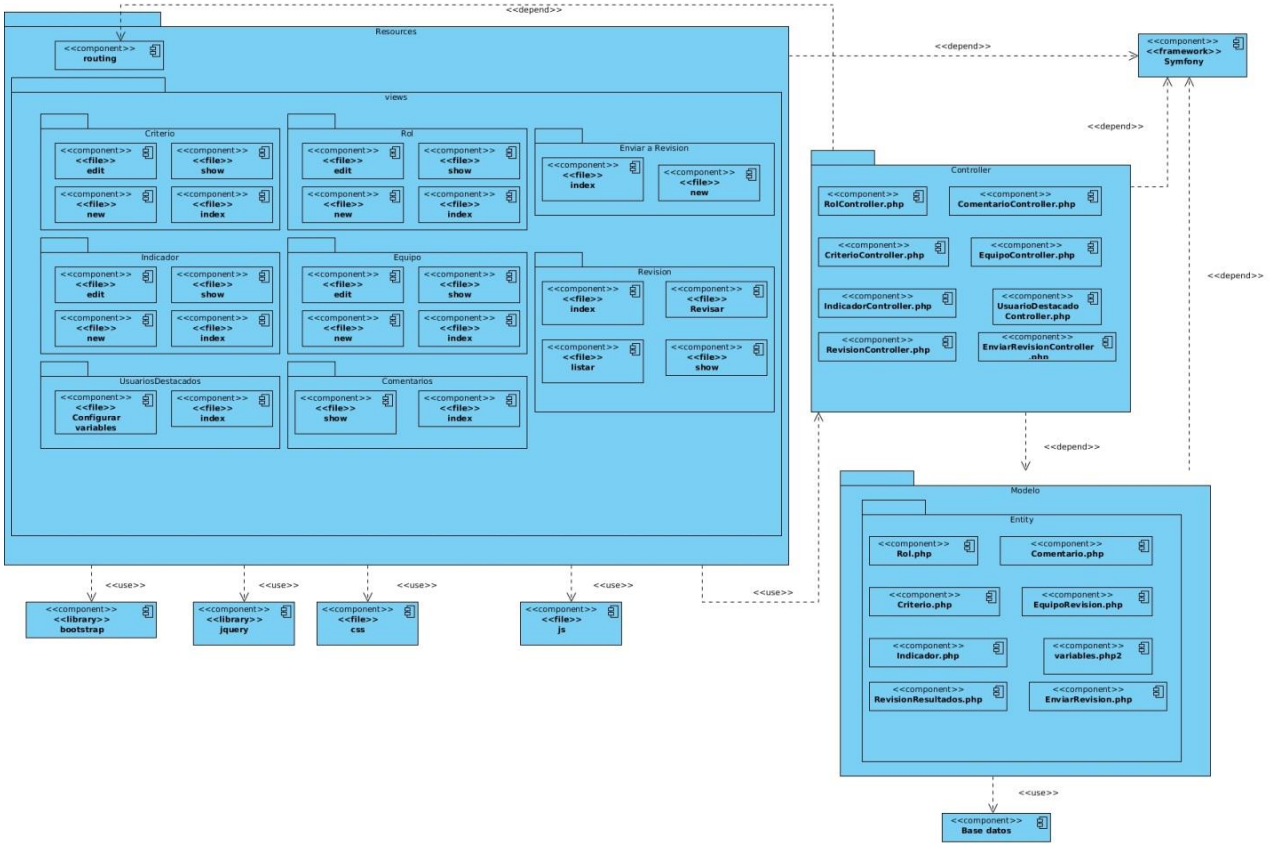


Figura 7: diagrama de componente.

3.2 Diagrama de despliegue

El diagrama de despliegue es un modelo de objetos que describe la distribución física del

sistema. Se utiliza como entrada principal en las actividades de diseño e implementación, debido a que la distribución del sistema tiene una influencia principal en su diseño. (38)

