

23 de junio del 2014

Universidad de las Ciencias Informáticas

Facultad 4



*Herramienta para personalizar la evaluación de
ejercicios creados bajo la especificación
IMS-QTI v2.0*

*Trabajo de diploma para optar por el título de
Ingeniero en Ciencias Informáticas*

Autores

Idainys Cruz Batista

Raúl Infante Fonseca

Tutores

Ing. Cindy Santos Salgado

Ing. Adrián García Sánchez

DECLARACIÓN DE AUTORÍA

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas (UCI) a que haga el uso que estimen pertinente con el mismo.

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

AUTORA

Idainys Cruz Batista

AUTOR

Raúl Infante Fonseca

TUTORA

Ing. Cindy Santos Salgado

TUTOR

Ing. Adrián García Sánchez



AGRADECIMIENTOS

AGRADECIMIENTOS

A mis padres Robertico y Kirenia por ser dedicados en mi educación, por darme valores y brindarme tanto amor.

A mi hermosa hermana por quererme tanto y darme mucha fuerza, ella es única, la adoro y estoy muy feliz de que forme parte de mi vida.

A toda mi familia, mis abuelos, mis tíos, mis primos que de una forma u otra siempre me han apoyado.

A mi novio Manuel por ser amigo y compañero, por compartir estos 4 años de su vida conmigo y brindarme todo el amor del mundo, por confiar en mí y ser mi apoyo en los momentos malos y buenos, Te Amo.

A mi cuñado Yariel, por ser la felicidad de mi hermana, por cuidarla y quererla, y por ser parte de mi familia.

A mis suegros por todo su cariño, apoyo y ser la niña linda de su casa.

A mis grandes amigas Yania y Yudis que han estado conmigo en las buenas y en las malas y con las cuales he pasado excelentes e inolvidables momentos. Las quiero mucho.

A mis compañeros de aula, en especial a mis amigos Frank, Delfín, Ernesto, Carlos Gonze, Angelito, Manolo, Luis Daniel, Leonel, Luciano, Lázaro al grupo de primer año, donde curse y pase los momentos más felices de la universidad con personas que actualmente adoro. Y a mis amiguitas del apartamento Claudia, Nany y Geo.

A todos los educadores que han contribuido con mi formación profesional, en especial a los profes Sandra, Dixon, Yusdanys, Dania, Mallea, Roxana, Adrián, Cindy por su apoyo y por creer en mí.

A mi compañero de tesis por su esfuerzo y dedicación en el desarrollo de la investigación.

A mis tutores Cindy y Adrián por confiar en mí, por brindarme todos sus conocimientos, por su dedicación para la realización del trabajo y por ser parte del mejor equipo de proyecto productivo en el que haya estado.

A todos los que de una forma u otra han contribuido a la realización del presente trabajo. A todos aquellos que no he mencionado por falta de espacio pero que estuvieron ahí brindando su ayuda.

*A todos, muchas gracias. **Idi***



AGRADECIMIENTOS

AGRADECIMIENTOS

Quiero agradecer a mis padres por el apoyo dado en todo momento.

A mi mami Zaida mi querida abuela que tanto cuidado de mí.

A mis abuelos Raúl y Emma, a mis tíos Alexander; Robert; Yuriel; Xiomara y Dago.

A mis primos que quiero tanto.

A mis tutores por la ayuda dada en todo momento y a mi compañera de tesis Idainys que mejor no pudo ser.

A Rasiel y Maggie; a mis amigos del IPVCE Angel y Yaniel, a mis amigos que he tenido en estos 5 años en los grupos en los que estuve y los compañeros del viejo apartamento y el nuevo.

Agradecerles a todos los que de una manera u otra han contribuido a que este trabajo allí sido realizado.

A todos, muchas gracias. Raúl



DEDICATORIA

DEDICATORIA

A mis padres y mi hermana por todo su amor, comprensión y dedicación. Porque a pesar de los malos días se levantan a luchar para que tengamos un futuro mejor, me han educado tanto, son los mejores y lo que más quiero en el mundo.

A Manu, por ser el gran amor de mi vida y amarme incondicionalmente.

A mis grandes amigos por estar ahí cada día y vivir mis tristezas y alegrías.

Idi

A mi familia; en especial a mis padres; que siempre me han apoyado y que gracias a ellos he llegado hasta aquí, y a mi abuelo papi Mejías que ya no se encuentra entre nosotros y a toda mi familia.

Raúl

RESUMEN

La evaluación generada por los Sistemas de Gestión del Aprendizaje es única en cada plataforma educativa. Los profesores son los encargados de crear los ejercicios, cuestionarios y pruebas necesarias para evaluar a los estudiantes en una determinada habilidad, sin embargo, no conocen cómo se genera el algoritmo que indica su evaluación en estos sistemas. La presente investigación tiene como objetivo desarrollar una herramienta que permita personalizar la evaluación de ejercicios creados bajo la especificación IMS-QTI v2.0. Esta herramienta contribuye a que los profesores con escasos conocimientos en algoritmia, modifiquen un algoritmo de forma visual, proporcionando un grupo de funcionalidades que le facilitan cambiar la evaluación de un ejercicio. Para guiar la documentación y estructura de la investigación se utilizó la metodología de desarrollo RUP, generándose en cada flujo de trabajo los artefactos correspondientes, PHP v5 como lenguaje de programación, las hojas de estilo en cascada para la presentación de los contenidos, como librería CSS, Bootstrap v3.0.0 y Symfony v2.3.7 como *framework* de desarrollo. Para la validación de la propuesta de solución se realizaron las pruebas funcionales y unitarias, y para determinar el grado de satisfacción del cliente se realizó una encuesta, obteniendo resultados satisfactorios.

Palabras claves: algoritmo, ejercicio, evaluación, IMS-QTI v2.0, LMS.

ÍNDICE DE CONTENIDOS



ÍNDICE DE CONTENIDOS

Introducción	1
CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA	7
1.1 Conceptos asociados al dominio del problema	7
1.1.1 Evaluación.....	7
1.1.2 Tipos de evaluación.....	8
1.1.3 Evaluación en entornos virtuales	10
1.2 Estándar IMS	15
1.3 Sistemas similares existentes	16
1.4 Herramientas y tecnologías a utilizar	19
1.4.1 <i>Framework</i> de desarrollo.....	20
1.4.2 Gestor de base de datos	20
1.4.3 Librería de CSS	21
1.4.4 Lenguajes del lado del cliente.....	21
1.4.5 Lenguaje al lado del servidor.....	23
1.4.6 Lenguaje de Modelado	23
1.4.7 Metodología para el desarrollo	24
1.4.8 Herramienta CASE (<i>Ingeniería de Software Asistida por Computadora</i>) para el modelado	27
1.4.9 Servidores Web.....	27
1.4.10 Entorno de Desarrollo Integrado (IDE)	28
1.5 Conclusiones parciales	29
CAPÍTULO 2. CARACTERÍSTICAS DEL SISTEMA	31
2.1 Modelo de dominio	31
2.1.1 Conceptos del dominio.....	31
2.1.2 Diagrama de modelo de dominio.....	32
2.1.3 Descripción de la herramienta propuesta	33
2.2 Requerimientos del <i>software</i>	33
2.2.1 Requerimientos funcionales.....	33
2.2.2 Requisitos no funcionales.....	34
2.3 Patrones de caso de uso	35
2.4 Modelo de casos de uso de la herramienta	35
2.4.1 Descripción de los actores del sistema.....	35
2.4.2 Diagrama de casos de usos de la herramienta	36
2.4.3 Descripción de los casos de usos de la herramienta.....	37
2.5 Conclusiones parciales	41
CAPÍTULO 3. ANÁLISIS Y DISEÑO DE LA PROPUESTA	42
3.1 Modelo de análisis	42
3.1.1 Diagramas de clases de análisis	42

ÍNDICE DE CONTENIDOS



3.1.2	Diagramas de colaboración.....	43
3.2	Patrón arquitectónico.....	44
3.2.1	Aplicación de los patrones de diseño en Symfony2.....	44
3.3	Modelo de diseño.....	46
3.3.1	Diagrama de clases del diseño.....	47
3.3.2	Diagrama de secuencia del diseño.....	48
3.3.3	Diagrama de despliegue.....	48
3.3.4	Diseño de la base de datos.....	49
3.4	Conclusiones parciales.....	50
CAPÍTULO 4. IMPLEMENTACIÓN Y PRUEBA.....		51
4.1	Modelo de implementación.....	51
4.1.1	Diagrama de componentes.....	51
4.2	Validación del diseño de la herramienta PERA (IMS-QTI v2.0).....	52
4.3	Ejemplo de los patrones de diseño alta cohesión y <i>strategy</i>.....	55
4.4	Pruebas de <i>software</i>.....	55
4.4.1	Métodos de prueba.....	56
4.5	Descripción de la validación de la hipótesis.....	63
4.6	Conclusiones parciales.....	65
Conclusiones generales.....		66
Recomendaciones generales.....		67
Referencias bibliográficas.....		68
ANEXOS.....		73
5.1	Anexo 1. Operacionalización de las variables.....	73
5.2	Anexo 2. Descripciones de los Casos de Usos.....	74
5.3	Anexo 3. Diagrama de clase de análisis.....	86
5.4	Anexo 4. Diagramas de colaboración.....	87
5.5	Anexo 5. Diagramas de secuencia.....	92
5.6	Anexo 6. Diagramas de Componentes Interfaz Principal.....	97
5.7	Anexo 7. Métricas de diseño para la validación de la herramienta PERA (IMS-QTI v2.0).....	99
5.8	Anexo 8. Diseño de casos de pruebas.....	103
5.9	Anexo 9. Encuesta para profesores.....	108

Introducción

El uso de las TIC (*Tecnologías de la Información y las Comunicaciones, por sus siglas en inglés*) en sus distintas fases y sistemas educativos, tienen un representativo efecto en el desarrollo del aprendizaje de los estudiantes, en el dinamismo de sus competencias para la vida y el trabajo que lo ayudará en su inserción dentro de la sociedad del conocimiento. Se vive en una sociedad que avanza rápidamente gracias al desarrollo tecnológico, donde las TIC han cambiado completamente el paradigma de la educación, dejando una huella en muchos campos del saber. En cuanto a educación, las TIC han probado que pueden ser de gran ayuda tanto para los profesores, como para los estudiantes. La facilidad de las herramientas tecnológicas en la educación no tiene como objetivo sustituir al profesor, sino que pretende ayudarlo aportándole más comodidad, elementos visuales y auditivos, conocimientos más actuales y distintos beneficios que permiten enriquecer aún más el proceso de enseñanza-aprendizaje.

Al no tener fronteras, Internet abre la posibilidad de la educación no presencial, y con ello, que cualquier alumno pueda participar en un curso desde cualquier ordenador que se encuentre en la red.

Al existir la posibilidad de poder educar de forma no presencial mediante los cursos en línea de los sistemas educativos, también fueron encontradas nuevas formas de evaluar automáticamente, permitiendo crear exámenes rápidos, cuestionarios, ejercicios incorporándole texto, imágenes, sonidos y videos. Con la aplicación de estos exámenes rápidos, el tutor podrá darse cuenta si los alumnos entendieron la lección y las dificultades que tuvieron, marcando así un proceso importante que ayuda de una forma u otra a que el maestro o consultante, tenga definido y controlado la situación de cada estudiante que esté matriculado en su curso.

La evaluación es un elemento importante del proceso de enseñanza-aprendizaje. Según Elizabeth Aguirre Rodríguez *“La evaluación de la docencia es un proceso de suma importancia, ya que es uno de los principales agentes del desarrollo educativo y son ellos quienes generan en los estudiantes el conocimiento, las habilidades, actitudes y valores, por ello, no se puede poner a discusión si la práctica docente debe ser evaluada o no”* (2000). También Vargas y Calderón afirman que *“la evaluación es un proceso objetivo que busca aclarar aspectos de la realidad, es el uso de parámetros, indicadores y métodos que promuevan la objetividad y la distancia entre lo que se evalúa y las personas involucradas, por lo tanto, es la medición de un fenómeno”* (2005).

En la terminología más común, se suele hablar de dos tipos de evaluación, sumativa y formativa. La evaluación sumativa, según Green (2004) no se restringe de forma específica a una prueba o examen que se exige a los alumnos al culminar un proceso de enseñanza-aprendizaje concreto. La evaluación sumativa engloba todos aquellos exámenes, ejercicios o pruebas que se realizan, única y exclusivamente con el objetivo de comprobar si el alumno conoce o no el contenido de una disciplina. La evaluación formativa por su parte tiene en cuenta todos los ámbitos del desarrollo de la persona, es decir se centra no solo en los aspectos cognitivos, sino también en factores afectivos y sociales. Supone un gran esfuerzo por parte del docente ya que exige una toma de decisiones continua (Marchesi y Martín, 1998), lo que permite incorporar por medio de la retroalimentación las ayudas, medios y recursos que un alumno necesite en el momento en que realmente los precisa.

Además, de la forma de evaluación que se quiera realizar con un grupo determinado de estudiantes que estén matriculados en un curso en línea, también los profesores necesitan buscar estrategias para retroalimentar los errores obtenidos por los evaluados. Debe considerarse que la evaluación es un proceso de retroalimentación en el cual el evaluador y el evaluado entran en constante interrelación.

Debido a la gran diversidad de los sistemas de educación a distancia y la producción masiva de información que son capaces de generar; surge la necesidad de estandarizar las áreas de la educación en estos sistemas educativos, teniendo a la evaluación como una de estas áreas.

Una de las principales instituciones vinculadas a la estandarización de los sistemas de educación a distancia es IMS¹, consorcio donde se agrupan vendedores, productores, implementadores y consumidores de estos sistemas, que se enfoca completamente a desarrollar especificaciones en formato XML². Esta institución presenta una serie de especificaciones de las cuales IMS-QTI v2.0³ es la solución más acertada para la estandarización de las evaluaciones, teniendo como mayor ventaja la reducción del costo y tiempo de desarrollo de estos objetos, gracias a la reutilización e intercambio de ejercicios y pruebas entre diferentes sistemas de educación a distancia.

¹ IMS (*Sistemas de Gestión Instruccional, por sus siglas en inglés*).

² XML es un Lenguaje de Etiquetado Extensible muy simple, pero estricto que juega un papel fundamental en el intercambio de una gran variedad de datos. Es un lenguaje muy similar a HTML pero su función principal es describir datos y no mostrarlos como es el caso de HTML. XML es un formato que permite la lectura de datos a través de diferentes aplicaciones.

³ IMS-QTI v2.0 (*Interoperabilidad de Cuestionarios y Pruebas versión 2.0, por sus siglas en inglés*).

Las plataformas educativas, permiten a los profesores crear ejercicios, cuestionarios, pruebas para evaluar a sus estudiantes en una determinada habilidad. Estos ejercicios contienen internamente una plantilla en donde aparece la evaluación y retroalimentación del ejercicio, esta parte de la plantilla es llamada *rptemplates* y está basada en la especificación QTI v2.0. La especificación QTI v2.0 está definida en XML, la misma plantea un modelo en el que se definen los componentes principales que intervienen en el proceso de evaluación y un formato de contenido para almacenar las preguntas de manera independiente del sistema o herramienta de autoría utilizada para crearlas. Luego este *rptemplates* es interpretado por el motor de evaluación⁴ que contiene la plataforma educativa y emite una nota y retroalimentación para los estudiantes que resuelvan los ejercicios propuestos por el profesor.

QTI v2.0 define un lenguaje utilizado para la evaluación de los ejercicios por parte del motor de evaluación; los algoritmos definidos con este lenguaje pueden ser propensos a errores por el hecho de ser interpretado, al ser escrito en XML dificulta su lectura para profesores y las modificaciones directamente pueden afectar el correcto funcionamiento del motor de evaluación. Por lo que existe una **lista de problemáticas** que dificultan al profesor participar en el proceso de generar la evaluación de un ejercicio.

- El proceso que genera cómo evaluar y retroalimentar un ejercicio lo realizan los desarrolladores que trabajen para la plataforma puesto que es en lenguaje XML y los profesores tienen poco conocimiento de algoritmia⁵ para modificar este *rptemplates*.
- Al ser XML un lenguaje interpretado, es propenso a errores por lo que si se realiza alguna modificación por personas que no tengan conocimiento del mismo, proporcionaría errores en la interpretación del motor de evaluación de la plataforma educativa.
- El motor de evaluación de la plataforma interpreta el *rptemplates* generado por los desarrolladores sin considerar las preferencias de los estudiantes, conocimientos y estilos de aprendizaje y desaprovecha las actualizaciones de los objetos de aprendizaje.

⁴ Motor de evaluación (assessment engine): Se encarga de generar los resultados de los test en función de los objetos del repositorio ASI que se hayan usado. Este motor tiene la función de procesar las respuestas de los estudiantes y emitir una evaluación.

⁵ La algoritmia es un pilar fundamental de las ciencias de la computación puesto que provee métodos de solución de problemas, que serán implementados en los programas de computadora. En este sentido, un programa de computadora es la implementación de un algoritmo en un determinado lenguaje de programación. Un algoritmo es una secuencia ordenada, finita y lógica de pasos que permite resolver un problema.

Luego de analizarse detenidamente la **situación problemática**, se plantea el siguiente **problema a resolver**: ¿Cómo permitir a los profesores con escasos conocimientos de algoritmia personalizar la evaluación de los ejercicios creados bajo la especificación IMS-QTI v2.0?

Esta problemática se enmarca en el **objeto de estudio**: La personalización de la evaluación de los ejercicios creados en las plataformas educativas. Teniendo como **campo de acción**: Las herramientas para la personalización de evaluación de ejercicio creados bajo la especificación IMS-QTI v2.0.

Para darle solución al problema planteado se especifica como **objetivo general**: Desarrollar una herramienta que permita a los profesores con escasos conocimientos de algoritmia, modificar algoritmos de forma visual.

Se proyecta como **hipótesis de la investigación**: El desarrollo de una herramienta que permita a profesores de escasos conocimientos en algoritmia, modificar algoritmos de forma visual, contribuye a personalizar la evaluación de ejercicios creados bajo la especificación IMS-QTI v2.0.

Definición conceptual de las variables independientes:

Modificar algoritmos de forma visual a profesores de escaso conocimiento en algoritmia: Las funcionalidades que tendrá la herramienta y que permitirán que el profesor pueda modificar fácilmente un algoritmo mediante un diagrama el cual será mostrado por la herramienta a desarrollar.

Definición conceptual de las variables dependientes:

Personalizar la evaluación de ejercicios creados bajo la especificación IMS-QTI v2.0: Lo que se logrará con la realización de la herramienta deseada.

En el **Anexo 1** se muestra la definición operacional de las variables.

Objetivos específicos:

1. Investigar los aspectos teóricos fundamentales que sustentan la investigación, mediante consultas, extracción y recopilación de información relevante sobre el problema a investigar.
2. Diseñar la herramienta para la personalización de evaluaciones ejercicios creados bajo la especificación IMS-QTI v2.0.
3. Implementar la herramienta para la personalización de evaluaciones de ejercicios creados bajo la

especificación IMS-QTI v2.0 en forma de producto.

4. Realizar pruebas funcionales para detectar los posibles errores que dificulten el funcionamiento de la herramienta.

Los métodos que se utilizan para la presente investigación son:

Métodos empíricos:

- Observación: Se utiliza para diagnosticar el funcionamiento de herramientas similares, que puedan ser de gran ayuda en el desarrollo de la herramienta deseada.
- Encuesta: Para verificar la aceptación de los clientes que van a utilizar la herramienta a desarrollar

Métodos teóricos:

- Modelación: Se crean abstracciones con el propósito de explicar la realidad. El modelo es el sustituto del objeto de investigación. Representa de manera gráfica los artefactos que intervienen en el proceso de desarrollo de software.
- Analítico– Sintético: Para el estudio de las fuentes bibliográficas y materiales relacionados con el tema, identificando elementos esenciales para darle solución al problema planteado.
- Inductivo-deductivo: Se usó en busca de la descripción e implementación de las funcionalidades propuestas, partiendo de un problema concreto a la generalización del mismo que permita la búsqueda de solución basada en la propuesta de esta investigación.
- Análisis histórico – lógico: Permite realizar un estudio de las tendencias de la evaluación en entornos virtuales y las herramientas similares existentes.

Estructura Capitular

La tesis está estructurada en cuatro capítulos.

En el **capítulo 1** se analiza y exponen los elementos teóricos que soportan la investigación, se presentan los lenguajes de programación y modelado, así como tecnologías y metodologías que se ajustan al desarrollo del trabajo, fundamentando su selección en base al estudio realizado. Se describen las soluciones similares existentes y se realizan comparaciones para llegar a la propuesta de solución que sustenta la investigación.

En el **capítulo 2** se realiza una descripción de las características de la herramienta a implementar. Además, se especifican los requisitos funcionales y no funcionales, se describen a los actores de la herramienta y se realizan los diagramas: Modelo de dominio y Casos de usos de la herramienta.

En el **capítulo 3** se modelan los diagramas de clases del análisis y del diseño para cada caso de uso de la solución propuesta. Además, se elaboran los diagramas de paquetes del análisis y diseño, los diagramas de interacción del análisis, el diagrama de despliegue, de colaboración y el diseño de la base de datos y sus respectivas descripciones.

En el **capítulo 4** se hace una descripción de la aplicación de los métodos de validación empleados para comprobar la validez de la hipótesis planteada. Se implementa la solución propuesta, utilizando el lenguaje de programación y la metodología seleccionada, documentándose todo el proceso. Se definen los tipos de pruebas, se diseñan los casos de prueba que se aplicarán a la herramienta y se muestran los resultados de los diferentes tipos de pruebas realizadas.

Por último se presentan las **conclusiones** y **recomendaciones** que derivan de la investigación, las referencias bibliográficas consultadas y los **anexos** que contienen la documentación probatoria de los aspectos más significativos del proceso de investigación.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

En este capítulo se fundamentan las bases de la investigación, se representan los conceptos relacionados con el proceso de evaluación de la enseñanza-aprendizaje, su importancia, así como sus características fundamentales para tener una mejor perspectiva de la investigación.

1.1 Conceptos asociados al dominio del problema

1.1.1 Evaluación

Con el objetivo de llegar a conclusiones acertadas sobre la amplitud de conceptos en el tema de la evaluación, se plantearán distintas definiciones.

En la década del 70, Daniel Stufflebeam propicia una concepción de la evaluación entendida como “*un proceso de recolección de información útil que permite facilitar la toma de decisiones*”, señala Pedro Ahumada Acevedo que esta concepción del proceso de evaluación se ha ido modificando lentamente a través del tiempo e incorporando lo mejor del juicio, lo mejor de la medición y lo mejor del logro de los objetivos. Esta definición se válida teniendo en cuenta los nuevos procesos de los sistemas evaluativos los cuales ayudan continuamente a los estudiantes ya que este proceso ha ido evolucionando tanto las herramientas como los métodos de evaluación.

Ahumada también hizo referencia acerca de un tema muy asociado a la presente investigación señalando: “*Hoy en día, los investigadores del aprendizaje suelen dar una mayor relevancia a la evaluación de los procesos de aprendizaje sobre los resultados, ya que consideran importante el desarrollo de ciertas capacidades y habilidades de pensamiento, la comprensión de los contenidos curriculares y su relación con la vida real. Esta concepción educativa exige el desarrollo de una evaluación individualizada que se contraponen con la concepción didáctica vigente, centrada en una enseñanza grupal y estandarizada*”. Es importante decir sobre esta definición, que la evaluación también tiene que ser productiva, estimulante, y al mismo tiempo tiene que brindarle conocimientos al estudiante.

La evaluación tiene varias formas de estimular a un estudiante, la primera es medir su nivel de aprendizaje, por lo que el mismo, nunca va a igualarse con el de sus compañeros, se concuerda con Ahumada ya que la evaluación debería ser personalizada para cada estudiante.

Pulgar señala que la evaluación, en sí misma, tiene una función revalorizadora de todo el proceso al aumentar la valoración que el alumnado tiene de la propia acción formativa. Da más valor a nuestras

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

acciones de educación al permitir crear el ciclo perfecto de diseño, ejecución, reajuste (2005). Es una definición muy acertada ya que la evaluación permitirá que el estudiante se mida personalmente, informándose él mismo de cuan bien o mal se encuentra en su proceso de aprendizaje, posibilitándole la opción de mejorar y conocer los errores cometidos para futuras evaluaciones.

Según Pila Teleña: *"La evaluación es una operación sistemática, integrada en la actividad educativa con el objetivo de conseguir su mejoramiento continuo, mediante el conocimiento lo más exacto posible del alumno en todos los aspectos de su personalidad, aportando una información ajustada sobre el proceso mismo y sobre todos los factores personales y ambientales que en esta inciden"* (2012).

El objetivo principal que debe tener la evaluación debe ser el mejoramiento continuo de los estudiantes, este autor caracteriza este proceso indicando que se le debe brindar al estudiante los conocimientos de la forma más sencilla posible, pero sin dejar atrás sus preferencias cotidianas y su manera de adquirir el conocimiento.

Por las definiciones expuestas anteriormente se puede decir que la evaluación ha ido evolucionado lentamente y lo seguirá haciendo ya que este proceso se desliza hacia la perfección. Para poder realizar de forma general una definición inequívoca y teniéndose en cuenta todos los conceptos asociados a la evaluación de los estudiantes se llegó a la conclusión de que esta, es el motor impulsor que ayuda directamente a los estudiantes, ayudándoles con un mejoramiento continuo y midiéndole su nivel de aprendizaje. Los alumnos no tienen las mismas formas de adquirir el conocimiento, la misma forma de razonar, comprender situaciones dadas; tienen preferencias individuales y características que los definen.

Si a una plataforma educativa le brindas un servicio que le permita al profesor personalizar el modo de evaluación y retroalimentación de un ejercicio en línea, sería un aporte importante en la educación electrónica ya que le ofrece a cada educado su propia forma de evaluación y de aprendizaje continuo.

1.1.2 Tipos de evaluación

En esta investigación se hace referencia a dos tipos de evaluación, formativa y sumativa. El primero que formuló el binomio evaluación sumativa y formativa fue Scriven (1967), observándose un claro avance al conceder relevancia no solo a los resultados (como señala Tyler), sino también al proceso de evaluación en sí. Se puede decir que distintos autores definen ambas modalidades de evaluación (Scrive, 1967; Coll, 1983; Rosales, 1990; Biggs, 1999; Brown y Glasner, 2003). Todos ellos coinciden en afirmar que la evaluación

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

sumativa es aquella que se desarrolla al final del proceso de enseñanza-aprendizaje y que permite tomar decisiones vinculadas a la expedición de certificados, títulos o promoción; mientras que la evaluación formativa se centra en el proceso propio de la enseñanza-aprendizaje con objeto de mejorarlo durante su fase de desarrollo. A continuación se especifica cada evaluación señalada anteriormente.

Formativa o Procesual

La función de esta evaluación es obtener información acerca del estado del aprendizaje de cada estudiante y, a partir de ello, tomar decisiones que ayuden a un mejor desarrollo de dicho proceso. Son factores de evaluación formativa: los trabajos extra clase de investigación y consulta, los proyectos, los trabajos individuales y grupales en clase, los cuestionarios, las exposiciones, las evaluaciones orales, las prácticas de laboratorio, los compromisos académicos, la participación activa, la asistencia, la puntualidad, el interés, la motivación, la creatividad, la responsabilidad y la actitud investigativa (Ministerio de Educación Nacional: República de Colombia, 1996).

La evaluación formativa consiste en la valoración que se realiza a través de la recogida continua y sistemática de datos, del funcionamiento de un centro, de un programa educativo, del proceso de aprendizaje de un alumno, de la eficacia de un profesor. A lo largo del período de tiempo fijado para la consecución de unas metas u objetivos. La evaluación procesual es de gran importancia dentro de una concepción formativa de la evaluación, porque permite tomar decisiones de mejora sobre la marcha (Alamo, 2010).

Sumativa o Final

Su objetivo es conocer y valorar los resultados conseguidos por el alumno al finalizar el proceso de enseñanza-aprendizaje, también es conocida con el nombre de evaluación final (Ramos, 1986). Para Fernando Ojeda (2006), la evaluación sumativa es la que se realiza al terminal una etapa del proceso de enseñanza-aprendizaje para verificar sus resultados. Determina si se lograron los objetivos educacionales estipulados, y en qué medida fueron obtenidos para cada uno de los alumnos. La evaluación final de modo principal tiene como finalidad la calificación del alumno y la valoración del proyecto educativo del programa desarrollado. Este autor plantea que entre los fines o propósitos de la evaluación sumativa se destacan la posibilidad de verificar si un alumno domina una habilidad o conocimiento, proporcionar bases objetivas para asignar una calificación, informar acerca del nivel real en que se encuentran los alumnos y señalar pautas para investigar acerca de la eficacia de una metodología.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

Se elaboró un cuadro comparativo a partir de (Neira, 2000), señalando algunos puntos importantes para la investigación.

Tabla 1. Tabla comparativa, elaboración a partir de Rodríguez Neira (2000).

	Evaluación Formativa	Evaluación Sumativa
Cuándo	Durante el proceso de enseñanza-aprendizaje.	Al final del proceso de enseñanza-aprendizaje.
Objetivo	Mejorar el proceso y también los resultados.	Determinar el grado en que los objetivos propuestos se han o no logrado.
Definición según (Rodríguez Neira, 2000)	Es una evaluación permanente para determinar el grado de adquisición de cada objeto, y detectar tanto los aspectos NO asimilados, como las causas para reajustar y “optimizar” los logros y el proceso de enseñanza-aprendizaje.	Es la evaluación final que emite un juicio sobre el alumno (a veces, aunque menos frecuente, también sobre el profesor o el currículo) en relación con el resultado del aprendizaje, reflejado en una calificación globalizada de un repertorio de objetivos, y de la que se sigue una toma de decisiones ejecutiva, a veces implícita.
Vinculada	Evaluación continua	Evaluación Final
Función	Retroalimentación o “ <i>feedback</i> ”	Control
Profesor	Agente activo, reflexivo.	Profesor técnico.
¿Qué evalúa?	Procesos	Resultados
Efectos	Continuos, permanentes.	A largo plazo.
Cuestiona	La validez interna del programa.	La validez externa del programa.
Permite tomar decisiones	Enseñanza correctiva.	Selección de entrada a un curso. Clasificación de sujetos, notas finales.

1.1.3 Evaluación en entornos virtuales

Los Sistemas de Gestión del Aprendizaje (*por sus siglas en inglés, LMS*), también llamados Entornos Virtuales de Aprendizaje (*por sus siglas en español, EVA*) son aplicaciones Web que proveen las funciones administrativas y de seguimiento necesario para posibilitar y controlar el acceso a los contenidos. Los LMS facilitan la interacción entre los docentes y los estudiantes, aportan herramientas para la gestión de contenidos académicos y permiten el seguimiento y la evaluación. (María Agudelo, 2008)

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA



La educación en los entornos virtuales se ha ido revolucionando y con ella la forma de pensamiento humano. Se ha logrado de una manera u otra utilizar la tecnología como herramienta para la educación propiciando así un nivel superior en la enseñanza educativa. Con el uso de las nuevas tecnologías de la información y las telecomunicaciones en la educación superior se ha querido simular prácticamente todo el proceso de enseñanza-aprendizaje. Esto conlleva a una enseñanza personalizada la cual nace con el fin de buscar las dificultades y posibilidades de cada alumno como sujeto individual.

Según algunos autores como Pellegrino, Chudowsky y Glaser, la evaluación educativa pretende determinar qué tan bien están aprendiendo los estudiantes y es parte integral de la búsqueda de una mejor educación, a través de un adecuado aprovechamiento de los recursos tecnológicos modernos. De donde es factible afirmar que sus propósitos básicamente son tres: apoyar el aprendizaje, medir el desempeño y valorar programas educativos (2001). Con estos recursos se puede llegar a lograr un sistema evaluativo personalizado que delimite cada uno de los propósitos anteriores, ayudando al mismo tiempo el proceso de enseñanza-aprendizaje. Señala Brusilovsky y Vassileva que *“el aprendizaje personalizado ya no es un tema de investigación el cual se enfrenta en la educación basada en la web de pequeña escala”* (2003), sino dicen Boticario y Santos *“un desafío concreto fomentado por la mayoría de los informes que se centran en atender necesidades individuales de cada alumno a través del uso de Tecnologías de la Información y las Comunicaciones”* (2006).

