UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

“Coliseo Virtual”

Aplicación informática de apoyo al aprendizaje de las técnicas de programación.

03/11/2010

Autor: Lic. Rolan Rober Bullain Dieguez.

Tutor: DrC. José Ortiz Rojas.
MsC. Febe Angel Ciudad Ricardo.

DECLARACIÓN DE AUTORÍA

Yo Rolan Rober Bullain Dieguez, con carné de identidad 79071716469, declaro que soy el autor principal del resultado que expongo en la presente memoria titulada *"Coliseo Virtual", aplicación informática de apoyo al aprendizaje de las técnicas de programación.* Para optar por el título de Máster en Informática Aplicada.

Este trabajo fue desarrollado en el período comprendido entre el mes de enero de 2006 y junio de 2007, en colaboración con la estudiante Janet Carreño Cáceres. Se continuó la investigación durante el curso 2008 – 2009 con los estudiantes: Abdiel Matos Nieto e Ismarai Núñez Viltres; los cuales reconocen la autoría principal del resultado expuesto en esta memoria.

Finalmente declaro que todo lo anteriormente expuesto se ajusta a la verdad, y asumo la responsabilidad moral y jurídica que se derive de este juramento profesional.

Y para que así conste, firmo la presente declaración jurada de autoría en Ciudad de la Habana a los___ días del mes de _____ del año 2010.

Autor Principal	Lic. Rolan Rober Bullain Dieguez	_____
Tutor:	DrC. José Ortiz Rojas	_____
Tutor:	MsC. Febe Angel Ciudad Ricardo	_____
Colaboradores:		
	Ing. Abdiel Matos Nieto	_____

DEDICATORIA

A Fidel Castro Ruz:

...el arquitecto indiscutible de este gran sueño hecho realidad: La Universidad de las Ciencias Informáticas.

A nuestro infinito apóstol, José Martí:

...que con magnánima seguridad dijo: "...escuelas no han de llamarse, sino talleres..."

A nuestros estudiantes:

...que tan titánica tarea les ha sido asignada.

AGRADECIMIENTOS

Agradezco infinitamente y sin palabras para poder expresar mis sentimientos, a mis padres y familiares por los valores que me han inculcado. Su constante tesón, espíritu de sacrificio, rectitud, sencillez y amor, siempre han sido fuente de inspiración para tratar de ser cada día una persona mejor y merecer la enorme confianza que han depositado en mí.

A mi cariñosa hermana por su dulce sonrisa y su enorme inocencia, a mi hermano por su nobleza, timidez y el amor que me profesa a pesar de la distancia. A mi abuelitas Agueda y Nancy, que no saben lo que es el descanso y de ellas aprendí a no temerle al trabajo. A mis tías Yamilet, Idal; mis tíos Paco y Nene que de tan cerca que siempre hemos estado nos queremos como hermanos. A mi esposa, que a pesar de ser la última persona que llegó a mi vida ha sido el sostén que necesitaba y lo que le faltaba a mi corazón para estar completo.

A mi abuelito, que en paz descanse, donde quiera que esté llegue mi infinito amor por ser la raíz de una hermosa familia, que con sus imperfecciones siempre trató de dar lo mejor de sí.

A mis tutores, por aventurarse conmigo en esta afanosa meta. A mis amigos, por su apoyo y espíritu crítico. A mi decana Yvonne, por darme la oportunidad de terminar una tarea pendiente.

¡A todos, muchas gracias! Por ser como son, por estar ahí y por formar parte, de una forma u otra, de mi vida.

RESUMEN

Desde el surgimiento de los primeros lenguajes de programación basados en el lenguaje de máquina, hasta los de más alto nivel, que soportan paradigmas avanzados y complejos como los orientados a: objetos, eventos, aspectos, entre otros; el aprendizaje de las técnicas de programación se ha caracterizado por ser un proceso complejo.

Los Entornos de Desarrollo Integrado (IDE, por sus siglas en inglés), normalmente usados en el desarrollo de software profesional, también son usados como medio de enseñanza en el proceso de asimilación de nuevos conocimientos y el desarrollo de habilidades. Estas aplicaciones informáticas presentan funcionalidades que permiten agilizar la codificación y depuración de programas; pero carecen de características didáctico-pedagógicas. Por lo que profesores y estudiantes se auxilian de aplicaciones informáticas educativas diseñadas con fines docentes.

El trabajo que se presenta a continuación hace un análisis de las habilidades fundamentales que debe desarrollar un Ingeniero en Ciencias Informáticas al recibir la disciplina de *Técnicas de Programación de Computadoras*, haciendo énfasis en un subconjunto de ellas y en las aplicaciones informáticas utilizadas como medio de enseñanza en la formación y desarrollo de dichas habilidades. Se realiza un análisis de las características fundamentales de estas aplicaciones y se clasifican teniendo en cuenta las mismas, presentando sus ventajas y limitaciones. Finalmente se presenta una aplicación que pueda ser utilizada como medio de enseñanza.

ÍNDICE DE IMÁGENES.

ÍNDICE DE TABLAS.

TABLA DE CONTENIDOS.

INTRODUCCIÓN

La importancia de la Informática es tal que muchos países le han apostado a ella como uno de sus principales eslabones en la economía, algunos han tenido resultados muy positivos como es el caso de la India, Israel e Irlanda; los que han tenido una evolución más exitosa en los últimos 10 años (Arora, y otros, 2005).

En nuestro país, desde los primeros años de la Revolución, el gobierno se dio a la tarea de potenciar el desarrollo de esta ciencia, llegándose a crear en los años 70 computadores cubanos: CID-201A, CID-201B y CID-300/10 (López, 1979). En este mismo año se crea en el Ministerio de la Educación el Grupo para el Desarrollo de la Informática de apoyo a la educación primaria, secundaria y media (Encinosa, 2002). Paulatinamente la computación fue masificándose, algunos de los hechos transcendentales que marcaron su curso fueron:

- Creación, en los años 70, de los lenguajes FOCAL y LINCO para apoyar el estudio de la Informática y la programación (Encinosa, 2002).
- Incorporación en el Instituto Preuniversitario de Ciencias Exactas Vladimir Ilich Lenin, años 70, de una CID-201B (Encinosa, 2002).
- Diciembre de 1984 se inauguró en el Palacio Central de Pioneros Ernesto Guevara un círculo de interés de computación electrónica, donde los niños disponían de una CID 300/10 (Encinosa, 2002).
- 8 de diciembre de 1984, en el acto de clausura del VI Congreso de la Federación de Estudiantes de la Enseñanza Media, el Comandante en Jefe Fidel Castro Ruz expresaba: “Se ha hablado ya de introducir la computación en los niveles universitarios, pero ya estamos pensando introducirla también en el nivel medio…” (Alonso, 2007).
- Incorporación de microcomputadoras IBM compatibles en los preuniversitarios vocacionales de ciencias exactas (Encinosa, 2002).
- 8 de septiembre de 1987, por iniciativa de nuestro Comandante en Jefe Fidel Castro Ruz, surgen los Joven Club de Computación y Electrónica, con el objetivo de contribuir a la informatización de la sociedad cubana (Jiménez Fernández, y otros, 2008).

Con la creación de los Joven Club de Computación y Electrónica comenzaría una verdadera masificación de la computación y la informática, en la actualidad nuestro país cuenta con más de 600 Joven Club de Computación y Electrónica, donde se imparten más de 50 tipos de cursos con más de un millón de egresados (Fiallo Gómez, y otros, 2006).

El gobierno revolucionario ha ido tomando acciones para elevar la preparación científico-técnica en los diferentes niveles de enseñanza y en la informatización de la sociedad; pero desde el punto de vista comercial, a nivel nacional, hasta principios del año 2002 existía un desempeño muy tímido, las exportaciones de aplicaciones informáticas no rebasaban los 10 millones de dólares (Triana Cordoví, 2002). En este mismo año el desarrollo de la Informática comienza a mirarse como una línea estratégica para el desarrollo de la economía del país y se emprenden un conjunto de acciones:

- Inicio de la de construcción y funcionamiento de la Universidad de las Ciencias Informáticas (UCI, en lo adelante).

- Estudio de la carrera de Ingeniería Informática, paulatinamente, en todas las provincias del país.
- Reorganización de los Institutos Politécnicos de Informática patrocinados por la UCI.
- Extensión de tres Facultades Regionales de la UCI.
- Creación de Centros de Desarrollo (Ciudad Habana, Villa Clara y Holguín) subordinados a la UCI.

El afianzamiento de las carreras universitarias especializadas y la formación de técnicos medios en las distintas áreas de la computación han traído consigo la conformación y consolidación de los distintos planes de estudios ajustados a las necesidades de las mismas. Algunas de las materias que se han insertado en estos planes de estudios, como las relacionadas con Ciencias Básicas y las Ciencias Humanísticas, han sido muy estudiadas desde el punto de vista pedagógico, en contraposición a las materias intrínsecas de estas especialidades a las que les urge una profundización en estos tipos de estudios, como pueden ser: la Programación Orientada a Objetos, Máquinas Computadoras, Sistemas Operativos, entre otras asignaturas.

Las materias de la especialidad de esta carrera presentan una complejidad intrínseca, que obliga el estudiante a cambiar su concepción del mundo, requiere de una gran abstracción y pensamiento lógico – algorítmico, algo que la mayoría de los seres humanos realizamos cotidiana e intuitivamente; pero en el instante de plasmarlo en una computadora se convierte en un proceso extremadamente complicado porque implica extrapolar esas habilidades, es decir, aplicar estas habilidades a nuevas situaciones en actividades productivas o académicas. Si a lo antes planteado le sumamos que la mayoría de los profesores son graduados de las mismas carreras técnicas y carecen de estudios pedagógicos, que los estudiantes disponen de un número limitado de herramientas que los ayuden en su proceso de aprendizaje, se puede inferir que esta situación incide de forma negativa en el proceso de enseñanza - aprendizaje, trayendo como consecuencia: deficientes conocimientos teóricos en las materias profesionales, insuficiente desarrollo de habilidades y baja calidad. Por ejemplo, en el curso 2008 – 2009, luego de realizar cuatro convocatorias del examen final de la disciplina *Técnicas de Programación de Computadoras*, sólo el 2.5% de los estudiantes terminó con una evaluación de avanzado, el 21.3 % con una evaluación de intermedio, el 71.6% con una evaluación de básico y 2.1% suspensos (Suarez-Inclank Rivero, 2009). Independientemente que el porcentaje de suspensos es bajo es necesario considerar que más de la tercera parte de los estudiantes egresaron con un cumplimiento mínimo de los objetivos de la disciplina. Lo anterior es más notable cuando los estudiantes tienen que demostrar sus habilidades en el ejercicio de la profesión.

La Universidad de las Ciencias Informáticas, centro donde se desarrolla esta investigación, cuenta con una matrícula de aproximadamente diez mil estudiantes por curso, divididos en trece facultades, diez en la sede central y tres facultades regionales; se estudia una sola especialidad, Ingeniería en Ciencias Informáticas. Este centro docente tiene como característica peculiar la vinculación de los estudiantes a la producción mediante el desarrollo de proyectos de desarrollo de software reales, donde la vinculación a la producción es considerada como una asignatura más del plan de estudio, solo que forma parte de la asignatura integradora Práctica Profesional, donde el estudiante tiene que demostrar todas las habilidades profesionales adquiridas.

En la carrera de Ingeniería en Ciencias Informáticas, la disciplina de *Técnicas de Programación de Computadoras* (TPC, en lo adelante), centro de atención de esta investigación, forma parte

del grupo de disciplinas principales. Además es la base para la asimilación del conocimiento de otras disciplinas que se imparten en años superiores.

El departamento docente central que dentro de la universidad dirige esta disciplina, en el año 2008 presentó en la comisión de carrera de dicho centro docente, el “Plan de Perfeccionamiento de la Disciplina de Programación” (Suarez-Inclan Rivero, 2008) el cual detalla las características de la disciplina en cada uno de los cursos precedentes y del cual se extrajo la información sobre el esquema pedagógico, los lenguajes y herramientas utilizadas.

Desde el punto de vista pedagógico las clases son impartidas siguiendo un esquema ascendente, en el curso 2002 – 2003, donde el punto de partida son los conceptos más sencillos como pudieran ser las variables y los tipos de datos, hasta los conceptos más generales y complejos como el de objeto; desde el punto de vista pedagógico es un método que va de lo específico a lo general. A partir del curso 2003 – 2004 hasta el curso 2006 – 2007 las clases son impartidas siguiendo el esquema descendente, donde el punto de partida son los conceptos más generales como el de objeto, hasta los conceptos más específicos como los de variable y tipo de dato.

Los lenguajes de programación utilizados para concretar los conceptos de programación han sido muy variados, impartiéndose inicialmente solo el lenguaje C++ en toda la universidad, seguido de la utilización experimental del lenguaje C# en el curso 2004 – 2005 en dos grupos de la facultad 6, seguido de una mayor diversidad en curso 2005 – 2006 donde los profesores en la búsqueda de facilitar el proceso de enseñanza – aprendizaje utilizaron los lenguajes de C++, C#, Java y Python. A partir del siguiente curso la selección del lenguaje de programación que sería utilizado en las clases estaría fuertemente ligado a los intereses productivos de cada una de las facultades, en estos cursos se consolidan los lenguajes de C++, C# y Java.

La utilización de aplicaciones informáticas para apoyar el proceso de aprendizaje ha sido muy pobre, hasta el curso 2005 – 2006 solo se hacía uso de los Entornos de Desarrollo Integrados (IDE, por sus siglas en inglés y en lo adelante) profesionales de cada uno de estos lenguajes de programación, en este curso comienza utilizarse la plataforma de teleformación soportada bajo Moodle. Otras como el PSEINT y el Scratch fueron utilizadas durante los cursos 2007 – 2008 y 2008 – 2009, respectivamente; pero tienen la limitación que por sus características solo pueden ser usadas durante un período de tiempo y no permiten trabajar el desarrollo de las habilidades de forma sistémica.

Teniendo en cuenta los elementos antes mencionados se definió el siguiente **problema científico**: la limitación de las aplicaciones informáticas como medio de enseñanza – aprendizaje en la disciplina *Técnicas Programación de Computadoras* para el desarrollo de las habilidades con un enfoque sistémico en el proceso de formación del Ingeniero en Ciencias Informáticas.

Para darle solución al problema científico antes mencionado se plantea como **objetivo general**: desarrollar una aplicación informática que asista el desarrollo de habilidades con un enfoque sistémico desde sus funcionalidades.