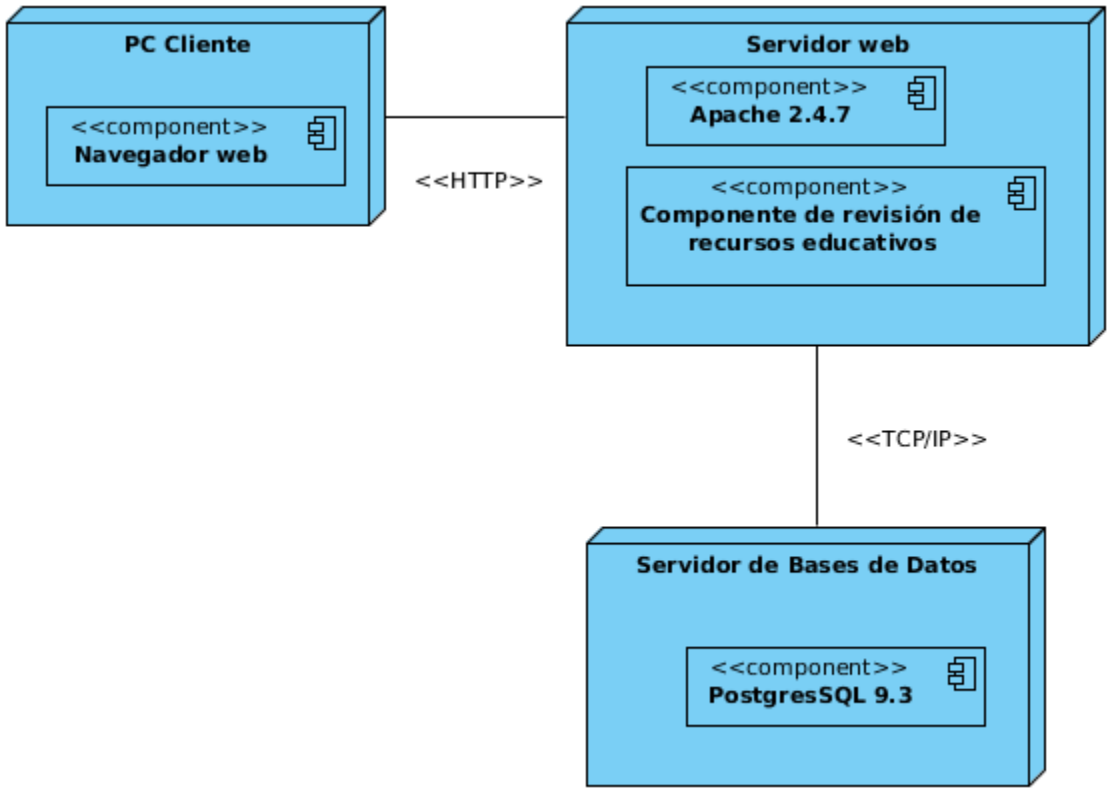


Figura 8: Diagrama de despliegue:

3.3 Estándares de codificación

Un estándar de codificación es un conjunto de reglas que se utilizan para escribir archivos de código fuente con el objetivo de lograr estructuras de código mucho más comprensibles e identificables. Cuando se construye código en Symfony2, se debe seguir su estándar de codificación (39)

Convención de Nombres

- Utilizar *camelCase* y no guiones bajos para variables, funciones y nombres de métodos.
- Utilizar guiones bajos para definir opciones, argumentos y nombres de parámetros.
- Utilizar los *namespace* para todas las clases.
- Utilizar *Symfony* como el *namespace* de primer nivel.
- Añadir como sufijo Interface a las interfaces.

- Utilizar caracteres alfanuméricos y guiones bajos para nombres de archivos.

3.4 Pruebas de software

Las pruebas de software consisten en la dinámica de la verificación del comportamiento de un programa en un conjunto finito de casos de prueba, debidamente seleccionados de por lo general infinitas ejecuciones de dominio, contra la del comportamiento esperado. Son una serie de actividades que se realizan con el propósito de encontrar los posibles fallos de implementación, calidad o usabilidad de un programa u ordenador, probando el comportamiento del mismo.

2.4.1 Niveles de prueba

Cuando se le van a aplicar pruebas a un software, se tienen en cuenta una serie de objetivos en diferentes escenarios y niveles de trabajo, debido a que las pruebas son agrupadas por niveles que se encuentran en distintas etapas del proceso de desarrollo.