El desarrollo de sistemas de aprendizaje adaptativo ha sufrido un cambio considerable en la última década. Inicialmente hubo prototipos de investigación para el desarrollo de entornos de aprendizaje adaptativo, pero los esfuerzos más recientes se centraron en el suministro de soluciones generales en la ampliación existente de estándares educativos para apoyar la entrega del curso de adaptación frente a las necesidades individuales de los estudiantes (Paramythis, Loidl-Reisinger y Kepler, 2004) . En este sentido, se han realizado dos tipos de enfoques. Por un lado están los que proporcionan soluciones inteligentes para cubrir diferentes temas tales como: pruebas de inteligencia (Guzmán, Conejo, Pérez-de-la-Cruz, 2007), la captura y el análisis de los estudiantes para crear tutores de colaboración (Harrer, McLaren, Walker, Bollen y Sewall, 2006), basado en normas de adaptación con la selección de la estabilidad (De Bra, Smits y Stash, 2006), de autoría de adaptación *hyperbooks*⁶ (Murray, 2003), la reutilización de las actividades educativas a través de

6 Un *hyperbook* adaptativo para el aprendizaje basado en la web es un repositorio de información, que integra y personaliza un conjunto de distribuidos materiales de aprendizaje (incluyendo proyectos, análisis, etc.) utilizando modelos semánticos explícitos y metadatos.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

servidores distribuidos (Brusilovsky, 2004a), la generación dinámica de curso a través de las técnicas de planificación de Inteligencia Artificial (*IA, por sus siglas en inglés*)⁷ (Brusilovsky y Vassileva, 2003).

Evaluación Adaptativa

Las evaluaciones adaptativas por computadora (*CAT, por sus siglas en inglés*) a diferencia de los exámenes tradicionales, requiere un menor número de preguntas, debido a que la presentación de las mismas o la decisión de terminar son adaptativas, esto es, la representación de la evaluación depende de cálculos basados en el desempeño del evaluado en preguntas anteriores (Linn, 1993). Aunque ya desde Alfred Binet se han intentado procedimientos de evaluación adaptativos mediante tests psicológicos (sus tests de inteligencia tenían formas diferentes según el nivel educativo de los niños).

La evaluación personalizada o adaptativa tiene en cuenta las características del estudiante, sus circunstancias sociales, sus posibilidades y limitaciones. Esta evaluación puede desarrollarse como experiencia educativa en la que participan todo un curso. (Figueroa, Barrios, Aliaga & Baez, 2009)

El Centro de Enseñanza Superior constituye uno de los escenarios esenciales del desarrollo de la personalidad de cada estudiante, ya que en todas sus actividades y espacios de interrelación el alumno está comprometido de una forma holística⁸ con sus recursos personales.

El desarrollo de la personalidad del estudiante define al mismo como individualidad, permitiendo la individualización de sus diferentes actividades, entre las cuales está la evaluación. Las emociones que se producen durante la evaluación intervienen de diferentes formas en la constitución de los estados afectivos complejos que caracterizan toda actividad humana, como son la seguridad, independencia, autodeterminación, autoestima e intereses, los cuales pueden ser esencialmente productores de emociones positivas o negativas. De acuerdo al tipo de producción emocional que caracterice la evaluación, se producirá o no un crecimiento del alumno en el curso del proceso docente-educativo.

Una de las consecuencias de haber ignorado la personalidad en el proceso educativo, ha sido la tendencia a la estandarización de todas las actividades docentes, en particular, la evaluación. Los estudiantes no

⁷ La Inteligencia Artificial (IA) es una combinación de la ciencia del computador, fisiología y filosofía, tan general y amplio como eso, es que reúne varios campos (robótica, sistemas expertos, por ejemplo), todos los cuales tienen en común la creación de máquinas que pueden pensar.

⁸ La holística se refiere a la manera de ver las cosas enteras, en su totalidad, en su conjunto, en su complejidad, pues de esta forma se pueden apreciar interacciones, particularidades y procesos que por lo regular no se perciben si se estudian los aspectos que conforman el todo, por separado.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

aprenden, ni se desarrollan al mismo ritmo, por lo tanto estandarizar la evaluación crea inseguridad y tensión en los alumnos, obligándolos al seguimiento memorístico y reproductivo de lo aprendido. Una vez que el estudiante opta por la reproducción memorística como vía de evitar el fracaso, lo estudiado deviene como algo externo a su lógica, lo cual nunca logra personalizar, sintiéndolo paradójicamente⁹ cada vez más distante en la medida en que más avanza sobre el material.

Considerar al alumno como sujeto en el proceso de evaluación significa darle en dicho proceso un espacio participativo, reflexivo y de toma de decisiones. De manera que la evaluación se transforme de externa a personalizada. La evaluación personalizada se concibe como una unidad subjetiva¹⁰ del desarrollo personal, donde se manifiestan contradicciones y existe comunicación, donde se produce aprendizaje integrando lo que se aprende a la experiencia personal y al desarrollo de actividades auto evaluativas determinadas y creadas por él mismo.

Sin duda, la evaluación constituye una de las categorías didácticas que requiere atención dentro de cualquier proyecto educativo. La evaluación debe responder al modelo educativo para el desarrollo de la personalidad vigente en la Institución Docente, en particular debe responder al modelo de aprendizaje dirigido al desarrollo de la personalidad del estudiante. No tenerla en cuenta significaría un grave error con consecuencias lamentables para el alumno, pues entraría en una contradicción entre los nuevos cambios que se introducen y la evaluación descontextualizada que responde a otro modelo, quizás menos avanzado. La individualidad está relacionada con el ritmo de aprendizaje, estilo cognitivo¹¹, personalidad, aptitudes, hábitos y técnicas de trabajo intelectual, niveles de esfuerzo, resultados satisfactorios e insatisfactorios. Para ello se establece el nivel que se considera razonable para cada estudiante, estableciendo metas de carácter optativo y libre, de profundización, ampliación y enriquecimiento de los objetivos mínimos, esenciales y obligatorios para todos. (Bernaza, Gala, del Valle, Rodríguez, Cogle & Pulido, 2008)

Según Verónica Amela Tarongí *“los sistemas de educación a distancia eran originariamente cursos consistentes en un conjunto de páginas Web estáticas a través de las cuales navegaban los estudiantes.*

⁹ Paradójicamente significa una idea extraña opuesta a lo que se considera verdadero o a la opinión general.

¹⁰ La subjetividad es la propiedad de las percepciones, argumentos y lenguaje basada en el punto de vista del sujeto, y por tanto influida por los intereses y deseos particulares del mismo.

¹¹ Cognitivo se define como la facultad de un ser vivo para procesar información a partir de la percepción, el conocimiento adquirido (experiencia) y características subjetivas que permiten valorar la información. Consiste en procesos tales como el aprendizaje, razonamiento, atención, memoria, resolución de problemas, toma de decisiones y procesamiento del lenguaje.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

Estos evolucionaron a los LMS, la última tendencia son los Sistemas Educativos Basados en Web Inteligentes y Adaptativos (por sus siglas en inglés, AIWBES), que van más allá al construir un modelo de los objetivos, preferencias y conocimiento de cada alumno individual y al emplearlo en la interacción del estudiante durante su proceso de aprendizaje, con la intención de adaptarlo a sus necesidades. AIWBES es el resultado de la unión de los Sistemas Tutores Inteligentes (por sus siglas en inglés, ITS) y los Sistemas Hipermedia Adaptativos (por sus siglas en inglés, AHS). Estas herramientas inteligentes utilizan métodos de extracción de conocimiento o minería de datos para descubrir información útil sobre los estudiantes y sus patrones de estudio, con el objetivo de guiarlos durante su proceso de aprendizaje con el fin de optimizarlo” (2010).

La tutorización personalizada es el modo más efectivo para enseñar, pero resulta una tarea laboriosa y costosa, sin embargo, los sistemas tutores inteligentes son sistemas informáticos de aprendizaje personalizado, que no requieren la intervención de tutores humanos y reducen el coste, al automatizar la selección de los materiales del curso, su presentación y la evaluación de los estudiantes. Los ITS son sistemas de inteligencia artificial, más concretamente, sistemas expertos que simulan las características de un tutor humano. (Ong y Ramachandran, 2001)

Según Ing. Eucario Parra Castrillón con respecto a la creación de ambientes tecnológicos para el aprendizaje pedagógicamente efectivo, *“los investigadores han enfocado sus esfuerzos hacia el desarrollo de los STI. Los mismos son ambientes flexibles, interactivos y adaptativos para el aprendizaje. Flexibles porque abren abanicos de posibilidades para las navegaciones de los estudiantes. Interactivos porque los canales para la comunicación pueden permitir cruce de ideas del sistema tutor hacia el estudiante y de este hacia el sistema, en un diálogo. Adaptativos porque sobre la marcha el sistema puede cambiar las estrategias de enseñanza, variando los ejemplos, rompiendo las secuencias, demostrando imposibilidades y evidenciado equivocaciones a partir de casos concretos” (2004).*

En todo el mundo existen muchos STI desarrollados, se pueden destacar: Guidon (Clancey, 1991), Debuggy (Brown, 1989), Proust (Johnson, 1986) y Sierra (VanLehn, 1988).

Los STI usan la Inteligencia Artificial para incorporar conocimiento y lograr adquirir técnicas educativas para impartir una materia. Logran así controlar el nivel de aprendizaje de cada uno de los estudiantes y de esta forma se garantiza una instrucción personalizada. Estos sistemas se caracterizan por poseer un elevado nivel de especialización en el contenido que imparten, convirtiéndose en sistemas expertos. Estos sistemas

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

brindan gran apoyo al proceso educacional. Pueden ser usados como complemento de la instrucción brindada por el profesor, ya sea simplemente para reforzar conocimiento, dar asistencia a los estudiantes más talentosos o para dar asistencia a los estudiantes más lentos en el aprendizaje; e incluso pueden llegar a sustituir en muchos casos la presencia del profesor.

Un sistema de esta índole, también, podría darle al maestro información sobre el desempeño del estudiante para que pueda aplicar las medidas que considere apropiadas. Vale analizar que a nivel mundial la construcción de STI es un proceso muy costoso; requiere el uso de muchos recursos como tiempo, personal altamente calificado y tecnologías informáticas de alto nivel, pero su impacto es muy positivo en el apoyo al proceso enseñanza-aprendizaje en cualquier entorno.

1.2 Estándar¹² IMS

Es cierto que los sistemas de enseñanza virtual implementados bajo la modalidad de educación a distancia¹³ han chocado contra una serie de problemas que abarcan tanto cuestiones tecnológicas como pedagógicas. Uno de los problemas básicos ha sido las relaciones entre contenidos educativos y las evaluaciones (González & Azzollini, 2010). Debido al gran volumen de información que generan estos sistemas, se tuvo que estandarizar algunas de las áreas de la educación. Unas de las principales instituciones vinculadas a la estandarización de los sistemas es IMS (*Sistemas de Gestión Instruccional, por sus siglas en inglés*).

Actualmente este consorcio es el principal promotor y desarrollador de especificaciones abiertas orientadas a la enseñanza electrónica. Su objetivo es que, a partir de estas especificaciones, se consiga la interoperabilidad de aplicaciones y servicios en la enseñanza electrónica para que los autores de contenidos y de entornos puedan trabajar conjuntamente (Marin, 2010).

Una de sus importantes especificaciones es IMS *Question and Test Interoperability (IMS-QTI, por sus siglas en inglés)*, el objetivo de este estándar es valorar los conocimientos que los alumnos han adquirido sobre los contenidos del curso durante el mismo, y que los resultados obtenidos puedan ser compartidos a través

¹² Estandarizar: Según la International Organization for Standardization (ISO), un estándar es como una normativa según la cual se establecen unas pautas particulares destinadas a realizar una función o acción particular. Cuando decimos que un producto cumple con un estándar, estamos diciendo que cumple todas y cada una de las directrices descritas en el estándar a la hora de cumplir una función determinada (Marin, 2010).

¹³ (Fernando Garrido,2012): El e-learning o educación a distancia es una modalidad de educación a distancia, totalmente virtualizada, en la cual, mediante herramientas o aplicaciones tales como Webs, foros, audios, vídeos, chat, etc., sirven de soporte para que se cumplan los procesos de enseñanza y aprendizaje.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

de diferentes LMS. Para ello, la especificación define una estructura básica que ofrece una forma unívoca de gestionar y representar tanto las preguntas individuales, llamadas *assessment ítem*¹⁴, como las evaluaciones o exámenes completos, llamados simplemente evaluaciones. Desde que la especificación QTI fue concebida, la amplitud de las especificaciones de IMS ha crecido y trabajan en el Empaquetamiento de Contenido, Ordenamiento Simple, y más recientemente el Diseño de Aprendizaje creó la necesidad de una revisión de todas las especificaciones.

La especificación QTI v2.0 se concentró sólo en los *ítems* de Evaluación individuales y no actualizó aquellas partes de la especificación que trataban con la agrupación de *ítems* en secciones y exámenes completos o el informe de resultados. Finalmente, la publicación QTI V2.1 completa la actualización de 1.xa 2.xal reemplazar aquellas partes restantes de la especificación QTI. En esta investigación se trabajará con la versión QTI v2.0, que trata solamente de los *ítems* de evaluación individuales.

La especificación IMS-QTI v2.0 se ha diseñado para apoyar tanto la interoperabilidad y la innovación a través de la provisión de puntos de extensión bien definidos. Estos puntos de extensión se pueden utilizar para envolver los datos especializados o de propiedad de manera que permite que sea utilizado junto con los elementos que se pueden representar directamente. El trabajo que se realiza con IMS-QTI v2.0 se refiere específicamente a los proveedores de contenidos (es decir, la pregunta y los autores y editores de las pruebas), desarrolladores de creación y gestión de contenidos de herramientas, sistemas de suministro de evaluación y los sistemas de aprendizaje. Los datos modelo para la representación de contenido basado en cuestión es adecuado para la orientación a los usuarios en el aprendizaje, la educación y la formación a través de todas las edades y contextos nacionales.

1.3 Sistemas similares existentes

Guía de Observación

Conjunto de funcionalidades o atributos que determinas para realizar una búsqueda de herramientas similares con estas características:

¹⁴ Un *assessment item* es un objeto complejo, que consiste en una cuestión, junto con sus datos asociados, como la puntuación, la retroalimentación. Puede ser en un formato estándar como la especificación IMS QTI o Macromedia Flash. Los productos que se agregan en las evaluaciones, ya sea antes de la evaluación o durante el proceso de evaluación en sí. Artículos para la agregación pueden ser seleccionados manualmente, automáticamente, o por una combinación de los dos.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

- Formas de representar los algoritmos.
- Formas de extensión de las herramientas.
- Compatibilidad de las herramientas.
- Multiplataforma.

JSMaker: Es una herramienta gráfica y experimental desarrollada por el programador y diseñador web Israelí Barak Igal, que te permite crear código JavaScript¹⁵ sin tener que escribir una sola línea de código. Todo es mediante diagramas de flujo que vas uniendo en la misma herramienta (que se ejecuta en el mismo navegador). La visión de esta herramienta es crear un marco modular que permite una fácil integración con otras bibliotecas y sitios web de JavaScript. (Igal ,2011)

Visustin: Es un programa automatizado para la creación de diagramas de flujo para desarrolladores de software y escritores de documentos. Visustin usa la ingeniería inversa en su código fuente para crear diagramas de flujo o diagramas de actividad UML. Visustin lee las instrucciones *if* y *else*, los bucles y saltos y construye un diagrama de flujo de forma completamente automatizada. No requiere dibujar a mano. Visustin crea diagramas de flujo para un gran número de lenguajes de programación. (aivosto.com, 2014)

DFD: Es un editor de diagramas de flujo con el cual puedes dar forma gráfica a un gran número de algoritmos, ejecutarlos y depurarlos en caso de hallar errores. En la barra superior de DFD se agrupan los objetos necesarios para la construcción de los diagramas. Se parte siempre de una plantilla con un estado inicial y uno final, conectados por una flecha. Los *ítems* se posicionan con un clic sobre el tramo de conexión elegido: DFD se encarga de redibujar los elementos. Dependiendo del tipo de objeto, hacer doble-clic abrirá un cuadro de diálogo diferente en el cual introducir datos. Por ejemplo, en Condición se puede establecer la dirección de la condición verdadera y la expresión que debe verificarse. DFD simplifica bastante el diseño de los subprogramas. (Smart Dfd, 1998)

PSeInt: Es una herramienta para asistir a un estudiante en sus primeros pasos en programación. Mediante un simple e intuitivo pseudolenguaje en español, le permite centrar su atención en los conceptos fundamentales de la algoritmia computacional, minimizando las dificultades propias de un lenguaje y

¹⁵ JavaScript es un lenguaje interpretado en el cliente por el navegador al momento de cargarse la página, es multiplataforma, orientado a eventos con manejo de objetos, cuyo código se incluye directamente en el mismo documento HTML.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

proporcionando un entorno de trabajo con numerosas ayudas y recursos didácticos. (pseint.sourceforge.net, 2014)

En la siguiente tabla se ponen de manifiesto las características estudiadas de las herramientas similares y cuál fue el aporte a la propuesta de solución.

Tabla 2. Tabla de aportes a la propuesta de solución. Fuente: Elaboración propia

Herramientas	Características importantes	Aportes a la propuesta de solución
JSMaker	<ul style="list-style-type: none">• Usa un código de colores para diferenciar los nodos.• Representa algoritmos mediante diagramas.• Crea código JavaScript sin tener que escribir una sola línea de código.• Los módulos agrupan los componentes por categorías.• Permite la utilización de combinaciones de teclas, teclas y mouse.• Es extensible.• Tiene definida una directiva para adicionar nuevos componentes.• Funciona usando Ajax lo cual hace que la herramienta funcione más rápido.	<ul style="list-style-type: none">• Los módulos serían las componentes que tiene la herramienta y se diferenciarían por categorías.• Representaría la especificación IMS-QTI v2.0 que su lenguaje es en XML mediante un diagrama.• Crearía el código nuevamente en XML del diagrama modificado.• Debe ser extensible.• Tendría la opción de adicionar nuevos componentes para utilizar en la modificación del diagrama.
Visustin	<ul style="list-style-type: none">• Convierte el código fuente en diagramas de flujo y diagramas de actividad UML de manera automática.• Usa la ingeniería inversa en su código fuente para los diagramas.• Crea diagramas de flujo para un gran número de lenguajes de programación (JavaScript, Java, C/C++, PHP, C#).• Guarda los diagramas de flujo como archivos de imágenes o como páginas web.• Ajusta los gráficos creados automáticamente.	<ul style="list-style-type: none">• Guardará los diagramas como archivos de imágenes o el ejercicio en formato IMS-QTI v2.0.• Convertirá el código fuente del ejercicio en un diagrama de evaluación de manera automática.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA



	<ul style="list-style-type: none">• Inserta comentarios, agrega formas, modifica los vínculos y afina el diseño.	
DFD	<ul style="list-style-type: none">• Es un evaluador de expresiones, intuitivo.• Permite realizar diagramas de flujos dándole forma gráfica a diferentes algoritmos.• Tiene una opción que te permite ejecutarlos y depurarlos en caso de hallar errores.	<ul style="list-style-type: none">• Permitirá modificar el diagrama exportado por el profesor, dándole forma de árbol y no sería a todos los algoritmos, solamente a XML.
PSelnt	<ul style="list-style-type: none">• Ayuda a escribir algoritmos utilizando un pseudo-lenguaje simple, intuitivo y en español.• Además del pseudocódigo¹⁶, permite trabajar con diagramas de flujo, convirtiendo automáticamente los algoritmos, siendo posible editarlos en ambos formatos.• El objetivo general de esta herramienta es permitir al estudiante centrar la atención en los conceptos fundamentales que debe aprender, sin perder tiempo en los detalles de un lenguaje o del uso de un intérprete o compilador.	Esta herramienta no aportó ninguna funcionalidad, pero se tuvo en cuenta ya que permite escribir algoritmos con un pseudo-lenguaje simple e intuitivo para un estudiante de poco nivel en algoritmia.

1.4 Herramientas y tecnologías a utilizar

Como parte de la definición de una arquitectura de referencia para el desarrollo de aplicaciones en el Centro FORTES, se ha iniciado la formalización de marcos de trabajo en cada Línea de Productos de Software, con el propósito de disminuir la diversidad tecnológica de las soluciones, arrojando como principal resultado un nuevo marco de trabajo denominado: Xalix. Constituye en un repositorio donde se guardan componentes

¹⁶ Pseudocódigo es una descripción de alto nivel compacta e informal del principio operativo de un programa informático u otro algoritmo. Utiliza las convenciones estructurales de un lenguaje de programación real, pero está diseñado para la lectura humana en lugar de la lectura mediante máquina, y con independencia de cualquier otro lenguaje de programación.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

que al ser creados con el mismo estándar de codificación y con las mismas herramientas y tecnologías es fácil de realizar la integración entre ellos.

La aplicación tiene como objetivo integrarse en un futuro con plataformas realizadas bajo el marco de trabajo Xalix por lo que se sustentará con las principales políticas de este marco, así como aspectos de organización del trabajo dentro de la Línea de productos Ambiente Integrado de Aprendizaje (AIA). A continuación se mencionan estas tecnologías y cuál es el aporte que realizan en la herramienta a desarrollar. Para más estudio de la selección de las herramientas referirse “*Indicaciones para el trabajo en el marco de trabajo Xalix*” (Orlando Felipe Salvador Broche, Jefe de Línea de Productos Ambiente Integrado de Aprendizaje).

A continuación se definen las tecnologías del marco de trabajo Xalix de apoyo a la realización de la herramienta.

1.4.1 Framework de desarrollo

Symfony v2.3.7

Su arquitectura interna está completamente desacoplada, lo que permite reemplazar o eliminar fácilmente aquellas partes que no encajan en el proyecto. (Potencier, 2009) Symfony2 es una colección de más de veinte librerías independientes que se pueden utilizar dentro de cualquier proyecto PHP. Estas librerías, llamadas componentes de Symfony2, son bastante útiles prácticamente para cualquier herramienta. Symfony toma el *framework Doctrine* y lo incorpora dentro de sí mismo. *Doctrine 2* es un asignador objeto-relacional que proporciona persistencia transparente de objetos PHP. Se sitúa en la parte superior de una poderosa capa de abstracción de base de datos. La principal tarea de los asignadores objeto-relacional es la traducción transparente entre objetos PHP y las filas relacionales de la base de datos. *Twig* es un motor de plantillas para el lenguaje de programación PHP. Su sintaxis se origina a partir de plantillas de *Jinja* y *Django*. Symfony2 viene con un soporte integrado para *Twig* como su motor de plantilla por defecto. (Sánchez, 2007)

1.4.2 Gestor de base de datos

PostgreSQL v9.1

Es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA



mercado y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales. PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. (postgresql.org.es, 2009)

1.4.3 Librería de CSS

Bootstrap v3.0.0

Es un *framework* compuesto por archivos CSS destinados al maquetado de los sitios web con el fin de facilitar el trabajo de los desarrolladores. Se caracterizan por ser fáciles e intuitivo. También se puede decir que bootstrap no es para principiantes que están aprendiendo CSS, se tienen que tener plenos conocimientos de este tipo de lenguaje. (Vermilion, 2013)

1.4.4 Lenguajes del lado del cliente.

Hyper Text Markup Language en su versión 5.0 (HTML v5)

Es una actualización de HTML, no constituye solamente la nueva versión del lenguaje HTML sino que va más allá con nuevas características y funcionalidades que permitan mejorar las aplicaciones web y que la ejecución de las mismas en un navegador no implique la falta de facilidades para los desarrolladores, para esto se están creando APIs (Application Programming Interface) que posibilitan utilizar todos los elementos de las páginas sin necesidad de ninguna tecnología intermediaria para realizar estas acciones. (Mir, 2013)

JavaScript v1.8.5

JavaScript v1.8.5: Es un lenguaje de scripts desarrollado para incrementar las funcionalidades del lenguaje HTML. Es lenguaje interpretado, es decir, no requiere compilación. El navegador del usuario se encarga de interpretar las sentencias JavaScript contenidas en una página HTML y ejecutarlas adecuadamente. Está orientado a eventos, cuando un usuario da clic sobre un enlace o mueve el puntero sobre una imagen se produce un evento, mediante JavaScript se pueden desarrollar scripts que ejecuten acciones en respuesta a estos eventos. El modelo de objetos de JavaScript está reducido y simplificado, pero incluye los elementos necesarios para que los scripts puedan acceder a la información de una página y puedan actuar sobre la interfaz del navegador. (Nieto Pérez, 2008)

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

CSS

CSS (*Hojas de Estilo en Cascada, por sus siglas en inglés*): Es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes. Al crear una página web, se utiliza en primer lugar el lenguaje HTML/XHTML para marcar los contenidos, es decir, para designar la función de cada elemento dentro de la página: párrafo, titular, texto destacado, tabla y lista de elementos. Una vez creados los contenidos, se utiliza el lenguaje CSS para definir el aspecto de cada elemento: color, tamaño, tipo de letra del texto, separación horizontal, vertical entre los elementos y posición de cada elemento dentro de la página. (Mir, 2013)

JsPlumb v1.4.1

Es una librería JavaScript que proporciona una manera de conectar los elementos de una interfaz de usuario en un gráfico agradable.

JQuery v1.10.2

JQuery v1.10.2: es un *framework* gratuito para el lenguaje JavaScript que ofrece una Interfaz de Programación de Aplicaciones, en lo adelante API, que permite programar sin importar el navegador que utilice cada usuario, ya que funcionará de igual manera para la mayoría de los navegadores modernos. Posee una infraestructura que facilita en gran medida el desarrollo de aplicaciones complejas del lado del cliente, brindando ayuda en la creación de interfaces de usuario, efectos dinámicos, aplicaciones que utilizan Ajax, y demás; donde el tiempo de creación y depuración de todos esos componentes es muy rápido. (Jquery Community, 2013)

JQuery UI v1.10.3

JQuery UI v1.10.3: jQuery UI es una biblioteca de componentes para el *framework* jQuery que le añaden un conjunto de plug-ins y efectos visuales para la creación de aplicaciones web. Es compatible con los navegadores (y sus versiones posteriores) Internet Explorer 6.0, Mozilla Firefox 3, Safari 3.1, Ópera 9.6 y Google Chrome. JQuery UI versión 1.10 incluye docenas de correcciones de errores y mejora de la accesibilidad. JQuery UI se construye para los diseñadores y desarrolladores por igual. Se han diseñado

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

plugins para conseguir que sea lo suficientemente flexible como para evolucionar con sus necesidades y resolver una gran cantidad de casos de uso. (Jquery Community, 2013)

1.4.5 Lenguaje al lado del servidor.

PHP 5

PHP 5 (*Hypertext Pre-processor, por sus siglas en inglés*): Es un lenguaje interpretado de propósito general ampliamente usado y que está diseñado especialmente para desarrollo web y puede ser incrustado dentro de código HTML. Generalmente se ejecuta en un servidor web, tomando el código en PHP como su entrada y creando páginas web como salida. Es un lenguaje multiplataforma, completamente orientado a la web. Presenta una capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL (*Lenguaje de Consulta Estructurado, por sus siglas en inglés*) y PostgreSQL. Con capacidad de expandir su potencial utilizando una enorme cantidad de módulos. Posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda. Es libre, por lo que se presenta como una alternativa de fácil acceso para todos. Es generalmente utilizado como módulo de Apache, lo que lo hace extremadamente veloz. (php.net, 2014)

1.4.6 Lenguaje de Modelado

UML v2.5

UML es un lenguaje para modelado de propósito general evolutivo, ampliamente aplicable, dable de ser soportado por herramientas e industrialmente estandarizado. Según Lehner se aplica a una multitud de diferentes tipos de sistemas, dominios, y métodos o procesos. Posibilita la captura, comunicación y nivelación de conocimiento estratégico, táctico y operacional para facilitar el incremento de valor, aumentando la calidad, reduciendo costos y reduciendo el tiempo de presentación al mercado; manejando riesgos y siendo proactivo para el posible aumento de complejidad o cambio (2002).

Algunas de las propiedades de UML como lenguaje de modelado estándar son concurrencia, es un lenguaje distribuido y adecuado a las necesidades de conectividad actual y futura. Modela estructuras complejas. Las estructuras más importantes que soportan tienen su fundamento en las tecnologías orientadas a objetos, clase, componentes y nodos. Emplea operaciones abstractas como guía para variaciones futuras, añadiendo variables si es necesario (Rumbaugh, Jacobson y Booch, 2002).

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA



1.4.7 Metodología para el desarrollo

Básicamente, las metodologías de desarrollo de software suelen dividirse en dos grupos: metodologías pesadas (tradicionales) y metodologías ágiles. (Piattini, 2013)

Las metodologías ágiles surgen como una alternativa, o una reacción a las metodologías tradicionales y principalmente a su burocracia. Están especialmente definidas para el desarrollo de proyectos con requisitos poco definidos o cambiantes y son aplicables a equipos pequeños de trabajo que resuelven problemas concretos minimizando los fallos y los costes. (Carvajal, 2008)

Dentro de las metodologías ágiles se incluyen: Extreme Programming (XP), SCRUM, Crystal Methodologies, XBreed, Adaptive Software Development (ASD), Dynamic Systems Development Method (DSDM), entre otras. (Penadés, 2008)

Para el desarrollo de la presente investigación se analizaron las metodologías ágiles SCRUM y XP debido a que ambas promueven el trabajo en equipo, fomentan la interacción sistemática entre el cliente y el equipo de desarrollo y están suficientemente documentadas.

Programación Extrema o XP

La Programación Extrema o Extreme Programming (XP) es una metodología ágil formulada por Kent Beck en 1996 a partir de un conjunto de principios, prácticas y técnicas que facilitan de manera exitosa la finalización de proyectos, que surgen como respuesta y posible solución a los problemas derivados del constante cambio en los requerimientos. (Beck, 1996)

Una de las prácticas más significativas que posee la metodología XP es la programación en pares, que proclama que dos personas escriban código en el mismo ordenador, lo que posibilita que forma se controla cada línea de código y decisión del diseño instantáneamente. Además, la buena conexión entre ambos desarrolladores generará discusiones que sin duda conducirá a mejores estructuras y algoritmos, aumentando así la calidad del software.

Scrum

SCRUM es una metodología ágil principalmente indicada para proyectos con un rápido cambio de requisitos (Palacio, 2007), dispone de herramientas para la gestión de cada una de sus fases, y es ideal para pequeños equipos de diez o menos miembros; sin embargo no indica o provee de ninguna práctica concreta para el

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA



desarrollo de software (Carvajal, 2008). Esta metodología se basa fundamentalmente en entregas iterativas con ciclos de corta duración (24-30 días). A estos ciclos la metodología SCRUM los denomina *sprint*.

Algunos autores (Schwaber y Beedle, 2002) proponen aplicar una combinación adecuada de SCRUM y XP debido a que opinan que estas son metodologías ágiles complementarias.

Por los planteamientos mencionados anteriormente, además del ambiente de desarrollo (dígase cliente, equipo de desarrollo, tiempo de desarrollo, documentación de la herramienta), se decidió desechar la aplicación de las metodologías anteriores en el proceso de desarrollo y se opta por la aplicación de una metodología tradicional.

Metodologías tradicionales:

Según (Carvajal, 2008) las metodologías tradicionales se caracterizan por:

- Requisitos fijados a lo largo de todo el proyecto.
- Basadas en los procesos
- Procesos muy bien documentados
- Gestión predictiva de los proyectos
- No siguen ni los principios, ni las técnicas de las metodologías ágiles.

Además, estas metodologías suelen aplicarse a proyectos fundamentados con un desarrollo orientado a objetos y un equipo de desarrollo grande. (Carvajal, 2008)

Con tales características, a pesar de que han demostrado ser bastante efectivas y necesarias en proyectos de gran tamaño (respecto a tiempo y recursos), donde por lo general se obtienen resultados satisfactorios cuando el equipo de trabajo tiene amplia experiencia en su aplicación, este enfoque resulta ser el más adecuado para proyectos.

Entre las metodologías tradicionales se encuentran: MSF (*Microsoft Solution Framework, por sus siglas en inglés*), RUP (*Rational Unified Process, por sus siglas en inglés*), y sus híbridos.

Para el desarrollo de la presente investigación se analizó la metodología RUP debido a que la misma trabaja con un equipo de desarrollo grande, están documentadas mediante artefactos definidos por esta metodología y el equipo de desarrollo tiene plenos conocimientos sobre la misma.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA



RUP (Proceso Unificado de Rational).

Los autores de RUP destacan que esta metodología tiene tres características esenciales: está dirigido por los Casos de Uso, está centrado en la arquitectura, y es iterativo e incremental. Según (Kruchten, 2000) “*los Casos de Uso son una técnica de captura de requisitos que fuerza a pensar en términos de importancia para el usuario y no sólo en términos de funciones que sería bueno contemplar*”.

