Universidad de las Ciencias Informáticas

Facultad 6

Trabajo de diploma para optar por el título

Ingeniero en Ciencias Informáticas

Módulo para la evaluación de competencias en la Plataforma de Laboratorio Virtual y a Distancia

Autor: Luis Enrique Argota Vega

Tutores: Msc. Omar Mar Comelio

Ing. Reisel González Pérez

La Habana, julio de 2015

"Año 57 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaro ser el único autor del presente trabajo de diploma y reconozco 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 2015.

_____

Firma del Autor

Luis Enrique Argota Vega

_____

Firma del Tutor

Ing. Reisel González Pérez

_____

Firma del Tutor

Msc. Omar Mar Cornelio

Dedicatoria

A mi hija que le sirva este trabajo de diploma para un futuro y sea una excelente profesional.

A mi mamá y mi papá, que siempre han estado en mi vida, por tanto amor y entrega, por ser ejemplo de sacrificio, por permitirme crecer… especialmente a mi mamá que me ha ayudado a avanzar desde mis primeros pasos, y ser en todo momento mi guía y mi bastón.

A mi hermana que ha sido la raíz de mis proyecciones como profesional y convertirse para mí en un ejemplo a seguir… a mi esposa por tanta dedicación y espera.

A la memoria de mi abuela Ramona que aunque hoy no esté aquí, ha sido y seguirá siendo el motor de mi vivir… he podido concluir uno de sus sueños.

Agradecimientos

En primer lugar agradecer eternamente a toda mi familia… en especial a mi abuela madre Ramona, que hoy a 1 año 3 meses y 9 días de no estar con nosotros, ha sido y seguirá siendo mi ejemplo de sacrificio, dedicación, humildad y valentía, a pesar de que hoy no esté aquí sentada viendo cumplir uno de sus sueños.

A mi mamá que siempre ha estado presente para guiarme, apoyarme y darme consejos…por darme ese beso en la frente… ser mi ejemplo como madre, persona y profesional, ser mi bastón, mi guía, mi todo.

A mi papá que aunque hemos podido compartir pocos momentos y ser una persona de carácter fuerte, nos ha enseñado que las cosas hay que ganárselas y sacrificarse en todo momento.

A mi hermana Irina que me ha ayudado, a pesar de los cocotazos que siempre nos dábamos, ha sido mi ejemplo a seguir como profesional y como persona, y siempre tenerme en sus oraciones, al igual que mi abuela María.

A mis tíos y tías, primos y primas, por confiar en mí y contar con su apoyo… en especial a mi tía Esperanza que siempre ha estado conmigo en las situaciones buenas y malas, por su dedicación, esmero y cariño.

A mi padrino Nene y familia, por todo ese amor incansable que tenemos hacia él, y ellos hacia nosotros.

A la Yaire por ser mi compañera eterna, mi apoyo y por soportarme todo este tiempo… por darme esa niña preciosa – Carolina - que es una adoración en persona, que cuando crezca sirva de apoyo la presente investigación en sus proyecciones futuras.

A mis hermanos Glenda y Manuel A. porque a pesar del poco tiempo que los conozco, siempre me han tenido presente.

A Osvel por su consultica de Hibernate y Andry por su apoyo transparente en Liferay y Spring, creo que sin su guía aún no hubiese avanzado.

A mis primeros amigos de la universidad: Damián, Glauver, Javier y Yeriel, que aunque el futuro nos aisló siempre pude contar con su brazo amigo, su apoyo incondicional, sirva para ellos este trabajo de diploma.

A Julián por ser compañero, Marlen esa tía de chocolate y Yuneiry profe y amigo incondicional…

A mi madre Ima, mis hermanas Glennis y Jessie, mi tía Rosy y amiga Qu, que fueron un conjunto familiar UCI de cumpleaños y emociones, en especial a la madre Ima, que ha sido mi empuje y siempre preocuparse por mí.

A mis compañeros de clases: a aquellos que hoy no están, los que aún no han concluido la meta y los que ya llegaron; por su compañerismo, amistad, sinceridad, apoyo y esfuerzo; en especial a Bernardo y Dayana ´pru´ que se han convertido en mi familia

AGRADECIMIENTOS

UCI por todo lo que han significado para mí en estos 3 años que los conozco; a Randy y Lisandra ´duducin´ ("mis supuestos dúos de tesis") y amigos de los que dicen la verdad aunque duela; a mis compañeros de apto (Jorgito como un hermano, Emilio, Ernesto, Fabiel, Alex, Alejandro, Yader, entre otros) creo que escucharme fue demasiado; a las chicas (Gretel, Tahimi 'pumbita', Laura, Jisel ´dudi´, Ibis, Aylin, entre otras) con las que compartí buenos momentos; a mi grupo de la FEU que tuvimos noches de reuniones, buenos y malos momentos y que hoy son parte de ese círculo de amigos (Lorenzo quien me tiro la soga, Enier, Lionel y Osviel fieles compañeros, Jairo mi consejero y colega, Lijandy, Yanitza, Alfredo, Rolando, Ana M., Luisma, Dayron, Yaisel, Dieter del sitio gladiadores, entre otros tantos).

A todo el colectivo de profesores que de cada uno tomé lo mejor de ellos, por formar parte de mi vocación profesional, por su apoyo en cada momento, dedicación y esmero a la hora de enseñar (Yaikiel, Dianet, Yurima, Trujillo, Liesner, Yanelis Benitez, Heydi, Nara, Yordanys, Omar, Aliosmi, Leonardo Castillo, Jeem, Asdrubal, entre otros).

A mis tutores, Reisel el salvavidas de 30min y en especial a Omar y familia, por soportar mis dudas, mis malcriadeces y ser tan valiente al tutorar la tesis; a mi oponente Félix, por ser quisquilloso y exigente; al tribunal por su ayuda, en especial a Vilmavis y soportarme los mediodías.

A los compañeros de los años anteriores de mi facultad ("feos" y "feas" como en ocasiones les digo), por el simple hecho de saludar y preguntar; a los de fiestas: Asiel, Ramón, etc.

A mis amigos de las facultades 1, 2, 3, 4, 5 y 7 que son muchos pero aportaron en mi día a día en estos 5 años de convivencia en la universidad.

A mis estudiantes y profesores de la secundaria básica que formaron parte de mi vocación como profe y permitirles enseñar un granito de arena.

A las tías del docente, del comedor y la residencia;

A mis amistades, vecinos y profes de Santiago de Cuba, por su preocupación hacia mí a pesar de la lejanía…

Un agradecimiento eterno a nuestro Comandante en Jefe y la Revolución por permitirnos desde lejos venir a estudiar, dejarnos crecer como personas integrales y dar hoy un "clic" al mundo de la informática.

A todo, muchas gracias!

Resumen

El empleo de medios informáticos constituye un eslabón fundamental en el Proceso de Enseñanza Aprendizaje. El uso de plataformas de Sistemas de Laboratorios Virtuales y a Distancia aporta facilidades en la adquisición de los conocimientos por parte de los estudiantes y les permite crear sus propios espacios para ejercitar, comprender y afianzar los contenidos. Con el estudio realizado se constató que es importante definir los indicadores que permitan evaluar las competencias de los estudiantes en la ejecución de las prácticas de laboratorios y las relaciones existentes entre estos. Además de establecer un mecanismo para la evaluación de competencias para determinar si el estudiante está apto o no para realizar la ejecución de los experimentos. Con esto se evita, estudiantes que no tengan un dominio de las competencias necesarias y no provoquen un mal funcionamiento de las estaciones de trabajo. Con tal de solucionar estas dificultades se desarrolla un módulo informático para la evaluación de competencias en la Plataforma de Laboratorio Virtual y a Distancia que contribuya a la toma de decisiones en la ejecución de prácticas de laboratorios. La confección de este módulo estuvo guiada por la metodología de desarrollo OpenUP, el lenguaje de programación Java, entre otras herramientas y tecnologías necesarias. Como resultado de la investigación se logró implementar un módulo para la evaluación de competencias e integrarlo en la Plataforma de Laboratorio Virtual y a Distancia para las ejecuciones de prácticas de laboratorios.

Palabras claves: Competencias, evaluación, sistema de laboratorios virtuales y a distancia.

Abstract

The use of information technology is a key link in the Teaching Learning Process. The using Platforms Virtual Systems Laboratories and Distance provides facilities for the acquisition of knowledge by students and allows them to create their own spaces for exercise, understand and secure the contents. The study found that it is important to define the indicators to assess the skills of students in the implementation of laboratory practice and the relationships between them. In addition to establishing a mechanism for evaluating competence to determine whether the student is eligible or not for the execution of the experiments. With this, students who do not have a mastery of the necessary skills and do not cause a malfunction of workstations is avoided. As long as resolve these difficulties a software module for the evaluation of skills developed in the Virtual Laboratory and Distance Platform to contribute to decision making in implementing laboratory practices. The preparation of this module was guided by the development methodology OpenUP, the Java programming language, including necessary tools and technologies. As a result of the investigation it was possible to implement a module for evaluating skills and integrate it into the Virtual Laboratory and Distance Platform executions laboratory practice.

Key words: Competence, evaluation, virtual systems laboratories and distance.

Índice

Introducción

La educación es un componente de la sociedad, a esta se asocian las nuevas Tecnologías de la Información y las Comunicaciones (TIC) que han provocado un impacto en el desarrollo y formación del hombre. El empleo de los medios informáticos constituye un eslabón fundamental en el Proceso de Enseñanza Aprendizaje (PEA), ya que permiten crear condiciones materiales favorables para cumplir con las exigencias científicas del mundo contemporáneo y ajustar los contenidos de cada materia de estudio, de manera que respondan a los avances tecnológicos que se suscitan a escala mundial.

La instrucción en torno a la práctica preocupa a los docentes de todas partes del mundo; ya que se ha considerado necesario complementar la enseñanza teórica con la ejecución de experimentos. Las prácticas de laboratorio en computadoras permiten manipular materiales, instrumentos e ideas, así como el desarrollo de habilidades creativas y de asimilación. De este modo, las TIC son utilizadas para aprender y para enseñar, y con su empleo se mejora el PEA.

En nuestros días, gracias a los nuevos adelantos tecnológicos, la educación ha alcanzado altos niveles de desarrollo, un ejemplo de ello lo demuestra el uso de plataformas de Aprendizaje Electrónico (del inglés, *E-learning)* que implica una nueva forma de aprender con el uso de Internet. Este sistema ha transformado el PEA, abriendo puertas al aprendizaje individual y organizacional. Por su parte, este nuevo concepto educativo es una revolucionaria modalidad de preparación que posibilita Internet y que hoy se posiciona como la forma de capacitación predominante en el futuro.

Dentro de este marco, como parte del *E-learning*, juegan un papel fundamental los Sistemas de Laboratorios Virtuales y a Distancia (SLVD), los cuales se pueden definir como: (…) *herramienta que utiliza una red de comunicaciones, donde los usuarios y los equipos del laboratorio están separados geográficamente y las tecnologías de las comunicaciones se utilizan para acceder a estos equipos* (Khamis 2006). Estos aportan facilidades en la adquisición de los conocimientos por parte de los estudiantes y les permite crear sus propios espacios para ejercitar, comprender y afianzar los contenidos, sin estar obligados a trasladarse a un centro de estudios.

Los sistemas de educación por medio de la computadora representan mayor flexibilidad, accesibilidad y adaptabilidad que los sistemas de educación convencionales incrementando las exigencias de calidad en

la formación. En este sentido se han ido perfeccionando en correspondencia con los requerimientos didácticos que demanda el PEA, los productos informáticos que posibilitan la realización de prácticas de laboratorio de manera virtual, buscando establecer una mayor interactividad con el usuario, además de lograr un ambiente virtual más cercano a la realidad e incluir orientaciones para el estudio.

Sin embargo, a partir del estudio realizado con la utilización de métodos empíricos tales como entrevistas y encuestas (Ver Anexo 1 y 2) acerca del uso de los SLVD, se pudo constatar que:

- No están definidos los indicadores que permitan evaluar las competencias de los estudiantes en la ejecución de las prácticas de laboratorios ni la relación existentes entre estos.
- Se carece de mecanismos para la evaluación de competencias de los estudiantes practicantes para determinar si están aptos o no para realizar la ejecución de un experimento determinado.
- No existe un sistema de comunicación automatizado que de manera individual ofrezca al estudiante la valoración de las competencias evaluadas.

Estas deficiencias traen como consecuencias que no sea posible predecir los problemas de los estudiantes en el trabajo con las prácticas realizas en el SLVD. Por lo que es posible que accedan a la plataforma estudiantes que no posean las competencias necesarias y puedan generar un mal funcionamiento en las estaciones de trabajo.

A partir de la situación descrita, surge el siguiente **problema de la investigación**: ¿Cómo incorporar la evaluación de competencias en un Sistema de Laboratorios Virtuales y a Distancia que contribuya a la toma de decisiones para la ejecución de prácticas de laboratorio? En correspondencia con el problema planteado, el **objeto de estudio** de la presente investigación es: Sistemas de Laboratorios Virtuales y a Distancia, enmarcado en el **campo de acción**: Evaluación de competencias en un Sistema de Laboratorios Virtuales y a Distancia.

Con el propósito de solucionar el problema planteado se define como **objetivo general** de la investigación: Desarrollar un módulo para la evaluación de competencias en la Plataforma de Laboratorio Virtual y a Distancia que contribuya a la toma de decisiones en la ejecución de prácticas de laboratorios.

Para el cumplimiento del objetivo general se plantean las siguientes **preguntas de investigación**:

1. ¿Cuáles son los fundamentos teóricos del proceso de evaluación de competencias asociadas a la ejecución de prácticas en el SLVD?
2. ¿Qué metodología, tecnologías y herramientas de desarrollo se podrían utilizar en la implementación de la propuesta de solución?
3. ¿Qué técnicas de pruebas se pueden utilizar en la validación de la propuesta de solución y cómo se deben aplicar?

Definido el objetivo general se desglosan las siguientes **tareas de la investigación:**

1. Análisis de los referentes bibliográficos de las plataformas existentes para la construcción del marco teórico de la investigación.
2. Identificación de los métodos, herramientas y tecnologías existentes para su selección de acuerdo a las necesidades del módulo sobre plataformas web.
3. Identificación de las funcionalidades de la arquitectura del módulo para la evaluación de competencias en un SLVD.
4. Elaboración de los artefactos y modelos asociados a la metodología de desarrollo de *software* escogida.
5. Implementación de los requisitos identificados luego de aplicada la ingeniería de requisitos, el análisis y diseño para el desarrollo del módulo.
6. Realización de las pruebas de *software* para comprobar el correcto funcionamiento del módulo propuesto.

Para el desarrollo de la investigación se utilizaron diferentes métodos científicos agrupados en **métodos teóricos y empíricos**, los cuales tienen su sustento en la concepción materialista dialéctica y permiten una mejor recopilación de la información para el modelado de análisis y diseño del módulo a realizar. Se utiliza de los métodos teóricos: analítico – sintético, y de los métodos empíricos: análisis de documentos, la entrevista y la encuesta.

El método **analítico - sintético** facilita el análisis de la bibliografía utilizada, además de garantizar la fiabilidad de la información empleada. Permite obtener, describir y resumir los elementos más importantes que intervienen en los procesos de análisis y gestión de los SLVD para construir el modelo de negocio y realizar el diseño del módulo adecuadamente. De los métodos empíricos, se utiliza el **análisis de**

documentos para definir los diferentes requerimientos del sistema y recopilar información referente a las características y funcionamiento de los SLVD.

Se utiliza la **entrevista** (Anexo 1: Entrevista a Jefes de Departamentos y Jefes de Asignaturas) en este caso a Jefes de Departamentos y Jefes de Asignaturas de la Universidad Central de las Villas "Martha Abreu" (UCLV) para conocer el estado actual sobre la utilización de los SLVD en el PEA y cómo se mide la evaluación de competencias en estos SLVD para la toma de decisiones. También se emplea la **encuesta** exploratoria (Anexo 2: Encuesta a profesores y metodólogos) para obtener información sobre el estado actual en cuanto a la utilización de los SLVD en el PEA en la UCLV a partir de un grupo de personas: profesores y metodólogos de la UCLV.

Como **posibles resultados** se espera la obtención de un módulo integrado a la Plataforma de SLVD para la evaluación de competencias, que contribuya a la toma de decisiones en la ejecución de prácticas de laboratorios.

Capítulo 1: Fundamentos teóricos para la evaluación de competencias en la Plataforma de SLVD

Introducción

En el presente capítulo se describen los conceptos básicos que se relacionan con el objeto de estudio y campo de acción de la investigación. Se caracterizan los diferentes Sistemas de Laboratorios Virtuales y a Distancia (SLVD); además de la evaluación de competencias en el uso de los SLVD para la toma de decisiones, y se fundamenta la selección de la metodología, tecnologías y herramientas a utilizar para el desarrollo del módulo.

1.1 Sistemas de Laboratorios Virtuales

A partir del estudio realizado en las bibliografías consultadas y para un mejor entendimiento de la definición de laboratorios virtuales; se exponen a continuación los principales conceptos planteados por diferentes autores sobre esta temática.

Una de las definiciones refieren que un laboratorio virtual (LV) es un sistema computacional que pretende aproximar el ambiente de un laboratorio tradicional (LT) (Rosado 2009). Por otra parte (Benavides and Morales 2009) se plantea que: *(…) un LV es una herramienta de tipo multimedia e interactiva que sirve para mejorar y complementar el PEA (…)*, lo cual presenta como principales características:

- ✓ Tiene una interfaz de usuario intuitiva y fácil de utilizar.
- ✓ Utiliza instrumentación simulada interactiva que posee una funcionalidad similar a la de los instrumentos reales.
- ✓ Relaciona los conceptos prácticos con los teóricos mediante un conjunto de experimentos adecuadamente diseñados.

Después de las consultas realizadas en la presente investigación se asume como definición de laboratorio virtual: a un entorno simulador de prácticas de laboratorio virtual tradicional donde a través de una computadora personal (PC, del inglés, *Personal Computer*) se pueda interactuar con materiales u objetos, permitiendo al usuario realizar investigaciones, experimentos o prácticas; siendo éste una herramienta significativa para el mejoramiento del PEA.

En la actualidad los LV se han aplicado a la enseñanza a distancia, por lo que se puede definir cómo SLVD a una simulación en computadora de una amplia variedad de situaciones, desde prácticas manipulables hasta visitas guiadas, en un ambiente interactivo sin limitación de lugar u horario para aprender. Partiendo de las ideas anteriores, se mencionan las características y el funcionamiento, además de las ventajas y desventajas que tienen en común los SLVD.

1.1.1 Características y funcionamiento de los SLVD

Los SLVD presentan características comunes expuestas por Sartorius (Sartorius 2005) entre las que destacan:

- ✓ Disponibilidad*:* El sistema está disponible las 24 horas del día con su adecuada protección.
- ✓ Accesibilidad: El SLVD puede ser accedido desde cualquier parte del mundo.
- ✓ Facilidad de uso: Para usar el sistema solo se debe tener los conocimientos básicos de la disciplina de objeto de prácticas.
- ✓ Interfaz de usuario fácil y rápida: La interfaz de usuario del SLVD está basada en páginas HTML, esto permite que los usuarios puedan acceder al sistema de una forma rápida y sin necesidad de descargar o instalar ningún software adicional.
- ✓ Administración de múltiples pedidos en forma paralela: Permite atender múltiples pedidos de forma paralela administrando de forma centralizada dispositivos similares que se encuentren geográficamente separados pero unidos por redes de área extensa (del inglés, *WAN*).
- ✓ Desarrollo de controladores de forma remota usando Matlab y Simulink: Permite a los usuarios diseñar sus propios controladores utilizando Matlab/Simulink.

Funcionamiento de los SLVD

A través de la Internet los usuarios interactúan con los SLVD. Al acceder al sitio web el usuario se identifica con su cuenta, elige la práctica que desea realizar, completa correctamente todos los datos en el formulario asociado a la práctica y finalmente ejecuta la misma.

Los datos de las prácticas son recibidos por el Servidor de Administración de Prácticas, el cual se encarga de enviarlo al Cliente de Administración de Prácticas de una estación que pueda ejecutarla y se encuentre

disponible, en caso de todas ocupadas elige la que menor cola de prácticas por atender tenga. El estado de las estaciones y las características de prácticas son almacenados en una base de datos.

1.1.2 Ventajas y desventajas de los SLVD

A continuación, se destacan algunas ventajas importantes de los SLVD (Nájera 2007b):

- ✓ Acerca y facilita a un mayor número de alumnos para la realización de experiencias, aunque alumno y el laboratorio no coincidan en el espacio. El estudiante accede a los equipos del laboratorio a través de un navegador, pudiendo experimentar sin riesgo alguno; además, se flexibiliza el horario de prácticas.
- ✓ Reducen el costo del montaje y mantenimiento de los LT, siendo los LV una alternativa poco costosa y eficiente, donde el estudiante simula los fenómenos a estudiar como si los observara en el LT.
- ✓ Es una herramienta de autoaprendizaje, donde el alumno altera las variables de entrada, configura nuevos experimentos, aprende el manejo de instrumentos, personaliza el experimento, etc.
- ✓ La simulación en el LV, permite obtener una visión más intuitiva de aquellos fenómenos que en su realización manual no aportan suficiente claridad gráfica. El uso de LV da lugar a cambios fundamentales en el proceso habitual de enseñanza, en el que se suele comenzar por el modelo matemático.
- ✓ Los estudiantes aprenden mediante prueba y error, sin miedo a sufrir o provocar un accidente, sin avergonzarse de realizar varias veces la misma práctica ya que pueden repetirlas sin límites; sin temor a dañar alguna herramienta o equipo.
- ✓ Pueden asistir al laboratorio cuando deseen y elegir las áreas del laboratorio más significativas para realizar prácticas sobre su trabajo.

