Universidad de las Ciencias Informáticas,

Facultad 1

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Título: Sistema evaluador de habilidades y de recomendación para el apoyo al proces de enseñanza-aprendizaje de la Ingeniería de Software

Autor(es): Ileana Naya Díaz

Karen Oliva Rondón

Tutor(es): MSc. Osiris Perez Moya

MSc. Radel Calzada Pando

La Habana, Junio de 2015

"Año 57 de la Revolución"

DECLARACIÓN DE AUTORÍA

Se declara que somos los únicos autores de este trabajo y autorizamos a la Facultad 1 de la Universidad de Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste se firma la presente declaración jurada de autoría en La Habana a los ____ días del mes de _____ del año _____.

Autoras:

_____ _____

Ileana Naya Díaz Karen Oliva Rondón

Tutores:

_____ _____

MSc. Osiris Perez Moya MSc. Radel Calzada Pando

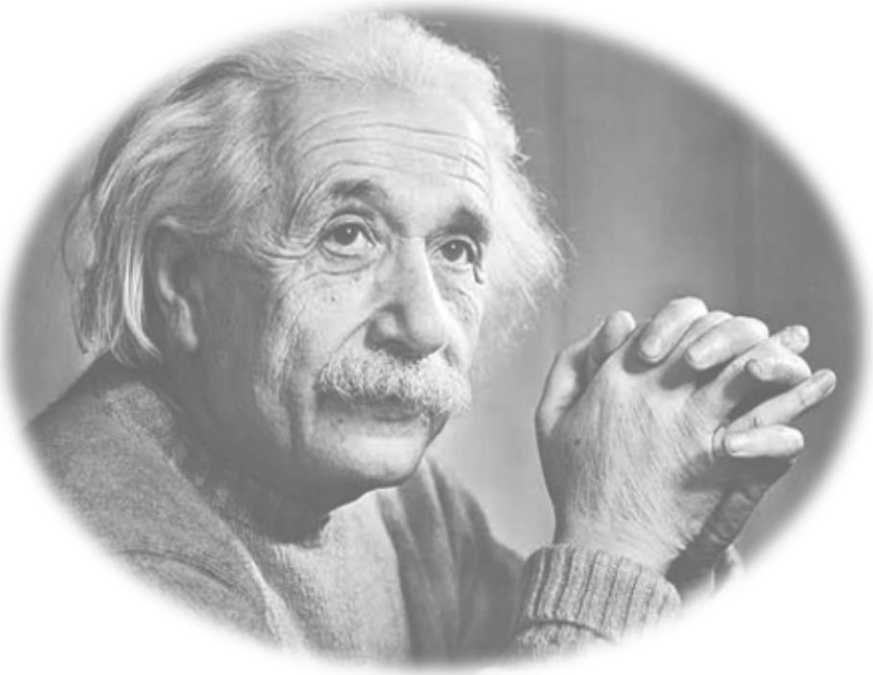
DATOS DE CONTACTO

Tutor: MSc. Osiris Perez Moya. Graduado como Ingeniero en Ciencias Informáticas por la Universidad de las Ciencias Informáticas en julio de 2007. Desde su graduación ha estado vinculado a la actividad productiva en la universidad ocupando varios roles (Jefe de proyecto, analista, asesor de calidad, probador). Su vinculación docente comienza en agosto del 2007, hasta la fecha, en la disciplina de Ingeniería y Gestión de Software, ha ocupado otras responsabilidades políticas y administrativas que van desde jefe de asignatura hasta subdirector de investigación y postgrado de un centro de desarrollo de *software* adscripto a la red de centros de la UCI. Actualmente se desempeña como jefe del Departamento Docente de Ingeniería de Software y Práctica Profesional de la Facultad 1. En enero de 2013 culminó sus estudios para obtener el título de Máster en Gestión de Proyectos Informáticos. Su actividad investigativa se resume a más de 20 publicaciones obtenidas en revistas, libros y memorias de eventos científicos.

Correo electrónico: operez@uci.cu **Teléfono:** 07-835-8793

Tutor: MSc. Radel Calzada Pando. Graduado como Ingeniero en Ciencias Informáticas por la Universidad de las Ciencias Informáticas en 2008. Desde la culminación de los estudios universitarios ha estado vinculado a la actividad docente en la universidad, trabajando como profesor en las disciplinas de Gestión de Software, Base de datos y Componente profesional de Ingeniería y Gestión de Software; desde la propia fecha se ha desempeñado como tutor y oponente de tesis de grado. Ha recibido 29 cursos de postgrado. Actualmente se desempeña como Profesor Asistente en la Facultad 1 de la Universidad de las Ciencias Informáticas y Asesor del vicerrector económico de la propia universidad. Culminó sus estudios para obtener el título de Máster en Gestión de Proyectos Informáticos, Gestión de riesgos. Su actividad investigativa se resume a 9 publicaciones obtenidas en revistas, libros y memorias de eventos científicos. Ha obtenido más de 14 premios y reconocimientos en importantes eventos.

Correo electrónico: rcalzada@uci.cu **Teléfono:** 05-337-2954



"Hay una fuerza motriz más poderosa que el vapor,

la electricidad y la energía atómica: la voluntad".

Albert Einstein

AGRADECIMIENTOS

Ileana

Me gustaría agradecer, en primer lugar a mis padres, por haberme apoyado a lo largo de toda la vida y, sobre todo, a través de esta etapa tan convulsa, llena de proyectos, aspiraciones, dudas, inconvenientes y motivaciones; que suele ser la juventud. El amor y la experiencia de mis padres me ha llevado por el buen camino de la perseverancia, el sacrificio y la determinación, permitiéndome alcanzar todas las metas que me he propuesto. La dicha de crecer en una familia muy unida y feliz, me enseñó a mirar siempre el lado positivo de las cosas y a sentirme orgullosa de todo cuanto consigo. Ellos han sido la razón principal por la que me he mantenido con el ánimo siempre en lo más alto y gracias al cual he conseguido cumplir todos mis objetivos. Sé de sobra, que no tengo que agradecer su preocupación ni su incondicional apoyo, y mucho menos su deseo de verme crecer en la vida; pero siento que la fecha es propicia para agradecer las cosas que únicamente ustedes significan para mí.

Agradezco a mi hermana Anaelys por brindarme siempre el apoyo necesario. Por ser la persona con la que puedo contar sin importar de qué asunto se trate. Porque sin ti, el paso por la vida habría sido absolutamente diferente. Porque nuestro carácter, diametralmente opuesto, ha hecho que precisemos una de la otra y encontremos nuestra fortaleza en la propia debilidad.

Agradezco a mi hermano Carlos por transmitirme sus experiencias en los primeros años de la carrera.

Agradezco a mi primo Alejandro por figurar entre las personas que alimentan y comparten mis sueños. Tu elocuencia, tu alegría perenne y tu espíritu rebelde hacen que te admire muchísimo.

Agradezco al resto de mi familia por estar siempre al tanto de todos mis pasos en el transitar escolar.

Me gustaría agradecer a mis tutores Osiris y Radel por haber sido profesores magistrales durante la carrera y por la dedicación prestada durante la realización de este trabajo de diploma. De igual manera agradezco a mi compañera de tesis Karen por impulsar juntas este proyecto decisivo en la vida de ambas, por ser partícipe de este viaje en el tiempo. Por haber colaborado para que estos cinco años hayan valido la pena y este gran sueño hoy sea realidad.

Agradecimientos

Muchas gracias también a todos los compañeros de brigada por orientarme y ayudarme de buena voluntad siempre que lo he necesitado.

Agradezco a aquellos profesores que se convirtieron en amigos y a aquellos amigos que fueron profesores en materia de sentimientos.

Agradezco a mi amigo Alejandro Rodríguez Toledo por estar cuando conté con pocos, por su preocupación y disposición para conmigo tanto en los estudios como en la salud.

Agradezco, a los amigos que brindaron su apoyo sin estar aquí, a los que estuvieron poco tiempo y me dejaron lecciones y a los que me han acompañado a lo largo de la carrera, sobre todo a esos compañeros que se convirtieron en grandes amigos, en familia sin parentesco: Angel Antonio Sánchez López-Castro, Ivis Rosa Madrigal Leiva, Yennifer Delgado Mesa y Yoandri Pablo Martínez Magaña, a ti Yoa, te agradezco todas las horas de dedicación a este proyecto, esas horas contribuyeron en gran medida a la alegría que estoy experimentando hoy. Desde el primer momento me he sentido como en casa, y estoy orgullosa de poder decir que encontré aquí a personas extraordinarias que hicieron de la estancia en la Universidad un suceso maravilloso. De corazón, muchas gracias por haber corrido este gran maratón a mi lado, de principio a fin.

Termino ya una preciosa etapa llena de felicidad, de momentos de incertidumbre, de exámenes, de clases, pero sobre todo de pasarla bien, que recordaré siempre con mucho cariño.

A los amigos que cumplen junto conmigo el sueño de ser ingenieros "Felicidades", a los que aún no terminan "Póngale ganas"; y a todos "Gracias" por haberme acompañado en mi paso por la Universidad de las Ciencias Informáticas.

Karen

Doy gracias a mis padres por haber hecho de mi lo que hoy soy, porque han estado y estarán a mi lado siempre que los necesite. Por depositar en mi toda su confianza y por creer que yo sí podía cuando el mundo entero dijo que no. Por llenarme de tanto amor y cariño, por haberme mimado tanto y haberme dado todo y más de lo que podían.

Gracias mamita por esos cariñitos que siempre me das, por ser tu niñita linda, TE AMO MAMI.

Gracias papito, porque a pesar de no ser el padre más cariñoso del mundo, eres el mejor del universo, TE AMO PAPI.

Gracias a los dos por ser mis motores impulsores y mis guías a seguir. Gracias y mil gracias más por ayudarme a construir mis sueños y por apoyarme toda mi vida para cumplir todas mis metas trazadas, LOS AMO.

A esa persona que ha estado a mi lado estos 5 años, en las amarguras y las alegrías. A mi amigo, compañero y novio. Gracias por aceptarme tal cual soy. Gracias por "comprenderme" cuando estoy en mis peores momentos, por aguantar todas mis malacrianzas y tratar de complacerme y darme todos los gustos, que bien difícil es. Gracias por haberlo dejado todo por estar conmigo. Simplemente TE AMO BEBÉ.

A mi pequeño cachorro Scooby que se ha convertido en una de las mayores alegrías de mi hogar, por brindarme su cariño y animar mis días cuando más triste estoy. TE QUIERO MUCHO MI CHIQUITO PRECIOSO.

A mi familia en general. Gracias a todos mis tíos y primos. Especialmente a mis tíos Teresa, Nilsa, Gerardo, Milagro y Modesto. A mis primos Melissa, Irianni, B. Alejandro, Naila y en especial a eso primos qué son como hermanos: Yordanis y Pedro, a mi hermano Yuniesky y a mi cuñada Odett. Gracias a la familia de mi novio: misuegra Carmita, mi suegro Luis, mi cuñada Yaima y la pequeña no tan pequeña ya Sintia; que a pesar de que no estén aquí, me han brindado todo su apoyo para que mis resultados sean los mejores. Gracias a todos por preocuparse por mí, por tenerme siempre presente y porque siempre estarán en mi corazón. LOS QUIERO.

Gracias a mis amigos del tecnológico y de toda la vida; Ariel, Yennifer, Ursinio y Abel; porque a pesar de que no hayan podido estar a mi lado estos 5 años, en la distancia han sabido seguir siendo buenos amigos. LOS QUIERO A TODOS.

Agradecimientos

Gracias a Ileana por ser la mejor compañera de tesis que pudiera haber tenido, por tu esfuerzo y dedicación todos estos meses. Gracias por haberme brindado tu amistad durante mi paso por la universidad. Gracias por entenderme en los momentos más difíciles de esta dura etapa. TE QUIERO, Gracias.

A mis tutores como muestra de mi gratitud y eterno reconocimiento. Especialmente a Osiris porque a pesar de haber sido tutor y maestro te has convertido en amigo, GRACIAS.

Gracias a una personita súper importante que se ha convertido en uno de los pocos amigos que llamo así, AMIGOS, a ti Yoandri Pablo Martínez Magaña, porque gran parte de nuestro resultado se debe a tu gran apoyo y ayuda en el cumplimiento del presente trabajo. GRACIAS AMIGO.

A mis compañeras de apartamento durante la carrera y a los que siempre estaban de agregados (Neyvis, Yisel, Yanet, Lisdania, Yennis, Ivis, Yennifer, Adachely, Arlety, Leonar, Manuel, Danilo, Arián, Roger).

Gracias especialmente a mi enana preciosa Lisdy; de ella aprendí que los mejores perfumes vienen en frascos pequeños. Gracias por ser tan grande mi pequeña gigante, siempre estaré ahí para ti; y mi familia también. Gracias al negro más fashion de la UCI alias Leonar, gracias mi negro por invadir mi privacidad y mi apartamento en todo momento, te quiero. Gracias al cuerpazo de la UCI, Beyoncé alias Yennis por estar ahí para mí. Gracias a la niña nueva alias Adachely, porque a pesar de no haber estado tanto tiempo conmigo, creaste tu lugarcito en mi corazón.

Gracias a todas esas, personas ya sean estudiantes, profesores o trabajadores que se convirtieron en amigos en el decursar de mis años aquí, en especial: Serguey, Damián, Leyanis, Alién, Sahilyn, Javier, Frank, Rosario, Yorle, Jacob, Pepito... Gracias a todos porque siempre están ahí. Mil gracias.

Gracias a mis compañeros de aula actuales y a los que por alguna razón nos tuvieron que dejar como es el caso de Maikel, Javier, Yadriel y otros, gracias a ustedes también porque han sido una gran muestra para todos de valor y de sacrificio.

Gracias a mis profesores que me formaron y educaron para ser una profesional abnegada.

A TODOS AQUELLOS QUE DE UNA FORMA U OTRA SIEMPRE HAN PUESTO SU GRANITO DE ARENA EN MI FORMACIÓN COMO PERSONA, SER HUMANO Y PROFESIONAL.

GRACIAS DESDE LO PROFUNDO DE MI CORAZÓN, GRACIAS.

DEDICATORIA

Ileana

Luego de todos los sacrificios, el trabajo duro, la larga espera, el apoyo y la "luz" que mis padres me han brindado durante todos estos años; no hay nada más gratificante que dedicarles a ellos este logro.

Karen

El resultado de este Trabajo de Diploma se lo dedico a las dos personas más importantes en mi vida, a ellos, que confiaron en mi cuando creía que era imposible, A MIS AMORES, los adoro MAMI y PAPI.

A mis preciosos pequeñines, los bebés más lindos, estudiosos y amorosos, mis sobrinos José Carlos González Ginarte y Christopher Oliva Hernández. Porque pretendo que sigan mis pasos y ser una de sus mayores motivaciones para el futuro, los quiero mis niños.

A mi compañera de tesis Ileana Naya Díaz (Ilo), porque el fruto de este trabajo se debe también a su esfuerzo y dedicación.

RESUMEN

El desarrollo de las Tecnologías de las Información y las Comunicaciones (TIC), ha tenido gran influencia en las diversas esferas sociales, destacándose su uso en la educación, incorporando elementos novedosos que han revolucionado los antiguos métodos de enseñanza-aprendizaje. Actualmente, la Universidad de las Ciencias Informáticas (UCI), se encuentra inmersa en el perfeccionamiento del proceso de formación de los estudiantes, considerando el estado cognitivo de los mismos, para proporcionar la ayuda pedagógica adecuada.

El objetivo del presente trabajo es desarrollar un Sistema evaluador de habilidades y de recomendación para apoyo al proceso de enseñanza-aprendizaje, que ayude a consolidar los contenidos del plan de estudio definido para la disciplina de Ingeniería de Software (ISW).

Para la realización del sistema se emplearon los *frameworks Symfony2* en su versión 2.3.7 y *Bootstrap* en su versión 3 para el diseño de las interfaces de usuario, *PostgreSQL* versión 9.1 como Sistema Gestor de Bases de Datos para los datos administrativos. Además, se expone el proceso de desarrollo siguiendo la metodología *OpenUp* a partir de la cual se generaron un conjunto de artefactos que capturan la información relacionada con la solución. Para la validación del sistema se realizaron varias pruebas funcionales, pruebas de rendimiento y pruebas de seguridad, las cuales arrojaron un conjunto de no conformidades que fueron resueltas en iteraciones posteriores.

El sistema desarrollado será de beneficio para estudiantes y profesores ya que permitirá evaluar habilidades y recomendar ejercicios a los estudiantes, considerando su estado cognitivo; y permitirá a los profesores realizar un seguimiento del desarrollo de los estudiantes mediante un resumen de evaluaciones.

Palabras claves: evaluación, proceso de enseñanza-aprendizaje, recomendación, Sistema evaluador de habilidades y de recomendación para el proceso de enseñanza-aprendizaje.

ÍNDICE

Introducción……1

Capítulo 1: Marco teórico conceptual del Sistema evaluador de habilidades y de recomendación para el apoyo al proceso de enseñanza-aprendizaje……..…7

1.1 Introducción……7

1.2 Definición del marco conceptual de la investigación……7

1.3 Tendencias actuales en el desarrollo de Sistemas Tutores Inteligentes……..…9

1.3.1 Sistemas Tutores Inteligentes. Análisis de soluciones homólogas 9

1.3.2 Sistemas basados en el conocimiento 11

1.4 Análisis de sistemas homólogos 14

1.4.1 Análisis de soluciones homólogas en el marco internacional 14

1.4.2 Análisis de soluciones homólogas en el marco nacional 16

1.4.3 Análisis crítico de los sistemas estudiados 17

1.5 Tendencias actuales en el desarrollo de sistemas de recomendación……..……17

1.5.1 Sistemas de recomendación. Análisis de soluciones homólogas 18

1.6 Algoritmos de recomendación 20

1.6.1 Filtrado colaborativo (FC) 20

1.8 Selección del entorno de desarrollo para la construcción de la solución……..……..22

1.8.1 Metodología de desarrollo de software 22

OpenUp..... 23

1.8.2 Análisis del soporte tecnológico para el desarrollo del software 24

Lenguajes de programación 24

Lenguajes de enmarcado 25

Servidor web 26

Sistema gestor de base de datos 26

Framework de desarrollo 27

Framework de diseño 28

Entornos de desarrollo Integrados (IDE por sus siglas en inglés) 28

Lenguaje de modelado 29

Herramienta de ingeniería de *software* asistida por computación 29

1.9 Conclusiones parciales……30

Capítulo 2: Características del Sistema evaluador de habilidades y de recomendación para el apoyo al proceso de enseñanza-aprendizaje……31
2.1 Introducción……31
2.2 Descripción actual del dominio del problema……31
2.3 Propuesta de solución……32
2.4 Entendimiento del negocio……33
2.4.1 Descripción de los conceptos asociados al dominio……34
2.5 Características del sistema……34
2.5.1 Técnicas de obtención de requisitos……34
2.5.2 Requisitos funcionales (RF)……34
2.5.2 Requisitos no funcionales (RNF)……35
Actores del sistema……37
2.6 Descripción de la solución……37
2.6.1 Diagrama de casos de uso del sistema……37
2.6.2 Especificación de casos de uso del sistema……38
2.6.3 Descripción de la arquitectura de la solución……41
2.7 Modelo de datos……44
2.8 Diagrama de clases del diseño……45
2.9 Conclusiones parciales……47
Capítulo 3: Implementación y prueba del Sistema evaluador de habilidades y de recomendación para el apoyo al proceso de enseñanza-aprendizaje……48
3.1 Introducción……48
3.2 Modelo de implementación……48
3.2.1 Diagramas de componentes……48
3.2.2 Diagrama de despliegue……49
3.3 Estándares de codificación……50
3.4 Seguridad en el sistema……51
3.5 Pruebas de software……52
3.5.1 Pruebas funcionales……52
3.5.2 Pruebas de carga y estrés……53
3.5.3 Pruebas de seguridad……55
3.6 Conclusiones parciales……56
Conclusiones generales……57

Recomendaciones……..58
Bibliografía referenciada……59
Bibliografía consultada……...62

ÍNDICE DE TABLAS

Tabla 1. Análisis crítico de los sistemas estudiados (elaboración propia) *17*
Tabla 2. Listado de requisitos funcionales *35*
Tabla 3. Actores del sistema *37*
Tabla 4. Descripción del caso de uso: Autenticar usuario *38*
Tabla 5. Descripción del caso de uso: Matricular asignatura *40*
Tabla 6. Descripción de las variables (elaboración propia) *52*
Tabla 7. Prueba funcional para el CU "Autenticar usuario" *53*
Tabla 8. Resultado de las pruebas con Jmeter *54*

ÍNDICE DE FIGURAS

Figura 1 : Interacción de los Módulos de un STI (Cataldi, 2009). *9*
Figura 2: Esquema de un STI con sus módulos principales (Cataldi, 2009). *10*
Figura 3: Modelo de razonamiento basado en casos (González, 2013). *13*
Figura 4: Esquema del proceso de generación de una recomendación *18*
Figura 5: Modelo de dominio. *33*
Figura 6: Diagrama de Casos de Uso del sistema. *38*
Figura 7: Diagrama del patrón arquitectónico MVC *43*
Figura 8: Modelo físico de la base de datos *45*
Figura 9: DCDEW para el CU Gestionar Usuario *46*
Figura 10: DCDEW para CU Gestionar Asignatura *47*
Figura 11: Diagrama de componente *49*
Figura 12: Diagrama de despliegue. *50*
Figura 13: Nombre de clases, namespace y use. *51*
Figura 14: Nombre las funciones en Lower Camel Case. *51*
Figura 15: Resultado de las pruebas funcionales *53*
Figura 16: Resultados de las pruebas de seguridad con Acunetix *56*

Introducción

En los últimos años, Internet se ha transformado en una fuente de intercambio de información y en un medio para realizar diversas acciones. La educación a distancia se integró a Internet como un medio para proporcionar nuevas opciones educativas a distancia, donde pueden interactuar un gran número de estudiantes de todas las edades y perfiles socio-culturales.