RUP es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de software. (...) utiliza el Lenguaje Unificado de Modelado (*Unified Modeling Language, UML*) para preparar todos los esquemas de un sistema de software. De hecho, UML es una parte esencial del Proceso Unificado. (Jacobson, Booch & Rumbaugh, 2000)

Luego del análisis anterior de las metodologías tradicionales para el desarrollo del software y teniendo en cuenta que el presente proyecto es grande, con un equipo de desarrollo grande y la disponibilidad del cliente, se decide seleccionar RUP. Debido a que esta metodología produce una gran cantidad de artefactos¹⁷, se define a continuación cuáles serán los que se generarán en todo el desarrollo de la investigación:

1. Modelo de dominio, con el objetivo de identificar y relacionar los conceptos más importantes.
2. Especificación de los requerimientos, permitiendo describir los casos de usos con el objetivo de definir las funcionalidades que tendrá la herramienta para su posterior validación.
3. Modelo de análisis como son las clases de análisis y los diagramas de colaboración, esto ayudará a que el desarrollador y el cliente entiendan desde otra perspectiva el funcionamiento del producto deseado.
4. Modelo de diseño como son las clases de diseño, diagrama de secuencia del diseño, diagrama de despliegue y modelo de datos lo cual contribuirá a mostrar la arquitectura que contiene el producto así como sus respectivas funcionalidades.
5. Modelo de implementación como son los diagramas de componentes que ayudarán a modelar las vistas de implementación de la herramienta y por último los métodos de pruebas para validar el funcionamiento del producto desarrollado dígame diseños de casos de pruebas y pruebas de caja negra.

¹⁷ Un producto o artefacto es un trozo de información que es producido, modificado o usado durante el proceso de desarrollo de *software*. Los productos son los resultados tangibles del proyecto, las cosas que va creando y usando hasta obtener el producto final

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA



1.4.8 Herramienta CASE (*Ingeniería de Software Asistida por Computadora*) para el modelado

Visual Paradigm v8.0

Propicia 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. Ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas.

Visual Paradigm proporciona una amplia utilización de características de modelado de caso, incluyendo la función completa diagrama de casos de uso y editor de flujo de eventos. Visual Paradigm *for* UML produce documentación del sistema en formato PDF, HTML. (visual-paradigm.com, 2011)

Se seleccionó la herramienta Visual Paradigm ya que tiene el diseño centrado en casos de usos y enfocado al negocio que genera un software de mayor calidad. Este tiene disponibilidad en múltiples plataformas (Windows, Linux). Soporta aplicaciones Web. Exporta los diagramas en formato de imagen.

Además, soporta el ciclo de vida completo del desarrollo de software (...). Ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. También utiliza UML como lenguaje de modelado y se integra con herramientas Java tales como: Netbeans, Eclipse y JBuilder.

1.4.9 Servidores Web

Un servidor Web es un programa que implementa el protocolo HTTP (*Protocolo de transferencia de hipertexto, por sus siglas en inglés*). Este protocolo está diseñado para transferir lo que se llama hipertextos, páginas Web o páginas HTML (*Lenguaje de marcas de hipertexto, por sus siglas en inglés*), textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproducciones de sonidos. A continuación se describen importantes características del servidor Web Apache en su versión 2.4.7. (Manuel, 2009)

Apache v2.4.7

Apache es un servidor *Hypertext Transfer Protocol (HTTP)* de código abierto que por su facilidad de configuración, robustez y estabilidad se encuentra desplegado en millones de servidores. Es multiplataforma, haciéndolo prácticamente universal, tecnología gratuita de código abierto, patentado a través de la licencia Apache, además es extensible (González-Vallés, 2014). Constituye el servidor HTTP

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

más popular y usado, lo que permite que sea fácil de adquirir y brinda una amplia fuente de ayuda. Muestra, entre otras características, mensajes de errores altamente configurables, posee bases de datos de autenticación y negociado de contenido, facilita la utilización de Perl, PHP, Java y otros lenguajes script. (apache.org, 2013)

Se seleccionó Apache v2.4.7 ya que es un servidor estable y los desarrolladores conocen el funcionamiento del mismo, además tiene muchas ventajas como:

- **Modular:** La arquitectura del servidor Apache consta de diversos módulos que aportan muchas de las funcionalidades que podrían considerarse básicas para un servidor web.
- **Código abierto:** La Licencia Apache permite la distribución de derivados de código abierto y cerrado a partir de su código fuente original.
- **Multiplataforma:** Es usado en varias plataformas, tanto, libres como propietarias.
- **Extensible:** Se pueden añadir módulos y escribir sus propios códigos adaptándolos a las funciones del servidor Apache, de forma tal, que se ajusten a las necesidades de los desarrolladores.
- **Popular:** Es fácil conseguir ayuda y soporte. Apache tiene amplia aceptación en la red desde 1996 y es el servidor HTTP más usado. Alcanzó su máxima cuota de mercado en el 2005 siendo el servidor empleado en el 70% de los sitios web en el mundo.
- **Nueva interfaz de programación (API) de Apache.** La API para los módulos ha cambiado significativamente en la nueva versión. Muchos de los problemas de ordenación y prioridad de módulos de la versión 1.3 deben haber desaparecido. Apache 2.0 hace automáticamente mucho de lo que es necesario, y la ordenación de módulos se hace ahora por *hooks*, lo que ofrece una mayor flexibilidad. También se han añadido nuevas llamadas que ofrecen capacidades adicionales sin tener que parchear el núcleo del servidor Apache. (apache.org, 2013)

1.4.10 Entorno de Desarrollo Integrado (IDE)

Un Entorno de Desarrollo Integrado o IDE es un conjunto de programas que se ejecutan desde una única interfaz de usuario. Es un entorno de programación, que consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica que además pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. (Maldonado, 2007)

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

NetBeans IDE v7.4

El NetBeans IDE es el entorno de desarrollo utilizado para dar solución a las funcionalidades del sistema. Es una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el NetBeans IDE. Es un producto libre y gratuito sin restricciones de uso. (netbeans.org, 2012)

Se seleccionó NetBeans IDE como entorno de desarrollo, ya que los desarrolladores tienen pleno conocimiento acerca de las funciones del mismo, además por las siguientes características definidas por (Maldonado, 2007):

- Creación de proyectos PHP: NetBeans provee a los desarrolladores de una estructura para los proyectos que se puedan crear junto a este IDE. Propone un esqueleto para organizar el código fuente, el editor conjuntamente integra los lenguajes como HTML, JavaScript y CSS. Además, NetBeans posee un sistema para examinar todos los directorios de cada proyecto, haciendo reconocimiento y carga de clases, métodos y objetos, para acelerar la programación.
- Editor de código fuente: Mejora en su editor, sobre todo en el editor de PHP, es mucho más rápido y a la vez robusto, contiene más ayuda en línea.
- Depuración de PHP: NetBeans integra muy bien la utilización Xdebug, permitiendo inspeccionar y examinar cada variable local, establecer puntos de interrupción y evaluar el código en nuestra lógica.
- Integración con sistemas de control de versiones: Esta es una de las condiciones necesarias para los proyectos y es la posibilidad de contar con la integración de sistemas de control de versiones, tales como SVN, CVS, Mercurial y Git. Desde el editor es posible realizar la administración de estos sistemas versionados, sus commit, branch, importar, exportar, revert, clonar, entre otras funciones.

1.5 Conclusiones parciales

Después de un análisis del estado del arte, se puede concluir que la evaluación es fundamental en cualquier proceso educacional, la misma no puede estar estandarizada, debido a que los estudiantes no tienen el mismo nivel de aprendizaje y tienen preferencias individuales, lo que los hace únicos a cada uno por separado. Además, que los profesores deberían tener en cuenta todos estos aspectos y usarlos como

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

ventajas en el proceso de evaluación, permitiendo así el mejoramiento de cada estudiante, según la estrategia definida para él.

Las herramientas estudiadas en el capítulo fueron seleccionadas mediante una guía de observación; una de las herramientas fue JSMaker, la cual tiene un gran desempeño en la generación de diagramas, ya que su forma de representar los algoritmos es muy sorprendente, además tiene directivas para adicionar otros componentes que contenga el lenguaje, también separa los componentes por categoría y representa los nodos por colores.

Existe una gran diversidad de tecnologías que actualmente ayudan a realizar una herramienta como la anteriormente descrita y que funcione para la educación electrónica. El estudio de estas herramientas sirvió para identificar las tecnologías necesarias en el desarrollo del producto deseado. Una de las tecnologías es JsPlumb la cual propiciará la conexión de elementos en un gráfico amigable, Symfony 2 será el *framework* utilizado para el desarrollo y se utilizará la metodología RUP para el orden y generación de los artefactos necesarios en la investigación, además de guiar el proceso de desarrollo del *software*.

CAPÍTULO 2. CARACTERÍSTICAS DEL SISTEMA

CAPÍTULO 2. CARACTERÍSTICAS DEL SISTEMA

En esta etapa de la investigación se realiza la correcta definición de los procesos que intervienen en la realización del sistema que se desea desarrollar. Pues de un buen funcionamiento de estos procesos dependerá todo el desarrollo de la herramienta.

2.1 Modelo de dominio

Un modelo del dominio se utiliza con frecuencia como fuente de inspiración para el diseño de los objetos software, y será una entrada necesaria para varios de los siguientes artefactos que se van generando en la investigación. 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. (Larman, 2003)

2.1.1 Conceptos del dominio

LMS: Es la plataforma educativa la cual se desarrolló con las bases del marco de trabajo Xalix donde el profesor realiza el examen o ejercicio.

Motor de Evaluación: Es un motor de evaluación el cual es utilizado por la plataforma y se encarga de general el resultado de los *test* emitiendo una nota y una retroalimentación de cada ejercicio.

Ejercicios: Son preguntas formuladas las cuales se generan en las plataformas educativas por un profesor y contienen algoritmos.

IMS-QTI v2.0: Es una especificación de IMS y tiene un formato a seguir que hereda del lenguaje XML.

Algoritmo_Inicial: Es un conjunto de pasos claramente definidos que partir de una cierta entrada (*input*), produce una determinada salida (*output*). Este se genera en la plataforma educativa al profesor crear el ejercicio y debe contener un formato IMS-QTI v2.0.

PERA (IMS-QTI v2.0): Es la herramienta integrada a la plataforma y que interactúa con el profesor, esta herramienta debe generar un diagrama de la evaluación del ejercicio IMS-QTI v2.0. También debe exportar un algoritmo de este mismo formato para que posteriormente se interprete con el motor de evaluación de las plataformas educativas.

CAPÍTULO 2. CARACTERÍSTICAS DEL SISTEMA

Profesor: Es la persona que utiliza la herramienta PERA (IMS-QTI v2.0) para realizar modificaciones en el diagrama generado por el algoritmo inicial que conlleven a personalizar las evaluaciones de un determinado ejercicio.

Diagrama: Es la representación gráfica en la que se muestran las relaciones que existen en un determinado algoritmo el cual es generado por un ejercicio.

Algoritmo_Final: Es la modificación del diagrama que contienen la herramienta PERA (IMS-QTI v2.0) en el cual se genera un algoritmo final que debe contener un ejercicio basado en el formato IMS-QTI v2.0 y que es interpretado por el motor de evaluación.

2.1.2 Diagrama de modelo de dominio

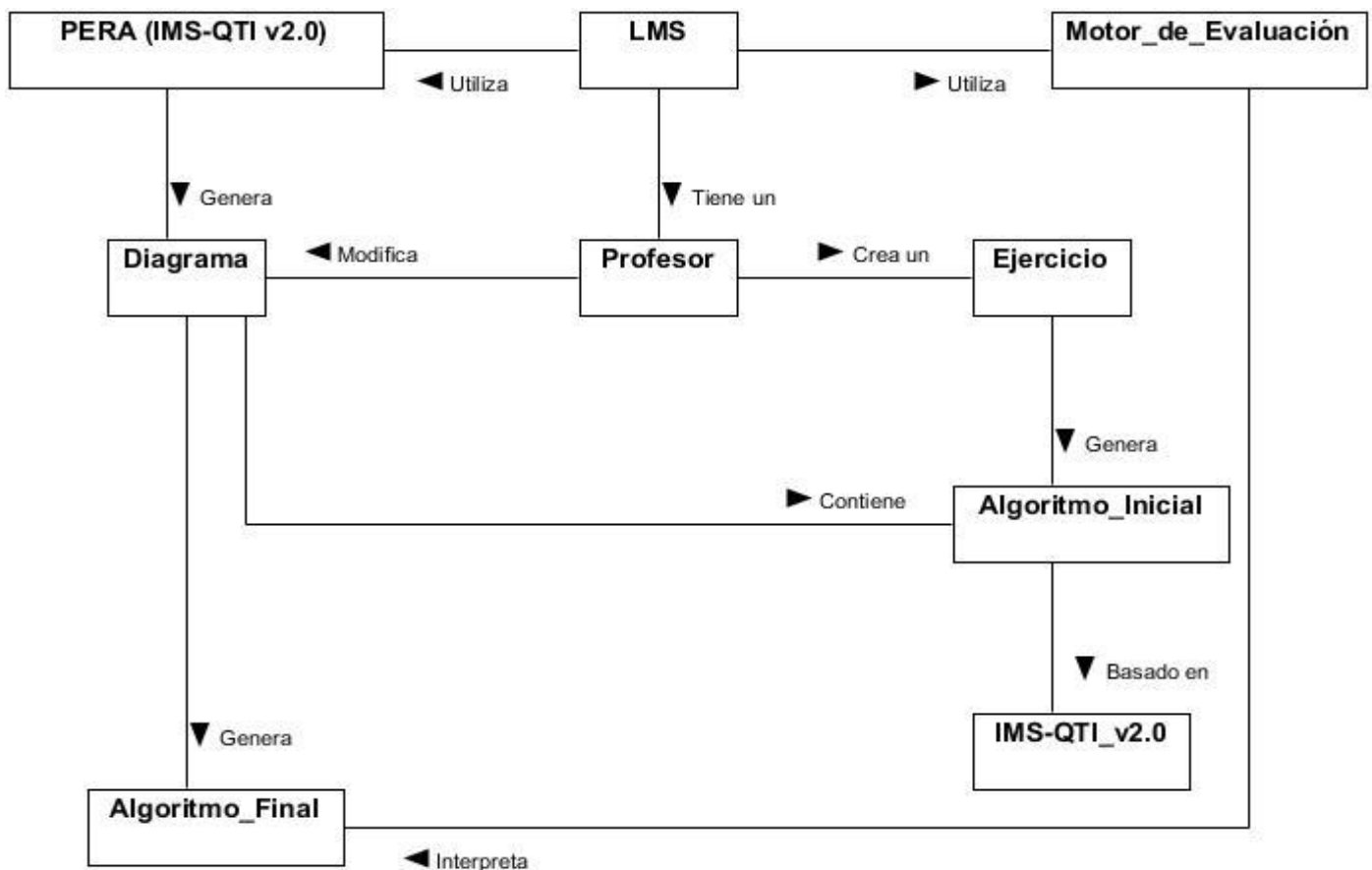


Figura 1. Diagrama de Modelo de Dominio.

CAPÍTULO 2. CARACTERÍSTICAS DEL SISTEMA

2.1.3 Descripción de la herramienta propuesta

PERA (IMS-QTI v2.0) hasta el momento debe ser independiente y tiene como objetivo principal que el profesor desempeñe un papel importante en la realización del algoritmo que emite la forma en que se va a evaluar y retroalimentar un determinado ejercicio. Los profesores son los que crean los ejercicios, estos ejercicios generan en su interior un algoritmo el cual se llama “Algoritmo_Inicial” el mismo tiene que estar basado en el formato IMS-QTI v2.0 para que la herramienta pueda interpretarlo y generar a su vez el diagrama de evaluación del algoritmo. El profesor podrá modificar ese diagrama mediante la herramienta PERA. Finalmente, la herramienta genera el código de las modificaciones que hace el profesor en el diagrama de evaluación, este código el cual se llama “Algoritmo_Final” ya con sus modificaciones hechas, es interpretado por el motor de evaluación el cual emite una nota y retroalimentación personalizada.

2.2 Requerimientos del *software*

Este es uno de los flujos de trabajo más importantes, ya que en él se establece qué debe hacer exactamente la herramienta que se desarrolle. Los requisitos son como un contrato donde se debe cumplir todas las condiciones que se definan, de modo que los usuarios finales tienen que comprender y aceptar los requisitos que se especifiquen. Los requisitos tienen como principal objetivo guiar el desarrollo hacia un sistema correcto. Este propósito es conseguido por medio de los requisitos funcionales y los no funcionales que presente el *software* a desarrollarse. (Wesley, 1999). A continuación se listan los requisitos funcionales y los requisitos no funcionales de la herramienta.

2.2.1 Requerimientos funcionales

Los requerimientos funcionales (RF) son declaraciones de los servicios que debe proporcionar la herramienta, de manera en que este debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares (Sommerville, 2005). Es decir representan la funcionalidad del sistema.

RF1 Agregar componentes al sistema.

RF2 Eliminar componentes del sistema.

RF3 Activar y desactivar componentes a utilizar.

RF4 Ver información de un componente.

RF5 Mostrar componentes en una barra de menú por categorías.

RF6 Permitir recuperar los datos borrados por el usuario con Deshacer y Rehacer.

CAPÍTULO 2. CARACTERÍSTICAS DEL SISTEMA

RF7 Importar ejercicio en formato IMS-QTI v2.0.

RF8 Validar que el ejercicio que se importe sea de formato IMS-QTI v2.0.

RF9 Representar un algoritmo de IMS-QTI v2.0 en un diagrama.

RF10 Modificar diagrama.

RF11 Exportar diagrama en formato IMS-QTI v2.0.

RF12 Exportar diagrama en formato JPG.

RF13 Funcionar como una herramienta independiente.

2.2.2 Requisitos no funcionales

Los requerimientos no funcionales (RNF) son aquellos que no se refieren directamente a las funciones específicas que proporciona la herramienta, sino a las propiedades emergentes de este como la fiabilidad, el tiempo de respuesta y capacidad de almacenamiento (Sommerville, 2005). Es decir representan aquellos atributos que debe exhibir la herramienta, pero que no son una funcionalidad específica.

RNF Soporte:

La solución propuesta debe tener las tecnologías establecidas por el marco de trabajo Xalix para su futura integración con plataformas realizadas bajo este marco.

RNF Hardware:

La solución propuesta deberá funcionar cuando esta acoplada a plataformas educativas realizadas bajo el marco de trabajo Xalix, con las características que tenga el sistema educativo.

Además, deberá funcionar cuando sea una herramienta independiente con las características que se evidencian a continuación:

Navegador: Google Chrome.

Procesador Pentium: Intel, Pentium 4 y posterior.

1 GB de RAM o superior.

20 GB de espacio en disco duro.

RNF Diseño:

Debe contener un diseño con una arquitectura basada en componentes y un menú de componentes legible.

CAPÍTULO 2. CARACTERÍSTICAS DEL SISTEMA

RNF Usabilidad:

La aplicación deberá contar con una interfaz y navegación funcional, accesible para los profesores.

RNF Portabilidad:

El sistema será multiplataforma.

2.3 Patrones de caso de uso

En esta sección se hace referencia a todos los patrones de caso de uso utilizados que ayudaron a confeccionar el diagrama de CU del sistema.

- *CRUD (Creating, Reading, Updating, Deleting)*: este patrón se basa en la fusión de caso de usos simples para formar una unidad conceptual. Se puede apreciar el uso de este en el caso uso Gestionar Componente.
- *Múltiples actores*: el uso de este patrón se evidencia específicamente en la variante roles comunes, la cual establece que varios actores jueguen el mismo rol sobre el caso uso.

2.4 Modelo de casos de uso de la herramienta

En esta sección se identifican los actores del sistema, se realiza una breve descripción de dichos actores y su diagrama correspondiente, además se incluye el diagrama de caso de uso del sistema y la descripción de los casos de uso identificados.

2.4.1 Descripción de los actores del sistema

Los actores identificados son las personas encargadas de inicializar e interactuar con los casos de uso del sistema.

Tabla 3. Descripción de los Actores del sistema.

Actores	Descripciones
Profesor	Actor que interactúa con la herramienta con el objetivo de personalizar las evaluaciones generadas por el algoritmo de un ejercicio.
Profesor superior	Actor que tiene permiso para agregar nuevos componentes al sistema necesarios para los profesores y también puede utilizar la herramienta para personalizar las evaluaciones generadas por el algoritmo de un ejercicio.

CAPÍTULO 2. CARACTERÍSTICAS DEL SISTEMA

2.4.2 Diagrama de casos de usos de la herramienta

El diagrama de casos de uso del sistema muestra las relaciones que existen entre actores y casos de uso. A continuación se muestra el Diagrama de casos de uso del sistema que contiene los casos de uso generados a partir de los requisitos identificados anteriormente, así como las relaciones que se establecen entre ellos. Los casos de usos (CU) identificados son:

- Gestionar componentes.
- Modificar diagrama de evaluación de ejercicio.
- Exportar ejercicio o diagrama de evaluación.
- Importar ejercicio.

Casos de usos críticos en el sistema:

- Generar diagrama de evaluación de ejercicio.
- Generar código del diagrama de evaluación de ejercicio.

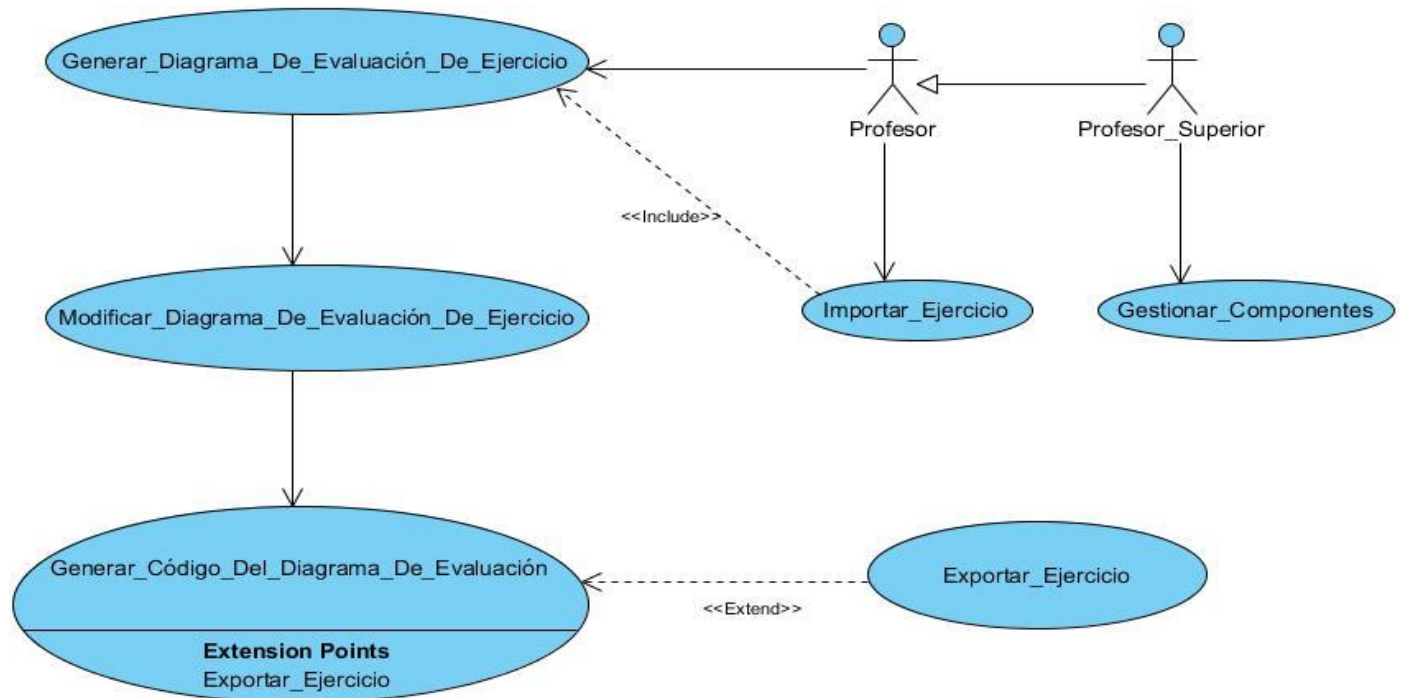


Figura 2. Diagrama de casos de usos de la herramienta.

CAPÍTULO 2. CARACTERÍSTICAS DEL SISTEMA

2.4.3 Descripción de los casos de usos de la herramienta

A continuación se presenta la descripción de un caso de uso, para el estudio de las restantes descripciones remitirse al **Anexo 2**.

CU 1. Gestionar componentes:

Tabla 4. Descripción del CU Gestionar componentes

Objetivo	El objetivo que persigue el actor con este caso de uso es activar, desactivar, eliminar, agregar y ver información de un componente en la herramienta.	
Actores	Profesor Superior: (Inicia) selecciona la opción componentes.	
Resumen	El caso de uso permite actualizar el <i>software</i> realizando acciones como activar, desactivar, eliminar, ver información y agregar componentes al sistema.	
Complejidad	Alta	
Prioridad	Opcional	
Precondiciones	Debe tener los permisos para la gestión de los componentes.	
Postcondiciones	Se activó, desactivó, eliminó, vio información y agregó componentes al sistema.	
Flujo de eventos		
Flujo básico		
	Actor	Sistema
1.	El caso de uso se inicia cuando el actor accede a la opción de componentes del sistema.	
2.		Muestra una interfaz con una lista de componentes en el sistema y brinda la posibilidad de realizar las siguientes acciones: <ul style="list-style-type: none">• Agregar componentes al sistema.• Activar/Desactivar componentes. Ver Sección 1: " <u>Activar/Desactivar componentes</u> ". <ul style="list-style-type: none">• Eliminar componentes. Ver Sección 2: " <u>Eliminar componentes</u> ". <ul style="list-style-type: none">• Ver información de un componente. Ver Sección 3: " <u>Ver información de un componente</u> ". <ul style="list-style-type: none">• Salir del área de componentes.
3.	Selecciona la opción subir componentes.	

CAPÍTULO 2. CARACTERÍSTICAS DEL SISTEMA



4.		Muestra una pestaña que te permite buscar los componentes que desea agregar. Permite realizar las acciones: <ul style="list-style-type: none"> • Subir • Cancelar
5.	Selecciona la opción examinar para buscar el componente que desea agregar.	
6.		Muestra la información de su PC. Permite realizar las acciones: <ul style="list-style-type: none"> • Abrir • Cancelar
7.	Selecciona el componente, la opción abrir y la acción subir.	
8.		Verifica que sea un fichero .zip.
9.		Si es fichero .zip. Verifica que tenga la estructura definida para los componentes.
10.		Si tiene la misma estructura: Verifica que no se encuentre en la base de datos de componentes.
11.		Si no está en la base de datos: Agrega el componente en la base de datos de componentes y la lista de componentes del sistema.
12.		Termina el caso de uso.

Flujos alternos

9. a No es un fichero .zip.

	Actor	Sistema
1.		Muestra un mensaje de ERROR: "La herramienta no admite este tipo de formato, verifique que sea .zip".
2.		Regresa al paso 2 del flujo de trabajo.

Flujos alternos

10. a No tiene la estructura definida de los componentes.

CAPÍTULO 2. CARACTERÍSTICAS DEL SISTEMA



	Actor	Sistema
1.		Muestra un mensaje de ERROR: "No es un componente, verifique nuevamente".
2.		Regresa al paso 2 del flujo de trabajo.
Flujos alternos		
12. a Esta en la base de datos.		
	Actor	Sistema
1.		Muestra un mensaje de ERROR: "El componente que desea adicionar ya existe en la base de datos".
2.		Regresa al paso 2 del flujo de trabajo.
Flujos alternos		
*.a Selecciona la opción cancelar petición.		
	Actor	Sistema
1.		Regresa al paso 2 del flujo de trabajo.
Flujos alternos		
*.a Selecciona la opción salir del área de gestionar componentes.		
	Actor	Sistema
1.		Termina el caso de uso.
Relaciones	CU incluidos	No tiene
	CU extendidos	No tiene
Requisitos no funcionales	-	
Asuntos pendientes	-	

Sección 1: Activar/Desactivar componentes.

Flujo básico: Activar/Desactivar componente.		
	Actor	Sistema
1.		Brinda la posibilidad de realizar las siguientes acciones: <ul style="list-style-type: none"> • Activar componentes • Desactivar componentes

CAPÍTULO 2. CARACTERÍSTICAS DEL SISTEMA



2.	Activa los componentes deseados mediante la selección de los mismos	
3.		Muestra los componentes en el panel de componentes.
4.		Termina el caso de uso.

Flujos alternos

2. a Desactiva los componentes deseados mediante la selección de los mismos

	Actor	Sistema
1.		Muestra los componentes en el panel de componentes.
2.		Regresa al paso 1 del flujo de trabajo.

Sección 2: Eliminar componentes.

Flujo básico: Eliminar componente.

	Actor	Sistema
1.	El actor selecciona opción eliminar componente.	
2.		Muestra el siguiente mensaje para validar la decisión: "Esta seguro que desea eliminar el componente". Permite realizar las acciones: <ul style="list-style-type: none"> • Aceptar • Cancelar
3.	Selecciona "Aceptar"	
4.		Muestra un mensaje de validación: "Se eliminó correctamente el componente".
5.		Elimina el componente de la base de datos de componentes y del sistema.
6.		Termina el caso de uso.

Flujos alternos

3. a Selecciona "Cancelar"

	Actor	Sistema
1.		Termina el caso de uso.

CAPÍTULO 2. CARACTERÍSTICAS DEL SISTEMA



Sección 3: Ver información de un componente.

Flujo básico: Ver información de un componente.	
Actor	Sistema
1.	Selecciona el componente y permite acceder a la información del mismo.
2.	Muestra los siguientes datos de un componente: <ul style="list-style-type: none">• Versión• Descripción• Dependencia• Nombre Permite la opción de Cerrar.
3.	Termina el caso de uso

2.5 Conclusiones parciales

PERA (IMS-QTI v2.0) será creada para apoyar el proceso de evaluación de los Sistemas de Gestión de Aprendizaje y ayudar a que el profesor ocupe un papel importante en el proceso. La misma podrá ser usada para modificar la evaluación de ejercicios creados bajo la especificación IMS-QTI v2.0, y será independiente hasta que se desarrolle alguna plataforma educativa creada bajo el marco de trabajo Xalix que desee integrarla como componente.

Para que se asegure toda la información antes expuesta quedaron de forma estructura los principales conceptos y relaciones en el modelo de dominio, además se capturaron 13 requisitos funcionales y 5 no funcionales, se confeccionó el diagrama de casos de usos del sistema y por último se describieron los casos de uso, dando paso al capítulo de análisis y diseño de la investigación.

CAPÍTULO 3. ANÁLISIS Y DISEÑO DE LA PROPUESTA



CAPÍTULO 3. ANÁLISIS Y DISEÑO DE LA PROPUESTA

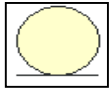
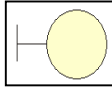

Este es uno de los capítulos más importantes de la investigación producto a que se aborda el tema de análisis y diseño, este es el tercer flujo de trabajo que propone la metodología RUP en el cual se generan diversos artefactos con el fin de guiar el desarrollo de la solución del sistema. El mismo tiene como objetivo principal transformar los requisitos en el diseño del sistema en creación.

3.1 Modelo de análisis

El modelo de análisis es una estructura de clases y paquetes estereotipados que proporciona la idea de cómo llevar a cabo la funcionalidad del sistema, es decir, un modelo de objetos conceptuales, que ayuda a refinar los requisitos y permite razonar sobre los aspectos internos del sistema. Se centra en los requisitos funcionales por lo que su objetivo fundamental es ver el funcionamiento del sistema (Wesley, 1999). En él, se analizan y refinan los requerimientos que fueron descritos en el diagrama de casos de uso para alcanzar una visión detallada de los mismos.

A continuación se explican los tres estereotipos básicos que se utilizan:

Tabla 5. Descripción de los estereotipos utilizados en los diagramas de clases de análisis.

Clases	Estereotipos	Descripción
Entidad		<ul style="list-style-type: none">• Modelan la información del Sistema.• Modelan el comportamiento asociado a una información.
Interfaz		<ul style="list-style-type: none">• Modelan la interacción Actor – Sistema.• Ventanas, Formularios, comunicación con otros sistemas o dispositivos.
Control		<ul style="list-style-type: none">• Coordinan el trabajo de las clases.• Encapsulan comportamiento de un CU.• Funciones complejas.

3.1.1 Diagramas de clases de análisis

A continuación se muestra el diagrama de análisis del CU Gestionar Componentes, para el estudios de los restantes diagramas remitirse al **Anexo 3.**

CAPÍTULO 3. ANÁLISIS Y DISEÑO DE LA PROPUESTA



Figura 3. Diagrama de clases de análisis: Gestionar Componentes.

3.1.2 Diagramas de colaboración

Los diagramas de colaboración describen las interacciones entre los objetos en un formato de grafo o red (Larman, 1999). A continuación se muestran los diagramas de colaboración por secciones del CU Gestionar componentes. Para estudio de los demás diagramas remitirse al **Anexo 4**.