- **Pruebas unitarias:** la prueba de unidad centra el proceso de verificación en la menor unidad del diseño de software, el módulo. Se prueba la interfaz del módulo para asegurar que la información fluye de forma adecuada hacia y desde la unidad de programa que está siendo probada. Se prueban las condiciones límite para asegurar que el módulo funciona correctamente en los límites establecidos como restricciones de procesamiento. Se ejercitan todos los caminos independientes de la estructura de control con el fin de asegurar que todas las sentencias del módulo se ejecutan por lo menos una vez. Y finalmente se prueban todos los caminos de manejo de errores. Estas pruebas tienen como objetivo separar cada parte del programa y mostrar que tienen un correcto funcionamiento de manera independiente. (40)

- **Pruebas de aceptación**: las pruebas de aceptación son llevadas a cabo para permitir que el cliente valide todos los requisitos. Son realizadas por el usuario final en lugar del responsable del desarrollo del sistema. Una prueba de aceptación puede ir desde la realización de casos de prueba hasta la ejecución de una serie de pruebas bien panificadas. De hecho, la prueba de aceptación puede tener lugar a lo largo de semanas o meses, descubriendo así errores acumulados que pueden ir degradando el sistema. (41)

3.4.2 Técnicas de prueba

Las técnicas de prueba facilitan una guía sistemática para diseñar pruebas que comprueben la lógica interna de los componentes de software y verifiquen los dominios de entrada y salida del programa para descubrir errores en la funcionalidad, el comportamiento y el rendimiento. (41)

1. **Pruebas de caja blanca**

   Las pruebas de caja blanca son un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. Haciendo uso de esta técnica se puede lograr que las pruebas diseñadas garanticen que se ejecuten:

   - Todos los caminos independientes de cada módulo por lo menos una vez.
   - Todas las decisiones lógicas en sus vertientes verdadera y falsa.
   - Todos los bucles en sus límites y con sus límites operacionales.
   - Las estructuras internas de datos para asegurar su validez. (41)

2. **Pruebas de caja negra**

   Las pruebas de caja negra se centran en los requisitos funcionales del software. Permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Con las pruebas de caja negra, se pretende demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta. Estas pruebas no pretenden inmiscuirse en el funcionamiento del código, es decir, lo que se comprueba es lo que hace, y no cómo lo hace. (41)

3.4.3 Casos de prueba

Consiste en el diseño de pruebas que tengan la mayor probabilidad de encontrar el mayor número de errores con la mínima cantidad de esfuerzo y de tiempo. Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.Los casos de prueba demuestran que (42):

- Las funciones del software son operativas.
- La entrada se acepta de forma adecuada.

- Se produce una salida correcta.
- La integridad de la información externa se mantiene.

A continuación, se presenta el caso de prueba Gestionar Rol. Los restantes casos de prueba se encuentran en los Anexos.

CP Gestionar Roles					
Descripción general El proceso inicia cuando un administrador del sistema decide gestionar los roles del mismo. Para ello el actor selecciona la opción Roles del menú. El sistema muestra una lista con los Roles, brinda la opción de crear un nuevo Rol. Si así lo desea el actor, el sistema muestra un formulario con los campos correspondientes a un Rol. Permite editar los datos de un Rol previamente creado y además permite mostrar todos los datos de un Rol. En caso de que desee eliminar un Rol, el actor deberá seleccionar la opción de eliminar del mostrar Rol. **Condiciones de ejecución** El administrador del sistema debe estar autenticado.					
SC1. Listar Roles					
Escenario	**Descripción**	**Nombre**	**Descripción**	**Respuesta del sistema**	**Flujo Central**
EC1.1	Selecciona la opción “Roles” del menú.			Muestra una lista con los datos correspondientes a los roles: • Nombre Muestra las siguientes opciones para cada Rol de la lista: • Mostrar. Ver	Menú /Roles

				SC3: ‘‘Mostrar Rol’’ • Modificar. Ver SC4: ‘‘Modificar Rol’’ Brinda la opción de crear un nuevo Rol. • Crear Rol. Ver SC2: ‘‘Crear Rol’’	
SC2 Crear Rol					
Escenario	**Descripción**	**Nombre**	**Descripción**	**Respuesta del sistema**	**Flujo Central**
EC2.1	Selecciona la opción ‘‘Crear Rol’’ de menú/Roles.			Se muestra un formulario con los siguientes campos: • Nombre • Descripción Y la opción: • Crear • Cancelar. dirige a listar roles Ver SC1: ‘‘Listar Roles’’.	Menú /Roles/ Crear Rol