El suministro de información útil y la experiencia educativa a través de nuevos recursos pedagógicos, facilita el proceso cognitivo del estudiante.

La educación constituye un proceso orientado al desarrollo personal donde el educando construye conocimientos y se desarrolla en planos diversos como persona. El proceso de enseñanza-aprendizaje que se desarrolla para lograrlo, es esencialmente interactivo y comunicativo, y en él interviene la subjetividad del que enseña y del que aprende, dada por motivaciones, aspiraciones, conflictos, necesidades, cualidades personales (Fernández, 2006). Constando las dos últimas entre las que menos se han trabajado, además se aprecia poca intencionalidad del profesorado en la adaptación del proceso de enseñanza-aprendizaje a cada estudiante según sus características y necesidades al aprender.

Varios son los autores que tratan estas temáticas, tal es el caso de Aguilera y Ortiz (2001) que centran su solución en el trabajo con las individualidades de cada estudiante y sus estilos de aprendizaje. González (2001) afirma que el aprendizaje es valorado de manera independiente. Se analizan las estrategias, las técnicas, tipos, niveles y autonomía en el aprendizaje, pero casi nunca se interrelacionan todas estas categorías para establecer nexos y demostrar la dependencia entre ellos.

Los problemas de aprendizaje en la educación superior se reflejan en el bajo rendimiento escolar e inclusive en la pérdida de la carrera. Existe un porcentaje de estudiantes que no cumple con los requerimientos mínimos en su aprendizaje y, conforme aumenta la complejidad de los contenidos que se imparten, ocurre su rezago; lo que ocasiona en el alumno angustia y frustración.

La misión de la educación superior cubana es, según Horruitinier: preservar, desarrollar y promover la cultura de la humanidad, a través de sus procesos sustantivos, en plena integración con la sociedad; llegando con ella a todos los ciudadanos, con pertinencia y calidad y contribuir así al desarrollo sostenible del país (Horruitinier, 1993).

Se basa en la idea de la universalización de la educación superior y el pleno acceso; principios que marcan en la nueva universidad cubana la responsabilidad de trabajar enfocada al logro de niveles de permanencia y de egreso que estén en correspondencia con dicho nivel de acceso. La idea de la universalización de la

educación superior cubana y el pleno acceso de todas las personas a las universidades, han provocado la necesidad de prestar especial atención a reducir al mínimo el fracaso académico (Horruitinier, 1993).

En este sentido la Universidad de las Ciencias Informáticas (UCI) ha dado pasos de avance, perfeccionando el plan de estudios y cada una de sus asignaturas que se imparten, prestando mayor atención al papel de los colectivos de año, los profesores principales, el profesor guía y el tutor.

La UCI tiene la misión de ser un centro docente-productor de *software*. El Ingeniero en Ciencias Informáticas tiene como objeto de trabajo el ciclo de vida de un *software*, con una perspectiva industrial, aplicado a los procesos de tratamiento y gestión de la información y del conocimiento en organizaciones productivas y de servicios, con el objetivo de incrementar la eficacia, la eficiencia y la competitividad en su funcionamiento. Entre las disciplinas que se deben cursar para graduarse como Ingeniero en Ciencias Informáticas, está la Ingeniería de Software, que junto a la de técnicas de programación constituyen el eje de la carrera (Díaz, 2010).

La Ingeniería de Software (ISW) es la disciplina encargada de proveer un enfoque cuantificable, disciplinado y sistemático al desarrollo, operación y mantenimiento del *software* (IEEE, 1993), por lo que juega un papel fundamental en el desarrollo del ingeniero en ciencias informáticas. Existe en la Universidad una disciplina docente llamada Ingeniería y Gestión de Software (IGSW) perteneciente al Departamento Docente Central que lleva el mismo nombre y que a su vez se replica en cada una de las facultades de la Universidad. Está conformada por las asignaturas Sistemas de Bases de Datos I, Sistemas de Bases de Datos II, Ingeniería de Software I, Ingeniería de Software II, Gestión de Software, Optimización de Bases de Datos, Almacenes de Datos y Desarrollo de aplicaciones con enfoque empresarial.

Ciudad (2009) como parte de su investigación doctoral, en el curso 2008-2009 identificó un conjunto de rasgos distintivos del proceso docente de las asignaturas de esta disciplina, algunos de los cuales persisten:

1. La definición de objetivos orientados en su mayoría a la transferencia de información y en menor grado al desarrollo de habilidades.
2. Insuficiente utilización de las Tecnologías de la Información y las Comunicaciones (TIC), siendo su uso fundamentalmente para la distribución de contenidos y en muy poco grado para la interacción y la colaboración entre los participantes; lo que aumenta los períodos de asimilación del conocimiento y produce deficiente desarrollo de las competencias profesionales relacionadas con la Ingeniería de Software.

El perfeccionamiento de la asignatura Ingeniería de Software I (ISWI) de la disciplina IGSW, durante el primer semestre del curso 2010-2011, ha ido desde el replanteamiento de los objetivos y los contenidos que garanticen los conocimientos y características de la personalidad necesarios para la profesión, hasta la organización y desarrollo del proceso de aprendizaje del estudiante, sobre la base de su papel activo y el nuevo rol del profesor. De esta forma se definieron acciones para erradicar las deficiencias detectadas por Ciudad y plasmadas cada año en los informes semestrales de esta asignatura. Una de las transformaciones importantes fue la organización del contenido.

Históricamente la Ingeniería de Software se había impartido a partir de un proceso de desarrollo de *software* guiado por una sola metodología, generalmente tradicional, sin estudiar a profundidad los fundamentos de las áreas del conocimiento de esta disciplina; o sea, que se estudia un caso particular, lo que impide luego que los estudiantes logren generalizar y aplicar sus conocimientos a otros casos particulares, diferentes al que estudiaron. El hecho de que actualmente estén invertidos estos procesos (generalización y particularización) implica un conocimiento más profundo sobre los fundamentos de la disciplina y el análisis de más casos particulares; por lo que el claustro debe invertir tiempo en su preparación. Ahora se pretende que los estudiantes conozcan los fundamentos del proceso de *software* y puedan seleccionar una metodología adecuada para el desarrollo. De igual forma deben conocer las primeras áreas de conocimiento asociadas a la Ingeniería de Software (Ingeniería de Requisitos y Modelación del Análisis de los Requisitos), sin circunscribirlas a una sola metodología.

Sin embargo, aún es insuficiente el aprovechamiento de los resultados de este diagnóstico en el trabajo con las individualidades de cada estudiante. De igual forma se aprecia como una constante en los resultados de los controles a clase realizados, la falta de experiencia del claustro para enfrentar las diferencias individuales de los estudiantes. En los informes semestrales de las asignaturas, sigue señalándose como deficiencia este aspecto[1].

Miguel Díaz-Canel ratificó que junto al ideal de mayor inclusividad, deberá asegurarse la calidad de los procesos de formación y la atención permanente que permita el apoyo necesario y oportuno al estudiante (Díaz-Canel, 2006).

Teniendo como base lo anteriormente descrito se plantea como **problema de investigación:** ¿cómo contribuir al desarrollo del proceso de enseñanza-aprendizaje de las asignaturas que corresponden a la disciplina de IGSW en la carrera de Ingeniería en Ciencias Informáticas?

[1] Tomado de una comunicación personal con el Jefe del Departamento de Ingeniería y Gestión de Software de la Facultad 1.

El **objeto de estudio** de la investigación son las aplicaciones informáticas que sirven de apoyo, al proceso de enseñanza-aprendizaje de la ISW. El **campo de acción** de la investigación lo constituyen los sistemas evaluadores de habilidades y de recomendación. Para dar solución al problema planteado se define como **objetivo general:** construir un sistema web que permita la evaluación de habilidades, el tratamiento diferenciado y el seguimiento de la evolución de los estudiantes que cursan la disciplina de IGSW en la carrera de Ingeniería en Ciencias Informáticas, de acuerdo a su disponibilidad de tiempo y sus necesidades de aprendizaje; derivando en los siguientes **objetivos específicos:**

- Elaborar el marco teórico-conceptual que sustenta la investigación.
- Elaborar la propuesta de aplicación *web* que implemente el basamento teórico de la investigación.
- Implementar la propuesta de solución.
- Demostrar que la solución implementada es funcional mediante el diseño y ejecución de pruebas de *software*.

Para conducir la investigación se plantea como **idea a defender:** con la construcción de un sistema web que permita la evaluación de habilidades, el tratamiento diferenciado y el seguimiento de la evolución de los estudiantes que cursan la disciplina de IGSW en la carrera de Ingeniería en Ciencias Informáticas; se contribuirá con el desarrollo del proceso de enseñanza-aprendizaje de la ISW.

Para dar cumplimiento al objetivo propuesto se ha decidido desarrollar las siguientes **tareas de la investigación:**

- Identificación de los conceptos que regirán la investigación.
- Análisis de los principales enfoques existentes en los Sistemas Tutores Inteligentes (STI).
- Análisis de los principales enfoques existentes en los Sistemas de recomendación.
- Identificación del algoritmo de recomendación a implementar.
- Análisis de la evaluación en el proceso de enseñanza-aprendizaje.
- Selección del entorno de desarrollo para construir la solución.
- Elaboración de la documentación asociada a la metodología seleccionada para la construcción del sistema.
- Implementación de las funcionalidades identificadas.
- Verificación de la idoneidad de las funciones implementadas.
- Demostración de la funcionalidad de la aplicación construida.

Posibles resultados:

- Aplicación web que permita analizar el avance de los estudiantes en las asignaturas que cursan a partir del cumplimiento de objetivos y habilidades definidas.
- Documentación asociada a la aplicación desarrollada.

Para el desarrollo de las tareas científicas se han combinado diferentes métodos teóricos y empíricos de la investigación científica en la búsqueda y procesamiento de la información. Estos son:

Métodos teóricos:

- Análisis documental: este método fue de gran utilidad en la elaboración de la fundamentación teórica de la investigación. Su estudio permitió realizar un análisis profundo de los antecedentes y tendencias actuales en torno a la investigación de los STI, los Sistemas Basados en el Conocimiento (SBC) y los Sistemas de recomendación, además condujo a la identificación de los principales métodos y algoritmos utilizados a nivel mundial y que pueden servir para la implementación del sistema en cuestión.
- Modelación: se empleó para elaborar los artefactos correspondientes a la metodología.
- Analítico-Sintético: este método fue utilizado durante todo el proceso investigativo, permitiendo realizar un análisis exhaustivo de la documentación relacionada con el tema de los STI y los Sistemas de recomendación, su aporte a las diferentes entidades y organizaciones y particularmente a las instituciones y centros educacionales, así como también permitió determinar las buenas prácticas asociadas al desarrollo de herramientas para el apoyo al proceso de enseñanza-aprendizaje.
- Inductivo-Deductivo: se realizó un análisis profundo que permitió hacer deducciones para llegar a tener una visión clara de lo que se quiere desarrollar y adquirir así nuevos conocimientos. A partir de la búsqueda de información referente a técnicas de Inteligencia Artificial (IA) que se emplean en el desarrollo de los STI, se propone la utilización del funcionamiento de dos de ellas para solucionar la problemática existente.

Métodos empíricos:

- Observación: permitió la comprensión del proceso de enseñanza-aprendizaje de los estudiantes de la educación superior en la disciplina de ISW, de igual manera viabilizó el entendimiento de las funciones de un tutor humano que debe emular la aplicación.
- Entrevista: este método fue de gran importancia en la obtención de los requisitos funcionales de la herramienta propuesta. Además fue de utilidad para interactuar con personal capacitado en temas IA de la universidad.

El contenido de este documento está estructurado como se muestra a continuación:

Capítulo 1 "Marco Teórico Conceptual del Sistema evaluador de habilidades y de recomendación para el apoyo al proceso de enseñanza-aprendizaje":

En este capítulo se fundamentan los elementos teóricos necesarios que sustentan la investigación. Se analizan las tendencias actuales en el desarrollo de STI y Sistemas de recomendación. Se realiza un estudio del estado del arte de ambos tipos de sistemas mediante el análisis de herramientas similares existentes a nivel nacional e internacional. Se selecciona el entorno de desarrollo para la construcción de la solución.

Capítulo 2 "Características del Sistema evaluador de habilidades y de recomendación para el apoyo al proceso de enseñanza-aprendizaje":

En este capítulo se documenta el proceso de elaboración del sistema de acuerdo a lo establecido por la metodología utilizada. Se especifican los casos de uso identificados. Se presenta el diagrama de CU, los diagramas de clases del diseño, el modelo físico de la base de datos, así como todo lo referente a la arquitectura del sistema así como los patrones de diseño utilizados. Se describe la solución propuesta partiendo del modelo de domino. Se delimitan las funcionalidades del sistema a través de requisitos funcionales y no funcionales.

Capítulo 3 "Implementación y prueba del Sistema evaluador de habilidades y de recomendación para el apoyo al proceso de enseñanza-aprendizaje":

En este capítulo se describe la estructura del código implementado, además se presentan los diagramas de componentes, algunos de los diseños de casos de prueba a utilizar en la validación del sistema, así como los diferentes tipos de pruebas realizadas al mismo y los resultados arrojados por estas.

Para finalizar se presentan las conclusiones, las recomendaciones, las referencias bibliográficas, la bibliografía consultada, un glosario de términos y los anexos.

Capítulo 1: Marco teórico conceptual del Sistema evaluador de habilidades y de recomendación para el apoyo al proceso de enseñanza-aprendizaje

1.1 Introducción

En el presente capítulo se abordan los conceptos básicos asociados al dominio del problema. Asimismo se evalúan las tendencias actuales en el desarrollo de Sistemas Tutores Inteligentes y los Sistemas de recomendación, así como el funcionamiento de las técnicas que estos emplean, para incluirlas en la solución. Se realiza un estudio de los sistemas homólogos. Finalmente se selecciona el entorno de desarrollo para la construcción de la propuesta.

1.2 Definición del marco conceptual de la investigación

Para comprender mejor el dominio de la investigación se hace necesario analizar algunas definiciones relacionadas a la esfera educacional y otras vinculadas a la informática.

Aprendizaje autónomo

El aprendizaje autónomo, también descrito como aprendizaje autodirigido, es definido por Knowles como el "proceso en el cual los individuos toman la iniciativa en el diseño de sus experiencias de aprendizaje, diagnóstico de necesidades, localización de recursos y evaluación de logros" (Knowles , 1995), además afirma que los adultos tienen una elevada necesidad psicológica de ser auto dirigidos y justifica el desarrollo de habilidades para el aprendizaje.

Otros autores han hecho planteamientos en el mismo sentido. Al respecto Kidd enunció: "el objetivo de la educación de adultos o de cualquier tipo de educación es convertir al sujeto en un estudiante continuamente dirigido desde adentro que opera por sí mismo" (Kidd, 1979).

María Pinto considera aprendizaje autónomo a "la facultad para dirigir, controlar, regular y evaluar su forma de aprender de manera consciente e intencionada haciendo uso de estilos de aprendizaje" (Pinto, 2009).

En el presente trabajo se asumirá esta última como definición de aprendizaje autónomo, ya que introduce el término "estilos de aprendizaje" que constituye otro concepto asociado a la investigación.

Estilos de aprendizaje

Para autores como Dunn los estilos de aprendizaje reflejan "la manera en que los estímulos básicos afectan a la habilidad de una persona para absorber y retener la información" (Dunn, y otros, 1979), mientras para

Hunt estos describen “las condiciones bajo las que un discente está en la mejor situación para aprender, o qué estructura necesita el discente para aprender mejor” (Hunt, 1979).

Por otro lado para Riechmann “es un conjunto particular de comportamientos y actitudes relacionados con el contexto de aprendizaje” (Riechmann, 1979). Gregorc en cambio, plantea que representan “los comportamientos distintivos que sirven como indicadores de cómo una persona aprende y se adapta a su ambiente” (Gregorc, 1985).

Las autoras de la investigación asumen como definición de estilos de aprendizaje “la forma en que los estímulos básicos afectan a la habilidad de una persona para absorber y retener la información, dicha habilidad indica cómo una persona aprende y se adapta a su ambiente”.

Esta enunciación resume la idea de Dunn que guardan relación con los tipos de aprendizaje y que se centra en los estímulos que el individuo es capaz de percibir y que condicionan su aprendizaje; añadiendo a su definición lo referido por Gregorc de que estos comportamientos son considerados indicadores del aprendizaje y de la adaptación al ambiente.

Tipos de aprendizaje

Es el término que se refiere al hecho de que cada persona utiliza un método propio o una estrategia para aprender. Aunque las estrategias varían según lo que se quiere aprender. Según Villanueva (1997) “cada uno tiende a desarrollar ciertas preferencias o tendencias globales”. Esas preferencias o tendencias a utilizar, más unas determinadas maneras de aprender que otras, constituyen el tipo de aprendizaje. Se resumen en tres tipos de aprendizaje: visual, auditivo y táctil kinestésico.

Gestión de evaluación

Varios autores refieren que la gestión de la evaluación es el proceso de “recopilar, organizar e interpretar informaciones obtenidas mediante diferentes técnicas e instrumentos, con la finalidad de valorar y emitir juicios sobre los aprendizajes de los alumnos/as y tomar decisiones que mejoren el proceso enseñanza aprendizaje” (Rodríguez, y otros, 2006).

Martínez (2007) considera la gestión de evaluación al “conjunto de operaciones que se realizan para comprobar y valorar el cumplimiento de los objetivos propuestos y la dirección didáctica de la enseñanza y el aprendizaje en sus momentos de orientación y ejecución”.

En el presente trabajo se asumirá como gestión de evaluación la definición de Martínez pues esta analiza el cumplimiento de los objetivos propuestos, factor que se tendrá en cuenta en el desarrollo de la solución.

1.3 Tendencias actuales en el desarrollo de Sistemas Tutores Inteligentes

1.3.1 Sistemas Tutores Inteligentes. Análisis de soluciones homólogas

Los Sistemas Tutores Inteligentes (STI) se delimitan como "un sistema que incorpora técnicas de Inteligencia Artificial (IA) a fin de crear un ambiente que considere los diversos estilos cognitivos de los alumnos que utilizan el programa" (Giraffa, 1997).

Carbonell los define como "un sistema de enseñanza asistida por computadora, que utiliza técnicas de IA, principalmente para representar el conocimiento y dirigir una estrategia de enseñanza; y es capaz de comportarse como un experto, tanto en el dominio del conocimiento que enseña (mostrando al alumno cómo aplicar dicho conocimiento), como en el dominio pedagógico, donde es capaz de diagnosticar la situación en la que se encuentra el estudiante y de acuerdo a ello ofrecer una acción o solución que le permita progresar en el aprendizaje" (Carbonell, 2000).

Otro criterio acerca de los STI es que "son sistemas que se adaptan al estado cognitivo de los estudiantes y que proporcionan la ayuda pedagógica adecuada para propiciar un buen aprendizaje" (Parra, 2010).

Los STI "comenzaron a desarrollarse con la idea de impartir el conocimiento usando alguna forma de inteligencia para asistir y guiar al estudiante en su proceso de enseñanza, identificando la forma en que resuelve un problema a fin de poder brindarle ayudas cognitivas cuando lo requiera" (Cataldi, 2009). Por lo que se puede lograr un proceso de enseñanza-aprendizaje adaptado al estado cognitivo del estudiante sin la presencia de un tutor humano.

