Universidad de las Ciencias Informáticas

Facultad 4

Título: Módulo competencia para la Plataforma Educativa Xauce Zera

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor (es): Rioger Hernández López

Tutor (es): Dr. C. María Caridad Valdés Rodríguez

Ing. Miguel Medina Ramírez

JUNIO DE 2018

PENSAMIENTO

“El hombre razonable se adapta al mundo: el insensato intenta hacer que el mundo se adapte a él. Por lo tanto, todo progreso depende del hombre insensato”.

George Bernard Shaw

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis que tiene por título : y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Rioger Hernández López

_____

Firma del Autor

Ing. Miguel Medina Ramírez

_____

Firma del Tutor

Dra. C. María Caridad Valdés

_____

Firma del Tutor

Datos del Contacto

Dr. C. María Caridad Valdés Rodríguez.

Correo electrónico: mvaldes@uci.cu

María Caridad Valdés Rodríguez. Doctora en Ciencias Pedagógicas. Profesora Titular. Imparte docencia en pre y postgrado de Metodología de la Investigación Científica, Comunicación Científica y Pedagogía en Tecnología. Líder del proyecto de investigación de Programa Nacional de Pedagogía en tecnología. Ha participado en varios eventos nacionales e internacionales, así como posee publicaciones en revistas de referencia Ha impartido docencia en Venezuela, Brasil, República Dominicana y Colombia y ha desarrollado Cursos en Eventos internacionales y Escuelas Internacionales de Verano e invierno. Actual Metodóloga del Centro de Innovación y Calidad de la Educación.

Ing. Miguel Medina Ramírez.

Correo electrónico: mmramirez@uci.cu

Miguel Medina Ramírez. Graduado de Ingeniero en Ciencias Informáticas en la universidad.

Agradecimientos

Agradecer es una acción gratificante, hoy me siento repleto de emoción al congratular a personas que han hecho posible que mi vida comience a ser diferente.

- A mi mamá, gracias por estar ahí, por ser mi principal fuente de inspiración para avanzar en la vida y por soportar mis malacrianzas, aunque seas peleona, eres la mejor del mundo.
- A mis tías por ser tan preocupadas y darme todo el apoyo necesario para convertirme en ingeniero.
- A mi hermano porque siempre ha sido y es mi orgullo, además porque junto a mí soñó con este momento.
- A mis dos grandiosos tutores por ser tan preocupados y brindarme tanto apoyo a pesar de sus responsabilidades.
- A mis amigos que me apoyaron siempre, y fueron mis hermanos durante estos 5 años y estuvieron cuando de veras los necesité, en especial a Alex y Raimaris.
- A todas las personas que de una forma u otra contribuyeron a que hiciera realidad este gran sueño, aunque sus nombres no aparezcan textualmente, sí están en mis agradecimientos diarios.

A todos, gracias y les regalo la satisfacción de que ya soy ***INGENIERO***.

Rioger Hernández López

Dedicatoria

A mi gran tesoro,
La que ha soñado con este momento,
Y representa mi mayor inspiración,
Mi Madre

Resumen

La evaluación es uno de los momentos más importantes dentro del proceso de enseñanza-aprendizaje y es más compleja cuando se trata de la evaluación por competencias en entornos virtuales. La evaluación por competencias es muy usada a nivel mundial, pues esta juzga en cuanto a la recogida, procesamiento y valoración de información que adquiere una persona para convertirla en conocimiento, además, mide el desempeño de los estudiantes en tareas asociadas a la práctica o a la realidad. La siguiente investigación tiene como objetivo desarrollar un módulo que facilite el desarrollo de competencias para el aprendizaje de los estudiantes en la Plataforma Educativa XAUCE Zera. Para la correcta realización de la evaluación por competencia en la plataforma fue necesario plantear todo el proceso desde que son creadas las competencias y asociadas a los instrumentos de evaluación. La Plataforma Educativa Zera es concebida para ser adaptable a diferentes entornos educativos, por lo que surge la necesidad de incorporarle este módulo de evaluación. Para dar cumplimiento a los objetivos planteados se definen los principales conceptos asociados a la evaluación por competencia, se investiga acerca de cómo se pone en práctica este tipo de evaluación en plataformas educativas con características similares a la Plataforma Educativa Zera, además se describe la metodología y herramientas a utilizar durante el desarrollo de la propuesta de solución. Como resultado se obtuvo un módulo competencia que cumple con lo definido durante la investigación y da solución a todos los objetivos propuestos.

PALABRAS CLAVE: evaluación por competencias, plataformas educativas

Índice de Contenido

INTRODUCCIÓN 1

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA 7

Introducción del capítulo 7

1.1 Definición y tipos de competencias 7

1.1.1 Tipos de competencias 9

1.2 Definición de evaluación por competencias 10

1.2.1 Características de la evaluación por competencias 11

1.3 Características generales de la Plataforma Educativa Xauce ZERA 12

1.4 Análisis a otras plataformas educativas con características similares 13

ATutor 13

Chamilo 14

Claroline 14

Dokeos 14

LRN 14

Moodle 15

1.5 Metodologías, tecnologías y herramientas empleadas 17

1.5.1 Metodologías de desarrollo 17

1.5.2 Metodología de desarrollo AUP-UCI 19

1.5.3 Marco de trabajo Xalix 20

1.5.4 Lenguaje de modelado: UML 2.0 20

1.5.5 Lenguajes de desarrollo 21

1.5.6 Framework de desarrollo 24

1.5.7 Herramienta de modelado UML: Visual Paradigm 8.0 26

1.5.8 Sistema de gestor de Base de Datos: PostgresSQL 9.1 26

1.5.9 Servidor Web 27

1.5.10 Entorno de desarrollo Integrado: NetBeans 8.0 28

1.6 Sistema de control de versiones 28

Conclusiones de capítulo 29

CAPÍTULO 2: DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA 30

Introducción del capítulo 30

2.1 Modelo de dominio 30
2.1.1 Conceptos del modelo de dominio 30
2.1.2 Diagrama de modelo de dominio 31
2.2 Descripción del sistema propuesto 32
2.3 Requisitos 32
2.3.1 Requisitos funcionales 32
2.3.2 Descripción de Requisitos 33
2.3.3 Requisitos no funcionales 37
2.4 Patrones de diseño 37
2.5 Descripción de las historias de usuario del sistema 37
2.6 Patrón arquitectónico 39
2.7 Modelo de diseño 40
2.7.1 Patrones de diseño 40
2.7.2 Diagramas de clases del diseño 41
2.8 Modelo de despliegue 42
2.9 Diseño de la base de datos 43
2.9.1 Diagramas entidad-relación 43
2.9.2 Descripción de las tablas de la base de datos 43
Conclusiones del capítulo 45
CAPÍTULO 3: EVALUACIÓN DE LA SOLUCIÓN 46
Introducción del capítulo 46
3.1 Modelo de implementación 46
3.1.1 Diagrama de componentes 46
3.1.2 Estándar de codificación 47
3.2 Pruebas de software 51
3.2.1 Niveles de pruebas 51
3.2.2 Métodos de prueba 52
3.2.3 Partición equivalente 52
3.2.4 Diseños de caso de prueba 52
3.2.5 Resultados obtenidos en las pruebas 55
Conclusiones del capítulo 56
CONCLUSIONES GENERALES 57
RECOMENDACIONES 58

REFERENCIAS BIBLIOGRÁFICAS 59

ANEXOS 61

ÍNDICE DE TABLAS

TABLA 1: DESCRIPCIÓN DE REQUISITOS 34
TABLA 2: HISTORIA DE USUARIO. INCLUIR COMPETENCIA 38
TABLA 3: DESCRIPCIÓN DE LA TABLA TB_COMPETENCE 43
TABLA 4: DESCRIPCIÓN DE LA TABLA TB_INDICATOR 44
TABLA 5: DESCRIPCIÓN DE LA TABLA TB_RCOMPETENCESPECIFICCONTENT 45
TABLA 6: CASO DE PRUEBA INCLUIR COMPETENCIA 53
TABLA 7: CASO DE PRUEBA EDITAR COMPETENCIA 54
TABLA 8: CASO DE PRUEBA EDITAR COMPETENCIA 55
TABLA 9:HISTORIA DE USUARIO MODIFICAR COMPETENCIA 62
TABLA 10: HISTORIA DE USUARIO VER COMPETENCIA 63
TABLA 11: HISTORIA DE USUARIO ELIMINAR COMPETENCIA 64
TABLA 12: HISTORIA DE USUARIO MOSTRAR COMPETENCIA DESARROLLADAS 65
TABLA 13: HISTORIA DE USUARIO MOSTRAR ESTADO DE LAS COMPETENCIAS DE LOS ESTUDIANTES 66

ÍNDICE DE FIGURAS

FIGURA 1: DIAGRAMA DE DOMINIO 31
FIGURA 2: DIAGRAMA DE DISEÑO DEL CASO DE USO GESTIONAR COMPETENCIA 41
FIGURA 3: DIAGRAMA DE DESPLIEGUE 42
FIGURA 4:MODELO ENTIDAD-RELACIÓN 43
FIGURA 5 DIAGRAMA DE COMPONENTES GESTIONAR COMPETENCIA 47
FIGURA 6: EJEMPLO DE CÓDIGO JAVASCRIPT 48
FIGURA 7: EJEMPLO DE CÓDIGO HTML 48
FIGURA 8: EJEMPLO DE CÓDIGO PHP 49
FIGURA 9: EJEMPLO DE CÓDIGO PHP 49
FIGURA 10: EJEMPLO DE CÓDIGO PHP 49
FIGURA 11: EJEMPLO DE CÓDIGO PHP 50
FIGURA 12: EJEMPLO DE CÓDIGO COMENTADO 50
FIGURA 13: EJEMPLO DE CÓDIGO COMENTADO 51
FIGURA 14:RESULTADOS OBTENIDOS DE LAS PRUEBAS 55
FIGURA 15: DIAGRAMA DE CASO DE USO PERSONALIZAR COMPETENCIA 61

FIGURA 16: DIAGRAMA DE CASO DE USO MOSTRAR COMPETENCIAS DESARROLLADAS 61
FIGURA 17:DIAGRAMA DE CASO DE USO MOSTRAR ESTADO DE COMPETENCIAS..... 62

INTRODUCCIÓN

En la actualidad las Tecnologías de la Información y las Comunicaciones (TIC) representan un eslabón vital en el desarrollo y progreso de la sociedad, permitiendo la interactividad, interconexión y digitalización de la información. El acelerado avance de las TIC ha influido considerablemente en el campo de la educación, creando nuevos hábitos de estudio, nuevas formas de enseñar y evaluar, surgiendo los Sistemas de Gestión del Aprendizaje.

Los Sistemas de Gestión del Aprendizaje o LMS (Learning Management System, por sus siglas en inglés) son software o herramientas informáticas instaladas en un servidor web que se emplean para administrar, distribuir y controlar las actividades de formación de los estudiantes de una institución u organización. Estos identifican y evalúan el aprendizaje individual de cada estudiante, siguen el progreso hacia el logro de los objetivos donde recogen y presentan el resultado. Su arquitectura y herramientas son apropiadas para las clases en línea.

Existe una gran variedad de LMS tales como Moodle, Sakai, Claroline y Chamilo entre otros, todos dirigidos a gestionar recursos y contenidos, así como materiales y actividades para la formación o enseñanza de un material determinado; administrar el acceso, controlar y hacer seguimiento del proceso de aprendizaje, y realizar evaluaciones. Uno de los tipos de evaluaciones que más se ponen de manifiesto en estos sistemas es la evaluación por competencia.

La evaluación es un proceso complejo pero inevitable. Es una fuerza positiva cuando sirve al proceso enseñanza-aprendizaje y se utiliza para identificar los puntos débiles y fuertes, para tender hacia una mejora. En el proceso de enseñanza y aprendizaje esta juega un papel fundamental. El docente debe tener claro que la evaluación se da como un proceso de recolección de datos para emitir un juicio final. Además, la evaluación determina los logros del sistema, la mediación docente y del alumno; por lo tanto, debe ser lo más confiable y aplicada con criterios técnicos y razonados. (1)

La Educación por Competencias en el marco de la formación pretende ser un enfoque integral que busca elevar el potencial de los individuos, de cara a las transformaciones que sufre el mundo actual y la sociedad contemporánea. Es un proceso mediante el cual se recopilan evidencias y se realiza un juicio de las mismas teniendo en cuenta criterios

preestablecidos para la retroalimentación en aras de mejorar la educación de los estudiantes. (2)

La educación por competencia es un proceso metacognitivo, se basa en criterios pertinentes al desempeño en el contexto educacional, busca articular lo cualitativo y lo cuantitativo, se centra en los aspectos esenciales del aprendizaje. Se orienta a evaluar las competencias en los estudiantes teniendo como referencia el desempeño de estos ante las actividades y problemas del contexto profesional, social, disciplinar e investigativo. Toma como referencia evidencias e indicadores, buscando determinar el grado de desarrollo de tales competencias en tres dimensiones (afectivo-motivacional, cognoscitiva y actuacional). Brinda retroalimentación en torno a fortalezas y aspectos a mejorar, por eso siempre tiene carácter formativo, independiente del contexto en que se lleve a cabo, siempre debe ser participativa, reflexiva y crítica. (1)

Cuando la evaluación se hace con fines de promoción y certificación, debe tenerse presente la discusión con los estudiantes y las posibilidades de revisarla para que se ajuste a las evidencias del proceso y de los aprendizajes obtenidos, siempre teniendo como referencia los indicadores previamente concertados en el proceso de normalización. La educación por competencias no es una tarea puntual sino un proceso que implica: (3)

- Definir con exactitud las competencias a evaluar con sus respectivas dimensiones.
- Construir los indicadores para evaluar las competencias de forma integral con criterios académicos y profesionales.
- Definir los tipos de evidencias que se deben presentar para llevar a cabo la evaluación.
- Establecer las estrategias e instrumentos con los cuales se llevará a cabo la evaluación.
- Analizar la información con base en los indicadores, determinar fortalezas y aspectos a mejorar, retroalimentar de forma oportuna a los estudiantes
- Generar un espacio de reflexión en ellos tanto sobre el proceso como en torno a los resultados de la evaluación, con la posibilidad de cambiar los resultados de acuerdo a los argumentos que ellos presenten. (3)

Cuba, a pesar de ser un país en vías de desarrollo, dedica cuantiosos recursos para dotar a la educación cubana de los beneficios que proporciona la educación por competencia. En la Universidad de Ciencias Informáticas (UCI), se encuentra el Centro de Tecnologías para la Formación (FORTES), cuya misión es desarrollar tecnologías que permitan ofrecer servicios y productos para la implementación de soluciones de formación aplicando las TIC a instituciones con diferentes modelos de formación y condiciones tecnológicas, garantizando la calidad de las soluciones y la formación de los recursos humanos a partir de investigaciones que combinen los elementos pedagógicos y tecnológicos más avanzados, integrando así los procesos de formación, producción e investigación.

En el Centro FORTES se ha desarrollado una Plataforma Educativa denominada ZERA que tiene como objetivo servir de apoyo al proceso de enseñanza-aprendizaje basada en una modalidad semipresencial, lo que flexibiliza la enseñanza, compartiendo la responsabilidad del aprendizaje entre el profesor y la plataforma

En la Plataforma Educativa Zera en su versión 1.0 a pesar de sus limitaciones, existía un proceso para gestionar competencias que la propia plataforma gestionaba donde era posible identificar las habilidades y valores que los educandos desarrollaban en su proceso docente, pero debido al avance tecnológico y al progreso continuo de las TIC en el ámbito educacional la Plataforma Educativa Zera evolucionó a una versión más acorde con las tendencias actuales del e-learning.

En esta última no existe un proceso de evaluación de las competencias donde los educadores puedan identificar con claridad las fortalezas y debilidades de los educandos de manera que a los docentes se les dificulta la emisión de juicios para evaluar de forma justa y objetiva respecto a los conocimientos, habilidades, valores y actitudes que el estudiante manifiesta en su proceso docente, no cuenta con una forma de determinar los logros y establecer el grado de formación de las competencias al final de determinados ciclos evaluativos.

Teniendo en cuenta lo anteriormente expuesto se plantea el siguiente **problema científico**: ¿Cómo contribuir al desarrollo de competencias en estudiantes para la mejora del aprendizaje de su proceso formativo en la Plataforma Educativa Zera?

Objeto de estudio: Desarrollo de competencias en plataformas educativas.