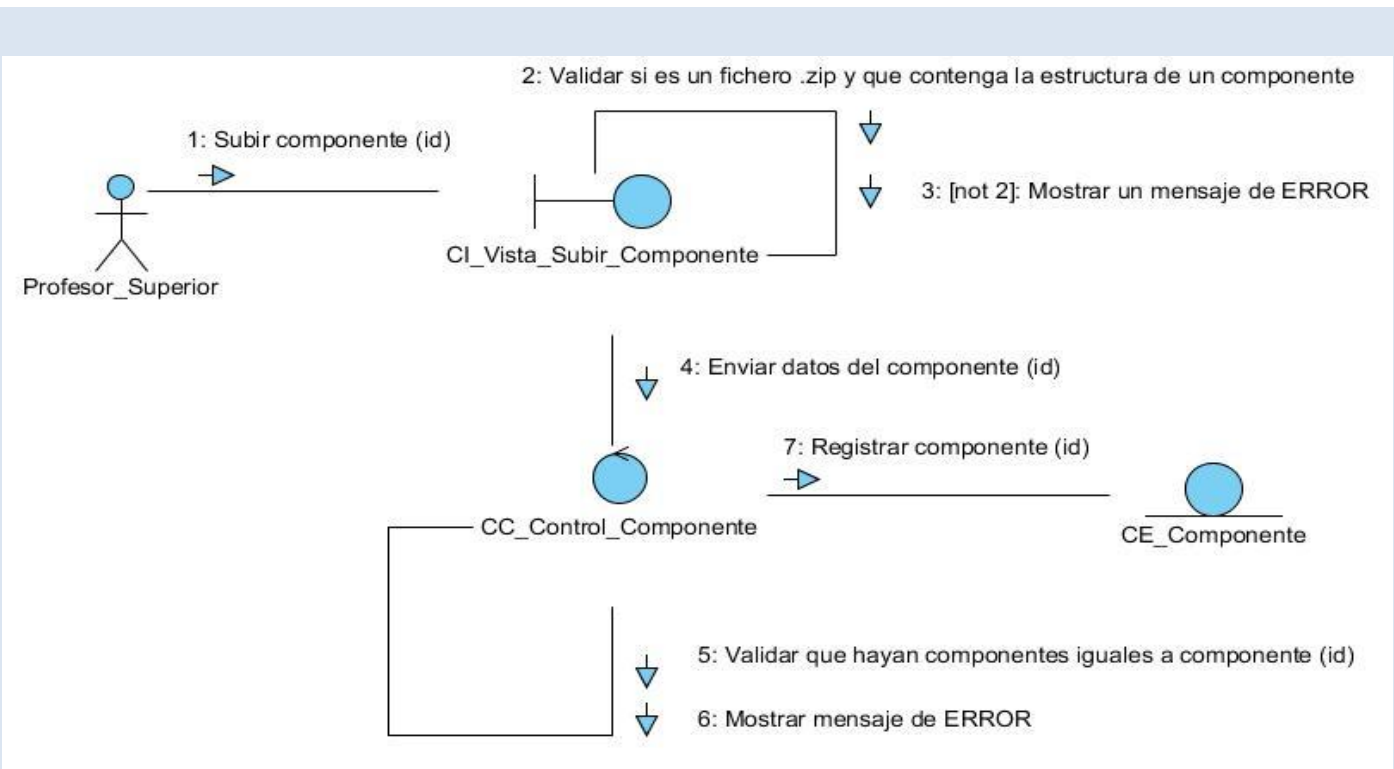


Figura 4. Diagramas de Colaboración Gestionar Componentes: Sección Agregar Componentes

CAPÍTULO 3. ANÁLISIS Y DISEÑO DE LA PROPUESTA



3.2 Patrón arquitectónico

Modelo – Vista – Controlador en Symfony2

Symfony es un *framework* PHP que permite fácilmente utilizar la arquitectura Modelo-Vista-Controlador (en lo adelante MVC). Este patrón separa en tres niveles las funcionalidades de una herramienta con el objetivo de aumentar la usabilidad de las mismas. Por lo que separa en tres capas los datos (modelo), la interfaz de usuario (vista) y la lógica de negocio (controlador). Esto permite una mayor reutilización de componentes, facilita las modificaciones del aspecto del programa/web sin tener que tocar el funcionamiento y simplifica el mantenimiento. Funciona de forma “triangular”, es decir, cada capa envía información a una y la recibe de la otra. (Symfony.es, 2009)

Los niveles en los que se dividen son:

- **Modelo:** Es el responsable de la conexión a la base de datos y la manipulación de los datos mismos. Esta capa está pensada para trabajar con los datos como así también obtenerlos, pero no mostrarlos, ya que la capa de presentación de datos es la vista.
- **Vista:** Todo lo que se refiera a la visualización de la información, el diseño, colores, estilos y la estructura visual en sí de nuestras páginas.
- **Controlador:** Su responsabilidad es procesar y mostrar los datos obtenidos por el Modelado. Es decir, este último trabaja de intermediario entre los otros dos, encargándose también de la lógica de negocio.

3.2.1 Aplicación de los patrones de diseño en Symfony2

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular. Identifica clases, instancias, roles, colaboraciones y la distribución de responsabilidades. Un *framework* simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. (Symfony.es, 2009)

Inyección de dependencia en Symfony2

Este patrón se utilizará en la arquitectura de los componentes particularmente en la configuración de estos para así modificar la menor cantidad de código posible.

CAPÍTULO 3. ANÁLISIS Y DISEÑO DE LA PROPUESTA

Patrones:

Patrones Grasp:

- **Experto:** Este patrón se evidenciará en la clase manager XML la cuál es la encargada de manipular los XML subidos al sistema; realizando principalmente su transformación a arreglos en php.
- **Creador:** Se utiliza en métodos de los controladores donde se crean instancias de clases del modelo para capturar datos insertados por el usuario y luego gestionar estos en la base de datos.
- **Bajo Acoplamiento:** Este patrón plantea que la dependencia entre las clases que conforman la arquitectura del sistema debe ser mínima, debido a que a la hora de realizar modificaciones en una clase, no afecte la estructura de las restantes.
- **Alta Cohesión:** Permite asignar responsabilidades con una alta cohesión, por ejemplo la clase *Actions* tiene la responsabilidad de definir las acciones para las plantillas y colabora con otras para realizar diferentes operaciones, instanciar objetos y acceder a las propiedades, es decir, está formada por diferentes funcionalidades que se encuentran estrechamente relacionadas proporcionando que el *software* sea flexible frente a grandes cambios.
- **Controlador:** El *framework* Symfony2 incluye un *bundle*, denominado *FrameworkBundle*, el cual presenta una serie de clases controladoras, encargadas de las principales operaciones dentro del funcionamiento de Symfony2. Además, se crean aquellas clases controladoras que darán solución a los principales requerimientos pertenecientes al flujo del negocio en cuestión, facilitando la centralización de las actividades.

Patrones GOF

- **Strategy:** Este patrón plantea la utilización de varias estrategias para representar comportamientos distintos, utilizando una misma interfaz. Estos comportamientos pueden ser modificados e intercambiados en la aplicación, sin que afecte la lógica del sistema.
- **Decorator:** Este patrón plantea la utilización de una o varias plantillas globales, que guarden el código que es usual para todo el sistema, para no tener que repetirlo en cada interfaz. En este caso se representan en la plantilla **base.html.twig**; las cuales establecen el maquetado para las plantillas del sistema.

CAPÍTULO 3. ANÁLISIS Y DISEÑO DE LA PROPUESTA

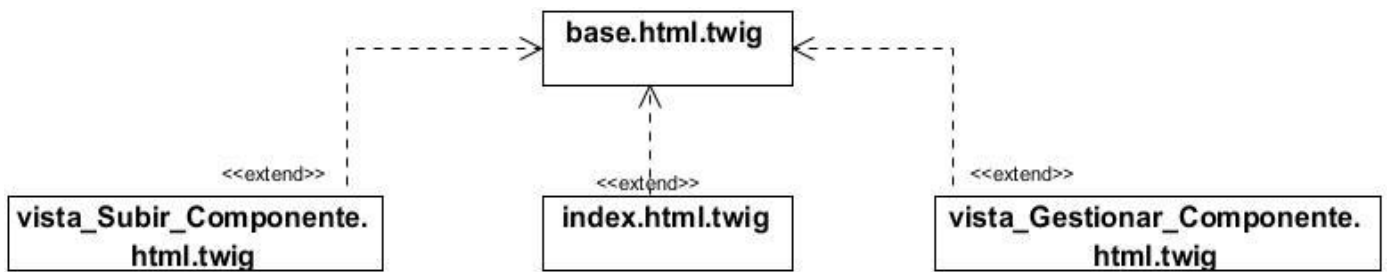


Figura 5. Diagrama de plantillas globales utilizadas.

3.3 Modelo de diseño

Es un modelo de objetos que describe la realización física de los CU, centrándose en cómo los RF y los RNF tienen impacto en la aplicación a desarrollar. Facilita la abstracción de la reconstrucción del módulo y es de ese modo, el artefacto fundamental de entrada de las actividades de implementación.

CAPÍTULO 3. ANÁLISIS Y DISEÑO DE LA PROPUESTA

3.3.1 Diagrama de clases del diseño

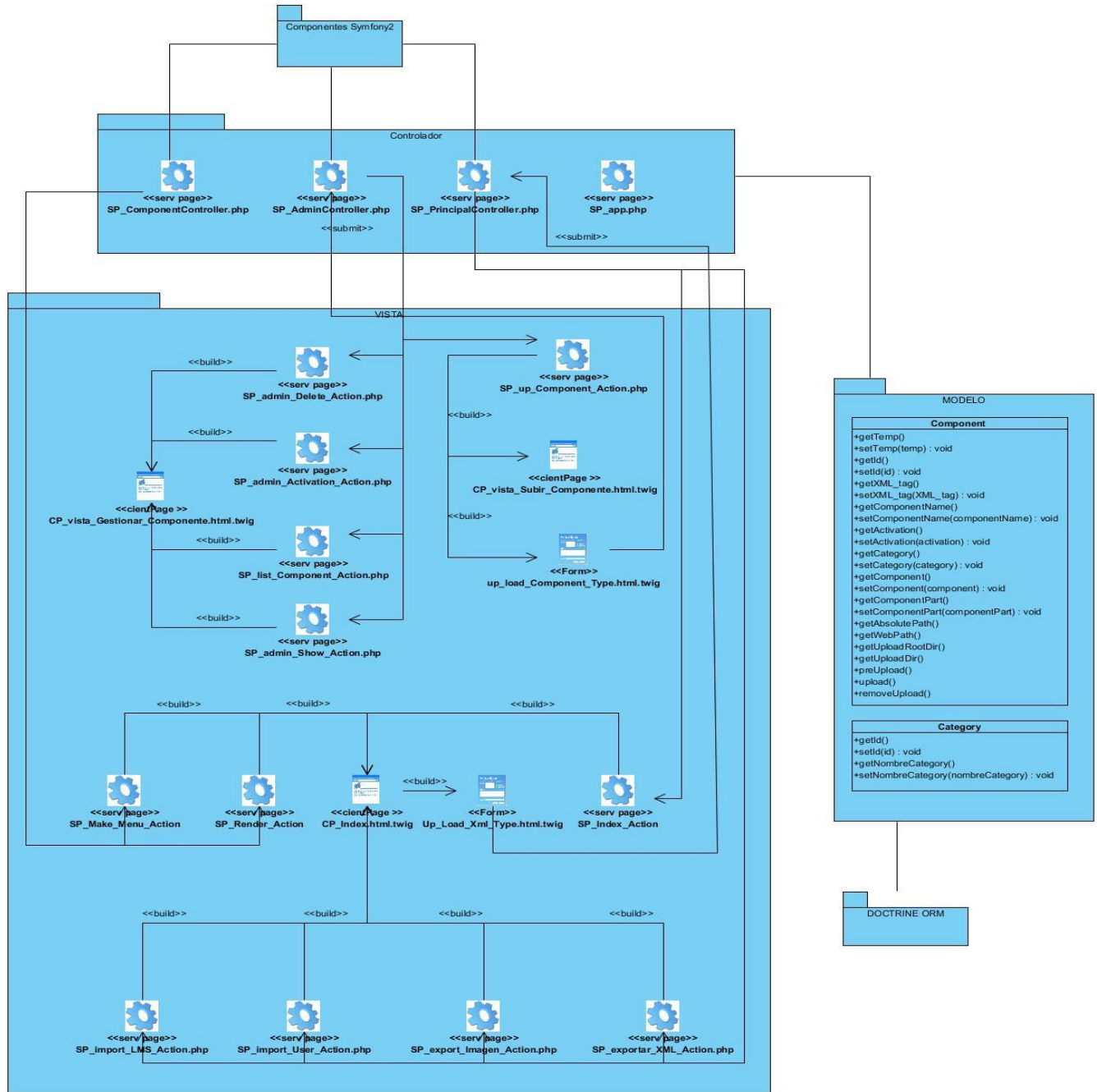


Figura 6. Diagrama de clase de diseño del sistema.

CAPÍTULO 3. ANÁLISIS Y DISEÑO DE LA PROPUESTA

3.3.2 Diagrama de secuencia del diseño

Se muestra una representación del diagrama de secuencia del diseño del CU Gestionar Componentes. Para estudio de los restantes diagramas remitirse al **Anexo 5**.

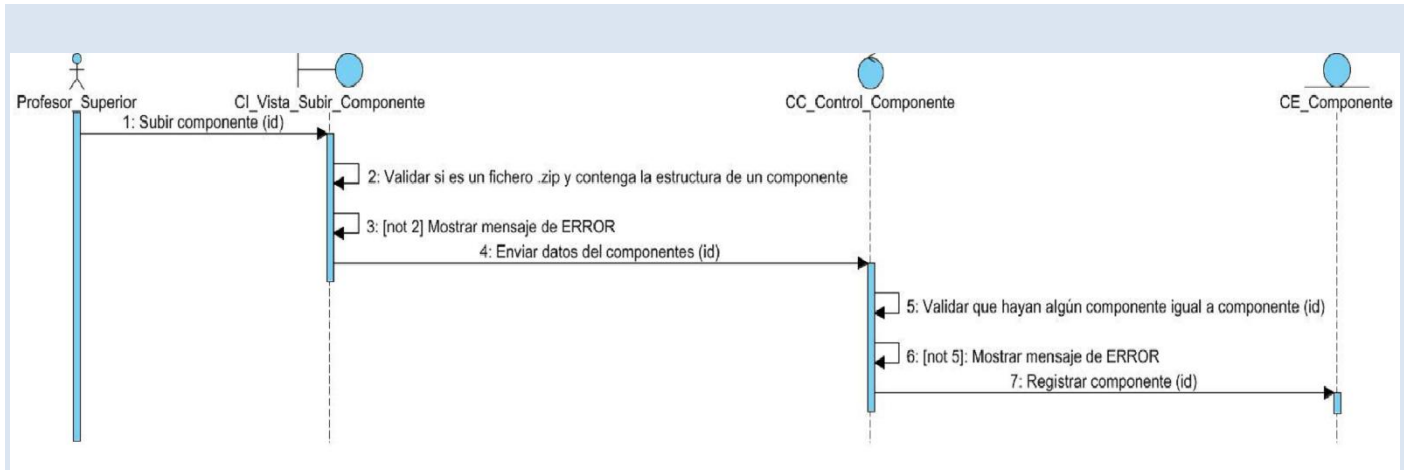


Figura 7. Diagrama de secuencia del CU Gestionar Componente: Sección Agregar Componente

3.3.3 Diagrama de despliegue



Figura 8. Diagrama de Despliegue

CAPÍTULO 3. ANÁLISIS Y DISEÑO DE LA PROPUESTA

3.3.4 Diseño de la base de datos

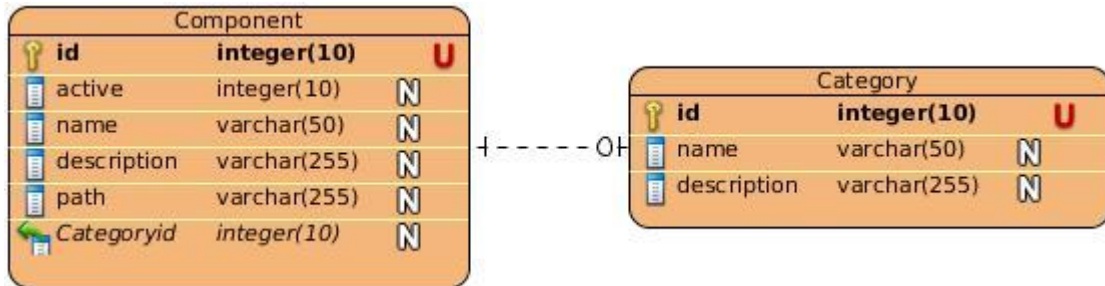


Figura 9. Diseño de la Base de Datos

Descripción de las tablas

Tabla 6. Descripción de la tabla de Base de Datos *Component*

Component: Almacena los componentes incorporados a la aplicación, su estado y configuración.	
Atributos	Descripción
<i>id</i>	Llave única que identifica las categorías
<i>activate</i>	Indica si el componente puede ser utilizado dentro de la aplicación
<i>name</i>	Contiene el nombre representativo del componente
<i>description</i>	Describe la utilidad y características del componente
<i>path</i>	Almacena la ubicación del componente dentro del directorio de archivos de la aplicación
<i>Categoryid</i>	Llave foránea de la tabla categoría que representa a que categoría pertenece el componente

Tabla 7. Descripción de la tabla de Base de Datos *Category*

Category: Contiene las categorías de agrupación de los componentes dentro de la aplicación.	
Atributos	Descripción
<i>id</i>	Llave única que identifica las categorías
<i>name</i>	Contiene el nombre representativo de la categoría
<i>description</i>	Brinda información básica sobre las características de cada categoría

CAPÍTULO 3. ANÁLISIS Y DISEÑO DE LA PROPUESTA



3.4 Conclusiones parciales

Se utilizó el patrón Modelo-Vista-Controlador producto a que separa en tres niveles las funcionalidades de la aplicación con el objetivo de aumentar la usabilidad de las mismas, inyección de dependencia para la arquitectura de los componentes.

La utilización de los patrones arquitectónicos de Symfony 2 para el desarrollo de la herramienta permitió que la misma tuviera un buen funcionamiento, no existieran problemas en el diseño general, además de simplificar el desarrollo de la herramienta ya que resuelven los problemas comunes. Los patrones que más se evidenciaron fueron, el experto el cual se verá reflejado en la clase manager XML, esta manipulará los XML subidos por el profesor en la herramienta PERA y realiza su transformación a arreglos en php. El controlador es encargado de las principales operaciones dentro del funcionamiento de Symfony2, dando solución a los principales requerimientos pertenecientes al flujo de negocio, facilitando así la centralización de las actividades. El patrón *decorator* utiliza varias plantillas globales para que guarden el código que es usual para todo el sistema y no tenerse que repetir en cada interfaz.

Se generaron los artefactos correspondientes con la metodología seleccionada como es el caso de diagrama de clases de análisis, diagrama de colaboración. El diagrama de clases de diseño se realizó con la arquitectura del patrón MVC evidenciando las clases de base de datos, así como las vistas y las controladoras. Al Symfony2 tomar el *framework* Doctrine 2 e incorporarlo dentro de sí mismo, este establece las peticiones de las controladoras, con las tablas de base de datos. Además, se realizaron los diagramas de secuencia, despliegue y por último se diseñó las tablas de base de datos con sus respectivas descripciones.

CAPÍTULO 4. IMPLEMENTACIÓN Y PRUEBA



CAPÍTULO 4. IMPLEMENTACIÓN Y PRUEBA

Uno de los objetivos más importantes de este capítulo es implementar la herramienta en forma de producto, para que se realice esta actividad los desarrolladores deben tener en cuenta toda la documentación reflejada anteriormente. Posteriormente se deben realizar las pruebas que permitan detectar los errores cometidos en la implementación de la herramienta, para darle solución lo más pronto posible. Finalmente, se debe validar la propuesta mediante los métodos definidos en la investigación para saber cuan satisfecho se sienten el cliente con la solución propuesta.

4.1 Modelo de implementación

El modelo de implementación describe cómo los elementos de diseño se implementan en componentes (ficheros de código fuente, de código binario, ejecutable, *scripts*), su objetivo general es desarrollar la arquitectura y el sistema como un todo. La fase de implementación se encarga de planificar integraciones del sistema, distribuir el sistema en nodos, implementar las clases y subsistemas, además de probar componentes individuales. (Marcos López Sanz, 2013)

4.1.1 Diagrama de componentes

Según (Marcos López Sanz, 2013) se utilizan para modelar la vista de implementación estática de un sistema y contienen:

- Componentes.
- Interfaces.
- Relaciones de dependencia, generalización, asociación y realización.
- Notas y restricciones.
- Paquetes o subsistemas.

A continuación se muestra el diagrama de componente del CU Gestionar componentes. Para estudio de los demás diagramas de componentes remitirse al **Anexo 6.**

CAPÍTULO 4. IMPLEMENTACIÓN Y PRUEBA

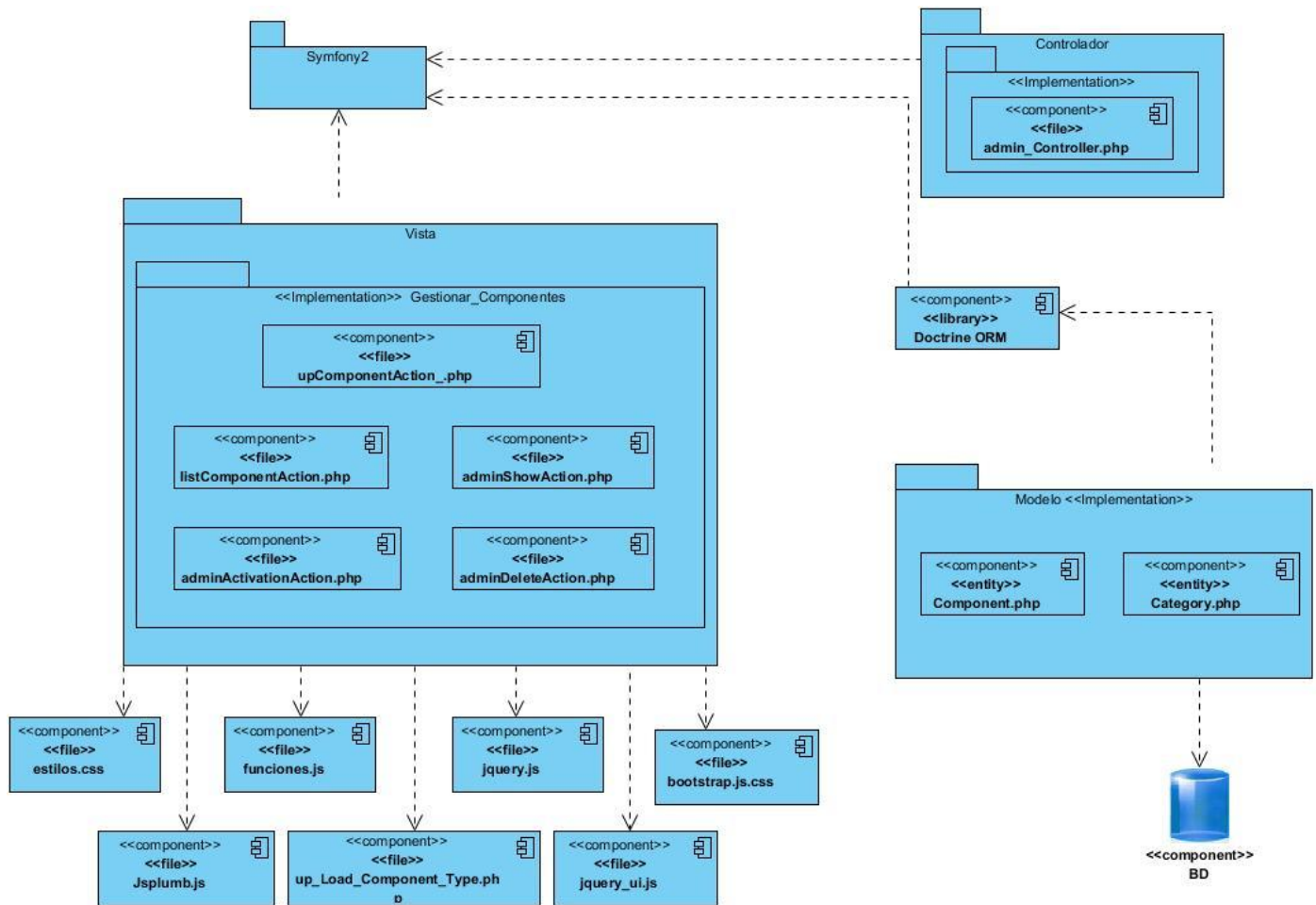


Figura 10. Diagrama de Componente del CU Gestionar Componentes

4.2 Validación del diseño de la herramienta PERA (IMS-QTI v2.0)

Para comprobar la calidad del diseño de la herramienta PERA (IMS-QTI v2.0) se emplearon las métricas Tamaño Operacional de Clase (TOC) y Relaciones entre Clases (RC), con el objetivo de medir de forma cuantitativa la calidad de los atributos internos del *software*. Esto permite evaluar la calidad durante el desarrollo del sistema. Para estudio de aspectos que se tuvieron en cuenta en la realización de las gráficas mostradas referirse al **Anexo 7**.

CAPÍTULO 4. IMPLEMENTACIÓN Y PRUEBA

Resultados de la validación de TOC

La métrica TOC permite medir la responsabilidad, la complejidad de implementación y la reutilización de las clases del diseño. Es importante destacar que para esta métrica, la responsabilidad y la complejidad son inversamente proporcionales a la reutilización, por lo que a mayor responsabilidad y complejidad de implementación de una clase, menor será su nivel de reutilización.

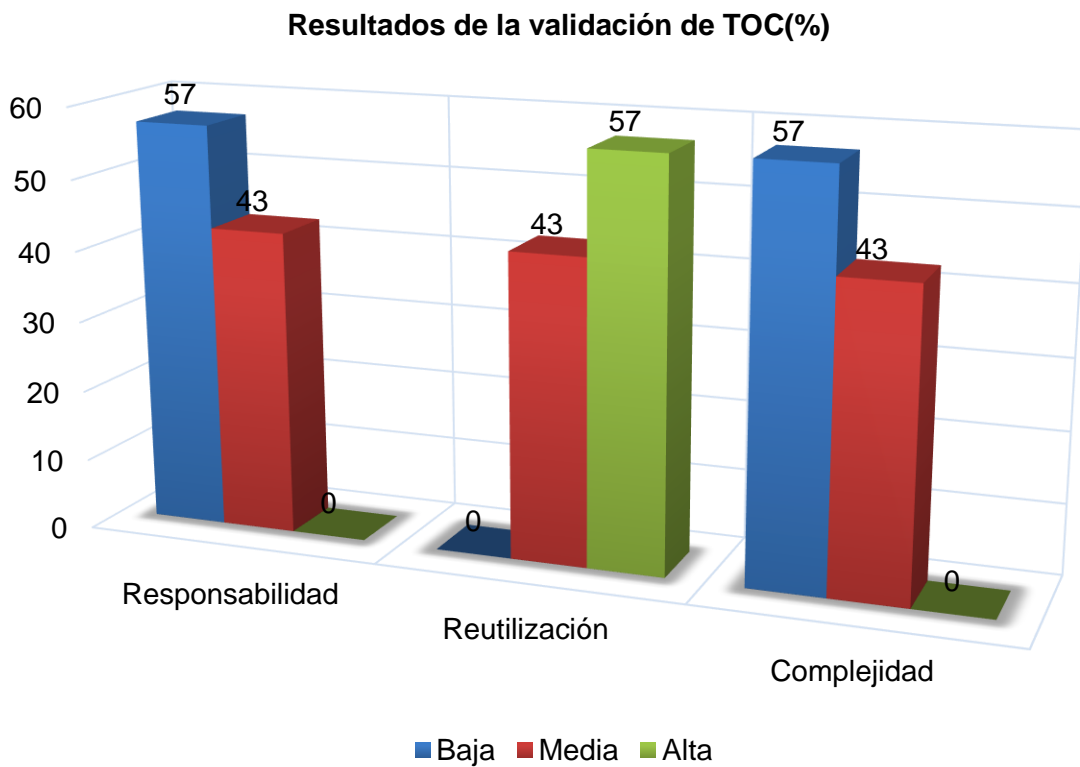


Figura 11. Representación gráfica de los resultados de la validación TOC

Luego de aplicada la métrica se observa que las clases del diseño de la herramienta PERA (IMS-QTI v2.0) no se encuentran sobrecargadas en cuanto a responsabilidades y el nivel de complejidad de las mismas no es muy alto, lo que favorece en gran medida la reutilización de estas.

CAPÍTULO 4. IMPLEMENTACIÓN Y PRUEBA

Resultados de la validación de RC

La métrica RC permite evaluar el acoplamiento, la complejidad de mantenimiento, la reutilización y la cantidad de pruebas de unidad necesarias para probar una clase teniendo en cuenta las relaciones existentes entre ellas.

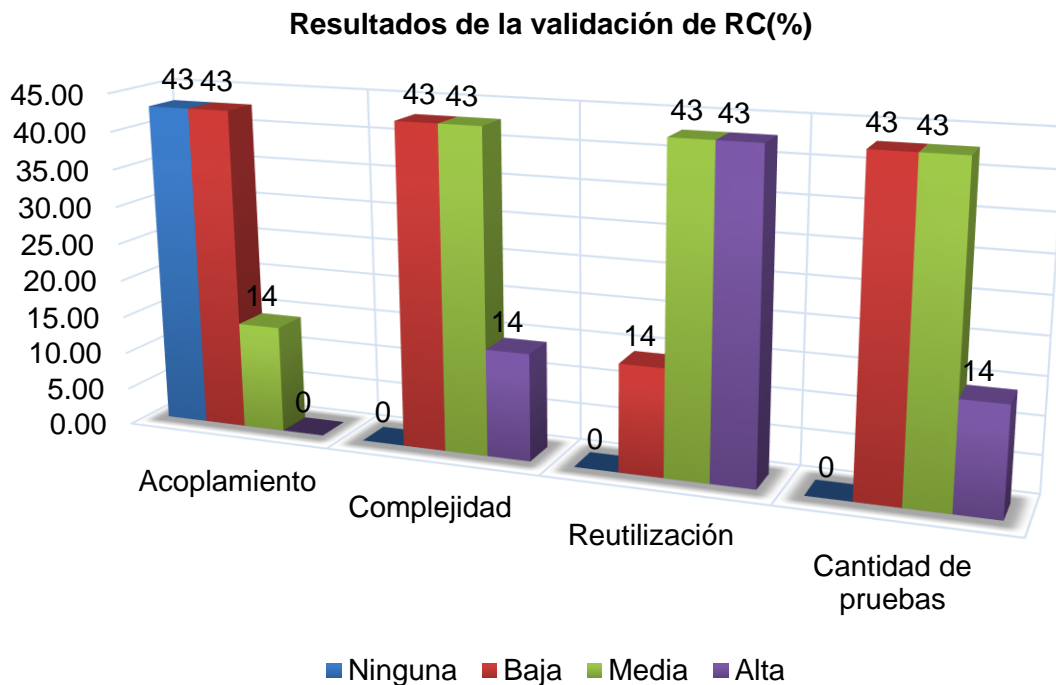


Figura 12. Representación gráfica de los resultados de la validación RC

Una vez aplicada la métrica se obtiene como resultado que las clases del diseño de la herramienta PERA (IMS-QTI v2.0) promueven el bajo acoplamiento, la complejidad de mantenimiento y la cantidad de pruebas no son altas y en consecuencia el grado de reutilización es mayor.

En sentido general, los resultados obtenidos de la aplicación de las métricas TOC y RC demuestran que el diseño de la herramienta PERA no es complejo, que las clases presentan bajo acoplamiento y un alto grado de reutilización.

CAPÍTULO 4. IMPLEMENTACIÓN Y PRUEBA

4.3 Ejemplo de los patrones de diseño alta cohesión y *strategy*

El patrón alta cohesión plantea que las clases se deben apoyar del funcionamiento de otras, para lograr su objetivo; y no incluir en ella misma todo el proceso del negocio, ya que sería difícil de comprender, de reutilizar y le afectarían constantemente los cambios. Ejemplo de ello dentro de la herramienta, es a la hora de editar un archivo, la acción *VistaComponenteAction ()* utiliza la colaboración de otras clases, tanto del modelo como de la vista. A continuación se muestra una imagen donde se evidencia lo antes mencionado.

```
public function VistaComponenteAction()
{
    $em = $this->getDoctrine()->getManager();
    $componentes = $em->getRepository('ComponenteBundle:Component')->findAll();

    return $this->render('ComponenteBundle:Default:vistaGestionarComponente.html.twig',
        array('componentes' => $componentes));
}
```

Figura 13. Ejemplo del patrón de diseño alta cohesión.

El patrón de diseño *strategy* se utiliza en la herramienta para decidir que diseño del botón Activar\Desactivar. A continuación se muestra un ejemplo de lo antes mencionado.

```
{% if componente.activation == true %}
    <a class="btn btn-success"   Activado</span></a>
{% else %}
    <a class="btn btn-success"   Desactivado</span></a>
{% endif %}
```

Figura 14. Ejemplo del patrón de diseño *strategy*.