Estos sistemas son considerados "inteligentes", pues realizan acciones pedagógicas sobre la forma de enseñanza y mantienen información sobre las necesidades del estudiante, ya que poseen la habilidad de determinar el qué, el cómo y a quién enseñar a través de la interacción de sus módulos (Cataldi, 2009), tal y como se muestra en la figura 1.

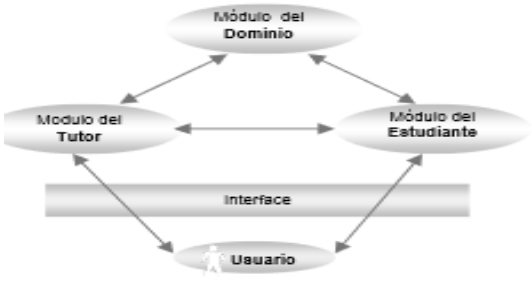


Figura 1 : Interacción de los Módulos de un STI (CATALDI, 2009).

Refiere Cataldi que por lo general los STI se estructuran en cuatro módulos (Cataldi, 2009):

El **Módulo del Estudiante** define el conocimiento del estudiante. Tiene como objetivo realizar un diagnóstico del estado cognitivo del mismo, ya que deberá ser capaz de determinar su comprensión, identificando deficiencias y progreso a través del desarrollo de ciertos ejercicios que evalúan el contenido, recomendando así, la estrategia de estudio más conveniente de acuerdo a sus resultados, guardando de esta forma sus datos personales, siendo capaz de determinar la capacidad de adaptación a sus necesidades, es decir, se encarga de obtener una representación de las características de cada estudiante (Ver figura 2).

El **Módulo del Dominio** define el dominio del conocimiento. Tiene como objetivo proporcionar los conocimientos de forma adecuada para que el estudiante adquiera las habilidades y conceptos que se esperan, siendo capaz de generar preguntas, explicaciones, respuestas y tareas para el estudiante, resolver los problemas y corregir las soluciones presentadas (ver figura 2).

El **Módulo Pedagógico** define el conocimiento pedagógico. Tiene como objetivo dirigir al estudiante en su proceso de aprendizaje y realizar ajustes a medida que progresa, definiendo y aplicando una estrategia pedagógica de enseñanza que contiene los objetivos que se persiguen así como los planes para alcanzarlos (Ver figura 2).

La **Interfaz** con el usuario permite la interacción del estudiante con el sistema. Debe ofrecer la información necesaria para la realización de tareas que el sistema proponga (Ver figura 2).

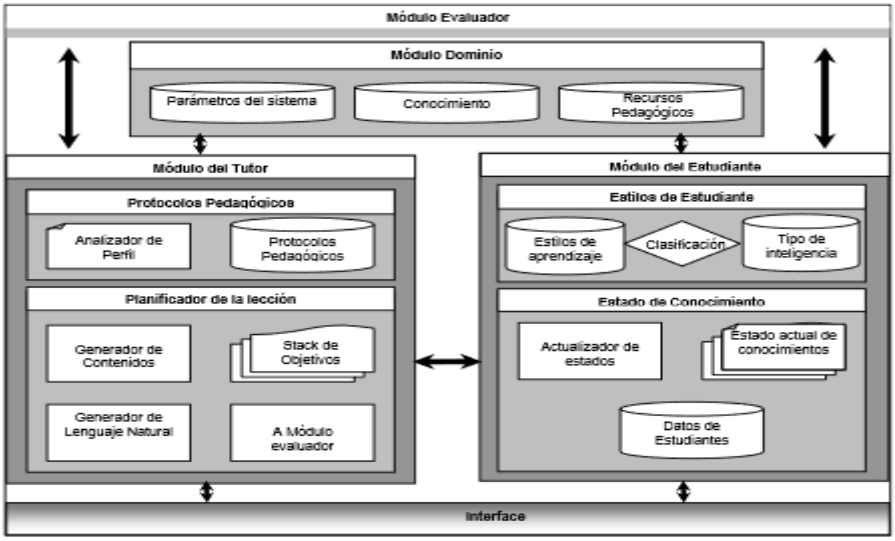


Figura 2: Esquema de un STI con sus módulos principales (CATALDI, 2009).

A través de la interacción entre los módulos, el sistema debe ser capaz de determinar lo que sabe el estudiante y cómo va en su progreso, por lo que para lograrlo, el sistema deberá identificar las fortalezas y debilidades de cada estudiante, con el fin de establecer un plan de instrucción en dependencia de sus resultados (Cataldi, 2009).

1.3.2 Sistemas basados en el conocimiento

Los sistemas basados en el conocimiento (SBC) constituyen técnicas de IA válidas para enfrentar la construcción de un STI, dada por sus aspectos afines. Pero no todos los paradigmas para crear SBC facilitan la concepción de un STI, donde lo fundamental para su desarrollo es determinar cómo representar el conocimiento requerido para sus módulos y a partir de dicho conocimiento realizar un diagnóstico del estudiante para que el sistema se adapte a sus características. Sin embargo, similitudes de los STI y los SBC son factores a estudiar para concebir todos los módulos de los STI y un diagnóstico adecuado de qué y cómo enseñar dependiendo del estudiante (Martínez, 2011).

Una característica distintiva de estos sistemas es la separación del conocimiento (base del conocimiento) del método de solución del problema (máquina de inferencia).

En la base del conocimiento (BC) se almacena el conocimiento necesario para resolver los problemas del dominio de aplicación para el cual se desarrolla el SBC. El conocimiento que se almacena en la BC puede ser de diferentes tipos: conocimiento simbólico sobre cómo resolver los problemas del dominio, el cual se puede representar mediante diversas formas (reglas de producción, redes semánticas), y probabilidades o frecuencias, que modelan cómo se relacionan los valores de los diferentes rasgos que caracterizan el dominio, pesos de una red neuronal, casos o ejemplos de problemas del dominio. Estos diferentes tipos de conocimiento dan lugar a diferentes tipos de sistemas basados en el conocimiento, entre ellos: Sistemas basados en reglas, Sistemas basados en probabilidades, Sistemas expertos conexionistas o redes expertas y Sistemas basados en casos.

La máquina de inferencia (MI) es un procedimiento en el cual está implementado algún método que utiliza el conocimiento de la BC para resolver los problemas del dominio. El tipo de conocimiento determina qué método de solución de problemas (MSP) será posible utilizar (Martínez, 2011).

Sistemas basados en reglas

Un sistema basado en reglas (SBR) es un sistema basado en el conocimiento en el que se hace una representación mediante reglas de producción [2] o reglas condicionales. Las reglas representan el conocimiento utilizando el formato SI-ENTONCES (IF-THEN).

En este tipo de sistema, en la BC se encuentra el conjunto de reglas que definen el problema, contiene la representación del conocimiento sobre el dominio de aplicación del sistema y se divide en la base de hechos y en la base de reglas. La base de hechos representa el conocimiento en las variables de entrada y salida del sistema; y en la base de reglas se combinan variables de la base de hechos con el IF y el THEN junto con los operadores lógicos AND, OR y NOT. De esta manera, la MI obtiene las conclusiones aplicando la lógica a estas reglas (Pepa, 2014).

Sistemas basados en probabilidades

Los sistemas basados en probabilidades (SBP) comenzaron a ser utilizados porque los SBR no tenían la capacidad de resolver situaciones que tuvieran incertidumbre, de ahí que se desarrollaran sistemas cuya BC contara con incertidumbre utilizando la probabilidad para medirla.

Los SBP, al igual que los SBR, cuentan con una BC pero con la diferencia de que esta se forma por el espacio probabilístico que describe el problema. En este tipo de sistemas la MI está basada en probabilidades condicionales y se encarga de actualizar las probabilidades con base en los hechos que se observen del ambiente en que se desempeña; y cuenta con el sistema de control de coherencia, la cual logra, especificando las restricciones que deben cumplir los datos introducidos (Gil, 2012).

Sistemas expertos conexionistas o redes neuronales artificiales

Las redes neuronales artificiales (RNA) son modelos matemáticos inspirados en las neuronas biológicas del cerebro humano y programado en sistemas digitales, que tienen habilidades de aprendizaje automático, generalización y abstracción. Con estos modelos pueden resolverse una gran variedad de problemas de reconocimiento, aproximación, predicción, clasificación, optimización (Gil, 2012).

Los términos sistemas expertos (SE) y SBC, se utilizan como sinónimos, a pesar de que tienen sus peculiaridades. Un SE es un conjunto de programas que, sobre una BC, posee información de uno o más

[2] Regla de producción: proposición lógica que relaciona dos o más objetos e incluye dos partes: la premisa (antecedente) o conclusión (consecuente). Cada una de estas partes consiste en una expresión lógica con una o más afirmaciones conectadas mediante operadores lógicos (AND, OR, NOT).

expertos en un área específica. Un SE que utiliza redes neuronales (RN), es mejor conocido como sistema experto conexionista (SEC).

En el desarrollo de una RN no hay que programar ni el conocimiento ni las reglas del procesamiento del conocimiento, ya que aprende mediante el ajuste de las conexiones ponderadas entre las neuronas de distintas capas de la red. El empleo de las RN en el desarrollo de SE ofrece entre sus ventajas principales el hecho de no necesitar un experto humano al cual extraerle el conocimiento, pues ellas adquieren su conocimiento a partir de ejemplos. No obstante tienen también desventajas, entre las que se encuentran el hecho de necesitar una gran cantidad de ejemplos y no poder explicar cómo se alcanzan los resultados (Gil, 2012).

Sistemas basados en casos

Un Sistemas basado en casos, o también conocido como sistemas de razonamiento basado en casos (RBC), es una técnica de IA que intenta llegar a la solución de nuevos problemas de forma similar a como lo hacen los seres humanos utilizando la experiencia acumulada hasta el momento de acontecimientos similares (Arias, 2009). Según Aamodt (1994) el RBC es un proceso cíclico que comprende un conjunto de etapas como se muestra en la figura 3.

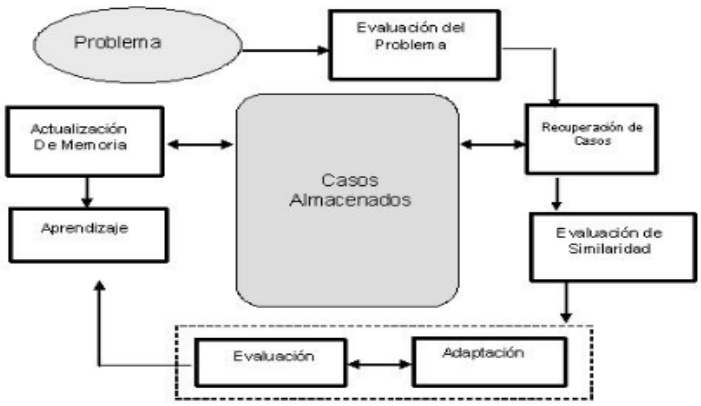


Figura 3: Modelo de razonamiento basado en casos (GONZÁLEZ, 2013).

Los "casos" son problemas resueltos y almacenados en la BC. Cuando hay un nuevo problema que resolver, éste es descrito para el módulo de recuperación, el cual realiza una búsqueda en la BC y encuentra problemas o casos similares. Estos problemas o casos similares resueltos son recuperados (caso recuperado) y enviados al módulo de adaptación, donde son analizados para construir una solución para el

nuevo problema y una vez hallada la solución se almacena junto con la descripción del problema en la BC, ya que constituye un nuevo caso (González, 2013).

1.4 Análisis de sistemas homólogos

El empleo de sistemas que permiten evaluar el desempeño de los estudiantes en las diferentes asignaturas, así como recomendar actividades a estos; se ha convertido en un nuevo paradigma en el proceso de enseñanza aprendizaje a nivel mundial. Estos sistemas tienen en cuenta métodos y técnicas de enseñanza para facilitar a los estudiantes la adquisición de conocimientos. Con estas nuevas características se han desarrollado sistemas más flexibles, adaptados a los intereses del estudiante y con métodos pedagógicos que facilitan el proceso de aprendizaje. A continuación se destacan algunos sistemas existentes a nivel nacional e internacional.

1.4.1 Análisis de soluciones homólogas en el marco internacional

- **Lingua.ly**

Es una solución móvil gratuita para el aprendizaje de idiomas a través de la inmersión digital. La plataforma adopta un enfoque innovador para cuantificar el aprendizaje de idiomas a través de la exposición y la interacción en línea, derivada del contenido del mundo real mediante el aprovechamiento del comportamiento de navegación natural de los usuarios. Proporciona dos tecnologías: una extensión del navegador de *Google Chrome* y una aplicación móvil para *Android* que permite aprender el idioma sobre la marcha. Tiene soporte para varios idiomas. La herramienta incluye varios módulos para el aprendizaje interactivo tales como: un diccionario gratuito para recopilar vocabulario, crear tarjetas de vocabulario del idioma personalizado centrado en torno a la repetición. Tiene disponible un narrador de texto: las palabras seleccionadas serán leídas en voz alta, definidas en la página y agregadas automáticamente a la colección. Entre las principales desventajas que presenta esta herramienta figura el hecho de que depende absolutamente de la navegación del usuario en otras páginas, de las cuales selecciona los textos de su interés. Además sugiere a todos los usuarios las mismas actividades, solo variando los textos, no personaliza los ejercicios atendiendo las necesidades de los usuarios (Lingua.ly, 2014).

- **Sistema Tutor Inteligente para la enseñanza musical**

Este sistema está centrado en el aprendizaje de música. Fue implementado en lenguaje C# y el conocimiento procedimental debe representarse con reglas de producción. La herramienta administra el

acceso a temas del contenido, permite informar al estudiante del contenido específico de cada tema, así como evaluarlo mediante ejercicios. Además asigna los temas incrementalmente. Publica lecciones específicas de acuerdo al tema escogido por el estudiante. Muestra mensajes de sugerencia, si existe problema en la realización de los ejercicios.

Por otra parte la herramienta no cuenta con un registro de las evaluaciones del estudiante. Los ejercicios se asignan por temas y no atendiendo al progreso del estudiante. No permite al usuario abandonar un ejercicio hasta que responda correctamente. La evaluación califica al estudiante como aprobado o reprobado pero no le recomienda otros ejercicios para rebasar sus limitaciones (Renato, 2013).

- **Plataforma virtual de Sistemas Tutores Inteligentes para la enseñanza de las matemáticas**

Este sistema fue desarrollado para uso interactivo de los estudiantes de matemáticas, como una plataforma virtual con tutores inteligentes para el apoyo al proceso de aprendizaje. A partir del uso del sistema de tutoría inteligente, la plataforma genera enseñanza en tiempo real y por demanda de las necesidades individuales, con base en herramientas gratuitas como PHP, *jQuery*, *Python* y un Sistema Algebraico Computacional[3], también conocido por sus siglas en inglés CAS (*Computer Algebra System*). Permite hacer gráficas para manipularlas a través de botones, con el fin de ilustrar el concepto de la derivada, representar el lanzamiento de un proyectil para enseñanza o investigación o simular modelos físicos y matemáticos. A medida que el usuario va respondiendo correctamente, se van desplegando las preguntas que formula el tutor inteligente hasta llegar a la solución. Si el usuario no puede contestar, el tutor le hace sugerencias hasta determinado número de intentos. Mediante estas herramientas se realizan también exámenes en línea, a través de *script* gratuitos y controlando el tiempo. En palabras del profesor Álvaro Humberto Salas, catedrático matemático y director del grupo Fizmako “Estamos desarrollando herramientas que permitan hacer cálculos científicos en la *web*. Utilizando el lenguaje de programación *Python* es posible mostrar soluciones de problemas matemáticos, además de desarrollar tutores con los cuales la gente pueda interactuar”.
El sistema, aunque muy completo, para el propósito para el cual fue pensado, carece de un mecanismo de recomendación de ejercicios personalizado. Evalúa solamente tres respuestas erróneas para un mismo ejercicio, por lo que un estudiante no tiene la oportunidad de conocer la solución de un ejercicio que

[3]Sistema Algebraico Computacional es un programa de ordenador o calculadora avanzada que facilita el cálculo simbólico; el cual trabaja ecuaciones y fórmulas donde no necesariamente hay números.

respondió incorrectamente en tres oportunidades. Además el sistema es temático para matemáticas (Universidad Nacional de Colombia, 2014).

1.4.2 Análisis de soluciones homólogas en el marco nacional

- **Sistema Tutorial Inteligente para Diagnóstico y Tratamiento de Infecciones de Transmisión Sexual (STIITS)**

Fue desarrollado por la Universidad de Cienfuegos, en conjunto con especialistas de segundo grado del hospital Dr. Gustavo Aldereguía Lima, que hicieron labor de expertos en la validación de la base de conocimientos para el posterior diagnóstico de las enfermedades, así como aportar todos los datos para la realización del tutorial. STIITS responde a una necesidad social de la provincia Cienfuegos, y también nacional por sus posibilidades de empleo en la docencia. No existe algún otro *software* en el país con características similares para el mismo fin (García García, y otros, 2009).

Entre las limitantes de este sistema está el hecho de que es informativo pero no interactivo, o sea, dispone de información para consulta pero no propone ninguna clase de ejercicios.

- **APA-Prolog**

Es el ambiente inteligente de enseñanza-aprendizaje asistido por computadora para la programación lógica que utiliza mapas conceptuales para su solución; y tiene asociado navegación dirigida por agentes pedagógicos inteligentes. Fue desarrollado en el Centro Universitario de Sancti Spíritus, con el apoyo de la Universidad Central de Las Villas y la Universidad de Granma y puesto a prueba con estudiantes de cuarto año de la carrera de Ingeniería Informática del Centro Universitario de Sancti Spíritus (Rodríguez, 2008).

El sistema dispone de contenidos de consulta interactivos pero no propone ejercicios a los estudiantes.

- **MacBay**

Es un sistema que posibilita elaborar sistemas de enseñanza-aprendizaje inteligentes mediante la combinación de redes bayesianas y mapas conceptuales de cualquier asignatura y de cualquier nivel de enseñanza. MacBay posibilita que los profesores que diseñen e implementen sistemas de enseñanza no necesariamente sean especialistas en computación.

La principal limitación de este sistema radica en que no fue pensado para recomendar ejercicios.

1.4.3 Análisis crítico de los sistemas estudiados

En la tabla 1 se muestran las características fundamentales de algunos de los sistemas estudiados, haciendo especial énfasis en los rasgos que permiten hacer un juicio tanto sobre su utilidad como de sus limitaciones; factores a tener en cuenta en la propuesta de solución, como son: el área del conocimiento en la que se emplea, el tipo de tecnología que se utilizó para su implementación, el tipo de sistema determinado por la técnica de IA que utiliza, el tipo de licencia y la apariencia gráfica del sistema en cuestión y si disponen de algún tipo de sistema de recomendación y evaluación de actividades, siendo las dos últimas críticas para el análisis.

Tabla 1. Análisis crítico de los sistemas estudiados (elaboración propia)

Nombre del STI	**Área de utilidad**	**Tecnología de desarrollo**	**Tipo de sistema**	**Tipo de licencia**	**Apariencia gráfica**	**Recomendación**	**Evaluación**
Lingua.ly	Idioma	Android y web	RBC	Gratuita	Interfaz amigable	**No personalizado**	**Flexible**
STI para la enseñanza musical	Música	C#	SBR	Gratuita	Interfaz amigable y dinámica	**No personalizado**	**Menos flexible**
Plataforma virtual de STI para la enseñanza de las matemáticas	Matemática	PHP, *jQuery*, *Python y* CAS	RNA	Gratuita	Interfaz compleja	**No personalizado**	**Totalmente Inflexible**

De manera general, las herramientas anteriormente analizadas (ver tabla 1) no resuelven la problemática de la investigación pues no cumplen con características necesarias que el sistema requiere: algunos de ellos son ilustrativos, o sea, disponen de información de diversos contenidos pero no proponen ejercicios a los usuarios, no tienen mecanismos de evaluación adecuados o simplemente carecen de estos. Están orientados, en su mayoría, a temáticas específicas que en ninguno de los casos se corresponden con ISW.