Campo de acción: Competencias en estudiantes para el aprendizaje en la Plataforma Educativa Xauce ZERA.

Objetivo: Desarrollar un módulo que facilite desarrollo de competencias en estudiantes para la mejora del aprendizaje de su proceso formativo en la Plataforma Educativa Zera.

Posibles resultados:

Al integrar el diseño teórico y metodológico de investigación, su argumentación en el desarrollo del presente trabajo, la implementación del módulo Competencias en estudiantes durante su proceso formativo para la Plataforma Educativa XAUCE ZERA 2.1, así como su validación mediante estrategia de pruebas adoptada, se espera alcanzar los siguientes resultados**:**

I. Documentación complementaria que incluya artefactos de análisis, diseño e implementación del módulo que facilite el desarrollo de competencias en los estudiantes en la Plataforma Educativa XAUCE ZERA v2.1 y permita a los profesores conocer en qué medida han sido desarrolladas.

II. Módulo que reconozca la gestión de competencias, su integración con los instrumentos de evaluación y su evaluación en la Plataforma Educativa XAUCE ZERA 2.1. Además, dicho módulo contará con un espacio que permitirá al profesor conocer en qué medida han sido desarrolladas las competencias por estudiantes.

Objetivos Específicos:

I. Determinar los referentes teóricos en que se fundamenta el desarrollo de las competencias para el aprendizaje de los estudiantes en plataformas educativas.
II. Elaborar una propuesta de solución que dará respuesta al problema de investigación.
III. Implementar las funcionalidades del módulo de desarrollo de competencia para el aprendizaje en estudiantes en la plataforma educativa Zera.
IV. Realizar pruebas a la solución propuesta para comprobar el cumplimiento de sus requerimientos.

Métodos teóricos:

Histórico – lógico: para la realización del estudio del estado del arte dirigido a investigar plataformas educativas que presentan soluciones similares.

Análisis – síntesis: para el estudio de las fuentes bibliográficas existentes referente al tema, que propicia la investigación de los elementos más importantes y necesarios para dar solución al problema planteado.

Inductivo – deductivo: para detectar las principales iniciativas de plataformas educativas en cuanto a la adaptación de contenidos y actividades de aprendizaje, las tecnologías y soluciones existentes y para determinar cuáles son las alternativas viables a incorporar en la presente investigación.

Hipotético – deductivo: para arribar a predicciones, las que posteriormente serán sometidas a verificaciones empíricas.

Métodos empíricos:

Estudio documental: empleado para la consulta de la literatura especializada en las temáticas afines de la investigación.

Observación participativa: se utiliza con la finalidad de registrar los resultados que se van obteniendo, referentes a cómo funciona el proceso de visualización de contenidos y los principales problemas asociados a este. El instrumento a utilizar será la guía de observación.

Estructura Capitular

El presente documento consta de tres capítulos, a continuación, se describe cada uno de ellos.

Capítulo I. Fundamentación teórica

En este capítulo se abordan todos los elementos teóricos que sustentan la investigación: conceptos asociados al dominio del problema como competencias, tipos de competencias y la evaluación de las competencias. Se presenta la metodología de desarrollo, así como todas las herramientas, tecnologías, lenguajes empleados para darle solución al problema planteado.

Capítulo II. Descripción de la solución propuesta

En este capítulo se realiza una descripción de todas las características de las funcionalidades a desarrollar. En el mismo se especifican los requisitos funcionales y requisitos no funcionales, descripción del modelo de dominio, y se describen los casos de uso del sistema y sus relaciones.

Capítulo III. Evaluación de la solución propuesta

En este capítulo se describen los elementos necesarios para la implementación, partiendo del resultado obtenido del diseño. Se muestra la organización de los componentes con sus relaciones lógicas a través del diagrama de componentes, quedando conformado el módulo *competencia*. Se describen pruebas de software tales como: pruebas de sistema y pruebas de aceptación, además, se mostrarán los casos de prueba y los resultados obtenidos en cada una de ellas.

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

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción del capítulo

En el presente capítulo se precisan los elementos para conformar el marco teórico relacionado con los aspectos definidos en el objeto de estudio. Se analizan los principales términos asociados al dominio de la investigación. Para dar cumplimiento a la correcta definición de estos puntos se realiza el estudio de soluciones similares y de las principales tendencias que se llevan a cabo en la adaptación de contenido y actividades de aprendizaje. Se destacan las principales características, tecnologías y buenas prácticas de los elementos asociados a la adaptación de contenido, con el fin de determinar la necesidad, el nivel de aceptación y la correcta forma de aplicación de estos elementos en la Plataforma Educativa ZERA.

1.1 Definición y tipos de competencias

Diversos han sido los autores que han definido el término competencia entre ellos se encuentran: Perrenoud, Lorenzo García, entre otros, a continuación, se muestra una selección de los conceptos que se adecuan con la investigación relacionada con las competencias en el ámbito educacional:

"Conjunto de disposiciones mentales del sujeto (conocimientos tácitos y explícitos, actitudes, habilidades, valores...) que le permiten llevar regularmente a cabo acciones observables y efectivas en un determinado dominio o contexto." (4)

"La capacidad de responder a demandas complejas y llevar a cabo tareas diversas de forma adecuada. Supone una combinación de habilidades prácticas, conocimientos, motivación, valores éticos, actitudes, emociones y otros componentes sociales y de comportamiento que se movilizan conjuntamente para lograr una acción eficaz." (5)

"Conjunto de conocimientos, destrezas y actitudes que ha de ser capaz de movilizar una persona, de forma integrada, para actuar eficazmente ante las demandas de un determinado contexto". (6)

Constituyen una habilidad o atributo personal de la conducta de un sujeto, que puede definirse como característica de su comportamiento, y, bajo la cual, el comportamiento orientado a la tarea puede clasificarse de forma lógica y fiable. (7)

Las competencias representan conjuntos de conocimientos, habilidades, disposiciones y conductas que posee una persona, que le permiten la realización exitosa de una actividad. (8)

Una competencia es una característica subyacente de un individuo, que está casualmente relacionada con un rendimiento efectivo o superior en una situación o trabajo, definido en términos de un criterio. (9)

Una competencia es una capacidad, susceptible de ser medida, necesaria para realizar un trabajo eficazmente, es decir, para producir los resultados deseados por la organización. El análisis de competencias tiene como objeto identificar los conocimientos, las destrezas, las habilidades y los comportamientos estimulantes que los empleados deben demostrar para que la organización alcance sus metas y objetivos. Para tener una competencia puede ser necesario, tal vez, sólo un tipo de conocimientos, o destreza, habilidad o comportamiento determinados, o bien puede requerir una combinación de todos ellos. (10)

La Norma Cubana reafirma en la consideración de competencias "conjunto sinérgico de conocimientos, habilidades, experiencias, sentimientos, actitudes, motivaciones, características personales y valores, basado en la idoneidad demostrada, asociada a un desempeño superior del trabajador y de la organización en correspondencia con las exigencias técnicas, productivas y de servicios. Es requerimiento esencial que estas competencias sean observables, medibles y que contribuyan al logro de los objetivos de la organización." (11)

Del análisis de estas definiciones puede concluirse que las competencias:

- Son características permanentes de la persona.
- Se ponen de manifiesto cuando se ejecuta una tarea o se realiza un trabajo.
- Están relacionadas con la ejecución exitosa en una actividad, sea laboral o de otra índole.
- Tienen una relación causal con el rendimiento laboral, es decir, no están solamente asociadas con el éxito, sino que se asume que realmente lo causan.
- Pueden ser generalizables a más de una actividad.

Luego de ser analizados los disímiles conceptos y criterios sobre competencia, se llegó a la conclusión que para el desarrollo de la presente investigación y en relación con el objetivo

planteado, se tomará como definición competencia la dada por la Norma Cubana en el año 2006.

1.1.1 Tipos de competencias

En la actualidad son múltiples las clasificaciones establecidas en torno al término competencia, atendiendo a variedad de criterios. En esta investigación se tomó como referencia a la clasificación señalada en el Proyecto Tuning[1], el cual incluye tres tipos de competencias en el ámbito educacional: las genéricas o transversales, las específicas y las básicas.

Clasificación (12)

1- Competencias genéricas o Transversales

Son las competencias claves de la Unión Europea. Son aquellas competencias que deben formarse en la educación básica, como instrumento de acceso general a la cultura. Las competencias que la integran son: lingüística, matemática, conocimiento e interacción con el mundo físico y natural, tratamiento de la información y competencia digital, social y ciudadana, cultural y artística, aprender a aprender, iniciativa y autonomía personal, y se expresan a través de las denominadas:

- Competencias instrumentales, de orden metodológico o de procedimiento, tales como la capacidad de análisis y síntesis, de organización y planificación, y de gestión de información.
- Competencias personales, tales como la capacidad para el trabajo en equipo, la habilidad para el manejo de las relaciones interpersonales, el compromiso ético.
- Competencias sistémicas, que se manifiestan en el aprendizaje autónomo, la adaptación a nuevas situaciones, la creatividad y el liderazgo, entre otras.

2- Competencias específicas

Son aquellas que se aplican a una situación o a una familia de situaciones en un contexto particular. Estas competencias se refieren al saber hacer en una situación y contexto

[1] Proyecto Tuning América Latina (2004-2007) busca "afinar" las estructuras educativas de América Latina iniciando un debate cuya meta es identificar e intercambiar información y mejorar la colaboración entre las instituciones de educación superior. Se definieron las competencias genéricas, las específicas y básicas en diferentes áreas.

concretos, siendo una serie de actos observables, esto es, de comportamientos específicos. Aplicadas a la escuela, son las relacionadas con cada área temática.

(…) Las competencias específicas están más centradas en el «saber profesional», el «saber hacer» y el «saber guiar» el hacer de otras personas; mientras que las competencias genéricas se sitúan en el «saber estar» y el «saber ser». Son transferibles en el sentido de que sirven en diferentes ámbitos profesionales. (13)

3- Competencias básicas

Son aquellas que aluden a la adquisición de habilidades fundamentales para el desarrollo del individuo en un contexto educativo. Estas se caracterizan por cumplir acciones básicas del proceso de un determinado nivel educativo.

1.2 Definición de evaluación por competencias

Múltiples han sido los autores que han definido el concepto de evaluación por competencia entre ellos los más relevantes son: Magalys Ruiz, Tobón, Pacienzia entre otros. A continuación, se muestra una selección de muchos de los conceptos enunciados por estos autores.

Proceso a través del que se verifica y valora la capacidad de una persona con relación al desempeño establecido, puede estar traducido en una norma. Se puede llevar a cabo mediante pruebas, exámenes prácticos, observación o examen de evidencias sobre el desempeño. Su propósito es formar un concepto sobre su competencia, a partir de un patrón (normas o criterios, indicadores y evidencias), e identificar aquellas áreas de desempeño que requieren ser fortalecidas mediante capacitación, para alcanzar la competencia. (14)

Se orienta a evaluar las competencias en los estudiantes teniendo como referencia el desempeño de estos ante las actividades y problemas del contexto profesional, social, disciplinar e investigativo. (15)

Es el proceso mediante el cual se recopilan evidencias y se realiza un juicio o dictamen de esas evidencias teniendo en cuenta criterios preestablecidos para retroalimentar en aras de mejorar la idoneidad. (16)

Existen grandes similitudes entre los conceptos anteriormente mencionados porque efectivamente la evaluación por competencia nos permite verificar, valorar, recopilar evidencia, tener ideas, etc. sobre el desempeño de cada estudiante y que han aprendido

mediante las actividades señalando donde están las dificultades que al analizarlas se buscan soluciones que interactúen sobre las mismas y hagan que se alcancen buenos resultados.

1.2.1 Características de la evaluación por competencias

La evaluación basada en competencia tiene varias características, a continuación, se referencian algunos de los autores que tratan este tema.

La evaluación por competencias, como señalan Prégent, Bernard y Kozanitis (2009):

- Propone a los estudiantes situaciones reales que se presentan en los contextos profesionales. Es una evaluación, en este sentido, contextualizada.
- Promueve el juicio crítico y la creación. No se trata de poner al estudiante en situación de «adivinar» las preguntas a las que debe responder, ni tampoco de reproducir los contenidos de una materia.
- Demanda a los estudiantes la creación de una producción personal (o grupal) propia, en la que puedan evaluarse de qué modo se han apropiado de los contenidos teóricos, y pueden aplicar los saberes teóricos y procedimentales movilizados para el desarrollo de una competencia en torno a una problemática o situación profesional dada.
- Integra la evaluación dentro de la propia enseñanza a través de feed-backs continuos del profesor que informa sobre los errores y avances de los estudiantes.
- Se organiza sobre actividades evaluativas más pertinentes y coherentes con un modelo de enseñanza basado en competencias: proyectos, realización de actividades en las que tengan que poner en práctica determinados gestos y habilidades profesionales, simulaciones profesionales (micro-enseñanza, entrevistas profesionales), portafolios profesionales, etc.

Según la Dra. Magalys Ruiz Iglesias[2] (2008):

- Es un proceso dinámico y multidimensional que realizan los diferentes agentes

[2] Dr. Magalys Ruiz Iglesias, directora del Centro de Internacionalización de competencias en Monterrey.

educativos implicados (docentes, estudiantes, institución y la propia sociedad).

- Tiene en cuenta tanto el proceso como los resultados de aprendizaje.
- Ofrece resultados de retroalimentación tanto cuantitativa como cualitativa.
- Tiene como horizonte servir al proyecto ético de vida (necesidades y fines, etc.) de los estudiantes.
- Reconoce las potencialidades, las inteligencias múltiples y la zona de desarrollo próximo de cada estudiante.
- Se basa en criterios objetivos y evidencias consensuadas socialmente, reconociendo además la dimensión subjetiva que siempre hay en todo proceso de evaluación.
- Se vincula con la mejora de la calidad de la educación ya que se trata de un instrumento que retroalimenta sobre el nivel de adquisición y dominio de las competencias y además informa sobre las acciones necesarias para superar las deficiencias en las mismas.

Para la propuesta de solución de la presente investigación nos basaremos en el concepto dado por la Dra. Magalys Ruiz Iglesias ya que ella tiene en cuenta la relación que existe entre todos los factores que intervienen en el proceso enseñanza-aprendizaje posibilitando los resultados del mismo. Además, analiza las particularidades de cada estudiante como son sus potencialidades, inteligencia y la zona de desarrollo próximo para realizar evaluaciones y según sus resultados plantearse nuevas tareas encaminadas a mejorar la calidad de la educación y elevar el nivel de aprendizaje de los estudiantes.

1.3 Características generales de la Plataforma Educativa Xauce ZERA

En la UCI se desarrolló un LMS denominado: plataforma educativa ZERA, en la cual se ha logrado fomentar un entorno que sirve en gran medida de apoyo en el proceso de enseñanza-aprendizaje. Es una plataforma educativa innovadora, flexible, fácil de usar, interactiva, adaptable y que reúne características de Sistemas Gestión de Contenidos y de aprendizaje, enriqueciendo y guiando el proceso de enseñanza-aprendizaje en empresas, organizaciones e instituciones de cualquier nivel de enseñanza a partir de un conjunto de herramientas centradas en los aprendices y ambientes de aprendizaje colaborativo, que le dan poder, tanto a la enseñanza como al aprendizaje. (17)

Entre sus principales características se pueden destacar:

- Basada en hiper-entornos de aprendizajes.
- Creación de cursos con una estructura capitular donde el contenido se muestra como un libro: capítulo, temas y subtemas.
- Creación de 5 tipos de recursos educativos.
- Creación de 8 tipologías de ejercicios.
- Gestión de grupos académicos.
- Soporte para la especificación IMS-QTI.
- Evaluación por rúbricas, escala de evaluación, co-evaluación manual y automática.
- Ideal para el trabajo semi-presencial b-learning y a distancia e-learning.
- Diseño responsive.
- Plantillas personalizadas para diversos tipos de contenidos.
- Fórum, sistema con varios roles, notificaciones etc.