4.4 Pruebas de *software*

Ya desarrollado el código que sustenta la herramienta, el *software* como producto puede tener defectos o fallos, por lo que se hace necesario probar cada una de sus funcionalidades para descubrir y corregir la mayor cantidad de errores posibles.

CAPÍTULO 4. IMPLEMENTACIÓN Y PRUEBA

Niveles de prueba

Tabla 8. Descripción de los Niveles de Prueba

Niveles de Pruebas				
Pruebas	Objetivo	Participantes	Ambiente	Método
Unitaria	Detectar errores en los datos, lógica, algoritmos.	Programadores	Desarrollo	Caja Blanca
Integración	Detectar errores de interfaces y relaciones entre componentes.	Programadores	Desarrollo	Caja Blanca
Funcional	Detectar errores en la implementación de requerimientos.	Analistas	Desarrollo	Caja Negra
Sistema	Detectar fallas en el cubrimiento de los requerimientos.	Analistas	Desarrollo	Caja Negra
Aceptación	Detectar fallas en la implementación del sistema.	Analistas, Clientes	Productivo	Caja Negra

4.4.1 Métodos de prueba

Caja Blanca

Pruebas con acceso al código fuente (datos y lógica). Se trabaja con entradas, salidas y el conocimiento interno. Son pruebas unitarias que se usan cuando se conoce la estructura interna y el funcionamiento del código a probar. (Perez, 2008)

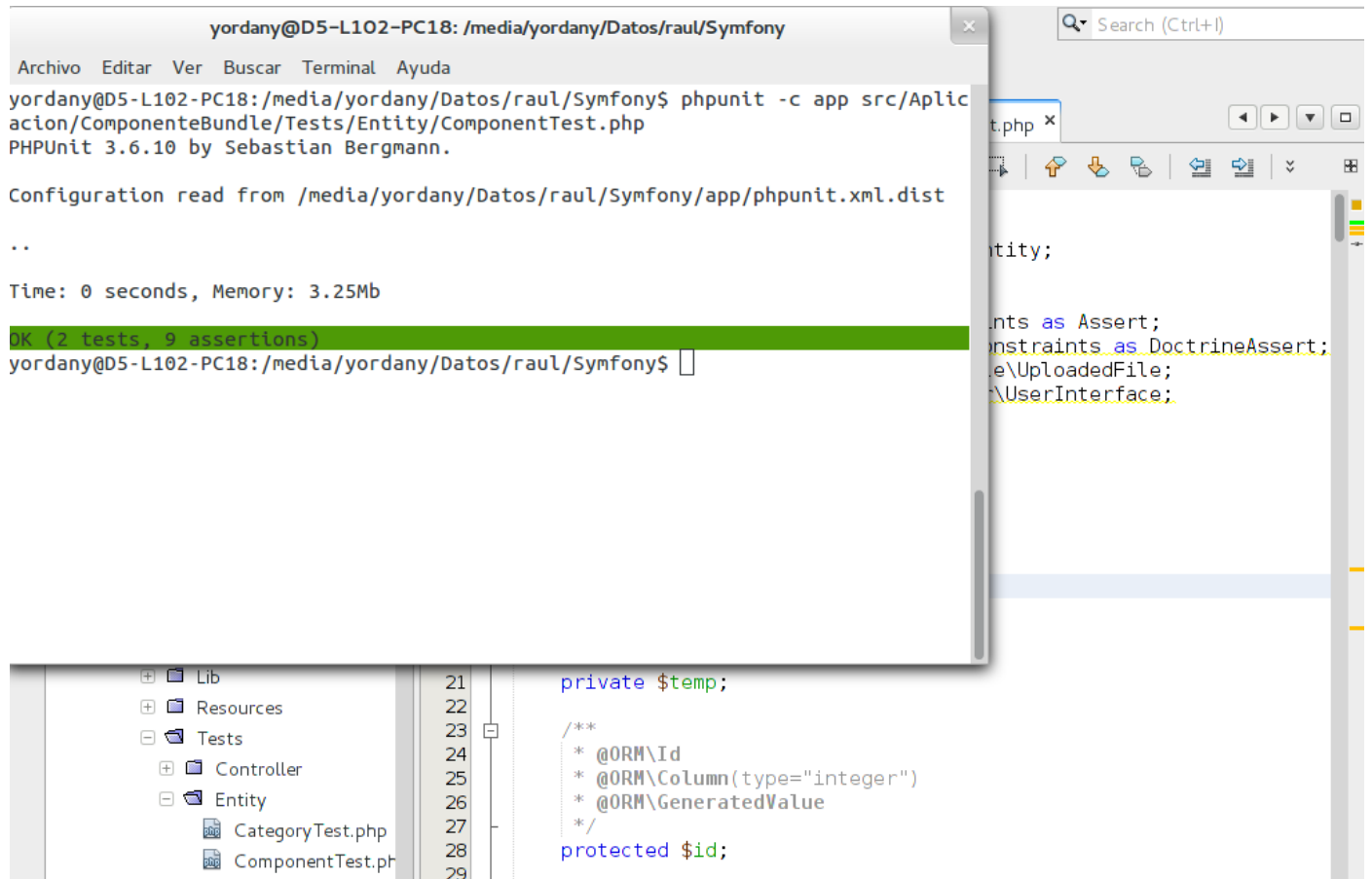
PHPUnit es un marco de pruebas programador orientada para PHP. Es un ejemplo de la arquitectura xUnit para marcos de pruebas unitarias. (Bergmann, 2014)

PHPUnit es un *framework* para PHP que facilita la creación de clases de pruebas sobre aplicaciones basadas en PHP. Es utilizado para realizar las pruebas unitarias de la herramienta PERA (IMS-QTI v2.0) ya que este presenta características tales como:

- Puerto completo de JUnit para PHP5.
- Almacena los resultados en una *Test Database*.
- Se integra con varias aplicaciones de *test*.

CAPÍTULO 4. IMPLEMENTACIÓN Y PRUEBA

A continuación se muestran algunos resultados obtenidos a partir de las pruebas unitarias generadas.



The image shows a terminal window and an IDE. The terminal window displays the output of a PHPUnit test run. The IDE shows the code for ComponentTest.php, which includes a protected property \$id and a private property \$temp.

```
yordany@D5-L102-PC18: /media/yordany/Datos/raul/Symfony
Archivo Editar Ver Buscar Terminal Ayuda
yordany@D5-L102-PC18:/media/yordany/Datos/raul/Symfony$ phpunit -c app src/Aplicacion/ComponenteBundle/Tests/Entity/ComponentTest.php
PHPUnit 3.6.10 by Sebastian Bergmann.

Configuration read from /media/yordany/Datos/raul/Symfony/app/phpunit.xml.dist

..

Time: 0 seconds, Memory: 3.25Mb

OK (2 tests, 9 assertions)
yordany@D5-L102-PC18:/media/yordany/Datos/raul/Symfony$
```

```
private $temp;

/**
 * @ORM\Id
 * @ORM\Column(type="integer")
 * @ORM\GeneratedValue
 */
protected $id;
```

Figura 15. Prueba unitaria realizada en la entidad *Component*

CAPÍTULO 4. IMPLEMENTACIÓN Y PRUEBA

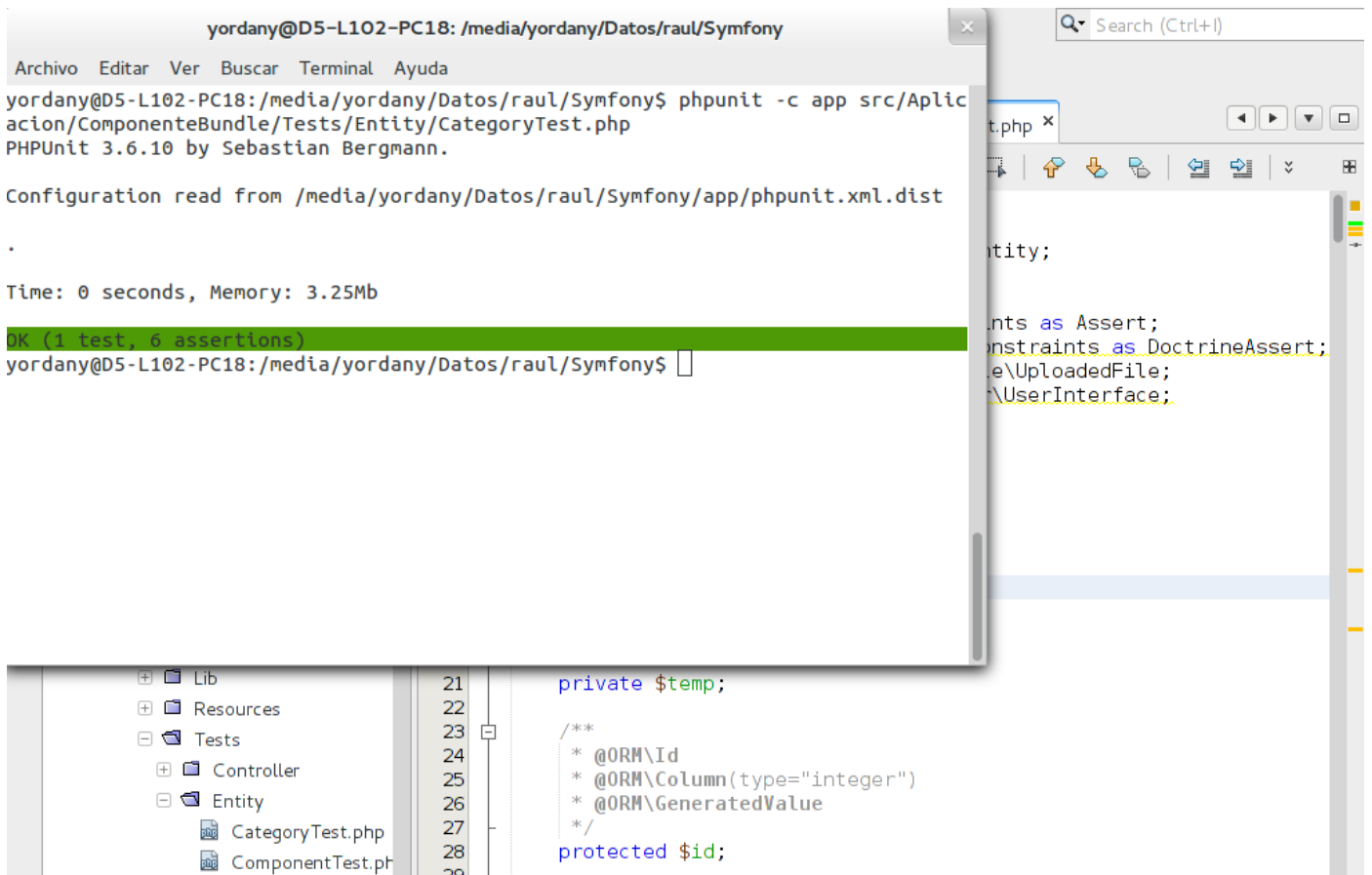


Figura 16. Prueba unitaria realizada en la entidad *Category*

Resultados de las pruebas unitarias.

Con las pruebas unitarias se validó que en las entidades *component* y *category*, el nombre de un componente o categoría no estuvieran en blanco y que el mismo tuviera la cantidad mínima de caracteres a la hora de crear un objeto de estas entidades.

Caja Negra

En estas Pruebas no se tiene acceso al código fuente de la herramienta, se trabaja con entradas y salidas. Las pruebas de caja negra se llevan a cabo sobre la interfaz del *software*, obviando el comportamiento interno y la estructura del programa. (Polo, 2008)

CAPÍTULO 4. IMPLEMENTACIÓN Y PRUEBA

Los casos de prueba de caja negra pretenden demostrar que:

- Las funciones del *software* son operativas.
- La entrada se acepta de forma correcta.
- Se produce una salida correcta.
- La integridad de la información externa se mantiene.

Pretenden encontrar estos tipos de errores:

- Funciones incorrectas o ausentes.
- Errores en la interfaz.
- Errores en estructuras de datos o en accesos a bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y de terminación.

Diseños de casos de pruebas

A continuación se presenta el CP Gestionar Componentes: sección Adicionar Componentes; para consultar el resto de Casos de Pruebas remitirse al **Anexos 8**.

Tabla 9. Diseño de CP del CU Gestionar Componentes

Escenario	Descripción	Componente (Fichero)	Respuesta del sistema	Flujo central
EC 1.1	El caso de prueba se inicia cuando el actor accede a la opción de componentes del sistema.	NA	Muestra una interfaz con una lista de componentes en el sistema y brinda la posibilidad de realizar las siguientes acciones: <ul style="list-style-type: none">• Agregar componentes al sistema.• Activar/Desactivar componentes. Ver SC: "Activar/Desactivar componentes". <ul style="list-style-type: none">• Eliminar componentes. Ver SC: "Eliminar componentes". <ul style="list-style-type: none">• Ver información de un componente. Ver SC: "Ver información de un	El profesor superior accede a gestionar componentes/Componentes/Mostrar.

CAPÍTULO 4. IMPLEMENTACIÓN Y PRUEBA



			componente". • Salir del área de gestionar componentes.	
EC 1.2	Selecciona la opción subir componentes.	V	Muestra una pestaña que te permite buscar los componentes que desea agregar. Permite realizar las acciones: • Subir • Cancelar	El profesor superior accede a gestionar componentes/Componentes/Mostrar/ Agregar nuevos componentes.
EC 1.3	Selecciona la opción examinar para buscar el componente que desea agregar.	V	Muestra la información de su PC. Permite realizar las acciones: • Abrir • Cancelar	El profesor superior accede a gestionar componentes/Componentes/Mostrar/ Agregar nuevos componentes/Agregar.
EC 1.4	Selecciona el componente, la opción abrir y la acción subir.	V	Verifica que sea un fichero .zip.	El profesor superior accede a gestionar componentes/Componentes/Mostrar/ Agregar nuevos componentes/Agregar.
EC 1.5	Si es fichero .zip.	V	Verifica que tenga la estructura definida para los componentes.	El profesor superior accede a gestionar componentes/Componentes/Mostrar/ Agregar nuevos componentes/Agregar.
EC 1.6		V		

CAPÍTULO 4. IMPLEMENTACIÓN Y PRUEBA



	Si tiene la misma estructura.		Verifica que no se encuentre en la base de datos de componentes.	El profesor superior accede a gestionar componentes/Componentes/Mostrar/Agregar nuevos componentes/Agregar.
EC 1.7	Si no está en la base de datos.	V	Agrega el componente en la base de datos de componentes y la lista de componentes del sistema.	El profesor superior accede a gestionar componentes/Componentes/Mostrar/Agregar nuevos componentes/Agregar.
EC 1.8	No es un fichero .zip.	I	Muestra un mensaje de ERROR: “La herramienta no admite este tipo de formato, verifique que sea .zip”.	El profesor superior accede a gestionar componentes/Componentes/Mostrar/Agregar nuevos componentes/Agregar.
EC 1.9	No tiene la estructura definida de los componentes.	I	Muestra un mensaje de ERROR: “No es un componente, verifique nuevamente”.	El profesor superior accede a gestionar Componentes/Componentes/Mostrar/Agregar nuevos componentes/Agregar.
EC 1.10	Está en la base de datos.	I	Muestra un mensaje de ERROR: “El componente que desea adicionar ya existe en la base de datos”.	El profesor superior accede a gestionar componentes/Componentes/Mostrar/Agregar nuevos componentes/Agregar.

CAPÍTULO 4. IMPLEMENTACIÓN Y PRUEBA



EC 1.11	Selecciona la opción cancelar petición.		Regresa a la interfaz anterior.	El profesor superior accede a gestionar componentes/Componentes/Mostrar/Agregar nuevos componentes/Cancelar.
EC 1.12	Selecciona la opción salir del área de gestionar componentes.		Termina el caso de prueba.	El profesor superior accede a gestionar componentes/Componentes/Mostrar/Salir de gestionar componentes.

Resultados de las pruebas funcionales

A continuación se muestran la cantidad de no conformidades registradas por iteraciones. Las pruebas realizadas se basaron en los Diseños de Casos de Pruebas elaborados mediante los Casos de Usos del sistema. Como se muestra en la **figura 17**, la cuarta iteración no presenta ninguna no conformidad, por lo que se cumplió el objetivo de las pruebas realizadas, validando así el funcionamiento de la herramienta.

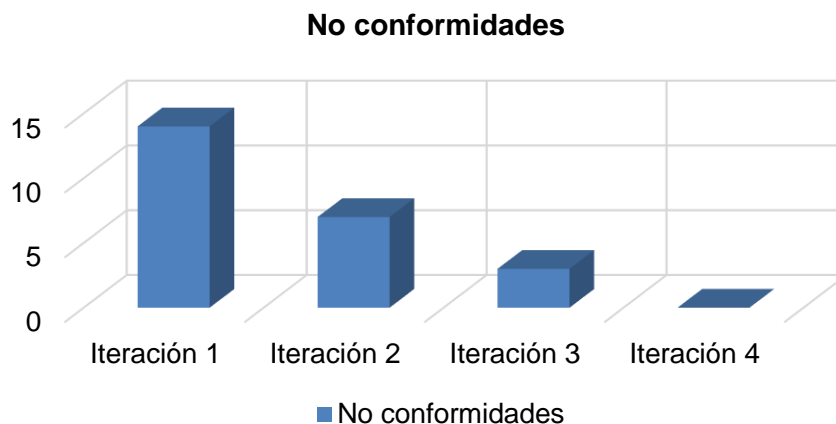


Figura 17. Cantidad de no conformidades en la herramienta PERA por iteraciones. Fuente: Elaboración propia

CAPÍTULO 4. IMPLEMENTACIÓN Y PRUEBA



4.5 Descripción de la validación de la hipótesis

La hipótesis planteada en la presente investigación: El desarrollo de una herramienta que permita a profesores de escasos conocimientos en algoritmia, modificar algoritmos de forma visual, contribuye a personalizar la evaluación de ejercicios creados bajo la especificación IMS-QTI v2.0.

Resultados obtenidos a partir de la operacionalización de las variables de la hipótesis.

Para validar las variables de la hipótesis se realizó una encuesta de 10 profesores en el Departamento de Ciencias Sociales y Humanidades de la Universidad de las Ciencias Informáticas, facultad 4. Para más detalles de los aspectos que se tuvieron en cuenta para medir las variables independientes y dependientes remitirse al **Anexo 1**.

Indicadores que se midieron en la encuesta:

Tabla 10. Indicadores utilizados para medir la encuesta desarrollada. Fuente: Elaboración propia

Indicadores	Descripciones
1. Nivel de conocimiento de los profesores en cuanto algoritmia.	Este indicador permitió saber cuál es el nivel actual de los profesores en cuanto a algoritmia y es representado gráficamente siguiendo un porcentaje.
2. Necesidad de una herramienta que le permita al profesor poder personalizar la evaluación y retroalimentaciones de un determinado ejercicio.	Muestra gráficamente el porcentaje de profesores que creen importante participar a la hora de formular e implementar la evaluación de un determinado ejercicio en línea.
3. Satisfacción de los profesores con la herramienta PERA en el proceso de evaluación de los estudiantes.	Se muestra un porcentaje de profesores que se encuentran satisfecho con la realización de la herramienta PERA (IMS-QTI v2.0)

Resultados de la encuesta:

A continuación se muestran las gráficas con los aspectos más importantes de la encuesta realizada. Para revisar los aspectos de la encuesta dirigirse al **Anexo 10**.

CAPÍTULO 4. IMPLEMENTACIÓN Y PRUEBA

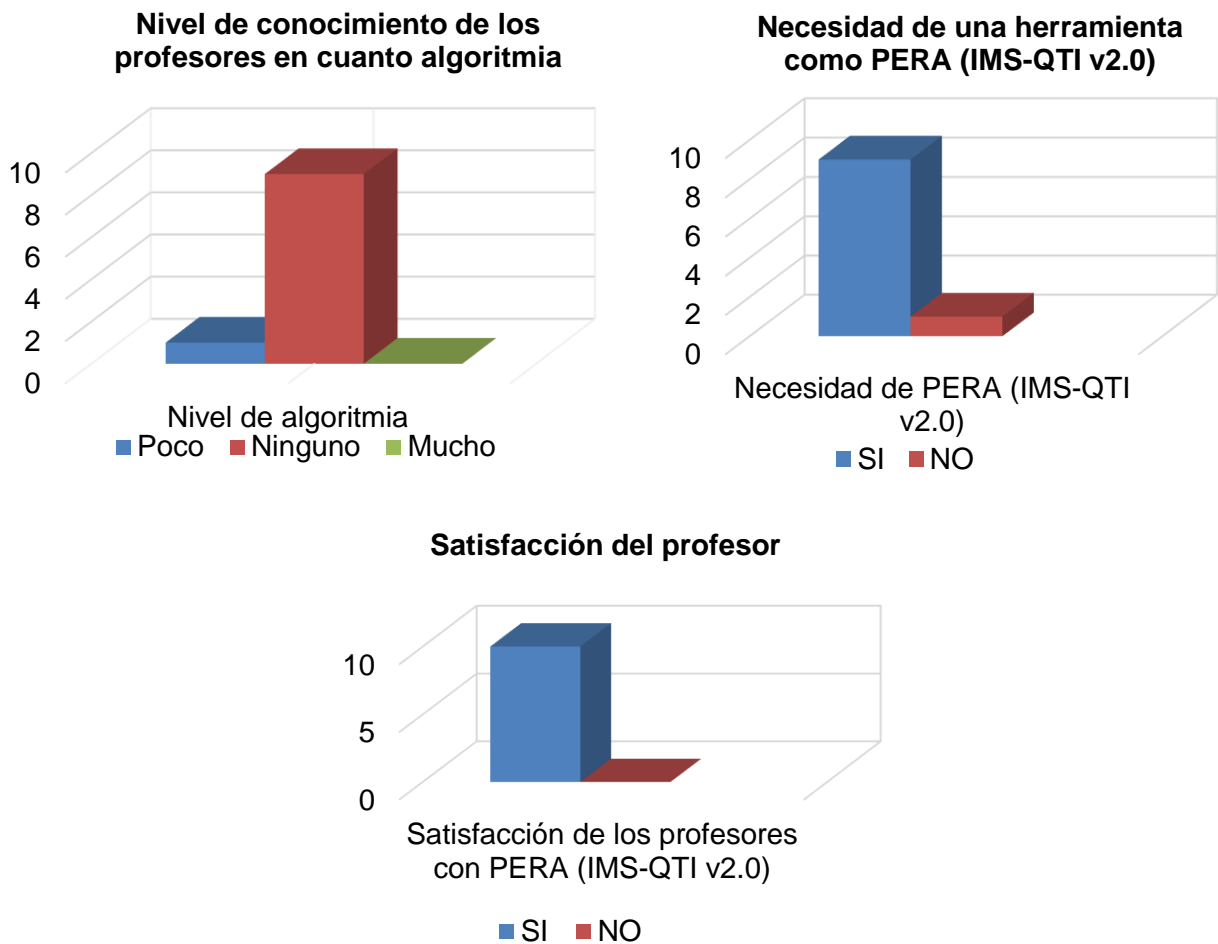


Figura 18. Resultados de los indicadores de la encuesta. Fuente: Elaboración propia

Los resultados obtenidos por la encuesta validaron la hipótesis proyectada en la investigación, demostrando la veracidad de la misma; se demostró que un 100% de los profesores coinciden en que presentan poco conocimiento o no contiene ningún conocimiento acerca de algoritmia, por lo que hay necesidad de un 99% de que exista una herramienta que ayude a los profesores a modificar algoritmos de forma visual para poder personalizar la evaluación de un ejercicio realizado bajo la especificación IMS-QTI v2.0.

Después de desarrollada la herramienta un 100% de los profesores quedó satisfecho del funcionamiento de la herramienta PERA (IMS-QTI v2.0).

CAPÍTULO 4. IMPLEMENTACIÓN Y PRUEBA

4.6 Conclusiones parciales

Después de realizadas las pruebas de caja blanca y caja negra, con el objetivo de validar la propuesta de solución, se comprobó el funcionamiento del sistema, además con la encuesta desarrollada se validó la autenticidad de la hipótesis y la satisfacción del cliente con la herramienta PERA (IMS-QTI v2.0).

Conclusiones generales

La investigación realizada permite llegar a las siguientes conclusiones:

- La personalización de la evaluación en línea no es un tema nuevo, aún se carecen de herramientas que faciliten al profesor aplicar estrategias didácticas-pedagógicas durante la confección de los mismos afectando, así la adaptabilidad de los sistemas de evaluación.
- El estudio de las herramientas para la representación de algoritmos, permitió adquirir un conjunto de buenas prácticas para el desarrollo de la solución propuesta.
- JsPlumb es una librería con un gran valor agregado por su facilidad para representar flujos de trabajos pues fueron utilizados en la representación de los algoritmos de IMS-QTI v2.0 con excelentes resultados.
- La arquitectura Modelo-Vista-Controlador facilita la migración o cambios de tecnología en cualquiera de sus capas, además brinda una organización que de seguirla los desarrolladores implementará la reutilización de componentes.
- Una herramienta que muestre los algoritmos de IMS-QTI v2.0 es un paso positivo para la personalización de ejercicios por parte de los profesores y será un gran valor agregado para el marco de trabajo Xalix

RECOMENDACIONES GENERALES

Recomendaciones generales

Para la continuidad de la presente investigación se describen una serie de recomendaciones a tener en cuenta en futuros trabajos, las recomendaciones que se especifican a continuación son dirigidas al equipo de desarrollo del marco de trabajo Xalix.

- Acoplar la herramienta PERA (IMS-QTI v2.0), al módulo para la gestión de ejercicios del marco de trabajo Xalix, a través de la interfaz de comunicación propuesta para el caso.
- Migrar el mecanismo de evaluación del marco de trabajo Xalix a un motor de evaluación de IMS-QTI v2.0 para facilitar la personalización y desacoplamiento de la evaluación de ejercicios.
- Realizar una encuesta a profesores que utilicen el marco de trabajo Xalix para conocer los criterios que utilizarían para personalizar la evaluación y realizar los componentes necesarios para que sean reflejados en PERA.

REFERENCIAS BIBLIOGRÁFICAS

Referencias bibliográficas

Álvarez, Miguel Ángel. Manual de Canvas del HTML5. Desarrollo web. [En línea]. [Citado el: 17 de enero del 2014]. Disponible en: <http://www.desarrolloweb.com>.

Alvarez, Miguel Ángel. Manual de jQuery. 2009.

Amela, Tarongí, Verónica. (2010). Sistema Tutor Inteligente Adaptativo para laboratorios virtuales y remotos.

Anuario de investigaciones - Factores cognitivos y actitudinales implicados en sistemas de personalización y de tutoría virtual en programas de e-learning aplicados a la enseñanza de la psicología. [En línea], [Citado el: 18 de octubre del 2013]. Disponible en: http://www.scielo.org.ar/scielo.php?script=sci_arttext&pid=S1851-16862010000100072

B., Mónica María Agudelo. Plataformas educativas. Programa de integración de tecnologías de la información y la comunicación a la docencia. [En línea] Programa Integración de Tecnologías a la Docencia, 2008. [Citado el: 15 de Enero del 2014]. Disponible en: <http://aprendeonline.udea.edu.co/banco/html/plataformaseducativas/>

Beck, Kent. Embracing Change with Extreme Programming. s.l.: Addison Wesley Longman, Inc, 1999. ISBN-10: 0201616416 | ISBN-13: 978-0201616415.

Biggs, J. B., & Tang, C. (1999). Teaching for quality learning at university. Buckingham: Open University Press.

Brown, S., & Glasner, A. (2003). Evaluar en la universidad. Problemas y nuevos enfoques. Madrid: Narcea.

Brusilovsky, P. (2004a). KnowledgeTree: a distributed architecture for adaptive e-learning. Proceedings of the 13th international World Wide Web conference, pp. 104-113.

Brusilovsky, P., Vassileva, J. (2003). Course sequencing techniques for large-scale webbased education. I. J. Cont. Engineering Education & Lifelong Learning, Vol. 13, Nos.1/2.

Carrión, Carmen. (2005), "Discusiones necesarias en torno a la evaluación de la educación", en Revista Mexicana de Investigación Educativa, México, COMIE, octubre-diciembre, vol. 10, núm. 27, pp. 1259-1263.

Carvajal Riola, J.C. Metodologías ágiles: Herramientas y modelo de desarrollo para aplicaciones Java EE como metodología empresarial. Barcelona: Tesis de maestría, 2008.

Ciberaula. Área Temática de Linux. Portal de Linux. [En línea]. [Citado el: 15 de marzo del 2014]. Disponible en: http://linux.ciberaula.com/articulo/linux_apache_intro/

REFERENCIAS BIBLIOGRÁFICAS

- Coll, C.** (1983). La evaluación en el proceso de enseñanza-aprendizaje. Cuadernos de pedagogía, 103, 13–17.
- DARIAS, PEREZ.** 2008. Darling, Análisis y Diseño de Componentes para Pruebas de Caja Blanca. UCI. Ciudad de La Habana, Cap. 1, pp.14, 15.
- De Bra, P., Smits, D., Stash, N.** (2006). Creating and Delivering Adaptive Courses with AHA! In W. W. Nejd and K. Tochtermann. Innovative Approaches for Learning and Knowledge Sharing. LNCS 4227, pp. 8-33.
- Elena Martín, Álvaro Marchesi.** Calidad de la enseñanza en tiempos de cambio. [En línea], [Citado el: 9 de junio del 2014]. Disponible en: <http://www.lecturalia.com/libro/17382/calidad-de-la-ensenanza-en-tiempos-de-cambio>
- Guzmán, Eduardo; Conejo, Ricardo; y Pérez de la Cruz, José Luis.** (2007): Improving Student Performance Using Self-Assessment Tests. IEEE Intelligent Systems, Vol. 22, No 4, Ago 2007.
- Harrer, A., McLaren B.M., Walker, E., Bollen, L., Sewall, J.** (2006). Creating cognitive tutors for collaborative learning: steps toward realization. User Modeling and User-Adapted Interaction, Volume 16, Numbers 3-4 / September.
- Igal, Barack.** About Me - Who is The JsMaker? [En línea]. [Citado el: 12 de febrero del 2014]. Disponible en: <http://jsmaker.com/>
- Jacobson, Ivar, Booch, Grady and Rumbaugh, James.** El Proceso Unificado de Desarrollo de *Software*. S.I.: Pearson Educacion, 2000.
- JQUERY COMMUNITY**, 2013, jQuery 2.0 Released [En línea]. [Citado el: 18 de abril del 2014]. Disponible en: <http://www.jquery.com>.
- Kabir, Mohammed J.** La Biblia Servidor Apache 2. S.I.: Anaya Multimedia, 1999. ISBN: 8441514682.
- Kareny, Brito.** Por qué utilizar RUP para desarrollar aplicaciones web. [En línea]. [Citado el: 4 de febrero del 2014]. Disponible en: <http://www.eumed.net/libros-gratis/2009c/584/Por%20que%20utilizar%20RUP%20para%20desarrollar%20aplicaciones%20web.htm>
- Kruchten, P.** The Rational Unified Process: An Introduction, 2000 Addison Wesley.
- Larman, Graig.** UML y patrones. Introducción al análisis y diseño orientado a objetos. 1999, México: Prentice Hall Hispanoamericana, S.A. pág. 167.
- Letelier, P. y Penadés.** Metodologías ágiles para el desarrollo de *software*: eXtreme Programming (XP). Valencia: Universidad de Valencia, 2008. Vol. 05, 26. ISSN 1666-1680.

REFERENCIAS BIBLIOGRÁFICAS

Linn R. 1993. Educational Measurment. Oryx Press. Estados Unidos.

Microsoft Word - ITS_using_AI_to_improve_training_performance_and_ROI.doc - ITS_using_AI_to_improve_training_performance_and_ROI.pdf. [En línea]. [Citado el: 12 de febrero del 2014]. Disponible en:

http://www.stottlerhenke.com/papers/ITS_using_AI_to_improve_training_performance_and_ROI.pdf

Mir Huguet, Josep. Estudio de los futuros estándares html5 y css3. Propuesta de actualización del sitio www.mpiua.net. Universidad de Lleida: Escuela Politécnica Superior Ingeniería Técnica en Informática de Gestión. Septiembre de 2012.

Mónica, Agudelo. Plataformas educativas. [En línea]. [Citado el: 4 de febrero del 2014]. Disponible en: <http://aprendeenlinea.udea.edu.co/banco/html/plataformaseducativas/>

Murray, T. (2003) MetaLinks: Authoring and affordances for conceptual and narrative flow in adaptive hyperbooks. In P. Brusilovsky and C. Peylo (eds.), International Journal of Artificial Intelligence in Education 13 (2-4), Special Issue on Adaptive and Intelligent Web-based Educational Systems, 199-233.