El problema descrito tiene como **objeto de estudio**: el proceso de enseñanza aprendizaje de las *Técnicas de Programación de Computadoras* en la UCI y como **campo de acción**: las aplicaciones informáticas como medios de enseñanza – aprendizaje en las *Técnicas de Programación de Computadoras*.

Se plantea como **hipótesis** de esta investigación que: la utilización de una aplicación informática que permita tratar problemas de las distintas asignaturas de la disciplina, que sea motivador, independiente de los IDE y lenguajes de programación como medio de enseñanza – aprendizaje en las *Técnicas de Programación de Computadoras* permitirá el desarrollo de habilidades con un enfoque sistémico.

Al tiempo que se plantearon las siguientes **tareas** de investigación para darle cumplimiento al objetivo general:

- *Identificar* el estado actual de las aplicaciones informáticas de apoyo al proceso de enseñanza – aprendizaje de las TPC.
- *Analizar* y *evaluar* las aplicaciones informáticas de apoyo al proceso de enseñanza – aprendizaje de las TPC.
- *Desarrollar* la solución propuesta.
- *Validar* por medio de criterio de experto la solución propuesta.

En el desarrollo de la investigación científica se han utilizado un conjunto de métodos científicos para la obtención, procesamiento y arribo a conclusiones. Dentro de estos podemos mencionar los siguientes:

Métodos Teóricos:

1. *Hipotético – deductivo:* para la inferencia de las conclusiones parciales y generales así como la verificación de la hipótesis.
2. *Analítico – sintético:* al descomponer el problema de investigación en elementos por separado y profundizar en el estudio de cada uno de ellos, para luego sintetizarlos en la solución propuesta.
3. *Sistémico:* para definir los componentes del sistema y sus relaciones.

Métodos Empíricos:

1. *Observación:* para identificar elementos relevantes en el diseño y desarrollo de la investigación.
2. *Entrevista:* utilizado para obtener información verbal y real del uso de aplicaciones informáticas de apoyo al proceso de enseñanza – aprendizaje y su incidencia en el mismo.
3. *Encuesta:* para la evaluación del impacto de la solución propuesta en la investigación.

Como **aspectos novedosos** de esta investigación se pueden mencionar los siguientes:

- La concepción y diseño de una aplicación informática que apoye la formación de habilidades en las TPC con un enfoque sistémico.
- El análisis de las características de las aplicaciones informáticas utilizadas como medio de enseñanza – aprendizaje en el proceso docente – educativo de las técnicas de programación.
- La clasificación de las aplicaciones informáticas usadas con fines docentes para apoyar el proceso de aprendizaje de las técnicas de programación en términos de sus características y las habilidades que mediante su utilización contribuye a formar.

Como **aporte práctico** del trabajo realizado se puede mencionar la construcción de una aplicación informática que permite el desarrollo de habilidades en las TPC con un enfoque sistémico.

El trabajo que se desarrolla en esta investigación está conformado dos capítulos, en los cuales se tratan los siguientes temas: *capítulo 1*, se realiza una caracterización de la disciplina *Técnicas de Programación de Computadoras* en la UCI así como de las habilidades fundamentales requeridas en el aprendizaje de las técnicas de programación, se realiza un análisis de aplicaciones informáticas existentes para apoyar el proceso de aprendizaje de las técnicas de programación y una clasificación de las mismas, se estudian las posibles metodologías, tecnologías y herramientas que serán utilizadas para el desarrollo de la propuesta de solución; *capítulo 2*, se realiza un análisis de los elementos principales que se tuvieron en cuenta en la elaboración de la solución propuesta y su validación mediante el criterio de expertos.

CAPÍTULO 1: APLICACIONES INFORMÁTICAS EDUCATIVAS Y HERRAMIENTAS DE DESARROLLO.

INTRODUCCIÓN.

En este capítulo se aborda el estudio realizado para la selección de los elementos necesarios que nos permitirán llevar a cabo la implementación de la solución propuesta. Comenzando por un estudio de la disciplina de TPC y el estado del arte de los medios de enseñanza generalmente usados en esta disciplina, seguido del análisis de las fundamentales metodologías de desarrollo, paradigmas de programación, lenguajes y herramientas.

1.1. PROCESO DE ENSEÑANZA DE LA DISCIPLINA DE TÉCNICAS DE PROGRAMACIÓN DE COMPUTADORAS EN LA UCI.

La disciplina de TPC ha tenido un largo proceso de transformación y perfeccionamiento durante cada uno de los cursos en la Universidad de las Ciencias Informáticas. La **Tabla 1.1** muestra como ha sido el comportamiento con respecto a la cantidad de horas lectivas de la disciplina en el plan de estudio y con respecto al resto de las disciplinas de la especialidad, sin tener en cuenta la Práctica Profesional.

Tabla 1.1: *Comportamiento por curso de la disciplina de Técnicas de Programación de Computadoras en cuanto a: cantidad de asignaturas, las horas que representa en el plan de estudio y su comparación con las disciplinas de la especialidad: Sistemas Digitales, Ingeniería y Gestión de Software e Inteligencia Artificial.*

Curso	Cant. de Asig. en la Disciplina	Total Hs de plan de Estudio	Total Hs de la Disciplina	Hs de las Disciplinas de la Especialidad	% de Hs vs Disciplinas de la Especialidad	% de Hs vs el Plan de Estudio
02-03	4	6730	288	1332	21.62	4.28
03-04	5	6528	360	1224	29.41	5.51
04-05	5	6512	360	1225	29.39	5.53
05-06	5	6512	360	1225	29.39	5.53
06-07	5	6424	360	1225	29.39	5.60
07-08	5	6424	360	1225	29.39	5.60
08-09	5	6376	360	1225	29.39	5.65
09-10	5	7050	436	1156	37.72	6.18

Fuente: *Elaboración propia a partir de la análisis de los planes de estudios A, C, D, E, F y G del Ingeniero en Ciencias Informáticas* (Comité de Carrera, 2002), (Comité de Carrera, 2003), (Comité de Carrera, 2004), (Comité de Carrera, 2006), (Comité de Carrera, 2008) *y* (Comité de Carrera, 2009).

Es durante el curso 2009 – 2010, Plan de Estudio G para el Ingeniero en Ciencias Informáticas (Comité de Carrera, 2009), que la disciplina cobra su mayor relevancia, llegando a representar el 6.18 % del total de horas del plan de estudio y más de la tercera parte, 37.72 %, de las horas planificadas para las disciplinas de la especialidad. Este incremento estuvo dado por el aumento en la planificación de horas para las asignaturas de Introducción a la Programación y Programación 1, de 72 horas durante los cursos anteriores a 90 horas en el último plan de estudio y al incremento en horas de las asignaturas de Programación 2 y Programación 3, de 72

horas a 96 horas cada una. Excepto la asignatura de Programación 4, el resto de las asignaturas de la disciplina tuvieron un aumento en el número de horas planificadas.

Del análisis del Plan de Estudio G (Comité de Carrera, 2009) se identificaron los siguientes objetivos relacionados con la disciplina de Técnicas de Programación de Computadoras:

- Aplicar correctamente los procesos lógicos del pensamiento abstracto y el razonamiento inductivo y deductivo para desarrollar procedimientos que se requieren para solucionar las tareas propuestas en las asignaturas.
- Emplear técnicas y conocimientos básicos de buenas prácticas de programación a través del trabajo colectivo, en la concreción de proyectos de curso y utilizando técnicas del proceso personal de software, documentación técnica generada bajo estándares de modelado, de codificación y de diseño de interfaz, realización de pruebas a código y confección de la ayuda de software.
- Aplicar prácticas de comunicación, trabajo independiente y en equipo para la solución de ejercicios y la realización de tareas y actividades que propicien el desarrollo de valores acordes con el modelo del profesional.
- Ejecutar de forma eficaz y eficiente las tareas y actividades a través de la integración de los conocimientos y las habilidades en la programación de aplicaciones y el diseño e implementación de Sistemas de Bases de Datos y haciendo uso eficiente de la microcomputadora y de los ambientes y herramientas.
- Diseñar sistemas informáticos y sistemas de apoyo a la decisión que pueden requerir la utilización de la matemática aplicada, las técnicas de inteligencia artificial y la programación multiparadigma según los aspectos legales, de seguridad, comunicación con el cliente y teniendo en cuenta los intereses del país.

Para que estos objetivos puedan ser alcanzados es necesario que los estudiantes desarrollen una serie de habilidades, de las cuales un subconjunto de las más generales se listan a continuación y que fueron extraídas del estudio de los Programas analíticos de las asignaturas de la disciplina y de la lectura de Pears (Pears, y otros, 2007), Ala-Mutka (Ala-Mutka, 2005), Kurland (Kurland, y otros, 1986), la ACM, la AIS y la IEEE (ACM, y otros, 2005) y (ACM, y otros, 2008) :

- Comprender y reconocer algoritmos y sus estructuras de datos en la solución de problemas.
- Conocer el hardware del ordenador desde una perspectiva del software, como pudieran ser: el uso del procesador, memoria, unidades de disco, pantalla, etc.
- Desarrollar software que permitan el uso de algoritmos y estructuras de datos fundamentales.
- Diseñar e implementar complejas unidades estructurales que utilicen algoritmos, estructuras de datos e interfaces a través del cual estas se comuniquen.
- Conocer y utilizar principios y tecnologías de la ingeniería de software para garantizar que las implementaciones de software sean robustos, fiables y apropiados para su objetivo público.
- Identificar y analizar, los criterios y especificaciones de las estrategias apropiadas para resolver problemas específicos, y un plan para su solución.

- Evaluar, testear, analizar el grado en que un sistema basado en computadora cumple con los criterios definidos para su uso actual y futuro.
- Utilizar las teorías, prácticas, y herramientas pertinentes para la especificación, diseño, implementación y mantenimiento, así como la evaluación de los sistemas informáticos.
- Evaluar los sistemas en términos de atributos de calidad general y posibles compensaciones presentadas en el problema dado.
- Utilizar conscientemente en el desarrollo de software el principio de la reutilización de código.

Todas las habilidades que debe desarrollar un estudiante al adentrarse en el estudio de la disciplina de TPC son resumidas en la **Tabla 1.2** haciendo uso del diagrama de Bejamín Bloom.

Tabla 1.2: *Niveles de la taxonomía de los objetivos instructivos y habilidades de Benjamín Bloom, propuestos por el grupo especial para Ciencia de la Computación de la ACM y la IEEE.*

Nivel	Objetivo	Habilidades
1	Recordar	Reconocer, recordar, describir
2	Entender	Interpretar, ejemplificar, clasificar, inferir, comparar, explicar y resumir
3	Aplicar	Ejecutar, implementar, computar, utilizar y resolver
4	Analizar	Diferenciar, organizar, discriminar y subdividir
5	Evaluar	Evaluar, criticar, estimar y comparar
6	Crear	Generar, planear, producir, innovar, diseñar, organizar

Fuente: Tomado de (ACM, y otros, 2008).

El desarrollo de habilidades en las técnicas de programación requiere de la utilización de varias aplicaciones informáticas, por lo general son utilizadas aplicaciones informáticas para el desarrollo de software profesional, con las cuales los estudiantes pueden aplicar los conocimientos adquiridos; pero que como su nombre lo indica requieren de cierta experticia y presentan limitaciones para su uso desde el punto de vista didáctico.

1.2. CLASIFICACIÓN DE LAS APLICACIONES INFORMÁTICAS EDUCATIVAS.

La programación de computadoras ha sido calificada desde sus inicios como una actividad compleja, por lo que son muchas las herramientas informáticas que han sido desarrolladas con el objetivo de asistir a los estudiantes en la asimilación de conocimientos y la formación de habilidades desde el proceso de aprendizaje, docente y extra docente.

Según Monteagudo (Monteagudo, 2003) el uso de la Informática puede facilitar el aprendizaje de conceptos, métodos, principios; puede ayudar a resolver problemas de variada naturaleza y puede contribuir a desarrollar diferentes tipos de habilidades.

En el área de la informática educativa son muchas las clasificaciones de este tipo de software que pueden ser encontradas, algunas de las clasificaciones definidas en Marqués (Marquès, 1996), Graells (Graells, y otros, 1999), García (González García, 2008) y resumidas en Marquès (Marquès, 2009) son realizadas de acuerdo a:

- *Los contenidos:* temas, áreas curriculares, etc.
- *Los destinatarios:* criterios basados en niveles educativos, edad, conocimientos previos.
- *Su estructura:* tutorial (lineal, ramificado o abierto), base de datos, simulador, constructor, herramienta.

- *Sus bases de datos:* cerrada o abierta.
- *Los medios que integra:* convencional, hipertexto, multimedia, hipermedia, realidad virtual.
- *Su "inteligencia":* convencional, experto o con inteligencia artificial.
- *Los objetivos educativos que pretende facilitar:* conceptuales, procedimentales, actitudinales o considerando otras taxonomías de objetivos.
- *Las actividades cognitivas que activa:* control psicomotriz, observación, memorización, evocación, comprensión, interpretación, comparación, relación (clasificación, ordenación), análisis, síntesis, cálculo, razonamiento (deductivo, inductivo, crítico), pensamiento divergente, imaginación, resolución de problemas, expresión (verbal, escrita, gráfica), creación, exploración, experimentación, reflexión metacognitiva, valoración.
- *El tipo de interacción que propicia:* recognitiva, reconstructiva, intuitiva/global y constructiva.
- *Su función en el aprendizaje:* instructivo, revelador, conjetural y emancipador.
- *Su comportamiento:* tutor, herramienta y aprendiz.
- *El tratamiento de errores:* tutorial y no tutorial.
- *Sus bases psicopedagógicas sobre el aprendizaje:* conductista, cognitivista y constructivista.
- *Su función en la estrategia didáctica:* entrenar, instruir, informar, motivar, explorar, experimentar, expresarse, comunicarse, entretener, evaluar y proveer recursos (calculadora, comunicación telemática).
- *Su diseño:* centrado en el aprendizaje, centrado en la enseñanza y proveedor de recursos.

Aunque cualquier software educativo pudiera ser clasificado en alguna de las categorías antes listadas propuesta por Marquès (Marquès, 2009) para una mejor comprensión y especificación, se han clasificado los software generalmente usados como medio de enseñanza en las técnicas de programación, en cuanto a su objetivo instructivo, formación de habilidades y características.