EC2.2	Introduce los datos y selecciona “Enviar”.	V	V	Valida los datos. Registra un nuevo rol en el sistema. Redirige a: Mostrar Rol. Ver SC3: “Mostrar Rol”.	Menú/Roles/Crear Rol/ Crear
EC2.3	Existen campos vacíos.	I	V	Muestra un indicador sobre el campo vacío. Regresa al EC2.1	Menú/Roles/Crear Rol/ Crear
		V	I		
SC3 Mostrar Rol					
Escenario	**Descripción**	**Nombre**	**Descripción**	**Respuesta del sistema**	**Flujo Central**
EC3.1	Selecciona la opción “Mostrar” de uno de los miembros de la lista de roles menú/Roles			Abre una ventana con los datos correspondientes a la solicitud: • Nombre • Descripción Muestra las siguientes opciones:	Menú /Roles/ Mostrar

				• Listar Ver SC1: ‘‘Listar Roles’’ • Modificar. Ver SC4: ‘‘Modificar Rol’’ • Eliminar	
EC3.2	Selecciona la opción: ‘‘Eliminar’’.			Muestra un mensaje informando: “Desea Eliminar Permanentemente el Rol”. Muestra las siguientes opciones: • Aceptar • Cancelar	Menú /Roles/ Mostrar/Eliminar
EC3.2.1	Selecciona la opción: ‘‘Cancelar’’			Regresa al EC3.1	Menú /Roles/ Mostrar/Eliminar/ Cancelar
EC3.2.1	Selecciona la opción: ‘‘Aceptar’’			Elimina rol del sistema. Redirige a: Listar Rol. Ver SC1: ‘‘Listar Roles’’.	Menú /Roles/ Mostrar/Eliminar/ Aceptar

SC4. Modificar Rol					
Escenario	**Descripción**	**Nombre**	**Descripción**	**Respuesta del sistema**	**Flujo Central**
EC4.1	Selecciona la opción "Modificar" de uno de los miembros de la lista de roles menú/Roles			Abre una ventana con los datos correspondientes al Rol permitiendo modificar: • Nombre • Descripción Y las opciones: • Guardar • Cancelar. dirige a listar roles Ver SC1: "Listar Roles"	Menú /Roles/ Modificar
EC4.2.1	Modifica los datos y selecciona la opción "Guardar".	V	V	Valida los cambios realizados. Actualiza el rol en el sistema. Dirige a listar roles Ver SC1: "Listar Roles"	Menú /Roles/ Modificar / Guardar
EC4.2.1	Existen campos vacíos.	V	I	Muestra un indicador sobre el campo vacío. Regresa al 4.1	Menú /Roles/ Modificar / Guardar
		I	V		

Tabla 5: caso de prueba gestionar rol.

3.4.4 Resultados obtenidos en las pruebas

En las pruebas realizadas al sistema mediante el método de caja negra, se pudo comprobar el cumplimiento de los requisitos funcionales y no funcionales del software obtenido.

A continuación, se presenta el resultado de las pruebas realizadas:

NO.	Casos de prueba	No Conformidades			Total
		Validación	Funcionalidad	Interfaz de usuario	
1	Gestionar Rol	1	-	-	1
2	Gestionar Indicador	2	1	-	3
3	Gestionar Comentario	2	-	1	3
4	Gestionar Equipo	-	3	-	3
5	Enviar recurso a revisión	-	1	2	3
6	Listar recursos asignados	1	3	4	8
7	Realizar revisión	2	-	3	5
8	Gestionar resultado de revisión	4	1	3	8
9	Comentar recurso	-	-	1	1
10	Gestionar denuncia	1	3	-	4
12	Gestionar usuarios problemáticos	-	-	2	2
13	Gestionar usuarios destacados	-	-	2	2
	Total	13	11	19	43

Tabla 6: resultados de las pruebas realizadas.

3.5 Conclusiones del capítulo

- Los diagramas de componente obtenidos en el flujo de trabajo de implementación permitieron concebir el modelo de desarrollo describiendo los elementos del modelo de

diseño se implementan en términos de componentes.

- Las pruebas realizadas al sistema fueron detectadas 43 no conformidades las cuales fueron corregidas para un correcto funcionamiento de la solución.