En los SLVD, también existen inconvenientes o desventajas (Nájera 2007b). Algunas de estas son mencionadas por Nájera:

- ✓ El LV no puede sustituir la experiencia práctica altamente enriquecedora del LT. Ha de ser una herramienta complementaria para formar a la persona y obtener un mayor rendimiento.
- ✓ En el LV se corre el riesgo de que el alumno se comporte como un mero espectador. Es importante que las actividades en el LV, vengan acompañadas de un guion que explique el concepto a estudiar,

así como las ecuaciones del modelo utilizado. Es necesario que el estudiante realice una actividad ordenada y progresiva, adecuada a alcanzar objetivos básicos concretos.

- ✓ El alumno no utiliza elementos reales en el LV, lo que provoca una pérdida parcial de la visión de la realidad. Además, no siempre se dispone de la simulación adecuada para el tema que el profesor desea trabajar.
- ✓ Son pocas las experiencias realizadas con LV en los centros educativos, donde aún impera el uso de recursos tradicionales, tanto en la exposición de conocimientos en el aula como en el laboratorio.

Resulta interesante definir las competencias en el uso de los SLVD así como sus ventajas.

1.2 Competencia en el uso de los SLVD

La introducción de SLVD tiene como fin propiciar ambientes de aprendizaje que promuevan el desarrollo de competencias genéricas y disciplinares mediante el desarrollo de un simulador multimedia educativo y realizando prácticas sin necesidad de adquirir equipo y materiales costosos, peligrosos o difíciles de conseguir o almacenar, donde se manipulen los mismos elementos que en una práctica experimental real, obteniendo los mismos resultados (Arroyo and Pedraza 2011).

Como ambientes de aprendizaje, los SLVD permiten que los alumnos pongan en práctica sus habilidades y conocimientos, motivándolos para desarrollar competencias que involucran la resolución de problemas; desempeños que serán requeridos por ellos en el futuro para adquirir nuevos conocimientos.

Según (Rubio 2010) define el término de competencia: *"(…) es un conjunto de conocimientos que al ser utilizados mediante habilidades de pensamiento en distintas situaciones, generan diferentes destrezas en la resolución de los problemas de la vida y su transformación, bajo un código de valores previamente aceptados que muestra una actitud concreta frente al desempeño realizado, es una capacidad de hacer algo (…)".*

Al respecto (Santos 2001) plantea, las competencias son: *"(…) características subyacentes a la persona, que están causalmente relacionadas con una actuación exitosa en un puesto de trabajo (…)".*

En los últimos años ha ido alcanzando gran importancia en diferentes sectores profesionales, el valor aplicado al término "competencia". Existen muchos tipos de competencias, pero se centran en desarrollar

el conjunto de habilidades, características y actitudes que conforma un individuo al desempeñar un puesto de trabajo.

Competencias conceptuales: Comprender, conocer, analizar, comparar y evaluar teorías, tendencias y metodologías generales relacionadas con el trabajo, sus características, para que ayuden a aprender las destrezas pertinentes y afrontar así los problemas específicos que forman las peculiaridades del puesto de trabajo. Modalidades e instrumentos generales de evaluación para el proceso de aprendizaje del trabajo. Dominar y valorar técnicas creativas y dinámicas de presentación y dinamización del puesto de trabajo deseado. También la ayuda de las nuevas tecnologías de la información y el conocimiento, propician a una mejor organización y planificación personal (Maldonado 2010).

Competencias procedimentales: Ser capaz de utilizar las teorías y métodos de trabajo, nombrados anteriormente, la habilidad de emplear los procedimientos adecuados a diferentes proyectos, realizándolo con una actitud creativa y dinámica el trabajo a desempeñar, integrando lo aprendido sobre métodos y teorías en la reflexión crítica derivada de la observación de los procesos de trabajo. Ser capaz de elaborar tomando como base lo aprendido, con actitud crítica y responsable; sus reflexiones críticas y sus tareas docentes sirvan para la autoevaluación y la autocorrección (Maldonado 2010).

A continuación se presentan algunas de las ventajas de las competencias al hacer uso de los SLVD (Nájera 2007b).

- ✓ Facilita el uso de un lenguaje común a la hora de medir comportamientos observables.
- ✓ Focalizar los esfuerzos de todas las personas hacia los resultados, ya que se pueden programar los sistemas de evaluación del personal de forma que se analizan los puntos débiles y fuertes para diseñar las acciones más adecuadas y puedan mejorar los resultados.
- ✓ Se utiliza como previsor del comportamiento futuro de la persona dentro del SLVD. Cuando una persona ha sido capaz de llevar a cabo un determinado comportamiento; en unas condiciones dadas, se puede esperar que sea capaz de repetir el mismo comportamiento en condiciones similares.
- ✓ El enfoque de competencia facilita la comparación entre el perfil de exigencias del puesto y el perfil de competencia de las personas.

Partiendo de los supuestos anteriores, se determinó qué y cuáles son esas características subyacentes relacionadas con la actuación exitosa de la persona en un puesto de trabajo, a lo que anteriormente se

llamó como competencias. Pero, ¿cómo medir si esa persona es competente o no haciendo uso de los SLVD? Por estas razones se ejemplifica el término de evaluación de competencias.

1.2.1 Evaluación de competencias en el uso de los SLVD

La evaluación es el proceso para comprobar y valorar el cumplimiento de los objetivos propuestos y la dirección dialéctica de la enseñanza y el aprendizaje en sus momentos de orientación y ejecución (Ruiz and Colunga 2010). El tema de la evaluación de competencias es de carácter multiforme, debido a la variedad de enfoques y de concepciones sobre la competencia.

La evaluación basada en las competencias es una modalidad de evaluación que se deriva de la especificación de un conjunto de resultados, que determina los resultados generales y específicos con una claridad tal que los evaluadores, los estudiantes y los terceros interesados pueden juzgar con un grado razonable de objetividad si se han alcanzado o no y que certifica los progresos del evaluado en función del grado en que se han alcanzado objetivamente esos resultados. Las evaluaciones no dependen del tiempo de permanencia en instituciones educativas formales.

Tobón, Rial, Carretero y García conciben que la evaluación basada desde el enfoque de competencias como: *"(...) se orienta a evaluar las competencias en los estudiantes teniendo como referencia el proceso de desempeño de estos ante actividades y problemas del contexto profesional, social, disciplinar e investigativo, teniendo como referencia evidencias e indicadores, buscando determinar el grado de desarrollo de tales competencias en sus tres dimensiones (afectivo - motivacional, cognoscitiva y actuacional) para brindar retroalimentación en torno a fortalezas y aspectos a mejorar (...)"* (Tobón 2009).

Por otra parte se define como evaluación de competencias según Álvarez como: *"(...) proceso que tiene como fin determinar si una persona es "competente" o "aún no es competente" para realizar una función productiva determinada de acuerdo a una metodología predefinida que incluye distintas etapas de recopilación de información sobre el desempeño del evaluado (…)"* (Álvarez 2009).

Al respecto, la evaluación basada en competencias se caracteriza por ser (Iglesias 2010):

- ✓ Continua: Implica la evaluación para aprender. La evaluación ha de ser por ella misma una experiencia de aprendizaje y un acto de fortalecimiento.

- ✓ Sistemática: Que el proceso de planificación sea estandarizado, que comprenda tareas diversas, aunque íntimamente ligadas entre sí, que demuestre una consistencia interna debidamente secuenciada.
- ✓ Basada en evidencias: Se asume la evidencia como una aportación que debe hacer un individuo en función de un criterio de verdad, a lo que se pudiera añadir más específicamente, que busca la manifestación de una cosa, de manera que no se dude de ella. Por ello al final, los indicadores, los criterios y evidencias son incluidos dentro de la categoría de evidencias.

Se propone a continuación el modelo para la evaluación de competencias en el uso de un SLVD.

1.2.2 Modelo para la evaluación de competencias en el uso de un SLVD

Para la presente investigación al formarse parte de la Plataforma de SLVD y tenerse en cuenta los requisitos definidos del negocio, se asume como modelo para la evaluación de competencias el planteado por (Mar 2014) ya que se ajusta a las condiciones de la misma. El modelo para la evaluación de competencias describe el proceso de inferencia y está orientado a soportar la toma de decisiones sobre evaluaciones de competencias para un SLVD. Se basa en tres etapas básicas: entrada, procesamiento y salida de información. La Fig. 1 representa un esquema general de la propuesta.

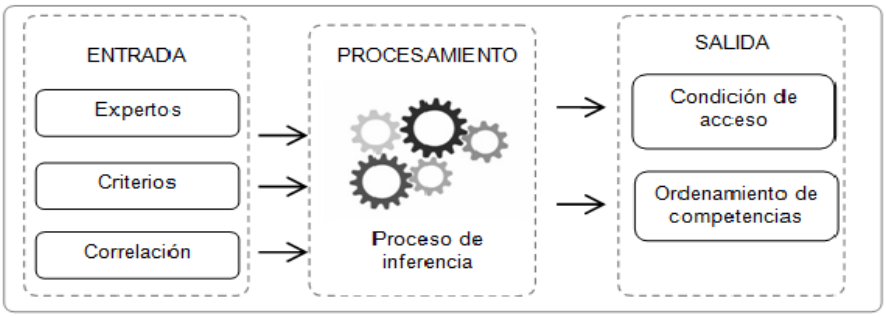


Fig. 1 Esquema general del modelo propuesto por Mar

Descripción de las etapas del modelo

Entrada de información: Proceso mediante el cual el modelo toma los datos que requiere para ser procesado (Santos 2013). En la propuesta, existen datos gestionados manualmente que son aquellos que se proporcionan de forma directa por el usuario como es la gestión expertos, indicadores y relaciones causales.

Procesamiento de información: Capacidad del modelo para efectuar cálculos de acuerdo con una secuencia de operaciones preestablecida, permite la transformación de datos de baja interpretabilidad en información organizada. El proceso de inferencia y organización de la información es guiado mediante la utilización de mapas cognitivos difusos.

Los mapas cognitivos difusos (MCD) son modelos difusos con retroalimentación para representar causalidad. Combinan herramientas teóricas de los mapas cognitivos, la lógica difusa, las redes neuronales, las redes semánticas, los sistemas expertos y los sistemas dinámicos no lineales (Glykas and Groumpos 2010) y (Lin and Lee 2002).

Esta técnica permite modelar el sistema con retroalimentación con grados difusos de causalidad en el intervalo [0,1], donde cada nodo representa un conjunto difuso o evento que ocurre en algún grado. Los nodos son conceptos causales y pueden modelar eventos, acciones, valores, metas o procesos. Con la utilización de esta técnica se obtienen además los beneficios de modelado visual, la simulación y la predicción (Salmeron 2010).

Un MCD puede ser representado a través de un dígrafo en el cual los nodos representan conceptos y los arcos indican relación causal. La intensidad de la relación causal es representada mediante valores difusos (Peña et al. 2007). Los valores de los conceptos son calculados en cada paso de la simulación. De acuerdo al vector inicial del MCD convergerá a un punto fijo, ciclo límite o atractor caótico.

Los MCD pueden ser representados mediante una matriz de adyacencia la cual es obtenida a partir de los valores asignados a los arcos. Esta puede ser escrita como:

$$E = \begin{bmatrix} \cdots & \cdots & \cdots \\ \cdots & w_{ij} & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix}$$

Cuando participa un conjunto de expertos (k), la matriz de adyacencia se formula mediante la ecuación.

$$E = \frac{1}{K}\left(E_1 + E_2 + \ldots + E_k\right)$$

Esta agregación de conocimiento permite mejorar la fiabilidad del modelo final, el cual es menos susceptible a creencias potencialmente erróneas de un único experto (Stach et al. 2010). Sin embargo la media aritmética es muy sensible a la presencia de valores atípicos.

Un aspecto a tener en cuenta son los errores que cometen los expertos para determinar el signo que acompaña a la relación de causalidad (Giordano and Vurro 2010), al utilizar la media aritmética además se anula la magnitud del peso. Se han propuesto métodos que tratan de minimizar el error los cuales requieren que se llegue a cierto consenso o la interacción posterior con el experto (Glykas et al. 2010) lo cual, aunque deseable, no siempre es posible.

Salida de información: La salida es la capacidad del modelo para representar los datos procesados y decidir alternativas (Sánchez and Valdés 2010). Para el modelo propuesto las informaciones fundamentales son la condición de acceso para la ejecución de las prácticas de control así como el ordenamiento de las competencias personales.

Durante el período de investigación se realizó la búsqueda de soluciones similares existentes en Cuba y el mundo, con el objetivo de encontrar apoyo para el desarrollo del trabajo. En este caso se considera necesario analizar las soluciones existentes de SLVD para poder establecer comparaciones entre estas y ver si se ajustan a las necesidades del proceso actual.

1.3 Análisis de soluciones existentes de SLVD en Cuba y en el mundo

Los LV comenzaron a desarrollarse en 1997 en el Centro de Investigación Académica de la Universidad Estatal a Distancia de Costa Rica. Si se juzga con base en la información disponible en Internet, fueron de los primeros laboratorios virtuales para enseñanza a distancia a nivel mundial (Nájera 2007a). Partiendo de lo supuesto anteriormente, se proponen las siguientes soluciones existentes:

1.3.1 VLabQ

VLabQ: Laboratorio Virtual de Química es un simulador interactivo para prácticas de laboratorio de Química, creado por *Sibees Soft* que utiliza equipos y procedimientos estándares para simular los procesos que intervienen en un experimento o práctica (Albarran 2010).

Este laboratorio presenta entre sus características más importantes su facilidad de manejo, ya que los elementos necesarios para llevar a cabo las prácticas se encuentran distribuidos en una barra de herramientas, además el programa explica paso a paso los procedimientos que deben llevarse a cabo para el éxito de cada práctica. El mismo contiene los instrumentos necesarios al igual que un laboratorio real, a la par cuenta con instrumentos para la medición y permite cambiar la velocidad de simulación, aunque será el diseñador de las prácticas el que determine si el usuario puede variar o no la velocidad de la simulación. Se destacan además: la posibilidad de guardar en cualquier momento todo el contenido del laboratorio, tanto el equipo como su contenido y condiciones, para así poder continuar con la práctica posteriormente. Consta de tres apartados que muestran el Marco teórico, el procedimiento y las conclusiones que contiene cada simulación.

VLabQ no solamente se limita a simular reacciones químicas, sino que también las lleva a cabo en tiempo real, este aspecto de *VLabQ* tiene como objetivo concienciar al estudiante que, para el estudio y las investigaciones de la química, se debe tener la paciencia necesaria para poder observar los fenómenos específicos y así obtener el producto final que se desea. Es un programa de gran ayuda para realizar prácticas que impliquen gran riesgo, además muestra las conclusiones y procedimientos que se han realizado. *VLabQ* posee varias características que favorecen el aprendizaje de los estudiantes.

1.3.2 Laboratorios virtuales en la Universidad virtual del CITMA

La Universidad Virtual del Ministerio de Ciencia Tecnología y Medio Ambiente (CITMA) es un sitio destinado a la educación a distancia, sobre diferentes temáticas, ofrece cursos gratis y pagados. Este servicio distintivo pertenece a la Empresa de Tecnologías de la Información y Servicios Telemáticos Avanzados (CITMATEL), el cual ofrece una amplia gama de títulos educacionales, además de preparar y promocionar diplomados, talleres especializados o de adiestramiento, eventos y otras modalidades educacionales, con la rigurosa supervisión de especialistas y la certificación de reconocidas instituciones académicas (Nodarse et al. 2008).

Son además LV, resultado del proyecto de I+D (Investigación más desarrollo) de alcance nacional, donde se integran también aulas, bibliotecas, eventos y museos virtuales, así como ambientes de trabajo colaborativo, para propiciar la generación, experimentación y descubrimiento de conocimientos. Los temas desarrollados abarcan Probabilidades y Combinatoria, Estadística Descriptiva y no Paramétrica; Cálculo

Diferencial Integral; Matemática Numérica, Mecánica, Óptica, Electricidad y Magnetismo. Las experiencias realizadas en temas de Física y Matemáticas han sido de hincapié para extenderse a otras áreas.

La simulación se usa preferentemente cuando el experimento real no es recomendable por razones de tiempo, peligrosidad, costo, o como preparación para el laboratorio, se orienta en dos direcciones: al comportamiento y al modelo. La simulación de comportamientos, donde el modelo está presente en el sistema y es conocido por el alumno, es el más extendido y empleado en el caso.

Las experiencias en la garantía y gestión de la calidad de los cursos sobre la web les permite identificar los elementos básicos a evaluar: propósitos del programa, presentación de los contenidos, contenido, recursos de navegación, uso de recursos multimedia, el uso de herramientas para el trabajo colaborativo y registro de la actividad del estudiante.

1.3.3 Sistema de Laboratorios a Distancia en Asignaturas de Regulación Automática

El Departamento de Automática y Sistemas Computacionales de la Universidad Central Marta Abreu de Las Villas (UCLV) en cooperación con el Departamento de Automática, Ingeniería Electrónica e Informática Industrial de la Universidad Politécnica de Madrid desarrollaron un SLD, que permite el ensayo de algoritmos de control de forma remota vía Internet. Está basado en *Matlab/Simulink* y permite la realización de prácticas tanto simuladas como reales en un entorno web sin necesidad de descargar software adicional. Se pueden ejecutar prácticas paramétricas (controlador predefinido) o con cambio de estrategia (controlador definido por el usuario) (Sartorius 2005).

El sistema permite la realización de actividades prácticas a distancia por los estudiantes en su tiempo de estudio independiente, con el objetivo fundamental de ampliar los conocimientos y realizar más ensayos a los previstos por los trabajos prácticos de la asignatura. Por otra parte, se demuestran las posibilidades de reutilización que presenta el SLVD al ser fácilmente implantado en otro laboratorio semejante.

1.3.4 Comparaciones de las soluciones existentes

En la Tabla. 1 se refleja en resumen las características esenciales de las soluciones encontradas para realizar una comparación entre estas.

Tabla. 1 Comparación entre las soluciones existentes

No	*Nombre de las soluciones existentes*	*Laboratorio Virtual*	*Laboratorio Virtual y a Distancia*	*Facilidad de manejo*	*Software propietario*	*Evalúa competencias*
1	VLabQ	SI	SI	SI	SI	NO
2	Laboratorio virtuales en la Universidad virtual del CITMA	SI	SI	SI	SI	NO
3	Sistema de Laboratorios a Distancia en Asignaturas de Regulación Automática	SI	SI	SI	SI	NO

Al comparar las herramientas teniendo en cuenta indicadores como saber si son laboratorio virtual, laboratorio virtual y a distancia, la facilidad de manejo que presentan, si son herramientas privativas y si evalúan competencias a los estudiantes practicantes. Con estas evidencias se llega a la conclusión que las herramientas analizadas no satisfacen las necesidades del proceso actual, ya que no permiten establecer la evaluación de competencias para la ejecución de prácticas en laboratorios, objeto principal de la presente investigación. Otro elemento importante es que todas son herramientas propietarias, por lo que no permiten acceder a su código fuente y ajustarlas para cumplir con el objetivo propuesto de la investigación. De estas conclusiones se propone la implementación del modelo para la evaluación de competencias a partir de un módulo informático que permitirá ser integrado a la Plataforma de SLVD.

1.4 Metodología, herramientas y tecnologías para el desarrollo del módulo propuesto

En el proceso de desarrollo de *software* es fundamental la correcta elección de las tecnologías y herramientas que mejor se adapten y mayores facilidades brinden para dar solución al problema de la investigación.

Para el desarrollo del módulo a implementar se adopta la metodología, herramientas y tecnologías definidas por el grupo de arquitectura de la Plataforma de SLVD. Este ambiente se ajusta a la necesidad de migrar progresivamente hacia plataformas y aplicaciones de código abierto, en concordancia con el desarrollo del proceso de informatización de la sociedad cubana como parte de una política orientada a alcanzar la seguridad, invulnerabilidad e independencia tecnológica.

1.4.1 Metodología de desarrollo de software

Una metodología de desarrollo de *software* es un conjunto de procedimientos, técnicas, herramientas y soporte documental que ayudan a los desarrolladores a producir un nuevo *software* (Menéndez and Asensio 2013).

Dentro de las metodologías de desarrollo de software se encuentra la metodología Proceso Unificado Abierto (*OpenUP*, del inglés, *Open Unified Process*). Seleccionada por ser una metodología ágil, diseñada para un equipo de desarrollo reducido. Además por ser extensible, con iteraciones cortas y permitir mantener la filosofía de RUP (del inglés, *Rational Unified Process*).