La realización de esta clasificación ha tenido como base el conjunto de habilidades que deben formase en un programador, resumidas en la **Tabla 1.2**.

La clasificación que se ha realizado en la **Tabla 1.3** permite profundizar en el estudio y análisis de este tipo de aplicaciones informáticas, facilitando la comprensión de sus ventajas y desventajas, así como posible uso futuro.

Tabla 1.3: *Clasificación de los software orientados al aprendizaje de las técnicas de computadoras según la taxonomía de Bejamín Bloom propuesta por la ACM y la IEEE.*

Grupo	Niveles	Características
I	1, 2	Diseño y representación de algoritmos
II	1, 2, 3	Diseño y representación de algoritmos, visualización y/o entornos de desarrollo integrados
III	4, 5, 6	Formación de habilidades y apropiación de conceptos

Fuente: *Elaboración propia.*

Seguidamente se realiza un análisis detallado de cada uno de los grupos propuestos.

1.2.1. Grupo I.

Como la **Tabla 1.3** indica, su característica fundamental es el diseño y la representación de algoritmos, los objetivos instructivos básicos a cumplir son los de los niveles 1 y 2 de Benjamín Bloom (Ver **Tabla 1.2**). Por lo general son usados en los cursos de Introducción a la Programación.

Como desventaja se pueden mencionar la necesidad que tienen estos sistemas de incluir un lenguaje de algoritmización propio, el cual debe ser dominado por el estudiante para que funcione correctamente, elemento que dificulta el proceso de aprendizaje. Un gran porciento de estos sistemas informáticos requiere de un IDE específico para su funcionamiento, lo cual incide de forma negativa, al demandar tiempo y esfuerzo por parte de los estudiantes para dominar las funcionalidades de dicho entorno.

Su ventaja fundamental radica en el hecho que un buen diseño de las actividades docentes, puede lograr que se cumplan los objetivos instructivos que fueron propuestos en la preparación de dicha actividad.

Como ejemplo de sistemas que se ajustan al análisis antes realizado se encuentran:

Loro

El proyecto está enfocado en el desarrollo de una herramienta informática que asista a los estudiantes que se inician en el mundo de la programación, es abierto, modular y flexible (Ruenda, 2007). Posee un leguaje propio muy parecido a un pseudocódigo y un IDE orientado a los estudiantes. Está centrado en lo que sus creadores han denominado el *"nivel I"*:

Hace gala de haber sido diseñado bajo los siguientes principios: programar para una interfaz, no para una implementación; diseño por contrato; desarrollo dirigido por pruebas (Ruenda 2007).

Lexico

Diseñado para los estudiantes que se inician en la programación pero que comienzan con un curso de algoritmización. Define su propio lenguaje, elemento que dificulta el aprendizaje, sus creadores lo consideran Orientado a Objetos. Los algoritmos codificados en *Lexico* pueden ser compilados y ejecutados. El sistema tiene las opciones de traducir el algoritmo en *Lexico* a código: CSharp y Visual Basic, JavaScript, JSchar y código MSIL; además puede crear dll y agregar dll a un algoritmo que se esté desarrollado en el sistema.

1.2.2. Grupo II.

Desarrollados con el fin de cumplir los objetivos educativos de los niveles 1, 2 y 3 del diagrama de Benjamín Bloom (Ver **Tabla 1.2**). Este tipo de sistema informático por lo general es usado en los dos primeros cursos de las Técnicas de Programación, es decir, Introducción a la Programación y Programación Orientada a Objetos.

Las desventajas de estos sistemas informáticos dependen de la forma en la que hayan sido desarrollados por lo cual se hace un análisis más específico de algunos ejemplos.

Scratch

Scratch está concebido para niños y adolescentes de aproximadamente 10 a 18 años, bajo la hipótesis que mientras el niño trabaja en los proyectos *Scratch* sobre historias o arte interactiva desarrolla un pensamiento algorítmico y matemático (Maloney, y otros, 2004).

Este software presenta una interfaz intuitiva, sencilla y agradable, permite crear recursos como: animaciones, juegos, videos y postales. Todo lo anterior usando bloques de programación auto conectables, diferenciando por el color las funciones de los conjuntos de bloques.

Scratch, a pesar de ser diseñado para niños, comienza a usarse en algunas universidades en la enseñanza de la Introducción a la Programación. La herramienta permite apropiarse de conceptos tan simples como el de una variable y tan complejos como los de programación concurrente, todo dependerá de la imaginación del profesor y del estudiante.

Se han considerado como elementos desfavorables, la inclusión de lenguaje de programación e IDE propios, alejados de los lenguajes de programación reales. El propio carácter lúdico que presenta la aplicación informática extravía demasiado al estudiante de los procesos reales del desarrollo de software.

Alice

Alice es una herramienta de programación 3D que permite la creación fácil de historias, juegos interactivos o videos, que pueden ser compartidos en la red. Es una herramienta de enseñanza para los estudiantes que se inician en la Programación Orientada a Objetos (Dann, y otros, 2006).

Posee una interfaz interactiva que permite a los estudiantes apropiarse de los principales conceptos de la programación mientras crean sus juegos o historias animadas. Su ventaja fundamental es que permite ver de una forma inmediata el resultado de sus programas y la relación entre los conceptos, interacción entre objetos, comportamiento, atributos, etc.

Es considerado una excelente herramienta pero como elementos negativos se pueden señalar los mismos que se argumentaron para el *Scratch*.

Jeliot 3

Jeliot 3, un software que es el último descendiente de una familia de programas denominada: *"Jeliot Family"*. El proceso de desarrollo suma más de 10 años y varias versiones, de lo que sería la raíz de dicha familia.

Es un conjunto de programas con herramientas de visualización, diseñado para ayudar a los estudiantes en el aprendizaje de la programación. Su característica fundamental es la visualización de los datos y flujo del programa a medida que este se ejecuta (Myller, 2004).

Esta característica del software tiene la ventaja de que el estudiante solo tiene que concentrarse en la corrección del programa y no en la forma de visualización. No usa un lenguaje propio sino que realiza la visualización de código real, hasta el momento solamente acepta código fuente en Java.

Para lograr la animación, *Jeliot 3* crea un código intermedio (*E-Code*) que es ejecutado por una máquina virtual (*E-Machine*) haciendo posible que en la visualización se pueda retroceder a pasos anteriores (Myller, 2004).

Dentro de los logros actuales se pueden identificar: la creación de un paquete para la integración con la plataforma Moodle; la creación del *JeCo*, un software que combina la visualización del *Jeliot* y la colaboración; el *BlueJ*, una extensión del *Jeliot* que mejora características y potencialidades de este último (Ben-Ari, y otros, 2002).

Considerado como una potente y flexible aplicación informática que se recomienda para el uso del proceso de enseñanza-aprendizaje de las técnicas de programación de las asignaturas introductorias. Tiene como ventaja que funciona sobre lenguajes reales, como desventaja que tiene su propio entorno de desarrollo y que no es funcional para algoritmos complejos.

Greenfoot

El proyecto *Greenfoot* está realizado bajo el principio de: la creación de un entorno de desarrollo integrado Java que presente herramientas que asistan el proceso de enseñanza-aprendizaje.

Greenfoot es un IDE que combina el desarrollo Java, soporta el lenguaje completo y que tiene como características un navegador de clases, ejecución interactiva, ejecución paso a paso y un framework para la creación de animación bidimensional, juegos y simulación de otros elementos (Henriksen, y otros, 2004). Además permite exportar el escenario como un applet Java acompañado de un fichero HTML y publicar el escenario en un sitio web para ser revisado y comentado por todos. El framework tiene dos responsabilidades fundamentales: hacer fácil la representación gráfica de objetos y el control de ejecución de un ciclo de simulación (inicio, parada y paso a paso). Posee un escenario que posibilita la interacción directa con los objetos y clases, una de sus más potentes características.

Este sistema ha sido probado en los cursos de Introducción a la Programación Orientada a Objetos con un enfoque lúdico en varias universidades, haciendo uso de su potencialidad para construir mundos animados y la interacción con los objetos y clases, arrojando resultados muy positivos (Külling, y otros, 2005). De todos los antes analizados se considera como el más avanzado y completo para utilizar en un curso de programación, pudiera ser el complemento perfecto en la preparación de las clases. Aunque es necesario señalar como elemento desfavorable la necesidad de usar su propio entorno.

1.2.3. Grupo III.

Desarrollados con el fin de cumplir los objetivos instructivos de los niveles 4, 5 y 6 del diagrama de Benjamín Bloom (Ver **Tabla 1.2**). Estas aplicaciones informáticas pueden ser utilizadas en cualquier asignatura de la disciplina de TPC, todo dependerá de la colección de problemas y la intencionalidad con la que sea utilizada.

A continuación se analizan algunas aplicaciones web, de las cuales no se tiene registro de uso con fines docentes en la bibliografía consultada; pero que por sus características pueden ser clasificados dentro de este grupo.

Jueces en línea

Existen muchas aplicaciones a nivel mundial cuyo objetivo es estimular el aprendizaje de la programación para lo cual usan como elemento motivacional las competencias entre programadores, específicamente nos referimos a los conocidos jueces en línea (Judge on-line, en inglés). La mayoría de estas aplicaciones informáticas son patrocinadas por la ACM:

- *http://acm.zju.edu.cn:* patrocinado por la Universidad de Zhejiang, China y la ACM.
- *http://acm.timus.ru:* patrocinado por la Universidad Estatal Ural, Rusia.
- *Copa Pascal:* es un evento dirigido por el Departamento de TPC y patrocinado por la dirección de la UCI.
- *Cátedra de Programación Avanzada:* cuenta con la publicación de un juez en línea (*http://cpav.uci.cu/)*.

Otros jueces en línea han sido publicados en la universidad de forma inestable, destacándose en los dos últimos cursos un juez en línea hospedado en la facultad 8 (*http://onlinejudgef8.uci.cu:5800/JudgeOnline/*).

Las aplicaciones informáticas antes analizadas tienen en común que solo incluyen como estilo de competencia el estilo libre, las competencias centralizadas y una amplia colección de problemas sobre los cuales se opera. Cada estilo de competencia, dependiendo de sus características, apoyará el desarrollo de habilidades; por ejemplo, mediante la participación del estilo libre y la competencia centralizada el estudiante puede desarrollar habilidades de: pensamiento lógico, lógico – matemático, lógico – algorítmico, interpretación, codificación, análisis y diseño de algoritmos de variados niveles de complejidad, trabajo en equipo y liderazgo. Todas estas características forman parte de las deseadas en la solución propuesta. No obstante, tienen como características limitantes: su hospedaje en servidores remotos (excepto los hospedados en la UCI), lo que los hace no accesibles por todos los estudiantes de la universidad; la inclusión de pocos estilos de competencia, limitando su personalización y restringiendo el uso docente.

- *http://imaginecup.com:* patrocinado por Microsoft, se desarrolla a nivel mundial. Su objetivo fundamental es la caza de talentos. Actualmente cuenta con más de diez estilos de competencias, entre ellos se destacan: diseño de software, desarrollo empotrado, desarrollo web, algoritmos, fotografía en arte digital, cortos en arte digital y diseño de interfaz. Las competencias se realizan por equipos, cuya cantidad de integrantes varía dependiendo del estilo de competencia, se realiza a nivel internacional en varias etapas.
- *http://www.mslatam.com/latam/msdn/comunidad/dce2005/:* popularmente conocido como *"Desarrollador Cinco Estrellas"*, patrocinado por Microsoft. Permite la certificación en línea. Una vez registrado se le suministran materiales de estudio a los usuarios y luego se le aplica un examen que va subiendo de nivel, las llamadas estrellas. En dichos exámenes se deben responder cierta cantidad de preguntas.

Imagecup, tiene como ventaja y característica deseada, la inclusión de varios estilos de competencia y *Desarrollador Cinco Estrellas*, su estilo de generar exámenes en-línea. Como elementos negativos: hospedaje remoto y ser patrocinados por Microsoft lo cual implica que muchas de sus funcionalidades deben ser pagadas para poder usarlas.

Los jueces en línea, antes analizados, solo son una pequeña representación de la amplia gama de este tipo de sistemas que pueden ser encontrados en la gran red de redes. Todos presentan características similares, solo varían en la cantidad de lenguajes y compiladores que soportan, el volumen de problemas y los datos almacenados. Excluyendo el *"Imagecup"* y el *"Desarrollador Cinco Estrellas"*, presentan dos estilos de competencias y no se tiene información de su uso con fines docentes.

Es pertinente aclarar que después de este proyecto haber sido terminado fue publicado por la Universidad de Valladolid, el juez en línea "*EduJudge*[1]" con el fin de responder a las demandas pedagógicas de los usuarios de incrementar el carácter pedagógico del uso de los jueces en líneas para las matemáticas y la computación.

1.3. METODOLOGÍAS DE DESARROLLO DE SOFTWARE.

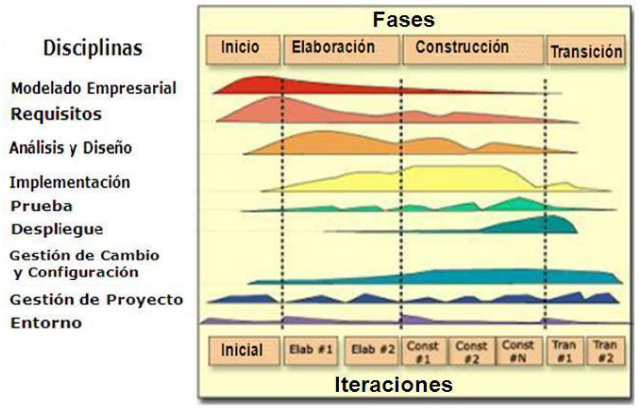
Son muchas las metodologías de desarrollo de software[2] que han proliferado en los últimos años, algunas derivadas de las metodologías tradicionales, pesadas o robustas y otras de las metodologías ágiles, como suelen ser conocidas y agrupadas. Como no es objetivo de esta investigación establecer una comparación entre las mismas, este trabajo se limita al análisis de dos de las más representativas de cada uno de los grupos antes mencionados. Como metodología tradicional, el Proceso Unificado de Desarrollo de Software (RUP, por sus siglas en inglés); como metodología ágil, Programación Extrema (XP, por sus siglas en inglés).

1.3.1. Proceso Unificado de Desarrollo de Software.

RUP es un proceso de desarrollo de software que ofrece un conjunto de actividades necesarias para transformar los requisitos de usuario en un sistema de software. El Proceso Unificado es más que un simple proceso: es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyectos (Jacobson, y otros, 1999).