Conclusiones

- La investigación de los procesos de revisión evidenció la existencia de diferentes criterios para la evaluación de la calidad del contenido educativo, que incluyen aspectos significativos a tener en cuenta en el análisis de estos. Debido a la diversidad existente, en la plataforma Zera2, los criterios de evaluación pueden configurarse en dependencia del tipo de recurso a revisar.
- La confección de los principales artefactos correspondientes a los flujos de trabajo de la metodología de desarrollo de software AUP-UCI sirvieron de apoyo en la implementación de las funcionalidades definidas en la investigación.
- La incorporación del componente de revisión a Zera2, permite contribuir a elevar la calidad del contenido educativo y a la uniformidad de las revisiones en la plataforma.
- Los métodos de validación definidos permitieron identificar y erradicar los errores del componente, evitando insatisfacciones por parte de los usuarios, demostrándose el cumplimiento de los objetivos trazados en la investigación.

Recomendaciones

- Enviar notificación a los usuarios del resultado de la revisión de su recurso y su inclusión a la lista de usuarios destacado o problemático.
- Brindar en la revisión colaborativa la posibilidad de denunciar los recursos.

Bibliografía

1. ***Diccionario Manual de la Lengua Española Vox.*** **s.l. : Larousse Editorial, 2007.**

2. K Dictionaries. [En línea] 2016. [Citado el: 13 de 2 de 2016.] http://www.kdictionaries.com/. 2.

3. Moreira, Manuel Area. *e-Learning: Enseñar y Aprender en Espacios Virtuales.* Malaga : s.n., 2009. 3.

4. instituto Nacional de Tecnologias de Educatibas y de Formacion del Profesorado(españa)(pajgins oficial). *educaLAB.* [En línea] [Citado el: 18 de 2 de 2016.] http://recursostic.educacion.es. 4.

5. Prof. Grisolía, M. web del profesor. [En línea] 29 de 8 de 2010. [Citado el: 22 de 3 de 2016.] http://webdelprofesor.ula.ve/humanidades/marygri/recursos.php.

6. MULLIGAN, A., HALL, L., RAPHAEL, E. Peer Review in a changing world: an international study measuring the attitudes of researchers. [En línea] [Citado el: 23 de 4 de 2016.] http://dx.doi.org/10.1002/asi.22798.

7. Rodríguez, Lic. Ernesto G. Revista cubana de Imformacion en ciencias de la salud. [En línea] 29 de 1 de 2013. [Citado el: 20 de 2 de 2016.] http://www.acimed.sld.cu/index.php/acimed/article/view/410/304. 5.

8. Dr:CAÑIZARES GONZÁLEZ, ROXANA. repositorio institucional. [En línea] [Citado el: 2 de 4 de 2016.] http://repositorio_institucional.uci.cu/jspui/handle/ident/7913.

9. e-ABC. [En línea] [Citado el: 23 de 2 de 2016.] http://www.e-abclearning.com/definicione-learning.

10. e-ABC. [En línea] [Citado el: 23 de 2 de 2016.] referencia 4http://www.e-abclearning.com/queesunaplataformadeelearning.

11. *Reflexiones en torno al concepto « Recursos de información».* Mocoso, P. 1998, Revitsa general de información y documnetaciòn, Vol. volumen 8.

12. *El movimiento open: la creación de un dominio público en la era digital.* ARI-ÑO, A. España : s.n., 2009.

13. Ministerio de Educación Nacional Colombiano. aprende en linea. [En línea] [Citado el: 23 de 2 de 2016.] http://aprendeenlinea.udea.edu.co/lms/men/oac1.html.

14. *Diseño y construcción de objetos de aprendizaje web desde la perspectiva tecnopedagógica para la enseñanza y aprendizaje en las comunidades virtuales.* Sprock,

Antonio Silva. 2013, Ciencias, tecnologías y culturas: Educación y nuevas tecnologías. .

15. Ainhoa Otamendi, Karen Belfer,John Nesbit,Tracey Leacock. *Instrumento para la evaluación de objetos de aprendizaje (LORI_ESP) Manual de usuario .* 2010.

16. California State University. MERLOT. *Multimedia Educational Resources for Learning an Online Teaching.* [En línea] [Citado el: 22 de 2 de 2016.] http://info.merlot.org/merlothelp/index.htm#who_we_are.htm.