La metodología *OpenUP* es un proceso unificado que aplica propuestas de gestión ágil; mantiene sus características esenciales: centrado en la arquitectura, dirigido por casos de uso y desarrollo iterativo e incremental dentro del ciclo de vida de un proyecto de software. No provee lineamientos para todos los elementos que se manejan en un proyecto, pero tiene los componentes básicos que pueden servir de base a procesos específicos. La mayoría de los elementos de *OpenUP* están declarados para fomentar el intercambio de información entre los equipos de desarrollo y mantener un entendimiento compartido del proyecto, sus objetivos, alcance y avances (Tabares and López 2013).

La metodología de desarrollo *OpenUP* está caracterizado por cuatro principios básicos interrelacionados (Balduino 2007):

- ✓ Colaboración para unificar intereses y compartir conocimientos.
- ✓ Equilibrio de prioridades competentes a maximizar el valor de los involucrados con el resultado del proyecto.
- ✓ Enfoque en la articulación de la arquitectura.
- ✓ Desarrollo continuo para obtener realimentación y realizar las mejoras respectivas.

Los elementos del *OpenUP* dirigen la organización del trabajo en los niveles personal, de equipo y de interesados. En cada iteración del ciclo de vida, incrementa progresivamente los objetivos de las iteraciones anteriores, añadiendo nuevas funcionalidades a las versiones estables del *software* que se tiene hasta el momento. Dentro del ciclo de vida existen cuatro fases: Inicio, Elaboración, Construcción y

Transición; que provee a los interesados un mecanismo de supervisión y dirección para controlar los fundamentos del proyecto, su ámbito, la exposición a los riesgos, el aumento de valor y otros aspectos.

Se puede destacar algunos de los beneficios del uso de la metodología seleccionada:

- ✓ Es una metodología ágil que tiene un enfoque centrado al cliente y con iteraciones cortas, por lo que es apropiado para proyectos pequeños y de bajos recursos.
- ✓ Permite disminuir las probabilidades de fracaso en los proyectos pequeños e incrementar las probabilidades de éxito.
- ✓ Permite detectar errores tempranos a través de un ciclo iterativo.

Por las características antes descritas y beneficios de *OpenUP* se considera que es la metodología adecuada para el desarrollo del módulo a implementar como solución de la investigación; además de analizarse como elemento para su selección, el poco tiempo que se dispone para el desarrollo del módulo y que se cuenta con un solo desarrollador. Por otra parte, se considera necesario describir las distintas herramientas y tecnologías estudiadas para el desarrollo del módulo propuesto.

1.4.2 Lenguaje de modelado de sistemas

Una exigencia de las instituciones es que los desarrollos de *software* se formalicen con un lenguaje estándar y unificado. Es decir, se requiere que cada una de las partes que comprende el desarrollo de todo software de diseño orientado a objetos, se visualice, especifique y documente con lenguaje común (Cornejo 2008). Un lenguaje unificado que cumple con estos requisitos, es ciertamente el Lenguaje Unificado de Modelado (UML, del inglés, *Unified Modeling Language*).

Lo fundamental de UML es la capacidad de diagramación y los diferentes tipos de diagramas que soporta (Cornejo 2009). UML 2.0 presenta como característica que es un lenguaje de modelado orientado a objetos y que contiene una viabilidad en la corrección de errores. Permite documentar todos los artefactos de un proceso de desarrollo: requisitos, arquitectura pruebas y versiones. Es un lenguaje muy expresivo que cubre todas las vistas necesarias para desarrollar y luego desplegar los sistemas.

De acuerdo a esto se decide utilizar UML 2.0 como lenguaje de modelado, ya que permite modelar el análisis, diseño e implementación, construyéndose así modelos precisos, no ambiguos y completos; además de ser flexible para admitir cambios no previstos durante el diseño o el rediseño.

1.4.3 Herramienta CASE

Se puede definir a las herramientas CASE (Ingeniería de Software Asistida por Ordenador, del inglés, *Computer Aided Software Engineering*) como un conjunto de programas que dan asistencia a los analistas, ingenieros de software y desarrolladores durante todos los pasos del ciclo de vida del *software,* proporcionándole un aumento en su productividad y logrando un mayor ahorro de tiempo (Scribd 2014).

Visual Paradigm for UML 8.0

La herramienta de modelado Visual Paradigm for 8.0 es una herramienta CASE profesional que soporta el ciclo de vida completo del desarrollo de *software*: análisis y diseño orientados a objetos, construcción, pruebas y despliegue; y no emplea una metodología en específico. El *software* de modelado UML ayuda a una rápida construcción de aplicaciones de calidad (Pressman 2008). Esta herramienta ofrece las siguientes características:

- ✓ Permite construir todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.
- ✓ Diseño centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad.
- ✓ Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- ✓ Capacidades de ingeniería directa (versión profesional) e inversa.
- ✓ Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- ✓ Disponibilidad de integrarse en los principales Entornos de desarrollo integrado como: *Eclipse/IBM webSphere, NetBeans IDE, Oracle JDeveloper*, entre otras (S. 2002).

Además de las ventajas expuestas, es válido resaltar que se selecciona Visual Paradigm for 8.0 debido a que es una herramienta que tiene disponibilidad en múltiples plataformas y es capaz de, a partir de un modelo relacional desplegar todas las entidades asociadas a las tablas para diferentes sistemas gestores de base de datos, aspecto importante en la realización de la presente investigación.

1.4.4 Sistema Gestor de Base de Datos (SGBD)

Un SGBD es un programa que facilita una serie de herramientas para la construcción de bases de datos (Scribd 2014) y obtener resultados de estas. Además de almacenar la información, se le pueden hacer preguntas sobre esos datos, obtener listados, generar pequeños programas de mantenimiento de la BD, o ser utilizado como servidor de datos para programas más complejos realizados en varios lenguajes de programación.

PostgreSQL 9.1

PostgreSQL constituye un potente gestor de BD de código abierto muy avanzado, ofrece un control de concurrencia multiversión lo que mejora las operaciones de bloqueo y las transacciones en sistemas multiusuario. Soporta casi toda la sintaxis SQL (del inglés, *Structured Query Language*), incluyendo sub-consultas, transacciones, tipos y funciones definidas (Martínez 2010). Además de la conectividad, velocidad, seguridad e interacción con el lenguaje de programación a utilizar para el desarrollo del módulo propuesto.

Para ello se escoge PostgreSQL 9.1 como SGBD debido a la estabilidad, potencia, robustez y facilidad de administración que provee. Además funciona con grandes cantidades de datos y una alta concurrencia de usuarios, lo que se traduce en una garantía de eficiencia, dado que el módulo en desarrollo debe ser capaz de enfrentar múltiples conexiones simultáneas.

1.4.5 Entorno de trabajo: Hibernate 4.1.3

Hibernate es un entorno de trabajo que tiene como objetivo facilitar la persistencia de objetos Java en bases de datos relacionales y al mismo tiempo la consulta de estas bases de datos para obtener objetos, mediante archivos declarativos (XML, del inglés, *eXtensible Markup Language*) o anotaciones en los *beans*[1] de las entidades que permiten establecer estas relaciones. Es una herramienta de Mapeo Objeto – Relacional (ORM, del inglés, *Object-Relational mapping)* de código abierto. Se encarga de transformar la consulta al dialecto SQL que toque, de los cuales soporta: DB2, MySQL, SAP DB, Oracle, , PostgreSQL, Microsoft SQL Server, entre otros (Hibernate 2015).

[1] Bean: componente software que tiene la particularidad de ser reutilizable y así evitar la tediosa tarea de programar los distintos componentes uno a uno.

Proporciona para la realización de consultas la opción de HQL (del inglés*, Hibernate Query Language*). Además dispone de otra modalidad de expresar las consultas: *Criteria*, la cual consiste en una serie de clases con las cuales se pueden expresar las condiciones. *Criteria* permite crear criterios de búsqueda para un objeto y hacer referencia a los objetos relacionados.

1.4.6 Entorno de desarrollo integrado

Un entorno de desarrollo integrado es una herramienta de *software* dedicada exclusivamente al desarrollo de programas informáticos, ofrece una serie de complementos que facilitan el desarrollo ágil de *software*. Entre los más usados actualmente para la programación en Java pueden mencionarse: Eclipse y NetBeans IDE. Para el desarrollo del módulo, se selecciona trabajar con Eclipse Helios 3.6 por una serie de aspectos que se detallan a continuación.

Eclipse Helios 3.6

Eclipse Helios es una plataforma de programación potente y completa para el desarrollo y compilación de elementos tan variados como sitios web, programas en C++ o aplicaciones Java. Es un entorno de desarrollo integrado (IDE, del inglés, *Integrated Development Environment*) en el que se encuentran todas las herramientas y funciones necesarias para el trabajo, recogidas además en una atractiva interfaz que lo hace fácil y agradable de usar. Emplea módulos (de inglés, *module*) como Tomcat, Axis2, Spring, JQuery para proporcionar toda su funcionalidad, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. La arquitectura basada en componentes permite escribir cualquier extensión deseada en el ambiente (Redinertho 2012).

Partiendo de las potencialidades anteriormente expuestas, se considera trabajar Eclipse Helios 3.6 como entorno de desarrollo integrado. A continuación se proponen los lenguajes utilizados para el desarrollo del módulo.

1.4.7 Lenguajes utilizados

Un lenguaje de programación está diseñado para describir el conjunto de acciones consecutivas que un equipo de cómputo debe ejecutar. Es aquella estructura con cierta base sintáctica y semántica, impone distintas instrucciones a un programa de computadora. Es un modo práctico para dar instrucciones a un

equipo de cómputo, que permite al desarrollador comunicarse con los dispositivos de *hardware* y *software* existentes (Lobos 2005).

Java es uno de los lenguajes de programación para la creación de aplicaciones. Ha desarrollado tres ediciones de plataformas diferentes, cada una de ellas destinada a cubrir un conjunto diferente de necesidades de programación. La Plataforma Java 2 Edición Estándar (J2SE, del inglés, *Java Platform 2 Standard Edition*) enfocada para crear una amplia variedad de aplicaciones y *applets*; la Plataforma Java 2 Edición Micro (J2ME, del inglés, *Java Platform 2 Micro Edition*) permite la creación de aplicaciones para micro-dispositivos; y la plataforma Java 2 Edición Empresarial (J2EE, del inglés, *Java Platform 2 Enterprise Edition*) destinada a crear aplicaciones de servidor (Allamaraju et al. 2006).

Las aplicaciones desarrolladas bajo la plataforma Java pueden ser desplegadas en cualquier sistema operativo donde se instale al JDK. **J2EE** representa la evolución de la plataforma de desarrollo del lado del servidor de Java hacia una especificación más madura y sofisticada. Es una plataforma para crear aplicaciones de servidor, proporciona la infraestructura para satisfacer estas necesidades. Entre las características más relevantes de esta plataforma se encuentran: el alto rendimiento, aplicaciones distribuidas y multiusuario, escalabilidad, gestión del estado, persistencia, transacciones, seguridad e interceptores.

La plataforma J2EE provee varias tecnologías que facilitan la creación de portales, incluyendo el desarrollo de interfaces de usuario, donde los portlets juegan un papel fundamental (Pérez 2008). Un portlet es un componente de interfaz web reutilizable, modular, gestionado y visualizado en un portal web, que producen un fragmento de lenguaje de marcado y puede ser mostrado en una página de un portal web. Los estándares de los portlets permiten al desarrollador de *software* que al ser creados puedan ser utilizados en cualquier portal que los soporte.

Los encargados de manipular y ejecutar el ciclo de vida de un portlet son los contenedores de portlets (Álvarez 2011). La Plataforma de SLVD que está en fase de desarrollo, a la cual se incorporará el módulo a implementar, utiliza para el desarrollo de sus aplicaciones el contenedor portlets **Liferay Portal 6.2** que al mismo tiempo es gestor de contenido, ofreciendo el soporte necesario para que de forma sencilla pueda publicarse continuamente la información.

Liferay Portal es un Sistema de Gestión de Contenidos basado en Java que permite la creación de portales web de una manera sencilla y rápida. Es una plataforma para el desarrollo, la integración y la colaboración. Ofrece todas las características necesarias para la creación de portales y aplicaciones web, todo ello desarrollado sobre una plataforma de código abierto. Incluye gran número de aplicaciones portlets tales como mensajería instantánea, foros y biblioteca de documentos. Soporta múltiples base de datos como: *PostgreSQL, MySQL, Oracle, SQL Server, Sybase e InterBase*. Se puede desplegar con muchos servidores como: *Jetty, JBoss, Sun GlassFish, Oracle AS* y *Apache Tomcat*. Además permite ejecutarse en cualquier sistema operativo como: Windows, Linux y MacOS (Jonas 2012).

Por lo antes expuesto, se empleará **Liferay Portal** como contenedor portlets en su versión **6.2**. Para la creación del portlet se tiene en cuenta otros lenguajes; los que se explican a continuación:

Para el proceso de validación de datos entrados por los usuarios al sistema y para la visualización del mapa cognitivo difuso, será empleado en la solución propuesta el lenguaje de programación: **JavaScript,** este permite ejecutar instrucciones como respuesta a las acciones del usuario. Además se utiliza principalmente para crear páginas web dinámicas. El uso más común de JavaScript es escribir funciones embebidas o incluidas en páginas HTML y que interactúan con el Modelo de Objetos del Documento (DOM, del inglés, *Document Object Model*) de la página web (Pérez 2009).

Para la creación de las páginas web se utilizarán hojas de estilo **CSS 3** (del inglés, *Cascading style sheets*) que servirán para: incluir márgenes, tipos de letra, fondos y colores.

Se selecciona el lenguaje **JSP 2.3** (*JavaServer Pages*) el cual permitirá crear páginas web dinámicas basadas en el Protocolo de Transferencia de HiperTexto (HTML, del inglés, *Hypertext Transfer Protocol*), el Lenguaje de marcas extensible (XML, del inglés, *Extensible Markup Language*), entre otros tipos de documentos. Para desplegar y correr JSP, se requiere un servidor web compatible con contenedores *servlet* como *Apache Tomcat* o *Jetty*.

La ventaja de JSP frente a otros lenguajes, es que Java es de propósito general que excede el mundo web y que es apto para crear clases que manejen lógica de negocio y acceso a datos de una manera detallada. Esto permite separar en niveles las aplicaciones web, dejando la parte encargada de generar el documento HTML en el archivo JSP.

1.4.8 Marco de trabajo

Los marcos de trabajo (del inglés, *frameworks)* son aplicaciones compuestas por componentes personalizables que permiten el desarrollo de aplicaciones. Las funcionalidades principales de estas aplicaciones es aligerar de carga a los desarrolladores realizando de manera automática e interactiva tareas comunes dentro de la programación, lo que permite reducir considerablemente el tiempo de desarrollo de aplicaciones complejas.

El marco de trabajo seleccionado es **Spring Framework 3.1.2**, éste permite el desarrollo de aplicaciones J2EE**.** Los autores lo definen como un *framework* ligero para construir aplicaciones empresariales, aunque se encuentre dividido en distintos módulos donde cada uno se encarga de partes diferentes de la aplicación. En general, estas son algunas de sus características (Alex 2010):

- ✓ Simplificación de la programación orientada a aspectos y del acceso a datos.
- ✓ Soporte para planificación de trabajos, para envió de correo (del inglés, email) y para acceso a componentes remotos.
- ✓ Manejo de Transacciones.
- ✓ Su propio framework MVC; además de su propio *Web Flow*.
- ✓ Manejo simplificado de excepciones.

La versión 3 de Spring incluye nuevas características, entre las que se incluyen (Alex 2010):

- ✓ Soporte para Java 5: Proporciona configuración basada en anotaciones. La parte web es compatible con las versiones 1.4 y 5 de J2EE.
- ✓ Lenguaje de Expresiones (SpEL): En esta nueva versión se incluye un lenguaje de expresiones que puede ser usando cuando se definen *beans*, tanto en XML como con anotaciones y también da soporte a través de todos los módulos de Spring.
- ✓ Soporte para Servicios Web REST: Spring soporta servicios web de tipo REST.
- ✓ Nueva organización de los módulos: Los módulos han sido revisados y separados en diferentes paquetes, más organizados, de acuerdo a su funcionalidad.

1.4.9 Servidor web

Un servidor web o servidor HTTP es un programa informático que procesa una aplicación del lado del servidor, realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente y generando o cediendo una respuesta en cualquier lenguaje o aplicación del lado del cliente (Mohammed 2010).

Apache Tomcat (también conocido como *Jakarta Tomcat* o Tomcat) es un servidor web de código abierto de *Java Servlet* y tecnologías *Java Server Pages* (JSP). Fue escrito en Java, lo que hace posible que funcione en cualquier sistema operativo que disponga de la máquina virtual Java (JVM, del inglés, *Java Virtual Machine*). Tomcat es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad (Tomcat 2015). Por lo antes expuesto, se empleará como servidor web: Apache Tomcat 7.0.42.

Conclusiones parciales

En el presente capítulo luego de una revisión bibliográfica, se pudo identificar que las soluciones que recoge la literatura científica, no satisfacen las necesidades desde la evaluación de competencias en los SLVD para la ejecución de prácticas de laboratorios. De las características, ventajas y desventajas de un SVLD, luego de analizar lo escrito en la literatura científica, para identificar las principales formas de evaluación de competencias, se seleccionó el modelo propuesto por (Mar 2014) para su implementación en el sistema de laboratorio propuesto.

Del análisis sobre las tendencias actuales en el desarrollo del software, se seleccionó la metodología de desarrollo y un grupo de tecnologías y herramientas, las adecuadas para realizar el módulo informático que se requiere: como metodología de desarrollo, la metodología ágil OpenUP; y como sistema de gestión de base de datos PostgreSQL 9.1, siendo Hibernate 4.1.3 el entorno de trabajo para facilitar las consultas a la base de datos; como lenguaje de modelado UML 2.0 y Visual Paradigm for UML 8.0 como herramienta CASE; Java como lenguaje de programación y Liferay Portal 6.2 como contenedor de portlets, haciendo uso de JavaScript, CSS 3 y JSP 2.3, además de Spring MVC 3.1.2 como marco de trabajo y Eclipse Helios 3.6 como IDE de desarrollo.

Capítulo 2: Propuesta de solución del módulo para la evaluación de competencias en la Plataforma de SLVD

Introducción

Sobre las bases de las ideas expuestas que se utilizarán en el desarrollo del módulo para la Plataforma de SLVD, se realiza el levantamiento de los requisitos, con el objetivo de dar una solución factible al problema existente. En este capítulo se definen el modelo conceptual, entidades y actores que intervienen, los requisitos funcionales y no funcionales, así como el diagrama de caso de uso del sistema y las relaciones existentes. El proceso de desarrollo a seguir estará guiado por las pautas que se rige la metodología de desarrollo de software OpenUP, lo cual será la base para la implementación del módulo informático para la evaluación de competencias en la Plataforma de SLVD.

2.1 Propuesta de solución

Se propone la implementación de un modelo para la evaluación de competencias a partir de un módulo informático que será integrado a la Plataforma de SLVD. El modelo basa su funcionamiento en mapa cognitivo difuso y operadores de agregación de la información como mecanismo regulatorio para el acceso a las prácticas de control en un SLVD. La propuesta contribuirá a determinar el índice de competencias del estudiante y apoyar el proceso de toma de decisiones para el acceso a la ejecución de la práctica de laboratorios dentro del SLVD.

2.2 Modelo conceptual

El modelo conceptual nos brinda los conceptos significativos para el dominio del problema. Puede mostrar: conceptos, asociaciones entre conceptos y atributos de conceptos. Se pretende con su realización unificar el vocabulario entre los usuarios y los desarrolladores, para poder entender el contexto en que se desarrollará un software (Larman 2003).

Debido a las características, grado de complejidad del problema que se plantea y el conocimiento que se posee sobre el mismo, se decide realizar el diagrama conceptual. Se considera suficiente el empleo de

esta variante de modelo de negocio, dado que se centra en una lo relacionado con el ámbito del proyecto. Se muestra en la Fig. 2 el diagrama del modelo conceptual de la problemática a resolver.

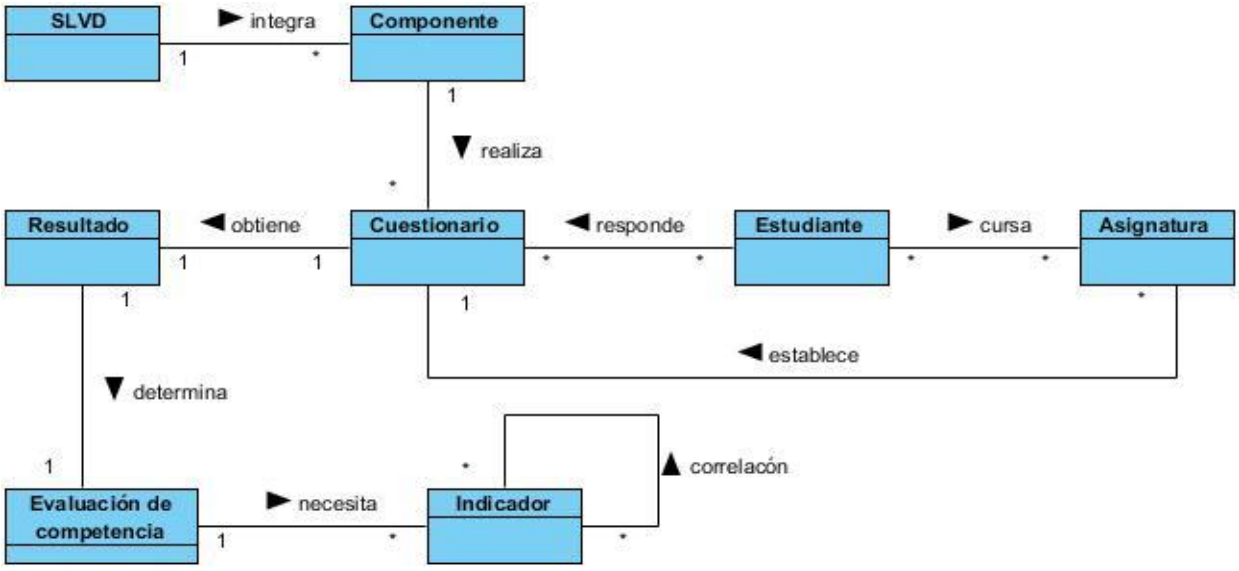


Fig. 2 Diagrama del modelo conceptual de la problemática a resolver

Definición de los conceptos del modelo conceptual

La Tabla. 2 muestra la definición de los conceptos principales que se emplean en el problema que se analiza.