RUP, es un proceso dirigido por casos de uso, centrado en la arquitectura e iterativo incremental. Como muestra la **Fig. 1.1** está compuesto por nueve flujos de trabajo o disciplinas que tienen lugar sobre las cuatro fases.

Figura 1.1: *Los flujos o disciplinas de trabajo que tienen lugar sobre las cuatro fases en RUP.*



Fuente: *Tomado de* (Rational Unified Process, 2006)**.**

[1] Proyecto con fines docentes que incluye la integración con el juez en línea de la Universidad de Valladolid, por la cual es patrocinado. El propósito es contribuir al desarrollo de habilidades tecnológicas y matemáticas, también la motivación, la creatividad y la capacidad para la gestión de tiempo. Es necesario aclarar que el proyecto se encuentra en su fase inicial, aun no puede ser explotado.

[2] En la amplia gama de bibliografía revisada para conceptualizar este término, algunas tan prestigiosas como Jacobson y colegas (Jacobson, y otros, 1999), Preesman (Pressman, 1998) y Larman (Larman, 1999). Se encontró una gran variedad de definiciones, donde no existe una contradicción conceptual. No obstante, se decide ajustarnos a la siguiente definición de (Chapman, 2007) "Metodología de desarrollo de software": la colección documentada de políticas, procesos y procedimientos utilizados por un equipo de desarrollo o a la organización práctica de la ingeniería de software.

El proceso unificado de desarrollo tiene la característica de generar un gran volumen de documentación, lo cual es ventajoso para grandes proyectos de desarrollo; además puede ser adaptado a proyectos pequeños.

Entre las características de *RUP* definidas por Rational Software Corporation en el año 2003 se encuentran las siguientes: el desarrollo iterativo del software, la gestión de los requerimientos, una arquitectura basada en componentes, el uso de software de modelado visual (ver epígrafe 1.5.2), la verificación de la calidad del software y el control de cambios del software.

Todo lo antes mencionado, además de definir la generación de artefactos y amplia documentación por cada fase y flujo que tributan a un buen producto final, han hecho de *RUP* una de las metodologías más usada en la actualidad. Su característica de idear un desarrollo iterativo. Su enfoque sistémico para gestionar los requisitos y la arquitectura basada en componentes la convierten en la metodología más adecuada para desarrollar la solución propuesta.

1.3.2. Metodologías ágiles.

Muchos son los equipos de desarrollos que enfrentan proyectos de mediana o pequeña envergadura, donde las metodologías tradicionales son de difícil aplicación, es entonces cuando se inclinan por una metodología ágil. Entre ellas se puede mencionar: Adaptative Software Development (ASD, por sus siglas en inglés), Scrum, eXtreme Programming (XP), Feature Driven Development (FDD, por sus siglas en inglés), Dynamic System Development Method (DSDM, por sus siglas en inglés), entre otras. El estudio realizado centró su atención en XP, por ser la de mayor éxito y aceptación en la comunidad de desarrolladores.

XP se basa en la realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y responsabilidad a la hora de afrontar los cambios. Se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

Teniendo en cuenta que la solución propuesta no se desarrollará en ciclos cortos, que el énfasis se encuentra en la documentación generada para garantizar la futura evolución del sistema y no en la capacidad de respuesta a los cambios, esta metodología no es considerada como la óptima para el desarrollo de la aplicación.

1.4. PARADIGMAS DE PROGRAMACIÓN.

En el área de la informática este término se ha consolidado como filosofía de programación o patrón de una disciplina, es decir, el método de estructuración que usa el código de un lenguaje, según un conjunto de teorías y principios previamente definidos.

Existen muchos paradigmas de programación que han ido evolucionando en el transcurso de los años, entre ellos se encuentran:

- *Programación lógica y programación funcional:* ambos pertenecen, jerárquicamente, al paradigma declarativo, donde para resolver un problema no se especifica el cómo sino el qué hacer, por lo que no se ajusta a nuestras necesidades, razón por la cual no se profundiza en su análisis.

- *Programación estructurada, modular, orientada a objetos y orientada a aspectos:* todos pertenecen, jerárquicamente, al paradigma imperativo. En ese mismo orden fueron surgiendo, cada nuevo paradigma contiene al paradigma precedente. Este trabajo se ha centrado la atención en los dos últimos antes mencionados.

Existen otros paradigmas de programación como: la programación concurrente, con restricciones, multi-agentes, entre otros; aplicables a dominios de problemas más específicos, ajenos al objetivo de la investigación, por lo cual no pasan a ser objeto de estudio.

1.4.1. Programación orientada a objetos.

De la lectura de Deitel (Deitel, y otros, 2005), Meyer (Meyer, 1999), Booch (Booch, 1998), Jacobson y colegas (Jacobson, y otros, 1999) y Pressman (Pressman, 1998) se infiere que la tecnología de objetos, que incluye el diseño y la codificación, tiene sus pilares en los conceptos de: modularidad, reutilización, protección de la información, abstracción, clase, objeto, herencia y polimorfismo. Siendo el concepto de objeto la piedra angular, mediante el cual se realiza una representación de conceptos y fenómenos del mundo real.

Deitel (Deitel, y otros, 2005) nos comenta como algunas empresas consideran que el principal beneficio de la orientación a objetos es hacer: "…el desarrollo de software más comprensible, mejor organizado, más fácil de mantener, modificar, depurar y evolucionar". Pues el mismo autor antes citado argumenta como el 80% del costo del software no está destinado a su construcción sino a su mantenimiento y evolución. Las características del paradigma lo convierten en el ideal para el desarrollo de la solución propuesta.

1.4.2. Programación orientada a aspectos.

Este es uno de los términos más jóvenes en el área de la programación, incluso algunos programadores aún no se han decidido por aceptar si estamos frente a un nuevo paradigma de programación o no.

Reina (Reina Quintero, 2000) define la programación orientada a aspectos (POA, en lo adelante) como: "…nueva metodología de programación que aspira a soportar la separación de competencias….es decir, que intenta separar los componentes y los aspectos unos de otros, proporcionando mecanismos que hagan posible abstraerlos y componerlos para formar todo el sistema. En definitiva, lo que se persigue es implementar una aplicación de forma eficiente y fácil de entender."

La POA define como concepto clave el aspecto: considerado como unidad modular que se disemina por la estructura de otras unidades funcionales. Los aspectos existen tanto en la etapa de diseño como en la de implementación. Un aspecto de diseño es una unidad modular del diseño que se entremezcla en la estructura de otras partes del diseño. Un aspecto de programa o de código, es una unidad modular del programa, que aparece en otras unidades modulares del programa (Kiczales, y otros, 2002). Como objetivos fundamentales persigue: separar conceptos y minimizar las dependencias entre los conceptos.

Este nuevo paradigma presenta como ventajas:

- Un código menos complicado, más natural y más reducido.

- Una mayor facilidad para razonar sobre las materias, ya que están separadas y tienen una dependencia mínima.
- Más facilidad para depurar y hacer modificaciones en el código.
- Se consigue que un conjunto grande de modificaciones en la definición de una materia tenga un impacto mínimo en las otras.
- Se tiene un código más reusable y que se puede acoplar y desacoplar cuando sea necesario.

A pesar de todo lo antes mencionado y lo prometedor de la propuesta, se considera que al igual que la POO en sus inicios, aún faltan muchos años antes de que este paradigma se consolide. Incluso es necesario que se incrementen las herramientas y se perfeccionen las metodologías de diseño que soporten este paradigma, lo cual justifica su no selección para llevar a cabo el desarrollo de este trabajo.

1.5. LENGUAJES DE PROGRAMACIÓN.

Terry define (Terry, 1996) el término de leguaje como: "…número finito de cadenas sobre un alfabeto que puede ser definido listando todas las cadenas o dando una regla para su derivación".

Específicamente, los lenguajes de programación (LP, en lo adelante) se ajustan a la definición antes mencionada, aunque incluyen muchas más restricciones, donde las cadenas no son más que las palabras reservadas y los identificadores. Los LP pueden ser de propósito general o específico, los primeros, como su nombre lo indican pueden usarse para el desarrollo que "cualquier" tipo de software; los segundos, solo para un ámbito específico.

El interés de este trabajo incide sobre los LP para el desarrollo de aplicaciones web, razón por la cual el estudio se ha limitado a realizar el análisis de un número reducido de este tipo de LP.

1.5.1. Lenguajes para el desarrollo de aplicaciones web.

Su verdadero potencial se encuentra en el desarrollo de aplicaciones web y para el desarrollo de esta investigación se ha definido como centro de atención los LP que sean multiplataforma y soporten la orientación a objetos.

Es necesario tener presente que las aplicaciones web son aquellas cuya interfaz con los actores y clientes del sistema se realiza mediante el protocolo HTTP y que están basados en la arquitectura cliente – servidor, por tal razón se analizan tecnologías del lado del cliente y tecnologías del lado del servidor.

- *Common Gateway Interface (CGI, por sus siglas en inglés):* estándar que especifica cómo se va a comunicar un servidor HTTP con una aplicación, escrito en cualquier LP ejecutándose en la misma computadora que recibe sus parámetros a través de variables de entorno. Las aplicaciones *CGI* necesitan crear un nuevo proceso en el servidor para atender cada solicitud del cliente por lo que hacen un uso intensivo de los recursos del servidor web. Producto de la ineficiencia que caracteriza a las aplicaciones *CGI* no será objeto de interés en el desarrollo de este trabajo.
- *Servlet:* introducida por Sun Microsystem, es la base del desarrollo de aplicaciones web usando Java, una de sus más importante tecnologías y la base de otras tecnologías

(Kurniawan 2002, 2). Tiene como desventaja la necesidad de incrustar las etiquetas HTML y la necesidad de la intervención de los programadores del servlet para realizar cambios, por pequeños que estos sean. Esta desventaja ocasionó que se desarrollara el JavaServerPage (JSP, por sus siglas en inglés), una extensión de servlet. Al estar basado en Java consume gran cantidad de recursos de memorias, presenta alta calidad en la detección de errores con el tipo de error y bajos tiempos de respuesta. Por las características antes mencionadas no se profundizará más en el estudio de la misma.

- *API de los servidores web:* protocolos de extensión que permiten crear librerías dinámicas que son cargadas por el servidor web más conocidas como *ISAPI* (Internet Server Application Programming Interface). Las *API* de los servidores web son librerías dinámicas que se cargan en el entorno de memoria del servidor. Por cada solicitud del cliente se levanta un hilo en el servidor, en sustitución del costoso mecanismo de crear un nuevo proceso. Esto permite hacer un uso más eficiente de los recursos del servidor y mejorar los tiempos de respuesta al cliente. Dentro de estas entran los Script del lado del servidor como: ASP, PERL, PHP, JSP entre otras.

PHP

Ha sido diseñado especialmente para crear páginas web dinámicas y en la interpretación del lado del servidor. Aunque actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas como aplicaciones de escritorio. Puede ser embebido dentro del código HTML, es multiplataforma y orientado a objetos desde *PHP3*. Otras de las características que se enumeran en Gallego (Gallego Vázquez, 2003), Converse y colegas (Converse, y otros, 2004) y Achour y colegas (Achour, y otros, 2008) son: alto rendimiento, capacidad de conexión con gran parte de los manejadores de datos existentes, posibilidad de expandir su potencial utilizando los módulos, amplia documentación donde se explican las funciones del LP, biblioteca nativa de funciones sumamente amplia, no requiere definición de tipos estática, manejo de excepciones (desde php5), amplia comunidad de soporte y desarrollo y es libre.

Por todo lo antes mencionado se define como la opción más viable para el desarrollo de la propuesta de solución, dado que la aplicación informática a desarrollar se requiere que sea desarrollada bajo un lenguaje multiplataforma, rápido, orientado a objetos y use la tecnología de script del lado del servidor.

1.5.2. Lenguajes de modelado.

Los lenguajes de modelado, tanto los textuales como los gráficos, permiten la representación del comportamiento y la estructura de los sistemas en el mundo real. La atención del estudio de estos lenguajes de modelado estará centrada en los lenguajes de modelado gráfico (LMG, en lo adelante), cuyo objetivo es según Zapata (Zapata, y otros, 2009): “especificar o describir métodos o procesos mediante modelos visuales y diagramas que realizan esa representación de manera precisa, entendible y económica”.

Object Management Group (OMG, por sus siglas en inglés), propone lenguajes de modelado gráfico orientados a cubrir varias situaciones de acuerdo con sus perspectivas. Es así como cobran importancia lenguajes como *UML*, *SysML*, *WebML*, entre otros. El desarrollo de estos lenguajes ha facilitado el uso y desarrollo de las herramientas *CASE*.

A pesar de existir una amplia variedad de *LMG*, como pueden ser los lenguajes de modelado gráfico para la especificación de circuitos o el proceso de ingeniería de software, el estudio se ha limitado solamente a estos últimos.

UML

El Lenguaje Unificado del Modelado o Unified Modeling Language, es la sucesión de una serie de métodos de análisis y diseño orientado a objetos que aparecen a fines de los 80 y principios de los 90. El UML, fusiona los conceptos de la orientación a objetos aportados por Booch, OMT y OOSE (Jacobson, y otros, 1999).

En Booch y colegas (Booch, y otros, 1999), Jacobson y colegas (Jacobson, y otros, 1999), Larman (Larman, 1999) y Awad (Awad, 2005) se definen como objetivos de *UML*: visualizar, especificar, construir y documentar procesos de negocio. Compuesto como diagrama que se dividen en tres categorías: estructura (clases, componentes, estructura compuesta, despliegue, objetos y paquetes), comportamiento (actividades, casos de uso y máquina de estados) e interacción, un subgrupo de los de comportamiento (secuencias, comunicación, vista de interacción y tiempos). Es definido por Booch y colegas (Booch, y otros, 1999) como el estándar de facto para modelar aplicaciones de software orientado a objetos. Todo lo antes mencionado justifica la selección del *UML* para especificar y documentar la solución propuesta en esta investigación.

1.5.3. Lenguajes de consulta.

De la lectura de Tsai (Tsai, 1990), Kroenke (Kroenke, 1996) y Korth y colegas (Korth, y otros, 2006) se infiere que los lenguajes de consulta son aquellos mediante los cuales los usuarios solicitan información de la base de datos.