Este conjunto de características convierte a la Plataforma Educativa XAUCE Zera en una plataforma capaz de competir con sus similares tanto a nivel nacional como internacional. (17)

1.4 Análisis a otras plataformas educativas con características similares

En la actualidad, existe una amplia gama de plataformas destinadas a la enseñanza virtual. Estas plataformas incluyen herramientas adaptadas a las necesidades de la institución para la que se desarrollan o adaptan. Algunas de ellas están estandarizadas (aunque permiten la adaptación a situaciones concretas), mientras que otras son completamente personalizadas. La presente investigación centra el siguiente análisis en algunas plataformas que por sus características pueden arrojar una visión más acertada en el desarrollo de competencias:

ATutor

ATutor es una plataforma que se destaca por el cumplimiento conforme a los estándares internacionales de accesibilidad, a través de los cuales permite el ingreso a estudiantes, profesores y administradores, incluyendo a usuarios con capacidades diferentes, quienes

cuentan con tecnologías especiales de apoyo para su acceso a la web. Con respecto a los usuarios involucrados, los educadores pueden rápidamente ensamblar, empaquetar y redistribuir contenido educativo, y llevar a cabo sus clases en línea; y los estudiantes pueden aprender en un entorno de aprendizaje adaptativo, dinámico y visualmente atractivo. Entre sus desventajas resalta que no cuenta con la posibilidad de crear itinerarios de aprendizaje, elemento que dificulta el seguimiento diferenciado a los estudiantes. (18)

Chamilo

Chamilo permite a los profesores construir cursos en línea como soporte a la modalidad presencial o netamente virtuales. Organiza los diferentes procesos de enseñanza-aprendizaje mediante diseño instruccional y colaborativo y está implementado de tal forma que permite al profesor escoger entre una serie de metodologías pedagógicas, siendo una de ellas el constructivismo social. Chamilo ayuda a mejorar las destrezas comunicativas a nivel individual y grupal, permitiendo al estudiante trabajar a su propio ritmo.

Claroline

Claroline brinda la posibilidad a los formadores de construir cursos en línea y gestionar las actividades de aprendizaje y colaboración en la web. Agrupa los contenidos en temas o módulos. Propone en uno de sus acápites la creación de “camino de aprendizaje”. Este concepto es similar al de “recorridos dirigidos” que se propone en la presente investigación, la diferencia radica en que en Claroline un “camino de aprendizaje”, es un resumen de un tema ya creado anteriormente y solo es editable durante la creación de los contenidos. (19)

Dokeos

Dokeos es un entorno de aprendizaje electrónico, una aplicación de administración de contenidos de cursos y también una herramienta de colaboración. Facilita la creación y organización de contenidos interactivos y ejercicios. Entre sus desventajas resalta que se deben mejorarse las herramientas de creación de contenidos. (20)

LRN

LRN cuenta con un sofisticado sistema de portales que permite administrar cursos, contenidos y herramientas de colaboración. La esencia de LRN está en la colaboración, todas las aplicaciones proveen formas intuitivas, dirigidas o espontáneas para interactuar

entre los participantes del proceso educativo. Resalta como una desventaja para el presente estudio el hecho de que los cursos están organizados en portafolios. (21)

Moodle

Moodle, en su versión 2.2, incorpora el Sistema avanzado de calificaciones diseñado exclusivamente para el módulo «Tareas», que consiste en la implementación de un sistema de evaluación basado en rúbricas. Esta versión incluye una extensión que calcula la calificación según un sistema de rúbricas, que consiste en la definición de una serie de «elementos a evaluar» de forma diferenciada dentro de la tarea y de la asignación de un valor a cada uno de esos aspectos. La calificación final se obtiene mediante el cálculo previo del porcentaje de los puntos obtenidos en todas las rúbricas sobre los puntos posibles a obtener, y este porcentaje se trasladará a la calificación final según el baremo establecido. Además, este sistema permite una retroalimentación no solo global de las competencias sino para cada una de las rúbricas establecidas, de manera que el estudiante podrá conocer en todo momento cómo ha sido evaluado en cada uno de los indicadores que conforman la calificación final de la competencia. Para adaptar el modelo a las necesidades de nuestro Plan de Evaluación de Competencias (en lo adelante PEC). El diseño en el aula virtual de se estructura del siguiente modo: (22)

- Fichas de evaluación por competencias. Se ofrecen al alumnado las fichas de cada una de las competencias que serán objeto de evaluación en la asignatura.
- Plan de actividades de evaluación. Este documento contiene información sobre las actividades que han de realizar los estudiantes. Una tabla resumen global que contiene los siguientes datos: denominación de la actividad, competencia/s implicada/s, subcompetencia/s implicada/s, indicador/es implicado/s y temporalización. Y para cada una de las actividades: indicador/es, tareas, recursos y criterios de evaluación.
- Elaboración de escala personalizada. Las rúbricas que se utilizarán para cada uno de los indicadores de la evaluación por competencias exigen, en nuestro caso, la creación de una escala numérica específica (0; 0,25; 0,5; 0,75 y 1) para cada uno de los siguientes valores: muy deficiente, insuficiente, bueno, muy bueno y excelente respectivamente.
- Diseño del sistema de calificación. En primer lugar, se definen las «categorías» que se corresponden con cada una de las competencias a evaluar. Posteriormente, los «ítems de calificación», que son cada uno de los indicadores, agrupados en su

competencia correspondiente. Para cada categoría (competencia) se selecciona la opción «Media ponderada de calificaciones», se determina cuáles serán las calificaciones máxima y mínima (sobre 10) y se establece el peso (porcentaje) con relación a la calificación final. Para cada uno de los ítems de calificación (indicadores) también se marca su peso y su calificación máxima y mínima (sobre 5), así como la escala personalizada, previamente definida.

- Actividades de evaluación. Se incorporan al aula virtual tantas «Tareas» como actividades de evaluación, y se utilizan diferentes modalidades: «Actividad no en línea», «Subir un solo archivo» y «Subida avanzada de archivos». Ninguna de estas tareas incorpora «calificación», puesto que la evaluación no se realiza directamente sobre la actividad sino sobre los indicadores asociados pertenecientes a una determinada competencia. También se incorporan «Foros», evaluados en la modalidad de «debate simple», que sí son calificados directamente por estar asociados con un indicador específico.
- Rúbricas para cada uno de los indicadores de todas las competencias a evaluar. Se incorpora una «Actividad no en línea» para cada indicador, en la que se selecciona la escala de evaluación personalizada, la «rúbrica» como método de calificación y la categoría de calificación, es decir, la competencia a la que pertenece el indicador.

El uso de un sistema de gestión de aprendizaje para el diseño y gestión de la evaluación por competencias facilita la incorporación del PEC, puesto que simplifica el diseño de la evaluación por parte del profesorado y ofrece una herramienta accesible para estudiantes y docentes. Añade al PEC dos elementos muy importantes: la recogida centralizada de información sobre evaluación por competencias, y la comunicación directa e inmediata de los resultados de aprendizaje a los estudiantes a través del calificador individualizado que incluye retroalimentación. Como pudimos ver, el uso de las rúbricas favorece el proceso evaluativo en entornos virtuales, es por ello que se decidió hacer uso de este instrumento de evaluación para realizar la evaluación por competencia en la nueva versión de la Plataforma Educativa ZERA; y dado que los objetivos se estructuran de manera similar a las competencias, se considera factible utilizar la rúbrica para evaluar, los objetivos en la Plataforma.

De manera general las plataformas estudiadas sirvieron de apoyo para la presente investigación ya que fueron identificados los tipos de evaluación, los instrumentos de evaluación, así como las estrategias de aprendizaje que emplean. En el caso específico de

la Plataforma Moodle, se pudo evidenciar la forma en que se realiza el proceso de evaluación del aprendizaje por competencias, haciendo uso de la rúbrica como instrumento de evaluación que favorece el proceso evaluativo.

1.5 Metodologías, tecnologías y herramientas empleadas

1.5.1 Metodologías de desarrollo

Actualmente las metodologías de desarrollo de software pueden considerarse como una base necesaria para la ejecución de cualquier proyecto de desarrollo de software. Las mismas necesitan sustentarse en algo más que la experiencia de las capacidades de sus programadores y equipos de desarrollo. Estas metodologías son necesarias para poder realizar un proyecto profesional, así como poder desarrollar efectiva y eficientemente el software. En la actualidad existen una gran cantidad de metodologías para el desarrollo de software, separadas en dos grandes grupos; las metodologías tradicionales o pesadas y las metodologías agiles. (23)

Metodologías robustas o tradicionales

Las metodologías tradicionales se basan en las buenas prácticas dentro de la ingeniería del software, siguiendo un marco de disciplina estricto y un riguroso proceso de aplicación. Está basada en normas provenientes de estándares seguidos por el entorno de desarrollo, presenta grupos de desarrollo bastante grandes y posiblemente distribuidos, posee más artefactos y más roles que la metodología ágil. El cliente interactúa con el equipo de desarrollo mediante reuniones. (23)

Dentro de esta clasificación encontramos a RUP (Proceso Unificado de Desarrollo) y MSF, a continuación, se presentan algunas de las principales características de estos dos tipos de metodologías robustas.

RUP es uno de los procesos de desarrollo de software más generales de los existentes en la actualidad, garantiza la elaboración de todas las fases en el desarrollo de un producto orientado a objetos y pensado para adaptarse a las características de cualquier proyecto. (24) Entre las características principales de RUP podemos encontrar:

Iterativo e incremental: Es la secuencia de iteraciones que pueden realizarse en forma de cascada. Cada iteración aborda una parte de la funcionalidad, pasando por todos los flujos de trabajo relevantes y refinando la arquitectura.

Dirigido por casos de uso: No son sólo una herramienta para especificar los requisitos del sistema. También guían su diseño, implementación y prueba. Los mismos constituyen

un elemento integrador y una guía del trabajo que comprende todos los flujos de trabajo: requerimientos, análisis, diseño, implementación y prueba.

Centrado en la arquitectura: La arquitectura se representa mediante varias vistas que se centran determinados aspectos del sistema. Está formado por las vistas lógicas, de implementación, de proceso y de despliegue, más la de Casos de Uso que es la que da cohesión a todas las vistas y juntas forman el llamado modelo 4+1 de la arquitectura.

En RUP se han agrupado las actividades en grupos lógicos definiéndose nueve flujos de trabajo principales. Los primeros seis son conocidos como flujos de ingeniería que abarcan: los flujos de trabajo Modelo de Negocio, Requerimientos, Análisis, Diseño, Implementación y Prueba; los últimos tres son de soporte: Administración del Proyecto, Administración de Configuración y Cambios y Ambiente. El desarrollo de un software realizado con la metodología RUP se divide en 4 fases: (24)
Inicio: Se determina la visión del proyecto y se describe el negocio del mismo.

Elaboración: Determina la óptima arquitectura del sistema y se obtiene una aplicación ejecutable, la cual responde a los requerimientos funcionales.

Construcción: Obtenemos el producto listo para su utilización, con al menos una versión del software pasado por pruebas. Además de estar documentado y tiene manual de usuario.

Transición: La versión del software ya está listo para su instalación. Esto puede implicar reparación de errores.

MSF: Es un compendio de las mejores prácticas en cuanto a administración de proyectos se refiere. Más que una metodología rígida de administración de proyectos, MSF es una serie de modelos que puede adaptarse a cualquier proyecto de tecnología de información.

MSF está separado en cinco fases: la fase de Visión y Alcances trata uno de los requisitos más fundamentales para el éxito del proyecto, la unificación del equipo detrás de una visión común; se definen los líderes y responsables del proyecto, adicionalmente se identifican las metas y objetivos a alcanzar. La segunda fase es la Planeación donde el equipo prepara las especificaciones funcionales, realiza el proceso de diseño de la solución, y prepara los planes de trabajo, estimaciones de costos y cronogramas de los diferentes entregables del proyecto. La tercera fase es: Desarrollo, durante esta fase el equipo realice la mayor parte de la construcción de los componentes. Estabilización es el nombre de la cuarta fase en la que se conducen pruebas sobre la solución. La quinta y última fase es la de Implantación

donde el equipo implanta la tecnología base y los componentes relacionados, estabiliza la instalación y obtiene la aprobación final del cliente. (25)

Metodología ágil

Las metodologías ágiles representan una solución a los problemas que requieren una respuesta rápida en un ambiente flexible y con cambios constantes, haciendo caso omiso a la documentación rigurosa y los métodos formales. Está basada en heurísticas provenientes de prácticas de producción de código, presenta grupos de desarrollo bastante pequeños, poseen pocos artefactos y roles, además permite que el cliente sea parte del equipo de desarrollo. (23)

A continuación, se presentan dos ejemplos de metodología ágil:

La programación extrema, o XP, es una metodología ágil concebida y desarrollada para dirigir las necesidades específicas del desarrollo de software conducido por equipos pequeños. Esta les da poder a los desarrolladores para responder con confianza a los requerimientos cambiantes del consumidor final. Se caracteriza además por fomentar la comunicación desarrollador-cliente desde el primer día. Es considerada ligera, flexible, predecible, de bajo riesgo, y no por ello menos científica. Entre otras ventajas pueden mencionarse los pocos requerimientos de documentación y planificación, así como la exigencia de tener siempre el cliente disponible para el desarrollo, implicando una mejor correspondencia entre el producto y la necesidad del negocio. (26)

Por otra parte, Scrum es otra metodología ágil que comparte muchas de las características de XP. No obstante, son notables algunas diferencias, entre las que están el hecho de que hace mayor énfasis en los flujos de gestión de proyectos y no define una práctica detallada de la ingeniería de software. Denomina las iteraciones como “Sprint” y requiere la existencia de un equipo de desarrollo auto-organizado que no sea afectado por nuevos cambios en los requerimientos una vez iniciado el sprint, brindando de esta forma poca flexibilidad ante cambios frecuentes. (27)

1.5.2 Metodología de desarrollo AUP-UCI

Enmarcándose en las necesidades de la UCI de converger hacia un único marco de desarrollo de software, se decide escoger como metodología a la variante AUP-UCI. AUP-UCI se caracteriza por hacer un enfoque hacia los requisitos evitando la excesiva

documentación que genera RUP, define un solo marco de trabajo para el desarrollo de software en la universidad y permite modelar una arquitectura que logre soportar y refinar los requisitos funcionales y no funcionales. (28)

En la presente investigación se estará utilizando esta metodología en el escenario 4 él cual posee como característica el modelado de historia de usuario. Esta selección se realiza, teniendo en cuenta, que el proyecto está bien definido, el cliente en todo momento estará junto al equipo de desarrollo para convenir los detalles de los requisitos. Además, a pesar de la complejidad mostrada por el proyecto, este no es extenso, posibilitando el uso de las Historias de Usuarios. (28)

1.5.3 Marco de trabajo Xalix

Como parte de la arquitectura de referencia para el desarrollo de aplicaciones en el Centro FORTES y con el propósito de disminuir la diversidad tecnológica de soluciones, se desarrolló un marco de trabajo en las Líneas de Producción de *Software* (LPS) llamado Xalix. Actualmente representa el marco de trabajo sobre el que se está desarrollando la Plataforma Educativa Xauce ZERA v2.1.

Las tecnologías utilizadas por este marco de trabajo son:

- Gestor de bases de datos: PostgreSQL.
- Framework de desarrollo: Symfony 2.7.16.
- Lenguaje de programación para el servidor: php 7.0.
- Lenguaje de programación para el cliente: HTML 5 (Hypertext Markup Language) y JavaScript.
- Librería CSS (Cascading Style Sheets): Boostrap 3.0.0 (compilado para el marco de referencia).
- Componentes nativos de Xalix (en desarrollo): Gestión de autenticación, Gestión de servicios web, Temas y Panel de administración.

1.5.4 Lenguaje de modelado: UML 2.0

Lenguaje Unificado de Modelado (UML) es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software. UML es utilizado para el modelado completo de sistemas, tanto en el diseño de los sistemas software como para la arquitectura hardware donde se ejecuten.

El lenguaje UML tiene una notación gráfica muy expresiva que permite representar en mayor o menor medida todas las fases de un proyecto informático: desde el análisis con los casos de uso, el diseño con los diagramas de clases, objetos y otros, hasta la implementación y configuración con los diagramas de despliegue. Este modelado visual está diseñado para que sea independiente del lenguaje de implementación, de tal forma que los diseños realizados usando UML se puedan implementar en cualquier lenguaje que soporte las posibilidades de UML (principalmente lenguajes orientados a objetos). (29)

Los objetivos de UML son muchos, pero se pueden sintetizar sus funciones:

- Visualizar: UML permite expresar de una forma gráfica un sistema de forma que otro lo puede entender.
- Especificar: UML posibilita especificar cuáles son las características de un sistema antes de su construcción.
- Construir: a partir de los modelos especificados se pueden construir los sistemas diseñados.
- Documentar: los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión.