17. Carnegie Mellon University. Software Engineering Institute. [En línea] [Citado el: 26 de 3 de 2016.] http://www.sei.cmu.edu/productlines/.

18. Letelier, Patricio. Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP). [En línea] 2006. [Citado el: 26 de 2 de 2016.] http://www.cyta.com.ar/ta0502/v5n2a1.htm. .

19. Universidad de las Ciencias Informáticas . *Metodología de desarrollo para la Actividad productiva de la UCI. .* La Habana : s.n., 2015.

20. James Rumbaugh, Ivar Jacobson,Grady Booch. *Unified Modeling Language Reference Manual.* s.l. : Pearson Higher Education, 2004 .

21. Visual Paradigm International Ltd. *Visual Paradigm Quick Start .* 2016.

22. Cáceres González, Abdiel. *Lenguajes de programación.* México : s.n., 2004.

23. BAKKEN, S. S. y A. AULBACH. *Manual de PHP.* 2001.

24. w3c. [En línea] [Citado el: 25 de 2 de 2016.] http://www.w3.org/standards/webdesign/htmlcss.

25. Rodriges y juan. gestiopolis. [En línea] 17 de 3 de 2005. [Citado el: 7 de 3 de 2016.] http://www.gestiopolis.com/definicion-javascript/ .

26. Potencier.Fabien, Zaninotto.François. *Symfony la guía definitiva.* 2007.

27. Alvarez, Miguel A. *Manual de jQuery.* 2009.

28. Otto, Mark. getbootstrap. [En línea] 2010. [Citado el: 29 de 4 de 2016.] http://getbootstrap.com/about/..

29. l Acedo, Jose. Apuntes de Programación. [En línea] 4 de 5 de 2015. [Citado el: 23 de 5 de 2016.] http://programacion.jias.es/2015/05/web-%C2%BFque-es-el-framework-bootstrap-ventajas-desventajas/. .

30. Oracle Corporation. netbeans. [En línea] [Citado el: 8 de 3 de 2016.] netbeans.org.

31. The PostgreSQL Global Development Grou. *PostgreSQL 9.3.11 Documentation.* California : s.n., 2015.

32. PRESSMAN, Roger. *Ingenieria del Software. Un enfoque práctico. 5ta.* Mexico : s.n., 2000.

33. cyta. [En línea] 15 de 1 de 2006. [Citado el: 23 de 3 de 2016.] http://www.cyta.com.ar/ta0502/v5n2a1.htm.

34. upiicsa. [En línea] [Citado el: 25 de 3 de 2016.] http://www.sites.upiicsa.ipn.mx/polilibros/portal/polilibros/Complemento Material Didactico/Maest-Ing-Soft-Sergio/Cuerpoconocimiento/Diseño de software.htm.

35. *Estilos y patrones de arquitectura y de diseno de software.* Elias, Dr.Rodríguez y Mario, Oscar. Instituto Tecnológico de Hermosillo : s.n., 2013.

36. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* Reynoso, Carlos, Kicillof y Nicolás. BUENOS AIRES : s.n., 2004.

37. EGUILUZ, J. *desarrollo aguil en symfony2.* 2013.

38. Larman, Craig. *UML y Patrones. Introducción al análisis y diseño orientado a objetos. .* México : s.n., 2003.

39. Jacobson, Ivar, Booch, Grady y Rumbaugh, James. *El Proceso Unificado de Desarrollo de Software.* 2000.

40. Symfony2 Spanish Translation Team. Symfony2 Spanish Documentation. [En línea] 2011. [Citado el: 5 de 5 de 2016.] http://test-sf-doc-es.readthedocs.io/en/latest/contributing/code/standards.html.

41. JUNTA DE ANDALUCÍA. JUNTA DE ANDALUCÍA. [En línea] [Citado el: 4 de 5 de 2016.] http://www.juntadeandalucia.es/servicios/madeja/contenido/libro-pautas/227.

42. Jacobson, Ivar, Booch, Grady y Rumbaugh, James. *El Proceso Unificado de Desarrollo de Software.* 2000.

43. Gutierrez, D. codecompiling. [En línea] [Citado el: 24 de 4 de 2016.] http://www.codecompiling.net/files/slides/UML_clase_06_UML_secuencia.pdf.