Dentro de este tipo de lenguajes, aquellos en los que el usuario inserta una serie de instrucciones de las cuales se deriva una secuencia de operaciones en la base de datos para calcular el resultado deseado son los denominados procedimentales. En los no procedimentales el usuario describe la información deseada sin dar un procedimiento específico para obtener dicha información (Korth, y otros, 2006).

Estos lenguajes están fundamentados en el álgebra relacional mediante el cual se definen operadores que funcionan sobre las tablas para llegar al resultado deseado. El álgebra relacional toma dos o más tablas como entrada y puede producir una nueva tabla como resultado de la serie de operaciones.

SQL

Lenguaje de Consulta Estructurado (SQL, por sus siglas en inglés y en lo adelante) permite a los desarrolladores acceder y manipular base de datos relacionales, se ha ido desarrollando con la explosión de las Tecnología de la Información y las Comunicaciones. Debido a la gran variedad de versiones (*PL/SQL*, *Transact-SQL*, *PL/pgSQL*, *SQLPL*, etc) que responden a problemas específicos, los organismos encargados de la normalización han publicado una amplia lista de características comunes entren las que se destacan (Kline, 2004): Iniciar y finalizar conexiones, control de la ejecución de una serie de sentencias *SQL*, proporcionar información de diagnóstico y plantear las excepciones y errores, comportamiento predeterminado de control y otros

parámetros para una sesión, ajuste del punto inicial y final de una transacción, capacidad de procesamiento condicional, control de las funciones de flujo entre otras.

1.6. HERRAMIENTAS.

Para llevar a cabo el desarrollo de la aplicación informática que responda a los requerimientos de esta investigación se hace necesario el uso de un conjunto de herramientas cuyas funcionalidades respondan a los lineamientos, principios y teorías de: metodologías, paradigmas, y lenguajes que han sido seleccionados, de modo tal que las mismas contribuyan a obtener un producto de calidad como lo señalara Pressman (Pressman, 1998).

Figura 1.2: *Modelo en capa para un proceso ingenieril propuesto por Pressman sobre el cual debe sustentarse el compromiso de calidad de la organización.*

Fuente: *Tomado de* **(Pressman, 1998)**.

1.6.1. Ingeniería de software asistida por computadora.

Para las fases de modelado se hizo uso de las herramientas clasificadas como *CASE* o Ingeniería de Software Asistida por Computadora. Estas herramientas *CASE*, de forma general, cubren en mayor o menor medida todo el ciclo de vida de un proyecto. Ofrecen la capacidad de modelar y visualizar sus procesos de negocios contribuyendo de forma positiva en la productividad, calidad, eficiencia, disminución de posibles errores e incrementa el entendimiento entre los clientes y el equipo de desarrollo y hasta el entendimiento entre los miembros del equipo en sí.

De las tantas herramientas que son clasificadas como *CASE* algunas de las que más éxito han alcanzado son: Rational Rose, Enterprise Architect y Visual Paradigm.

Rational Rose, dispone de una suite que cubre el ciclo completo de desarrollo, solo está disponible para el sistema operativo Windows y es software propietario (Rational Rose, 2009), razón por la cual no se considera como opción de trabajo. *Enterprise Architect*, es una herramienta muy completa, incluye características avanzadas como la ingeniería inversa, integración con diversos IDE y soporte para diferentes metodologías, es software propietario y no es multiplataforma (Enterprise Architect, 2009), por lo que tampoco es considerado como una opción viable, lo que nos lleva profundizar en el *Visual Paradigm*.

Visual Paradigm

Herramienta *CASE* profesional que da soporte al análisis y diseño orientados a objetos, construcción, pruebas y despliegue del ciclo de desarrollo de un software. Permite la rápida construcción de aplicaciones de calidad, brinda la posibilidad de crear todos los tipos de diagramas de clases, generación de código e ingeniería inversa para más de diez lenguajes de programación.

El sitio oficial del *Visual Paradigm* enumera otro conjunto de características como: permitir exportar la documentación a varios formatos incluidos *MS Word* y *PDF*, comparación entre diagramas resaltando las diferencias, diseño de patrones y trazabilidad en el desarrollo de los modelos, soporta el proceso de ingeniería de varios gestores de base de datos (Visual Paradigm, 2009).

Tiene como ventajas incluir funcionalidades para el desarrollo colaborativo, integración con varios IDE y controladores de versiones, es multiplaforma y puede importar ficheros en el formato de *Rational Rose*. Aunque no es completamente libre tiene versiones para el uso académico. Por todo lo antes planteado es seleccionado como la herramienta para llevar a cabo el proceso de modelado de la solución propuesta en esta investigación.

1.6.2. Gestores de contenido.

Los sistemas de gestión de contenidos (Content Management Systems, en inglés y *CMS* en lo adelante) son software que se utilizan principalmente para facilitar la gestión de las Webs, ya sea en Internet o en una intranet. Los *CMS* proporcionan un entorno que posibilita la actualización, mantenimiento y ampliación de la Web con la colaboración de múltiples usuarios, aportan herramientas para que los creadores sin conocimientos técnicos puedan concentrarse en el contenido. La funcionalidad de los sistemas de gestión de contenidos se divide en cuatro categorías: creación de contenido, gestión de contenido, publicación y presentación (Almarcegui Aguerri, y otros, 2009).

En las distintas comunidades de software libre se han creado un gran número de sistemas de gestión de contenidos útiles y de gran calidad como: *Mambo*, *Plone*, *Joomla!*, *Typo3*, *PHPnuke*, *Drupal*, *Moodle*, entre otros.

Almarcegui y colegas (Almarcegui Aguerri, y otros, 2009) al resumir la comparación de varios *CMS*, concluyen que cada gestor de contenidos tiene sus propias ventajas y desventajas, además señalan que la selección del *CMS* adecuado responde a las características del software a desarrollar y los intereses del equipo de desarrollo.

Teniendo en cuenta el estudio y la revisión realizada, el interés inicial se inclinó por: *Moodle*, *Drupal* y *Jommla!*.

Moodle, es un *CSM* orientado a los procesos académicos, publicación de cursos y otras funcionalidades con carácter docente; aunque tiene características deseables en el desarrollo de nuestro trabajo, la selección de este *CMS* complicaría el proceso de desarrollo y no se contaría en el beneficio de reutilizar los complementos implementados por la comunidad *Moodle*, pues en su mayoría van encaminados a resolver problemas específicos. *Drupal*, el gran competidor de *Joomla!,* cualquiera de los dos pudiera ser la opción, no obstante teniendo en cuenta que el desarrollo de complementos en *Drupal* es más complejo, que el dominio y la bibliografía existente sobre *Joomla!* es superior, se ha seleccionado el último de los antes mencionados para llevar a cabo la solución propuesta.

Joomla!

Es un *CMS* de código abierto desarrollado en *PHP* bajo una licencia *GPL* y multiplataforma. Es usado para publicar en Internet e intranets utilizando una base de datos *MySQL* (LeBlanc, 2007).

Almarcegui y colegas (Almarcegui Aguerri, y otros, 2009) señalan que *Joomla* puede administrarse desde un navegador web y una conexión a Internet y su interface funciona exclusivamente por *HTTP*. No requiere grandes conocimientos de informática ni de programación o HTML para actualizar, mantener y personalizar los contenidos de las webs por parte de los administradores del sitio web.

Joomla 1.5, su última versión estable, ha tenido más de 11 millones de descargas y se le han producido más de 3000 extensiones; por lo que su comunidad se cataloga como una de las más dinámicas (Joomla Spanish, 2009).

LeBlanc (LeBlanc, 2007) y el sitio oficial de *Joomla* en español (Joomla Spanish, 2009) describen a los componentes de *Joomla* como las aplicaciones más importantes que juegan un papel fundamental en la gestión de contenidos del *CMS* ya que estos son pequeños programas independientes entre sí, que se integran al Core (Núcleo del sistema) de *Joomla*, además cuenta con los módulos que son aplicaciones mucho más simples que los componentes y generalmente son una extensión o complemento de algún componente.

Joomla está basado en la arquitectura modelo-vista-controlador, lo que acompañado a una estructura jerárquica de archivos, carpetas, regla de nombramiento y un panel de administración visual sencillo y flexible, ha facilita el desarrollo y escalabilidad del *CMS*.

1.6.3. Gestores de base de datos.

Una base de datos es una colección de datos lógicamente relacionados entre sí, a lo que muchos añadirían: organizados y estructurados con información referente a algo. Como define Kline (Kline, 2004) los sistemas cuyo objetivo es gestionar (definir, crear, mantener y controlar acceso) las bases de datos se les denomina Sistemas Gestores de Bases de Datos (SGBD, en lo adelante).

Un *SGBD* debe garantizar un conjunto de características argumentadas por Kline (Kline, 2004) y Hernández (Hernández, 2003) entre las que se destacan: lenguaje de definición y manipulación de datos, seguridad e integridad de los datos, gestión de usuarios, acceso concurrente, mecanismo para la realización de copias de seguridad y herramientas de administración de las bases de datos.

Para llevar a cabo el cumplimiento del objetivo propuesto se hará uso del *SGBD* relacional *MySQL*, su elección está ligada a la definición del *CMS* como se justificó en el *epígrafe 1.6.2.*

MySQL

MySQL es un *SGBD* relacional multiplataforma y libre hasta su versión 5, por lo que posee una licencia dual. Tiene una excelente integración con *Linux*, *Apache* y *PHP*. Gilfillan (Gilfillan, 2004) y Converse y colegas (Converse, y otros, 2003) destacan las siguientes características: velocidad, es mucho más rápido que la mayor parte de sus rivales; funcionalidad, dispone de muchas de las funciones que exigen los desarrolladores profesionales, como compatibilidad completa con *ACID*, compatibilidad para la mayor parte de *SQL ANSI*, volcados en línea, duplicación, funciones SSL e integración con la mayor parte de los entornos de programación. Así mismo, se desarrolla y actualiza de forma mucho más rápida que muchos de sus rivales; portabilidad, se ejecuta en la inmensa mayoría de sistemas operativos y en la mayor parte de los casos los datos se pueden transferir de un sistema a otro sin dificultad; facilidad de uso, fácil de

utilizar y de administrar. Las herramientas de *MySQL* son potentes y flexibles, sin sacrificar su capacidad de uso.

1.6.4. Entornos de desarrollo integrado.

Los *IDE* pueden ser aplicaciones que brindan funcionalidades propias o que sirven de apoyo a aplicaciones existentes. El editor de código puede dar soporte a uno o varios lenguajes. Su gran poder está centrado en las herramientas de automatización que disminuyen el tiempo de desarrollo y contribuyen a disminuir los errores del programador (Clayberg, y otros, 2006). En la actualidad se exige más de los *IDE*, como que tengan integración con el software para el control de versiones, herramientas de modelado y herramientas para el desarrollo colaborativo.

El área de la programación está inundada de los *IDE*, sin embargo uno de los más populares por su flexibilidad y posibilidad de ser extendido es el Eclipse, entorno de código abierto y multiplataforma, que cuenta con una dinámica y prolífera comunidad de desarrollo. Precisamente por ser de código abierto tiene una amplia descendencia, especializados en el desarrollo de determinados tipos de sistemas o lenguajes. Para el desarrollo de nuestro sistema usaremos el *Aptana Studio*, por lo que se profundiza en el análisis de sus características.

Aptana Studio

Es multiplataforma y de código abierto. Soporta las librerías más populares: Prototype, Scriptaculous, *Dojo*, *MochiKit*, *Yahoo UI*, *Aflax*, *JQuery* y *Rico*, *HTML*, *CSS* y *JavaScript*.

Aptana Studio contiene también información de soporte para los principales navegadores web: IE, Firefox, Opera, Netscape y Safari.

Sus creadores resaltan como otras características interesantes (Aptana, 2009): explorador de código en forma de árbol, librerías populares AJAX/Javascript, extensión de funcionalidad mediante macros y acciones, visor de errores y advertencias, servidor local para probar el código, publicar y descargar archivos directamente en un servidor de producción asociando el proyecto a una conexión *FTP*.

A pesar que para el desarrollo del proyecto se pudo seleccionar cualquier entorno que soporte el lenguaje *PHP*, la selección de Aptana está ligada al hecho que también soporta la librería de JavaScript, *ExtJS*. Además de poder instalárseles los plugin desarrollados para Eclipse, en el caso de ser necesarias otras funcionalidades.

CONCLUSIONES DEL CAPÍTULO.

A lo largo de este capítulo se han abordado los diferentes elementos teóricos sobre los cuales se sustenta esta investigación que a través de su análisis nos permite llegar a las siguientes conclusiones parciales:

- Las aplicaciones informáticas educativas utilizadas en el área de las TPC pueden ser clasificadas según las habilidades que se pueden desarrollar desde su utilización y sus características, lo que permite una mejor utilización de profesores y estudiantes.

- Las limitaciones más comunes en las aplicaciones informáticas educativas para el aprendizaje de las técnicas de programación son la inclusión de un *IDE* y lenguaje de programación propio, elementos que complican aún más el aprendizaje.
- La mayoría de las aplicaciones informáticas educativas para el aprendizaje de las técnicas de programación no permiten el desarrollo de habilidades con un enfoque sistémico, lo que limita su utilización a un breve período de tiempo.

CAPÍTULO 2: LA APLICACIÓN INFORMÁTICA "COLISEO VIRTUAL".

INTRODUCCIÓN.

Como respuesta al problema planteado en esta investigación, en este capítulo se presenta la aplicación informática "*Coliseo Virtual*". La solución propuesta es una aplicación informática con carácter lúdico clasificado dentro del *Grupo III* (ver epígrafe 1.2.3), para lo cual se definieron los siguientes principios:

- Que al ser utilizada por los estudiantes como herramienta para la asimilación de conocimientos, por su carácter lúdico, estimule la actividad productiva y motive el aprendizaje.
- Que pueda ser utilizada como medio de enseñanza para el desarrollo de habilidades con un enfoque sistémico.
- Que amplíe las posibilidades de los estudiantes, sin reducirlas al uso de IDE y lenguajes de programación específicos.
- Que posibilite el uso de varios estilos de competencia y personalización de acuerdo a los intereses de los usuarios.

Teniendo en cuenta que la metodología de desarrollo de software seleccionada para dirigir el proceso de desarrollo de la aplicación informática ha sido *RUP* (*ver epígrafe 1.3.1*), que se posee un dominio pleno del problema y de los riesgos que se puedan presentar, además de la experiencia del equipo de desarrollo; se selecciona como estrategia de planificación el patrón de ciclo vital incremental (Rational Unified Process, 2006).