1.5 Tendencias actuales en el desarrollo de sistemas de recomendación

Los sistemas de recomendación son herramientas que sugieren al usuario un determinado objeto de estudio o *ítem*, a partir de las preferencias y opiniones suyas o de otros usuarios. Generalmente, un sistema de

recomendación compara el perfil del usuario con algunas características de referencia de los temas, y busca predecir el *ranking* o ponderación que el usuario le daría a un *ítem* que aún el sistema no ha considerado.
El uso de estos sistemas se está poniendo cada vez más de moda en Internet debido a que son muy útiles para evaluar y filtrar la gran cantidad de información disponible en la Web con el objetivo de asistir a los usuarios en sus procesos de búsqueda y recuperación de información.

1.5.1 Sistemas de recomendación. Análisis de soluciones homólogas

Un sistema de recomendación está asociado con un conjunto de *ítems* I = {i1,..., in} y su objetivo es recomendar a los usuarios *ítems* de I que les puedan ser de interés.
Este tipo de sistemas para generar recomendaciones, usan las entradas del usuario activo, pero también información sobre los *ítems* o información del resto de usuarios del sistema, que actúan como colaboradores. En este sentido, la realimentación por parte de los usuarios es muy importante de cara a albergar una información más completa ante futuros procesos de generación de recomendaciones. La figura 4 refleja el proceso de generación de las recomendaciones.

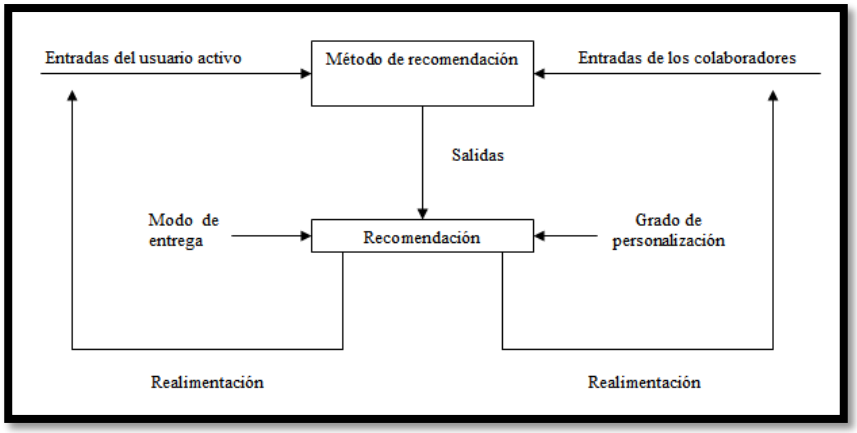


Figura 4: Esquema del proceso de generación de una recomendación

Para poder realizar una recomendación a un usuario, es necesario conocer algún tipo de información sobre sus preferencias. Además, dependiendo del tipo de sistema también se necesita información sobre los *ítems* a recomendar o información reunida sobre el resto de usuarios del sistema (comunidad de usuarios o colaboradores). Esta información necesaria para realizar las recomendaciones constituye la entrada o entradas del sistema.

La información sobre los usuarios puede venir dada de dos formas que no tienen por qué ser mutuamente exclusivas: por extensión o intencionalmente. Por extensión se refiere a la información que se tenga sobre las experiencias pasadas del usuario con respecto a los *ítems* encontrados, lo que también se conoce como navegación implícita, pues el usuario no es consciente de estos seguimientos. Por información expresada intencionalmente se entiende alguna especificación de los *ítems* deseados por los usuarios. También se le llama navegación explícita y consiste en que el usuario expresa intencionalmente (de forma explícita) al sistema la información sobre sus preferencias.

La salida del sistema está constituida por las recomendaciones generadas por el sistema, que variarán dependiendo del tipo, cantidad y formato de la información proporcionada al usuario. Algunas de las formas comunes de representar la salida son las siguientes:

- Sugerencia o lista de sugerencias al usuario de una serie de *ítems*.
- Presentar al usuario predicciones del grado de satisfacción que se asignará al *ítem* concreto. Estas estimaciones pueden ser presentadas como personalizadas al usuario o como estimaciones generales del conjunto de colaboradores.
- Cuando la comunidad de usuarios es pequeña o se conocen bien los miembros de dicha comunidad, podría ser útil visualizar las valoraciones individuales de los miembros que permitiría al usuario activo obtener sus propias conclusiones sobre la efectividad de una recomendación.

Independientemente de estos formatos de salida, puede resultar muy interesante incluir una breve descripción o explicación sobre el *ítem* recomendado a modo de justificación del porqué de dicha recomendación.

Para generar las recomendaciones hay dos posibilidades comúnmente aceptadas que dan lugar a dos grandes grupos de sistemas: los de recomendación colaborativos y los no colaborativos o basados en contenidos (también conocidos como reclusivos).

Los sistemas de recomendación no colaborativos realizan las recomendaciones usando únicamente las preferencias del usuario activo y los atributos de los *ítems* a recomendar. Estos sistemas usan correlaciones entre *ítems* para identificar *ítems* asociados frecuentemente a un *ítem* por el que el usuario ha mostrado interés y por tanto recomendarle dichos *ítems* al usuario.

Los sistemas de recomendación se califican como colaborativos si usan la información conocida sobre las preferencias de otros usuarios para realizar la recomendación al usuario que la precise. Los sistemas colaborativos identifican usuarios cuyas preferencias sean similares a las de otros usuarios dados; y

recomiendan a los primeros los elementos que hayan satisfecho a los otros; de esta forma, si dos usuarios U1 y U2, comparten el mismo sistema de valores (tienen las mismas preferencias) y al usuario U1 le ha satisfecho un *ítem* i, probablemente este *ítem* también satisfaga al usuario U2 por lo que debe recomendársele. Por ello, en estos sistemas la definición de medidas de similitud entre preferencias es un punto crítico. La situación puede ser representada como una matriz de usuarios e *ítems*, donde cada celda representa la valoración de un usuario con respecto a un *ítem* concreto (Pepa, 2014).

Luego de haber analizado sistemas homólogos (ver tabla 1) puede concluirse que reflejan limitantes en una de las cuestiones críticas que aborda la investigación, es el hecho de que no recomiendan ejercicios atendiendo al nivel del estudiante, sino que las actividades son comunes a todos los usuarios.

1.6 Algoritmos de recomendación

Existe una amplia variedad de algoritmos de recomendación, por ello, se ha hecho una selección de aquellos más representativos para realizar una comparación entre ellos sobre distintos conjuntos de datos con diferentes características.

A continuación se definen con detalle y por grupos los algoritmos que se han incluido en la comparativa realizada.

1.6.1 Filtrado colaborativo (FC)

Esta familia de algoritmos de recomendación se caracteriza porque se utiliza información de unos usuarios para producir recomendaciones a otros. Así pues, en estos algoritmos los usuarios se benefician de la experiencia de otros usuarios.

Dentro de la amplia diversidad de algoritmos desarrollados en esta vertiente, las dos estrategias de filtrado colaborativo más extendidas y actualmente exitosas son, en cuanto a los métodos basados en memoria, los algoritmos de vecinos próximos, y entre los basados en modelo, la factorización de matrices (Pepa, 2014).

Algoritmo de vecinos próximos

Este método selecciona los k vecinos más similares al usuario o al *ítem* objetivo, de forma que mediante la combinación lineal del *rating*[4] de los vecinos se realiza una predicción de *rating*. A partir de esta predicción,

[4] Rating: un valor para cada par usuario-ítem.

es posible determinar el *ranking* de *ítems* a recomendar, simplemente ordenándolos por orden descendente del *rating* predicho (Koren, 2010).

Existe una gran cantidad de variaciones para estos algoritmos. Los algoritmos de vecinos próximos se han desarrollado en dos perspectivas posibles: recomendación de vecinos próximos por usuario y por *ítem*.

- **Basado en usuario (*User* kNN)**

Se recomiendan al usuario los *ítems* que han satisfecho a usuarios similares. Su fórmula es la siguiente:

$$f(u,i) = \sum_{\substack{v \in N_k(u) \\ r(v,i) \neq \emptyset}} sim(u,v) * r(v,i)$$

donde:

- **sim(u, v)** es la función de semejanza entre el usuario 'u' y el usuario 'v'.
- **r(v, i)** es el peso asociado del ítem 'i' por el usuario 'v'.

- **Basado en *ítem* (*Ítem* kNN)**

Se recomiendan al usuario los *ítems* semejantes a *ítems* que le han satisfecho. Su función de *ranking* es similar a la de User kNN:

$$f(u,i) = \sum_{j \in N_k(i)} sim(i,j) * r(u,j)$$

donde:

- **sim(i, j)** es la función de semejanza entre el ítem 'i' y el ítem 'j'.
- **r(u, j)** es el peso asociado del ítem 'j' por el usuario 'u'.

La función de semejanza entre los usuarios o *ítems*, puede calcularse mediante varios métodos:

Coseno:

Mide el ángulo entre los vectores (ratings) de cada par de usuarios o ítems, de forma que son más similares cuando su valor se aproxime a la unidad.

$$sim(u,v) = \frac{\sum_{\substack{i:\, r(u,i) \neq \emptyset \\ r(v,i) \neq \emptyset}} r(u,i) * r(v,i)}{\sqrt{\sum_{i: r(u,i) \neq \emptyset} r(u,i)^2} \sqrt{\sum_{r(v,i) \neq \emptyset} r(v,i)^2}}$$

Jaccard:

Dos usuarios o ítems son más similares cuanto más parecida sea la intersección a

$$sim(u,v) = \frac{|u \cap v|}{|u \cup v|} = \frac{|u \cap v|}{|u| + |v| - |u \cap v|}$$

la unión de ambos, es decir, cuantos más ratings en común tengan.

1.7 Fundamentos de la selección

Para la solución se propone el uso del funcionamiento de los SBR, pues son comúnmente utilizados para la representación del conocimiento puesto que son sistemas modulares y uniformes. Cada regla es una unidad de conocimiento que puede ser añadida, modificada o removida independientemente de las otras reglas existentes, y todo el conocimiento del sistema se expresa en el mismo formato. Además las reglas son un formato natural para expresar conocimiento en algunos dominios. Los expertos lógicamente piensan en los problemas y sus soluciones.

Los métodos de vecinos próximos (kNN) son algoritmos de recomendación clásicos, que a pesar de su simplicidad, suelen ofrecer buenos resultados. Aunque, los modelos de factorización de matrices son superiores a los de kNN en términos de error de predicción, no ocurre así en calidad de *ranking*.

Todo lo anterior condujo a que se decidiera combinar el uso del funcionamiento de los SBR y el algoritmo de recomendación kNN vecinos más cercanos (*Ítem* kNN) en la implementación del sistema, los cuales permitirán que el mismo evalúe de manera correcta los ejercicios y a partir de ahí recomiende a los estudiantes otras actividades adecuadas a su nivel y a las habilidades y objetivos vencidos, y que estén en correspondencia con las habilidades y objetivos que necesita cumplir.

1.8 Selección del entorno de desarrollo para la construcción de la solución

Al analizar el entorno de desarrollo de *software* del centro de Ideoinformática (CIDI) y las especificaciones del cliente para el sistema, se decidió indagar sobre las características y ventajas de la metodología de desarrollo *OpenUp* para guiar el desarrollo de la aplicación, así como otras tecnologías para la implementación de la propuesta de solución.

1.8.1 Metodología de desarrollo de software

Una metodología de desarrollo, en ingeniería de *software*, es un conjunto de herramientas, técnicas, procedimientos y soporte documental encaminados a estructurar, planificar y controlar el proceso de desarrollo de forma organizada y lógica, que tiene como objetivo apoyar a los desarrolladores en la creación de un nuevo *software* (Zambrano, 2013).

“Existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo. Por una parte están aquellas propuestas más tradicionales que se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, y las herramientas y notaciones que se usarán. Estas propuestas han demostrado ser efectivas y necesarias en un gran número de proyectos, pero también han presentado problemas en otros muchos. Una posible mejora es incluir en los procesos de desarrollo más actividades, más artefactos y más restricciones, basándose en los puntos débiles detectados. Sin embargo, el resultado final sería un proceso de desarrollo más complejo que puede incluso limitar la propia habilidad del equipo para llevar a cabo el proyecto. Otra aproximación es centrarse en otras dimensiones, como por ejemplo el factor humano o el producto *software*. Esta es la filosofía de las metodologías ágiles, las cuales dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del *software* con iteraciones muy cortas” (Penadés, y otros, 2006).

OpenUp

Esta metodología de desarrollo ágil y unificado, contiene el conjunto mínimo de prácticas que ayudan a los equipos a ser más eficiente en el desarrollo de *software*. Su selección en el desarrollo de la solución propuesta se debe a que establece una filosofía ágil que se centra en la naturaleza colaborativa del desarrollo de *software*. *OpenUp* genera la documentación necesaria para el desarrollo del proyecto, lo cual favorece las entregas del producto en el tiempo estimado.

Entre los beneficios de esta metodología se pueden citar:

- 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 ser una metodología ágil tiene un enfoque centrado en el cliente e iteraciones cortas.

OpenUP es un proceso mínimo, completo y extensible. Promueve técnicas ágiles y principios, mientras que tiene un ciclo de vida estructurado que hace hincapié en la continua entrega de *software*, que es valioso para los interesados en el desarrollo rápido de aplicaciones de calidad (Rodríguez , y otros, 2013).

1.8.2 Análisis del soporte tecnológico para el desarrollo del software

La correcta selección de las tecnologías a emplear en el desarrollo de un *software* es uno de los pasos más importantes en el ciclo de vida del mismo, debido a que influye directamente en la calidad del producto final y en el esfuerzo del equipo de desarrollo para obtenerlo.

Lenguajes de programación

PHP5

Actualmente existe gran diversidad de lenguajes de programación para desarrollar aplicaciones *web* tales como: *Java*, PHP y *Python*, los cuales permiten interactuar con los usuarios y utilizar diversos sistemas de bases de datos.

PHP es un lenguaje de programación, *Personal Home Page* (PHP por sus siglas en inglés). Este es un lenguaje multiplataforma, compatible con la mayoría de los gestores de base de datos que se utilizan en la actualidad, destaca su conectividad con *MySQL y PostgreSQL*, es un lenguaje de programación del lado del servidor utilizado para generar páginas dinámicas. Se distribuye bajo licencia libre, por lo que se presenta como una alternativa de fácil acceso para todos. No requiere definición de tipos de variables y cuenta con manejo de excepciones (García, 2010).

Este lenguaje de programación presenta como ventajas (García, 2010):

- Es completamente orientado a la *web*[5].
- Presenta gran capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados *ext* o extensiones).
- Permite las técnicas de Programación Orientada a Objetos.
- Biblioteca nativa de funciones sumamente amplia e incluida.
- PHP generalmente es utilizado como módulo de Apache, lo que lo hace extremadamente veloz.
- Está completamente escrito en C, de modo que se ejecuta rápidamente utilizando poca memoria.

Por las características anteriores se decide seleccionar PHP en su versión 5.3.10.

[5] web se utiliza en el ámbito tecnológico para nombrar a una red informática y, en general, a Internet (en este caso, suele escribirse en mayúsculas.

Su selección también está dada por el *framework*[6] de desarrollo escogido.

JavaScript

JavaScript es un lenguaje de programación utilizado para crear pequeños programas encargados de realizar acciones dentro del ámbito de una página *web*. Entre las acciones típicas que se pueden realizar en *JavaScript* se tienen dos vertientes:

1. Por un lado los efectos especiales sobre páginas *web*, para crear contenidos dinámicos y elementos de la página que tengan movimiento, cambien de color o cualquier otro dinamismo.
2. Por el otro, permite ejecutar instrucciones como respuesta a las acciones del usuario, lo cual posibilita crear páginas interactivas con programas como calculadoras, agendas, o tablas de cálculo.

JavaScript se trata de un lenguaje de programación del lado del cliente, porque es el navegador el que soporta la carga de procesamiento (Groussard, 2010).

Se decidió utilizar *JavaScript* en su versión 8 como lenguaje de programación debido a que resulta sencillo de aprender. La selección de este lenguaje viene sustentada además por la amplia documentación existente, la experiencia que poseen los desarrolladores con el mismo y porque el *framework* de diseño hace uso de él.

Lenguajes de enmarcado

HTML5

El Lenguaje de Marcas de Hipertexto, *Hyper Text Mark up Language* (HTML por sus siglas en inglés), en la versión 5, es la quinta revisión del lenguaje básico de la *Word Wide Web* (WWW). Este lenguaje de programación es usado por múltiples desarrolladores web por sus avances, mejoras y ventajas con respecto a las versiones anteriores. El principal criterio de diseño de HTML5 ha sido el de resolver problemas prácticos, y con este objetivo adopta soluciones dirigidas a facilitar el trabajo en situaciones reales (2013).

Además de las ventajas anteriores, otro criterio por el cual se seleccionó este lenguaje es porque se puede integrar con el *framework* de diseño.

[6] Framework (marco de trabajo) es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de *software* concretos, que puede servir de base para la organización y desarrollo de *software*.

CSS3

CCS3 es la nueva versión del CSS (*Cascading Style Sheet*, Hojas de Estilo en Cascada). Es simplemente el lenguaje con el cual se puede dar diseño y apariencia a las páginas HTML o XML. Es desarrollado y distribuido por la W3C (*World Wide Web Consortium*) con el fin especificado anteriormente.

A continuación se muestran algunas características que facilitan su uso.

- CSS3 propone una navegación más rápida y menores tiempos de respuesta producidos por una reducción de imágenes, las cuales ya no serán requeridas para diseñar botones o efectos de texto.
- Deja atrás una excesiva dependencia de *JavaScript* para fines de representación visual, como lo son las animaciones, dando como resultado menos código y mejor rendimiento.
- Representa una futura menor dependencia de *software* para gráficos que resultan bastante caros (Ojeda Navajas, 2012).

Otro de los motivos por las cuales se empleó este lenguaje se debe a que el *framework* de diseño lo utiliza.

Servidor web

Apache

El servidor HTTP Apache es un servidor de código abierto. Tiene amplia aceptación en la red. La mayoría de las vulnerabilidades de la seguridad descubiertas y resueltas tan sólo pueden ser aprovechadas por usuarios locales y no remotamente. Está diseñado para ser un servidor *web* potente y flexible que pueda funcionar en la más amplia variedad de plataformas y entornos. Apache se ha adaptado siempre a una gran variedad de entornos a través de su diseño modular. Para el desarrollo de la solución se decide utilizar Apache en su versión 2.2.22 pues es un servidor *web* flexible, rápido y eficiente, continuamente actualizado y adaptado a nuevos protocolos. Es multiplataforma, con los diferentes módulos de apoyo que proporciona. Incentiva la realimentación de los usuarios, obteniendo nuevas ideas, informes de fallos y parches para la solución de los mismos. Se desarrolla de forma abierta. Es extensible ya que se han desarrollado diversas extensiones entre las que destaca PHP, un lenguaje de programación del lado del servidor (Bowen, 2007).

Sistema gestor de base de datos

Un Sistema Gestor de Base de Datos (SGBD, en inglés DBMS: *Database Management System*) es un sistema de *software* que permite la definición de bases de datos; así como la elección de las estructuras de datos necesarios para el almacenamiento y búsqueda de los datos, ya sea de forma interactiva o a través de un lenguaje de programación *PostgreSQL* (Eisentraut, y otros, 2008).

PostgreSQL

*PostgreSQ*L es un sistema gestor de base de datos relacionales orientada a objetos, derivado de *Postgre*. Es un gestor de bases de datos de código abierto y multiplataforma. Soporta gran parte de la sintaxis SQL y cuenta con un extenso grupo de enlaces con lenguajes de programación. El motor de datos puede incluir las sub-consultas, los valores por defecto, las restricciones a valores en los campos y los disparadores. Permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases, eliminando la necesidad del uso de bloqueos explícitos.

Se escogió *PosgreSQL* en su versión 9.1 ya que es un sistema ampliamente popular e ideal para tecnologías *web*. Es fácil de administrar. Su sintaxis SQL es estándar y fácil de aprender. Es multiplataforma, cuenta con capacidades de replicación de datos y brinda soporte empresarial disponible (Eisentraut, y otros, 2008).

Framework de desarrollo

Un *framework* de aplicaciones web permite el desarrollo de sitios *web* dinámicos, servicios *web* y aplicaciones *web*. Su propósito es permitir a los desarrolladores construir aplicaciones *web* y centrarse en los aspectos interesantes, aliviando la típica tarea repetitiva asociada con patrones comunes de desarrollo *web*. La mayoría de los *frameworks* de aplicaciones *web* proporcionan los tipos de funcionalidad básica común, tales como sistemas de plantillas, manejo de sesiones de usuario, interfaces comunes con el disco o el almacenamiento en base de datos de contenido cacheado, y persistencia de datos. Normalmente, los *frameworks* de aplicación *web* además promueven la reutilización y conectividad de los componentes, así como la reutilización de código, y la implementación de bibliotecas para el acceso a base de datos.