1.5.5 Lenguajes de desarrollo

XHTML (Lenguaje Extensible de Marcado de Hipertexto)

Es una versión más estricta y limpia de HTML, que nace precisamente con el objetivo de remplazar a HTML ante su limitación de uso. XHTML extiende HTML 4.0 combinando la sintaxis de HTML, diseñado para mostrar datos, con la de XML, diseñado para describir los datos. Este surge como el lenguaje cuyo etiquetado, más estricto que HTML, va a permitir una correcta interpretación de la información independientemente del dispositivo desde el que se accede a ella, puede incluir otros lenguajes como MathML, SMIL o SVG, al contrario que HTML. (30)

Exige una serie de requisitos básicos a cumplir en lo que a código se refiere. Entre estos requisitos básicos se puede mencionar una estructuración coherente dentro del documento donde se incluirían elementos correctamente anidados, etiquetas en minúsculas, elementos cerrados correctamente, atributos de valores entrecomillados. (30)

CSS (Hojas de Estilo en Cascada) 3.0

Hojas de Estilo en Cascada (Cascading Style Sheets), es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo va a ser pronunciada la información presente en ese documento a través de un dispositivo de lectura. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos. (31)

Se utiliza para dar estilo a documentos HTML y XML, separando el contenido de la presentación. Los Estilos definen la forma de mostrar los elementos HTML y XML. Permite a los desarrolladores Web controlar el estilo y el formato de múltiples páginas Web al mismo tiempo. Cualquier cambio en el estilo marcado para un elemento en la CSS afectará a todas las páginas vinculadas a esa CSS en las que aparezca ese elemento. (31)

Funciona a base de reglas, es decir, declaraciones sobre el estilo de uno o más elementos. Las hojas de estilo están compuestas por una o más de esas reglas aplicadas a un documento HTML o XML. La regla tiene dos partes: un selector y la declaración. A su vez la declaración está compuesta por una propiedad y el valor que se le asigne. (31)

Con este lenguaje se permite darles estilo a los documentos se garantiza el acabado de cada página posibilitando un formato adecuado.

JavaScript

Es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. (32)

Con la utilización de este lenguaje de programación se pueden crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. (32)

Ajax

AJAX es un acrónimo de Asynchronous JavaScript + XML, que se puede traducir como "JavaScript asíncrono + XML". “Ajax no es una tecnología en sí mismo. En realidad, se trata de varias tecnologías independientes que se unen de formas nuevas y sorprendentes.”

Las tecnologías que forman AJAX son: (33)

- XHTML y CSS, para crear una presentación basada en estándares.
- DOM, para la interacción y manipulación dinámica de la presentación.
- XMLHttpRequest, para el intercambio asíncrono de información.
- JavaScript, para unir todas las demás tecnologías.

3.Lenguajes del lado del servidor

PHP

PHP 7.0: es un lenguaje de *script* interpretado, utilizado para la generación de páginas web dinámicas. PHP es un acrónimo recursivo que significa "*Hypertext Pre-processor*". La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características específicas de sí mismo. El objetivo que persigue este lenguaje es permitir rápidamente a los desarrolladores la generación dinámica de páginas web. No es un lenguaje de marcas como podría ser HTML o XML. Al ser un lenguaje libre dispone de una gran cantidad de características que lo convierten en la herramienta ideal para la creación de páginas web dinámicas. (34)

Entre sus ventajas principales se pueden encontrar:

- El código se pone al día continuamente con mejoras y extensiones de lenguaje para ampliar las capacidades de PHP.
- Soporte para una gran cantidad de bases de datos: MySQL, PostgreSQL, Oracle, MS SQL *Server*, entre otras.
- Integración con varias bibliotecas externas, permite generar documentos en PDF (documentos de Acrobat Reader) hasta analizar código XML.
- Perceptiblemente más fácil de mantener y poner al día que el código desarrollado en otros lenguajes.

XML (Lenguaje de Etiquetado Extensible)

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. Las tecnologías XML son un conjunto de módulos que ofrecen servicios útiles

a las demandas más frecuentes por parte de los usuarios. XML sirve para estructurar, almacenar e intercambiar información. (35)

1.5.6 Framework de desarrollo

Los frameworks son un conjunto de librerías o funciones que se pueden emplear en el desarrollo de las aplicaciones. Se puede decir que son de fácil uso puesto que nos abstrae de complejas y básicas implementaciones para emplearlas de una forma más sencillas y organizadas.

Capa de presentación: Es la capa que ve el usuario, presenta el sistema, le comunica la información y captura la información del usuario en un mínimo proceso. Esta capa se comunica únicamente con la capa de negocio. En esta capa se emplean el framework Jquery 1.5.2 y Bootstrap 2.2.

jQuery es una biblioteca de JavaScript rápida y concisa que simplifica el recorrido de un documento HTML, de manejo de eventos, animaciones e interacciones Ajax para el desarrollo web rápido. La versión en cuestión se le realizaron mejoras en cuanto a compatibilidad con los navegadores y errores de código que se encontraban en versiones anteriores.

Bootstrap en su versión 2.2 es otro de los framework utilizados en la capa de presentación. Posee varias características tales como su diseño adaptable sus componentes se adaptan y escalan según las resoluciones de pantalla y dispositivos para proporcionar una experiencia consistente, creado para soportar elementos nuevos HTML5 y sintaxis, permite personalizar plugins, es uno de los conjuntos de herramientas más completos con docenas de componentes funcionales completos listo para poner en uso, componentes mejorados progresivamente para un estilo máximo de CCS 3.0.

Con la utilización de estos framework implica que nuestra aplicación se verá de igual forma en todos los navegadores. (36)

Capa lógica de negocio: Esta es la capa que recibe todas las peticiones del usuario y se envían las respuestas tras el proceso, se establecen todas las reglas que deben cumplirse, se comunica con la capa de presentación para recibir las solicitudes y presentar los resultados y con la de acceso a datos para solicitar al gestor de base de datos para almacenar o recuperar datos de él. En esta capa se utiliza Symfony en la versión 2.1.

Symfony es un completo framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web. Symfony está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows. (37)

Capa de acceso a dato: Es donde residen los datos y es la encargada de acceder a los mismos, está formada por uno o más gestores de base de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio. En esta capa se utiliza el ORM (Object Relation Mapper) Doctrine 1.2

Un mapeador de Objeto Relacional o ORM es una tecnología de programación que permite convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional, es decir, las tablas de la base de datos pasan a ser clases y registros, objetos que se manejan con facilidad. (38)

Doctrine 1.2 presenta las siguientes características: (38)

- Soporte para datos jerárquicos;
- Soporte para hooks (métodos que pueden validar o modificar las escrituras y lecturas de la base de datos) y eventos para manejar la lógica de negocio relacionada.
- Herencia.
- Bajo nivel de configuración que necesita para empezar un proyecto.
- Genera clases a partir de una base de datos existente y después el programador puede especificar relaciones y añadir funcionalidad extra a las clases autogeneradas.

La implementación con Doctrine es fácil, legible y organizada además agiliza el trabajo permitiendo centrar la atención en procesos más complejos del desarrollo.

1.5.7 Herramienta de modelado UML: Visual Paradigm 8.0

Es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Soporta todos los diagramas UML y el diagrama Entidad-Relación, ayudando a una rápida construcción de aplicaciones con calidad y a un menor coste. Posee una interfaz amigable y se puede modelar en varios idiomas. Es una herramienta colaborativa, pues soporta múltiples usuarios trabajando sobre el mismo proyecto. La documentación del proyecto puede ser generada automáticamente en varios formatos (web o .pdf), y permite control de versiones. Soporta la notación UML 2.1, ingeniería inversa, generador de informes, editor de figuras, integración con MS Visio, *plugins*, integración IDE con Visual Studio, IntelliJ IDEA, Eclipse, NetBeans y otros. (39)

A continuación, se presentan 5 de las principales ventajas de esta herramienta:

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

Esta herramienta se utilizará en la investigación para general los diferentes diagramas que se utilizarán para mostrar un mejor entendimiento de la propuesta de solución.

1.5.8 Sistema de gestor de Base de Datos: PostgresSQL 9.1

Un Sistema de gestor de Base de Datos es un (…) conjunto de programas, procedimientos y lenguajes que nos proporcionan las herramientas necesarias para trabajar con una base de datos. Incorpora una serie de funciones que nos permita definir los registros, sus

campos, sus relaciones, insertar, suprimir, modificar y consultar los datos. Dentro de los gestores de base de datos se encuentra PostgresSQL el mismo es utilizado para establecer la conexión con la base de datos. (40)

Es un potente motor de bases de datos, que tiene prestaciones y funcionalidades equivalentes a muchos gestores de bases de datos comerciales.

Entre la características del PostgresSQL, a continuación, se muestran 10 de la más importantes: (41)

- PostgresSQL implementa un subconjunto extendido de los estándares SQL92 y SQL99.
- Permiten el paso entre dos estados manteniendo la integridad de los datos.
- PostgresSQL soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos.
- Postgress ofrece varios modos de bloqueo para controlar el acceso concurrente a los datos en tablas.
- Tienen la función de mantener la integridad y consistencia en la Base Datos.
- Como todos los manejadores de bases de datos, PostgreSQL implementa los tipos de datos definidos para el estándar SQL3 y aumenta algunos otros.
- PostgreSQL soporta operadores, funciones métodos de acceso y tipos de datos definidos por el usuario.
- Incorpora una estructura de datos Array.
- Conectividad TCP/IP, JDBC y ODBC
- Interfaz con diversos lenguajes C, C++, Java, Delphi, Python, Perl, PHP, Bash.

1.5.9 Servidor Web

Nginx es un servidor web que está listo para ser utilizado como un proxy inverso. En este modo, se utiliza para equilibrar la carga entre los servidores back-end, o para proporcionar almacenamiento en caché para un servidor back-end lento. Es un software asíncrono, a diferencia de Apache que está basado en procesos. La ventaja principal de ser asíncrono, es su escalabilidad. En un sistema basado en procesos, cada conexión simultánea requiere

de un hilo, lo que puede llevar a sobrecargar el servidor, mientras que en un servidor asíncrono se gestionan las peticiones en muy pocos hilos, reduciendo las posibilidades de sobrecarga en el servidor. Nginx puede almacenar contenido estático como imágenes con lo que se quita carga a los servidores web, mejorando la velocidad de carga de las páginas. (42)

1.5.10 Entorno de desarrollo Integrado: NetBeans 8.0

NetBeans: es un programa compuesto por un conjunto de herramientas de programación que permite elaborar aplicaciones de escritorio, para la web y para dispositivos portátiles sin que cambie la forma de programar. La programación mediante este se realiza a través de componentes de software modulares, también llamados módulos, que le aportan gran funcionalidad y versatilidad posibilitando la integración con PHP y la librería jQuery de JavaScript. Ofrece todas las funciones de los entornos de desarrollo integrado avanzados como diseño de interfaces, asistentes para la conexión con bases de datos, creación automática de propiedades y clases. La versión a utilizar es la 8.0. (43)

1.6 Sistema de control de versiones

El control de versiones (VCS) es un sistema que registra los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo, de modo que puedas recuperar versiones específicas más adelante.

El mismo permite revertir archivos a un estado anterior, revertir el proyecto entero a un estado anterior, comparar cambios a lo largo del tiempo, ver quién modificó por última vez algo que puede estar causando un problema, quién introdujo un error y cuándo, y mucho más. Usar un VCS también significa que, si se dañan o pierden archivos, se pueden recuperar fácilmente. *(44)*

Git: es un sistema de control de versiones distribuido diseñado por Linus Torvalds en el año 2005. Surge como alternativa a BitKeeper[3], un control de versiones privativo que usaba en ese entonces para el kernel.

Es liberado bajo una licencia GNU GPLv2[4] y su última versión estable fue publicada a inicios

[3] Es un sistema de control de versiones distribuido para el código fuente de los programas producidos a partir de Bit Mover Inc.

[4] Licencia de derecho de autor más ampliamente usada en el mundo del software libre y código abierto que garantiza a los usuarios finales la libertad de usar, estudiar, compartir (copiar) y modificar el software.

de abril de 2017. Por sus disímiles ventajas, se ha convertido en uno de los más usados alrededor del mundo, puesto que no depende de acceso a la red o un repositorio central; además, está enfocado en la velocidad, uso práctico y manejo de proyectos grande. *(44)*

Conclusiones de capítulo

Teniendo en cuenta el estudio realizado previamente sobre las principales tendencias en el campo de la informática, se toma como decisión desarrollar una aplicación *web* utilizando el lenguaje para el servidor PHP v7.0, para el cliente html5 y JavaScript, con la integración de los frameworks Symfony v2.7.16 y Boostrap v3.0.0, buscando ganar en velocidad de elaboración e incorporar buenas prácticas de desarrollo con el uso de patrones. Para la construcción del sistema se determina hacer uso de las herramientas NetBeans v8.0 y Visual Paradigm v8.0 debido a las facilidades que estas brindan. Todo este proceso será controlado y orientado por la metodología de desarrollo AUP en su versión UCI, en el escenario 4 dirigido por historias de usuario.

CAPÍTULO 2: DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

Introducción del capítulo

Al terminar el estado del arte se hace necesario tener una mayor claridad del sistema que se desarrollará y para esto se debe realizar el modelo de negocio o modelo de dominio. En base a ello se deben elaborar todos los requisitos funcionales y no funcionales que tendrá la solución propuesta, dejando en evidencia las capacidades y cualidades del producto. Estos luego darán paso al surgimiento de los casos de uso, los actores y diagramas de casos de uso.

2.1 Modelo de dominio

En la presente investigación se realiza un modelo de dominio ya que los procesos de negocio no estaban bien definidos, para ello se puso de manifiesto todos los conceptos y términos presentes en el ambiente donde se desarrollarán las funcionalidades, ya que puede utilizarse para capturar y expresar el entendimiento ganado en un área bajo análisis como paso previo al diseño de un sistema.

2.1.1 Conceptos del modelo de dominio

A continuación, se muestran los conceptos de las diferentes clases que intervienen en el modelo de dominio:

Competencia: Es el conjunto de los conocimientos, habilidades y actitudes que el estudiante va adquiriendo durante su período docente.

Escala de evaluación: es un conjunto de criterios y estándares, generalmente relacionados con objetivos de aprendizaje, que se utilizan para evaluar un nivel de desempeño o una tarea.

Administrador: Persona responsable de gestionar las competencias y asociarlas al índice de contenido.

Docente: Persona responsable de educar, guiar y supervisar a los estudiantes en el proceso de enseñanza - aprendizaje.

Estudiante: Persona que cursa estudios para obtener el grado de bachillerato.

Conocimientos específicos: Conjunto de habilidades que deberá tener el estudiante para vencer un contenido determinado.

Evaluación: Proceso consustancial al desarrollo del proceso docente educativo.

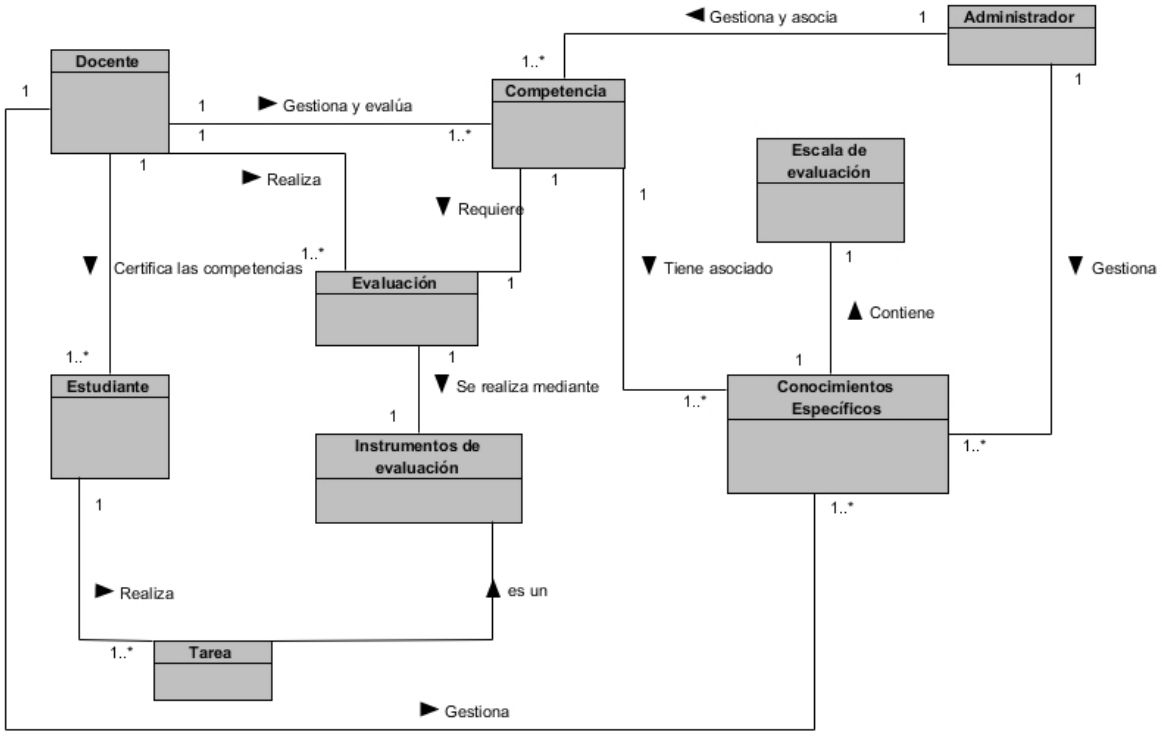
Instrumento de evaluación: Herramientas utilizadas para recolectar información nos ayudan a la medición de una actividad determinada.