Dado que para llevar a cabo el desarrollo de la solución propuesta los procesos no están claramente definidos, el flujo de información es difuso, dudoso, inexacto y en ocasiones con múltiples orígenes, tal y como la plantearan Ricardo & Martínez (Ciudad Ricardo, y otros, 2007), se procedió a realizar el modelado del dominio, mostrado en la **Fig. 2.1** y fundamentado en la siguiente estructura: forma de organización, conceptos y eventos principales del entorno en el cual se desarrolla esta investigación:

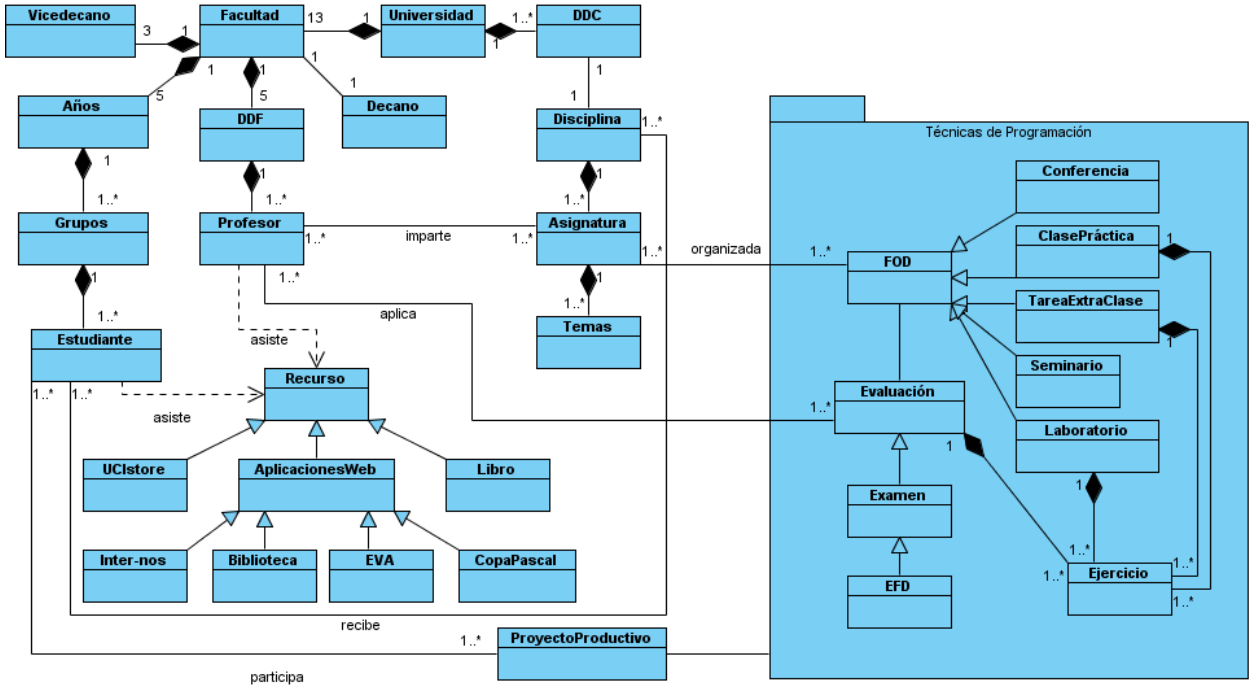
- *Estructura*: universidad, departamentos docentes centrales, facultad, departamentos docentes de facultad, años, grupos, decano, vicedecanos, jefes de departamentos, profesores y estudiantes.
- *Estructura del proceso docente*: disciplina, asignaturas, temas.
- *Formas de organización del proceso docente*: conferencias, clases prácticas, tareas extra clase, laboratorios, seminarios.
- *Eventos*: Copa Pascal, proyectos productivos, exámenes.
- *Objetos reales*: ucistore, libros, inter-nos, biblioteca, entorno virtual de aprendizaje.

El modelo conceptual presentado en la **Fig. 2.1** contribuirá en el entendimiento y comprensión por parte del equipo de desarrollo del dominio del problema facilitando el desarrollo de las fases y actividades posteriores, como puede ser:

- La definición y especificación de requisitos funcionales como el registro de usuarios que debe tener presente la estructura y organización de la universidad.

- La organización de las asignaturas para definir el estilo de competencia *"Prueba en línea"*.
- La solución propuesta, aplicación web que apoyará el proceso docente de estudiantes y profesores.

Figura 2.1: *Diagrama de clases del dominio.*



2.1. REQUISITOS DE LA SOLUCIÓN PROPUESTA.

Siguiendo la metodología de desarrollo de software RUP y el patrón de ciclo vital incremental y como parte de la fase de elaboración (Rational Unified Process, 2006) se procede a describir los requisitos funcionales y no funcionales en los *epígrafes 2.1.1 y 2.1.2*, respectivamente, que respondan al conjunto de características deseadas que debe incluir la solución propuesta y al modelo conceptual definido en la **Fig. 2.1.**

2.1.1. Requisitos funcionales.

RF1. Registrar usuarios.

El sistema permitirá que los usuarios invitados puedan registrarse y luego tener acceso al menú principal del sistema, para lo cual deberá proporcionar: nombre y apellidos, usuario que usará en el sistema, dirección de correo, universidad, facultad, año y contraseña.

RF2. Autenticarse en el sistema.

El sistema permitirá la autenticación de los usuarios, ante las solicitudes de estos de funciones para usuarios registrados, a través de la introducción por este de un nombre y una contraseña previamente suministrados al registrarse en el sistema.

RF2. Recuperar contraseña.

El usuario registrado podrá recuperar la contraseña en caso de haberla olvidado, para lo cual deberá proporcionar la dirección de correo y el sistema verificará que el correo insertado corresponde con el de un usuario registrado y procederá a enviar la contraseña.

RF2.1 Responder pregunta de verificación: el usuario inserta la respuesta a la pregunta de verificación previamente definida al registrarse en el sistema.

RF3. Modificar datos personales del usuario registrado.

El usuario registrado tendrá la posibilidad de actualizar los datos de su perfil, a través de la opción del menú principal “Mi Perfil”, que al hacer clic en dicha opción el sistema le mostrará su información personal (nombre y apellidos, usuario que usará en el sistema, dirección de correo, universidad, facultad, año y contraseña) para que pueda ser actualizada.

RF4. Gestionar usuarios registrados.

El administrador del sistema podrá gestionar (Actualizar, Eliminar, Insertar) los usuarios registrados proporcionado la información pertinente en cada caso y que se describe seguidamente:

RF4.1. Actualizar datos de usuarios registrados: el administrador podrá actualizar cualquiera de los datos de los usuarios registrados al introducir nombre del mismo.

RF4.2. Eliminar usuarios registrados en el sistema: el administrador podrá dar de baja a los usuarios registrados en el sistema para lo cual solo deberá seleccionar el usuario previamente listado.

RF4.3. Insertar nuevos usuarios registrados: el administrador del sistema podrá registrar nuevos usuarios proporcionando los datos pertinentes.

RF5. Gestionar problemas del estilo libre de competencia.

El administrador del sistema y los administradores de contenidos podrán gestionar (Actualizar, Eliminar, Insertar) el conjunto de problemas correspondientes al estilo libre de competencia, proporcionando la información pertinente en cada caso y que se describe seguidamente:

RF5.1. Actualizar un problema para el estilo libre: el administrador del sistema y los administradores de contenidos, una vez listados el conjunto de problemas, podrán actualizar el título, descripción, la entrada de datos como ejemplo del problema, la salida de datos como respuesta del problema, el tiempo de memoria máxima que debe ocupar la ejecución de la solución propuesta, el tamaño límite del código fuente que debe tener la solución propuesta y la complejidad del problema.

RF5.2. Insertar un problema para el estilo libre: el administrador del sistema y los administradores de contenidos podrán insertar nuevos problemas correspondientes al estilo libre proporcionando: el título del problema, descripción del problema, la entrada de datos como ejemplo del problema, la salida de datos como respuesta del problema, el tiempo de memoria máxima que debe ocupar la ejecución del problema, el tamaño límite del código fuente que debe tener la solución que propongan los usuarios registrados en el sistema y la complejidad del problema.

RF5.3. Eliminar un problema para el estilo libre: el administrador del sistema y los administradores de contenidos podrán eliminar problemas pertenecientes al estilo libre de competencia seleccionando los problemas, previa organización a partir de uno de los atributos.

RF6. Gestionar pruebas en línea del estilo de competencia.

El administrador del sistema y los administradores de contenidos podrán gestionar (Actualizar, Eliminar, Insertar) el conjunto de asignaturas, temas y preguntas correspondientes al estilo de competencia "pruebas en línea", proporcionando la información pertinente en cada caso y que se describe seguidamente:

RF6.1. Actualizar asignaturas, temas y preguntas para el estilo de competencia pruebas en línea: el administrador del sistema y los administradores de contenidos podrán actualizar asignaturas, temas y preguntas proporcionado los siguientes datos:

- *Asignatura:* nombre de la asignatura.
- *Tema:* nombre del tema y asignatura a la que pertenece el tema.
- *Pregunta:* tipo de la pregunta (Verdadera o falsa, selección simple, selección múltiple y completamiento), nombre de la pregunta, cantidad de elementos, etiquetas y respuestas. Estos últimos deben estar en correspondencia con el atributo cantidad de elementos.

RF6.2. Insertar asignaturas, temas y preguntas para el estilo de competencia pruebas en línea: el administrador del sistema y los administradores de contenidos podrán insertar asignaturas, temas y preguntas proporcionado los siguientes datos:

- *Asignatura*: nombre de la asignatura.
- *Tema*: nombre del tema y asignatura a la que pertenece el tema.
- *Pregunta*: tipo de la pregunta (Verdadera o falsa, selección simple, selección múltiple y completamiento), nombre de la pregunta, cantidad de elementos, etiquetas y respuestas. Estos últimos deben estar en correspondencia con el atributo cantidad de elementos.

RF6.3. Eliminar asignaturas, temas y preguntas para el estilo de competencia pruebas en línea: el administrador del sistema y los administradores de contenidos podrán eliminar asignaturas, temas y preguntas previamente listadas. Al eliminar una asignatura se eliminarán todos los temas y preguntas subordinadas a esa asignatura, al eliminar un tema se eliminarán todas las preguntas subordinadas a ese tema.

RF7. Gestionar el estilo de competencia test.

El administrador del sistema y los administradores de contenidos podrán gestionar (Actualizar, Eliminar, Insertar) el conjunto preguntas correspondientes al estilo de competencia "test", proporcionando la información pertinente en cada caso y que se describe seguidamente:

RF7.1. Actualizar preguntas para el estilo de competencia test: el administrador del sistema y los administradores de contenidos podrán actualizar las preguntas correspondientes al estilo de competencia test proporcionado los siguientes datos: nombre de la pregunta, cantidad de elementos, etiquetas y respuestas.

RF7.2. Insertar preguntas para el estilo de competencia test: el administrador del sistema y los administradores de contenidos podrán insertar las preguntas correspondientes al estilo de competencia test proporcionando los siguientes datos: nombre de la pregunta, cantidad de elementos, etiquetas y respuestas.

RF7.3. Eliminar preguntas el estilo de competencia test: el administrador del sistema y los administradores de contenidos podrán eliminar preguntas correspondientes al estilo de competencia test previamente listadas.

RF8. Gestionar estilo de competencia retos.

El administrador del sistema y los administradores de contenidos podrán gestionar (Actualizar y Eliminar) el estilo de competencia “retos”, proporcionando la información pertinente en cada caso y que se describe seguidamente:

RF8.1. Actualizar datos para el estilo retos: el administrador del sistema y los administradores de contenidos podrán actualizar los usuarios del retador, del retado y el identificador del problema seleccionado para el reto.

RF8.2. Eliminar retos: administrador del sistema y los administradores de contenidos podrán eliminar retos del sistema previamente listados, seleccionando el nombre del retador.

RF9. Gestionar estilo de competencia “competencia personalizada”.

El administrador del sistema y los administradores de contenidos podrán gestionar (Actualizar, Eliminar, Insertar) los datos correspondientes a estilo de competencia “competencia personalizada”, proporcionando la información pertinente en cada caso y que se describe seguidamente:

RF9.1. Actualizar datos para el estilo competencia personalizada: el administrador del sistema y los administradores de contenidos podrán actualizar datos (nombre de la competencia, cantidad de problemas, contraseña, fecha y hora de inicio, fecha y hora de culminación, si será pública o no, los equipos y sus integrantes y los problemas seleccionados para la competencia) de competencias personalizadas por los usuarios registrados.

RF9.2. Insertar competencia personalizada: el administrador del sistema y los administradores de contenidos podrán insertar nuevas competencias personalizadas proporcionando los siguientes datos: nombre de la competencia, cantidad de problemas, contraseña, fecha y hora de inicio, fecha y hora de culminación, si será pública o no, los equipos, sus nombres y sus integrantes y los problemas seleccionados para la competencia de competencias personalizadas por los usuarios registrados.

RF9.3. Eliminar competencia personalizada: el administrador del sistema y los administradores de contenidos podrán eliminar competencias personalizadas seleccionando los nombres previamente listados.

RF10. Gestionar el estilo de competencia concurso centralizado.

El administrador del sistema y los administradores de contenidos podrán gestionar (Actualizar, Eliminar, Insertar) los datos correspondientes a estilo de competencia “concurso centralizado”, proporcionando la información pertinente en cada caso y que se describe seguidamente:

RF10.1. Actualizar datos para el estilo de competencia concurso centralizado: el administrador del sistema y los administradores de contenidos podrán actualizar los datos (nombre del concurso, cantidad de problemas, contraseña, fecha y hora de inicio, fecha y hora de culminación, si será pública o no, los equipos y los problemas seleccionados) de algún concurso previamente creado.

RF10.2. Insertar concurso: el administrador del sistema y los administradores de contenidos podrán insertar nuevas convocatorias a concursos proporcionando los siguientes datos: nombre del concurso, cantidad de problemas, contraseña, fecha y hora de inicio, fecha y hora de culminación, si será pública o no, los equipos y los problemas seleccionados.

RF10.3. Eliminar concursos: el administrador del sistema y los administradores de contenidos podrán eliminar concursos centralizados previamente creados seleccionando los nombres previamente listados.

RF11. Gestionar la publicación de noticias.