Symfony2

Como rasgos descriptivos de *Symfony2* se tienen (Potencier, y otros, 2014):

- **Flexibilidad ilimitada**: cualesquiera que sean las necesidades, *Symfony2* será adaptable. Su inyector de dependencias y su manejador de eventos, lo hacen enteramente configurable, con cada una de las partes siendo completamente independientes.
- **Estable y sostenible**: la mayor parte de las versiones son soportadas 3 años por la compañía.
- **Facilidad de uso**: completamente flexible para satisfacer las necesidades de profesionales y usuarios avanzados y es muy accesible. Cuenta con documentación abundante, soporte y permiten a un principiante sentirse a gusto muy rápido.

Symfony2 es un *framework* fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas *Windows* y *Unix* estándares). Propone diferentes patrones de diseño para la *web*. Puede integrarse con diferentes gestores de bases de datos. El código resulta fácil de leer, incluye comentarios de *phpDocumentor* y que permite un mantenimiento sencillo (Potencier, y otros, 2014). Es extensible, flexible como para adaptarse a los casos más complejos. Es orientado a objetos. Por todo lo cual se escogió *Symfony en su versión 2.3.7* como *framework* de desarrollo.

Framework de diseño

Bootstrap

Bootstrap es un *framework* que simplifica el proceso de creación de diseños *web* combinando CSS y *JavaScript*. Ha sido desarrollado por *Twitter* que recientemente liberó su versión 3.x La mayor ventaja es que permite crear interfaces que se adapten a los distintos navegadores (*responsive design*) apoyándose en un *framework* potente con numerosos componentes *web* que ahorrarán mucho esfuerzo y tiempo (Genveta, 2013).

Características principales:

- *Bootstrap* ofrece una serie de plantillas CSS y ficheros *JavaScript* que permiten integrar el *framework* de forma sencilla y potente en proyectos *webs*.
- Permite crear interfaces que se adapten a los diferentes navegadores, tanto de escritorio como *tablets* y móviles a distintas escalas y resoluciones.
- Se integra perfectamente con las principales librerías *JavaScript*, por ejemplo *JQuery*.
- Ofrece un diseño sólido usando *LESS* y estándares como CSS3/HTML5.
- Funciona con todos los navegadores, incluido *Internet Explorer* usando HTML *Shim* para que reconozca los *tag* HTML5.
- Dispone de distintos *layout* predefinidos con estructuras fijas a 940 píxeles de distintas columnas o diseños fluidos (Genveta, 2013).

Se incluirá *Bootstrap* en su versión 3 en la solución propuesta para el desarrollo de las interfaces de usuario, ya que brinda disímiles facilidades en este tema.

Entornos de desarrollo Integrados (IDE por sus siglas en inglés)

Son programas que reúnen funcionalidades para el trabajo con uno o varios lenguajes de programación. Son por lo general un conjunto de herramientas empaquetadas en una sola aplicación informática, así como

un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Los IDE pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes.

Netbeans 7.4

Netbeans 7.4 es un entorno de desarrollo integrado de código abierto para desarrolladores de *software*. Cuenta con todas las herramientas necesarias para crear aplicaciones profesionales de escritorio, empresariales, *web* y aplicaciones móviles con la plataforma *Java*, así como con PHP y *JavaScript*.

NetBeans IDE 7.4 amplía el soporte avanzado de desarrollo HTML5, ofrece un nuevo soporte para el desarrollo *web* móvil. Está disponible en varios idiomas.

Se selecciona *Netbeans* 7.4 como herramienta IDE, por su soporte a tecnologías, su flexibilidad, y su potencia para el desarrollo de aplicaciones *web* (Oracle, 2013).

Lenguaje de modelado

Lenguaje unificado de Modelado (UML por sus siglas en inglés)

UML es un leguaje de modelado estandarizado de propósito general en el campo de la ingeniería de *software* orientada a objetos. UML incluye un conjunto de técnicas de notación gráfica para crear modelos visuales de programación orientada a objetos.

Los estándares propuestos por el lenguaje UML ofrecen amplias ventajas para el desarrollo de la solución (Groussard, 2010), por lo que se escoge como lenguaje de modelado a utilizar en el presente trabajo.

Herramienta de ingeniería de *software* asistida por computación

Las herramientas de ingeniería de *software* asistida por computación, también conocidas por sus siglas en inglés CASE (*Computer Aided Software Engineering*) facilitan el trabajo de ingeniería de un proyecto, aportando funcionalidades para la organización, el modelado y el seguimiento de un *software*.

Visual Paradigm

Visual Paradigm es una herramienta CASE. La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. *Visual Paradigm* ha sido concebido para soportar el ciclo de vida completo del proceso de desarrollo del *software* a través de la representación de todo tipo de

diagramas. Esta herramienta fue diseñada para una amplia gama de usuarios interesados en la construcción de sistemas de *software* libre, de forma fiable a través de la utilización de un enfoque orientado a objetos (Sommerville, 2005).

Entre sus principales características se encuentran (Sommerville, 2005):

- Entorno de creación de diagramas para UML.
- 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 múltiples versiones, para cada necesidad.
- Generación de bases de datos, transformación de diagramas de Entidad-Relación en tablas de base de datos.

Se selecciona *Visual Paradigm* en su versión 8.0 como herramienta CASE. La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. Ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del *software* a través de la representación de todo tipo de diagramas.

1.9 Conclusiones parciales

Luego de haber realizado un estudio de los conceptos y algoritmos referentes a los STI, SBC y Sistemas de recomendación se selecciona para la solución el uso del funcionamiento de los SBR, así como el método de vecinos próximos (kNN).

Se realizó un análisis del soporte tecnológico para la construcción de la solución. Como resultado del estudio realizado se seleccionó la metodología de desarrollo de *software OpenUP* para la planeación y elaboración del sistema. Se seleccionaron como lenguajes de desarrollo PHP 5.3.10 del lado del servidor y *JavaScript* 8 del lado del cliente y CSS3 para proporcionar estilo. Se eligió *Symfony 2.3.7* como *framework* para el desarrollo del sistema y *Bootstrap 3* para el diseño. Se escogió Apache 2.2.22 como servidor *web*, se optó por *PostgreSQL 9.1* como sistema gestor de base de datos. Como entorno de desarrollo integrado se decidió utilizar *NetBeans* 7.4 y para el modelado de los diagramas propuestos se definió el *Visual Paradigm 8.0.*

Capítulo 2: Características del Sistema evaluador de habilidades y de recomendación para el apoyo al proceso de enseñanza-aprendizaje

2.1 Introducción

En el presente capítulo se explican las características del sistema a desarrollar, definiendo el modelo de dominio según el objeto de estudio haciendo uso de la metodología *OpenUp*. Se identifican los requisitos funcionales y no funcionales con los que debe cumplir el sistema. Se explica además la arquitectura y el diseño del sistema a desarrollar.

2.2 Descripción actual del dominio del problema

La disciplina de Ingeniería de Software de la UCI concibió recientemente una serie de cambios en los planes de estudio de las asignaturas de la disciplina como una estrategia para apoyar el desarrollo de los estudiantes y del claustro de profesores; pero aún se busca que los estudiantes adquieran las competencias necesarias para vencer los temas de las asignaturas, así como también les sea posible desenvolverse, valiéndose de estas, en la esfera productiva; y que a la vez puedan desarrollar una actitud de autonomía.
Existe el Entorno Virtual de Aprendizaje (EVA), el cual permite el estudio de manera autónoma de los contenidos de cada una de las asignaturas de a disciplina. Incorporando a docentes y alumnos a un sistema que les permite evaluar los conocimientos del estudiante.
El EVA asesora en línea a la gran mayoría de los estudiantes de la universidad, estos sistemas permiten no sólo la ubicación de recurso sino también las diversas formas de comunicación entre los miembros de la comunidad de aprendizaje. Se propicia el aprendizaje en grupos llamados comunidades, además del aprendizaje individual. Sin embargo no satisface todas las necesidades del modelo pedagógico utilizado en la disciplina Ingeniería de Software. A pesar de sus numerosas ventajas esta plataforma propone las actividades de forma general a todos los estudiantes, sin atender a las particularidades de cada uno, no recomienda ejercicios según el nivel de aprendizaje del estudiante, sino que los cuestionarios y ejercicios a resolver están disponibles para todos y es el usuario quien lo selecciona.
Con la creación de un sistema evaluador de habilidades y de recomendación se pretende obtener un ambiente alternativo de evaluación, que les permita a los estudiantes trabajar de manera independiente, profundizando en sus debilidades y en los temas que son de su interés, además de evaluar de manera efectiva el desarrollo de las habilidades de forma autónoma. Un sistema que facilite el proceso de gestión de evaluación para aprendizaje ajustándose a las necesidades de aprendizaje de los estudiantes y que sea

capaz de generar posibles planes de actividades evaluativas para los distintos niveles y fases del aprendizaje de forma personalizada. De acuerdo a su ruta de aprendizaje el usuario accede a los recursos debidamente clasificados con tales fines.

2.3 Propuesta de solución

Se tiene como propósito desarrollar un sistema evaluador de habilidades y de recomendación que contribuya de manera positiva al proceso de enseñanza-aprendizaje. Dicha solución permitirá a los usuarios matricularse virtualmente en asignaturas de la disciplina de IGSW; y realizar ejercicios que estén en correspondencia con su nivel de dominio en la asignatura, así como también con su coeficiente de aprendizaje.

Entre las características principales del sistema se encuentran.

- Permitir darle la bienvenida al asistente.
- Permitir realizar diferentes diagnósticos y encuestas al usuario mediante un módulo de Diagnóstico General.
- Permitir conocer los conocimientos que posee el usuario en las asignaturas de la disciplina de IGSW mediante un módulo de Diagnóstico.
- Permitir obtener un perfil del usuario con los datos obtenidos en los dos módulos anteriores.
- Permitir recomendar actividades y evaluar las mismas a los diferentes usuarios.

Este sistema debe llevar a cabo el procesamiento de la información a partir de un conjunto de encuestas iniciales respondidas por el usuario, además de determinar el estado actual del mismo a la vez que genere la ruta de autoaprendizaje a seguir durante el entrenamiento; podrá además evaluar las respuestas de los estudiantes y sugerir nuevos ejercicios que incrementen el conocimiento que este tenía en el nivel de partida, mejorando la calidad en el proceso de formación académica en esta disciplina. Para ello utilizará el uso del funcionamiento de los SBR, para evaluar los ejercicios y determinar el nivel de cada estudiante; y el método de vecinos próximos (kNN), para según el nivel en que se encuentra cada estudiante, recomendarle nuevos ejercicios.

Es importante garantizar un sistema automatizado, en línea 24 horas, donde el usuario pueda completar una ruta de aprendizaje sin la imprescindible intervención de un profesor (tutor, guía o asesor físico).

El sistema contará con una interfaz *web* para la autenticación del usuario y la presentación de las asignaturas, permitiendo esto que sea más comprensible para el usuario del sistema. Los resúmenes de

evaluación ofrecerán a directivos o personal del departamento nuevos elementos para apoyar la toma de decisiones dentro del mismo.

De manera general su funcionamiento se implementará en código PHP. El sistema servirá para analizar el nivel de conocimiento que tiene un estudiante sobre la materia, así como el ritmo de su aprendizaje, proponer y evaluar ejercicios, incrementando el nivel y almacenar los datos para futuros análisis.

2.4 Entendimiento del negocio

Para el modelado de negocio, se utiliza el modelo del dominio; el cual está constituido por conceptos y sus relaciones. El mismo está compuesto por las entidades que se han interpretado y descubierto en el análisis del negocio. Posteriormente este diagrama conceptual será traducido a un diagrama de clases. Según Craig Larman en la 2da edición del libro UML y Patrones "un modelo del dominio es una representación de las clases conceptuales del mundo real, no de componentes *software*. No se trata de un conjunto de diagramas que describen clases *software*, u objetos *software* con responsabilidades" (Larman, 2003).

La figura 6 muestra el diagrama de modelo de dominio que representa las relaciones que existen entre los principales conceptos asociados al tema de investigación. Un estudiante tiene un perfil de usuario y se matricula en una o varias asignaturas, además al estudiante se le recomiendan ejercicios a partir de la evaluación de otros ejercicios que haya resuelto con anterioridad. Las asignaturas a su vez, tienen objetivos y habilidades que el estudiante debe vencer. Cada ejercicio pone a prueba los objetivos y habilidades que le corresponden. El estudiante recibe una evaluación por cada ejercicio que resuelva y a partir de esta se recomiendan otros de mayor o menor nivel.

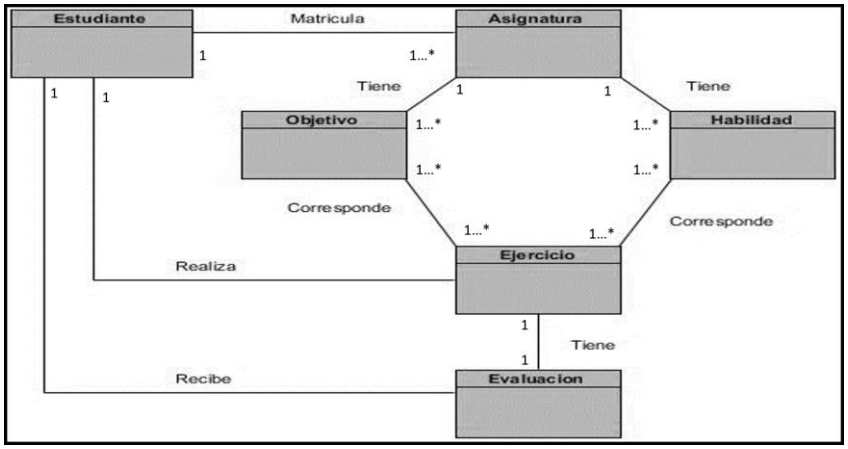


Figura 5: Modelo de dominio.

2.4.1 Descripción de los conceptos asociados al dominio

- **Usuario:** está compuesto por los estudiantes, que se hayan autenticado previamente en el sistema y hayan resuelto el test de capacidad.
- **Ejercicio:** es una actividad relacionada a la disciplina de IGSW que va dirigida a los usuarios y que estos pueden resolver en dependencia de su nivel de dominio en la materia y que es evaluado a partir de las habilidades y objetivos que deben vencer, para recomendar otro ejercicio de mayor o menor nivel.
- **Asignatura:** es la disciplina de IGSW (Sistemas de Bases de Datos I, Sistemas de Bases de Datos II, Ingeniería de Software I, Ingeniería de Software II y Gestión de Software) a la cual corresponden los ejercicios que pueden resolver los usuarios.
- **Objetivos:** son los aspectos básicos que un estudiante debe dominar de una materia.
- **Habilidades:** son las destrezas que un estudiante debe desarrollar en la resolución práctica de un ejercicio.
- **Evaluación:** es la valoración que brinda el sistema sobre la respuesta que el usuario da a un ejercicio.

2.5 Características del sistema

2.5.1 Técnicas de obtención de requisitos

Para la realización de este trabajo se empleó la técnica de entrevista en la obtención de los requisitos, tanto funcionales como no funcionales, los cuales permitieron comprender el dominio del sistema, buscar y recolectar información para definir sus límites y restricciones, así como identificar a las personas involucradas en el mismo.

2.5.2 Requisitos funcionales (RF)

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir. Se mantienen invariables sin importar con qué propiedades o cualidades se relacionen y muestran cómo los casos de uso serán llevados a la práctica (Pressman, 2005).
Las entrevistas realizadas arrojaron un total de 30 requisitos funcionales, que a continuación se presentan:

Tabla 2. Listado de requisitos funcionales

RF1. El sistema debe permitir iniciar sesión de usuario.	**RF17**. El sistema debe permitir modificar objetivos.
RF2. El sistema debe permitir finalizar la sesión de usuario.	**RF18**. El sistema debe permitir modificar habilidades.
RF3. El sistema debe permitir autenticar un usuario.	**RF19**. El sistema debe permitir eliminar objetivos.
RF4. El sistema debe asignar un rol.	**RF20**. El sistema debe permitir eliminar habilidades.
RF5. El sistema debe permitir a los usuarios matricular asignaturas.	**RF21**. El sistema debe permitir crear un nuevo ejercicio.
RF6. El sistema debe permitir mostrar asignaturas matriculadas.	**RF22**. El sistema debe permitir asignar el tipo de ejercicio.
RF7. El sistema debe permitir a los usuarios resolver ejercicios.	**RF23**. El sistema debe permitir asignar complejidad a un ejercicio.
RF8. El sistema debe permitir mostrar objetivos vencidos por asignaturas.	**RF24**. El sistema debe permitir evaluar ejercicios.
RF9. El sistema debe permitir mostrar habilidades vencidas por asignaturas.	**RF25**. El sistema debe permitir modificar ejercicios.
RF10. El sistema debe permitir mostrar objetivos sin vencer por asignaturas.	**RF26**. El sistema debe permitir eliminar ejercicios.
RF11. El sistema debe permitir mostrar habilidades sin vencer por asignaturas.	**RF27**. El sistema debe permitir ver ejercicios recomendados.
RF12. El sistema debe permitir crear asignaturas.	**RF28**. El sistema debe permitir al usuario observar el resumen de sus evaluaciones.
RF13. El sistema debe permitir editar asignaturas.	**RF29.** El sistema debe permitir al profesor observar el resumen de evaluaciones de los usuarios.
RF14. El sistema debe permitir eliminar asignaturas.	**RF30.** El sistema debe permitir recomendar ejercicios a los estudiantes según su nivel y a partir de los objetivos y habilidades que este haya vencido.
RF15. El sistema debe permitir insertar objetivos.	
RF16. El sistema debe permitir insertar habilidades.	

2.5.2 Requisitos no funcionales (RNF)

Como parte de la captura de requisitos no funcionales se identificaron un total de 16, agrupados en 8 categorías:

Seguridad

RNF1. Para proteger el acceso a la información se definirá un mecanismo de autenticación basado en usuario y contraseña.

RNF2. Para el almacenamiento de la contraseña en la base de datos se empleará cifrado SHA512.

RF3. Se empleará el modelo de control de acceso basado en roles.

RF4. El flujo de información entre el usuario y la aplicación se realizará mediante el protocolo HTTPS diseñado para resistir ataques *MAN-IN-THE-MIDDLE*.

Usabilidad

RNF5. La aplicación *web* tendrá una interfaz intuitiva.

Confiablidad

RNF6. La información no podrá ser modificada por usuarios no autorizados, protegiendo así la integridad de los datos.

Licencia

RNF7. Se requiere el uso de herramientas y tecnologías de *software* libre, las cuales se podrán usar, modificar y distribuir libremente.

Diseño e Implementación

RNF8. Se requiere el uso de PHP 5.3.10 en adelante como lenguaje de programación del lado del servidor.

RNF9. Se requiere el uso de *Symfony* 2.3.7 como *framework* para el desarrollo y *Bootstrap* 3 para el diseño de la interfaz de usuario.

RNF10. Se requiere *PosgreSQL* 9.1 como gestor de bases de datos para los datos administrativos.

RNF11. Se requiere el uso de Apache2 como servidor *web*.

Hardware

RNF12. Para garantizar un buen funcionamiento del sistema, el servidor donde estará desplegado deberá cumplir como mínimo las siguientes condiciones: ser Pentium IV o sus equivalentes de otras compañías, con una velocidad de 3.00 GHz, 160 GB de disco duro y 3 GB de RAM, esta última restricción está condicionada a la cantidad de tráfico generada por la concurrencia de usuarios. Los requerimientos de hardware para los clientes, son mínimos, será posible utilizar un cliente ligero.

Software

RNF13. Para el cliente como versión más antigua del navegador: *Mozilla Firefox* 25 o superior, *Google* Chrome 15 o superior, Opera 9 o superior e *Internet Explorer* 11 o superior.

RNF14. Para el servidor *web* Apache 2.2 o superior con módulo PHP 5.x disponible.

Eficiencia

RNF15. El sistema debe ser capaz de responder con rapidez a las peticiones de los usuarios, demorando como promedio en una transición de 1 a 5 segundos.

RNF16. El sistema debe permitir trabajar de manera concurrente a 150 usuarios.

Actores del sistema

Tabla 3. Actores del sistema

Actores	Descripción
Usuario	Persona que tendrá acceso al sistema como un cliente del mismo.
Administrador del sistema	Persona encargada de realizar cualquier modificación en el sistema teniendo acceso total al mismo.