Tabla. 2 Descripción de los conceptos que interactúan

Conceptos	***Descripción***
SLVD	Plataforma de Sistema de Laboratorio Virtual y a Distancia.
Componente	Módulos disponibles que permiten manipular materiales, instrumentos e ideas.
Cuestionario	Representa los cuestionarios que realizan los estudiantes para la evaluación de competencias.
Estudiante	Persona que realiza la evaluación de competencias.
Asignatura	Materia de estudio que se desea realizar.
Resultado	Se obtiene como resultado la propuesta que si la persona es competente o no para la realización de la práctica de laboratorio.

Evaluación de competencia	Modelo decisional que permite predecir problemas en el funcionamiento futuro o la necesidad de volver a capacitar a una persona en la plataforma.
Indicador	Factor que interviene en la evaluación de competencia.

2.3Especificación de los requisitos del sistema

Una etapa inicial y muy importante dentro del proceso de la Ingeniería de Software, es la Ingeniería de Requisitos, donde se lleva a cabo el proceso de descubrir, analizar, escribir y verificar los servicios y restricciones, del sistema de software que se desea producir; este proceso se realiza mediante la obtención, el análisis, la especificación, la validación y la administración de los requerimientos del software. Los requisitos de software son las necesidades de los clientes, los servicios que los usuarios desean que proporcione el sistema de desarrollo y las restricciones en las que debe operar (Sommerville 2005). Se arrojaron los requisitos funcionales y no funcionales presentados a continuación.

2.3.1 Requisitos Funcionales

Los 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 (Sommerville 2005). En el módulo a desarrollar se identifican los siguientes requisitos funcionales:

RF1: Adicionar nuevo indicador por asignatura para la evaluación de competencias.

RF2: Modificar indicador por asignatura para la evaluación de competencias.

RF3: Listar indicador por asignatura en la evaluación de competencias.

RF4: Eliminar indicador por asignatura en la evaluación de competencias.

RF5: Adicionar nueva asignatura para la evaluación de competencias.

RF6: Modificar asignatura para la evaluación de competencias.

RF7: Listar asignatura en la evaluación de competencias.

RF8: Eliminar asignatura en la evaluación de competencias.

RF9: Adicionar correlación de indicadores por asignatura para la matriz de adyacencia asociada.

RF10: Modificar correlación de indicadores por asignatura para la matriz de adyacencia asociada.

RF11: Listar correlación de indicadores por asignatura para la matriz de adyacencia asociada.

RF12: Calcular valores absolutos por asignatura para la evaluación de competencias.

RF13: Construir la matriz de adyacencia de valores absolutos por asignatura para la evaluación de competencias.

RF14: Visualizar la matriz de adyacencia de valores absolutos por asignatura para la evaluación de competencias.

RF15: Construir el Mapa Cognitivo Difuso de valores absolutos por asignatura para la evaluación de competencias.

RF16: Visualizar el Mapa Cognitivo Difuso de valores absolutos por asignatura para la evaluación de competencias.

RF17: Calcular grados de entrada de los indicadores por asignatura para la evaluación de competencias.

RF18: Calcular grados de salida de los indicadores por asignatura para la evaluación de competencias.

RF19: Calcular la centralidad causal de los indicadores por asignatura para la evaluación de competencias.

RF20: Calcular grados de entrada normalizado de los indicadores por asignatura para la evaluación de competencias.

RF21: Calcular grados de salida normalizado de los indicadores por asignatura para la evaluación de competencias.

RF22: Calcular la centralidad causal normalizada de los indicadores por asignatura para la evaluación de competencias.

RF23: Visualizar el análisis estático por asignatura para la evaluación de competencias.

RF24: Adicionar nueva pregunta por asignatura para la evaluación de competencias.

RF25: Modificar pregunta por asignatura para la evaluación de competencias.

RF26: Listar pregunta por asignatura en la evaluación de competencias.

RF27: Eliminar pregunta por asignatura en la evaluación de competencias.

RF28: Adicionar respuesta por pregunta de la asignatura para la evaluación de competencias.

RF29: Modificar respuesta por pregunta de la asignatura para la evaluación de competencias.

RF30: Listar respuesta por pregunta de la asignatura en la evaluación de competencias.

RF31: Eliminar respuesta por pregunta de la asignatura en la evaluación de competencias.

RF32: Subir imagen por pregunta para la evaluación de competencias.

RF33: Realizar cuestionario por asignatura para la evaluación de competencias.

RF34: Evaluar al estudiante por asignatura con la evaluación de competencias.

2.3.2 Requisitos No Funcionales

Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidos por el sistema (Sommerville 2005). Estas restricciones se ven como las características que hacen al producto agradable, usable, rápido o confiable. A continuación se identifican los siguientes:

Requisitos de Software

Para obtener un óptimo funcionamiento del sistema se requiere la existencia de los siguientes requisitos en el servidor y las máquinas clientes que harán uso de la aplicación.

Cliente

- ✓ Se requiere con sistema operativo: Ubuntu LTS 14.04 y Windows Seven o superior.
- ✓ Se requiere que cuente con un navegador web con capacidad de interpretación de JavaScript. Se recomienda Mozilla Firefox 32.0 o superior.

Servidor

- ✓ Se requiere con sistema operativo Ubuntu LTS 14.04 y Windows 7 o superior.
- ✓ Liferay Portal 6.2 o superior con servidor Apache Tomcat 7 o superior.
- ✓ Máquina virtual de Java 7 o superior
- ✓ Debe contar con PostgreSQL 9.1 o superior como SGBD.

Requisitos de Hardware

El buen estado de las conexiones de red es imprescindible y de suma importancia para la recogida y entrega de la información que será almacenada en el servidor y posteriormente solicitada por los usuarios.

Cliente

- ✓ Requiere que sea un Procesador Intel Pentium IV o superior a 2.5 Gigahercio (GHz), de 256 Megabytes (Mb) de RAM y posea una tarjeta de red a 100Mb o superior.

Servidor

- ✓ Requiere que sea un Procesador Intel Pentium IV o superior a 3GHz, con memoria RAM 2.0 Gigabytes (Gb) o superior y capacidad de disco duro 80Gb o superior.
- ✓ Cuente con tarjeta de red a 100Mb o superior.

Requisitos de Diseño e Implementación

- ✓ Se debe utilizar algunos de los estándares de codificación de Java (epígrafe 3.2: Código fuente).
- ✓ El acceso a la base de datos se debe garantizar a través del patrón Objeto de Acceso a Datos (DAO).
- ✓ Se empleará Hibernate 4.1.3 para la manipulación de las bases de datos.

Requisitos de Confiabilidad

El sistema implementa un conjunto de roles, para garantizar el control de acceso a los recursos en dependencia del nivel de acceso que posee el usuario. Por lo que permite que cada usuario tenga privilegios establecidos de acuerdo a su rol, lo que garantiza un alto porcentaje de fiabilidad con la información que se trabaja.

2.4 Modelo de caso de uso del sistema

Un caso de uso es una secuencia de interacciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema (Pressman 2008). A continuación se definen los actores que interactúan en el sistema:

2.4.1 Definición de los actores del sistema

Los actores del sistema son entidades distintas a los usuarios en el sentido de que estos son las personas reales que utilizaran el sistema, mientras que los actores representan cierta función que una persona real realiza (Weitzenfeld 2005). Se identifica en la Tabla. 3 los actores del sistema del módulo propuesto:

Tabla. 3 Actores del módulo propuesto

Actor	*Descripción*
Administrador	Es el encargado de administrar en el sistema las asignaturas y los indicadores valorativos por asignatura para la evaluación de competencias.
Experto	Es el encargado de establecer en el sistema la correlación entre los indicadores valorativos propuestos por asignatura. Además, puede captar información referente al análisis de evaluación de competencias por asignatura para las prácticas de laboratorio.
Profesor	Es el encargado de la gestión de preguntas por asignaturas para la realización del cuestionario a la evaluación de competencias por asignaturas para las prácticas de laboratorio.
Estudiante	Persona que desea acceder a las prácticas de control.

2.4.2 Diagrama de caso de uso del sistema

Los diagramas de caso de uso del sistema (CUS) sirven para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios y/u otros sistemas (Jacobson 2000). Los treinta y cuatro RF del sistema definido anteriormente (epígrafe 2.3.1: Requisitos Funcionales) quedan agrupados en diez CUS, de los cuales nueve de estos se agrupan de la siguiente manera:

CUS1 Gestionar indicador: RF1, RF2, RF3, RF4.

CUS2 Gestionar asignatura: RF5, RF6, RF7, RF8.

CUS3 Administrar correlación de indicadores: RF9, RF10, RF11.

CUS4 Generar matriz de adyacencia: RF12, RF13, RF14.

CUS5 Generar mapa cognitivo difuso: RF12, RF15, RF16.

CUS6 Generar el análisis estático: RF17, RF18, RF19, RF20, RF21, RF22, RF23.

CUS7 Gestionar pregunta: RF24, RF25, RF26, RF27.

CUS8 Gestionar respuesta: RF28, RF29, RF30, RF31.

CUS9 Evaluar competencia: RF33, RF34.

La Fig. 3 muestra el diagrama de CUS donde se ponen de manifiesto los doce CUS, empleando los patrones de estructuración de casos de uso:

- ✓ CRUD (del inglés, *Creating, Reading, Updating, Deleting*) completo vigente en los CUS: “Gestionar indicador”, “Gestionar asignatura”, “Gestionar pregunta”, “Gestionar respuesta”.
- ✓ CRUD parcial vigente en el CUS “Administrar correlación de indicadores”.
- ✓ Múltiples actores (herencia o generalización entre actores) con roles comunes: el actor especializado (Experto) hereda el CUS del actor general (Profesor).

- ✓ Inclusión abstracta se evidencia en el CUS “Gestionar pregunta” que incluye al CUS “Gestionar respuesta”.
- ✓ Extensión concreta que se presenta en el CUS “Gestionar pregunta” que extiende o añade al CUS “Subir imagen”.

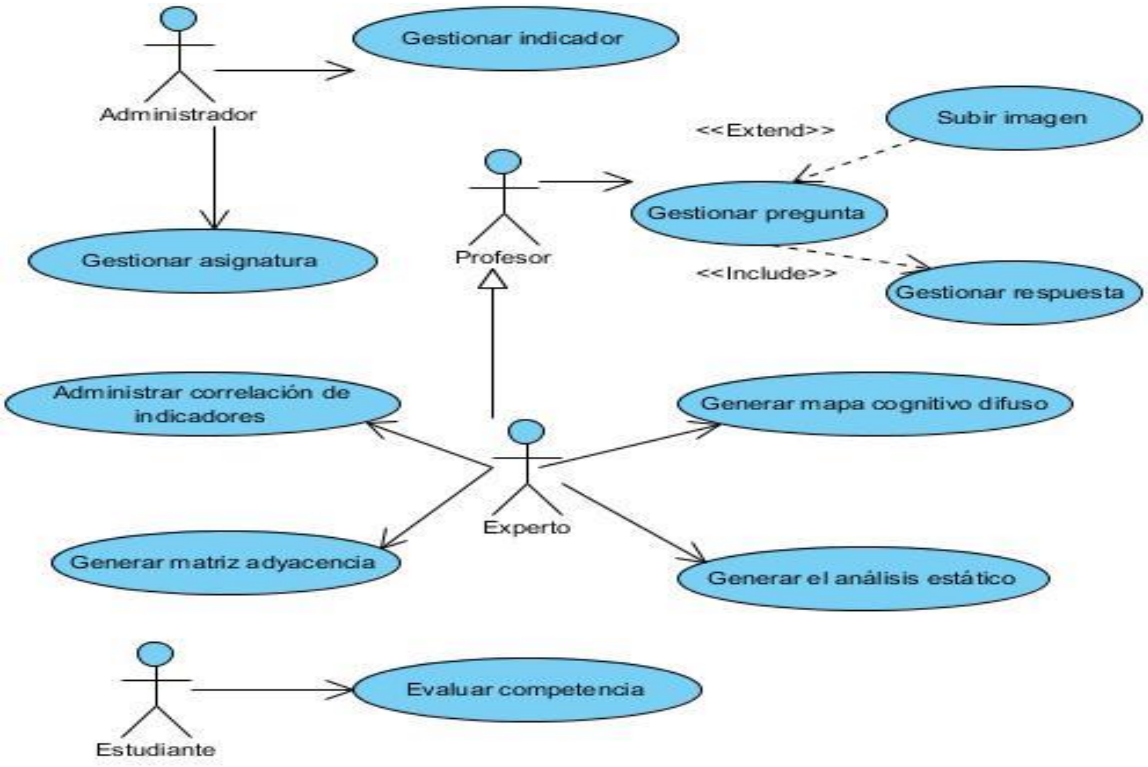


Fig. 3 Diagrama del Modelo de Casos de Uso del Sistema

2.4.3 Descripción textual de los casos de uso del sistema

Cada CUS se detalla habitualmente mediante una descripción textual que describe la funcionalidad que se construirá en el módulo propuesto. La Tabla. 4 presenta un ejemplo de la descripción textual del CUS3: Administrar correlación de indicadores.

CUS3: Administrar correlación de indicadores

Tabla. 4 Descripción CUS3: Administrar correlación de indicadores

Caso de Uso	*Administrar correlación de indicadores*

Actores	*Experto*
Resumen	*Se registra todo lo referente a lo expresado con la correlación de los indicadores, para lo cual se toma la escala definida. Las acciones que se podrán ejecutar son: evaluar un indicador, listar y modificar en caso de que el mismo ya tuviera valores. Estas acciones sólo se verán reflejadas en la BD cuando se guarde el formulario.*
Complejidad	*Alto*
Prioridad	*Crítico*
Precondiciones	*Para que se ejecute el caso de uso, el experto debe de estar previamente autenticado en el sistema. Además debe el sistema tener registrada al menos una asignatura y para esta al menos dos indicadores, para poder establecer la correlación entre estos.*
Postcondiciones	*En dependencia de la acción que ejecute:* • *Evaluar correlación de indicador.* • *Listar correlación de indicador.* • *Modificar correlación de indicador.*
Referencias	*RF9, RF10, RF11.*

Flujo de eventos		
Flujo básico		
	Actor	**Sistema**
1.	*Selecciona el menú administrar correlación de indicadores.*	
2.		*Permite realizar varias acciones:* - *Evaluar correlación de indicador. Ver Sección 1: "Adicionar correlación de indicadores".* - *Listado correlación de indicadores. Ver Sección 2: "Listar correlación de indicadores".* - *Modificar correlación de indicador. Ver Sección 3: "Modificar correlación de indicadores".*
3.		*Termina el caso de uso*
Prototipo de interfaz		

Fig. 4 Prototipo de interfaz CUS Administrar correlación de indicadores

Sección 1 "Adicionar correlación de indicadores"		
	Actor	**Sistema**
	1. Selecciona un indicador y escoge la opción Evaluar Indicador.	*2. Muestra la ventana Evaluar Indicador.*
	3. Selecciona para el indicador la correlación que existe con el indicador seleccionado y selecciona la opción Aplicar/Aceptar.	*4. Inserta en la BD el valor correlacionar establecido.*
		5. Se cierra la ventana del evaluar.
Flujos alternos		
	Actor	**Sistema**
	3b. Selecciona la opción Cancelar.	*4b. Cierra la ventana del evaluar y no efectúa ningún cambio.*
Prototipo de interfaz		

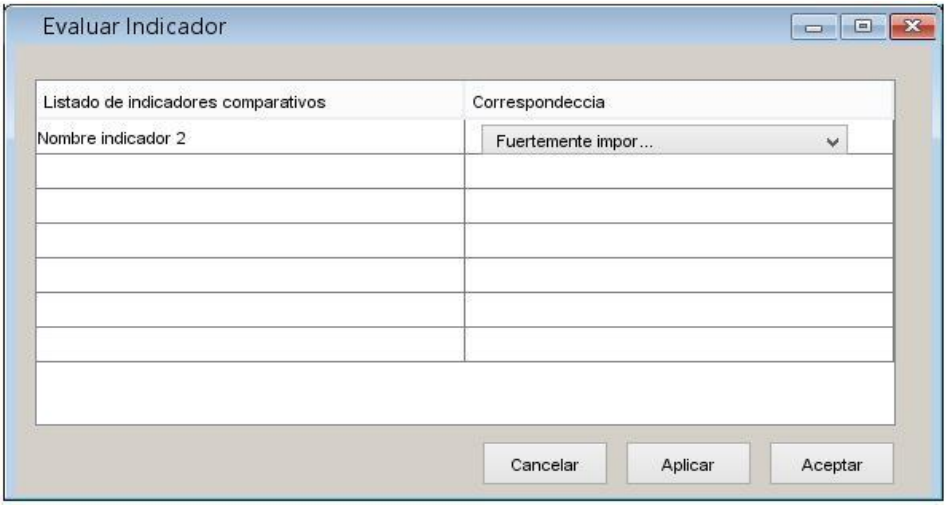


Fig. 5 Prototipo de interfaz: Evaluar indicador

Sección 2 "Listar correlación de indicadores"		
	Actor	**Sistema**
	1. Selecciona un indicador.	*2. Lista el indicador y su correlación con los otros indicadores y el impacto que se estableció.*

Prototipo de interfaz

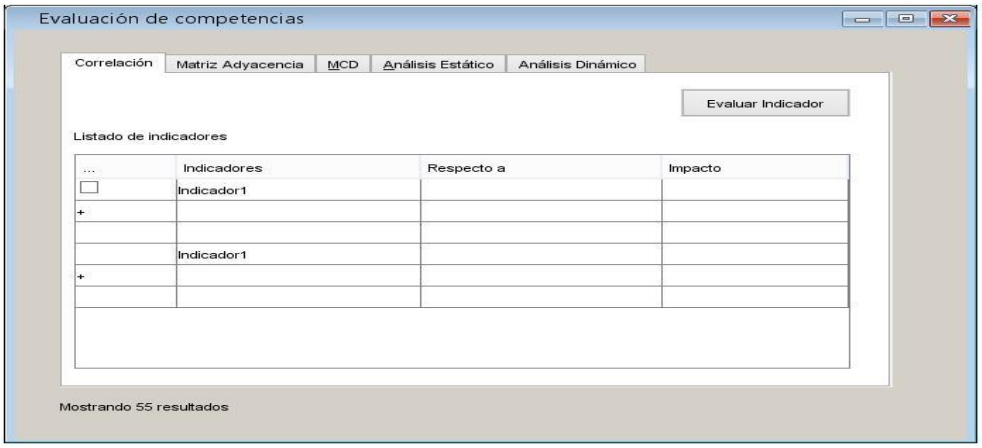


Fig. 6 Prototipo de interfaz: Listar correlación de indicadores

Sección 3 "Modificar correlación de indicadores"		
	Actor	**Sistema**
	1. Selecciona un indicador y escoge la opción Evaluar Indicador.	*2. Muestra la ventana Evaluar Indicador.*
	3. Modifica el valor que desea establecer para	*4. Modifica en la BD el valor correlacionar*

	el indicador con relación a otro indicador y selecciona la opción Aplicar/Aceptar.	*establecido.*
		5. Se cierra la ventana del evaluar.
Flujos alternos		
	Actor	**Sistema**
	3b. Selecciona la opción Cancelar.	*5b. Cierra la ventana del evaluar y no efectúa ningún cambio.*

Prototipo de interfaz

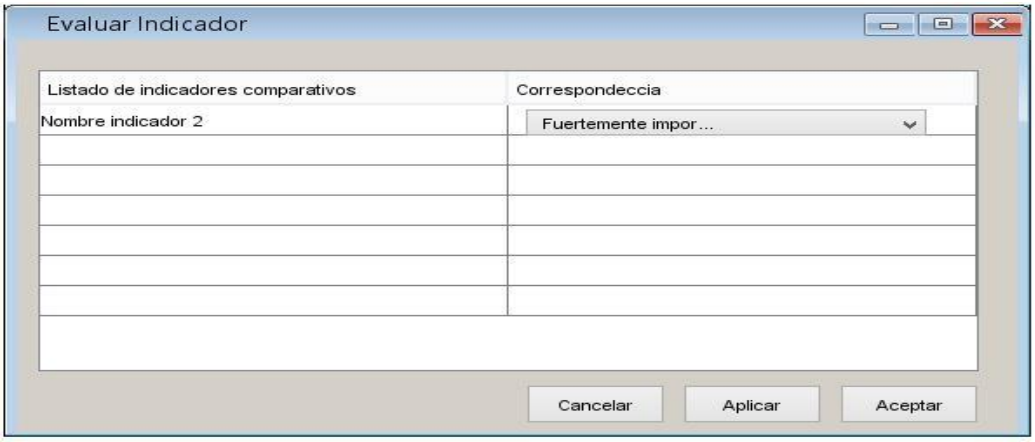


Fig. 7 Prototipo de interfaz: Modificar correlación de indicadores

Relaciones	**CU Incluidos**	*No procede*
	CU Extendidos	*No procede*
Requisitos no funcionales	**RNF: Requisito de Diseño e Implementación** *Cada clase cuenta con la información necesaria en el acceso a la base de datos para cumplir la responsabilidad; el acceso a la BD se garantizará a través del patrón Objeto de Acceso a Datos (DAO)*	

2.5 Diseño del sistema

"El diseño del software es un proceso iterativo mediante el cual los requisitos se traducen en un "plano" para construir el software" (Pressman 2008). El diseño del sistema define la arquitectura y estructura de hardware, software, componentes, módulos y datos de un sistema de cómputo. Se refiere a la formulación de especificaciones para el nuevo subsistema propuesto, de manera que satisfaga los requisitos determinados durante la fase de análisis, tanto desde el punto de vista funcional como del no funcional.