Tarea: Es el trabajo que se asigna a los estudiantes por sus profesores, y que se indica que debe completarse, en su caso, fuera del aula y de la jornada escolar, en el entorno doméstico, con o sin ayuda de la familia.

2.1.2 Diagrama de modelo de dominio

El modelo de dominio que a continuación se presenta explica brevemente la relación de los conceptos anteriormente identificados en forma de clases.

Figura 1: Diagrama de dominio



2.2 Descripción del sistema propuesto

El sistema formará parte de la Plataforma Educativa ZERA, tiene como objetivo proporcionarle al docente una herramienta para la evaluación de las competencias adquiridas durante el transcurso del curso de sus estudiantes. El administrador del sistema es el encargado de gestionar las competencias y los conocimientos específicos que serán utilizados de manera general en la plataforma, en el caso de ser un docente este podrá crear competencias dentro del curso que le corresponde sin poder modificar o eliminar las competencias creadas por el administrador del sistema, además el docente podrá evaluar las competencias de un determinado grupo o estudiante, también podrá observar todas las competencias de los estudiantes y a su vez el estado de desarrollo en que se encuentran esas competencias. El estudiante podrá ver todas las competencias asociadas a dicha actividad, así como la nota correspondiente si ya ha sido resuelta por él. Además, brinda la posibilidad de generar una nota general de la competencia ya que esta puede estar asociada a varios de los instrumentos.

2.3 Requisitos

En este epígrafe se especifican todos los requisitos funcionales y no funcionalidades existentes, estos deben ser lo suficientemente entendibles como para que pueda llegarse a un acuerdo entre el cliente y los desarrolladores sobre qué debe y qué no debe hacer el sistema en cuestión.

2.3.1 Requisitos funcionales

Son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos, los requerimientos funcionales de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer. (45)

RF1: Gestionar competencia

RF 1.1: Incluir competencia.

RF 1.2: Modificar competencia.

RF 1.3: Ver competencia.

RF 1.4: Eliminar competencia.

RF 2: Gestionar conocimientos específicos dentro del curso

RF 1.1: Incluir conocimientos específicos dentro del curso.

RF 1.2: Modificar conocimientos específicos dentro del curso.

RF 1.3: Ver conocimientos específicos dentro del curso.

RF 1.4: Eliminar conocimientos específicos dentro del curso.

RF 3: Gestionar competencias dentro del curso

RF 3.1: Incluir competencias dentro del curso.

RF 3.2: Modificar competencias dentro del curso.

RF 3.3: Ver competencias dentro del curso.

RF 3.4: Eliminar competencias dentro del curso.

RF 4: Gestionar conocimientos específicos

RF 1.1: Incluir conocimientos específicos.

RF 1.2: Modificar conocimientos específicos.

RF 1.3: Ver conocimientos específicos.

RF 1.4: Eliminar conocimientos específicos.

RF 5: Evaluar competencia

RF 6: Mostrar competencia desarrolladas

RF 7: Mostrar estado de las competencias de los estudiantes

2.3.2 Descripción de Requisitos

Los requisitos funcionales identificados para dar solución a la problemática planteada se describen a continuación.

Tabla 1: Descripción de requisitos

No.	Requisito funcional	Descripción
RF1	Crear competencia	Permitir crear las competencias que se utilizan de forma general en la plataforma. Para ello el sistema muestra un formulario con los siguientes campos: • Nombre • Descripción • Tipo • Activa • Default • Escala • Conocimientos específicos
RF2	Modificar competencia	Permitir modificar todos los campos de las competencias que se utilizan de forma general en la plataforma.
RF3	Ver competencia	Permitir ver las competencias que se utilizan de forma general en la plataforma. Para ello el sistema muestra un formulario con los siguientes campos: • Nombre • Descripción • Tipo • Activa • Default • Escala • Conocimientos específicos Todos ellos con sus valores correspondientes.
RF4	Eliminar competencia	Permitir eliminar las competencias que se utilizan de forma general en la plataforma.
RF5	Crear conocimientos específicos	Permitir crear los conocimientos específicos que se utilizan de forma general en la plataforma. Para ello el sistema muestra un formulario con los siguientes campos: • Nombre • Descripción • Activa • Default • Escala • Conocimientos específicos • Tipo

RF6	Modificar conocimientos específicos	Permitir modificar todos los campos de los conocimientos específicos que se utilizan de forma general en la plataforma
RF7	Ver conocimientos específicos	Permitir ver los conocimientos específicos que se utilizan de forma general en la plataforma. Para ello el sistema muestra un formulario con los siguientes campos: • Nombre • Descripción • Tipo • Activa • Default • Escala • Conocimientos específicos Todos ellos con sus valores correspondientes.
RF8	Eliminar conocimientos específicos	Permitir eliminar los conocimientos específicos que se utilizan de forma general en la plataforma.
RF9	Crear competencia dentro del curso	Permitir a los profesores crear una competencia dentro de su curso. Para ello el sistema muestra un formulario con los siguientes campos: • Nombre • Descripción • Tipo • Activa • Default • Escala • Conocimientos específicos
RF10	Modificar competencia dentro del curso	Permitir a los profesores modificar todos los campos de una competencia dentro de su curso.
RF11	Ver competencia dentro del curso	Permitir a los profesores ver una competencia dentro de su curso. Para ello el sistema muestra un formulario con los siguientes campos: • Nombre • Descripción • Tipo • Activa • Default • Escala • Conocimientos específicos

		Todos ellos con sus valores correspondientes.
RF12	Eliminar competencia dentro del curso	Permitir a los profesores eliminar una competencia dentro de su curso.
RF13	Evaluar competencia	Permitir a los profesores evaluar las competencias
RF14	Mostrar competencias desarrolladas	Permitir a los profesores ver las competencias desarrolladas por los estudiantes
RF15	Mostrar estado de las competencias de los estudiantes	Permitir a los profesor ver el estado de las competencias de los estudiantes.
RF16	Crear conocimientos específicos dentro del curso	Permitir crear los conocimientos específicos dentro de los cursos Para ello el sistema muestra un formulario con los siguientes campos: • Nombre • Descripción • Tipo • Activa • Default • Escala • Conocimientos específicos
RF17	Editar conocimientos específicos dentro del curso	Permitir editar todos los campos de los conocimientos específicos dentro de los cursos
RF18	Ver conocimientos específicos dentro del curso	Permitir ver los conocimientos específicos dentro de los cursos Para ello el sistema muestra un formulario con los siguientes campos: • Nombre • Descripción • Tipo • Activa • Default • Escala • Conocimientos específicos Todos ellos con sus valores correspondientes.
RF19	Eliminar conocimientos específicos dentro del curso	Permitir eliminar los conocimientos específicos dentro de los cursos

2.3.3 Requisitos no funcionales

Son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los requerimientos no funcionales a menudo se aplican al sistema es su totalidad. Normalmente apenas se aplican a características o servicios individuales del sistema. (45)

De seguridad

- Negar el acceso de los usuarios a acciones no permitidas, haciendo uso de credenciales.

De soporte

- Compatibilidad con los navegadores: Explorer 7.0 o superior, Mozilla Firefox, Opera, Chrome.

2.4 Patrones de diseño

En esta sección se hace referencia a todos los patrones de diseño utilizados en el sistema.

- CRUD (Creating, Reading, Updating, Deleting): este patrón se basa en la fusión de CU simples para formar una unidad conceptual. Se puede apreciar el uso de este en el requisito funcional Gestionar Competencia.

- 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 requisito. Este rol es representado por otro actor, heredado por los actores que comparten este rol.

- Concordancia: Extrae una subsecuencia de acciones que aparecen en diferentes lugares del flujo de casos de uso y es expresado por separado.

2.5 Descripción de las historias de usuario del sistema

Después de realizar un análisis de los requisitos funcionales se elaboraron los casos de uso que describen toda la secuencia entre el actor y el sistema.

A continuación de describe el caso de uso Incluir Competencia.

Tabla 2: Historia de usuario. Incluir competencia

Número: 1	**Nombre del requisito: Incluir competencia**
Programador: Rioger Hernández López	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: NA	**Tiempo Real:** 2 días

Descripción:

1- Objetivo:

Permitir incluir una competencia

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

Para incluir competencias hay que:

- Tener en cuenta los siguientes datos: nombre y descripción de la competencia.
- Estar autenticado en el sistema con el rol de profesor.

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

Los campos nombre y descripción son obligatorios.

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

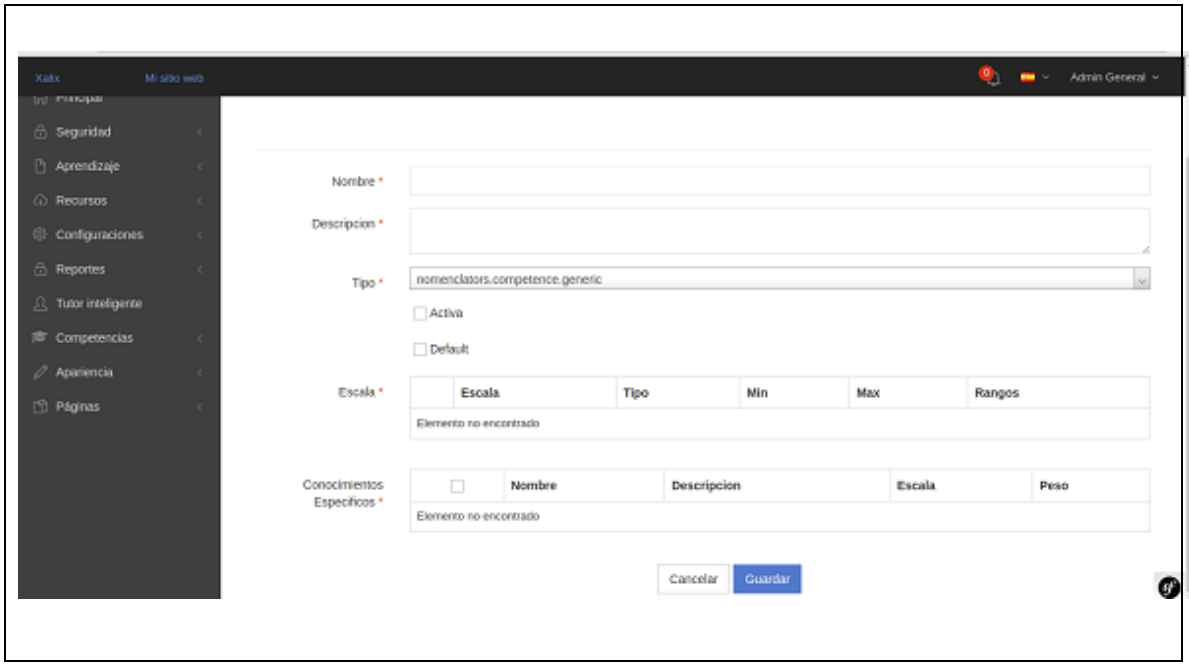
Descripción: campo de texto que permite cualquier carácter

4- Flujo de la acción a realizar:

- El sistema debe permitir incluir una competencia, esta acción puede realizarse seleccionando la opción incluir en el listado de competencia.
- Cuando el usuario incluye de forma correcta los datos necesarios y selecciona la opción Guardar, se muestra un mensaje de información de que la competencia fue incluida de forma correcta.
- Si los datos están incompletos o incorrectos se señalarán los campos en cuestión dando la posibilidad al usuario de realizar nuevamente la acción en cuestión.
- Si selecciona la opción Cancelar regresará a la vista previa.

Observaciones: las categorías son para agrupar los cursos y pueden tener dependencias con otras categorías.

Prototipo de interfaz:



2.6 Patrón arquitectónico

Symfony está basado en un patrón clásico del diseño web conocido como arquitectura Modelo vista Controlador, en lo adelante MVC, que está formado por tres niveles:

- Modelo: Esta es la representación específica del dominio de la información sobre la cual funciona la aplicación. El modelo es otra forma de llamar a la capa de dominio. La lógica de dominio añade significado a los datos.
- Vista: Esta presenta el modelo en un formato adecuado para interactuar, usualmente un elemento de interfaz de usuario.
- Controlador: se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

El principio más importante de la arquitectura MVC es la separación del código del programa en tres capas, dependiendo de su naturaleza. La lógica relacionada con los datos se incluye en el modelo, el código de la presentación en la vista y la lógica de la aplicación en el controlador. La programación se puede simplificar si se utilizan otros patrones de diseño. De esta forma, las capas del modelo, la vista y el controlador se pueden subdividir en más capas.

2.7 Modelo de diseño

El modelo de diseño se crea tomando el modelo de análisis como entrada principal, pero adapta el entorno de implementación elegido, este funciona como esquema para la implementación. También define clasificadores (clases, subsistemas e interfaces), relaciones entre sus clasificadores y colaboraciones que llevan a cabo los casos de uso.

2.7.1 Patrones de diseño

Patrones GRASP: Patrones para asignar responsabilidades, son parejas de problema solución con un nombre, que codifican buenos principios y sugerencias relacionados frecuentemente con la asignación de responsabilidades.

- Experto: La responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos). Una clase, contiene toda la información necesaria para realizar la labor que tiene encomendada. Hay que tener en cuenta que esto es aplicable mientras estemos considerando los mismos aspectos del sistema:
    - Lógica de negocio
    - Persistencia a la base de datos
    - Interfaz de usuario
- Creador: Se asigna la responsabilidad de que una clase B cree un Objeto de la clase A solamente cuando
    - B contiene a A.
    - B es una agregación (o composición) de A.
    - B almacena a A.
    - B tiene los datos de inicialización de A (datos que requiere su constructor)
    - B usa a A.

La creación de instancias es una de las actividades más comunes en un sistema orientado a objetos. En consecuencia, es útil contar con un principio general para la asignación de las responsabilidades de creación. Si se asignan bien el diseño puede soportar un bajo acoplamiento, mayor claridad, encapsulación y reutilización.

- Bajo Acoplamiento: Debe haber pocas dependencias entre las clases. Para determinar el nivel de acoplamiento de clases, son muy buenos los diagramas de colaboración de UML. Uno de los principales síntomas de un mal diseño y alto

acoplamiento es una herencia muy profunda. Siempre hay que considerar las ventajas de la delegación respecto de la herencia**.**

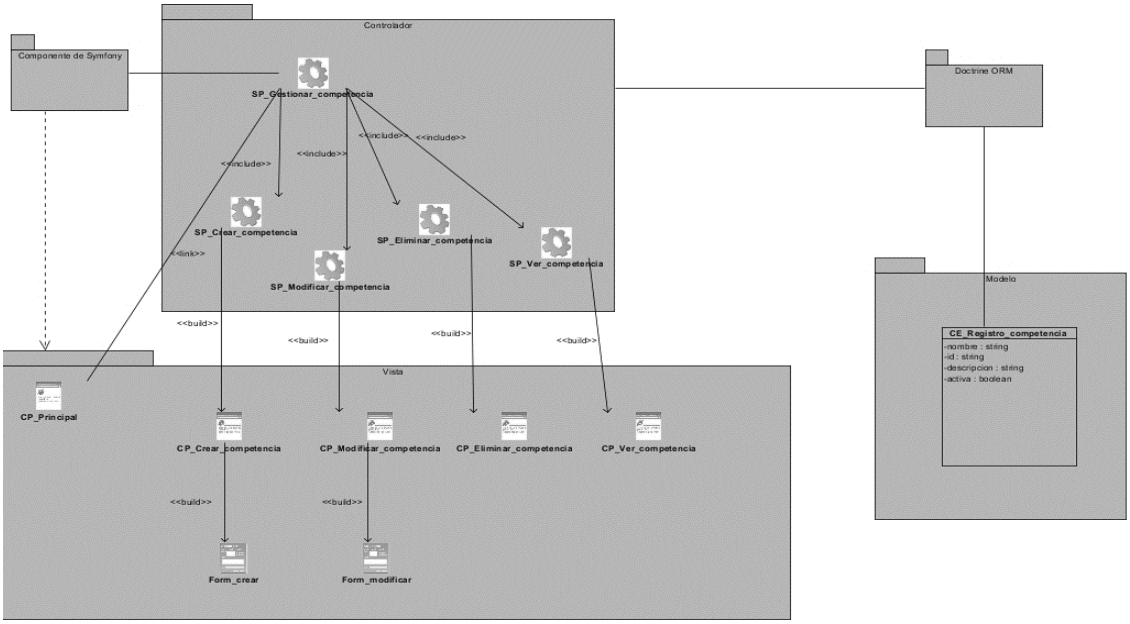
- Alta Cohesión: Cada elemento de nuestro diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable.