2.6 Descripción de la solución

2.6.1 Diagrama de casos de uso del sistema

Para la descripción de la solución se utilizan diagramas de casos de uso, ya que representan la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios u otros sistemas. Se utilizan para ilustrar los requerimientos del sistema al mostrar cómo reacciona a eventos que se producen en su ámbito o en él mismo. A continuación se presenta el diagrama de casos de uso de la aplicación (ver figura 7).

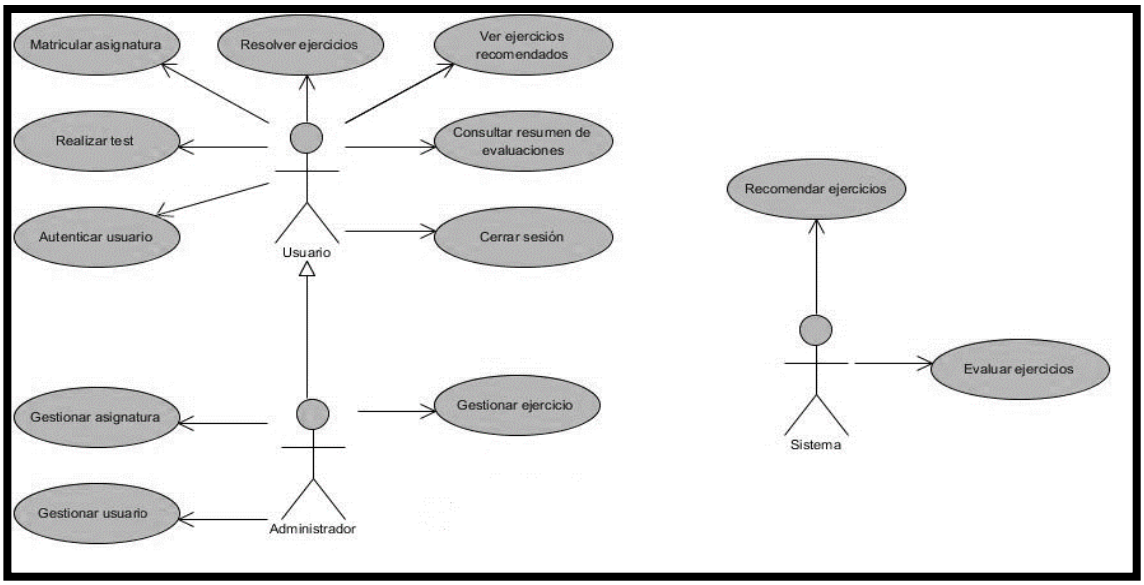


Figura 6: Diagrama de Casos de Uso del sistema.

2.6.2 Especificación de casos de uso del sistema

CU "Autenticar Usuario".

Tabla 4. Descripción del caso de uso: Autenticar usuario

Objetivo	Autenticar usuario en el sistema
Actores	Usuario
Resumen	El CU se inicializa cuando el actor (en lo adelante usuario) desea acceder al sistema
Complejidad	Baja
Prioridad	Crítico
Precondiciones	El usuario accede a la dirección de la aplicación web
Postcondiciones	Se accede al sistema según el Rol que posea el usuario

Prototipo	

Flujo de eventos

	Actor	Sistema
1.	Escribe la dirección de la aplicación web en el navegador o accede mediante un link	
2.		Muestra un formulario de autenticación con los siguientes campos: • Usuario • Contraseña
3.	Introduce los datos	
4.	Pulsa la opción “Login”	
5.		Verifica que los datos introducidos son correctos
6.		Autentica en la aplicación
7.		Redirecciona a la vista referente al rol que posea
8.		Termina el Caso de Uso

Flujos alternos

1 Campos vacíos

	Actor	Sistema
1.		Verifica que los campos están vacíos
2.		Se muestra el formulario con los mismos campos y el mensaje: "!!Error!! Por favor verifique sus credenciales"
Flujos alternos		
2 Usuario o contraseña incorrectos		
	Actor	**Sistema**
1.		Verifica que los datos son incorrectos.
2.		Se muestra el formulario con los mismos campos y el mensaje: "!!Error!! Por favor verifique sus credenciales"
Relaciones	**CU incluidos**	
	CU extendidos	Solicitar cuenta: Paso 1 del Flujo Básico

CU "Matricular Asignatura".

Tabla 5. Descripción del caso de uso: Matricular asignatura

Objetivo	Matricular asignatura en el sistema
Actores	Usuario
Resumen	El CU se inicializa cuando el actor (en lo adelante usuario) desea matricular una asignatura en la aplicación
Complejidad	Media
Prioridad	Crítico
Precondiciones	El usuario accede a la dirección de la aplicación web

Postcondiciones	Se matricula la asignatura
Prototipo	

Flujo de eventos

	Actor	Sistema
1.	Pulsa la opción “Matricular en Asignatura” del menú principal	
2.		Muestra en la vista una lista con las asignaturas disponibles
3.	Selecciona la asignatura de la lista	
4.	Pulsa la opción “Matricular”	
5.		Redirecciona a la vista principal
6.		Termina el CU

2.6.3 Descripción de la arquitectura de la solución

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de *software*. En otras palabras, brindan una solución ya probada y documentada a problemas de desarrollo de *software* que están sujetos a contextos similares. Se deben tener presente los siguientes elementos de un patrón: su nombre, el problema (cuando aplicar un patrón), la solución (descripción abstracta del problema) y las consecuencias (costos y beneficios) (Kabytes, 2010).

Los Patrones Generales de Software para Asignar Responsabilidades (GRASP del inglés *General Responsibility Assignment Software Patterns*) tienen una importante utilidad en el diseño de una aplicación al igual que los *Gang-of-Four* o Pandilla de los Cuatro (GoF por sus siglas en inglés). Los empleados en la solución de la problemática planteada, se exponen a continuación:

- Patrones GRASP:

Symfony2 utiliza el patrón **experto** con la inclusión de Doctrine para el mapeo de base de datos. Se utiliza específicamente para crear una capa de abstracción en el modelo, encapsular toda la lógica de los datos y generar las clases entidades con todas las funcionalidades comunes (GET, SET y el constructor de la entidad); las clases de abstracción de datos poseen un grupo de funcionalidades que están relacionadas directamente con la entidad que representan y contienen la información necesaria de la tabla asociada.
En la solución implementada se pone de manifiesto este patrón en la clase Usuario, cuando se añade un nuevo usuario, pues el sistema debe generar el "código de seguimiento" para dicho usuario y lo hace mediante esta clase la cual posee la información del mismo.
El patrón **creador** se utiliza para la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento.
Este patrón es utilizado principalmente para el trabajo con las entidades que se almacenan en la base de datos, un ejemplo de su utilización es cuando el usuario solicita la creación de una cuenta, la clase *UsuarioController* es la encargada de instanciar los métodos de la entidad Usuario, que luego se almacenará en la base de datos, debido a que obtiene todos los datos necesarios para la inicialización de los mismos.
El **bajo acoplamiento** se evidencia en el proyecto completo, ya que todas las clases controladoras heredan únicamente de la clase *Controller* para alcanzar un bajo acoplamiento de clases y no existe otra dependencia o herencia entre clases en la implementación.
El patrón de **alta cohesión** se puede observar claramente en el sistema ya que cada clase controladora se ajusta a manejar solo las responsabilidades correspondientes a las entidades con las que se relaciona, además para cada página cliente existe una página servidora encargada de manejar sus solicitudes poniéndose de manifiesto el patrón de diseño alta cohesión.
El patrón **controlador** se pone de manifiesto en todo el sistema ya que cada uno de los eventos generados por el usuario es redirigido a una clase controladora para que realice las operaciones solicitadas.

- Patrones **GoF**:

La solución muestra la utilización del patrón **observador** en los objetos de tipo Objetivo, ya que la clase entidad Objetivo, presenta una relación uno-a-mucho con la entidad Asignatura y de esta forma si se produce algún cambio en un objetivo, se notificará a la entidad de Asignatura correspondiente.
Symfony2 utiliza el patrón **decorador** en la creación de plantillas generales para que otras las utilicen. Este patrón se utiliza en la solución con la creación de la platilla base.html.twig, la cual posee en su contenido el

código que será general para todas las interfaces de la solución, evitando de esta forma, tener que copiar este código en cada plantilla de la misma (Larman, 2003).

- Patrón arquitectónico implementado por Symfony 2

Symfony2 es un *framework* diseñado para optimizar, proporcionando varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación *web*. *Symfony2* está fundamentado en un patrón clásico del diseño *web* conocido como arquitectura modelo vista controlador (MVC). El modelo representa la lógica de negocio, la vista transforma el modelo en una página *web* que permite al usuario interactuar con ella y el controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista. En el sistema, el modelo es la representación específica de la información y gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, el controlador responde a eventos (usualmente acciones del usuario) e invoca peticiones al 'modelo' cuando se hace alguna solicitud sobre la información.

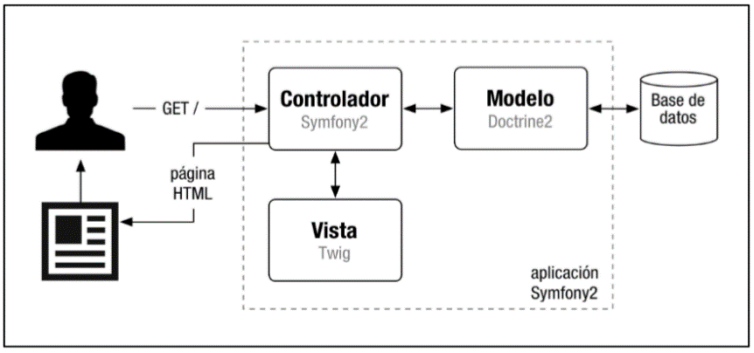


Figura 7: Diagrama del patrón arquitectónico MVC

"Symfony2 no es un *framework* MVC. Symfony2 sólo proporciona herramientas para la parte del Controlador y de la Vista. La parte del Modelo es responsabilidad del desarrollador". La separación de la vista y el modelo trae ventajas; es posible tener diferentes representaciones de la misma información, haciendo uso del mismo código dentro del modelo. Es posible además programar el código del modelo, abstrayéndose de la representación visual que se le dará a la información (Sebastián, 2010).

- Estructura en bundles de la aplicación

Los bundles son la base de la nueva filosofía de trabajo de Symfony2. El código de las aplicaciones y el propio código fuente de Symfony2 se estructura mediante bundles. Técnicamente, un bundle es un directorio que contiene todo tipo de archivos dentro una estructura jerarquizada de directorios. Los bundles de las

aplicaciones Symfony2 suelen contener clases PHP y archivos *web* (JavaScript, CSS e imágenes). No obstante, no existe ninguna restricción sobre lo que puedes incluir dentro de un bundle. Tampoco existen límites técnicos sobre el tamaño que puede llegar a tener un bundle (Eguiluz, 2013).

Para el desarrollo de la solución al problema planteado se decide crear un proyecto de Symfony2 que consta de dos bundles:

SistemaBundle: Es el encargado de almacenar toda la información necesaria para gestionar los datos que maneja la aplicación.

LdapBundle: Es el encargado realizar las autenticaciones a partir de roles definidos.

2.7 Modelo de datos

El modelo físico de la base de datos, permite que un ingeniero del *software* identifique objetos de datos y sus relaciones mediante una notación gráfica. En el contexto del análisis estructurado, define todos los datos que se introducen, se almacenan, se transforman y se producen dentro de una aplicación. Es específicamente útil para aplicaciones en donde los datos y las relaciones que gobiernan los datos son complejos (Pressman, 2005).

La figura 9 muestra la estructura de la base de datos, la cual contiene un total de 19 tablas, de las cuales 13 representan las entidades del sistema y el resto las relaciones de muchos-a-muchos que se establecen entre algunas de ellas. En la figura también pueden apreciarse las columnas o atributos de las mismas.

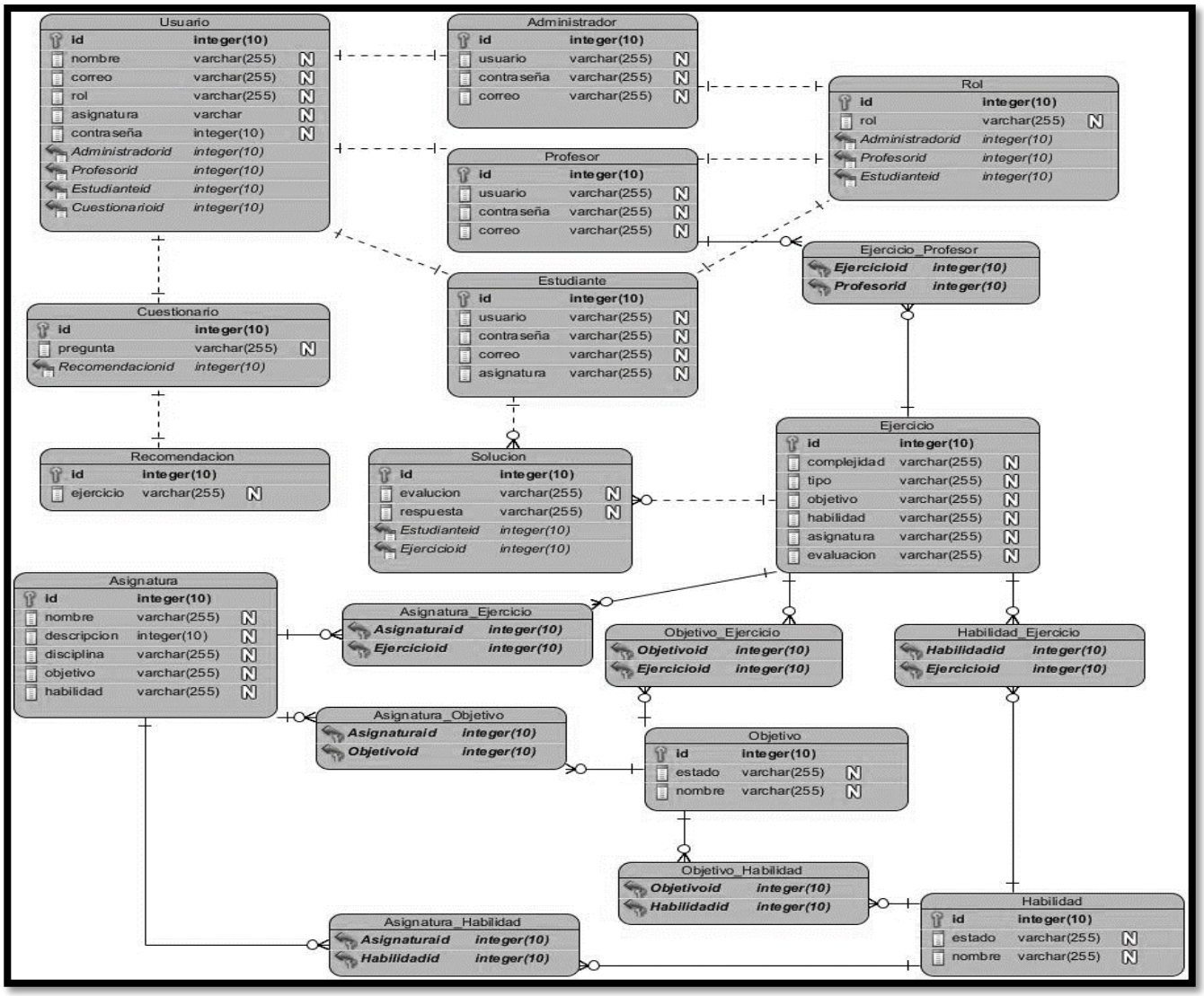


Figura 8: Modelo físico de la base de datos

2.8 Diagrama de clases del diseño

Los diagramas de clases del diseño permiten describir gráficamente las especificaciones de las clases del *software*. Muestran las clases (descripciones de objetos que comparten características comunes) que componen el sistema y cómo se relacionan entre sí (Pressman, 2005).

Las siguientes figuras muestran los diagramas de clases del diseño con estereotipos web (DCDEW) de los caso de uso expuestos anteriormente en la especificación de casos de uso.

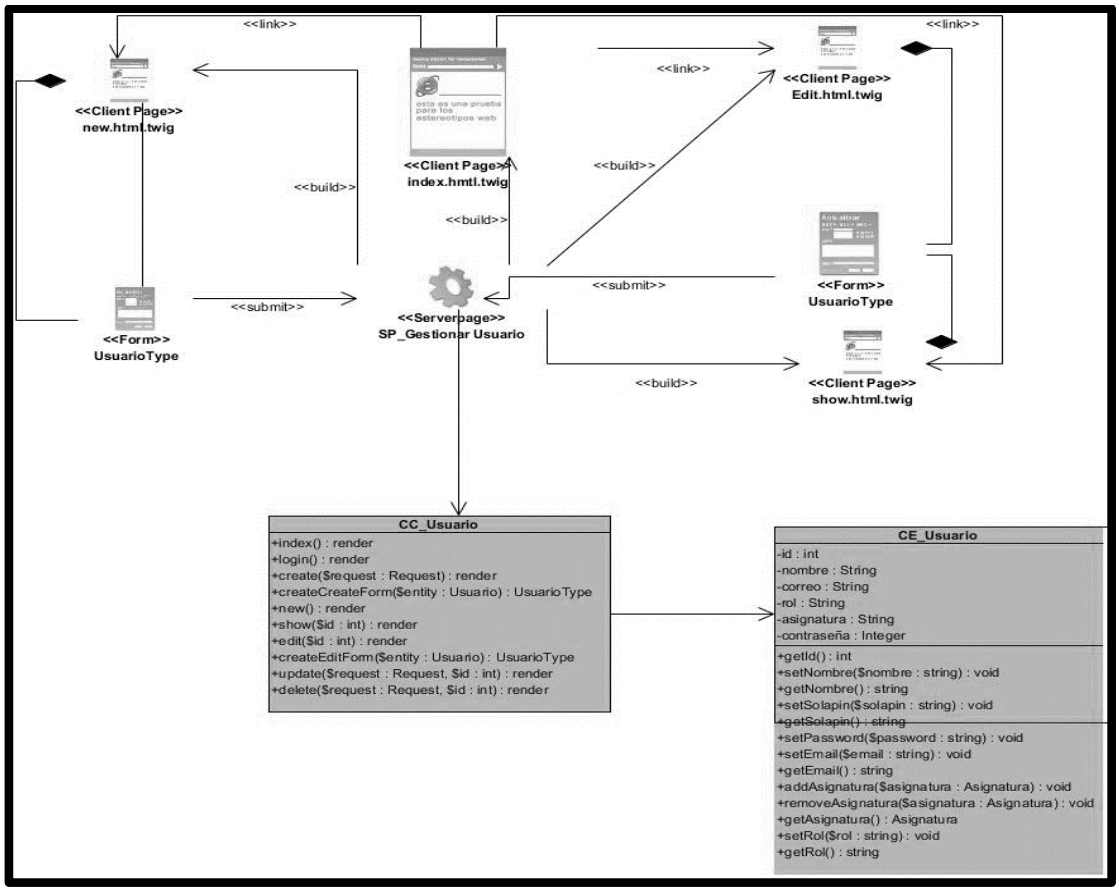


Figura 9: DCDEW para el CU Gestionar Usuario

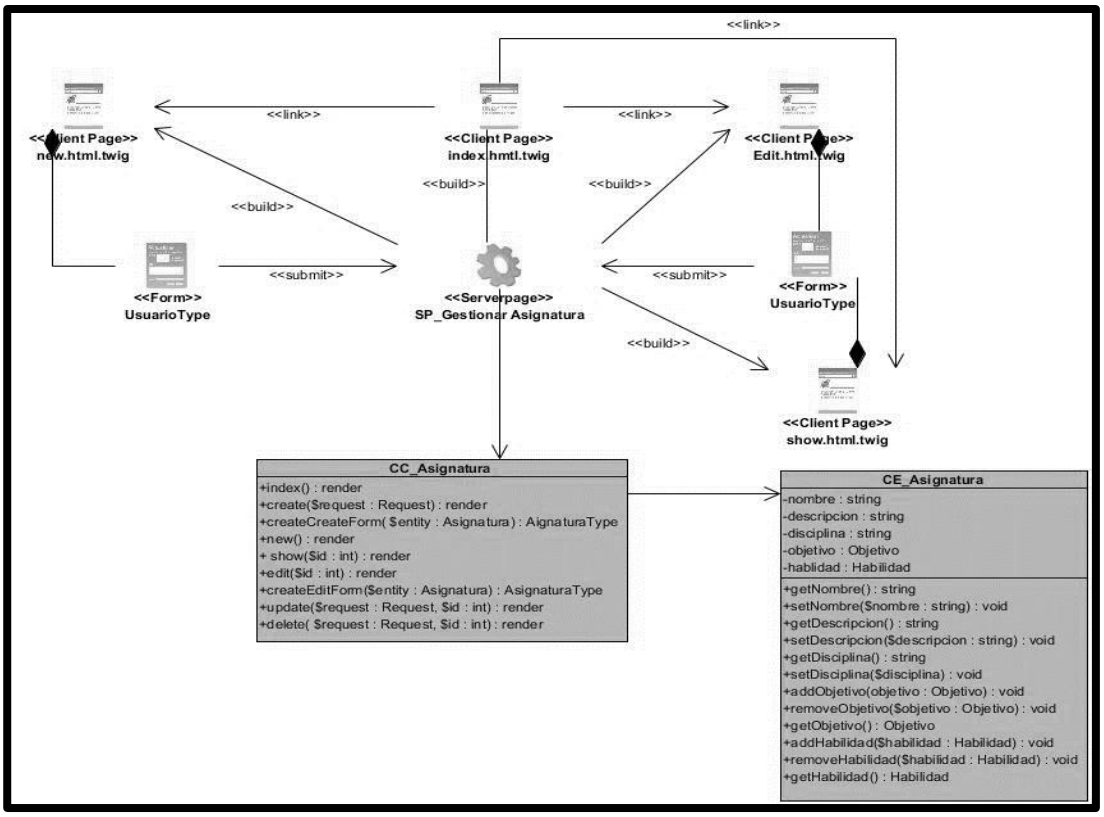


Figura 10: DCDEW para CU Gestionar Asignatura

2.9 Conclusiones parciales

En el presente capítulo se expuso la propuesta de solución, que permitió visualizar las principales funcionalidades del mismo para una mejor comprensión de la propuesta actual. Se estructuró el modelo de dominio, describiendo los conceptos fundamentales que en él se exponen, lo que permitió mostrar una visión del proceso encaminado a satisfacer las necesidades existentes. Se listaron los requerimientos funcionales y no funcionales que debe cumplir el *software* para satisfacer las necesidades del cliente, lo que constituye un paso de gran importancia para las próximas etapas de desarrollo.