2.5.1 Patrón arquitectónico

Un patrón es una solución reutilizable de problemas recurrentes que ocurren durante el desarrollo del *software*. Existen diferentes tipos de patrones dentro de los que se encuentran los patrones arquitectónicos. Estos definen la estructura de un sistema *software*, los cuales a su vez se componen de subsistemas con sus responsabilidades. También tienen una serie de directivas para organizar los componentes del mismo sistema, con el objetivo de facilitar la tarea del diseño de tal sistema (Bass et al. 2015).

Patrón arquitectónico Modelo-Vista-Controlador

Para la realización del módulo a implementar se define utilizar el patrón arquitectónico Modelo-Vista-Controlador (MVC), ya que admite una separación de conceptos permitiendo que el desarrollo de la aplicación esté estructurado de una mejor forma, además de ofrecer una mayor escalabilidad (Almeira and Cavenago 2007). Como característica esencial se destaca la manera que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos: el modelo, la vista y el controlador.

Además, éste permite crear nuevos tipos de datos si la aplicación lo requiere, pues el funcionamiento de las otras capas es independiente y facilita el tratamiento de errores. Por otra parte, se tiene en cuenta que al utilizarse Spring como marco de trabajo, utiliza lo mejor de la arquitectura MVC e implementa de forma que el desarrollo de aplicaciones sea rápido y sencillo representándolo a través de tres elementos fundamentales antes expuestos; siendo esta otras de las ventajas principales para la selección del patrón arquitectónico a utilizar. Se muestra en la Fig. 8 una representación de la estructura del patrón MVC para la propuesta de solución, en la Fig. 9 la vista lógica de la misma y luego se detalla la utilización de los tres componentes.

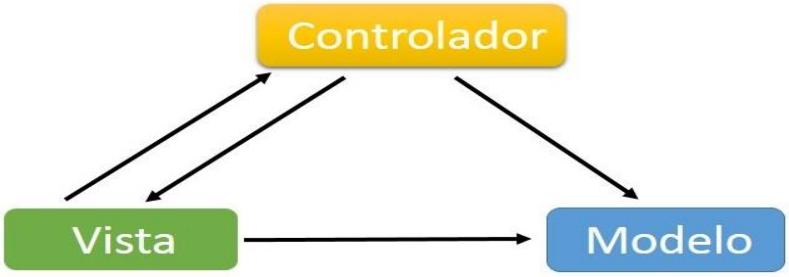


Fig. 8 Representación de la estructura del patrón MVC para la propuesta de solución

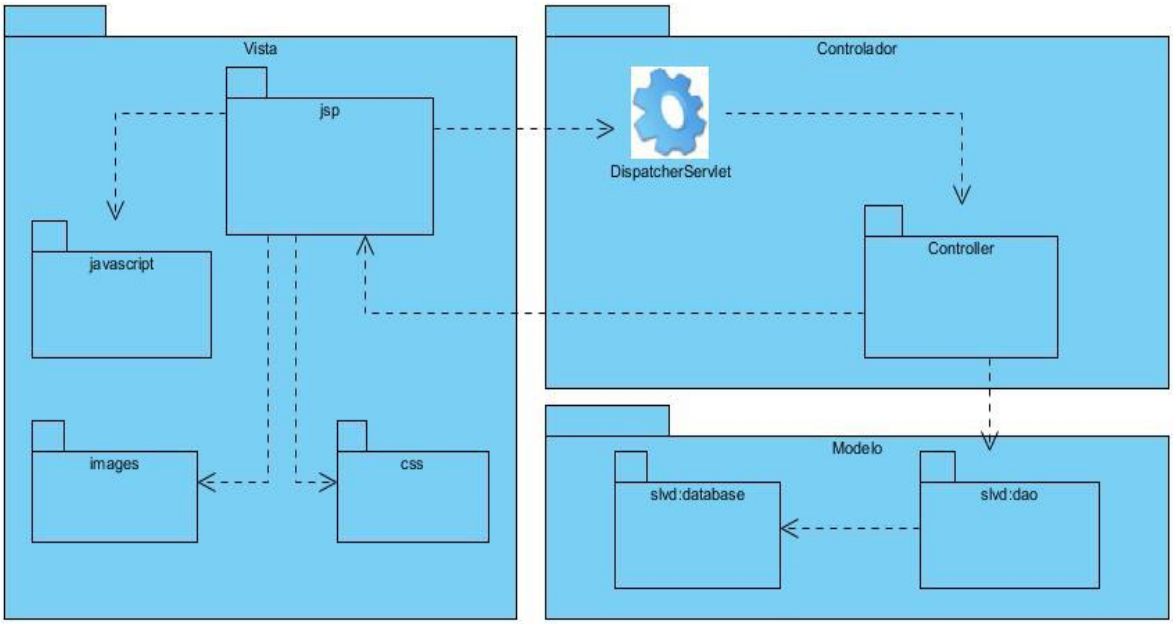


Fig. 9 Vista lógica de la estructura del patrón MVC para la propuesta de solución

El modelo es la capa que gestiona los datos y accede a la capa de almacenamiento de datos. Para la solución propuesta, la capa agrupa los paquetes *"slvd.database"* y *"slvd.dao"*. Para el primer caso, el paquete *"slvd.database"* recoge las entidades del sistema que se crean a partir de las tablas de la base de datos. Al respecto, el paquete *"slvd.dao"* contiene las entidades que se encargan del manejo y la selección de los datos.

Se observa también, las **vistas,** la cual presenta la información del sistema al usuario y genera los eventos de la interacción con éste. Por su parte captura eventos del usuario y se los envía al sistema a

través del controlador; además de recibir mandatos del controlador y mostrar información al usuario. Sobre la base de las ideas expuestas, en la propuesta de la solución, dicha capa contiene los paquetes *"jsp"*, *"css"*, *"js"* e *"images".* El primero recoge las páginas JSP que conforman al sistema; el segundo las hojas de estilo aplicadas a dichas páginas; el tercero contiene los archivos *JavaScript*; y el último los íconos e imágenes que presenta en la página inicial y los botones a seleccionar para determinadas funciones.

Finalmente, en el **controlador** se gestionan todas las peticiones hechas por los usuarios: recibe eventos del usuario, invoca servicios ofrecidos por el modelo y selecciona la vista adecuada para presentar los resultados. El sistema utiliza el *DispatcherServlet* de Spring como controlador frontal, una vez llegada una petición asigna las peticiones a las clases controladoras encargadas. De este modo, se utilizan las clases controladoras quienes se encargan de crear y devolver una respuesta una vez que han recibido una petición, las cuales se agrupan en el paquete *"slvd.controllers"*.

2.5.2 Modelo de diseño

El principal propósito del modelo de diseño consiste en modelar el sistema y encontrar su forma para que soporte todos los RF, para profundizar en los aspectos relacionados con los RNF y restricciones relacionadas con lenguajes de programación, sistemas operativos, tecnologías de interfaz de usuario y componentes reutilizables. Este modelo lo describe la realización física de los casos de uso, constituyendo una entrada principal en la actividad de implementación (Jacobson 2000), el cual está compuesto por los diagramas de clases del diseño donde se muestra en la Fig. 10 un ejemplo.

2.5.3 Diagrama de clase del diseño (DCD)

Los diagramas de clases del diseño son los más utilizados en el modelado de sistemas orientados a objetos. Estos muestran un conjunto de clases, interfaces y colaboraciones, así como sus relaciones. Se utilizan para modelar la vista de diseño estática de un sistema y se obtiene como resultado del refinamiento del modelo conceptual (Jacobson 2000). Los DCD en la propuesta de solución se modelaron separados por CUS. En la Fig. 10 se muestra un ejemplo de DCD del CUS3: Administrar correlación de indicadores.

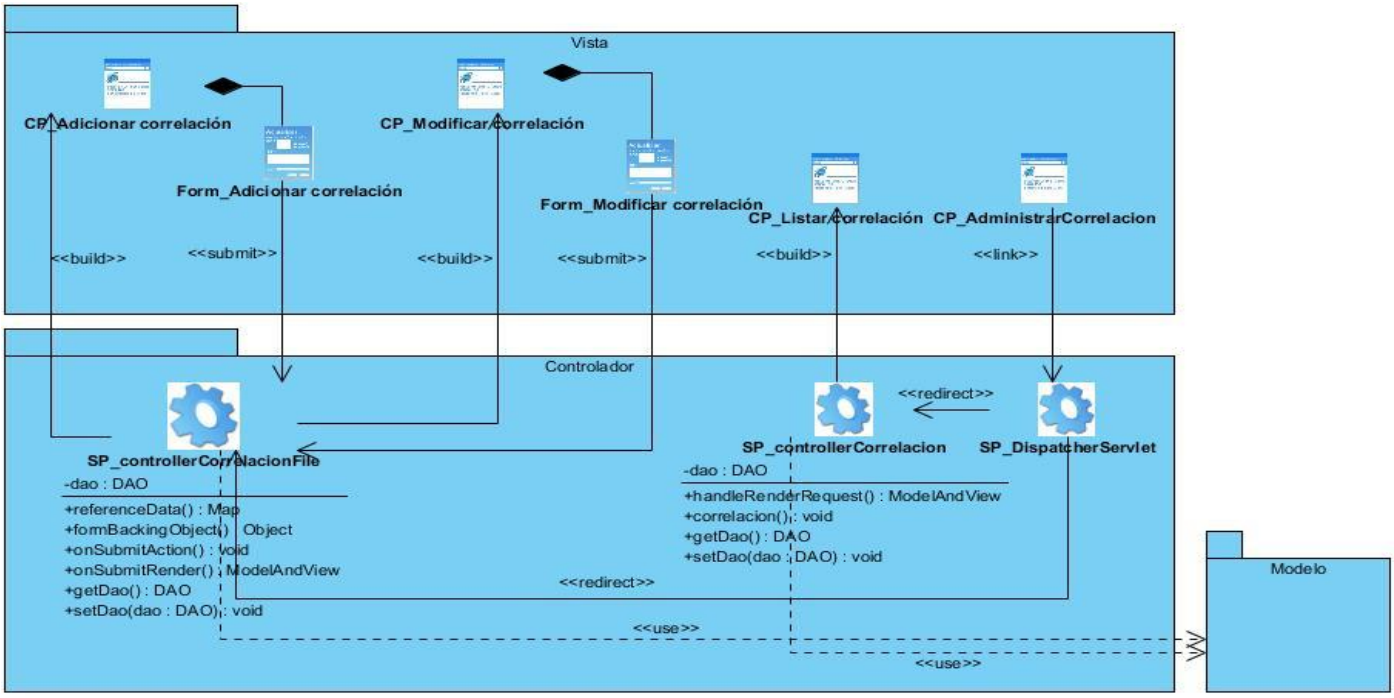


Fig. 10 DCD CUS3: Administrar correlación de indicadores

La Fig. 11 representa el paquete Modelo del DCD donde se ubican todas las clases necesarias para la gestión de la BD, función que realiza a partir del empleo de la interfaz *DAO.java*. La clase *DAOImplementado.java* implementa todos los métodos especificados en la clase interfaz y utiliza las clases indicadorHome.java, *matrizAdyAsocHome.java*, *matrizAdyHome.java* y *variableLengHome.java* para acceder a las tablas del modelo, representado el acceso a datos para los demás componentes del sistema.

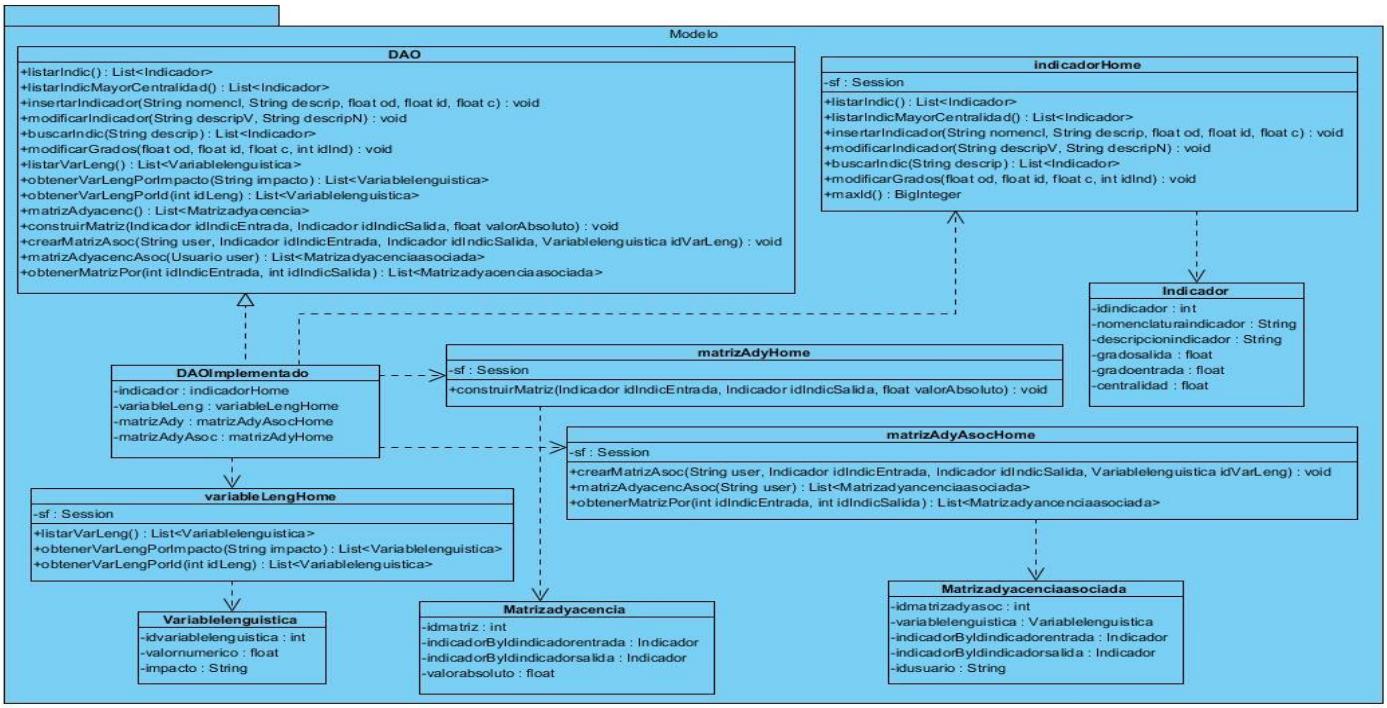


Fig. 11 Paquete Modelo del diagrama de clases del diseño

2.5.4 Patrones de diseño

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y para lograr una mayor calidad en el diseño (Bass, Clements and Kazman 2015).

Durante el diseño del módulo se utilizaron Patrones Generales de Software para Asignación de Responsabilidades (GRASP, del inglés, *General Responsability Asignment Software Patterns*). Los patrones GRASP describen entre otros elementos la asignación de responsabilidades a objetos (López et al. 2004). Los patrones empleados en el diseño fueron:

- ✓ **Experto**: Plantea que se debe asignar la responsabilidad de realizar determinada operación, a la clase que tiene la información necesaria para cumplir con dicha responsabilidad (López, Ramon, Sarroca and Seone 2004).

En la solución propuesta el patrón se pone de manifiesto debido que al utilizarse Hibernate como herramienta ORM para la capa del modelo, éste se encarga de crear una clase experta por cada tabla de

la BD, como se pudo evidenciar en la Fig. 11: *Indicador*, *MatrizAdyacencia*, entre otros; lo que permite que se pueda manejar su información como un objeto de la entidad mapeada.

- ✓ **Creador**: Este patrón se tiene en cuenta para la asignación de responsabilidades a las clases relacionadas con la creación de objetos, de forma tal que una instancia de un objeto solo pueda ser creada por la clase que contiene la información necesaria (López, Ramon, Sarroca and Seone 2004).

Se muestra en la Fig. 12 un ejemplo donde se pone de manifiesto el patrón Creador. Es el caso del *controllerMatriz.java* quien tiene la responsabilidad de crear el objeto matriz e instancia a la clase *DAO.java* quien contiene la información necesaria para crearla. El uso del mismo permite crear las dependencias mínimas necesarias entre las clases, favoreciendo al mantenimiento del sistema.

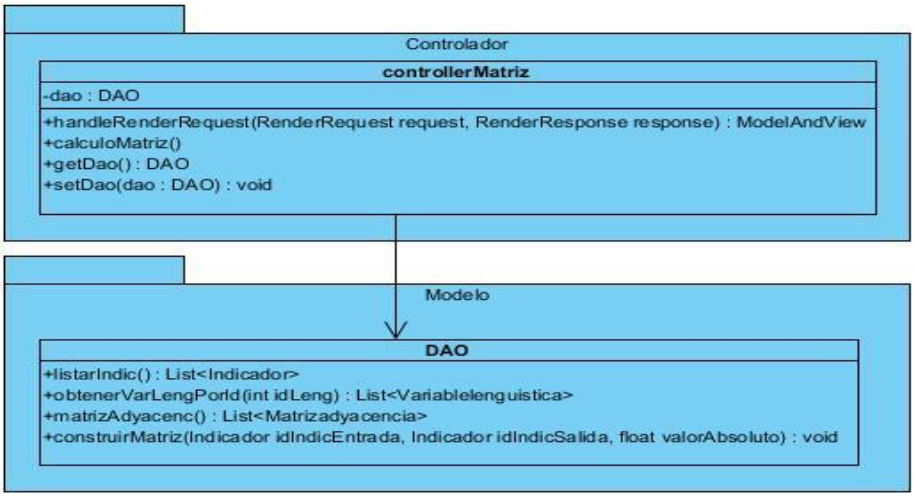


Fig. 12 Comportamiento del patrón Creador en el modelado de diseño

- ✓ **Alta Cohesión**: Asignar a las clases responsabilidades de forma tal que trabajen sobre una misma área de la aplicación y que no tengan mucha complejidad (López, Ramon, Sarroca and Seone 2004).

Este patrón se evidencia en las clases del modelo representado en la Fig. 11, las cuales tienen solo los atributos y métodos necesarios para que estas cumplan con su función.

- ✓ **Bajo Acoplamiento:** Asignar las responsabilidades de forma tal que las clases se comuniquen con el menor número de clases que sea posible (López, Ramon, Sarroca and Seone 2004).

En la capa del modelo representada en la Fig. 11 se encuentran las clases que implementan la lógica de acceso a datos, estas clases tienen pocas asociaciones con las entidades por lo que la dependencia en este caso es baja, poniéndose de manifiesto el patrón Bajo Acoplamiento. En la Fig. 13, se refleja un ejemplo de clase que está contenida dentro del Paquete Modelo donde se manifiesta dicho patrón.

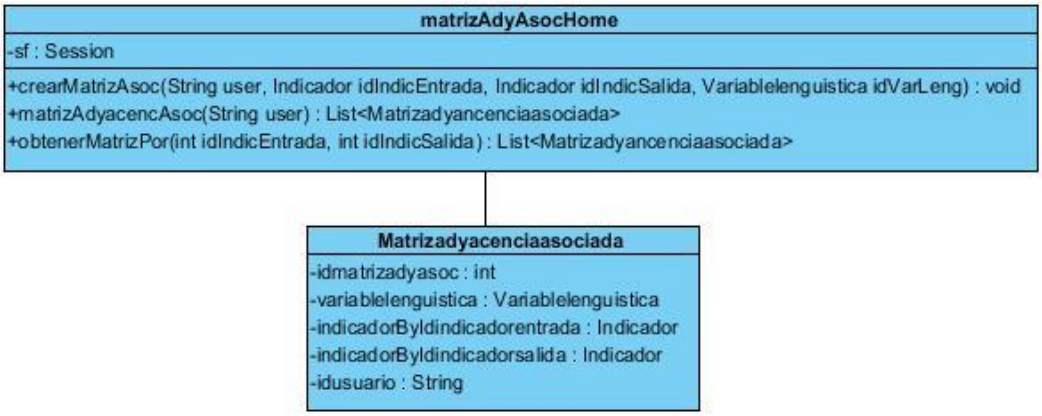


Fig. 13 Comportamiento del patrón Bajo Acoplamiento en el modelado de diseño

- ✓ **Controlador:** Consiste en tener una clase que sirva como intermediaria entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado (López, Ramon, Sarroca and Seone 2004).