  Ejemplos de una baja cohesión son clases que hacen demasiadas cosas. En todas las metodologías se considera la refactorización. Uno de los elementos a refactorizar son las clases saturadas de métodos. Ejemplos de buen diseño se producen cuando se crean los denominados “paquetes de servicio” o clases agrupadas por funcionalidades que son fácilmente reutilizables (bien por uso directo o por herencia).

- Controlador: Asignar la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades (validaciones, seguridad, etc.). El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión. Un error muy común es asignarle demasiada responsabilidad y alto nivel de acoplamiento con el resto de los componentes del sistema.

2.7.2 Diagramas de clases del diseño

Figura 2: Diagrama de diseño del caso de uso gestionar competencia

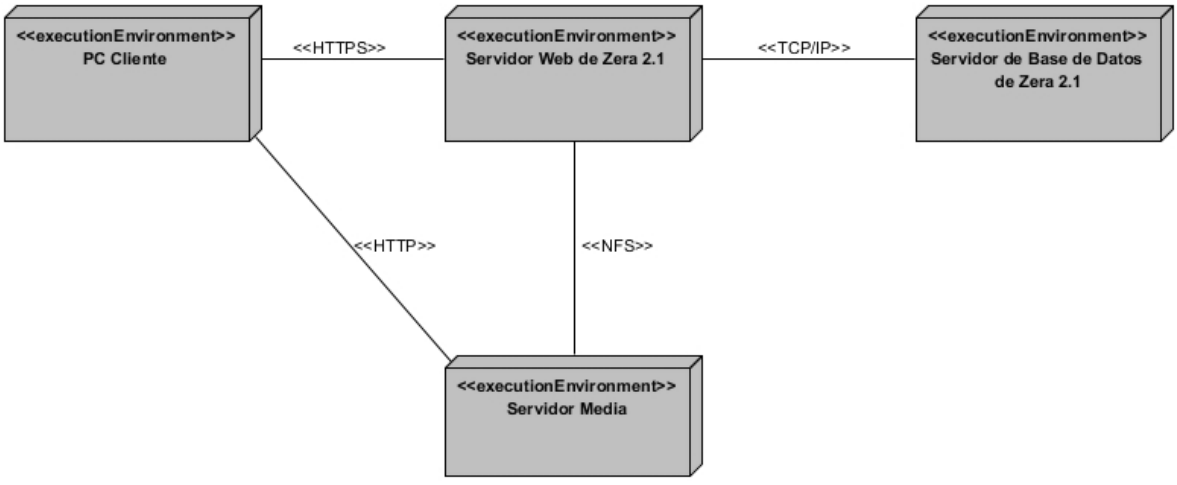


2.8 Modelo de despliegue

Los diagramas de despliegue muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados mediante enlaces de comunicación. A continuación, se describen los elementos que componen el diagrama de despliegue correspondiente al presente trabajo de diploma.

- Nodo PC Cliente: se refiere a las computadoras que utilizarán los usuarios para interactuar con la aplicación. Se comunica con el Servidor de Aplicación a través del protocolo HTTP.
- Nodo Servidor de Aplicación: representa el servidor web donde se encuentra instalado el sistema.
- Nodo Servidor de Base de Datos: es el servidor PostgresSQL donde se almacena la base de datos del sistema.

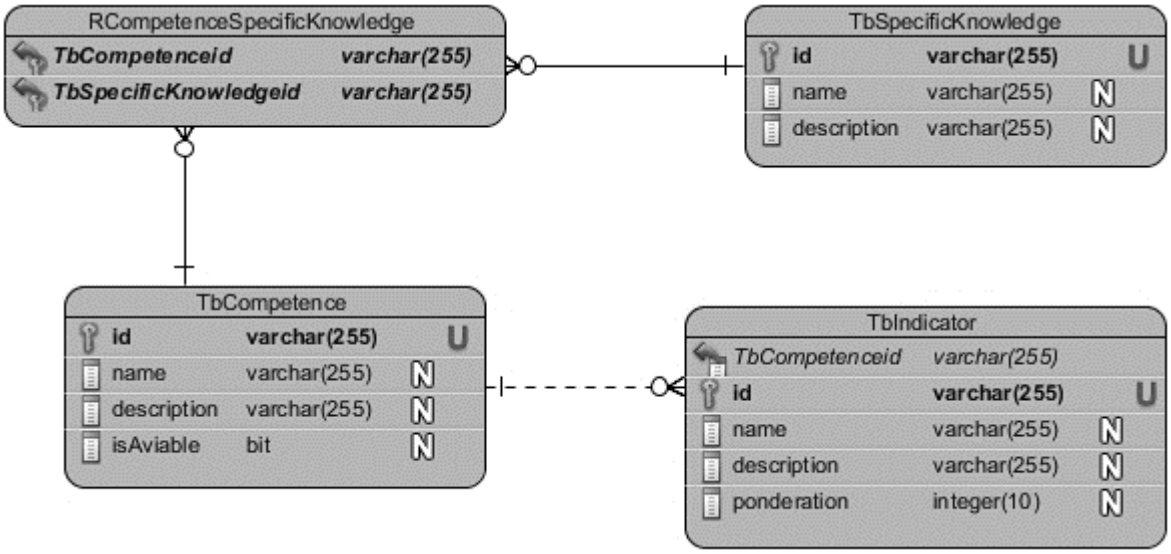
Figura 3: Diagrama de despliegue



2.9 Diseño de la base de datos

2.9.1 Diagramas entidad-relación

Figura 4:Modelo entidad-relación



2.9.2 Descripción de las tablas de la base de datos

En esta sección se realiza la descripción de las tablas de la base de datos relacionadas con el requisito funcional Gestionar Competencia.

Tabla 3: Descripción de la tabla tb_Competence

tb_Competence		
Descripción: Almacena los datos de las competencias.		
Atributo	**Tipo**	**Descripción**
Id	varchar	Campo que contiene el identificador de la competencia.
name	varchar	Campo que contiene el nombre de la competencia.

description	varchar	Campo que contiene la descripción del indicador.
isAvailable	boolean	Campo te indica si la competencia está activa

Tabla 4: Descripción de la tabla tb_Indicator

tb_Indicator		
Descripción: Almacena los indicadores que posee la competencia.		
Atributo	**Tipo**	**Descripción**
id	varchar	Campo que contiene el identificador de la competencia.
name	varchar	Campo que contiene el nombre de la competencia.
description	varchar	Campo que contiene la descripción del indicador.
ponderation	Integer (10)	Campo que contiene el nivel de importancia del indicador.

Tabla 5: Descripción de la tabla tb_RCompetenceSpecificContent

RCompetenceSpecificKnowledge		
Descripción: Representa la relación entre los contenidos específicos y la competencia		
Atributo	**Tipo**	**Descripción**
id	varchar	campo que contiene el identificador de la tabla.
name	varchar	Campo que contiene el nombre de la competencia.
description	varchar	Contiene la descripción del indicador.

Conclusiones del capítulo

El artefacto generado tales como: diagrama de diseño, diagrama entidad-relación, diagrama de despliegue, diagrama de dominio, historias de usuario entre otros, permitieron alcanzar un mayor entendimiento de lo que el sistema final debe propiciar. Se generó el diseño de la base de datos en la que quedaron plasmadas todas las relaciones de las tablas como por ejemplo la relación existente entre la tabla competencia y la tabla conocimiento específico, además, el diagrama de despliegue permitió obtener detalladamente los nodos físicos como son: PC Cliente, Servidor Web, Servidor de Base de Datos, este diagrama también nos muestra la forma por las cuales se comunican estos nodos físicos, las cuales son los protocolos HTTP, TCP/IP y NFS.

CAPÍTULO 3: EVALUACIÓN DE LA SOLUCIÓN

Introducción del capítulo

Los artefactos generados en el diseño, constituyen la entrada esencial para el flujo de trabajo de implementación, el cual propone mostrar cómo se implementan las funcionalidades en términos de componentes y cómo lograr la organización de los mismos. Después de realizar este flujo se elabora el de pruebas, donde se efectúan las pruebas necesarias a las funcionalidades implementadas para verificar que estén correctamente desarrolladas, permitiendo comprobar su adecuado funcionamiento antes de ser usado por los usuarios finales.

3.1 Modelo de implementación

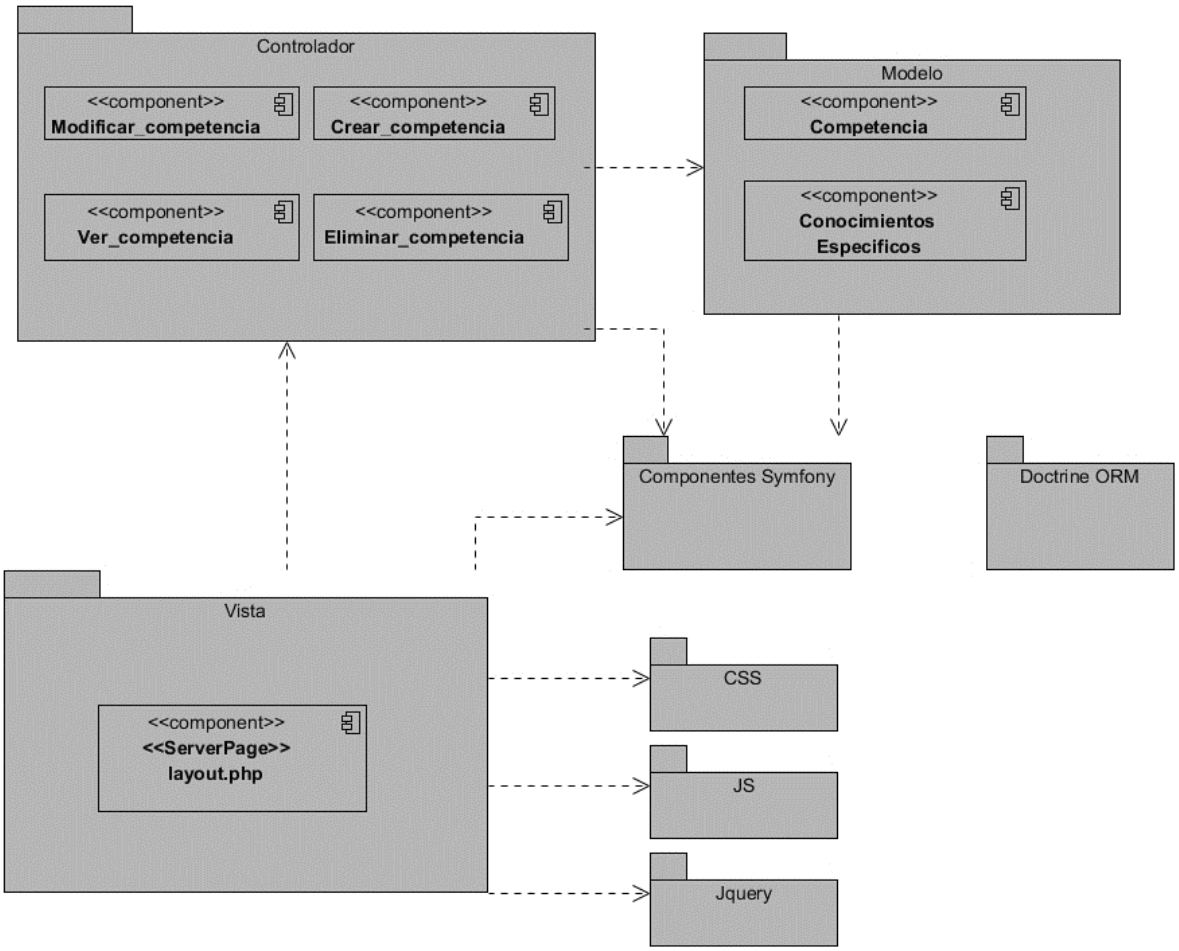
El modelo de implementación describe cómo los elementos del diseño se implementan en términos de componentes. Describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y los lenguajes de programación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

Los diagramas de despliegue y componentes, que son artefactos generados en este flujo de trabajo, conforman lo que se conoce como un modelo de implementación al describir los componentes a construir, su organización y dependencia entre nodos físicos en los que funcionará la aplicación. (46)

3.1.1 Diagrama de componentes

Los diagramas de componente permiten modelar la vista de implementación del sistema a partir del cual se construye la aplicación. Muestran la organización y las dependencias lógicas entre un conjunto de componentes de software y organiza los subsistemas de implementación en capas. Un componente es el empaquetamiento físico de los elementos de un modelo, como son las clases en el modelo de diseño. (46)

Figura 5 Diagrama de componentes Gestionar competencia



3.1.2 Estándar de codificación

JavaScript

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

Ejemplo:

Figura 6: Ejemplo de código JavaScript

```
function addItemForm($collectionHolder, $newLinkLi) {

    var prototype = $collectionHolder.data('prototype');
    var index = $collectionHolder.data('index');
    var newForm = prototype.replace(/__name__/g, index);

    $collectionHolder.data('index', index + 1);
    var $newFormLi = $('<li class="list-unstyled"></li>').append(newForm);
    $($newFormLi).css({'border': '1px solid #b3b3b3', 'margin': '5px', 'padding': '15px'});

    // instanciando los select del formulario
    $($newFormLi).find('select').select2({theme: 'bootstrap'});


    $newLinkLi.before($newFormLi);
    addItemFormDeleteLink($newFormLi);
    removeLabelsOfEmbededForm($newFormLi);


}
```

HTML

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

Ejemplo:

Figura 7: Ejemplo de código HTML

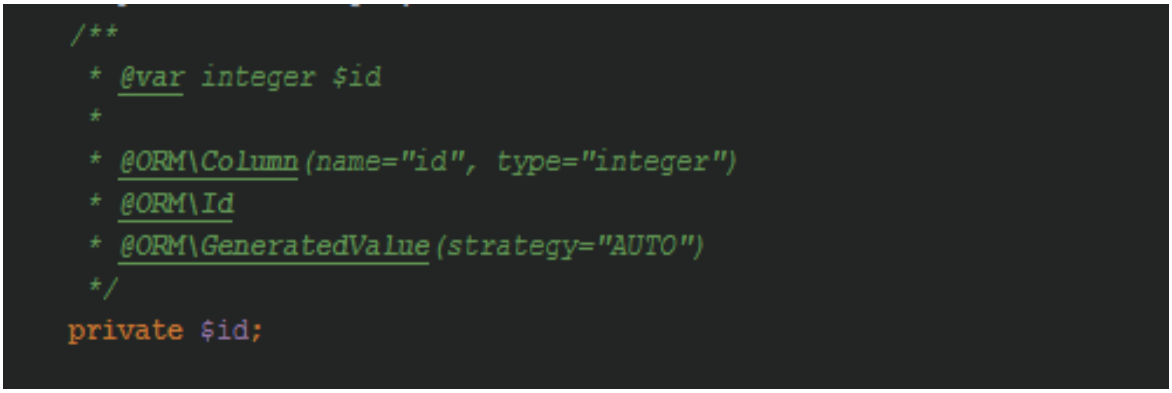
```
<div class="table-responsive">
    <table id="table#viewcompetence" class="table table-striped table-bordered table-hover dataTable no-footer">
        <thead>
        <tr>
            <th>{{ 'competence.name'|trans }}</th>
            <th>{{ 'competence.description'|trans }}</th>
            <th>{{ 'competence.type'|trans }}</th>
            <th>{{ 'competence.isAviable'|trans }}</th>
            <th>{{ 'competence.specificKnowledge'|trans }}</th>
        </tr>
        </thead>
        <tbody>

        {% for evaluationForm in evaluationForms %}
            <tr id="{{ competenceForm.id }}" class="row-evaluationForm-info" >
                <td>{{ competetenceForm.name }}</td>
                <td>{{ competetenceForm.description }}</td>
                <td>{{ competetenceForm.type }}</td>
                <td>{{ competetenceForm.isAviable }}</td>
                <td>{{ competetenceForm.specificKnoweldge }}</td>


            </tr>
        {% endfor %}
        </tbody>
    </table>
</div>
```