La modelación de los diferentes diagramas de clases del diseño, casos de uso, y modelo físico de la base de datos, junto con el eficiente levantamiento de los requisitos, ayudaron al equipo de desarrollo a entender con más facilidad el problema planteado. Los patrones de diseño escogidos, propiciaron una arquitectura sólida y robusta al sistema.

Capítulo 3: Implementación y prueba del Sistema evaluador de habilidades y de recomendación para el apoyo al proceso de enseñanza-aprendizaje

3.1 Introducción

En el presente capítulo se detallará el funcionamiento del sistema propuesto para dar solución a la problemática planteada. Durante la implementación se lograrán las tareas de programación definidas para darle cumplimiento al desarrollo de la solución propuesta, además se presentarán aspectos como estándares de codificación utilizados durante la misma.

Durante la etapa de prueba se persigue encontrar errores cometidos al realizar el diseño y construcción de un producto de *software*. En este capítulo se detallan los principales resultados de la realización de pruebas de carga y estrés, seguridad y funcionales.

3.2 Modelo de implementación

3.2.1 Diagramas de componentes

Los diagramas de componente describen los elementos físicos del sistema y sus relaciones. Los componentes representan todos los tipos de elementos *software* que entran en la fabricación de aplicaciones informáticas. Pueden ser simples archivos, paquetes o bibliotecas cargadas dinámicamente. Las relaciones de dependencia se utilizan en los diagramas de componentes para indicar que un componente utiliza los servicios ofrecidos por otro componente (Sommerville, 2005).

A continuación se muestra el diagrama de componente general del sistema:

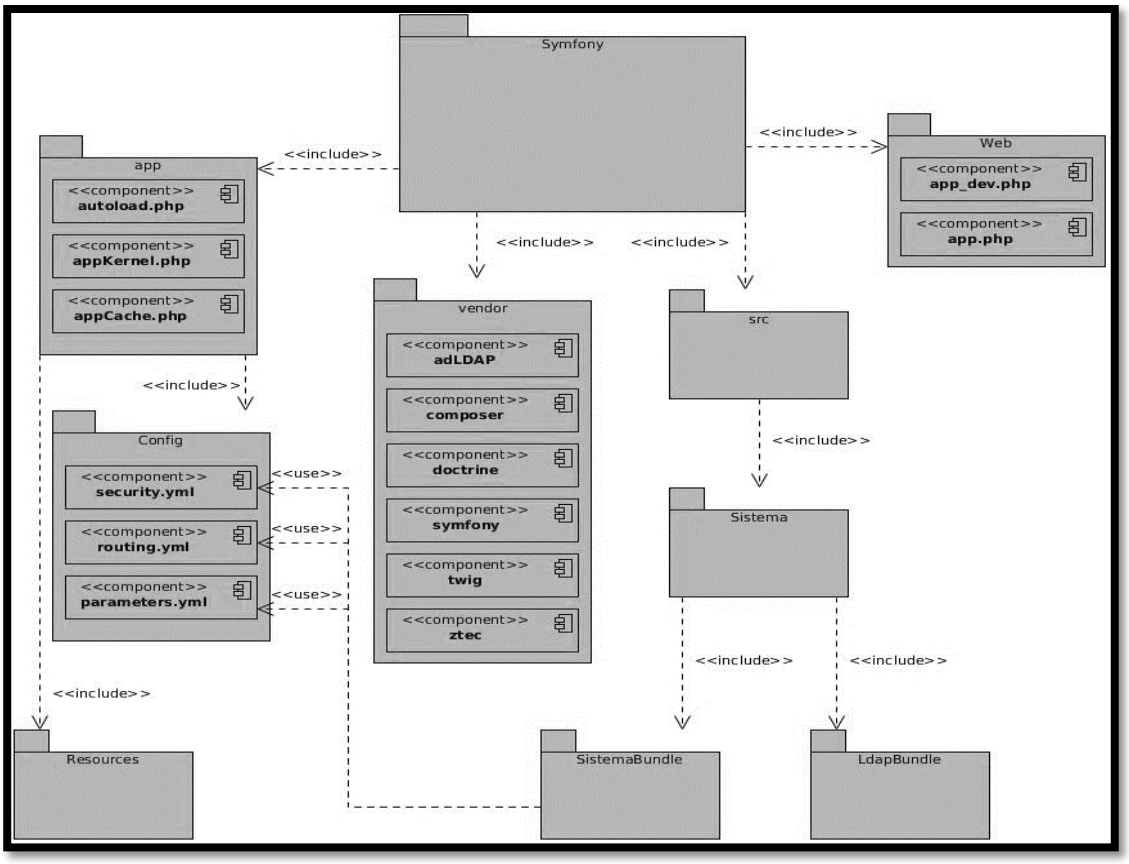


Figura 11: Diagrama de componente

3.2.2 Diagrama de despliegue

El modelo de despliegue es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo. Un nodo es un elemento físico que existe en tiempo de ejecución y representa un recurso computacional, que generalmente tiene memoria y a menudo, capacidad de procesamiento. Los nodos se utilizan para modelar la topología del *hardware* sobre el que se ejecuta el sistema. Representan un procesador o un dispositivo sobre el que se pueden desplegar los componentes. La relación entre un nodo y el componente que despliega puede mostrarse con una relación de dependencia (Larman, 2003).

El siguiente diagrama de despliegue muestra la distribución de física del sistema mediante nodos. Las relación de dependencia entre los nodos PC_Cliente y el Servidor web Apache2 utiliza protocolo HTTPS

por el puerto 443, mientras que la comunicación entre el Servidor web Apache2 y Servidor PostgreSQL ocurre a través del protocolo TCP por el puerto 5432.

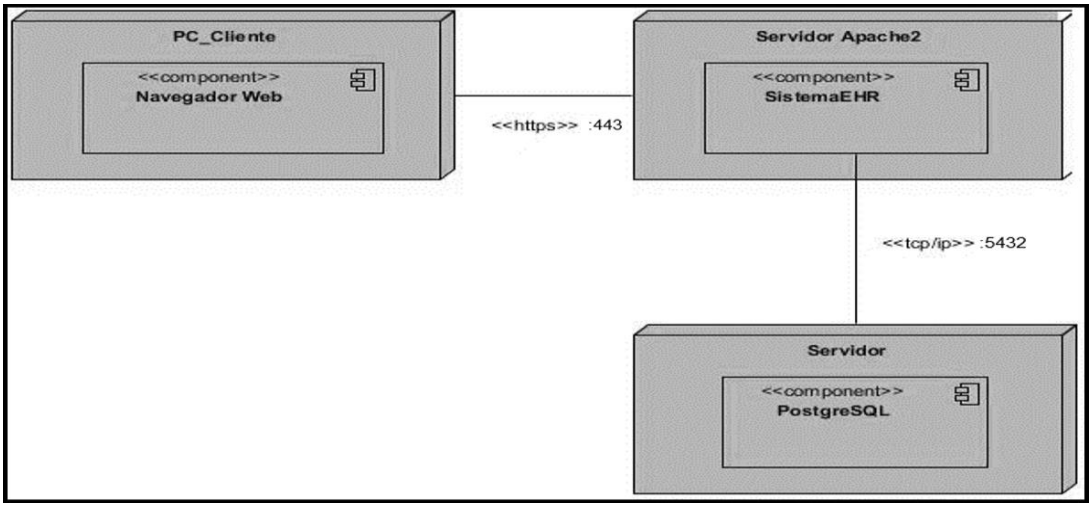


Figura 12: Diagrama de despliegue.

3.3 Estándares de codificación

Un estándar es un modelo, norma, patrón, referencia o la especificación que regula la realización de ciertos procesos o la fabricación de componentes para garantizar la interoperabilidad. En la implementación de la aplicación *web* a desarrollar, se utilizarán varios estándares de codificación, que certificarán legibilidad y organización al código de la misma, simplificando esfuerzos a la hora de darle mantenimiento y seguimiento a la aplicación.

A continuación se especifican los estándares de codificación a utilizar en la construcción de la solución:

Camel Case: la notación "*Camel Case*" consiste en escribir frases o palabras compuestas eliminando los espacios intermedios y poniendo en mayúscula la primera letra de cada palabra (Lago, 2008).

Lower Camel Case: similar al *Camel Case* sólo que la primera letra de la primera palabra es también en minúscula.

Upper Camel Case: similar al *Camel Case* siendo la primera letra de la primera palabra en mayúscula.

El nombre de las clases debe ser declarado utilizando el estilo *Upper Camel Case* y cuando en un fichero se define el espacio de nombre (*namespace*) debe existir una línea en blanco a continuación de la declaración. Además cuando están presentes todas las declaraciones de uso se escriben a continuación de la definición del *namespace* y se debe utilizar la palabra reservada *use* una vez por cada declaración además debe concluir con una línea en blanco.

```
namespace Sistema\SistemaBundle\Entity;

use Doctrine\ORM\Mapping as ORM;

/**
 * @ORM\Table (name="asignatura")
 * @ORM\Entity
 */
class Asignatura {
```

Figura 13: Nombre de clases, namespace y use.

El nombre de las funciones debe ser declarado utilizando el estilo *Lower Camel Case*.

```
class DefaultController extends Controller
{
    public function indexAction($name)
    {
        return $this->render('SistemaBundle:Default:index.html.twig', array('name' => $n
    }

    public function estaticaAction($nombre)
    {


        return $this->render('SistemaBundle:Estaticas:'.$nombre.'.html.twig');

    }
}
```

Figura 14: Nombre las funciones en Lower Camel Case.

3.4 Seguridad en el sistema

En el desarrollo de una aplicación web, la seguridad del sistema es uno de los aspectos más importantes, ya que es la garantía de la confidencialidad, integridad y disponibilidad de la información. Muchas aplicaciones web se ven afectadas en numerosas ocasiones por diferentes vulnerabilidades del propio código fuente o por descuido de los desarrolladores. En este sentido Symfony2 implementa una serie de

mecanismos para garantizar una aplicación segura, definiendo una lista de control de acceso, en el fichero app/config/security.yml.

Además de la seguridad que proporciona el *framework* utilizado, se estableció el mecanismo de autenticación al sistema basado en usuario y contraseña, además, para esta última se empleó un algoritmo de cifrado para almacenarla en la base de datos y se utilizó el modelo de control de acceso basado en roles, con estos procedimientos se asegura que sólo un usuario correctamente identificado pueda acceder al sistema, garantizando la confidencialidad e integridad de los datos, y estos usuarios solo podrán accionar sobre la información precisa que por el rol establecido en el sistema pueden controlar, también se estableció el protocolo HTTPS para asegurar el flujo de información entre el usuario y la aplicación.

3.5 Pruebas de software

Las pruebas son un conjunto de actividades que se planean con anticipación y se realizan de manera sistemática. Por tanto, se debe definir una plantilla para las pruebas de *software* (un conjunto de pasos en que se puedan incluir técnicas y métodos específicos del diseño de casos de pruebas) (Pressman, 2005).

3.5.1 Pruebas funcionales

Las pruebas funcionales están basadas en la ejecución, revisión y retroalimentación de las funcionalidades previamente diseñadas para el *software*. Las mismas se hacen mediante el diseño de modelos de prueba que buscan evaluar cada una de las opciones con las que cuenta el paquete informático. Son pruebas específicas, concretas y exhaustivas para comprobar y validar que el *software* hace lo que debe y sobre todo, lo que se ha especificado (Adame, 2009).

Se ejecutaron pruebas de este tipo para todos los casos de uso con diferentes entradas del usuario, con el objetivo de determinar que los resultados obtenidos fueran los esperados bajo cualquier situación, y así, dar por cumplidos los requerimientos del cliente. A continuación se presenta una muestra de los casos de pruebas que se elaboraron para cada caso de uso:

Tabla 6. Descripción de las variables (elaboración propia)

Nº	Nombre de campo	Clasificación	Valor nulo	Descripción
1	Usuario	Campo de texto	No	Se debe especificar un nombre de usuario válido.
2	contraseña	Campo de texto	No	Se debe especificar una frase como contraseña.

Tabla 7. Prueba funcional para el CU "Autenticar usuario"

Escenario	Descripción	Variable: usuario	Variable: contraseña	Respuesta del sistema	Flujo central
EC 1.1 Autenticar usuario correctamente	El usuario introduce los valores correctamente en el sistema	V	V	Almacena la información (la primera vez que el usuario se autentica en el sistema) o comprueba sus datos.	El usuario llena todos los datos y da clic en el botón "Entrar"
EC 1.2 Campo(s) incorrecto(s)	El usuario ingresa datos incorrectos	I	I	Selecciona el campo(s) y muestra el mensaje "!!Error!! Por favor verifique sus credenciales".	
		I	V		
		V	I		

Las celdas de la tabla contienen V, I, o N/A; V indica válido, I indica inválido, y N/A que no es necesario proporcionar un valor del dato en este caso, ya que es irrelevante.

Estas pruebas fueron realizadas a la aplicación, arrojando como resultado un total de 11 no conformidades agrupadas en 4 tipos: Error-Idioma, Redacción-Aplicación, Ortografía-Aplicación y Funcionalidad. La ejecución de estas pruebas se distribuyó en 2 iteraciones, en la primera iteración se encontraron las 10 no conformidades quedando resueltas para la segunda iteración (Ver figura 16).

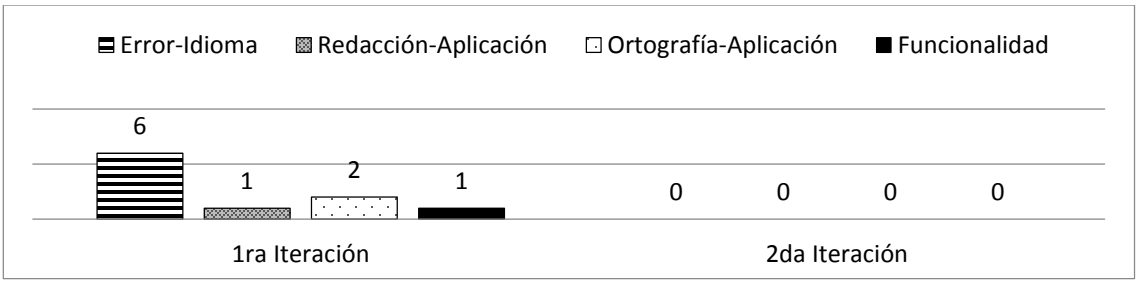


Figura 15: Resultado de las pruebas funcionales

3.5.2 Pruebas de carga y estrés

Estas pruebas se realizan para medir la respuesta de las aplicaciones a los distintos volúmenes de carga esperados (cantidad de usuarios y/o peticiones) y cómo responde el mismo frente a dicha carga. Para ello

se utilizó la herramienta *jMeter*, con el objetivo de realizar mediciones exactas, revisar el comportamiento funcional y medir el rendimiento de la misma. La realización de dichas pruebas fueron divididas en varios *tests* de 50, 100 y 150 hilos cada uno, los cuales simulan la cantidad de usuarios que acceden a las funcionalidades concurrentemente. Las mismas se realizaron sobre todas las funcionalidades de la aplicación, permitiendo comprobar su correcto desempeño dentro de la misma, a continuación se presentan los resultados para un grupo de funcionalidades específicas escogidas por el equipo de desarrollo, teniendo en cuenta su complejidad e importancia dentro del sistema. Es de válida aclaración que las pruebas fueron realizadas en un servidor de bajas prestaciones con características: sistema operativo Linux-Ubuntu 12.04, microprocesador *Celeron* a 2.50 GHz y memoria RAM de 1GB, considerablemente inferior a los descritos en los requisitos.

Tabla 8. Resultado de las pruebas con Jmeter

Funcionalidad	Hilos	# Muestras	Min	Max	% Error	Rendimiento
Añadir usuario	50	200	554	5999	0.00	18.6/seg
	100	400	724	10689	0.00	21.4/seg
	150	600	706	20170	0.00	20.5/seg
Matricular asignatura	50	100	233	6462	0.00	10.3/seg
	100	200	1358	9206	0.00	13.7/seg
	150	300	153	39294	0.00	7.2/seg
Resolver ejercicio	50	1400	0	4336	0.00	212.8/seg
	100	2800	0	8095	0.00	208.3/seg
	150	4200	1	12614	0.00	209.8/seg
Ver ejercicio recomendado	50	50	114	1528	0.00	21.8/seg
	100	100	348	3742	0.00	21.7/seg
	150	150	290	6079	0.00	21.2/seg

Hilos: Total de usuarios concurrentes.

 # Muestras: Total de peticiones realizadas al servidor.

Min: Tiempo (mili-segundos) mínimo de respuesta de la aplicación.

Max: Tiempo (mili-segundos) máximo de respuesta de la aplicación.

% Error: Porciento de error en la respuesta de la aplicación al total de peticiones.

Rendimiento: Tiempo (segundos) de respuesta de la aplicación por cada petición.

3.5.3 Pruebas de seguridad

Las pruebas de seguridad comprueban que los mecanismos de protección integrados en el sistema realmente lo protejan de irrupciones inapropiadas. En ellas se debe intentar conseguir las claves de acceso por cualquier medio, deben producir a propósito errores del sistema para ser corregidos luego por los desarrolladores del sistema (Quality, 2011).

Para garantizar la seguridad del sistema se ejecutaron pruebas de seguridad mediante una lista de chequeo ajustable a aplicaciones *web*. El propósito de esta lista es evaluar a través de indicadores la seguridad de las aplicaciones en un primer nivel, establecido por especialistas del grupo de seguridad del Departamento de Pruebas de Software (DEPSW) de la UCI, los 4 tipos de prueba examinan 15 indicadores. Los resultados de las mismas se presentan a continuación.

Forma de Uso de la Lista de Chequeo:

Peso: Define si el indicador a evaluar es crítico o no.

Evaluación (Eval): Es la forma de valorar el indicador en cuestión. El mismo se evalúa de 1 en caso de mal y 0 en caso que elemento revisado no presente errores.

Cantidad de elementos afectados: Especifica la cantidad de errores encontrados sobre el mismo indicador.

Comentario: Especifica los señalamientos o sugerencias que quiera incluir la persona que aplica la lista de chequeo.

Indicadores a Evaluar: Descripción del parámetro a evaluar.

N.P. (No Procede): Se usa para especificar que el indicador a evaluar no se puede aplicar en ese caso.

La aplicación de la lista de chequeo arrojó un total de 3 no conformidades: una dentro de la clasificación “Comprobación del Sistema de Autenticación” y 2 dentro de “Validación de Datos”; eliminándose ambas.