NETBEANS.ORG. NetBeans IDE 7.2.1 Release Notes. [En línea]. [Citado el: 4 de febrero del 2014]. Disponible en: <https://netbeans.org/community/releases/72/relnotes.html>

Nieto, Perez, Iván. Capítulo 1: Introducción. [En línea]. [Citado el: 4 de febrero del 2014]. Disponible en: <http://www.elcodigo.net/tutoriales/javascript/javascript1.html>

Ong J. & Ramachandran S. (2000): "Intelligent tutoring systems: the what and the how", Learning Circuits, publicado por la American Society of Training and Development. Disponible en: <http://www.learningcircuits.org/feb2000/ong.html>

Palacio, J. Flexibilidad con Scrum, principios de diseño e implantación en campos Scrum. s.l. : SafeCreative, 2007.

Paramythis A., Loidl-Reisinger S., and Kepler J. Adaptive Learning Environments and e-Learning Standards. Electronic Journal of eLearning, EJEL: Vol 2. Issue 1, March, 2004.

Pedro, Ahumada Acevedo. 2001, LA EVALUACIÓN EN UNA CONCEPCIÓN DE APRENDIZAJE SIGNIFICATIVO, ISBN 956-17-0323-8.

Pellegrino, J., Chudowsky, N., & Glaser, R. (2001). Knowing what students know: The science and design of educational assessment. Washington, DC: National Academy Press.

PEREDA, Ernesto Vladimir. 2013, Pautas de Codificación, Centro FORTES [En línea]. [Citado el: 9 de febrero del 2014]. Disponible en: http://gespro.fortes.prod.uci.cu/attachments/20023/Estandar_de_codificaci%C3%B3n_Xalix_.pdf_Versión_1.0

Pérez Eguíluz, Javier. Introducción a JavaScript. Madrid: Autoedición, 2009.

REFERENCIAS BIBLIOGRÁFICAS

- POLO USAOLA, Dr. Macario.** *Curso de doctorado sobre Proceso software y gestión del conocimiento. Pruebas del Software.* Departamento de Tecnologías y Sistemas de Información. Ciudad Real. 2006.
- PGADMIN.ORG.** pgAdmin: PostgreSQL administration and management tools. [En línea]. [Citado el: 4 de febrero del 2014]. Disponible en: <http://www.pgadmin.org/>
- PHP.** PHP. [En línea]. [Citado el: 24 de abril del 2014]. Disponible en: www.php.net
- Pila Teleña, A.** 1995. Preparación física. Tomos I-II-III. Madrid. Editorial. Augusto Pila Teleña, 1985.
- Pressman, Roger S.** Ingeniería de *software*, un enfoque práctico, quinta. Madrid y Carachelejo: s.n. 2001. pp. 294.
- Rivera, E.A., Zamora, R.G. y Soria, M.G.** Sistema de Educación a Distancia. S.I.: IV Congreso de Tecnología en Educación, 2012.
- Rosales, C.** (1990). Evaluar es reflexionar sobre la enseñanza. Madrid: Narcea. Disponible en: <http://dialnet.unirioja>
- Rodríguez Neira, T.** (2000). Enseñanza Escolar: Situaciones y perspectivas. Oviedo, ICE, 2000.
- Sampier, Roberto Hernández.** Metodología de la investigación. La Habana: Félix Varela, 2008.
- Santos, O.C., Boticario, J.G.** (2006). Meaningful pedagogy via covering the entire life cycle of adaptive eLearning in terms of a pervasive use of educational standards: the aLFanet experience. In W. W. Nejdil and K. Tochtermann eds. Innovative Approaches for Learning and Knowledge Sharing. LNCS 4227, pp. 691-696, 2006. [En línea]. [Citado el: 1 de marzo del 2014]. Disponible en: <http://www.ia.uned.es/~jgb/publica/42270691-ectel06-final.pdf>
- Bergmann, Sebastian.** (2014) [En línea]. [Citado el: 10 de marzo del 2014]. Disponible en: <http://phpunit.de/>
- Schwaber, K. y Beedle, M.** Agile *Software Development with Scrum*. s.l.: Upper Saddle River, Prentice Hall, 2002.
- Scriven, Michael.** (1967), "The methodology of evaluation", en R. W. Tyler, R. M. Gagné y M. Scriven (eds.), *Perspectives of curriculum evaluation*, Chicago, Rand McNally, pp. 39-83
- Sierra, Manuel, 2009.** Qué es un servidor y cuáles son los principales tipos de servidores (proxy, dns, web, ftp, smtp...). [En línea]. [Citado el: 1 de junio 2014]. Disponible en: http://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=542:que-es-un-servidor-y-cuales-son-los-principales-tipos-de-servidores-proxydns-webftpsmtp&catid=57:herramientas-informaticas&Itemid=179

REFERENCIAS BIBLIOGRÁFICAS

Sobre PostgreSQL | www.postgresql.org.es. [En línea], [Citado el: 4 de febrero del 2014]. Disponible en: http://www.postgresql.org.es/sobre_postgresql

Sommerville, I. Ingeniería de *software*. 7ª edición. Capítulo 6, Pág. 109-116

Studio, Ideadelivery. Ideadelivery Studio. 2011, [En línea]. [Citado el: 14 de Enero de 2014.]. Disponible en: http://www.ideadeliveryla.com/5_ventajas_del_HTML5_para_el_dise%C3%B1o_de_sitios_web

Stufflebeam, D. L. (1966). A depth study of the evaluation requeriment. *Theory into Practice*, 5, 3, 121-134.

Stufflebeam, D. L., Foley, WJ, Gephart, WJ, Guba, EG, Hammond, RL, Merriman, HO & Provus, MM. (1971). *Educational Evaluation and Decision-making*, Itasca, Illinois: F. E. Peacock Publishing.

The Apache Software Foundation. Apache HTTP SERVER PROJECT. [En línea]. [Citado el: 10 de junio del 2014]

Disponible en:

<http://httpd.apache.org/>.

http://httpd.apache.org/docs/2.0/es/new_features_2_0.html

Uazuay. uazuay. uazuay. 2011, [En línea]. [Citado el: 11 de junio del 2014]. Disponible en: http://www.uazuay.edu.ec/estudios/sistemas/lenguaje_iii/MAnnualJavaScript/introduccion.htm.

VERMILION. 2013, Responsive CSS *Framework* Comparison. Bootstrap vs. Foundation vs. Skeleton. [En línea]. [Citado el: 10 de febrero del 2014]. Available from:

<http://responsive.vermilion.com/compare.php>

W3C. Guía Breve de Tecnologías XML. [En línea]. [Citado el: 10 de febrero del 2014]. Disponible en: <http://www.w3c.es/Divulgacion/GuiasBreves/TecnologiasXML>

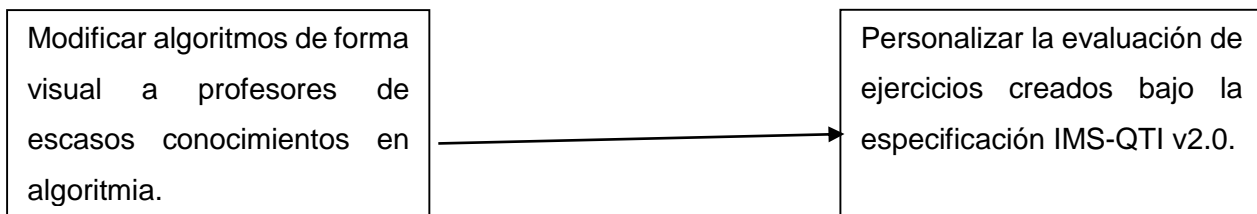
Wesley, Addison. El proceso unificado de desarrollo de *software*. 1999.

ANEXOS

5.1 Anexo 1. Operacionalización de las variables

A continuación se puede presenciar la relación entre las variables independientes y dependiente:

Variable Independiente Variable Dependiente



Variables independientes:

Modificar algoritmos de forma visual a profesores de escaso conocimiento en algoritmia: Las funcionalidades que tendrá la herramienta y que permitirán que el profesor pueda modificar fácilmente un algoritmo mediante un diagrama el cual será mostrado por la herramienta a desarrollar.

Método de validación: Una encuesta para saber el conocimiento que tienen los profesores en algoritmia y las pruebas del sistema para validar las modificaciones de algoritmos referentes a los ejercicios creados bajo la especificación IMS-QTI v2.0.

5.2 Anexo 2. Descripciones de los Casos de Usos

Descripciones de los casos de usos Exportar ejercicio o diagrama de evaluación, Importar ejercicio, Modificar diagrama de la evaluación del ejercicio, Generar código del diagrama modificado, Generar diagrama del código interno del ejercicio.

CU 2. Exportar ejercicio o diagrama de evaluación

Anexo 2. Tabla 1. Descripción del CU Exportar ejercicio o diagrama de evaluación

Objetivo	El objetivo que persigue el actor con este caso de uso es poder exportar el ejercicio con el formato IMS-QTI v2.0 o el diagrama de evaluación en JPG, luego de ser generado el diagrama de evaluación del ejercicio y modificada la evaluación avanzada.	
Actores	Profesor: (Inicia) exporta el ejercicio.	
Resumen	El caso de uso permite que el actor realice la acción de exporta ejercicio en formato IMS-QTI v2.0 y el diagrama de evaluación en formato JPG, luego de que se haya modificado la evaluación avanzada del ejercicio.	
Complejidad	Baja	
Prioridad	Alta	
Precondiciones	Tiene que mostrarse un diagrama en la herramienta. Debe ser modificada la evaluación avanzada del ejercicio para que sea exportado en IMS-QTI v2.0.	
Postcondiciones	Se exportó el ejercicio en formato IMS-QTI v2.0 o JPG. El código generado de ese diagrama que se exporte tiene que ser de formato IMS-QTI v2.0. Si se exporta el ejercicio en formato IMS-QTI v2.0 debe generar primeramente el algoritmo IMS-QTI v2.0.	
Flujo de eventos		
Flujo básico		
	Actor	Sistema
1.	El caso de uso se inicia cuando el actor selecciona la opción de exportar un ejercicio.	
2.		Muestra los formatos que existen para exportar el ejercicio: <ul style="list-style-type: none"> • IMS-QTI v2.0

ANEXO 2

		<ul style="list-style-type: none"> JPG
3.	Selecciona exportar el ejercicio en formato IMS-QTI v2.0.	
4.		Permite: <ul style="list-style-type: none"> Exportar Cancelar petición
5.	Selecciona el lugar donde guardará el ejercicio o diagrama de evaluación.	
6.		Exporta el ejercicio o el diagrama en el formato que se especificó.
7.		Termina el caso de uso
Flujos alternos		
4. a Selecciona Cancelar petición.		
	Actor	Sistema
1.		Regresa a la vista anterior.
2.		Termina el caso de uso.
Flujos alternos		
3. a Selecciona exportar el diagrama de evaluación en JPG.		
	Actor	Sistema
1.		Exporta la imagen.
2.		Regresa la paso 4 del flujo de trabajo.
Relaciones	CU incluidos	No tiene
	CU extendidos	No tiene
Requisitos funcionales	no	-
Asuntos pendientes		-

CU 3. Importar ejercicio

Anexo 2. Tabla 2. Descripción del CU Importar ejercicio

Objetivo	El objetivo que persigue el actor con este caso de uso es importar un ejercicio en formato IMS-QTI v2.0 para su posterior modificación en la herramienta.	
Actores	Profesor: (Inicia) importa el ejercicio que contiene un algoritmo en formato IMS-QTI v2.0 o se genera de la plataforma mediante una opción que brinde.	
Resumen	Este caso de uso permite importar un ejercicio de formato IMS-QTI v2.0, para su posterior modificación de la evaluación avanzada del mismo.	
Complejidad	Baja	
Prioridad	Alta	
Precondiciones	El ejercicio a importar debe estar en formato IMS-QTI v2.0.	
Postcondiciones	Se importó el ejercicio de formato IMS-QTI v2.0. Debe generar un diagrama y mostrarlo en la herramienta.	
Flujo de eventos		
Flujo básico		
	Actor	Sistema
1.	El caso de uso se inicia cuando el actor selecciona la opción abrir.	
2.		Brinda la opción de buscar el ejercicio que se quiere importar. Permite realizar las acciones: <ul style="list-style-type: none"> • Subir • Cancelar
3.	Busca el ejercicio mediante la opción examinar.	
4.		Permite realizar la acción: <ul style="list-style-type: none"> • Abrir • Cancelar
5.	Selecciona el ejercicio, realiza la acción abrir y sube el ejercicio.	
6.		Válida que el ejercicio sea de formato IMS-QTI v2.0.

7.		Si el ejercicio es en formato IMS-QTI v2.0, genera un diagrama de evaluación a partir del código interno de ese ejercicio importado. <u>Ver CU Generar Diagrama de Evaluación del Ejercicio.</u>
8.		Muestra el diagrama en la herramienta y permite modificar el diagrama de evaluación del ejercicio. <u>Ver CU: Modificar Diagrama de Evaluación del Ejercicio.</u>
9.		Termina el caso de uso.
Flujos alternos		
*.a Selecciona la opción de cancelar petición.		
	Actor	Sistema
1.		Termina el caso de uso.
Flujos alternos		
7. a Si el ejercicio no es de formato IMS-QTI v2.0.		
	Actor	Sistema
1.		Muestra un mensaje de error: “Este formato no está validado por el sistema, verifique nuevamente”
2.		Regresa al paso 2 del flujo de trabajo.
Relaciones	CU incluidos	Generar Diagrama de Evaluación del Ejercicio. Ver CU Generar Diagrama de Evaluación del Ejercicio.
	CU extendidos	No tiene
Requisitos no funcionales	no	-
Asuntos pendientes		-

CU 4. Modificar diagrama de evaluación de ejercicio

Anexo 2. Tabla 3. Descripción del CU Modificar Diagrama de Evaluación de Ejercicio

Objetivo	El objetivo que persigue el actor con este caso de uso es modificar el diagrama de evaluación de ejercicio con el objetivo de modificar la evaluación del ejercicio para su previa utilización.
Actores	Profesor: (Inicia) importa el ejercicio en formato IMS-QTI v2.0 o se genera de la plataforma mediante una opción que brinde.

Resumen	En este caso de uso se modifica el diagrama de evaluación de ejercicio a través de agregar nuevas instancias de componentes, eliminar instancia de un componente y todas sus relaciones, relacionar las instancias de los componentes, eliminar relaciones entre instancias de los componentes y gestionar parámetros de las relaciones de las instancias.	
Complejidad	Baja	
Prioridad	Baja	
Precondiciones	El sistema debe generar previamente un diagrama y mostrarlo para su posterior modificación.	
Postcondiciones	Se modificó el diagrama a partir de cualquiera de las opciones permitidas. Se actualizó el código interno del ejercicio con las modificaciones realizadas al diagrama de evaluación.	
Flujo de eventos		
Flujo básico		
	Actor	Sistema
1.		El caso de uso inicia cuando el sistema muestra la evaluación del ejercicio en forma de diagrama.
2.		<p>Permite realizar varias acciones:</p> <ul style="list-style-type: none"> • Eliminar instancia de un componente y todas sus relaciones. <p>Ver Sección 1: <u>Eliminar instancia de un componente y todas sus relaciones.</u></p> <ul style="list-style-type: none"> • Agregar nuevas instancias de componentes. • Relacionar las instancias de los componentes. • Eliminar relaciones entre instancias de los componentes. <p>Ver Sección 2: <u>Eliminar relaciones entre instancias de los componentes.</u></p> <ul style="list-style-type: none"> • Gestionar parámetros de las relaciones de las instancias. <p>Ver Sección 3: <u>Gestionar parámetros de las relaciones de las instancias.</u></p> <ul style="list-style-type: none"> • Cancelar la operación realizada con Deshacer.

		<ul style="list-style-type: none"> • Recuperar la acción cancelada con Rehacer. • Finalizar modificación. • Guardar cambios realizados. • Exportar un ejercicio en un tipo de formato definido.
3.	Selecciona la acción de exportar ejercicio.	
4.		Exporta el ejercicio. <u>Ver CU Exportar Ejercicio o Diagrama de Evaluación.</u>
5.		Termina el caso de uso.
Flujos alternos		
3. b Agregar nuevas instancias de componentes.		
	Actor	Sistema
1.		Permite la acción de arrastrar instancias de componentes de la barra de componentes a la vista principal.
2.	Realiza la acción arrastrar la instancia del componente hasta la vista principal.	
3.		Muestra el componente deseado en la vista principal.
4.		Regresa al paso 2 del flujo de trabajo.
Flujos alternos		
3. c Selecciona la acción de relacionar las instancias de componentes.		
	Actor	Sistema
1.		Muestra el componente en el área de trabajo. El cual trae consigo un nodo que lo relaciona con cualquier otra instancia de otro componente.
2.	Realiza la acción de hacer en el nodo origen de la instancia del componente y arrastra el mouse hasta el nodo destino del componente deseado.	
3.		Se valida que cumplan los componentes a relacionar con las restricciones propias del componente.
4.		Si cumplen las restricciones los componentes a relacionar:

		Se muestra la relación entre los dos nodos deseados.
5.		Regresa al paso 2 del flujo de trabajo.
Flujos alternos		
4. a No cumple con las restricciones del componente a relacionar.		
	Actor	Sistema
1.		Muestra un mensaje de ERROR: "Las instancias de estos componentes no son compatibles".
2.		Regresa al paso 2 del flujo de trabajo.
Flujos alternos		
3. f Selecciona la acción de cancelar la operación realizada con Deshacer		
	Actor	Sistema
1.	Selecciona la opción Deshacer.	
2.		Desase la acción realizada actualmente.
3.		Regresa al paso 2 del flujo de trabajo.
Relaciones	CU incluidos	No tiene
	CU extendidos	No tiene
Requisitos funcionales	no	-
Asuntos pendientes		-

Sección 1: Selecciona la acción de eliminar instancia de un componente y todas sus relaciones.

Flujo básico: El actor selecciona la acción de eliminar instancia de un componente y todas sus relaciones.		
	Actor	Sistema
1.		Permite: <ul style="list-style-type: none"> Hacer clic sobre la instancia del componente y tocar la tecla "Delete" de la lista de operaciones de la interfaz principal.
2.	Realiza la acción de dar clic arriba de la instancia del componente y presionar la tecla "Delete" de la interfaz principal.	

3.		Muestra mensaje de verificación: “Seguro que desea eliminar las instancias” y permite realizar acciones: <ul style="list-style-type: none"> • Aceptar • Cancelar
4.	Selecciona la opción “Aceptar”	
5.		Ejecuta la acción y borra la instancia del componente y sus relaciones.
6.		Termina el caso de uso.

Flujos alternos

3.a Selecciona la opción “Cancelar”

	Actor	Sistema
1.		Muestra el componente en el área de trabajo.
2.		Regresa al paso 2 del flujo de trabajo.

Sección 2: Selecciona la acción de eliminar relaciones entre instancia de componentes.

Flujo básico: El actor selecciona la acción de eliminar relaciones entre instancias de componentes.

	Actor	Sistema
1.		Muestra las instancias de componentes y sus respectivas relaciones.
2.	Selecciona la relación que desea eliminar y arrastra la misma hasta cualquier parte de la interfaz principal que no tenga instancias.	
3.		Ejecuta la acción y elimina relaciones entre instancias de componentes.
4.		Termina la sección 2.

Sección 3: Selecciona la acción de gestionar parámetros de las relaciones de las instancias.

Flujo básico: El actor selecciona la acción de gestionar parámetros de las relaciones de las instancias.

	Actor	Sistema
1.		Muestra las instancias de los componentes en el área de trabajo y sus respectivas relaciones. Donde en cada

		instancia de componentes existe también un icono que te permite gestionar los parámetros de la relación de esa instancia.
2.	Selecciona el icono que te permite gestionar los parámetros de esa relación.	
3.		Permite editar ese parámetro, mediante el doble clic arriba del espacio del componente.
4.	Escribe le nuevo parámetro.	
5.		Muestra el parámetro actualizado en las relaciones de los componentes.
6.		Termina la sección 3.

CU 5. Generar diagrama de evaluación del ejercicio

Anexo 2. Tabla 4. Descripción del CU Generar diagrama de evaluación del ejercicio

Objetivo	El objetivo del este caso de uso es generar un diagrama a partir de la evaluación de un ejercicio.	
Actores	Profesor: (Inicia) importa un ejercicio o se genera automáticamente de una plataforma.	
Resumen	El caso de uso permite que se genere un diagrama de evaluación a partir del algoritmo IMS-QTI v2.0 que sea importado o generado por la plataforma.	
Complejidad	Alta	
Prioridad	Alta	
Precondiciones	Tiene que importarse un ejercicio o generarse automáticamente de la plataforma que utilice la herramienta.	
Postcondiciones	Se generó y se mostró el diagrama del código interno de las evaluaciones de un ejercicio.	
Flujo de eventos		
Flujo básico		
	Actor	Sistema
1.	El caso de uso inicia cuando el actor accede a la opción de Evaluación Avanzada desde una plataforma o importa el ejercicio manualmente.	

ANEXO 2

2.		Valida que el ejercicio sea formato IMS-QTI v2.0.
3.		Si se cumple con el formato: Valida que la herramienta tenga los componentes necesarios para graficar las etiquetas que contiene el ejercicio.
4.		Si se tienen los componentes necesarios: Asigna un componente por cada etiqueta y se guarda en un arreglo.
5.		Se genera un diagrama de la evaluación y retroalimentación del ejercicio con esos componentes.
6.		Muestra el diagrama de evaluación y retroalimentación del ejercicio en la interfaz principal de la herramienta.
7.		Termina el caso de uso.
Flujos alternos		
3. a No cumple con el formato que debe tener el ejercicio.		
	Actor	Sistema
1.		No carga el ejercicio y muestra un mensaje de ERROR: "No se permite ejercicios de este tipo de formato verifíquelo".
2.		Regresa a la vista anterior.
Flujos alternos		
4. a No se tienen los componentes necesarios para las etiquetas del ejercicio.		
	Actor	Sistema
1.		Muestra un mensaje de ERROR: "No existen los componentes necesarios para poder cargar el ejercicio"
2.		Regresa a la vista anterior.
Relaciones	CU incluidos	No tiene
	CU extendidos	No tiene
Requisitos no funcionales		-
Asuntos pendientes		-

CU 6. Generar código del diagrama de evaluación del ejercicio

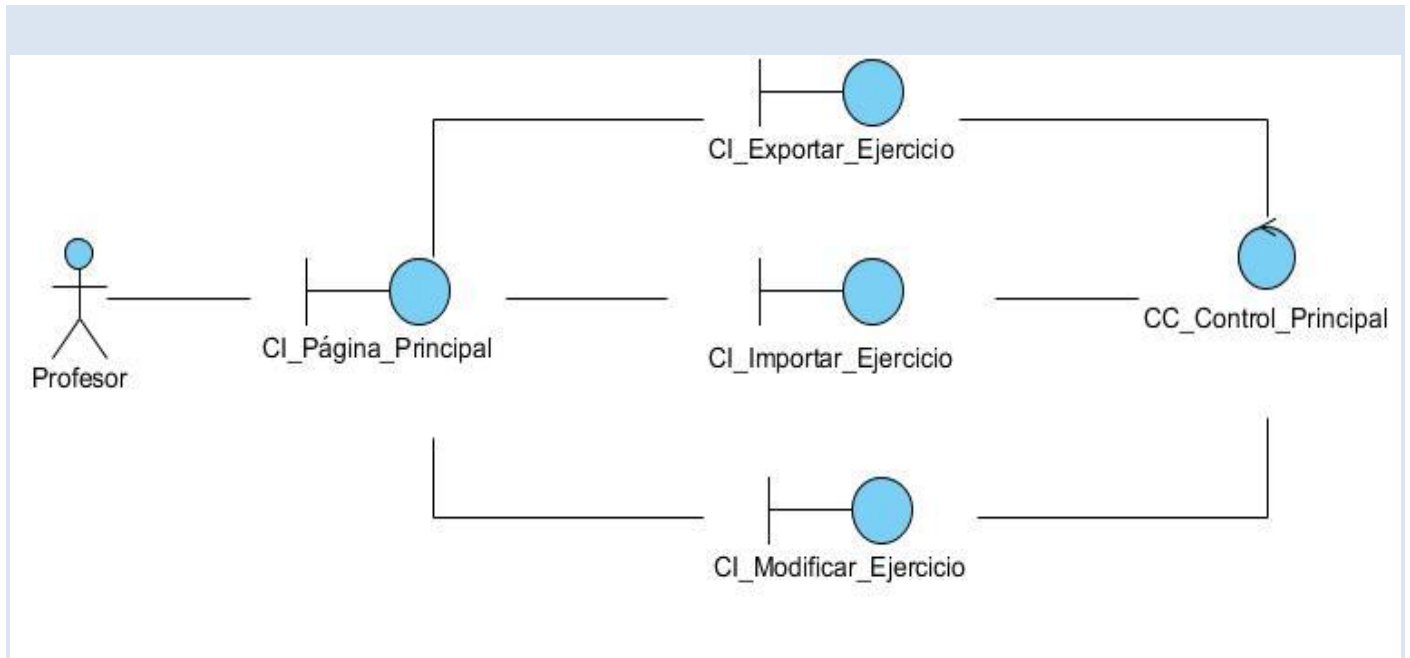
Anexo 2. Tabla 5. Descripción del CU Generar código del diagrama de evaluación del ejercicio

Objetivo	El objetivo del este caso de uso es generar el código a partir del diagrama de evaluación modificado por el profesor.	
Actores	Profesor: (Inicia) exporta un ejercicio en formato IMS-QTI v2.0.	
Resumen	El caso de uso permite que el actor pueda generar el código del diagrama de la evaluación modificado por el profesor para que posteriormente sea interpretado por el motor de evaluación de la plataforma que utilice la herramienta o simplemente generar este código para su futura reutilización.	
Complejidad	Alta	
Prioridad	Alta	
Precondiciones	Tiene que existir un diagrama en la vista principal de la herramienta. Tiene que modificarse el diagrama importado o generado por el profesor o la plataforma.	
Postcondiciones	Se generó el código interno del diagrama de evaluaciones de un ejercicio.	
Flujo de eventos		
Flujo básico		
	Actor	Sistema
1.	El caso de uso inicia cuando el actor accede a la opción de Finalizar la modificación del diagrama.	
2.		Verifica de donde vino el ejercicio si se importó manualmente por el actor o simplemente se generó de la plataforma.
3.		Si se importó manualmente: Permite que el usuario exporte el ejercicio o el diagrama de evaluación del ejercicio en el directorio que el especifique. <u>Ver CU Exportar Ejercicio o Diagrama de Evaluación.</u>
4.		Si se exporta en formato IMS-QTI v2.0: Convierte en etiquetas de formato php los componentes actuales del diagrama de evaluación, generando así el código del ejercicio en formato XML y se guarda el fichero con este tipo de formato.

5.		Termina el caso de uso.
Flujos alternos		
3. a Se generó de la plataforma.		
	Actor	Sistema
1.		Captura la información del diagrama y la envía al servidor donde se crea un árbol el cual se utiliza para crear el XML; luego este se envía automáticamente el código del ejercicio a la plataforma.
2.		Termina el caso de uso.
Flujos alternos		
4. a Si se exporta a JPG.		
	Actor	Sistema
1.		Se captura el diagrama de evaluación que aparece en la interfaz principal y se guarda en el directorio que desee el actor.
2.		Regresa al paso 4 del flujo básico.
Relaciones	CU incluidos	No tiene
	CU extendidos	No tiene
Requisitos funcionales	no	-
Asuntos pendientes		-

5.3 Anexo 3. Diagrama de clase de análisis

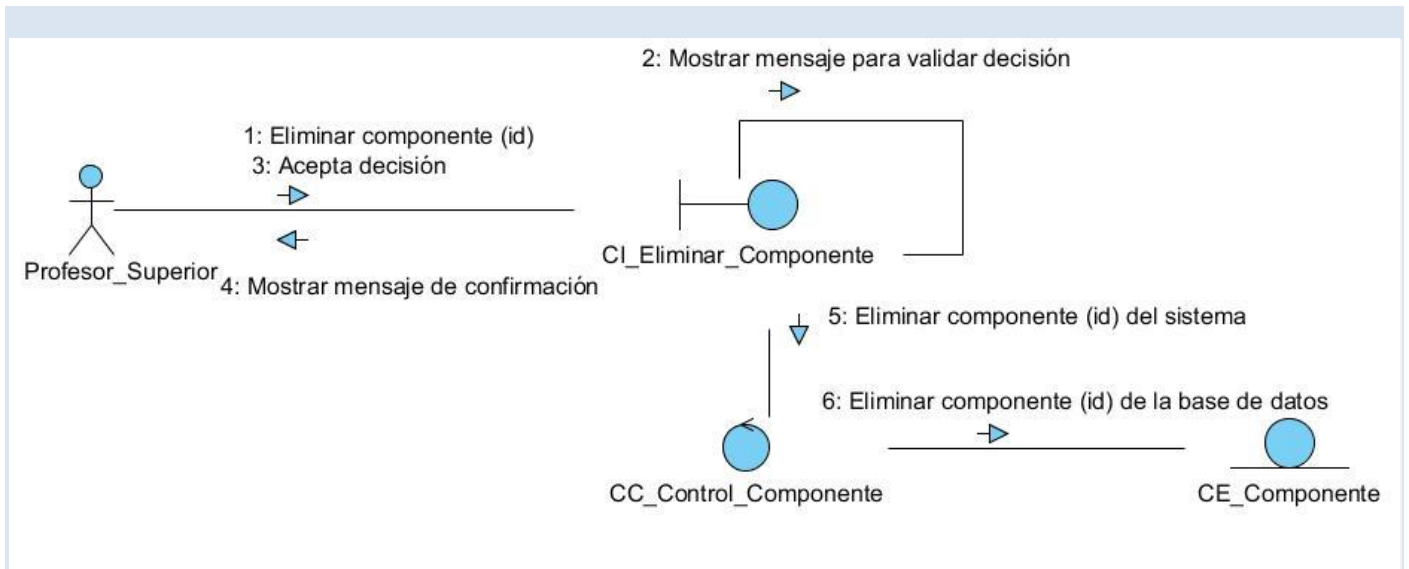
Diagrama de clases de análisis de la interfaz principal.



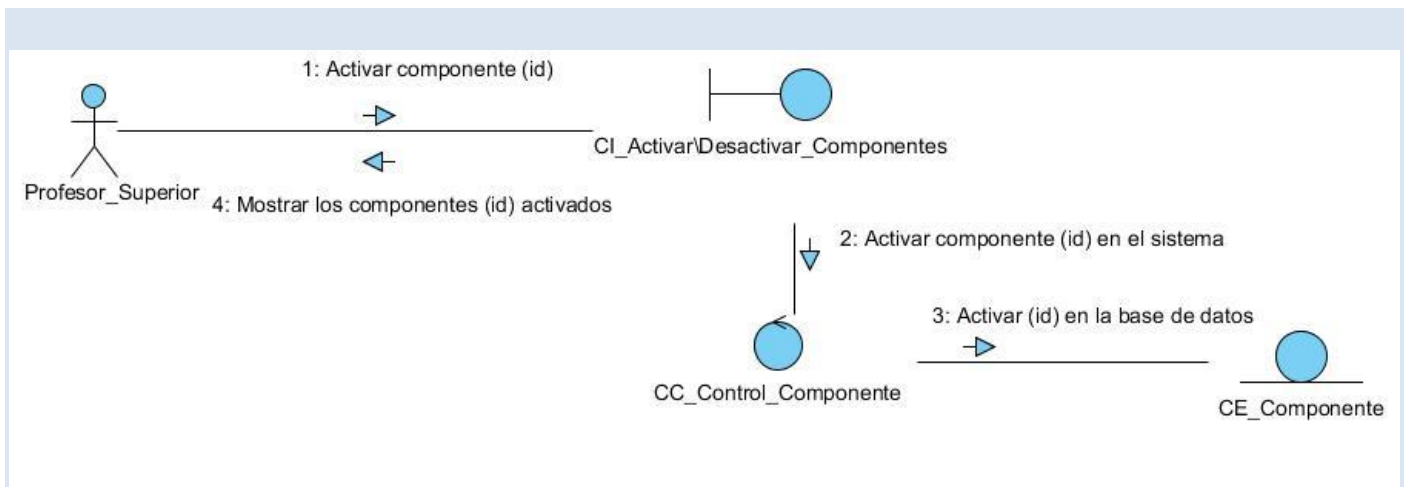
Anexo 3. Figura 1. Diagrama de clases de análisis de la interfaz principal

5.4 Anexo 4. Diagramas de colaboración

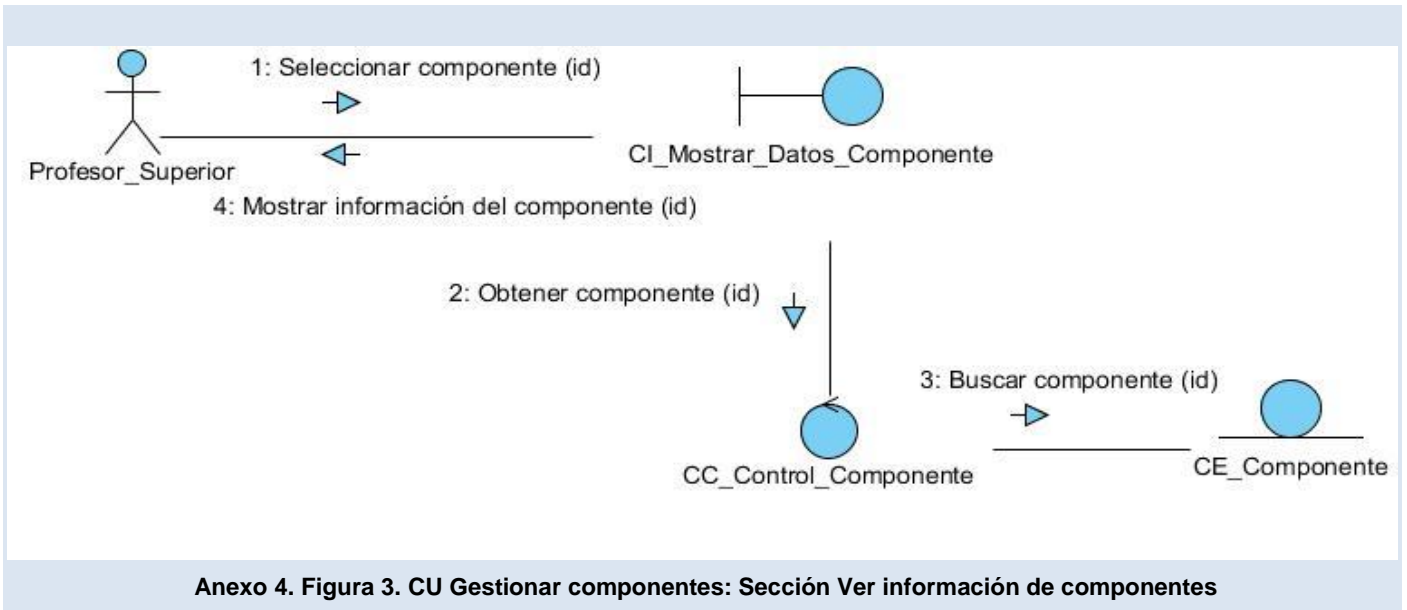
Resto de diagramas de colaboración del CU Gestionar Parámetros.