PHP

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

Ejemplo:

Figura 8: Ejemplo de código PHP

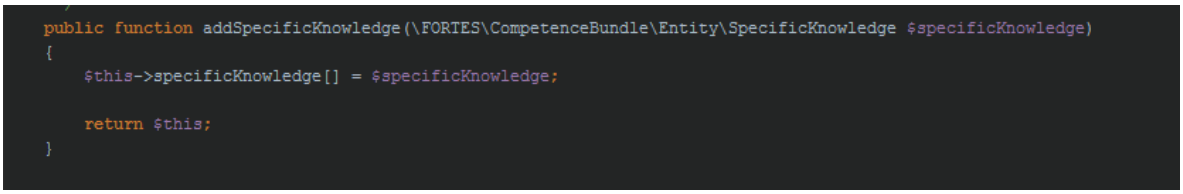


Convención de nombres

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

Ejemplo:

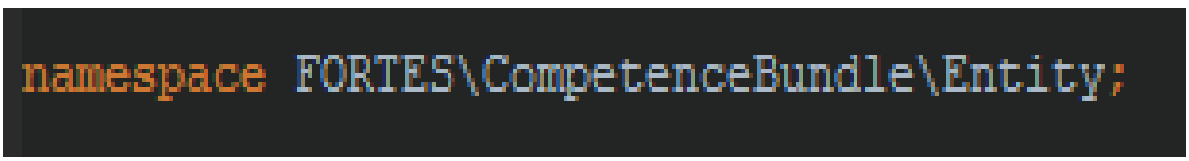
Figura 9: Ejemplo de código PHP



Descripción: se utiliza namespaces para todas las clases.

Ejemplo:

Figura 10: Ejemplo de código PHP



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

- ✓ Los nombres de los servicios contienen grupos separados por puntos.

- ✓ El alias de Inyección de Dependencias del bundle es el primer grupo.
- ✓ Se utiliza minúsculas para los nombres de servicios y sus parámetros.
- ✓ Un nombre de grupo utiliza la notación guion bajo.

Ejemplo:

Figura 11: Ejemplo de código PHP

```
competence.form.competence_type:
    class: FORTES\CompetenceBundle\Form\CompetenceType
    arguments: ['@service_container']
    tags:
        - { name: form.type, alias: xalix_competence_type }
```

Comentarios en el código

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

Ejemplo:

Figura 12: Ejemplo de código comentado

```
// ************ MENU COmpetence **********************//
```

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

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

Ejemplo:

Figura 13: Ejemplo de código comentado

```
        * Load data fixtures with the passed EntityManager.
        *
        * @param ObjectManager $manager Manager of Object.
        *
        * @return void
        */
    public function load(ObjectManager $manager)
    {
        $menuParent = $this->container->get('ktw_database_menu.provider')->get('backend_left_menu');
        $factory    = $this->container->get('ktw_database_menu.factory');
```

3.2 Pruebas de software

Las pruebas del software son una técnica dinámica de validación y verificación (V&V), las cuales implican ejecutar una implementación del *software* con datos de prueba, se examinan las salidas del software y su entorno operacional para comprobar que funciona tal y como se requiere.

Con la ejecución de las pruebas de software se persigue descubrir defectos en el sistema asociados a comportamientos incorrectos o no deseables y para verificar que cumple con los requerimientos del cliente, con el fin de suplir sus necesidades. Al ejecutarse esta actividad no se obtiene un sistema totalmente libre de errores, pero si apto para ser usado por el usuario final.

3.2.1 Niveles de pruebas

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

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

Pruebas de aceptación: comprueban el comportamiento del sistema frente a los requisitos del cliente (suele participar el mismo cliente o los usuarios). Se lleva a cabo con el objetivo de que el cliente valide los requisitos que debe poseer el sistema. (47)

3.2.2 Métodos de prueba

Existen distintas técnicas de pruebas que proporcionan criterios para generar casos de pruebas que provoquen fallos en los programas. Se agrupan en:

- **Técnicas de caja blanca o estructural:** se basan en un minucioso examen de los detalles procedimentales del código a evaluar, por lo que es necesario conocer la lógica del programa.
- **Técnicas de caja negra o funcional:** realiza pruebas sobre la interfaz del programa a probar, entendiendo por interfaz las entradas y salidas de dicho programa. No es necesario conocer la lógica del programa, únicamente la funcionalidad que debe realizar. (47)

Para la validación de la propuesta de solución será aplicado el método de caja negra, debido a que este permitirá corregir problemas en la interfaz del usuario y se comprobará que esta propuesta realiza las funciones requeridas por el usuario.

3.2.3 Partición equivalente

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

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

3.2.4 Diseños de caso de prueba

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

Tabla 6: Caso de prueba Incluir competencia

Tabla 6: Caso de prueba Incluir competencia								
Caso de Prueba: Incluir competencia								
Descripción: La funcionalidad inicia cuando una persona con rol de administrador decide incluir una competencia. Para ello el administrador selecciona opción competencia del menú de administración la opción incluir competencia. El sistema muestra un formulario con los campos correspondientes. Una vez llenado el formulario se selecciona la opción aceptar y se guarda el formulario en la base de datos. **Condición de ejecución:** La persona debe estar autenticada como administrador								
Escenario	descripción	nombre	descripción	tipo	activa	Conocimientos específicos	Respuesta del sistema	Flujo central
EC1.1	Seleccione incluir para insertar nueva competencia						Se muestra un formulario con los siguientes campos Nombre Descripción Tipo Activa Conocimientos específicos	Competencia /Nueva competencia
EC1.2	Introduce los datos y seleccione Aceptar	V	V	V	V	V	Valida los datos y registra una nueva competencia	Competencia /Nueva competencia/ Aceptar
EC1.3	Seleccione la opción cancelar						Elimina los datos y regresa a la vista anterior	Competencia /Nueva competencia/ Cancelar
EC1.4	Existen campos vacíos	V	I	I	V	V	Muestra un mensaje informando que existen campos vacíos, muestra un indicador sobre el campo vacío. Regresa a la EC1.2	Competencia /Nueva competencia/ Aceptar

Tabla 7: Caso de prueba Editar competencia

Tabla 6: Caso de prueba Editar competencia								
Caso de Prueba: Editar competencia								
Descripción: La funcionalidad inicia cuando el administrador decide editar una competencia. Para ello selecciona en la sección de competencias la que él desea editar. El sistema muestra un formulario con los campos correspondientes. Una vez editado el formulario se selecciona la opción aceptar y se guardan los cambios en la base de datos. **Condición de ejecución:** La persona debe estar autenticada como administrador								
Escenario	descripción	nombre	descripción	tipo	activa	Conocimientos específicos	Respuesta del sistema	Flujo central
EC1.1	Selecciona la competencia que va a editar	n/a	n/a	n/a	n/a	n/a	Se muestra un formulario con los siguientes campos • Nombre • Descripción • Tipo • Activa • Conocimientos específicos	Competencia/ Editar competencia
EC1.2	Introduce los datos y seleccione Aceptar	V	V	V	V	V	Valida los datos y registra una nueva competencia	Competencia/ Editar competencia/Aceptar
EC1.3	Seleccione la opción cancelar						Elimina los datos y regresa a la vista anterior	Competencia/ Editar competencia/Cancelar
EC1.4	Existen campos vacíos	V	I	I	V	V	Muestra un mensaje informando que existen campos vacíos, muestra un indicador sobre el campo vacío. Regresa a la EC1.2	Competencia/ Editar competencia/Aceptar

Tabla 8: Caso de prueba Editar competencia

Tabla 4: Caso de prueba Eliminar competencia								
Caso de Prueba: Eliminar competencia								
Descripción: La funcionalidad inicia cuando el administrador decide eliminar una competencia. Para ello selecciona en la sección de competencias la que él desea eliminar. El sistema muestra un mensaje de confirmación, si se selecciona si, se elimina la competencia, en caso contrario no se elimina la competencia **Condición de ejecución:** La persona debe estar autenticada como administrador								
Escenario	descripción	nombre	descripción	tipo	activa	Conocimientos específicos	Respuesta del sistema	Flujo central
EC1.1	Selecciona la competencia que va a eliminar	n/a	n/a	n/a	n/a	n/a	El sistema muestra un mensaje de aceptación	Competencia/ Eliminar competencia
EC1.2	Si se selecciona la opción ACEPTAR	n/a	n/a	n/a	n/a	n/a	El sistema elimina la competencia	Competencia/ Editar competencia/Aceptar
EC1.3	Seleccione la opción cancelar	n/a	n/a	n/a	n/a	n/a	El sistema no elimina la competencia	Competencia/ Editar competencia/Cancelar

3.2.5 Resultados obtenidos en las pruebas

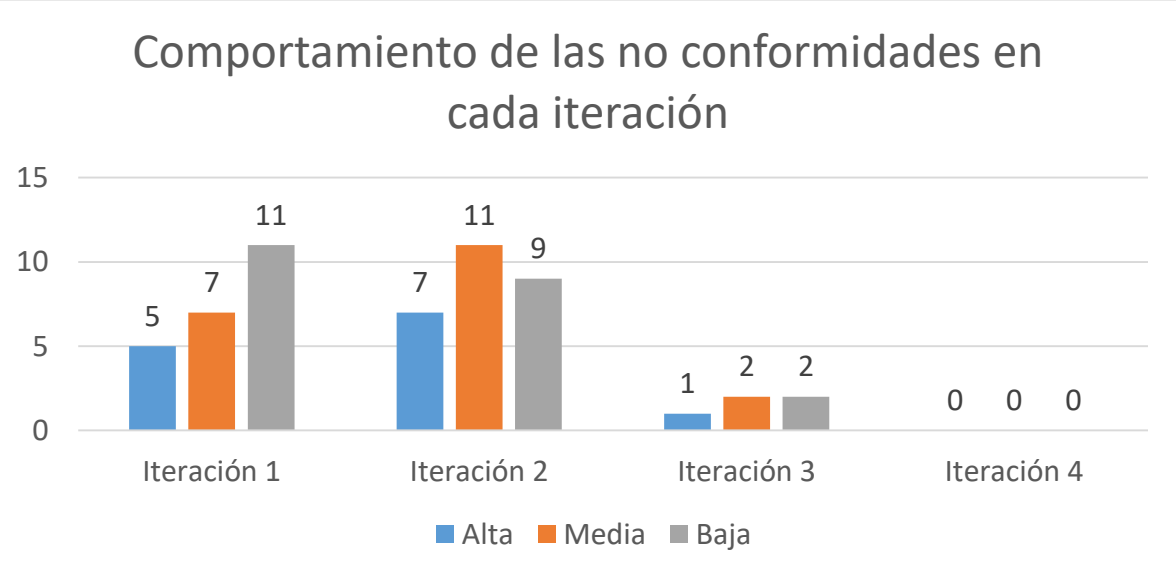
En las pruebas realizadas al sistema mediante el método de caja negra, se pudo comprobar el cumplimiento de los requisitos funcionales y no funcionales del software obtenido.

Figura 14:Resultados obtenidos de las pruebas

Iteraciones	**Cantidad de casos de prueba**	**No conformidades detectadas**			
		Alta	**Media**	**Baja**	**Total**
1	19	5	7	11	23
2	19	7	11	9	26

3	19	1	2	2	5
4	19	0	0	0	0

A continuación, se muestra un gráfico que muestra el comportamiento de las no conformidades en cada iteración.



Conclusiones del capítulo

Los artefactos generados durante la disciplina de implementación permitieron entender la relación de dependencia y por último el desarrollo final de la solución. Además, se aplicaron pruebas de sistema y de aceptación, permitiendo determinar y erradicar las deficiencias encontradas, obteniendo finalmente una solución con un alto nivel de calidad.

CONCLUSIONES GENERALES

Luego de culminar el proceso de análisis, diseño, implementación y pruebas desarrollados al sistema módulo competencia para la Plataforma educativa XAUCE Zera, se arribó a las siguientes conclusiones:

- El estudio realizado sobre los principales conceptos asociados a la evaluación competencia y de plataformas con características similares permitió obtener una visión de las características en entornos de aprendizaje virtual, las facilidades que brinda el uso de los modelos de evaluación y cómo son implementados estos métodos de evaluación, obteniendo experiencias que forman parte de la propuesta de solución de la presente investigación.
- La metodología AUP-UCI y las herramientas y tecnologías tales como: NGINX como servidor web, PostgresSQL como servidor de base de datos, Symfony como framework de desarrollo, PHP como lenguaje de programación, posibilitaron el desarrollo de la capa de abstracción, y la correcta documentación del proceso permitió crear las historias de usuarios, y los distintos diagramas para un mejor entendimiento de la propuesta de solución.
- Como propuesta de solución se creó un módulo competencia que apoya el proceso de formación de competencias para el aprendizaje de los estudiantes en la Plataforma Educativa XAUCE Zera v2.1 acompañada de la documentación correspondiente a todo el proceso de implementación.
- Se realización de las pruebas de aceptación y de sistema realizadas permitió solucionar errores y demostrar que las funcionalidades fueron correctamente implementadas.

RECOMENDACIONES

Para la continuidad de la investigación se recomienda:

- Darle seguimiento a la propuesta de solución integrándola al instrumento de evaluación de tipo cuestionario.
- Darle continuidad al estudio realizado indagando conceptos y definiciones dados por otros autores que hallan incursionado en el tema, buscando nuevas características y nuevas formas de aplicar la evaluación por competencia en plataformas educativas.