Para evaluar la seguridad del sistema se utilizó además la herramienta *Acunetix Web Vulnerability Scanner 8.0,* la cual es capaz de escanear sitios web en busca de posibles fallos de seguridad que puedan poner en peligro la integridad de la página publicada en Internet. Esta aplicación ejecuta una serie de pruebas, totalmente configurables por el usuario, para identificar las vulnerabilidades tanto en la programación de la página como en la configuración del servidor (Marco de desarrollo de la Junta de Andalucía. *Acunetix Web Vulnerability Scanner* (2014). Esta herramienta agrupa las vulnerabilidades encontradas en cuatro categorías: Alta, Media, Baja e Informacional. Se realizaron dos iteraciones de pruebas, la primera arrojó un total de tres vulnerabilidades de tipo Media, y cinco de tipo Informacional, estas últimas son referentes al

auto-completamiento de los campos “Correo” y “Contraseña” en los formularios de *login* con que cuenta la aplicación, las cuales se determinaron que no procedían pues se implementó dentro del sistema la opción “Recuérdame”, para brindar mayor facilidad a los usuarios del mismo. En una segunda iteración solo se detectaron las cinco vulnerabilidades de tipo Informacional (Ver Figura 17).

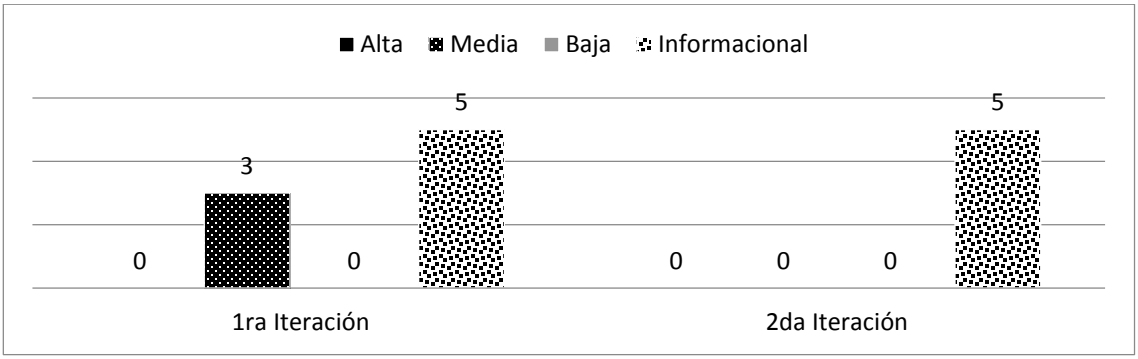


Figura 16: Resultados de las pruebas de seguridad con Acunetix

3.6 Conclusiones parciales

La realización de pruebas funcionales, de seguridad y de carga y estrés garantizó la detección de errores e inconvenientes en el sistema, y permitió su posterior solución. Del mismo modo brindó la posibilidad de monitorear el funcionamiento de la aplicación en diferentes entornos de ejecución y carga, con el objetivo de determinar posibles mejoras al rendimiento y capacidad de la misma.

Conclusiones generales

El presente trabajo dio cumplimiento cumplió a los objetivos trazados, enfatizando de manera general, las conclusiones siguientes:

1. El estudio de las técnicas de inteligencia artificial, así como de soluciones homólogas en el ámbito nacional y extranjero, las tendencias actuales de las mismas y las exigencias del cliente, permitieron elegir las herramientas, tecnologías, los lenguajes de programación y la metodología para guiar el proceso de desarrollo.
2. Los artefactos y actividades generados, basados en la metodología seleccionada, así como las entrevistas realizadas al cliente permitieron un mejor dominio del problema planteado, con vistas a la implementación del sistema.
3. Como resultado de la implementación y dándole cumplimiento al objetivo general de la investigación se obtuvo un Sistema evaluador de habilidades y de recomendación para el apoyo al proceso de enseñanza-aprendizaje, que permite el tratamiento diferenciado, el seguimiento de la evolución y el desarrollo de habilidades en los estudiantes que cursan la disciplina de IGSW en la carrera de Ingeniería en Ciencias Informáticas, desarrollando la capacidad de resolver problemas de manera autónoma y creativa de acuerdo a su disponibilidad de tiempo y sus necesidades de aprendizaje.
4. La aplicación de pruebas funcionales, de rendimiento y de seguridad permitió validar el correcto funcionamiento de la aplicación desarrollada, simulando ambientes reales en diferentes entornos de ejecución.

Recomendaciones

Al término de esta investigación se recomienda:

1. Profundizar más en las técnicas de inteligencia artificial para lograr la implementación de una funcionalidad “Reconocimiento de voz”, con el fin de realizar diagnósticos basados en la habilidad de expresión oral.
2. Agregar al sistema una funcionalidad de reporte con el fin de mostrar gráficamente a los profesores el avance de cada uno de los estudiantes y a la vez que estos también puedan observar la evolución propia a lo largo de su trayectoria en el sistema.
3. Evaluar la solución según los indicadores propuestos por la MSc. Yuniet del Carmen Toll Palma en la “Guía de evaluación de la calidad de los Objetos de Aprendizaje producidos en la Universidad de las Ciencias Informáticas”.

Bibliografía referenciada

Aamodt, A. 1994. *Explanation-Driven case based reasoning.* 1994.

Aguilera and Ortíz. 2001. La caracterización de perfiles de estilos de aprendizaje en la educación superior, una visión integradora. *Revista Estilos de Aprendizaje*, nº4, Vol 4. 2001.

Arias. 2009. *Modelo de planificación instruccional en sistemas tutoriales inteligentes.* 2009.

Bowen, Rich. 2007. *What´s New in Apache Web server.* 2007.

Bruner, J. 1991. *Actos de significado. Más allá de la revolución cognitiva.* 1991.

Bruner, Jerome S. 1966. *Toward a Theory of Instruction.* 1966.

Carbonell, J.R. *An artificial intelligence approach to computer assisted instruction.*

—. 2000. *An artificial intelligence approach to computer assisted instruction.* 2000. IEEE.

Cataldi, Zulma. 2009. *Sistemas Tutores Inteligentes Orientados a la Enseñanza para la comprensión.* Buenos Aires : s.n., 2009.

—. 2009. *Sistemas Tutores Inteligentes Orientados a la Enseñanza para la Comprensión.* Buenos Aires : s.n., 2009.

Ciudad Ricardo, F.A. 2009. *Propuesta de perfeccionamiento de la enseñanza de la Ingeniería de Software en la Universidad de las Ciencias Informáticas:Departamento Docente de Práctica Profesional e Ingeniería y Gestión de Sofware.* 2009.

Díaz Sardiñas, A. 2010. *Modelo del Profesional yObjetivos de la carrera de Ingeniería en Ciencias Informáticas: Vicerrectoría de Formación del Profesional.* UCI : s.n., 2010.

Díaz-Canel Bermúdez, Miguel. 2006. 2006.

Dunn, R, Dunn, K and Prince , G. 1979. *Learning Style Inventory (LSI) for Students in Grade.* Lawrence, Kansas : s.n., 1979.

Eguiluz, Javier. 2013. *Desarrollo web ágil con Symfony2. Madrid.* Madrid : s.n., 2013. s:n.

Eisentraut, Peter and Helmle, Bernd. 2008. *PostgreSQL Administration.* 2008.

Fernández González, A. M. 2006. *Estilos de comunicación. La comunicación y su importancia en la educación. Preparación pedagógico integral para profesores integrales.* La Habana : Félix Varela, 2006.

García García, Maribel, León Vidal, Lisset and Toledo Rivero , Viviana. 2009. *STIITS Sistema Tutor Inteligente para el Diagnóstico y Tratamiento de Infecciones de Transmisión Sexual.* 2009.

García, Guines Bravo. 2010. *PHP5 A través de ejemplos.* 2010.

Genveta. 2013. Bootstrap framework de Twitter. [Online] 2013. https://netbeans.org/index_es.html.

Gil. 2012. *Temario de curso C291-69 Tópicos Avanzados: Redes Neuronales Artificiales.* 2012.

Giraffa. 1997. *Selección y adopción de estrategias de educación en Sistemas Tutores Inteligentes.* Porto Alegre : s.n., 1997.

González Clavero, M. V. 2001. *Estilos de aprendizaje: su influencia para aprender a aprender.* 2001. 7.

González, J.C. Burguillo, J.C. Vidal, M. Llamas. 2013. *Sistemas Tutores Inteligentes: Propuesta de una arquitectura para aprendizaje en salud pública.* Poyapán : s.n., 2013.

Gregorc, A. 1985. *Gregorc Style Delineator.* 1985.

Groussard, Thierry. 2010. *Recursos Informáticos Java Enterprise Edition.* Barcelona : s.n., 2010.

2013. Hipertextual. [Online] 2013. [Cited: 11 25, 2014.] http://bitelia.com/2013/05/entendiendo-html5-guia-para-principiantes.

Horruitinier, J. 1993. *La Universidad Cubana: El Modelo de Formación.* 1993.

Hunt, D.E. 1979. *Learning Styles and Students Needs: An Introduction to Conceptual Level.* Reston, Virginia : s.n., 1979.

IEEE. 1993. *Standard Collection: Software Engineering.* 1993. IEE Standard 610..

Kabytes, Pablo Javie Suárez. 2010. *Excelentes plugins para crear gráficas con jquery.* 2010.

Kidd, J R. 1979. *Cómo aprenden los adultos.* Buenos Aires : s.n., 1979.

Knowles , M.S. 1995. *How to Develop Better Leaders.* New York : s.n., 1995.

Lago, Diego. 2008. *Guía de estilode programación.* 2008.

Larman, Craig. 2003. *UML y patrones Tomo II.* 2003. Tomo II.

Lingua.ly. 2014. lingua.ly. *lingua.ly.* [Online] lingua.ly, 2014. https://lingua.ly/learn-a-language/french-vocabulary/.

Martínez Márquez, Yoan. 2007. *Propuesta de sistema de evaluación del aprendizaje autónomo del idioma inglés en un entorno virtual de aprendizaje en la Universidad de las Ciencias Informáticas.* La Habana : s.n., 2007.

Martínez Sánchez, Natalia. 2011. *Guía de Orientación para la ingeniería del conocimiento en el diseño de sistemas de enseñanza-aprendizaje inteligentes utlizando el razonamiento basado en casos.* 2011.

Ojeda Navajas, Antonio. 2012. *Guía completa de CSS3.* 2012.

Oracle. 2013. Netbeans. [Online] Oracle, 2013. [Cited: 11 19, 2014.] https://netbeans.org/index_es.html..

Parra, Franklin. 2010. *Sistema Tutorial Inteligente.* Guayaquil : s.n., 2010.

Penadés, Ph.D Mª Carmen and Letelier, Patricio. 2006. *Agile methodology for the development of the software: eXtreme Programming (XP).* Valencia : s.n., 2006.

Pepa, Sofía Marina. 2014. *Suite de algoritmos de recomendación.* Madrid : s.n., 2014.

Pinto María. 2009. *Sistemas Tutores Inteligentes.* 2009.

Potencier, Fabien and Weaver, Ryan. 2014. *Symfony 2.3.* 2014.

Pressman, Roger. 2005. *Ingeniería del Software. Un enfoque práctico.* 2005.

Renato, Velasquez Luis. 2013. *MODELO DE UN SISTEMA TUTOR INTELIGENTE PARA LA ENSEÑANZA MUSICAL.* La Paz – Bolivia : s.n., 2013.

Riechmann, S.W. 1979. *Learning Styles: Their Role in Teaching Evaluation and Course Design.* Ann Arbor, Michigan : s.n., 1979.

Rodríguez , Ana P., Polanco, Josué and Hernández, Darwin. 2013. *Open UP.* 2013.

Rodríguez, Lydia Rosa Ríos. 2008. *Un ambiente de aprendizaje asistido por computadora para la Programación Lógica.* s.l. : Revista de Informática Educativa y Medios Audiovisuales, 2008. Vol. 5. 1667-8338.

Rodríguez, Raúl, et al. 2006. *Aprendizaje con las nuevas tecnologías paradigma emergente.* 2006.

Scribd. 2011. Reglas de producción. [Online] 2011. [Cited: 12 02, 2014.] http://es.scribd.com/doc/90856631/REGLAS-DE-PRODUCCION.

Sebastián, J. 2010. *Modelo Vista Controlador- Definición y Características.* 2010.

2010. Sistemas expertos conexionistas. [Online] 2010. http://menteerrabunda.blogspot.com/2010/01/sistemas-expertos-conexionistas.html..

Sommerville, Ian. 2005. *Ingeniería de software.* 2005.

Universidad Nacional de Colombia. 2014. *Tutores inteligentes mejoran aprendizaje en matemáticas.* Bogotá : s.n., 2014.

Villanueva, María Luisa. 1997. *Los estilos de aprendizaje de Lenguas.* 1997.

Zambrano, Renán Alexi. 2013. *Metodología de la investigación.* 2013.

Bibliografía consultada

Aamodt, A. 1994. *Explanation-Driven case based reasoning.* 1994.

Aguilera and Ortíz. 2001. La caracterización de perfiles de estilos de aprendizaje en la educación superior, una visión integradora. *Revista Estilos de Aprendizaje*, nº4, Vol 4. 2001.

Arias. 2009. *Modelo de planificación instruccional en sistemas tutoriales inteligentes.* 2009.

Bowen, Rich. 2007. *What´s New in Apache Web server.* 2007.

Bruner, J. 1991. *Actos de significado. Más allá de la revolución cognitiva.* 1991.

Bruner, Jerome S. 1966. *Toward a Theory of Instruction.* 1966.

Carbonell, J.R. *An artificial intelligence approach to computer assisted instruction.*

—. 2000. *An artificial intelligence approach to computer assisted instruction.* 2000. IEEE.

Cataldi, Zulma. 2009. *Sistemas Tutores Inteligentes Orientados a la Enseñanza para la comprensión.* Buenos Aires : s.n., 2009.

—. 2009. *Sistemas Tutores Inteligentes Orientados a la Enseñanza para la Comprensión.* Buenos Aires : s.n., 2009.

Ciudad Ricardo, F.A. 2009. *Propuesta de perfeccionamiento de la enseñanza de la Ingeniería de Software en la Universidad de las Ciencias Informáticas:Departamento Docente de Práctica Profesional e Ingeniería y Gestión de Sofware.* 2009.

Díaz Sardiñas, A. 2010. *Modelo del Profesional yObjetivos de la carrera de Ingeniería en Ciencias Informáticas: Vicerrectoría de Formación del Profesional.* UCI : s.n., 2010.

Díaz-Canel Bermúdez, Miguel. 2006. 2006.

Dunn, R, Dunn, K y Prince , G. 1979. *Learning Style Inventory (LSI) for Students in Grade.* Lawrence, Kansas : s.n., 1979.

Eguiluz, Javier. 2013. *Desarrollo web ágil con Symfony2. Madrid.* Madrid : s.n., 2013. s:n.

Eisentraut, Peter y Helmle, Bernd. 2008. *PostgreSQL Administration.* 2008.

Fernández González, A. M. 2006. *Estilos de comunicación. La comunicación y su importancia en la educación. Preparación pedagógico integral para profesores integrales.* La Habana : Félix Varela, 2006.

García García, Maribel, León Vidal, Lisset y Toledo Rivero , Viviana. 2009. *STIITS Sistema Tutor Inteligente para el Diagnóstico y Tratamiento de Infecciones de Transmisión Sexual.* 2009.

García, Guines Bravo. 2010. *PHP5 A través de ejemplos.* 2010.

Genveta. 2013. Bootstrap framework de Twitter. [En línea] 2013. https://netbeans.org/index_es.html.

Gil. 2012. *Temario de curso C291-69 Tópicos Avanzados: Redes Neuronales Artificiales.* 2012.

Giraffa. 1997. *Selección y adopción de estrategias de educación en Sistemas Tutores Inteligentes.* Porto Alegre : s.n., 1997.

González Clavero, M. V. 2001. *Estilos de aprendizaje: su influencia para aprender a aprender.* 2001. 7.

González, J.C. Burguillo, J.C. Vidal, M. Llamas. 2013. *Sistemas Tutores Inteligentes: Propuesta de una arquitectura para aprendizaje en salud pública.* Poyapán : s.n., 2013.

Gregorc, A. 1985. *Gregorc Style Delineator.* 1985.

Groussard, Thierry. 2010. *Recursos Informáticos Java Enterprise Edition.* Barcelona : s.n., 2010.

2013. Hipertextual. [En línea] 2013. [Citado el: 25 de 11 de 2014.] http://bitelia.com/2013/05/entendiendo-html5-guia-para-principiantes.

Horruitinier, J. 1993. *La Universidad Cubana: El Modelo de Formación.* 1993.

Hunt, D.E. 1979. *Learning Styles and Students Needs: An Introduction to Conceptual Level.* Reston, Virginia : s.n., 1979.

IEEE. 1993. *Standard Collection: Software Engineering.* 1993. IEE Standard 610..

Kabytes, Pablo Javie Suárez. 2010. *Excelentes plugins para crear gráficas con jquery.* 2010.

Kidd, J R. 1979. *Cómo aprenden los adultos.* Buenos Aires : s.n., 1979.

Knowles , M.S. 1995. *How to Develop Better Leaders.* New York : s.n., 1995.

Lago, Diego. 2008. *Guía de estilode programación.* 2008.

Larman, Craig. 2003. *UML y patrones Tomo II.* 2003. Tomo II.

Lingua.ly. 2014. lingua.ly. *lingua.ly.* [En línea] lingua.ly, 2014. https://lingua.ly/learn-a-language/french-vocabulary/.

Martínez Márquez, Yoan. 2007. *Propuesta de sistema de evaluación del aprendizaje autónomo del idioma inglés en un entorno virtual de aprendizaje en la Universidad de las Ciencias Informáticas.* La Habana : s.n., 2007.

Martínez Sánchez, Natalia. 2011. *Guía de Orientación para la ingeniería del conocimiento en el diseño de sistemas de enseñanza-aprendizaje inteligentes utlizando el razonamiento basado en casos.* 2011.

Ojeda Navajas, Antonio. 2012. *Guía completa de CSS3.* 2012.

Oracle. 2013. Netbeans. [En línea] Oracle, 2013. [Citado el: 19 de 11 de 2014.] https://netbeans.org/index_es.html..

Parra, Franklin. 2010. *Sistema Tutorial Inteligente.* Guayaquil : s.n., 2010.

Penadés, Ph.D Mª Carmen y Letelier, Patricio. 2006. *Agile methodology for the development of the software: eXtreme Programming (XP).* Valencia : s.n., 2006.

Pepa, Sofía Marina. 2014. *Suite de algoritmos de recomendación.* Madrid : s.n., 2014.

Pinto María. 2009. *Sistemas Tutores Inteligentes.* 2009.

Potencier, Fabien y Weaver, Ryan. 2014. *Symfony 2.3.* 2014.

Pressman, Roger. 2005. *Ingeniería del Software. Un enfoque práctico.* 2005.

Renato, Velasquez Luis. 2013. *MODELO DE UN SISTEMA TUTOR INTELIGENTE PARA LA ENSEÑANZA MUSICAL.* La Paz – Bolivia : s.n., 2013.

Riechmann, S.W. 1979. *Learning Styles: Their Role in Teaching Evaluation and Course Design.* Ann Arbor, Michigan : s.n., 1979.

Rodríguez , Ana P., Polanco, Josué y Hernández, Darwin. 2013. *Open UP.* 2013.

Rodríguez, Lydia Rosa Ríos. 2008. *Un ambiente de aprendizaje asistido por computadora para la Programación Lógica.* s.l. : Revista de Informática Educativa y Medios Audiovisuales, 2008. Vol. 5. 1667-8338.

Rodríguez, Raúl, y otros. 2006. *Aprendizaje con las nuevas tecnologías paradigma emergente.* 2006.

Scribd. 2011. Reglas de producción. [En línea] 2011. [Citado el: 02 de 12 de 2014.] http://es.scribd.com/doc/90856631/REGLAS-DE-PRODUCCION.

Sebastián, J. 2010. *Modelo Vista Controlador- Definición y Características.* 2010.

2010. Sistemas expertos conexionistas. [En línea] 2010. http://menteerrabunda.blogspot.com/2010/01/sistemas-expertos-conexionistas.html..

Sommerville, Ian. 2005. *Ingeniería de software.* 2005.

Universidad Nacional de Colombia. 2014. *Tutores inteligentes mejoran aprendizaje en matemáticas.* Bogotá : s.n., 2014.

Villanueva, María Luisa. 1997. *Los estilos de aprendizaje de Lenguas.* 1997.

Zambrano, Renán Alexi. 2013. *Metodología de la investigación.* 2013.