Anexo 4. Figura 1. CU Gestionar componentes: Sección Eliminar componentes.

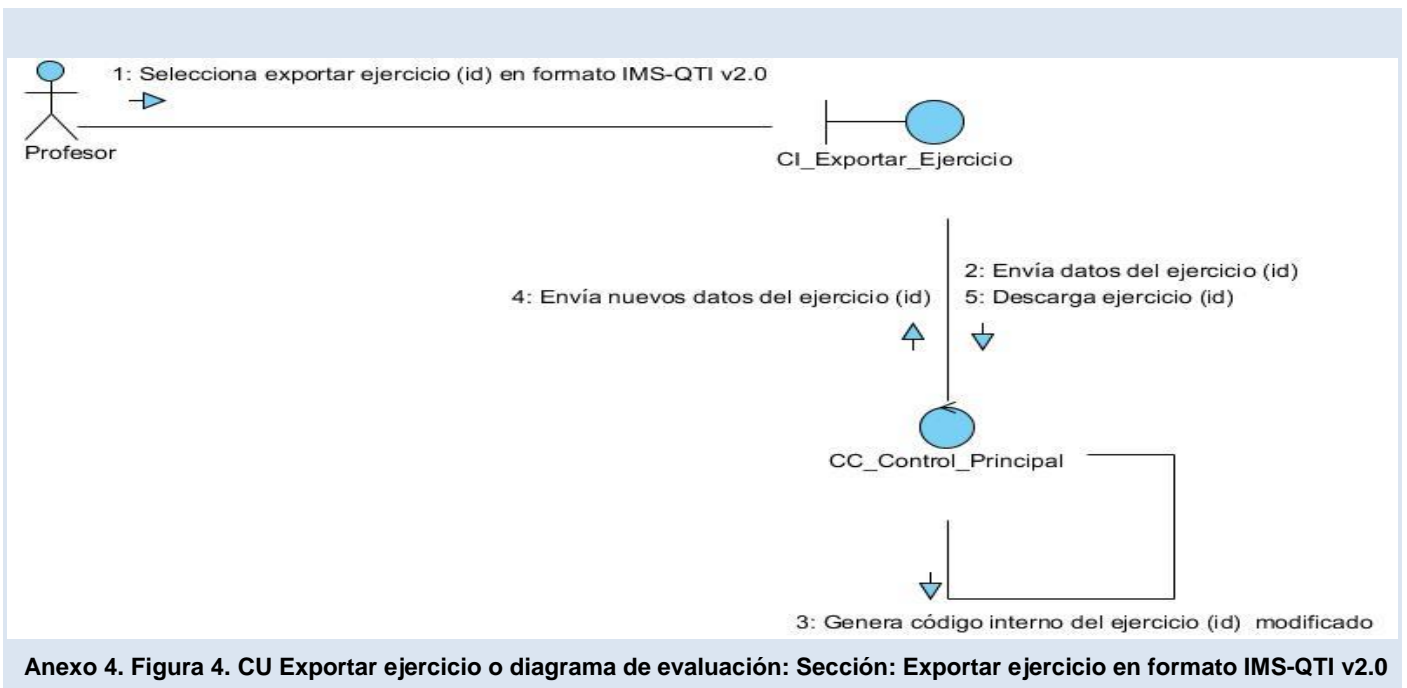


Anexo 4. Figura 2. CU Gestionar componentes: Sección Activar componentes.

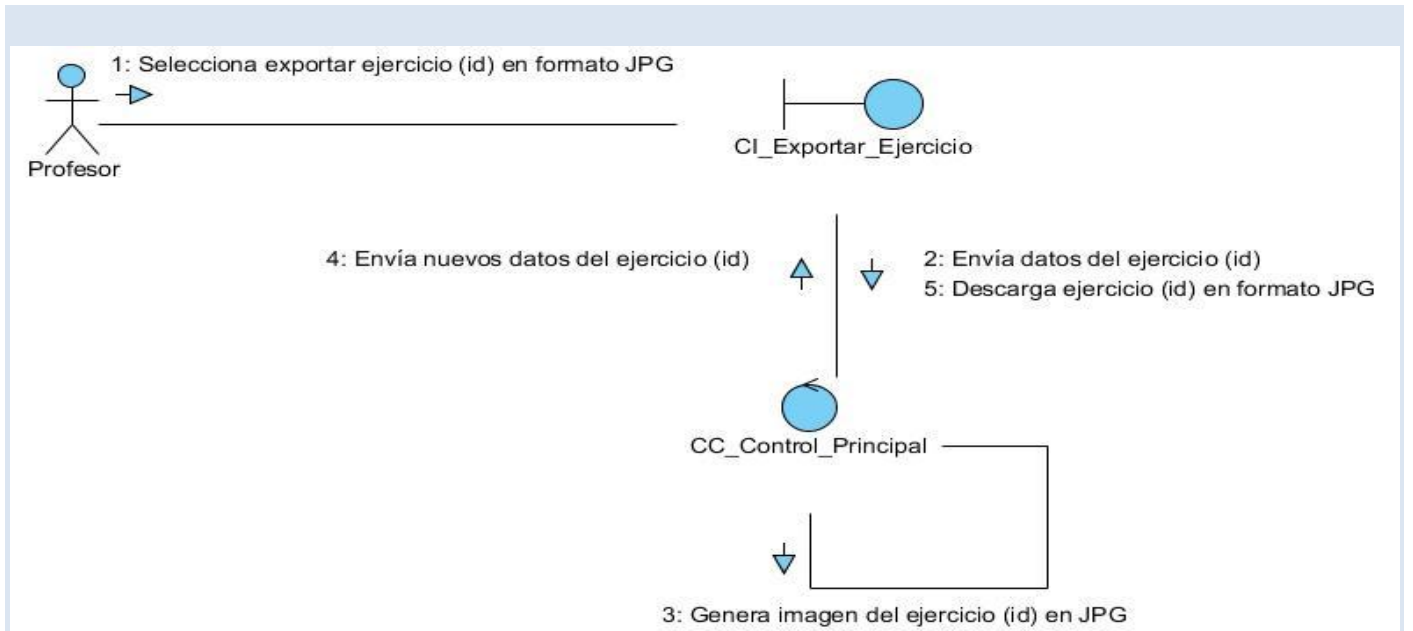


Anexo 4. Figura 3. CU Gestionar componentes: Sección Ver información de componentes

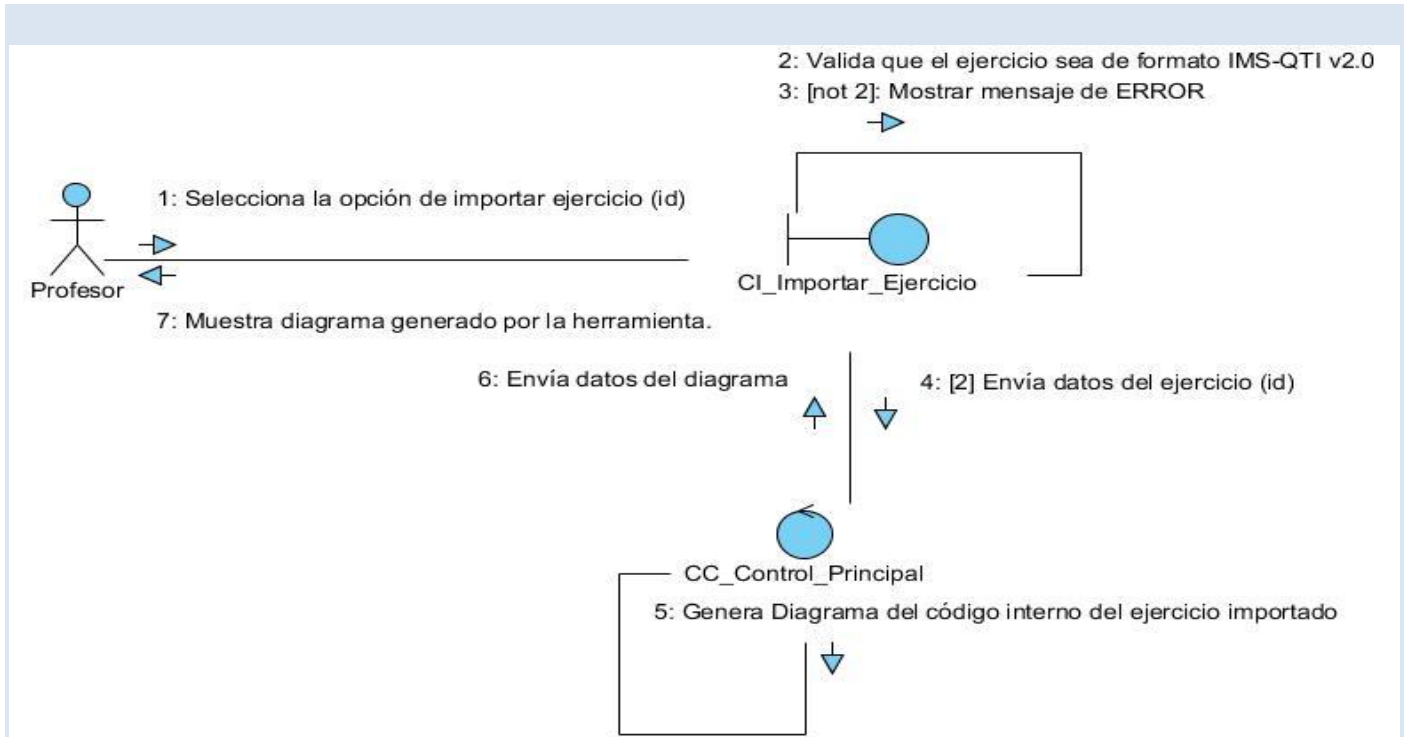
Diagramas de colaboración por secciones de la Interfaz Principal.



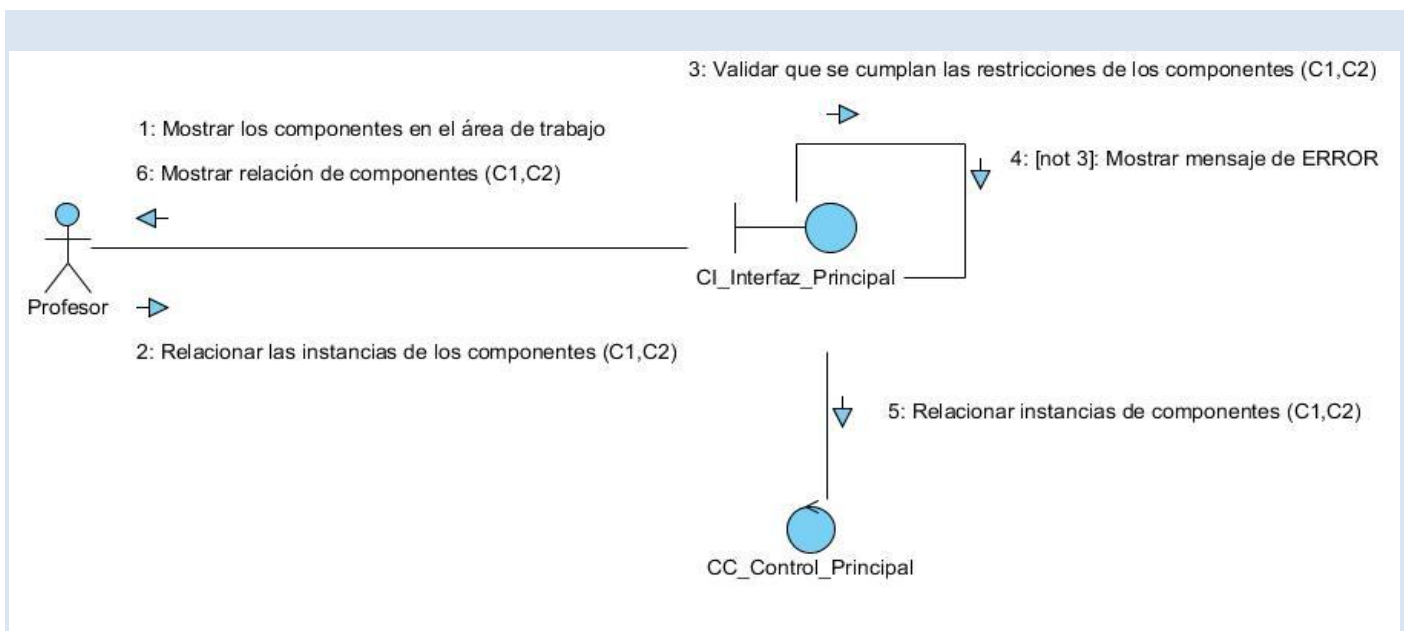
Anexo 4. Figura 4. CU Exportar ejercicio o diagrama de evaluación: Sección: Exportar ejercicio en formato IMS-QTI v2.0



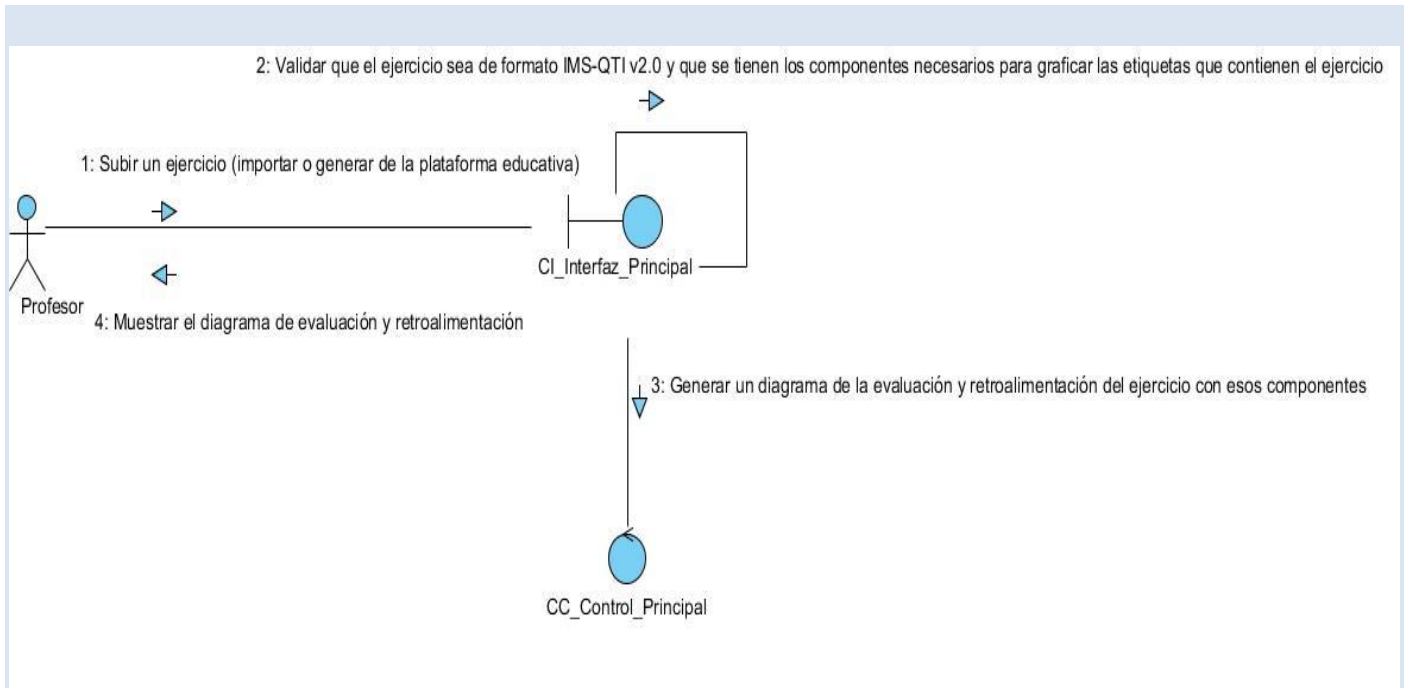
Anexo 4. Figura 5. CU Exportar ejercicio o diagrama de evaluación: Sección: Exportar diagrama de evaluación en formato JPG



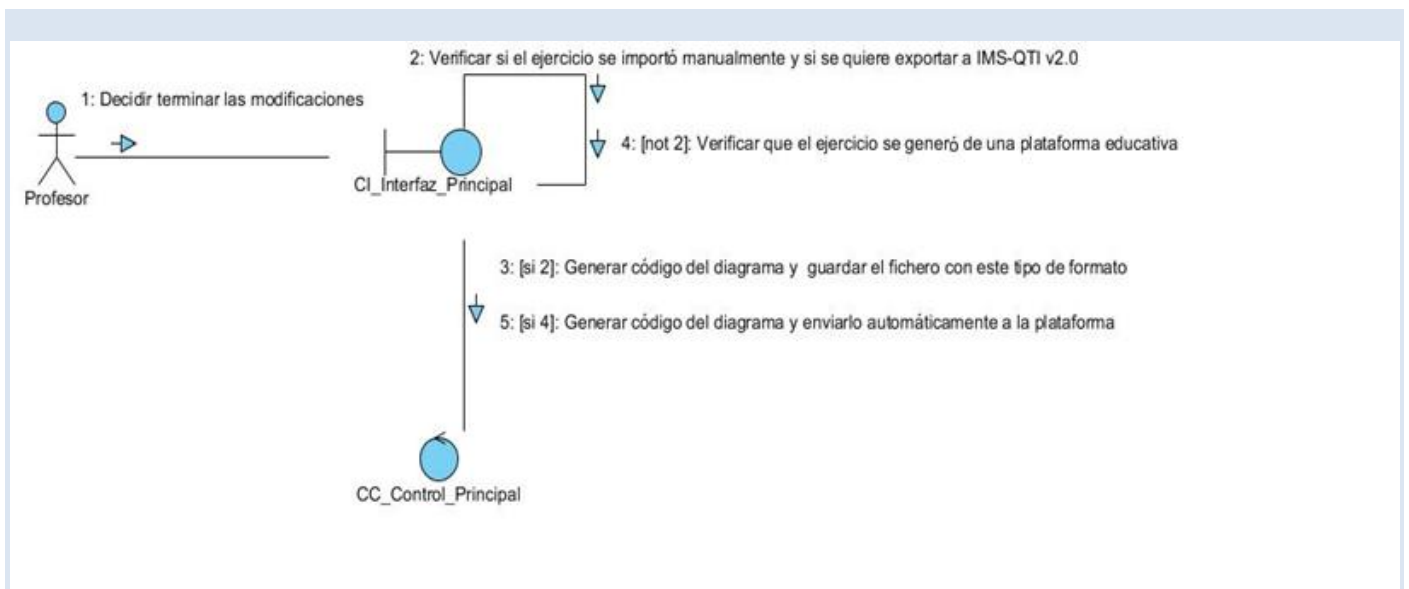
Anexo 4. Figura 6. CU Importar ejercicio: Sección: Importar ejercicio



Anexo 4. Figura 7. CU Modificar diagrama de evaluación: Sección: Relacionar Instancias de componentes



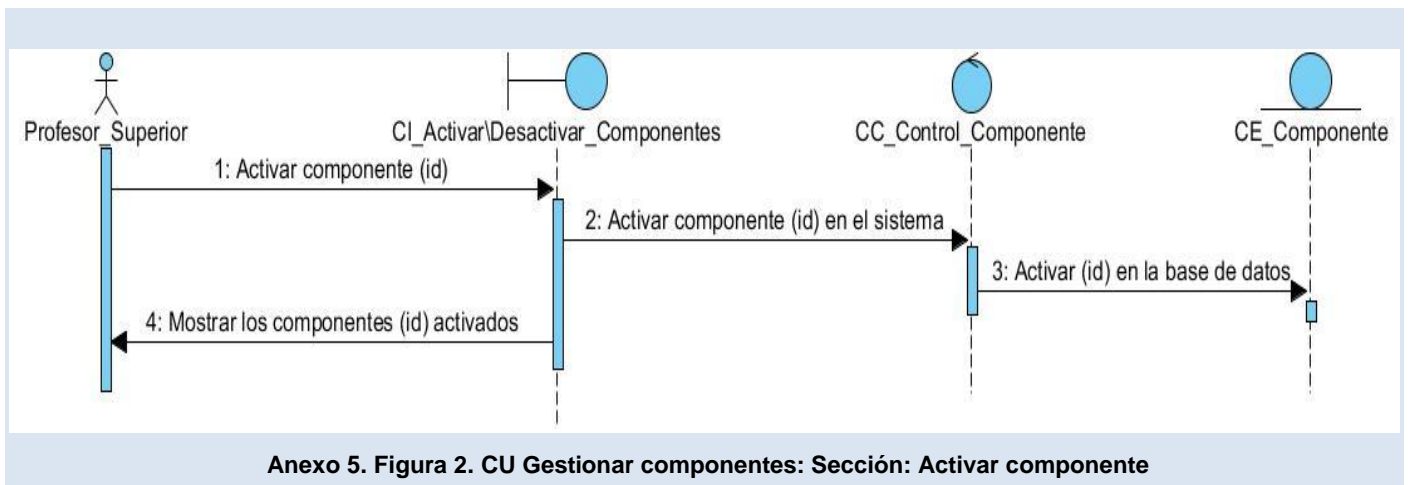
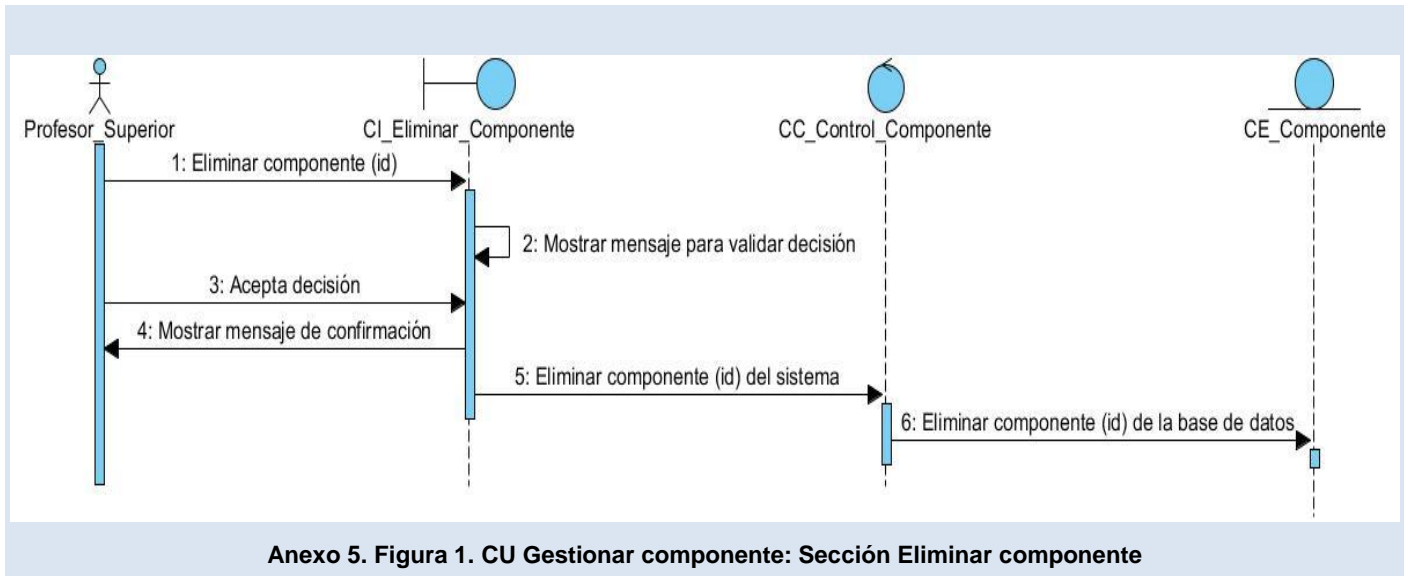
Anexo 4. Figura 8. CU Generar diagrama de evaluación

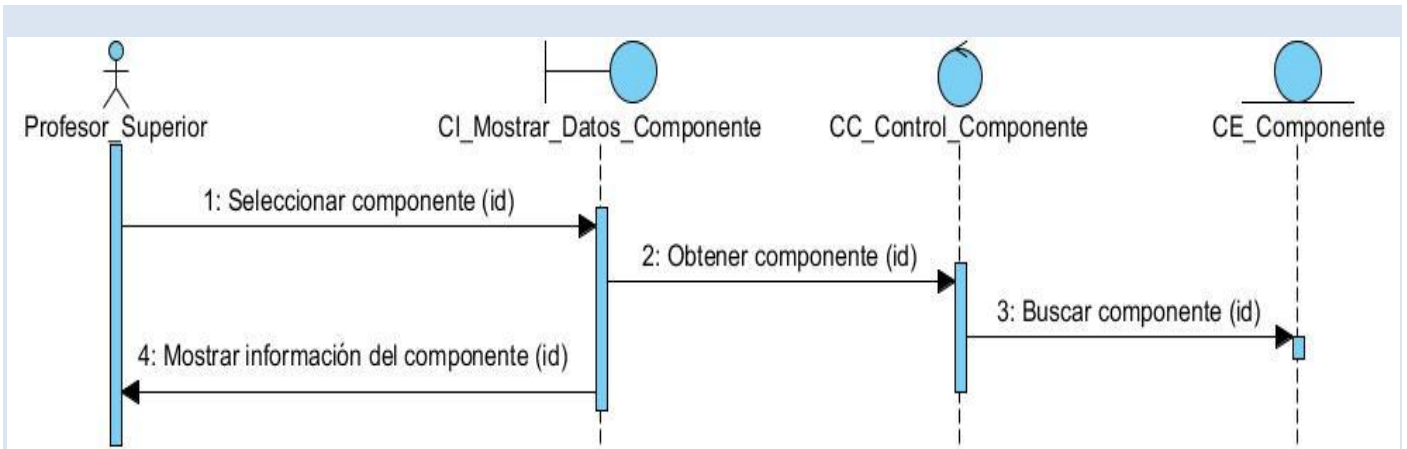


Anexo 4. Figura 9. CU Generar código del diagrama de evaluación

5.5 Anexo 5. Diagramas de secuencia

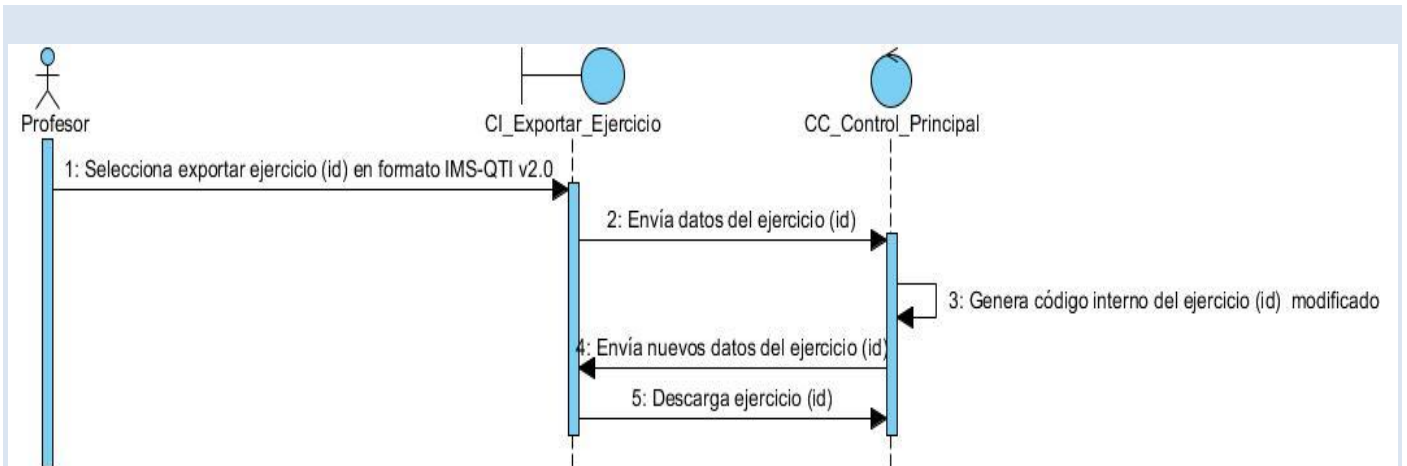
Diagrama de secuencia del resto del CU Gestionar Parámetros.