El sistema utiliza el *DispatcherServlet* de Spring como controlador frontal, una vez llegada una petición asigna al controlador "*controllerCorrelacion*", "*controllerCorrelacionFile*" o cada uno de los controladores que se encargan de procesar una petición en particular; para que la responda, en dependencia de la URL solicitada se retorna la información correspondiente en formato JSP. A través del DCD de la Fig. 10 se puede observar la evidencia de lo anteriormente planteado.

Los patrones GoF (conocidos como Grupo de los cuatro, del inglés, *Gang of Four*), también forman parte de los patrones de diseño y proponen soluciones para diferentes clases de problemas en el desarrollo del *software* (Larman 2003). Se explica a continuación los patrones GoF utilizado en el módulo a implementar.

- ✓ **Solitario (del inglés, *Singleton*):** Se manifiesta al garantizarse que una clase solo tenga una única instancia, lo cual proporcione un único punto de acceso local a esta (Larman 2003). Este patrón se evidencia por ejemplo en la clase "*SessionFactory*" encargada de acceder a las funcionalidades de *Hibernate*, la cual al constituir una única instancia es accedida directamente sin que exista la posibilidad de crear nuevos objetos de su tipo.
- ✓ **Fachada (del inglés, *Facade*):** Este patrón define una clase con una interfaz común a un grupo de componente o a un conjunto heterogéneo de interfaces. Los elementos heterogéneos pueden ser las clases de un paquete, un conjunto de funciones, un esquema o un subsistema (Larman 2003).

En la propuesta ofrecida será definida una interfaz común para interactuar con la base de datos, asignándole la responsabilidad de ejecutar todas las consultas. Esta interfaz queda representada en la clase *DAOImplementado.java* donde se refleja dicho patrón mostrado en la Fig. 14.

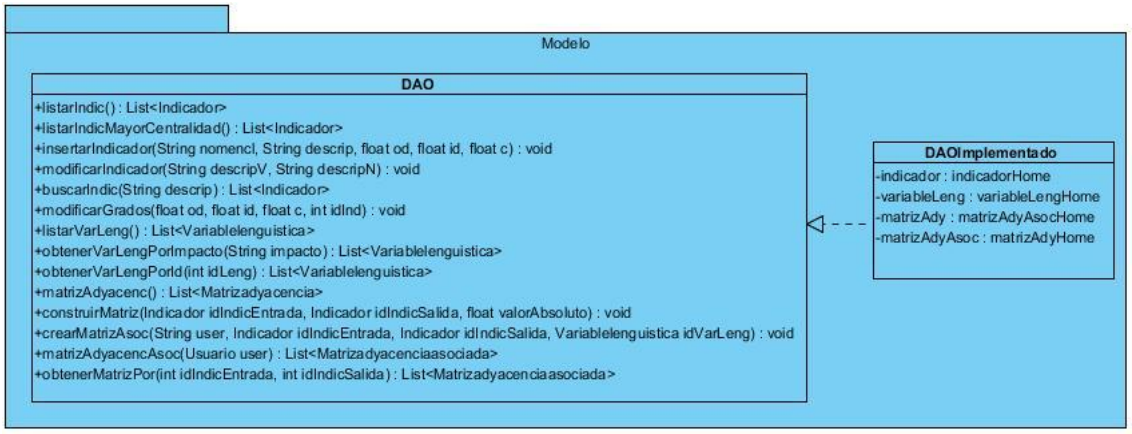


Fig. 14 Comportamiento del patrón Fachada en el modelado de diseño

2.5.5 Modelo de datos

El modelo de datos describe la representación lógica y física de los datos persistentes (Jacobson 2000). El modelo de datos estará compuesto por las entidades que pasarán a ser las tablas de la base de datos que será utilizada por el sistema. Se presenta en la Fig. 15 el diagrama entidad relación.

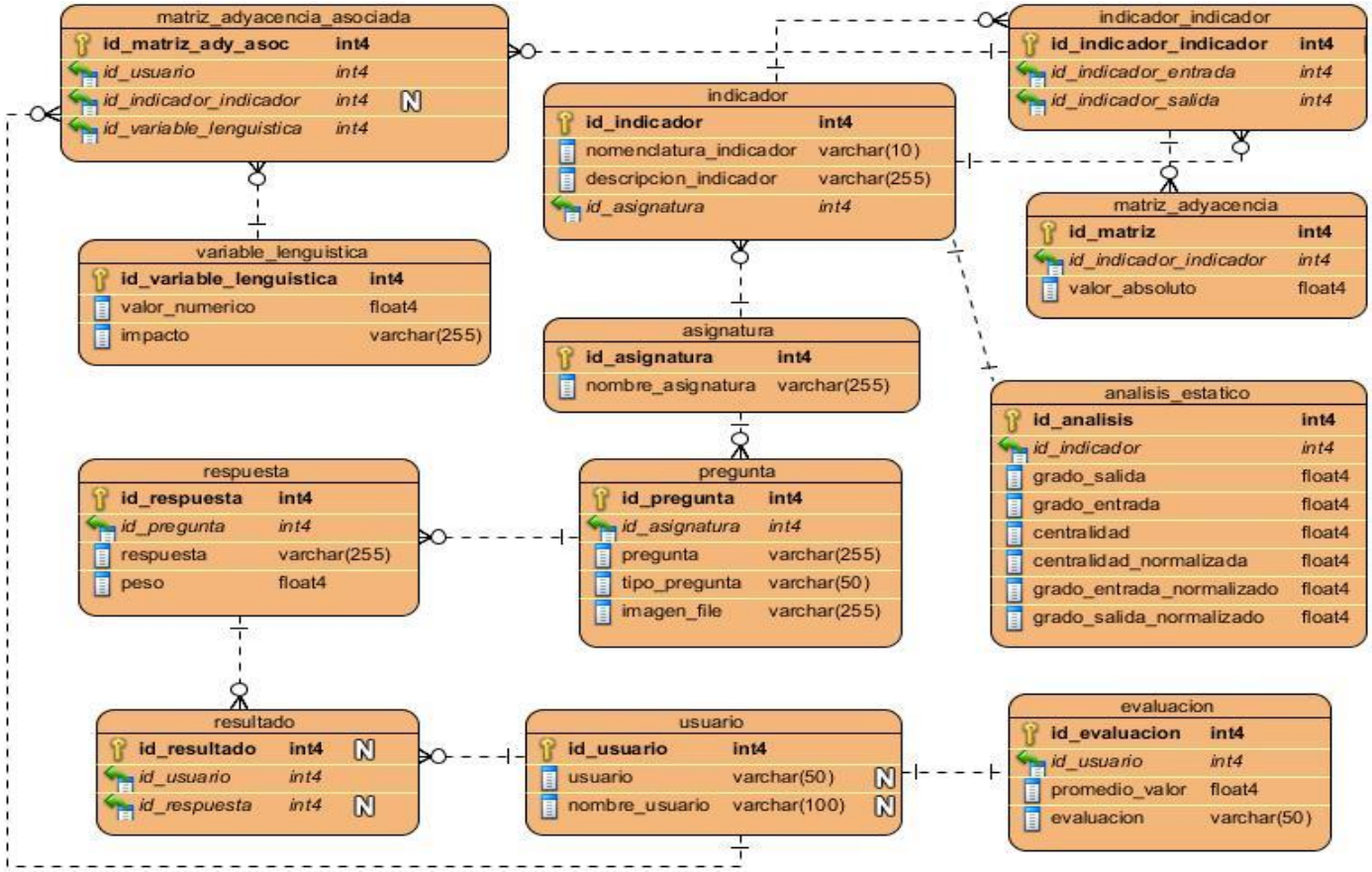


Fig. 15 Diagrama Entidad Relación

Conclusiones parciales

La realización del análisis del módulo a implementar permitió obtener una representación visual de las principales relaciones entre los conceptos asociados a la problemática a través del modelo conceptual. Se identificaron treinta y cuatro requisitos funcionales que fueron agrupados en diez casos de uso que el sistema debe cumplir para su correcto funcionamiento; y diez requisitos no funcionales que le aportan cualidades significativas al producto. Para estructurar el sistema se utilizó el patrón arquitectónico MVC que propone Spring, lo que facilitó la escalabilidad, adaptabilidad y el mantenimiento. La utilización de patrones GRASP (experto, creador, alta cohesión, bajo acoplamiento y controlador) y GoF (solitario y fachada) permitió obtener diseños de clases robusto al estructurarlos adecuadamente. El diseño del modelo de datos representado evidencia la descripción de la estructura de datos y sus restricciones.

Capítulo 3: Implementación y prueba del módulo para la evaluación de competencias en la Plataforma de SLVD

Introducción

En este capítulo, basado en la propuesta de solución planteada en el capítulo anterior, se procede a desarrollar el flujo de trabajo de implementación. En la que se describen sus principales artefactos, destacando el modelo de implementación que incluye componentes, subsistemas de implementación, diagramas de componentes y el modelo de despliegue del sistema. Posteriormente se le aplican las pruebas al módulo partiendo de la confección y descripción de los casos de prueba.

3.1 Modelo de implementación

El modelo de implementación describe la estructura general del *software* con vistas a su construcción, ejecución e instalación. Describe cómo los elementos del modelo de diseño, como las clases, se implementan en términos de componentes, ficheros de código fuente, ejecutables, etc. También describe cómo se organizan los componentes de acuerdo a los lenguajes de implementación empleados, y cómo dependen los componentes unos de otros (Jacobson 2000). Con el modelado del diagrama de componentes y el modelo de despliegue del sistema se describe la estructura del módulo para su construcción, ejecución y despliegue.

3.1.1 Diagrama de componentes

Un diagrama de componentes muestra las organizaciones y dependencias lógicas entre componentes de *software*: código fuente, binarios o ejecutables. Desde el punto de vista del diagrama de componentes se tienen en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión del *software*, la reutilización y las limitaciones imputadas por los lenguajes de programación y las herramientas utilizadas en el desarrollo (Jacobson 2000). En la Fig. 16 se muestra el diagrama de componentes del CUS5: Generar mapa cognitivo difuso.

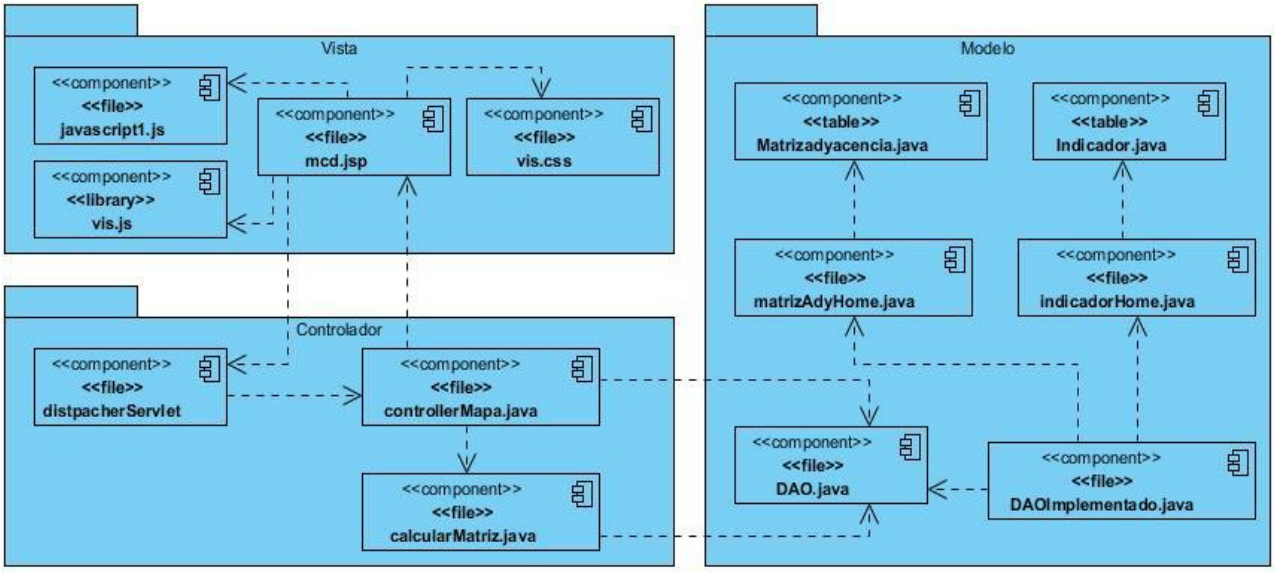


Fig. 16 Diagrama de componentes CUS5: Generar mapa cognitivo difuso

Se muestra en la Tabla. 5 una breve descripción de los componentes expuestos en la Fig. 15 del diagrama de componentes del CUS5: Generar mapa cognitivo difuso.

Tabla. 5 Descripción de los componentes del CUS5: Generar mapa cognitivo difuso

Componentes	***Propósito del componente***
mcd.jsp	Página JSP que muestra el mapa cognitivo difuso.
vis.css	Archivo de visualización que permite realizar conexiones y relaciones entre diferentes nodos.
vis.js	Librería de visualización que permite realizar conexiones y relaciones entre diferentes nodos.
javascript1.js	Archivo que permite capturar lo que se quiere visualizar.
distpacherServlet	Archivo de configuración encargado de atender todas las peticiones y enviarlas a las clases controladoras encargadas.
controllerMapa.java	Clase java que procesa y responde a la petición de construir y visualizar el mapa cognitivo difuso.
calcularMatriz.java	Clase java que procesa y responde a la petición de calcular y construir la matriz de adyacencia.

DAO.java	Interfaz del lenguaje Java que declara los métodos necesarios para acceder a la BD.
DAOImplementado.java	Clase java que se encarga de implementar los métodos que permiten el acceso a la BD.
matrizAdyHome.java	Clase responsable de manipular la tabla Matrizadyacencia.
indicadorHome.java	Clase responsable de manipular la tabla Indicador.
Matrizadyacencia.java	Archivo responsable de asociar a la matrizadyacencia con la BD.
Indicador.java	Archivo responsable de asociar el indicador con la BD.

3.1.2 Modelo de despliegue

Un diagrama de despliegue muestra la configuración de los nodos que participan en la ejecución y de los componentes que residen en estos. Muestra la configuración de los elementos de procesamiento en tiempo de ejecución, los componentes de software, hardware, procesos y objetos que los ejecutan. Este diagrama es útil para ilustrar la arquitectura física de un sistema (Jacobson 2000).

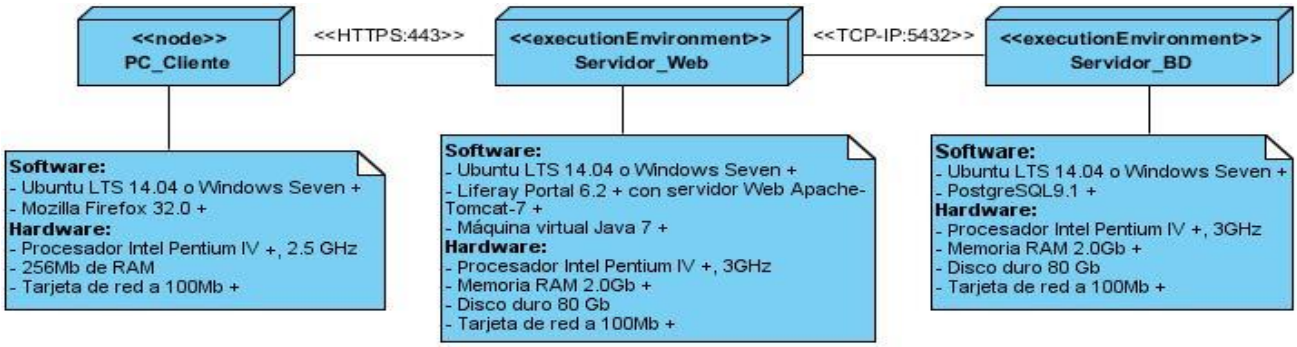


Fig. 17 Modelo de despliegue

Para interactuar con la aplicación es necesario contar con una *PC_Cliente* que tenga instalado un navegador web que permita comunicación con el servidor de aplicaciones mediante el puerto 443 y el protocolo HTTPS (Protocolo Seguro de Transferencia de Hipertexto, del inglés, *HyperText Transfer Protocol Secure*). En el servidor de aplicaciones (*Servidor_Web*) se cuenta con la lógica de la aplicación, los datos persistentes resultantes de la misma se almacenan en el servidor de bases de datos (*Servidor_BD*), accesible mediante el puerto 5432 y el protocolo TCP-IP (protocolo de control de transmisión de datos entre computadoras, del inglés, *Transmission Control Protocol – Internet Protocol*).

3.2 Código Fuente

El código fuente de un sistema informático constituye el conjunto de líneas de texto que contienen las instrucciones que deben ser procesadas mediante compiladores, ensambladores o intérpretes hacia el código de máquina para su posterior ejecución. Estas instrucciones son escritas en un lenguaje de programación que consiste en un conjunto de símbolos, reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. En diversas ocasiones la calidad de un *software* depende en gran medida de las buenas prácticas que se manifiesten en su proceso de implementación. Para la implementación de la propuesta de solución se tuvo en cuenta los siguientes estándares de codificación.

3.2.1 Estándares de codificación

Los estándares de codificación permiten entender de manera rápida, fácil y sencilla el código empleado en el desarrollo de un *software*. Además facilitan un mantenimiento óptimo de dicho código por parte del programador. Se presenta los estilos de codificación utilizado:

- ✓ La Codificación de caracteres (del inglés, *Character Encoding*) en JSP y XML será UTF-8 (del inglés, *8-bit Unicode Transformation Format*).
- ✓ Los bloques de código siempre deben estar encerrados por llaves, si consta de una línea no es necesario utilizar llaves.
- ✓ Las llaves de apertura de las estructuras de control deben estar en la misma línea, y las de cierre deben ir en la línea siguiente al cuerpo.
- ✓ Los paréntesis de apertura en las estructuras de control no deben tener un espacio después de ellos, y los paréntesis de cierre no deben tener un espacio antes de estos.
- ✓ El estilo Pascal (*PascalCase*) que plantea que la primera letra del identificador y la primera letra de las siguientes palabras concatenadas están en mayúsculas, se utiliza para identificar las entidades.
- ✓ El estilo Camel Case (*camelCase*) que define que la primera letra del identificador debe estar en minúscula y la primera letra de las siguientes palabras concatenadas en mayúscula; se utiliza para identificar los métodos y los parámetros que pudiesen contener estos, las clases y los nombres de los ficheros JSP.

En la Fig. 18 se muestra un fragmento de código del método *calculoMatrizAdy* que hace instancia de llamada la clase *controllerMatriz.java*; donde se evidencia el uso de los estilos de codificación anteriormente explicados.

```
public void calculoMatrizAdy(Asignatura asig){
        List<Indicador> indicadores = dao.listarIndicxAsign(asig);
        float valorSuma = 0;
        float valorAbs = 0;

        for (Indicador indicadorE : indicadores) {
                for (Indicador indicadorS : indicadores) {
                        int indicE = indicadorE.getIdIndicador();
                        int indicS = indicadorS.getIdIndicador();
                        if (indicE != indicS) {
                                dao.insertIndicIndic(indicadorE, indicadorS);
                                IndicadorIndicador idIndicIndic = dao.obtenerIndInd(indicadorE, indicadorS);
                                List<MatrizAdyacenciaAsociada> lista = dao.obtenerMatrizAdyacPor(idIndicIndic);
                                int cantElem = lista.size();
                                for (MatrizAdyacenciaAsociada listado : lista) {
                                        VariableLenguistica listvar = dao
                                                        .obtenerVarLengPorId(listado.getVariableLenguistica().getIdVariableLenguistica());
                                        float var = listvar.getValorNumerico();
                                        valorSuma += var;
                                }
                                if(cantElem!=0)
                                        valorAbs = valorSuma / cantElem;
                                dao.construirMatriz(idIndicIndic, valorAbs);
                                valorSuma = 0;
                                valorAbs = 0;
                        }
                }
        }
}
```

Fig. 18 Fragmento de código fuente del método *calculoMatrizAdy*

Este método es el encargado de calcular los valores absolutos sobre la valoración emitida por los expertos de cada indicador en correlación con otros indicadores, para ir construyendo la matriz de adyacencia de valores absolutos.

3.3 Pruebas del software

Las pruebas del software son un elemento crítico para la garantía de calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación. El objetivo de las mismas es encontrar el máximo número posible de errores con una cantidad manejable de esfuerzo aplicado en un

período realista de tiempo (Pressman 2008). Con las mismas se garantiza que el producto final funcione como fue diseñado e implemente de manera correcta los requisitos identificados.