REFERENCIAS BIBLIOGRÁFICAS

1. **Tobón, Sergio.** *Competencias, calidad y educación superior.* Colombia : s.n., 2006.
2. **Martínez, Magda Cejas.** *La educación basada en competencia.* Venezuela : s.n.
3. **Iglesias, Dra. Magalys Ruiz.** La evaluación de competencias. Junio 2008.
4. **Lorenzo, Aretio García.** *Sociedad del Conocimiento y la Educación.* Madrid : s.n., 2012.
5. **Andalucía, Consejería de Educación de la Junta de.** *Orientaciones para la evaluación del alumno en la Educación Secundaria Obligatoria.* Andalucía : s.n., 2012.
6. **Perrenoud.** 2004.
7. ***Gestión de Recursos Humanos por Competencia.***
8. **Ansorena Cao, Álvaro.** ***15 casos de selección personal con éito.*** **Barcelona : Paidós Empresa, 1996.**
9. **Rodriguez, Nelson T. y Felix, Pedro S.** ***Curso Basico de Psicometría.***
10. **Marelli, Anne.** ***Introducción al análisis y desarrollo del modelo de competencia.*** **2000.**
11. **Ileana Hernández Darias, Sonia Fleitas Triana, Diana Salazar Fernández. LA GESTIÓN DEL CAPITAL HUMANO EN EMPRESAS CUBANAS Y SUS PARTICULARIDADES. 2011.**
12. **Tuning.** ***Informe final Proyecto Tuning América Latina.*** **2007.**
13. **Corominas. 2001.**
14. **Iglesias, Magalys Ruiz.** ***La evaluación basada en competencias.*** **2014.**
15. **Iglesias, Dra. Magalys Ruiz.** ***Maestría internacional de competencias profesionales.*** **Universidadde la Mancha, Castilla : s.n., junio 2008.**
16. **Tobón, Sergio.** ***Competencias,calidad y educación superior.*** **2006.**
17. **Plataforma Educativa XAUCE Zera. [En línea] 2018. https://eva.uci.cu/es/aboutAs.**
18. **[En línea] http://web.archive.org/web/20170305221707/http://cent.uji.es/doc/eveauji_es.pdf.**
19. **[En línea] https://sourceforge.net/articles/.**
20. **[En línea] https://web.archive.org/web/20170110203848/http://www.dokeos.com:80/fr/essai-gratuit/.**
21. **[En línea] https://www.uv.es/avirtual/manual/ch02s01.html.**
22. **[En línea] https://docs.moodle.org/35/en/Main_page.**
23. **Esteban, Gabriel Maida y Pacienzia, Julian.** ***Metodología de Desarrollo de Software.*** **2015.**
24. **COMPUTACIÓN, DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y.** ***Rational Unified Process.*** **Universidad Politécnica de Valencia : s.n.**
25. **FIGUEROA, Roberth G., SOLÍS, Camilo J. and CABRERA, Armando A.** ***Metodologías Tradicionales Vs. Metodologías Ágiles.***
26. **Beck, Kent. Extreme Programming Explained. 1999.**
27. **XP vs Scrum. [En línea] http://www.ferolen.com/blog/xp-vs-scrum/..**
28. **Sánchez, Tamara Rodríguez.** ***Metodología de desarrollo para la actividad productiva de la UCI.***
29. **Orallo, E. Hernández.** ***El Lenguaje Unificado UML.*** **2014.**
30. **[En línea] http://w3c.es/Divulgacion/GuiasBreves/XHTML..**
31. **[En línea] http://w3c.es/Divulgacion/GuiasBreves/HojasEstilo..**
32. **Pérez, Javier Eguíluz.** ***Introducción a JavaScript.*** **2008.**
33. **—.** ***Introducción a AJAX.*** **2008.**

34. Cracteristicas de PHP. [En línea] http://www.linuxcentro.net/linux/staticpages/index.php?page=CaracteristicasPHP.
35. [En línea] http://w3c.es/Divulgacion/GuiasBreves/TecnologiasXML..
36. Autores, Colectivo de. Introducing Bootstrap. [En línea] http://twitter.github.com/bootstrap/.
37. Potencier, Fabien y Francois, Zaniotto. *Symfony la guía definitiva.*
38. [En línea] http://www.doctrine-project.org.
39. Visual Paradigm. [En línea] http://www.visual-paradigm.com/.
40. [En línea] http://www.guia-ubuntu.org/index.php?title=PostgreSQL.
41. Espinoza, Humberto. *Soluciones tecnológicas al servicio de la nación.* 2005.
42. S, Corona. *Nginx: A Practical Guide to High Performance.* Estados Unidos : O'Reilly Media., 2008.
43. Oracle Corporation. NetBeans. [En línea] 2017. http://netbeans.org/community/releases/69/index_es.html..
44. S, Chacon. *Pro Git, el libro oficial de Git.* 2014. 2da edicción.
45. Sommerville, Ian. *Ingeniería de Software.* 2005.
46. Jacobson, Ivar, Booch, Grady y Rumbaugh, James. *El proceso unificado de desarrollo de software.* 2018.
47. Londoño, Jorge Hernan Abad. Tipos de pruebas de software. [En línea] 2018. http://ing-sw.blogspot.com/2005/04/tipos-de-pruebas-de-software.html.
48. Sánchez, Tamara Rodríguez. *Metodoligía de desarrollo para la actividad productiva de la UCI.* 2015.
49. Patrones de diseño. [En línea] https://msdn.microsoft.com/es-es/library/bb972240.aspx.

ANEXOS

Figura 15: Diagrama de caso de uso personalizar competencia

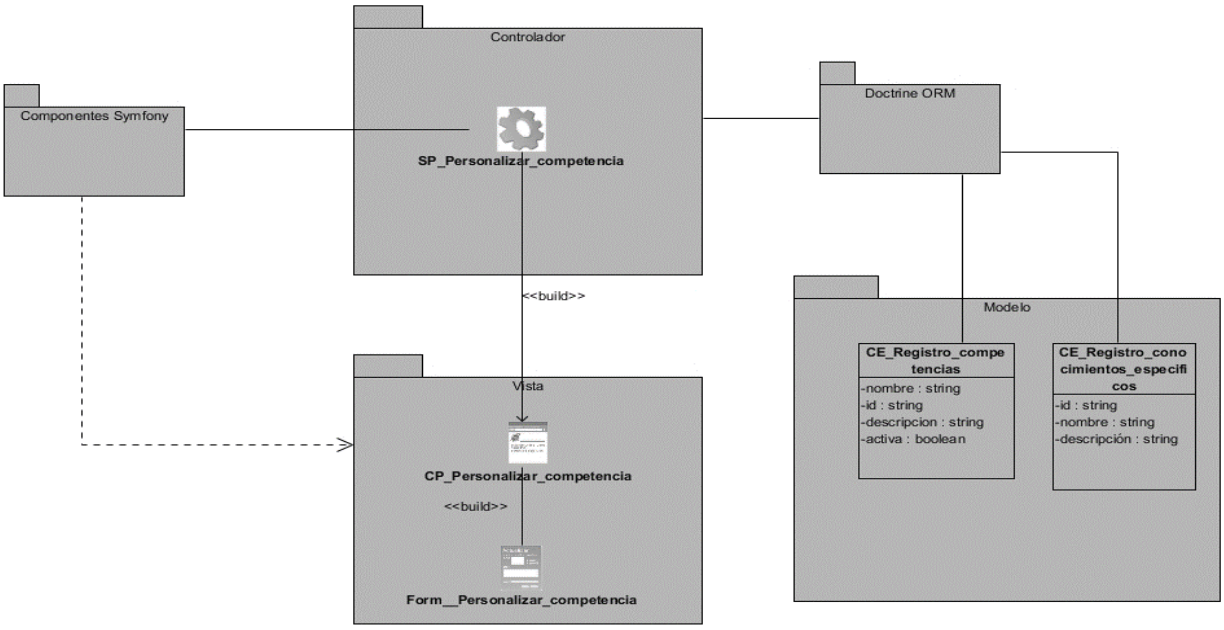


Figura 16: Diagrama de caso de uso Mostrar competencias desarrolladas

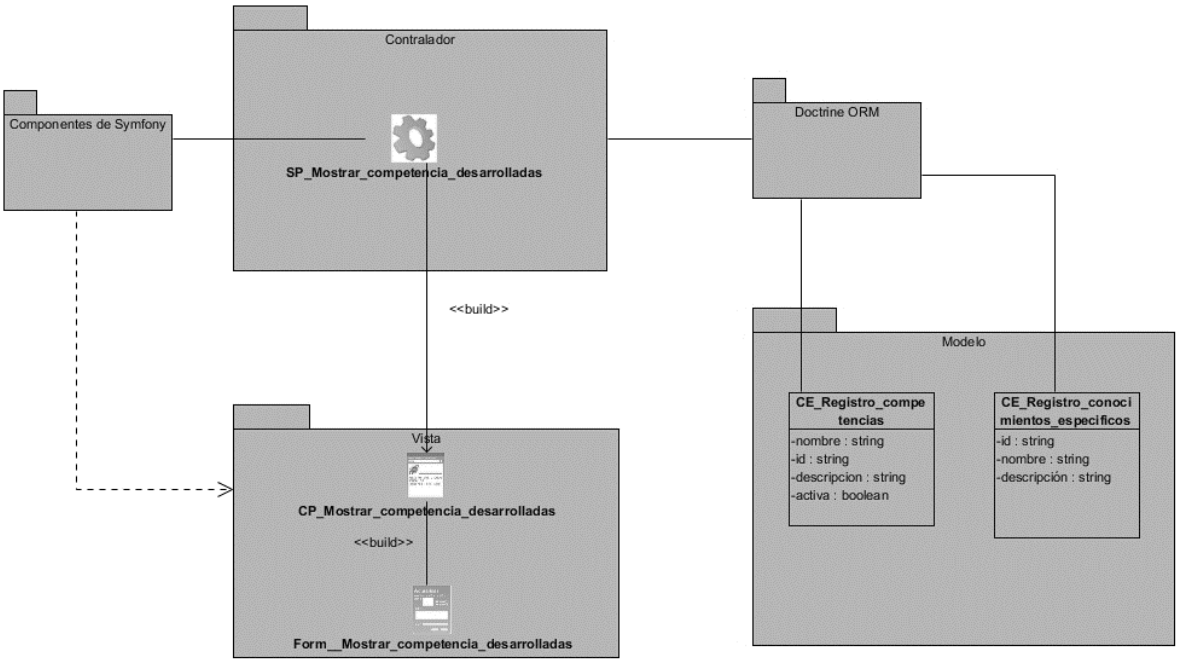


Figura 17:Diagrama de caso de uso Mostrar estado de competencias

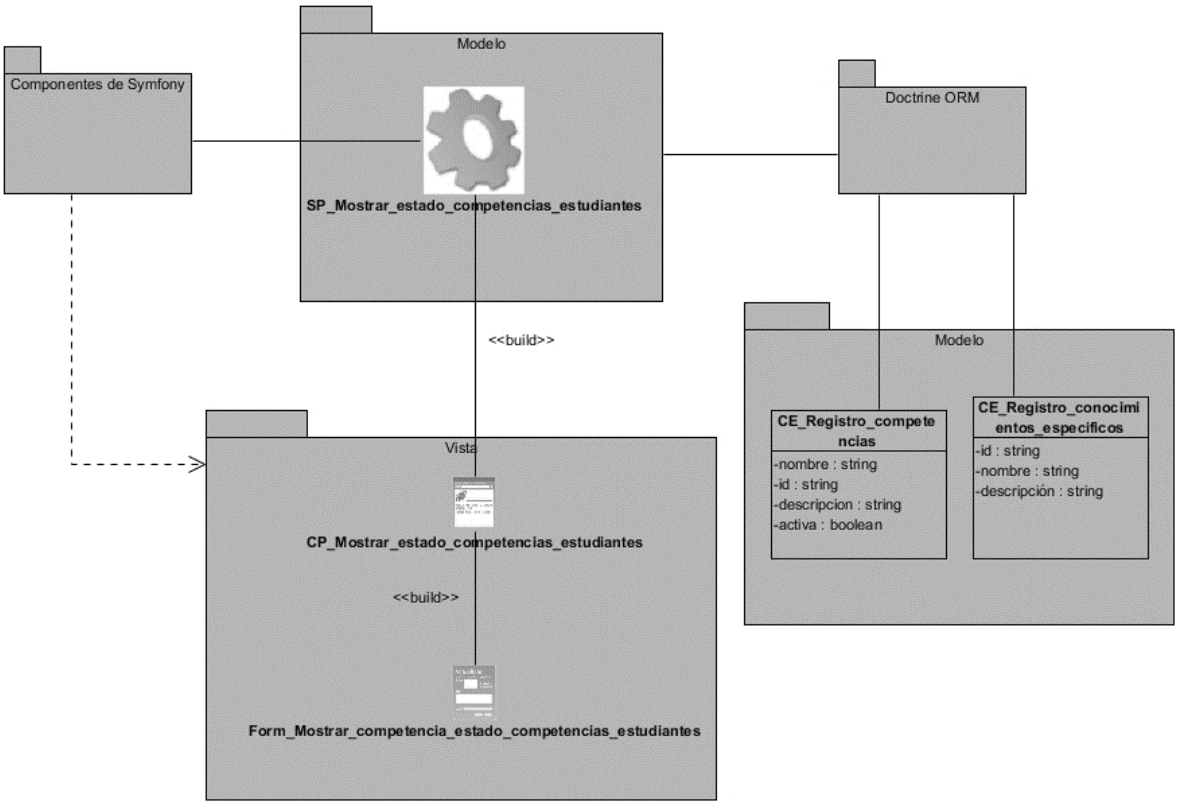


Tabla 9:Historia de usuario Modificar competencia

Número: 2	**Nombre del requisito: Modificar competencia**
Programador: Rioger Hernández López	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: NA	**Tiempo Real:** 2 días
Descripción:	

1- Objetivo: Permitir modificar una competencia. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para modificar competencias hay que: - Tener en cuenta los siguientes datos: nombre y descripción de la competencia. - Estar autenticado en el sistema con el rol de profesor. **3- Comportamientos válidos y no válidos (flujo central y alternos):** Los campos nombre y descripción son obligatorios. Nombre: campo de texto que admite caracteres alfabéticos y tiene un máximo de hasta 100 caracteres. Descripción: campo de texto que permite cualquier carácter. **4- Flujo de la acción a realizar:** - El sistema debe permitir modificar una competencia, esta acción puede realizarse seleccionando la opción modificar en el listado de competencia. - Cuando el usuario incluye de forma correcta los datos necesarios y selecciona la opción Guardar, se muestra un mensaje de información de que la competencia fue modificada de forma correcta. - Si los datos están incompletos o incorrectos se señalarán los campos en cuestión dando la posibilidad al usuario de realizar nuevamente la acción en cuestión. - Si selecciona la opción Cancelar regresará a la vista previa.
Observaciones: las categorías son para agrupar los cursos y pueden tener dependencias con otras categorías.
Prototipo de interfaz:

Tabla 10: Historia de usuario Ver competencia

Número: 3	**Nombre del requisito: Ver competencia**

Programador: Rioger Hernández López	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: NA	**Tiempo Real:** 2 días
Descripción: **1- Objetivo:** Permitir ver una competencia. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para ver competencias hay que: - Estar autenticado en el sistema con el rol de profesor. **3- Comportamientos válidos y no válidos (flujo central y alternos):** Los campos nombre y descripción son obligatorios. Nombre: campo de texto que admite caracteres alfabéticos y tiene un máximo de hasta 100 caracteres. Descripción: campo de texto que permite cualquier carácter. **4- Flujo de la acción a realizar:** - El sistema debe permitir ver una competencia, esta acción puede realizarse seleccionando la opción ver en el listado de competencia.	

Tabla 11: Historia de usuario Eliminar competencia

Número: 4	**Nombre del requisito: Eliminar competencia**
Programador: Rioger Hernández López	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 3 días

Riesgo en Desarrollo: NA	**Tiempo Real:** 2 días
Descripción: **1- Objetivo:** Permitir eliminar una competencia. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para eliminar competencias hay que: - Tener en cuenta los siguientes datos: nombre y descripción de la competencia. - Estar autenticado en el sistema con el rol de profesor. **3- Comportamientos válidos y no válidos (flujo central y alternos):** Los campos nombre y descripción son obligatorios. Nombre: campo de texto que admite caracteres alfabéticos y tiene un máximo de hasta 100 caracteres. Descripción: campo de texto que permite cualquier carácter. **4- Flujo de la acción a realizar:** - El sistema debe permitir eliminar una competencia, esta acción puede realizarse seleccionando la opción eliminar en el listado de competencia. - Cuando el usuario elimina de forma correcta los datos necesarios, se muestra un mensaje de información de que la competencia fue eliminada de forma correcta. Si selecciona la opción Cancelar regresará a la vista previa.	

Tabla 12: Historia de usuario Mostrar competencia desarrolladas

Número: 6	**Nombre del requisito: Mostrar competencia desarrolladas**
Programador: Rioger Hernández López	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: NA	**Tiempo Real:** 2 días

Descripción:

1- Objetivo:

Permitir mostrar las competencias desarrolladas por los estudiantes.

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

Para mostrar competencias desarrolladas por los estudiantes hay que:

- Estar autenticado en el sistema con el rol de profesor.

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

Los campos nombre y descripción son obligatorios.

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

Descripción: campo de texto que permite cualquier carácter

4- Flujo de la acción a realizar:

- El sistema debe mostrar las competencias desarrolladas por los estudiantes, esta acción puede realizarse seleccionándola opción mostrar competencias del menú del curso.
- El sistema muestra una tabla con todas las competencias desarrolladas por los estudiantes.

Tabla 13: Historia de usuario Mostrar estado de las competencias de los estudiantes

Número: 7	**Nombre del requisito: Mostrar estado de las competencias de los**

	estudiantes
Programador: Rioger Hernández López	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: NA	**Tiempo Real:** 2 días

Descripción:

1- Objetivo:

Permitir mostrar el estado de las competencias que van desarrollando los estudiantes.

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

Para mostrar competencias estado de las competencias:

- Estar autenticado en el sistema con el rol de profesor.

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

Los campos nombre y descripción son obligatorios.

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

Descripción: campo de texto que permite cualquier carácter.

4- Flujo de la acción a realizar:

- El sistema debe mostrar el estado de las competencias de los estudiantes, esta acción puede realizarse seleccionándola opción mostrar estado de competencias del menú del curso.
- El sistema muestra una tabla con todos los estados de las competencias desarrolladas por los estudiantes.