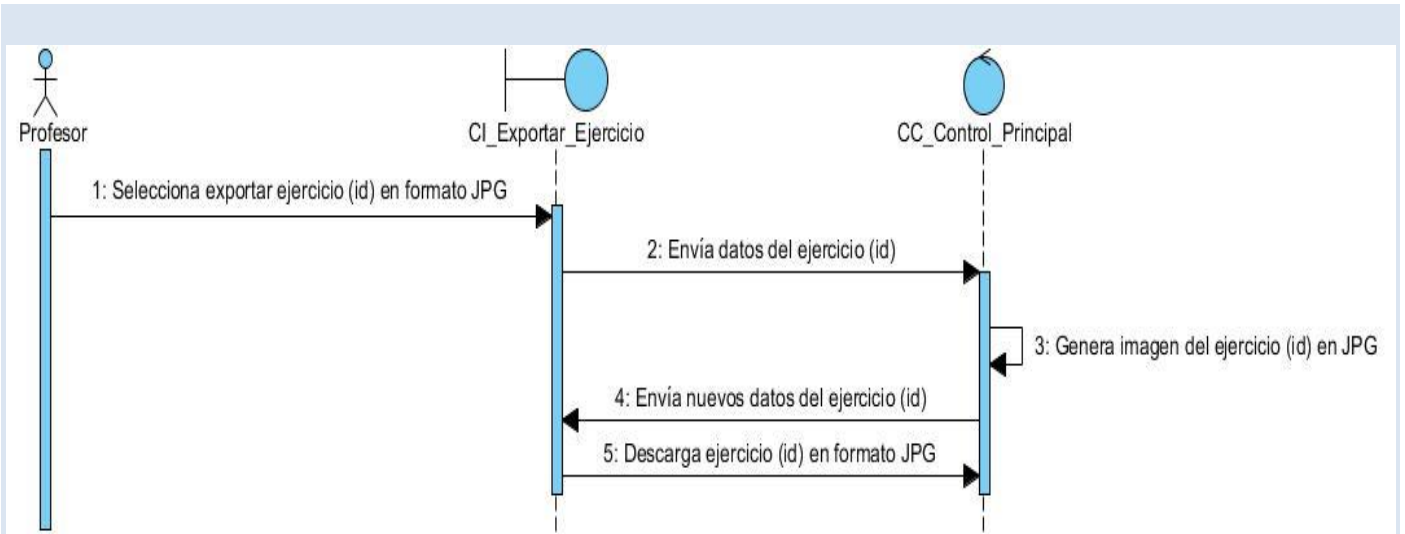


Anexo 5. Figura 3. CU Gestionar componente: Sección: Ver información del componente

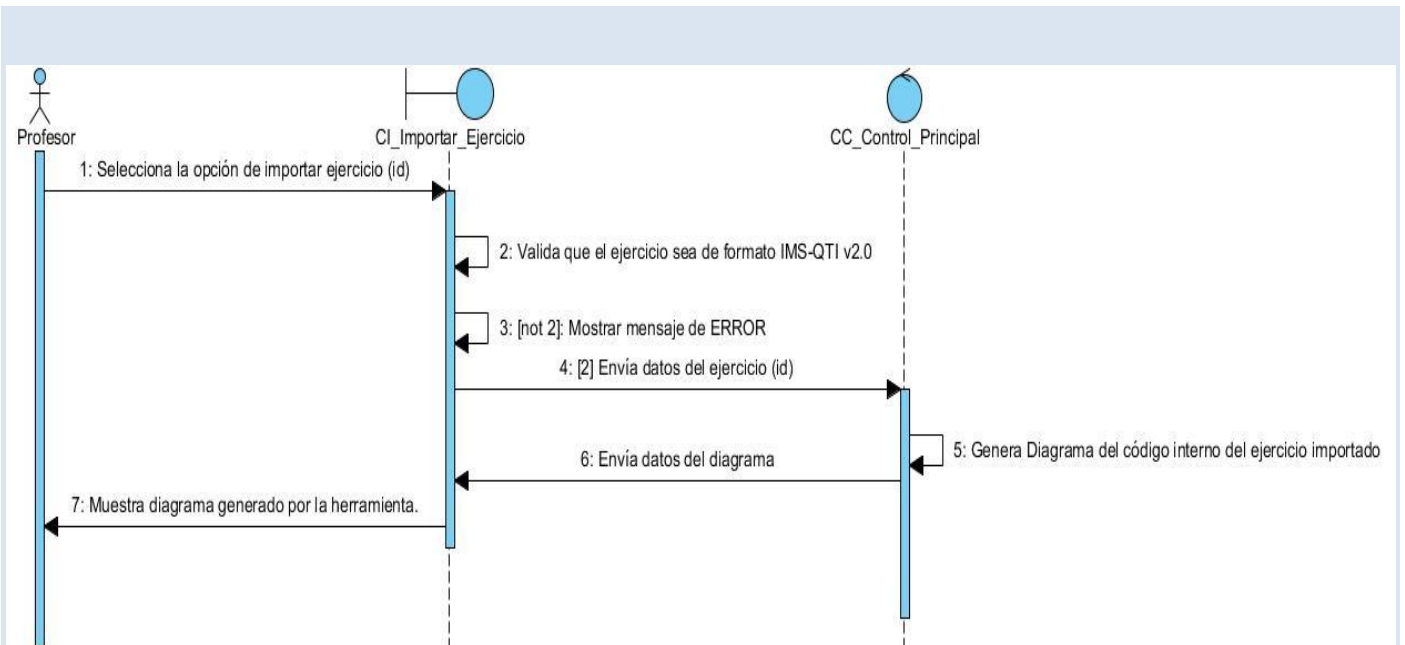
Diagramas de secuencia de la Interfaz Principal por secciones.



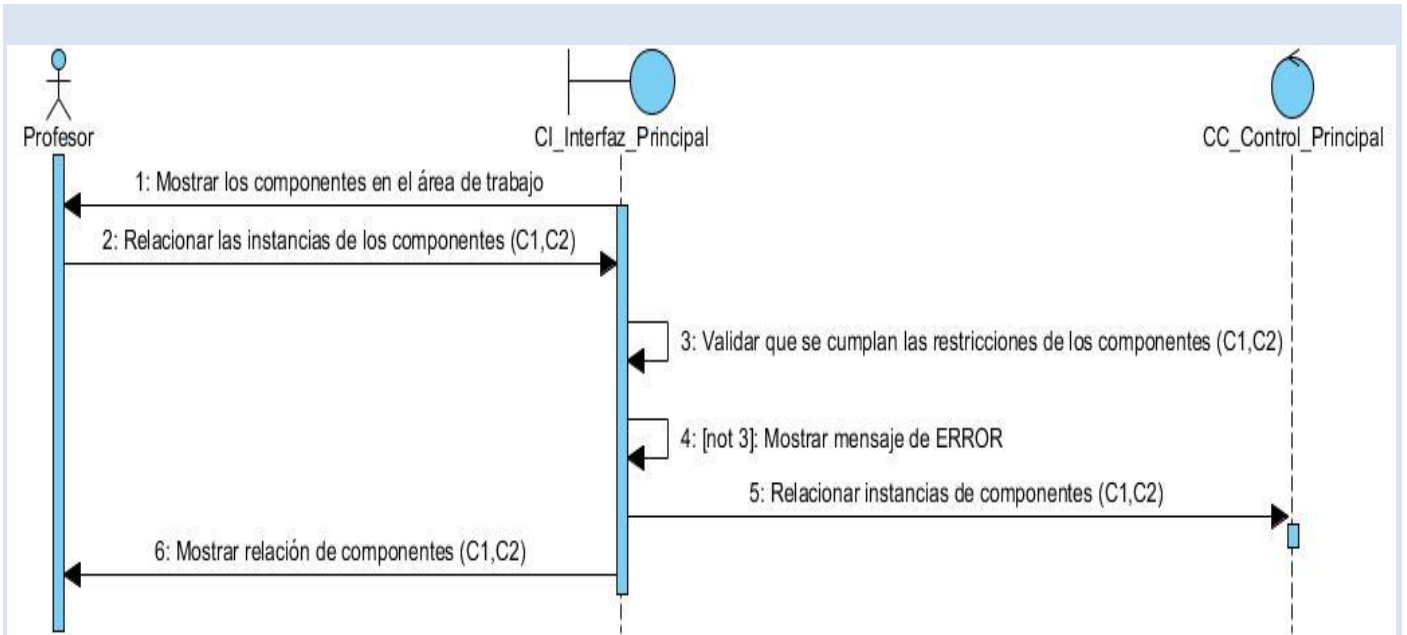
Anexo 5. Figura 4. CU Exportar ejercicio o diagrama de evaluación: Sección: Exportar ejercicio en formato IMS-QTI v2.0



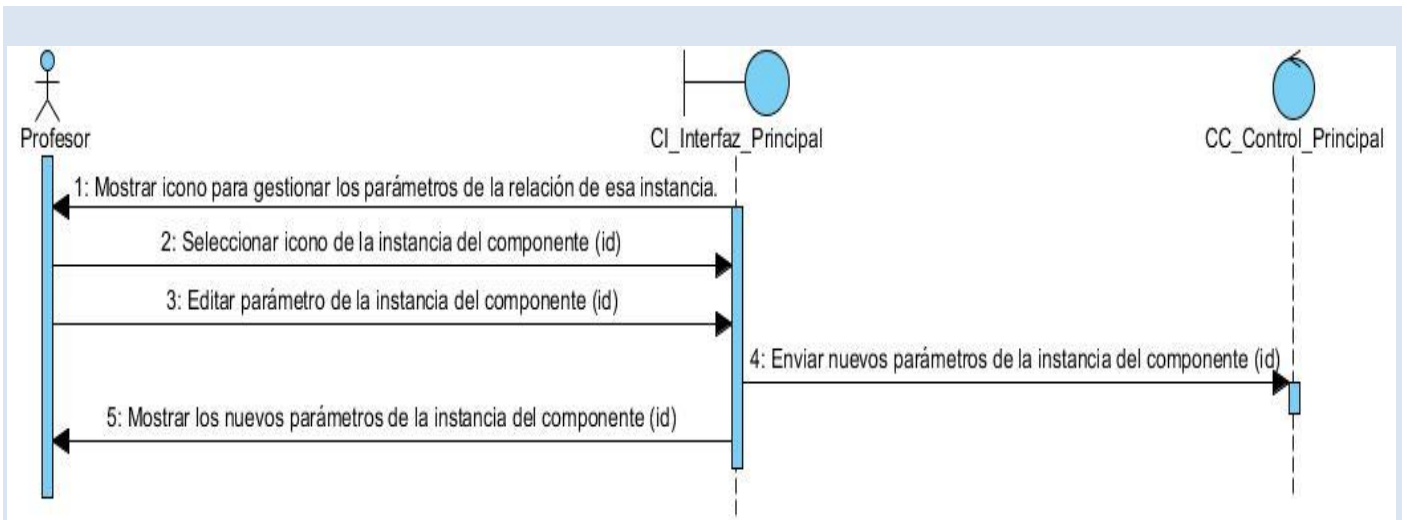
Anexo 5. Figura 5. CU Exportar ejercicio o diagrama de evaluación: Sección: Exportar ejercicio en formato JPG



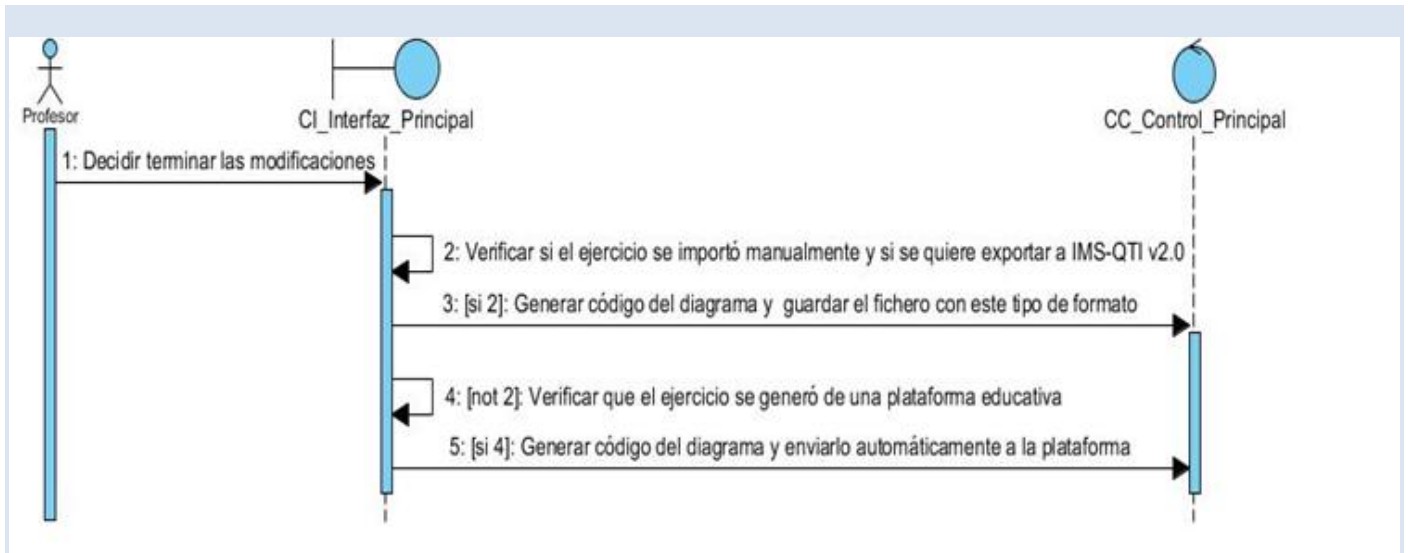
Anexo 5. Figura 6. CU Importar ejercicio: Sección: Importar ejercicio



Anexo 5. Figura 7. CU Modificar diagrama de evaluación: Sección: Relacionar las instancias de los componentes

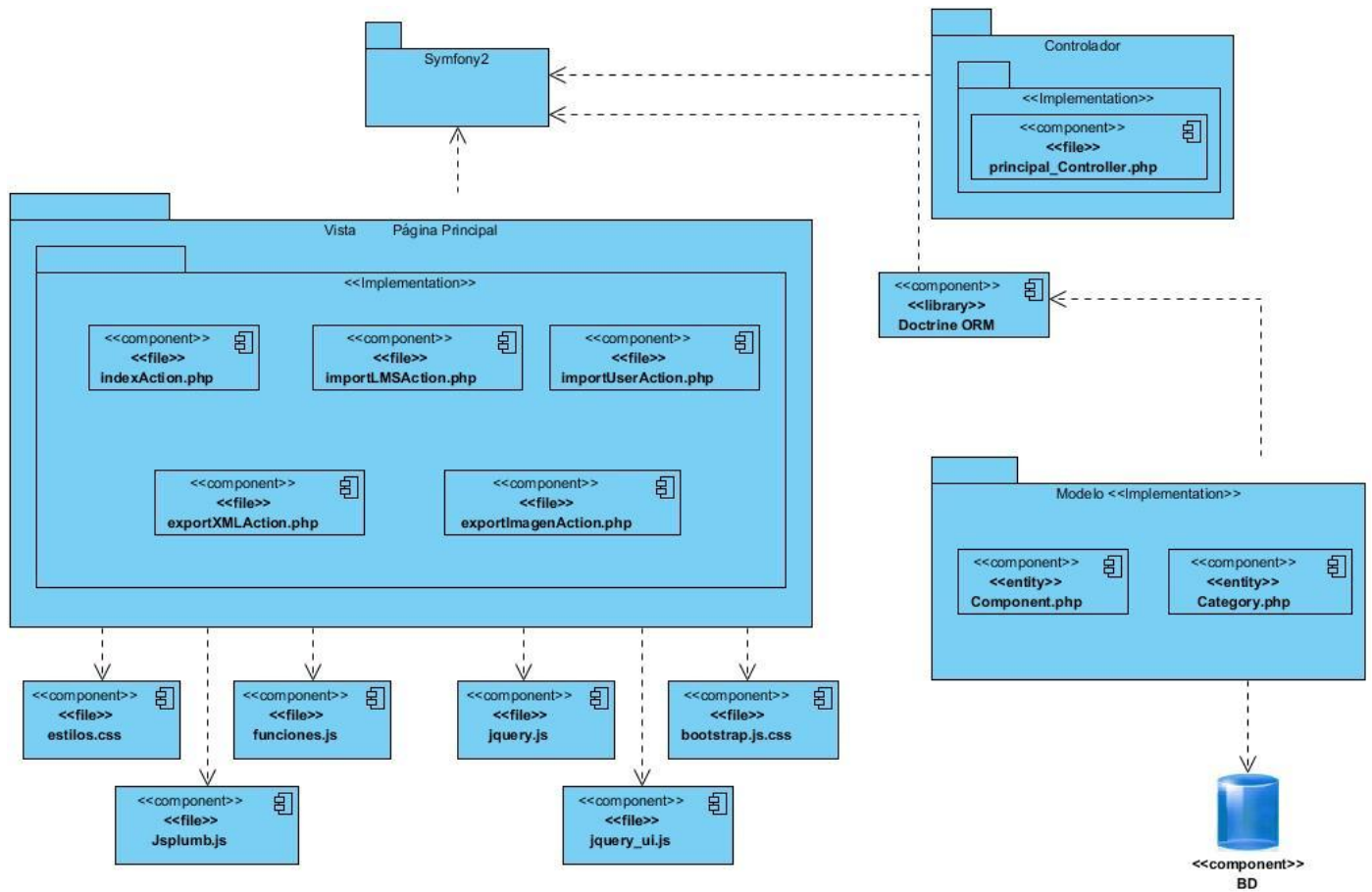


Anexo 5. Figura 8. CU Modificar diagrama de evaluación: Sección: Gestionar parámetros

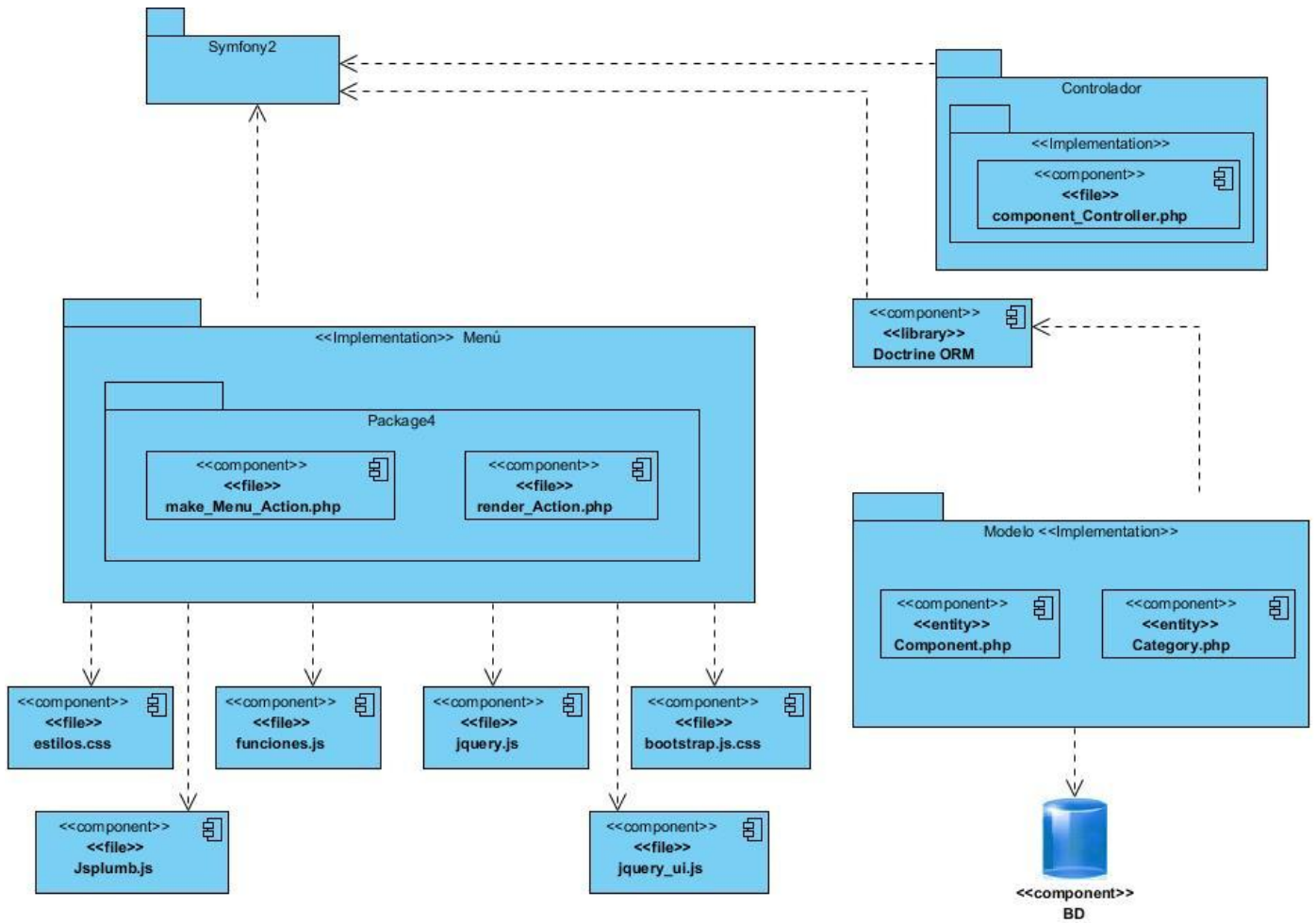


Anexo 5. Figura 9. CU Generar código a partir del diagrama de evaluación

5.6 Anexo 6. Diagramas de Componentes Interfaz Principal



Anexo 6. Figura 1. Diagrama de componente de la interfaz principal



Anexo 6. Figura 2. Diagrama de componente de la interfaz principal

5.7 Anexo 7. Métricas de diseño para la validación de la herramienta PERA (IMS-QTI v2.0)

Son varios los puntos de vista relacionados con la calidad del *software*. Las métricas de diseño a nivel de componentes se concentran en las características internas de los componentes del *software* con medidas que pueden ayudar al desarrollador a juzgar la calidad de un diseño a nivel de componente. Las métricas se centran en cuantificar tanto la complejidad, como la funcionalidad y eficiencia inmersa en el desarrollo de *software*. Inclina sus objetivos a mejorar la comprensión de la calidad del producto, a estimar la efectividad del proceso y mejorar la calidad del trabajo.

Las métricas empleadas están diseñadas para evaluar los siguientes atributos de calidad:

- Responsabilidad. Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.
- Complejidad de implementación. Consiste en el grado de dificultad que tiene implementado un diseño de clases determinado.
- Reutilización. Consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de *software*.
- Acoplamiento. Consiste en el grado de dependencia o interconexión de una clase o estructura de clase, con otras, está muy ligada a la característica de Reutilización.
- Complejidad del mantenimiento. Consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de *software*. Puede influir indirecta, pero fuertemente en los costes y la planificación del proyecto.
- Cantidad de pruebas. Consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad (Unidad) del producto (componente, módulo, clase, conjunto de clases, etc.) diseñado.

Tamaño operacional de clase (TOC)

Está dado por el número de métodos asignados a una clase y evalúa los siguientes atributos de calidad:

- Responsabilidad: Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.

- Complejidad de implementación: Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
- Reutilización: Un aumento del TOC implica una disminución del grado de reutilización de la clase.

Relaciones entre clases (RC)

Están dadas por el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad:

- Acoplamiento: Un aumento del RC implica un aumento del Acoplamiento de la clase.
- Complejidad de mantenimiento: Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.
- Reutilización: Un aumento del RC implica una disminución en el grado de reutilización de la clase.
- Cantidad de pruebas: Un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

A continuación se explica las métricas que se toman en cuenta para el análisis de TOC y RC

Anexo 7: Tabla 1. Métricas TOC

Métrica Tamaño operacional de clase (TOC)		
Atributo	Categoría	Criterio
Responsabilidad	Baja	\leq Prom.
	Media	Entre Prom y 2^* Pom.
	Alta	$> 2^*$ Prom.
Complejidad implementación	Baja	\leq Prom.
	Media	Entre Prom y 2^* Pom.
	Alta	$> 2^*$ Prom.
Reutilización	Baja	$> 2^*$ Prom.
	Media	Entre Prom y 2^* Pom.
	Alta	\leq Prom.

Anexo 7: Tabla 2. Métricas RC

Métrica de Relaciones entre clase (RC)		
Atributo	Categoría	Criterio
Acoplamiento	Ninguna	0
	Baja	1
	Media	2
	Alta	> 2
Complejidad de mantenimiento	Baja	$\leq PO$
	Media	Entre PO y $2 * PO$
	Alta	$> 2 * PO$
Reutilización	Baja	$> 2 * PO$
	Media	Entre PO y $2 * PO$
	Alta	$\leq PO$
Cantidad de pruebas	Baja	$\leq PO$
	Media	Entre PO y $2 * PO$
	Alta	$> 2 * PO$

Anexo 7: Tabla 3. Promedios generados por los valores que toman TOC y RC

	Promedio generado por la cantidad de clases	
	TOC	RC
Total de clases	7	7
Total	80	5
Promedio	11,42857143	0,714285714
2* Pom.	22,85714286	1,428571429

Anexo 7: Tabla 4. Matriz de las clases que contiene la implementación de PERA (IMS-QTI v2.0)

Matriz de las clases de la aplicación, cantidad de métodos y sus respectivas relaciones			
No.	Clase	Cantidad de Procedimientos	Cantidad de Relaciones
1	XLECategory	4	0
2	XLEComponent	21	0
3	PrincipalController	10	1
4	ManagerXML	9	0
5	AdminController	10	1
6	ComponentController	12	1
7	ManagerParent	14	2

Anexo 7: Tabla 5. Valores que toman TOC y RC después de ser medidos con las métricas de cada uno

Valores que toman los TOC y RC después de ser medidos teniéndose en cuenta la cantidad de métodos de cada clase para los TOC y la cantidad de relaciones para RC						
TOC			RC			
Responsabilidad	Complejidad	Reutilización	Acoplamiento	Complejidad de mantenimiento	Reutilización	Cantidad de pruebas
Baja	Baja	Alta	Ninguna	Baja	Alta	Baja
Media	Media	Media	Ninguna	Baja	Alta	Baja
Baja	Baja	Alta	Baja	Media	Media	Media
Baja	Baja	Alta	Ninguna	Baja	Alta	Baja
Baja	Baja	Alta	Baja	Media	Media	Media
Media	Media	Media	Baja	Media	Media	Media
Media	Media	Media	Media	Alta	Baja	Alta

5.8 Anexo 8. Diseño de casos de pruebas

CP Gestionar componentes, Exportar ejercicio o diagrama de evaluación, Importar ejercicio y demás secciones:

SC <Activar/Desactivar Componentes>

EC	Descripción	Componente	Respuesta del sistema	Flujo central
EC 1.1	El actor selecciona la opción de Activar/Desactivar componente.	V	Brinda la posibilidad de realizar las siguientes acciones: <ul style="list-style-type: none"> • Activar componentes. • Desactivar componentes. 	El profesor superior accede a gestionar componentes/Componentes/Mostrar/Activar componente.
EC 1.2	Activa los componentes deseados mediante la selección de los mismos	V	Muestra los componentes en el panel de componentes.	El profesor superior accede a gestionar componentes/Componentes/Mostrar/Activar componente.
EC 1.3	Desactiva los componentes deseados mediante la selección de los mismos	V	Muestra los componentes en el panel de componentes.	El profesor superior accede a gestionar componentes/Componentes/Mostrar/Desactivar componente.

SC <Eliminar Componentes>

EC	Descripción	Componente	Respuesta del sistema	Flujo central
EC 1.1	El actor selecciona opción	V	Muestra el siguiente mensaje para validar decisión: "Esta seguro	El profesor superior accede a gestionar componentes/Componentes/Mostrar/Eliminar componente.

	eliminar componente.		que quiere eliminar el componente". Permite realizar las acciones: <ul style="list-style-type: none"> • Aceptar • Cancelar 	
EC 1.2	Selecciona "Aceptar"	V	Muestra un mensaje de validación: "Se eliminó correctamente el componente". Elimina el componente de la base de datos de componentes y del sistema.	El profesor superior accede a gestionar componentes/Componentes/Mostrar/Desactivar componente/Si.
EC 1.3	Selecciona la opción "Cancelar"	NA	No ocurre ninguna acción.	El profesor superior accede a gestionar componentes/Componentes/Mostrar/Desactivar componente/No.

SC <Ver información de un componente>

EC	Descripción	Componente	Respuesta del sistema	Flujo central
EC 1.1	Selecciona el componente y permite acceder a la información del mismo.	NA	Muestra los siguientes datos de un componente: <ul style="list-style-type: none"> • Versión • Descripción • Dependencia • Nombre Permite la opción de Cerrar.	El profesor superior accede a gestionar componentes/Componentes/Mostrar/Datos.

EC 1.2	Visualiza los datos del componente.	NA	Selecciona opción "Cancelar". Y se muestra la vista anterior.	El profesor superior accede a gestionar componentes/Componentes/Mostrar/Cancelar.
-----------	-------------------------------------	----	---	---

SC <Exportar ejercicio o diagrama de evaluación>

EC	Descripción	Nombre	Formato	Respuesta del sistema	Flujo central
EC 1.1	El caso de prueba se inicia cuando el actor selecciona la opción de exportar un ejercicio.	V	V	Muestra los formatos que existen para exportar el ejercicio: <ul style="list-style-type: none"> • IMS-QTI v2.0 • JPG Permite: <ul style="list-style-type: none"> • Exportar • Cancelar petición 	El profesor termino de modificar el ejercicio/Selecciona la opción Exportar ejercicio/Mostrar formatos.
EC 1.2	Selecciona exportar el ejercicio en formato IMS-QTI v2.0.	V	V	Permite seleccionar el lugar donde deseas guardar el ejercicio o el diagrama.	El profesor termino de modificar el ejercicio/Selecciona la opción Exportar ejercicio/Mostrar formatos/ IMS-QTI v2.0.
EC 1.3	Selecciona el lugar donde guardará el ejercicio o diagrama de evaluación.	V	V	Exporta el ejercicio o el diagrama en el formato que se especificó.	El profesor termino de modificar el ejercicio/Selecciona la opción Exportar ejercicio/Mostrar formatos/ IMS-QTI v2.0.
EC 1.4	Selecciona exportar el diagrama de evaluación en JPG.	V	V	Exporta la imagen.	El profesor termino de modificar el ejercicio/Selecciona la opción Exportar

					ejercicio/Mostrar formatos/ JPG.
EC 1.5	Selecciona Cancelar	V	V	No realiza ninguna acción.	El profesor termino de modificar el ejercicio/Selecciona la opción Exportar ejercicio/Mostrar formatos/ Cancelar Petición

SC <Importar ejercicio>

Escenario	Descripción	Ejercicio	Respuesta del sistema	Flujo central
EC 1.1	El caso de prueba se inicia cuando el actor selecciona la opción abrir.	V	Brinda la opción de buscar el ejercicio que se quiere importar. Permite realizar las acciones: • Subir • Cancelar	El profesor accede a importar un ejercicio/Buscar Ejercicio
EC 1.2	Busca el ejercicio mediante la opción examinar	V	Permite realizar la acción: • Abrir • Cancelar	El profesor accede a importar un ejercicio/Buscar Ejercicio.
EC 1.3	Selecciona el ejercicio, realiza la acción abrir y sube el ejercicio.	V	Válida que el ejercicio sea de formato IMS-QTI v2.0. Si el ejercicio es en formato IMS-QTI v2.0, genera un diagrama de evaluación a partir del código interno de ese ejercicio importado. Muestra el diagrama en la herramienta y permite modificar el diagrama de evaluación del ejercicio.	El profesor accede a importar un ejercicio/Buscar Ejercicio/Importar Ejercicio.

ANEXO 8

EC 1.4	Si no es de formato IMS-QTI v2.0	V	Muestra un mensaje de error: "Este formato no está validado por el sistema, verifique nuevamente".	El profesor accede a importar un ejercicio/Buscar Ejercicio/Importar Ejercicio.
EC 1.5	Selecciona Cancelar	V	No realiza ninguna acción.	El profesor accede a importar un ejercicio/Buscar Ejercicio/Cancelar.

5.9 Anexo 9. Encuesta para profesores

Existe una escasa participación de los profesores a la hora de formular e implementar la forma de evaluar y retroalimentar un determinado ejercicio, ya que es un proceso realizado por los programadores. Ellos son los que generan el ejercicio y posteriormente la plataforma tiene un mecanismo para que esta nota y retroalimentación llegue al estudiante evaluado.

Para transformar esta situación se ha desarrollado una herramienta que ayude al profesor a personalizar la evaluación y retroalimentación que tiene un ejercicio. La herramienta generaría un diagrama de evaluación a partir del código existente, y entonces el profesor modificaría ese diagrama de evaluación; así este podría interpretar la forma que desea que se evalúe y retroalimente el ejercicio seleccionado.

1. Elaboración de la evaluación y retroalimentación de un determinado ejercicio en línea.

¿Si utiliza una plataforma educativa para evaluar a sus estudiantes le gustaría poder personalizar la evaluación y retroalimentación de los ejercicios que está evaluando?

No.	Respuestas	Selección
1	Si	
2	No	

Marque con una X la frecuencia en un mes, con que realiza un cuestionario o cualquier test evaluativo a sus estudiantes.

No.	Frecuencia	Selección
1	Nunca	
2	A veces	
3	A menudo	

Marque con una X a través de qué herramientas realiza los cuestionarios o ejercicios evaluativos a sus estudiantes.

Correo electrónico

Entorno de Aprendizaje (EVA)

___ Otro Moodle: ¿Cuál? o ¿Cuáles? _____

___ Ninguno

___ Otra vía: _____

Marque con una X que nivel de aprendizaje tiene en cuanto algoritmia.

No.	Nivel	Selección
1	Poco	
2	Mucho	
3	Ninguno	

¿Conoce alguna herramienta que permita al profesor personalizar la evaluación de los estudiantes?

___ Sí ___ No

Si la respuesta es positiva, mencione alguna de ellas:

Cree Ud. necesario que su Universidad haga uso de alguna herramienta que permita personalizar la evaluación y retroalimentación de un determinado ejercicio que se evalúe por este medio.

___ Si ___ No

2. Calidad de las evaluaciones y retroalimentaciones que se emiten en herramientas de uso educativo.

Marque las opciones que considere necesarias y que puedan mejorar la calidad de las evaluaciones y retroalimentaciones, las cuales se generan en los ejercicios evaluados.

___ Poder realizar diferenciaciones de evaluaciones y retroalimentación en los ejercicios evaluados según el estudiante que esté realizando el ejercicio.

___ Poder retroalimentar a los estudiantes de forma individual según sus preferencias y formas de adquirir el conocimiento.

___ Brindar la posibilidad de ver y emitir valoraciones a los contenidos de los recursos educativos.

__ Posibilidad de que los usuarios evaluados puedan rectificar sus notas mediante una retroalimentación adecuada.

__ Permitir realizar sugerencias a los profesores que evalúen los ejercicios, acerca de las preferencias de los estudiantes, para emitir una retroalimentación más efectiva.

3. **Cómo se comporta la herramienta PERA (IMS-QTI v2.0), después de interactuar con la misma.**

Cree Ud. que la herramienta posee gran utilidad en el proceso de evaluación y retroalimentación de los estudiantes.

__ Sí __ No

Si la respuesta es negativa, diga el por qué.