Para dar inicio a las pruebas de un *software* lo primero es describir una estrategia de prueba donde quede plasmado los niveles de prueba a tratar, así como los tipos de prueba a emplear en cada nivel, los métodos de prueba a aplicar, así como las técnicas a utilizar para cada método. Las estrategias de pruebas describen y verifican el enfoque de la misma. Por lo que se decide aplicar pruebas de unidad y de sistema para medir el cumplimiento del objetivo propuesto en la presente investigación.

3.3.1 Pruebas de unidad

Las **pruebas de unidad** permiten comprobar que el módulo implementado, entendido como una unidad funcional de un programa independiente, está correctamente codificado. Para la generación de casos de prueba de unidad, se decidió utilizar el **tipo de prueba funcional**, aplicándose el método de caja blanca con la utilización de la técnica de camino básico; para probar de la manera más completa posible el módulo implementado.

Método de caja blanca

El **método de caja blanca** del software comprueba los caminos lógicos del software proponiendo casos de prueba que ejerciten conjuntos específicos de condiciones y/o bucles. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado o mencionado.

La técnica de prueba de caja blanca que se propone aplicarse al módulo es la **técnica de camino básico**. Esta técnica permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Para esto, determina la complejidad ciclomática de una porción de código, siendo una métrica del *software* que proporciona una medición cuantitativa de la lógica de un programa. Esta complejidad puede calcularse de tres formas:

1. El número de regiones del grafo de flujo coincide con la complejidad ciclomática.
2. La complejidad ciclomática, V(G), de un grafo de flujo G se define como: V(G) = A-N+2 donde A es el número de aristas del grafo de flujo y N es el número de nodos del mismo.

3. La complejidad ciclomática, V(G), de un grafo de flujo G también se define como V(G) = P+1 donde P es el número de nodos predicado contenido en el grafo de flujo G (Pressman 2008).

Desarrollo de la prueba

Se determinó aplicar este método al código de la aplicación, para este caso a la función que permite seleccionar una asignatura, a un usuario con rol específico entrar sobre la sección correspondiente y realizar sus operaciones. Para la función *"handleActionRequest"* del controlador *"controllerSeleccion"* se identificaron los bloques de ejecución y se enumeraron para identificarlos, mostrándose en la Fig. 19. Se obtuvieron 15 bloques y se determinó el camino básico ilustrado en la Fig. 20, identificando en cada sentencia condicional un nodo predicado del cual se derivan más de un camino a seguir, tal es el caso de los nodos 2, 5, 6, 8 y 11.

```
public ModelAndView handleRenderRequest(RenderRequest request, RenderResponse response) throws Exception {
        Map userMap = (Map) request.getAttribute(PortletRequest.USER_INFO);                    1
        if (userMap == null) {                                                                 2
            return new ModelAndView("errorAutenticado");                                       3
        }
        ThemeDisplay themeDisplay = (ThemeDisplay) request.getAttribute(WebKeys.THEME_DISPLAY);
        String liferayUser = themeDisplay.getRealUser().getFullName();
        List<Role> liferayUserRoles = themeDisplay.getUser().getRoles();
        boolean usuario= false;                                                                4
        int i=0;
        String sms = null;
        String seleccion = request.getParameter("sel");
        String accion = null;
        if(seleccion.equalsIgnoreCase("1")){                                                   5
            while (i <= liferayUserRoles.size() && usuario==false){                            6
                String rol = liferayUserRoles.get(0).getName();                                7
                if(rol.equals("Experto"))                                                      8
                    usuario=true;                                                              9
                i++;                                                                           10
            }
            if(!usuario){                                                                      11
                return new ModelAndView("errorRole");   }                                      12
            sms = "Seleccione que asignatura realizar el Diagnóstico de competencia";
            accion= "correlacion";                                                             13
        }
        String idexperto = (String) userMap.get("liferay.user.id");
        List<Asignatura> asignatura = dao.listarAsign();
        Map<String, Object> aux = new HashMap<String, Object>();
        String path = request.getContextPath();
        String txt=null;
        aux.put("txt", txt);
        aux.put("sms", sms);                                                                   14
        aux.put("accion", accion);
        aux.put("idexperto", idexperto);
        aux.put("asignatura", asignatura);
        aux.put("path", path);
        return new ModelAndView("seleccionAsign", "aux", aux);
} 15
```

Fig. 19 Función que permite seleccionar una asignatura, a un usuario con rol específico y realizar sus operaciones. El código se divide por bloques de ejecución, los cuales están enumerados y constituyen los nodos del camino básico

Con el camino básico determinado, se aplica una de las tres formas para calcular la complejidad ciclomática, se utilizó la fórmula V(G)=A-N+2, para la cual se obtuvo 19 artistas y 15 nodos, por lo tanto:

V(G)=19-15+2, quedando V(G)=6. De la misma forma se pueden comprobar que las otras variantes explicadas de calcular la complejidad ciclomática arriban al mismo resultado.

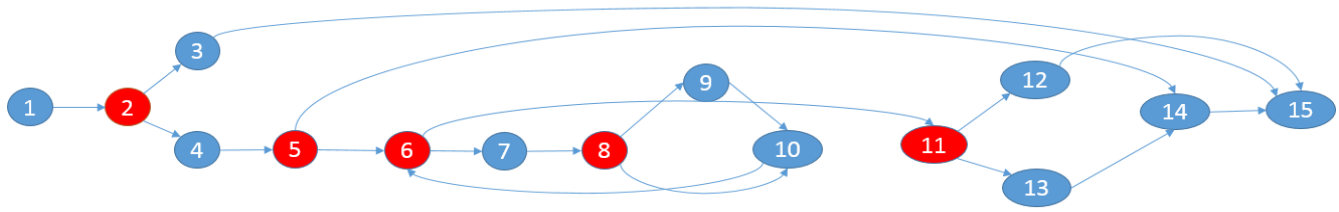


Fig. 20 Camino básico función "handleActionRequest" Los nodos resaltados de color rojo corresponden a las condicionales del código de la función y por tanto son nodos predicados. Las aristas indican los posibles caminos a seguir a partir del nodo correspondiente

El valor de V(G) da el número de caminos linealmente independiente de la estructura de control del programa. En este caso, hay que especificar seis caminos básicos:

Camino 1: 1-2-3-15
Camino 2: 1-2-4-5-14-15
Camino 3: 1-2-4-5-6-11-12-15
Camino 4: 1-2-4-5-6-11-13-14-15
Camino 5: 1-2-4-5-6-7-8-9-10-6-…
Camino 6: 1-2-4-5-6-7-8-10-6-…

Los puntos suspensivos (...) que siguen a los caminos 5 y 6 indican que cualquier camino del resto de la estructura de control es aceptable. A continuación se preparan los CP que forzaran la ejecución de cada camino del conjunto básico. Se muestra la ejecución de uno de ellos:

CP Camino 1

Entrada:

request = representa la solicitud enviada al portlet de manejar un *render*.

response = define un objeto para ayudar a un portlet en el envío de una respuesta al portal.

Resultado esperado

Se obtiene una página de error de autenticación, en este primer camino, si el valor del atributo *request* del portlet es vacío al identificar que no hay existencia de usuario autenticado al sistema.

Es importante darse cuenta de que algunos caminos independientes (caminos 3 y 4) no se pueden probar de forma aislada. Es decir la combinación de datos requerida para recorrer el camino no se puede conseguir con el flujo normal del programa (Pressman 2008). En tal caso, estos caminos se han de probar como parte de otra prueba de camino (debe ser probado los caminos 5 y 6).

3.3.2 Prueba de sistema

Se considera necesario realizar las pruebas a **nivel de sistema**. Este tipo de prueba permite ejercitar profundamente el sistema para verificar que se han integrado adecuadamente todos los elementos del sistema (*hardware, software*) y que realizan las funciones adecuadas. Se debe comprobar que:

- ✓ Se cumplen los requisitos funcionales establecidos.
- ✓ El funcionamiento y rendimiento de las interfaces *hardware*, *software* y de usuario.
- ✓ Rendimiento y respuesta en condiciones límite y de sobrecarga.

Se decide realizar para este nivel el **tipo de prueba funcional, integración y de carga**. Para la generación de los casos de pruebas de tipo funcional e integración se utiliza el método de caja negra aplicándose la técnica de partición equivalente. Para calcular el rendimiento del sistema se realiza las pruebas de carga que permiten determinar cómo la aplicación en su ambiente del lado del servidor responderá ante varias condiciones de carga, haciendo uso de la herramienta JMeter en su versión 2.3.

Método de caja negra

El **método de prueba de caja negra**, también denominada prueba de comportamiento, se centra en los requisitos funcionales del software. O sean, permite obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales del programa.

Para el caso de las pruebas de caja negra, se aplicará la técnica de **partición equivalente**, la cual divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar (Pressman 2008).

Desarrollo de la prueba

El diseño de casos de prueba (DCP) para la partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada. A continuación se muestra un ejemplo de los DCP realizados, específicamente el del CUS1: Gestionar indicador.

DCP del CUS1: Gestionar indicador

- **Descripción general**

El CUS se inicia cuando el usuario selecciona la opción Gestionar indicador y culmina con la realización de una de las acciones: Adicionar, Modificar, Visualizar y Eliminar indicador.

- **Condiciones de ejecución**

El usuario debe estar autenticado en el sistema, así como contar con los permisos para realizar alguna de las operaciones antes mencionadas. En el caso de que se pretenda Modificar, Visualizar o Eliminar algún indicador, este debe haber sido adicionado anteriormente al sistema. En el caso de que se pretenda Adicionar algún indicador, el usuario solo podrá realizar la acción si en el sistema exista al menos registrado una asignatura, en ese caso podrá añadirlo o no ejecutar la acción hasta que se tenga registrado una asignatura.

- **Secciones a probar en el CUS1: Gestionar indicador**

Tabla. 6 DCP Gestionar indicador

Nombre de la sección	***Escenarios de la sección***	***Descripción de la funcionalidad***	***Flujo Central***
SC 1: "Adicionar indicador".	**EC 1.1:** El usuario inserta los datos requeridos de manera correcta.	El sistema adiciona un nuevo indicador.	1. Clic en "Gestionar indicador". 2. Clic en "Adicionar indicador". 3. Inserta datos en los campos. 4. Clic en "Aceptar".
	EC 1.2: El usuario deja campos vacíos.	El sistema muestra un mensaje "El campo está vacío" especificando que debe rellenar el campo que dejo en blanco.	1. Clic en "Gestionar indicador". 2. Clic en "Adicionar Indicador". 3. Inserta datos en los campos. 4. Clic en "Aceptar".

	EC 1.3: El usuario introduce algunos de los datos requeridos de manera incorrecta.	El sistema muestra los siguientes mensajes indicando que existen datos incorrectos: "No puede haber espacio al inicio ni al final", "El campo no debe exceder de 255 letras", "El campo debe ser solo letras" y "El campo debe comenzar con letra mayúscula".	1. Clic en “Gestionar indicador”. 2. Clic en “Adicionar Indicador”. 3. Inserta datos en los campos. 4. Clic en “Aceptar”.
SC 2: “Modificar indicador”.	**EC 2.1:** El usuario modifica los datos de manera correcta.	El sistema actualiza los datos correspondientes al indicador modificado.	1. Clic en “Gestionar indicador”. 2. Se listan todos los indicadores. 3. Clic en el ícono “Editar” del indicador seleccionado. 4. Modifica los datos de los campos. 5. Clic en “Aceptar”.
	EC 2.2: El usuario deja campos a modificar vacíos.	El sistema muestra un mensaje “El campo está vacío” especificando que debe rellenar el campo que dejo en blanco.	1. Clic en “Gestionar indicador”. 2. Se listan todos los indicadores. 3. Clic en el ícono “Editar” del indicador seleccionado. 4. Modifica los datos de los campos. 5. Clic en “Aceptar”.
	EC 2.3: El usuario introduce datos a actualizar incorrectos.	El sistema muestra los siguientes mensajes indicando que existen datos incorrectos: "No puede haber espacio al inicio ni al final", "El campo no debe exceder de 255 letras", "El campo debe ser solo letras" y "El campo debe comenzar con letra mayúscula".	1. Clic en “Gestionar indicador”. 2. Se listan todos los indicadores. 3. Clic en el ícono “Editar” del indicador seleccionado. 4. Modifica los datos de los campos. 5. Clic en “Aceptar”.
SC 3: “Visualizar indicador”.	**EC 3.1:** Muestra detalles de los indicadores.	El sistema muestra los datos asociados a los indicadores registrados.	1. Clic en “Gestionar indicador”. 2. Se listan todos los indicadores.
SC 4: “Eliminar indicador”.	**EC 4.1:** Eliminar un indicador.	El sistema muestra un mensaje de confirmación: "¿Está usted seguro que desea eliminar?" y elimina el indicador seleccionado con todos sus datos y sus correlaciones registrados.	1. Clic en “Gestionar indicador”. 2. Se listan todos los indicadores. 3. Clic en el ícono “Eliminar” del indicador seleccionado. 4. Clic en Aceptar para confirmar la eliminación.

- **Descripción de las variables**

Tabla. 7 Descripción de las variables

No.	*Nombre del Campo*	*Clasificación*	*Valor Nulo*	*Descripción*

1	Descripción del indicador	Campo de texto	No	No puede excederse de 255 caracteres; contenga cualquier combinación de caracteres, excepto numérico; comenzando con letra inicial mayúscula; sin haber espacios al inicio ni al final.
2	Asignatura	Lista desplegable	No	Debe seleccionar una de las asignaturas insertadas con posterioridad al sistema.

- **Matrices de datos**

Adicionar indicador

Tabla. 8 Matriz de datos. Adicionar indicador

Id. Del escenario	Escenario	Descripción	Asignatura	Respuesta del sistema	Resultado de la prueba
EC 1.1	El usuario inserta los datos requeridos de manera correcta.	V[2]/ (Diseño de sistemas de control Automatizado)	V/ (Control Automático)	El sistema adiciona un nuevo indicador.	Satisfactorio
EC 1.2	El usuario deja campos vacíos.	I[3]/ ("")	V/ (Control Automático)	El sistema muestra un mensaje "El campo está vacío" especificando que debe rellenar el campo que dejo en blanco.	Satisfactorio
EC 1.3	El usuario introduce alguno de los datos requeridos de manera incorrecta.	I/ (diseño 2)	V/ (Control Automático)	El sistema muestra los siguientes mensajes indicando que existen datos incorrectos: "No puede haber espacio al inicio ni al final" y "El campo debe ser solo letras".	Satisfactorio

Modificar indicador

[2] V: Entrada Válida

[3] I: Entrada Inválida

Tabla. 9 Matriz de datos. Modificar indicador

Id. Del escenario	Escenario	Descripción	Asignatura	Respuesta del sistema	Resultado de la prueba
EC 2.1	El usuario modifica los datos de manera correcta.	V/ (Diseño de sistemas de control Automatizado)	V/ (Control Automático)	El sistema modifica los datos persistentes de la solicitud.	Satisfactorio
EC 2.2	El usuario deja campos vacíos.	I/ ("")	V/ (Control Automático)	El sistema muestra un mensaje "El campo está vacío" especificando que debe rellenar el campo que dejo en blanco.	Satisfactorio
EC 2.3	El usuario introduce datos a actualizar incorrectos.	I/ (diseño 2)	V/ (Control Automático)	El sistema muestra los siguientes mensajes indicando que existen datos incorrectos: "El campo debe comenzar con letra mayúscula" y "El campo debe ser solo letras".	Satisfactorio

Visualizar indicador

Tabla. 10 Matriz de datos. Visualizar indicador

Id. Del escenario	Escenario	Descripción	Asignatura	Respuesta del sistema	Resultado de la prueba
EC 3.1	Mostrar listado de indicadores.	NA[4]	NA	El sistema muestra los datos de los indicadores insertados en el sistema por asignatura.	Satisfactorio

Eliminar indicador

[4] NA: No Admite

Tabla. 11 Matriz de datos. Eliminar indicador

Id. Del escenario	Escenario	Descripción	Asignatura	Respuesta del sistema	Resultado de la prueba
EC 4.1	Eliminar un indicador.	NA	NA	El sistema muestra un mensaje de confirmación: "¿Está usted seguro que desea eliminar?" y elimina el indicador seleccionado con todos sus datos y sus correlaciones registrados.	Satisfactorio

Resultado de la aplicación de las pruebas

Como parte de la ejecución de las pruebas de caja negra se realizaron 35 casos de pruebas en 3 iteraciones de pruebas para comprobar el funcionamiento del módulo en la Plataforma de SLVD. En la primera iteración se identificaron 19 no conformidades (NC). Una vez corregidas, se procedió a realizar una segunda en la que se identificaron 7 nuevas NC y finalmente se realizó una iteración en la que no se encontraron deficiencias, razón por la que se definió no realizar más iteraciones. Para mayor entendimiento de este proceso se muestra en la Tabla. 12 la descripción de las iteraciones realizadas durante las pruebas y en la Fig. 21 un gráfico de barras que ilustra las NC por cada iteración.

Tabla. 12 Descripción de las iteraciones realizadas durante las pruebas

No. Iteración	Cantidad NC	Tipo Error	Impacto
1	***19***	5 Lenguaje y concordancia	Bajo
		3 Interfaz	Medio
		9 Validación	Alto
		2 Funcional	Alto
2	***7***	4 Validación	Alto
		2 Interfaz	Medio
		1 Ortografía	Bajo
3	***0***	-	-

Las iteraciones permitieron garantizar que el sistema cumpla con las especificaciones que se trazaron.

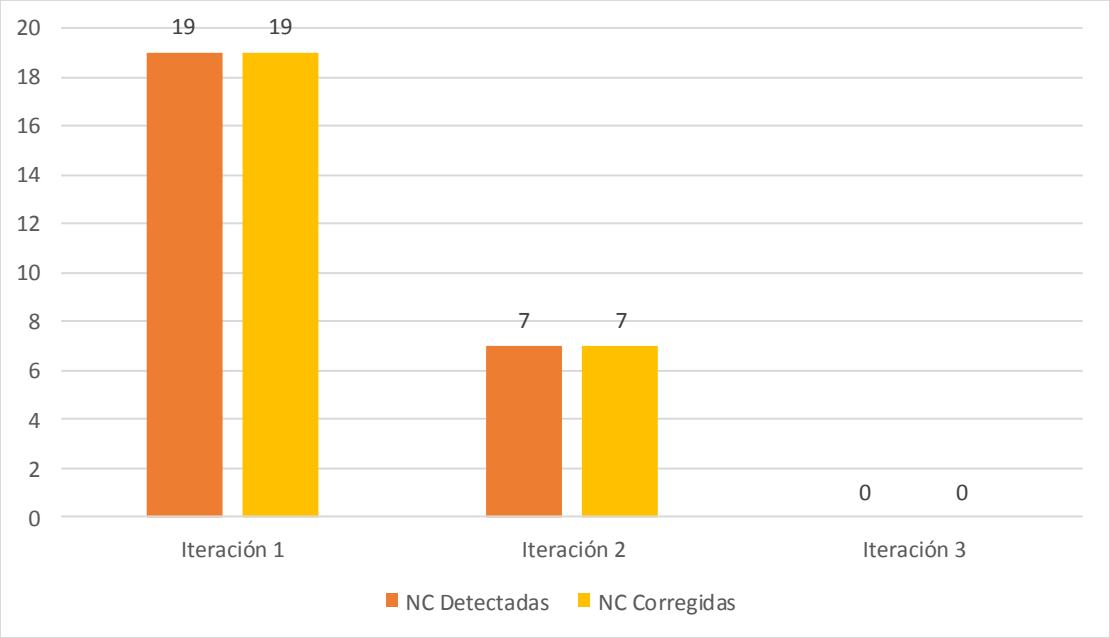


Fig. 21 Resultados de las iteraciones de las pruebas

Desarrollo de la prueba de integración

Luego de realizar las pruebas de caja negra en la Plataforma de SLVD se lograr integrar el módulo (*portlet*) a esta y se pudo comprobar el correcto funcionamiento con los requisitos definidos; aunque cabe aclarar que el módulo puede ser también utilizado por otras aplicaciones indistintamente de las tecnologías usadas en su confección. Para realizar el proceso de integración se siguieron los siguientes pasos:

- ✓ Generar en el Eclipse un archivo WAR[5] (del inglés, *Web Application Archive*) de la aplicación.
- ✓ Desplegar el servidor de la plataforma y esperar a que corra sobre el navegador web.
- ✓ Una vez autenticado con rol de Administrador en el sistema *Liferay Portal*, acceder al Panel de Control de *Liferay*.

[5] WAR: Es una extensión del formato JAR (del inglés, *Java Application Archive*). Es utilizado para transportar las aplicaciones web desarrolladas en Java y poder desplegarlas en cualquier otro contenedor.

- ✓ Una vez dentro del Panel de Control buscar la opción “instalación de complementos” (del inglés, *plugins installation*).
- ✓ Clic en el botón “instalar más complementos” (del inglés, *install more plugins*).
- ✓ Seleccionar subir archivo (del inglés, “*upload file*”).
- ✓ Se busca el archivo con extensión “.war” generado. Luego clic en el botón Instalar (del inglés, *Install*).
- ✓ Si el proceso de subir el archivo se efectuó correctamente se mostrarán dos mensaje con fondo verde claro, ahora el portlet está desplegándose, damos clic en “Ir al *Liferay*” (del inglés, *Back to Liferay*).
- ✓ Se busca el módulo (*portlet*) dentro de los *portlets* de *Liferay*.
- ✓ Se busca la categoría donde se especificó (mediante el fichero “*liferay-display.xml*”) en que categoría se mostraría (en este caso está dentro de la categoría Sample y el nombre del *portlet* es Evaluación de competencia), presionamos en “Añadir” (del inglés, *Add*) y listo.
- ✓ Es importante, cargar el *script* de la BD dentro de alguna herramienta de administración de PostgreSQL, con nombre de BD “diagnostico_competencia”.