El administrador del sistema podrá gestionar (Actualizar, Eliminar, Insertar) la publicación de noticias, proporcionando la información pertinente en cada caso y que se describe seguidamente:

RF11.1. Actualizar noticias: el administrador del sistema podrá actualizar una noticia previamente publicada proporcionando los datos pertinentes: título, alias, sección, si será publicada o no, la página principal, la categoría, formatear el texto, el contenido de la noticia, la fecha de creación, el rango de fecha que estará publicada, si caduca o no la noticia y el autor del artículo.

RF11.2. Insertar noticias: el administrador del sistema podrá insertar una nueva noticia proporcionando los siguientes datos: título, alias, sección, si será publicada o no, la página principal, la categoría, formatear el texto, el contenido de la noticia, la fecha de creación, el rango de fecha que estará publicada, si caduca o no la noticia y el autor del artículo.

RF11.3. Eliminar noticias: el administrador del sistema podrá eliminar noticias previamente creadas seleccionándolas una vez que han sido listadas.

RF12. Gestionar la publicación de cursos.

El administrador del sistema podrá gestionar (Actualizar, Eliminar, Insertar) la publicación de cursos, proporcionando la información pertinente en cada caso y que se describe seguidamente:

RF12.1. Actualizar cursos: el administrador podrá actualizar la publicación de cursos o datos relativos a estos.

RF12.2. Insertar curso: el administrador podrá publicar nuevos cursos subiéndolos como un archivo compactado o como un enlace a la dirección real donde está publicado el curso.

RF12.3. Eliminar curso: el administrador podrá eliminar cursos previamente publicados para lo cual requerirá del nombre del curso y él mismo podrá eliminar el archivo compactado o eliminar el enlace a la referencia real del curso.

RF13. Gestionar la publicación de recursos.

El administrador del sistema podrá gestionar (Actualizar, Eliminar, Insertar) la publicación de recursos, proporcionando la información pertinente en cada caso y que se describe seguidamente:

RF13.1. Actualizar recursos: el administrador podrá actualizar la publicación de recursos o datos relativos a estos. Para lo cual podrá proporcionar los datos de: carpetas, archivos, grupos, nivel de acceso.

RF13.2. Insertar recurso: el administrador podrá publicar nuevos recursos. Para lo cual podrá proporcionar los datos de: carpeta, archivo, grupo, nivel de acceso, repositorio (local o remoto)

RF13.3. Eliminar recurso: el administrador podrá eliminar un recurso previamente publicado seleccionando los nombres de los recursos listados.

RF14. Gestionar la publicación de enlaces de interés.

RF14.1. Actualizar enlaces de interés: el administrador podrá actualizar enlaces previamente publicados para lo cual requerirá de la dirección URL.

RF14.2. Insertar enlaces de interés: el administrador podrá publicar nuevos enlaces de interés para lo cual proporcionará la dirección URL.

RF14.3. Eliminar enlaces de interés: el administrador podrá eliminar enlaces de interés para lo cual requerirá del URL.

RF15. Gestionar la conformación de equipos.

El administrador del sistema, el administrador de equipos y los usuarios registrados en las competencias personalizadas podrán gestionar la conformación de equipos, para lo cual necesitarán los siguientes datos:

RF15.1. Actualizar equipos: el administrador del sistema, el administrador de equipos y los usuarios registrados en las competencias personalizadas podrán actualizar equipos previamente creados, para lo cual requerirán del nombre del equipo y los integrantes. El administrador del sistema y el administrador de equipos podrán hacerlo para los concursos centralizados y el usuario registrado para las competencias personalizadas que haya creado previamente.

RF15.2. Insertar equipos: el administrador del sistema, el administrador de equipos y los usuarios registrados en las competencias personalizadas podrán insertar nuevos equipos, para lo cual requerirán del nombre del equipo y los integrantes. El administrador del sistema y el administrador de equipos podrán hacerlo para los concursos centralizados y el usuario registrado para las competencias personalizadas que deseen crear.

RF15.3. Eliminar equipos: el administrador del sistema, el administrador de equipos y los usuarios registrados en las competencias personalizadas podrán eliminar equipos

previamente creados, para lo cual requerirán del nombre del equipo. El administrador del sistema y el administrador de equipos podrán hacerlo para los concursos centralizados y el usuario registrado para las competencias personalizadas que haya creado previamente.

RF16. Mostrar descripción de problemas del estilo libre de competencia.

Los usuarios registrados, una vez que se hayan autenticado en el sistema, podrán ver la descripción de los problemas pertenecientes al estilo libre de competencias para su posterior solución.

RF17. Desarrollar prueba en-línea.

Los usuarios registrados, una vez que se hayan autenticado en el sistema, podrán definir un criterio, generar, resolver, salvar, reanudar o abandonar una prueba en línea. Para lo cual es necesaria la siguiente información:

RF17.1. Definir un criterio de selección: los usuarios registrados, una vez que se hayan autenticado en el sistema, podrán definir un criterio para la generación de una prueba en línea seleccionando: años, asignaturas y temas. El sistema generará una prueba en línea basada en el criterio definido por el estudiante.

RF17.2. Generar prueba en línea: los usuarios registrados, una vez que se hayan autenticado en el sistema, podrán generar una prueba en línea definiendo un criterio previamente.

RF17.3. Resolver prueba en línea generada: los usuarios registrados, una vez que se hayan autenticado en el sistema, podrán resolver las pruebas generadas pregunta a pregunta, dado un criterio previo, sin la posibilidad de ver las anteriores.

RF17.4. Salvar la prueba en línea generada: los usuarios registrados, una vez que se hayan autenticado en el sistema y se encuentren resolviendo una prueba generada podrán salvar la pregunta en la que se encuentran y reanudarla en otro instante.

RF17.5. Reanudar una prueba en línea salvada: los usuarios registrados, una vez que se hayan autenticado en el sistema, podrán reanudar una prueba en línea que haya salvado previamente. El sistema le mostrará la prueba salvada a partir de la próxima pregunta en la que se quedó momento antes de salvar la prueba.

RF17.6. Abandonar una prueba en línea: los usuarios registrados, una vez que se hayan autenticado en el sistema, que hayan generado una prueba dado un criterio previamente definido podrán abandonar la realización de la misma.

RF18. Resolver test.

Los usuarios registrados, una vez que se hayan autenticado en el sistema, podrán resolver test psicométricos para lo cual deberán seleccionar uno del listado de test disponibles. El sistema mostrará el test pregunta a pregunta de un total de 5.

RF19. Crear reto.

Los usuarios registrados, una vez que se hayan autenticado en el sistema, podrán retar a otros usuarios ya autenticados a resolver un problema. Para lo cual deberán seleccionar el usuario a

retar del listado de usuarios en línea, enviar una invitación al reto para lo cual debe definir hora de inicio y fin. El sistema guardará el reto como una actividad pendiente.

RF20. Aceptar reto.

Los usuarios registrados, una vez que se hayan autenticado en el sistema, podrán aceptar o rechazar un reto que le haya enviado otro usuario registrado. En caso de aceptarlo el sistema automáticamente lo llevara a la página del reto.

RF21. Crear competencia personalizada.

Los usuarios registrados, una vez que se hayan autenticado en el sistema, podrán crear sus propias competencias personalizadas proporcionando los siguientes datos: nombre de la competencia, contraseña de acceso para los participantes, fecha y hora de inicio y fin, conformar los equipos seleccionando los integrantes y los problemas a resolver. El sistema permitirá filtrar los usuarios por universidad, facultad y año. Una vez conformado los equipos el sistema enviará una notificación a los participantes.

RF22. Actualizar datos competencia personalizada.

Los usuarios registrados, una vez que se hayan autenticado en el sistema, podrán modificar los datos de las competencias personalizadas previamente creadas. Proporcionando los siguientes datos: nombre de la competencia, contraseña de acceso para los participantes, fecha y hora de inicio y fin, conformar los equipos seleccionando los integrantes. El sistema permitirá filtrar los usuarios por universidad, facultad y año. Una vez conformado los equipos el sistema enviará una notificación a los participantes.

RF23. Participar competencia personalizada.

Los usuarios registrados, una vez que se hayan autenticado en el sistema, podrán participar en las competencias personalizadas en las cuales hayan sido invitados, para lo cual deberán proporcionar la contraseña que le fue enviada junto con la invitación a la participación en la competencia.

RF24. Enviar Solución.

Los usuarios registrados, una vez que se hayan autenticado en el sistema, podrán enviar una solución a un problema de los estilos de competencia: estilo libre, concurso centralizado, competencia personalizada, reto. Los datos requeridos son: código fuente del programa, identificador del problema resuelto y el lenguaje en el cuál fue desarrollado. El sistema enviará la solución propuesta al juez en línea encargado comprobar la solución.

RF25. Participar concurso centralizado.

Los usuarios registrados, una vez que se hayan autenticado en el sistema, podrán participar en los concursos centralizados a través de un usuario y contraseña de equipo proporcionada por el administrador del equipo.

RF26. Definir equipos para concurso centralizado

El administrador de equipos definirá los equipos según las restricciones de los organizadores. Deberá definir el nombre del equipo y sus integrantes.

RF27. Visualizar problemas de los concursos centralizados.

Los usuarios registrados, una vez que se hayan autenticado en el sistema con el usuario de equipo y la contraseña de acceso al concurso centralizado que le hayan enviado junto con la invitación de participación al concurso centralizado, podrán visualizar los problemas propuestos para el concurso centralizado en cuestión.

RF28. Contactar Juez.

Los usuarios registrados, una vez que se hayan autenticado en el sistema con el usuario de equipo y la contraseña de acceso al concurso centralizado que le hayan enviado junto con la invitación de participación al concurso centralizado, podrán contactar a los jueces (organizadores del concurso) para aclarar cualquier duda.

RF28. Mostrar Información del Ranking y Ayuda

El sistema permitirá mostrar información del ranking general del sistema y la ayuda de cómo usar las funcionalidades del mismo a todos los usuarios.

2.1.2. Requisitos no funcionales.

RNF1 Usabilidad.

La interfaz del sistema debe ser amigable e intuitiva para todos los usuarios, independientemente de su cultura informática, cumpliendo con los requisitos siguientes:

- La entrada de datos es precisa y bien estructurada permitiendo la interpretación correcta e inequívoca de la información.
- El diseño de la interfaz de usuario está orientada a la ejecución de acciones de una manera rápida, minimizando los pasos a dar en cada proceso.
- Se define una pauta de diseño visual para el uso de los íconos con el objetivo de distinguir los elementos de la ventana a través del uso de estos.
- La validación de los datos de entrada se realiza inmediatamente después de introducir los datos, señalando el conjunto de datos incorrectos.
- De los datos de entrada se validarán: su consistencia, corrección y exactitud.
- Fácil identificación de campos obligatorios en cada ventana.
- Todos los textos y mensajes en pantalla aparecen en idioma Español.
- Al usuario para operar el sistema requerirá información mínima y será intuitivo.
- El sistema presenta una ayuda que permite informar y documentar a los usuarios sobre cómo operar el sistema.

RNF2 Rendimiento.

El sistema estará diseñado sobre una arquitectura cliente/servidor, cumpliendo con los requisitos siguientes:

- Registrar varios servidores para la distribución de la carga en tiempo de ejecución.
- Alta capacidad de procesamiento para ejecutar algoritmos complejos.
- El tiempo de procesamiento de una solución de un problema debe estar ajustado al tiempo de CPU definido para dicho problema.

RNF3 Disponibilidad

El sistema deberá estar disponible las 24 horas del día.

RNF4 Soporte.

El Soporte técnico comprenderá dos niveles fundamentales según su complejidad:

- Primer nivel: soporte remoto a nivel de escritorio ayuda.
- Segundo nivel: soporte presencial.

RNF5 Portabilidad.

El sistema será portable y multiplataforma, cumpliendo con los siguientes requisitos:

- Hará uso de tecnologías rápidas, livianas y multiplataforma.
- No requerirá de acciones adicionales para la migración de una plataforma a otra.
- No necesitará de la instalación de herramientas adicionales en el cliente, a parte de un navegador web.

RNF6 Seguridad.

El sistema requerirá autenticación para hacer uso de sus funcionalidades principales y cumplirá con los siguientes requisitos:

- Autenticación local.
- Posibilidad de incluir autenticación ldap o mediante clientes de correos externos cuando sea necesario.
- Protección a la inyección de código SQL.
- Log de las acciones realizas mientras el usuario está autenticado.
- Un historial para la información que ha caducado.
- Vías alternativas para la recuperación de la contraseña por parte de los usuarios.

RNF7 Legales.

El sistema será desarrollado bajo el uso de herramientas y tecnologías no propietarias.

RNF8 Escalabilidad

El sistema debe garantizar mecanismo de extensiones motivada por futuros requisitos sin afectar el estado previo, además cumplirá con los siguientes:

- Esquema sencillo de extensiones para todo el sistema, de fácil gestión y administración.
- Alta cohesión y bajo acoplamiento entre los diferentes componentes del sistema.

RNF9 Hardware

El sistema debe ser capaz de correr con prestaciones mínimas para el servidor:

- Pentium II de 256 MB de RAM para el cliente y preferiblemente 512 MB de RAM a 1GHz para el servidor.

2.2. ARQUITECTURA DEL SISTEMA.

Para describir y representar la arquitectura de los elementos arquitectónicamente significativos se hará uso del modelo 4+1 vistas propuesto por *RUP*, dado que el mismo facilita la comprensión de la arquitectura propuesta y aportan argumentos para la identificación de los elementos claves

del sistema en desarrollo, proporcionando información sobre las decisiones arquitectónicas y sobre los elementos arquitectónicamente significativos.

La metodología *RUP* en su modelo 4+1 propone las siguientes vistas:

- *La Vista de guión de uso:* contiene guiones de uso y casos de ejemplos que abarcan riesgos técnicos, de clases o de comportamiento arquitectónicamente significativos. Esta vista puede ser especificada mediante el paquete de nivel superior, paquetes de guiones de uso o actores y casos de usos.
- *La Vista lógica:* contiene las clases de diseño más importantes y su organización en paquetes y subsistemas. Es un subconjunto del Modelo de diseño.
- *La Vista de implementación:* contiene una visión general del Modelo de implementación y su organización en términos de módulos en paquetes y capas. Es un subconjunto del Modelo de implementación.
- *La Vista de proceso:* contiene la descripción de las tareas implicadas, sus interacciones y configuraciones. Es un subconjunto del Modelo de diseño.
- *La Vista de despliegue:* contiene la descripción de los diferentes nodos físicos para la mayoría de las configuraciones típicas de la plataforma y la asignación de tareas. Es un subconjunto del Modelo de despliegue.