Con esto se logra integrar el módulo (*portlet*) implementado a la Plataforma de SLVD y comprobar que funcione como un todo. Se prueban las funcionalidades del módulo para asegurar que los datos que se introducen en la consulta correspondan con los necesarios para el correcto funcionamiento del mismo. Se comprueba que queden validados todos los datos que se introducen.

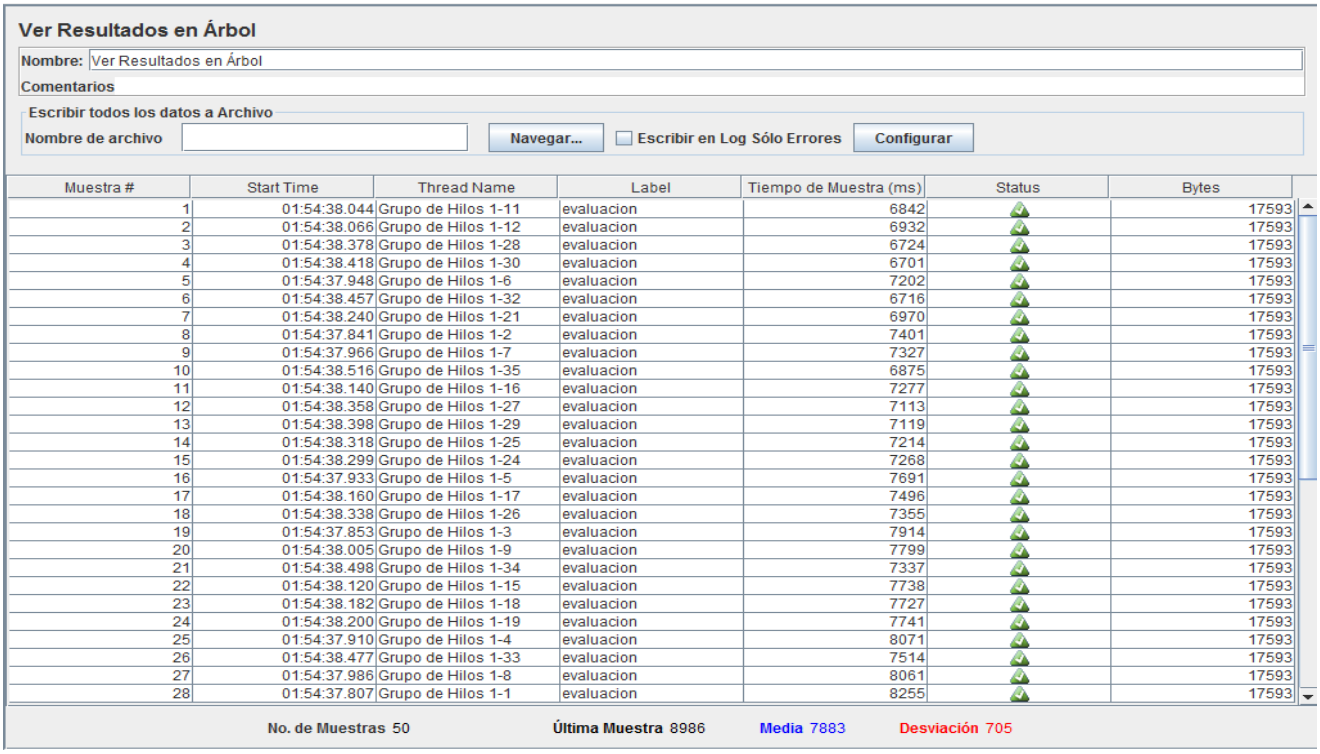
Después de concluido el proceso de integración del portlet a la Plataforma de SLVD, se obtuvo como resultado una satisfactoria instalación del mismo, mostrándose en la Fig. 22 la interfaz principal.

Fig. 22 Interfaz del módulo una vez integrado a la Plataforma de SLVD

Se muestra la interfaz inicial del módulo implementado. Esta posibilita gestionar: asignaturas e indicadores, realizar la evaluación de competencias por parte de los expertos; además de gestionar las preguntas de los cuestionarios por asignaturas y evaluar al practicante.

Desarrollo de la prueba de carga

Para la realización de las pruebas de carga se utilizó la herramienta JMeter 2.3, luego de realizarse las pruebas de caja negra en la Plataforma de SLVD, la cual resultó que al estar conectados 50 usuarios simultáneamente el tiempo de respuesta del sistema está entre los 7 y 8 segundos aproximadamente.



Ver Resultados en Árbol

Nombre: Ver Resultados en Árbol

Comentarios

Escribir todos los datos a Archivo

Nombre de archivo | Navegar... | Escribir en Log Sólo Errores | Configurar

Muestra #	Start Time	Thread Name	Label	Tiempo de Muestra (ms)	Status	Bytes
1	01:54:38.044	Grupo de Hilos 1-11	evaluacion	6842		17593
2	01:54:38.066	Grupo de Hilos 1-12	evaluacion	6932		17593
3	01:54:38.378	Grupo de Hilos 1-28	evaluacion	6724		17593
4	01:54:38.418	Grupo de Hilos 1-30	evaluacion	6701		17593
5	01:54:37.948	Grupo de Hilos 1-6	evaluacion	7202		17593
6	01:54:38.457	Grupo de Hilos 1-32	evaluacion	6716		17593
7	01:54:38.240	Grupo de Hilos 1-21	evaluacion	6970		17593
8	01:54:37.841	Grupo de Hilos 1-2	evaluacion	7401		17593
9	01:54:37.966	Grupo de Hilos 1-7	evaluacion	7327		17593
10	01:54:38.516	Grupo de Hilos 1-35	evaluacion	6875		17593
11	01:54:38.140	Grupo de Hilos 1-16	evaluacion	7277		17593
12	01:54:38.358	Grupo de Hilos 1-27	evaluacion	7113		17593
13	01:54:38.398	Grupo de Hilos 1-29	evaluacion	7119		17593
14	01:54:38.318	Grupo de Hilos 1-25	evaluacion	7214		17593
15	01:54:38.299	Grupo de Hilos 1-24	evaluacion	7268		17593
16	01:54:37.933	Grupo de Hilos 1-5	evaluacion	7691		17593
17	01:54:38.160	Grupo de Hilos 1-17	evaluacion	7496		17593
18	01:54:38.338	Grupo de Hilos 1-26	evaluacion	7355		17593
19	01:54:37.853	Grupo de Hilos 1-3	evaluacion	7914		17593
20	01:54:38.005	Grupo de Hilos 1-9	evaluacion	7799		17593
21	01:54:38.498	Grupo de Hilos 1-34	evaluacion	7337		17593
22	01:54:38.120	Grupo de Hilos 1-15	evaluacion	7738		17593
23	01:54:38.182	Grupo de Hilos 1-18	evaluacion	7727		17593
24	01:54:38.200	Grupo de Hilos 1-19	evaluacion	7741		17593
25	01:54:37.910	Grupo de Hilos 1-4	evaluacion	8071		17593
26	01:54:38.477	Grupo de Hilos 1-33	evaluacion	7514		17593
27	01:54:37.986	Grupo de Hilos 1-8	evaluacion	8061		17593
28	01:54:37.807	Grupo de Hilos 1-1	evaluacion	8255		17593

No. de Muestras 50 | Última Muestra 8986 | Media 7883 | Desviación 705

Fig. 23 Resultados de las pruebas de carga

El resultado de esta prueba se presenta en la Fig. 23, donde muestra las 28 primeras peticiones de ejemplo, el tiempo de respuesta en milisegundos por cada una en la columna “Tiempo de Muestra (ms)”, los errores detectados por la herramienta durante el proceso de petición y respuesta, en la columna “Status”, el tiempo medio de respuesta para la muestra de 50 peticiones simultáneas (Media) y el valor de la desviación entre el mayor y el menor tiempo de respuesta (Desviación).

Conclusiones parciales

En este capítulo se construyó el modelo de implementación correspondiente al módulo para la evaluación de competencias; se obtuvo como resultado del proceso de ingeniería, la construcción del diagrama de componentes para el CUS “Generar mapa cognitivo difuso” por ser el más relevante, definiéndose los componentes del sistema necesario para el correcto funcionamiento de este CUS. Con la elaboración del diagrama de despliegue, permitió una compresión de los recursos necesarios para el despliegue y correcto funcionamiento de la solución. La descripción del empleo de los estándares de codificación y los

estilos de programación permitieron a su vez hacer más fácil el entendimiento del código del programador y facilitar el mantenimiento futuro del módulo.

Finalizada la implementación del módulo se realizó las pruebas a nivel de unidad y de sistema; logrando probar de la manera más completa posible la solución y el cumplimiento de los requisitos funcionales y no funcionales trazados durante la presente investigación. Con las pruebas de sistema se garantizó que el módulo se adaptara al ambiente para el cual fue concebido permitiendo la evaluación de competencias en la Plataforma de SLVD, aunque cabe resaltar que el mismo puede ser utilizado por otras aplicaciones indistintamente de las tecnologías usadas en su confección.

Conclusiones Generales

Como resultado de la investigación se logró implementar un módulo para la evaluación de competencias e integrarlo en la Plataforma de SLVD para las ejecuciones de prácticas de laboratorios; por lo que se llega a las siguientes conclusiones:

- ✓ El análisis de las bibliografías existentes sobre la evaluación de competencias en SLVD demostró que las soluciones no se ajustan a las necesidades y características del negocio, aunque sirvieron para la selección de los principales requisitos funcionales y no funcionales de la misma.
- ✓ Durante el proceso de desarrollo se obtuvieron y especificaron los requisitos a tener en cuenta para el cumplimiento del objetivo de la investigación, así como toda la documentación asociada al proceso.
- ✓ Se destaca la utilización del patrón arquitectónico MVC y los patrones de diseño: GRASP y GoF donde fueron aplicados en la realización de los diagramas de clase de diseño y del modelo del diseño, obteniendo un sistema separado en tres elementos fundamentales: el modelo, la vista y el controlador.
- ✓ Las tecnologías utilizadas y la arquitectura definida en el desarrollo del módulo favorecen la continuidad del resultado alcanzado, permitiendo la integración a la Plataforma de SLVD o puede ser utilizado en otros contextos.
- ✓ La realización de las pruebas de software: unidad y de sistema, permitieron comprobar el cumplimiento de los requisitos funcionales y no funcionales en la solución, así como la correcta integración del módulo a la Plataforma de SLVD.

Referencias bibliográficas

ALBARRAN, J. F. VLabQ Simulador de Experimentos Químicos. In. Pte. 202-2 Col. Universidad. Toluca Estado de México.: SIBEES SOFT S.A. de C.V., 2010.

ALEX. Spring 3 - Parte 1: Introducción. In., 2010.

ALMEIRA, A. S. AND V. P. CAVENAGO. Arquitectura de Software: Estilos y Patrones Universidad Nacional De La Patagonia San Juan Bosco. Argentina, 2007.

ÁLVAREZ, F. Introducción al Middleware. Portales. In., 2011.

ÁLVAREZ, R. S. Técnicas e instrumentos de evaluación de competencias 2009.

ALLAMARAJU, S., C. BEUST AND J. DAVIE *Programación Java Server con J2EE Edicion 1.3*. Edtion ed., 2006. 1206 p.

ARROYO, A. F. AND M. D. L. R. PEDRAZA. Diseño de Laboratorios virtuales para el Bachillerato a Distancia de la UANL: una propuesta. In A. NÚMERO 6. 2011.

BALDUINO, R. *Introduction to Open UP*. Edtion ed., 2007.

BASS, L., P. CLEMENTS AND R. KAZMAN *Software Architecture Practice Third Edition*. Edtion ed., 2015.

BENAVIDES, G. A. M. AND C. E. O. MORALES. Laboratorio Virtual Basado en la Metodología de aprendizaje basado en problemas, ABP. In., 2009.

CORNEJO, J. E. G. ¿Qué es UML? El Lenguaje de Modelado Unificado. In., 2008.

CORNEJO, J. E. G. ¿Cuáles son las características que debe tener una herramienta UML? In., 2009.

GIORDANO, R. AND M. VURRO Fuzzy cognitive map to support conflict analysis in drought management fuzzy cognitive maps. M. Glykas. Fuzzy cognitive maps. Studies in fuzziness and soft computing. 247, 2010, 403-425.

GLYKAS, M. AND P. GROUMPOS *Fuzzy Cognitive Maps: Basic Theories and Their Application to Complex Systems Fuzzy Cognitive Maps*. Edtion ed., 2010. 1-22 p.

GLYKAS, M., G. XIROGIANNI AND G. STAIKOURAS *Fuzzy Cognitive Maps in Banking Business Process Performance Measurement*. Edtion ed., 2010.

HIBERNATE. Hibernate ORM. Idiomatic persistence for Java and relational databases. In., 2015.

IGLESIAS, M. R. *La Evaluación por Competencias* Edtion ed. Monterrey, 2010.

JACOBSON, I. *El proceso unificado de desarrollo de software*. Edtion ed., 2000.

JONAS, J. *Liferay Portal Enterprise Intranets. Birmingham*. edited by E.E.U.U.P. PUBLISHING. Edtion ed., 2012. ISBN 978-1-847192-72-1.

KHAMIS, R. Interacción Remota con Robots Móviles Basada en Internet. Universidad Carlos III de Madrid, 2006.

LARMAN, C. *UML y Patrones. 2da Edición*. Edtion ed. México: Prentice Hall, 2003.

LIN, C. AND C. LEE Neuralnetwork- based fuzzy logic control and decision system 2002, 40(IEEE), 1320-1336.

LOBOS, M. E. Aprende a programar. Capítulo 4: Concepto de lenguaje de programación. In., 2005.

LÓPEZ, E. T., A. O. RAMON, E. M. SARROCA AND C. G. SEONE *Diseño de sistemas software en UML*. edited by C.U.P. 8498800757. Edtion ed., 2004.

MALDONADO, M. Á. G. *Currículo con enfoque de competencias*. edited by C. EDIC. ECOE. BOGOTÁ. Edtion ed., 2010.

MAR, O. Modelado mediante mapa cognitivo difuso para la evaluación de competencias en un Sistema de Laboratorios Virtual y a Distancia. VIII Simposio de Ingenieria Industrial y Afines, 2014.

MARTÍNEZ, R. Sobre PostgreSQL. In., 2010.

MENÉNDEZ, R. AND B. ASENSIO. Informática Aplicada a la Gestión Pública. Facultad Derecho UMU. Capítulo 2. Ingeniería del software. Metodologías de desarrollo. In. Departamento Informática y Sistemas. Universidad de Murcia, 2013.

MOHAMMED, K. J. Apache Server 2 Bible [online]. 2010. Available from World Wide Web:<https://yexia.files.wordpress.com/2010/09/mohammed-j-kabir-la-biblia-del-servidor-apache-21.pdf>.

NÁJERA, J. M. La evolución de los laboratorios virtuales durante una experiencia de cuatro años con estudiantes a distancia 2007a.

NÁJERA, J. M. Ventajas y desventajas de usar laboratorios virtuales en educación a distancia: la opinión del estudiantado en un proyecto de seis años de duración. Revista Educación, 2007b, 31(1), 91-108.

NODARSE, F. A. F., E. P. GONZÁLEZ AND F. F. LIMA Laboratorios virtuales en la Universidad virtual del CITMA 2008.

PEÑA, A., H. SOSSA AND A. GUTIÉRREZ Mapas Cognitivos: un Perfil y su Aplicación al Modelado del Estudiante. Centro de Investigación en computación, IPN, 2007, 10, 230-250.

PÉREZ, I. T. Los portlets en la construcción de portales web: OracleWebCenter. Convención Científica de Ingeniería y Arquitectura, 2008, 14, 1-4.

PÉREZ, J. E. *Introducción a JavaScript*. edited by AUTOEDICIÓN. Edtion ed., 2009.

PRESSMAN, R. *Ingeniería del Software. Un enfoque práctico. Sexta Edición*. Edtion ed., 2008.

REDINERTHO. Ventajas en la utilización de Eclipse. In., 2012.

ROSADO, L. New teaching contributions of virtual and remote laboratories in teaching physics. In *Recent Research Developments in Learning Technologies.* Portugal, 2009.

RUBIO, L. F. *Desarrollo de competencias en educación básica*. edited by D. COYOACÁN. Edtion ed. México, 2010.

RUIZ, J. G. AND C. J. B. COLUNGA *Preparación Pedagogica Integral para Profesores Universitarios*. edited by F. VARELA. Edtion ed. La Habana, 2010.

S., P. R. *Ingeniería de Software, un enfoque práctico. 2002*. Edtion ed. McGraw-Hill Companies, 2002.

SALMERON, J. Supporting decision makers with Fuzzy Cognitive Maps. Industrial Research Institute, 2010, 52, 53-59.

SÁNCHEZ, B. AND Y. VALDÉS Diseño de Sistemas de Información Documental. Consideraciones teóricas. Ciencias de la Información, 2010, 39 Nº 3.

SANTOS, A. C. *Gestión de Competencias*. Edtion ed., 2001.

SANTOS, I. Modelo de gestión de información digital agraria cubana. Ciencias de la Información, 2013, 44 Nº2.

SARTORIUS, A. Platform for distance development of complex automatic control strategies usin Matlab. The Internacional Journal of Engineering Education, special issue on Matlab and Simulink in Engineering Education, 2005, 790-797.

SCRIBD Herramientas CASE 2014.

SOMMERVILLE, I. *Ingeniería de software. Séptima edición*. Edtion ed., 2005.

STACH, W., L. KURGAN AND W. PEDRYCZ *Expert-Based and Computational Methods for Developing Fuzzy Cognitive Maps*. edited by I.M.G. (ED.). Edtion ed. Berlin: Springer, 2010. 23-41 p.

TABARES, Z. E. AND D. S. LÓPEZ Aplicación web para la realización de estudios farmacocinéticos, versión 2.0. Revista Cubana de Informática Médica. Habana, Cuba, 2013, 5.

TOBÓN, S. *Competencias, calidad y educación superior*. Edtion ed. Bogotá, Colombia, 2009.

TOMCAT, A. The Apache Software Foundation. Apache Tomcat 7. In., 2015.

WEITZENFELD, A. Ingeniería de software orientada a objetos con UML, Java e Internet. Cengage Learning Editores, 2005.

Anexos

Anexo 1: Entrevista a Jefes de Departamentos y Jefes de Asignaturas

Soy estudiante de la carrera de Ingeniería en Ciencias Informáticas de la UCI, realizo un proyecto de investigación del que usted forma parte del objeto de estudio. Considero que tiene información valiosa para el desarrollo de esta investigación, por lo que solicito su colaboración y le informamos el carácter confidencial de sus respuestas.

Gracias de antemano.

Objetivo: Diagnosticar el estado actual de la utilización de los SLVD en el PEA de la Universidad Central de Las Villas.

1. ¿Cómo valora el empleo de los SLVD en las condiciones actuales en el PEA?
2. ¿Existe algún mecanismo para evidenciar por parte de los profesores al supervisar las estrategias de control diseñadas por los estudiantes en la práctica de laboratorios?
3. ¿Ha existido en ocasiones un mal funcionamiento en las estaciones de trabajo de los SLVD?

   Sí ___ ¿Han podido encontrar el porqué de ese mal funcionamiento? No ___

4. ¿Sabe Ud. cómo identificar si algún usuario que acceda al LV es o no competente haciendo uso del mismo?

   Sí ___ ¿Qué mecanismos utilizan para verificar que el usuario es o no competente? No ___

Anexo 2: Encuesta a profesores y metodólogos

Soy estudiante de la carrera de Ingeniería en Ciencias Informáticas de la UCI, realizo un proyecto de investigación del que usted forma parte del objeto de estudio. Considero que tiene información valiosa para el desarrollo de esta investigación, por lo que solicito su colaboración y le informamos el carácter confidencial de sus respuestas.

Gracias de antemano.

Objetivo: Diagnosticar el estado actual de la utilización de los SLVD en el PEA de la Universidad Central de Las Villas.

1. ¿Cómo valora el empleo de los SLVD en las condiciones actuales en el PEA?

   __ Sin importancia __ Importante __ Poco importante _ Muy importante

2. ¿Existe algún mecanismo para evidenciar si un profesor supervisa las estrategias de control diseñadas por los estudiantes en la práctica de laboratorios?

   Sí ___ No ___

3. ¿En ocasiones ha existido un mal funcionamiento en las estaciones de trabajo de los LV?

   Sí ___ No ___ ¿Sabe Ud. el porqué de ese mal funcionamiento? _____

4. ¿Sabe Ud. cómo identificar si algún usuario que acceda al LV es o no competente haciendo uso del mismo?

   Sí ___ No ___ ¿Qué mecanismos se utilizan para verificar que el usuario es o no competente?