La metodología de desarrollo *RUP* propone que algunas de las vistas del modelo 4+1 pueden ser omitidas, esto dependerá de la complejidad del sistema. Agrega además que aunque las vistas del modelo 4+1 pueden representar el diseño completo de un sistema, la arquitectura trata únicamente algunos aspectos específicos como: la estructura del modelo, los elementos esenciales y casos de ejemplos claves que muestran los flujos de control principal en el sistema y los servicios (Rational Unified Process, 2006), (Pressman, 2005).

2.2.1. Representación de la arquitectura.

A partir de lo mencionado en el *epígrafe 2.2* para la representación de la arquitectura se seleccionaron del modelo 4+1: la vista lógica, despliegue, patrones y capas; por considerarse que estos son los que más información aportan en la comprensión de la arquitectura de la solución propuesta.

Teniendo en cuenta la variedad de definiciones y propuestas existentes sobre los estilos arquitectónicos asumiremos la clasificación realizada por Reynoso (Reynoso, 2005), clasificando la propuesta de solución dentro de la clase de estilos arquitectónicos de “Llamada y retorno”, dado que el mismo enfatiza en modificabilidad, la escalabilidad y llamada a procedimientos remotos. Este estilo arquitectónico incluye: arquitectura basada en capas, arquitectura basada en componentes, arquitectura Modelo – Vista – Controlador y arquitectura orientada a objetos.

La decisión anterior se justifica por los siguientes elementos:

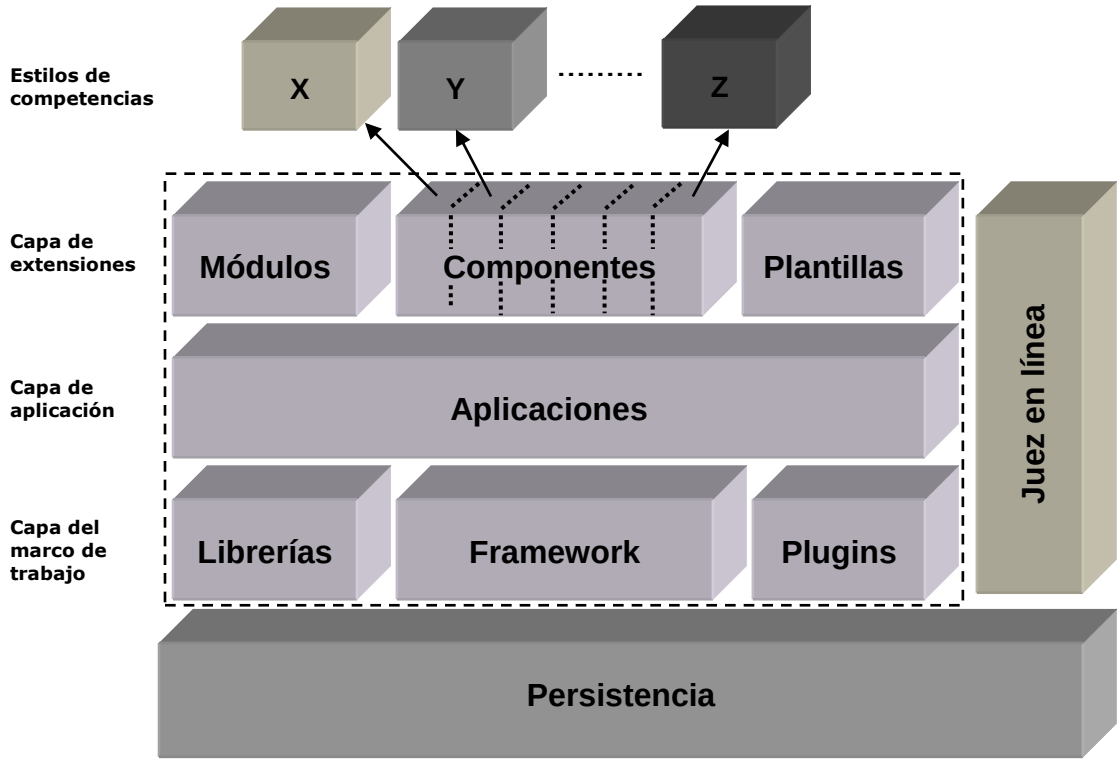
- La estructuración del sistema en capas viene dada por las ventajas que propicia como son: la flexibilidad, escalabilidad, reutilización y capacidad de mantenimiento.
- La inclusión de la arquitectura basada en componentes por facilitar la reutilización del software, simplificar las pruebas y facilitar el mantenimiento del sistema.
- El estilo Modelo – Vista – Controlador, por permitir separar de manera clara la lógica de negocio (modelo) de la vista, potenciando la reusabilidad del modelo, de modo que la

misma implementación de la lógica de negocio que opera un cliente pueda ser usado por otros, además por posibilitar una división de roles.

- La orientación a objetos por facilitar una adecuada organización jerárquica, distribución de responsabilidades y encapsulamiento.

En su nivel más abstracto la representación de la arquitectura del sistema se definió como:

Figura 2.2: *Representación de la arquitectura en capas de la aplicación informática "Coliseo Virtual".*



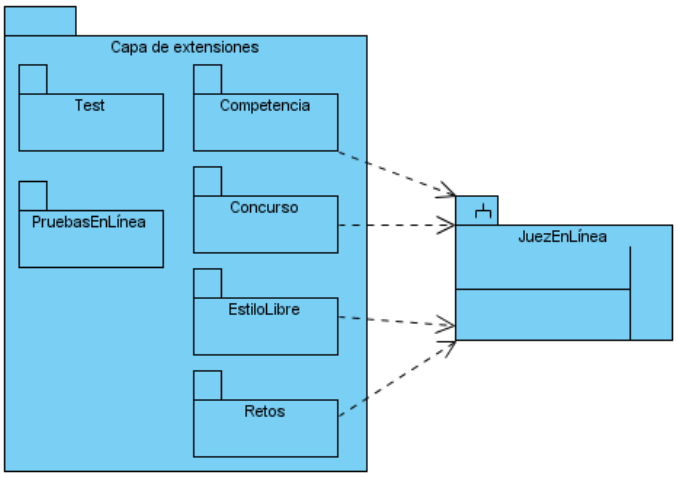
Fuente: *Elaboración propia.*

Las capas del marco de trabajo, aplicación y extensión describen la arquitectura del *CMS Joomla* y cada una es responsable de:

- *Capa del marco de trabajo:* está conformado por una colección de clases, librerías necesarias para el funcionamiento del marco de trabajo y el uso de otros desarrolladores y los plugins que son pequeños programas que extienden las funcionalidades del marco de trabajo.
- *Capa de aplicaciones:* amplían la funcionalidad del marco de trabajo, actualmente incluye cuatro aplicaciones: *JInstallation,* responsable de la instalación de *Joomla* en el servidor de aplicaciones web y debe ser eliminada después de completado el proceso de instalación; *JAdministrator,* responsable de la administración; *JSite,* responsable de mostrar la parte frontal del sistema; *XML-RPC,* la que hace posible la administración remota de las aplicaciones basadas en *Joomla*.
- *Capa de extensiones:* responsable de extender las funcionalidades del marco de trabajo y las aplicaciones, dando la posibilidad de personalizar el funcionamiento del *CMS*.
    - *Componentes:* son pequeños programas independientes entre sí, que están junto al núcleo de *Joomla* o bien instalados posteriormente desde el panel de

administración. Cada componente está estructurado siguiendo el patrón arquitectónico *MVC*.
    - *Módulos:* son elementos del sistema que muestran bloques de información en diferentes posiciones o zonas de la plantilla.
    - *Plantillas:* responsables de cambiar la forma en la que será vista la aplicación basada en *Joomla*. Existen dos tipos de plantillas, las de administración y las de la parte frontal.
- *Estilos de competencia:* esta subcapa lógica será desarrollada como componentes del *CMS* y dará respuesta a los requerimientos funcionales del sistema. Incluirá los estilos de competencia: estilo libre, retos, test, pruebas en línea, competencias y concursos.
- *Subsistema "Juez en línea":* es el responsable de calificar las propuestas de solución enviada por los usuarios finales a cada uno de los problemas. Se comunicará con la aplicación web mediante una conexión socket.

Figura 2.3: *Vista lógica. Subconjunto arquitectónicamente significativo.*



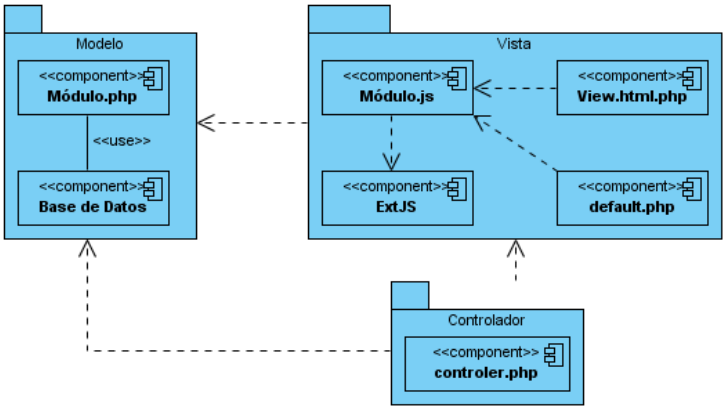
Para un mejor entendimiento y una mayor concreción de la vista lógica mostrada en la **Fig. 2.3** y considerando que ya se ha explicado el subsistema Juez en línea, seguidamente se describe el resto de los paquetes:

- *Estilo Libre:* encargado de gestionar el desarrollo de los problemas que intentan solucionar los usuarios registrados para este estilo de competencia.
- *Concurso*: se encuentra toda la información relacionada con la publicación y desarrollo de los concursos centralizados. Este paquete se auxiliará de un servicio de mensajería para contactar a los jueces durante el desarrollo de los concursos centralizados.
- *Competencia*: encargado de gestionar todo el proceso de creación y desarrollo de las competencias personalizadas por parte de los usuarios registrados en el sistema.
- *Retos:* encargado de gestionar la creación y desarrollo de los retos por parte de los usuarios registrados en el sistema.
- *Pruebas en líneas:* contiene un generador de pruebas configurable según el año, materia y temas que el usuario registrado defina. Gestiona todo el proceso de creación y desarrollo de las pruebas en línea.

- *Test:* responsable de gestionar el desarrollo y creación de los test que estarán disponibles en el sistema

Cada uno de los paquetes corresponde al desarrollo de un componente *Joomla*, los cuales están desarrollados siguiendo una arquitectura *MVC* como muestra la **Fig. 2.4.**

Figura 2.4: *Vista lógica de la arquitectura MVC de los componentes Joomla.*

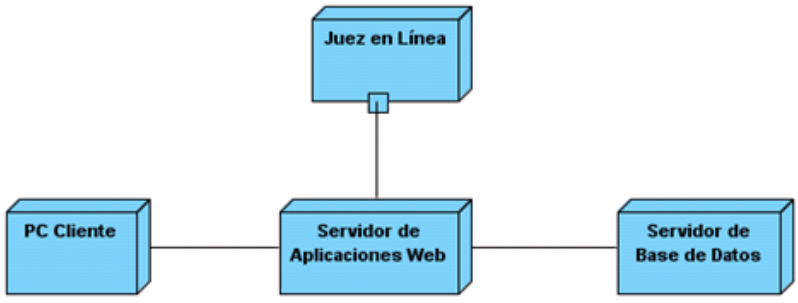


La arquitectura *MVC* es materializada mediante la implementación del patrón arquitectónico que lleva el mismo nombre, este patrón separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos (Reynoso, 2005).

En el caso específico de *Joomla* el uso del patrón MVC no requiere de un fichero de configuración, *Joomla* proporciona el mapeo las acciones del controlador a través del nombre que se le tiene que dar a cada directorio y fichero del componente, un error en los nombres y el CMS no podrá encontrar los ficheros requeridos implicando el mal funcionamiento del componente (LeBlanc, 2007). La **Fig. 2.4** muestra un ejemplo de algunos de los ficheros que deben ser creados en cada uno de los componentes *Joomla*.

Como parte del modelo 4+1 vista de *RUP* para representar la arquitectura se muestra en la **Fig. 2.5** el modelo de despliegue con el objetivo de ilustrar la arquitectura física del sistema para su ejecución, mediante procesadores, dispositivos y componentes de software se ha elaborado el modelo de despliegue mostrando la topología definida para la solución propuesta.

Figura 2.5: *Vista de despliegue para la representación de los elementos arquitectónicamente significativos.*



2.2.2. Componentes reutilizables.

Como parte de la fase de elaboración *RUP* propone dentro de las actividades esenciales, la selección de componentes, los mismos pueden ser creados, comprados o reutilizados (IBM).

Considerando que la solución propuesta es basada en software libre, la selección de componentes se basó en la reutilización, las mismas se describen a continuación:

- *Juez en línea:* constituye una solución de software que tiene como entrada un segmento de código y los parámetros de tiempo y memoria definidos para el desarrollo del mismo. Llevará a cabo el proceso de compilación y evaluación, además de emitir un resultado. Trabaja integrado al sistema en el proceso de calificación de las soluciones emitidas por el usuario. Este proceso será transparente para los usuarios.
- *Remository*: es un gestor de descargas. Permite subir y descargar archivos que estarán disponibles para los usuarios, crear categorías, dar permisos a usuarios, alojamiento local y remoto de ficheros, entre otras funcionalidades.
- *UddeIM*: es un componente que ofrece todas las características que necesita un sistema de mensajería privada. Es altamente configurable, permite definir el tamaño límite de los mensajes, recibir notificaciones, enviar mensajes para todos los usuarios, mostrar mensajes de alerta, listas de distribución restringida, mensaje *RSS*, mensaje cifrado, entre otras.
- *Autentication-Ldap*: plugin para *Joomla* que permite la autenticación mediante directorio activo.

El componente *Juez en Línea* fue integrado a la aplicación mediante conexión socket. Lo cual se define como un método para la comunicación entre un programa del cliente y un programa del servidor en una red mediante una dirección IP, un protocolo y un número de puerto. El mismo permitirá dar cumplimiento a los requisitos funcionales: **RF17**, **RF18**, **RF20**, **RF23**, **RF24**, **RF25**.

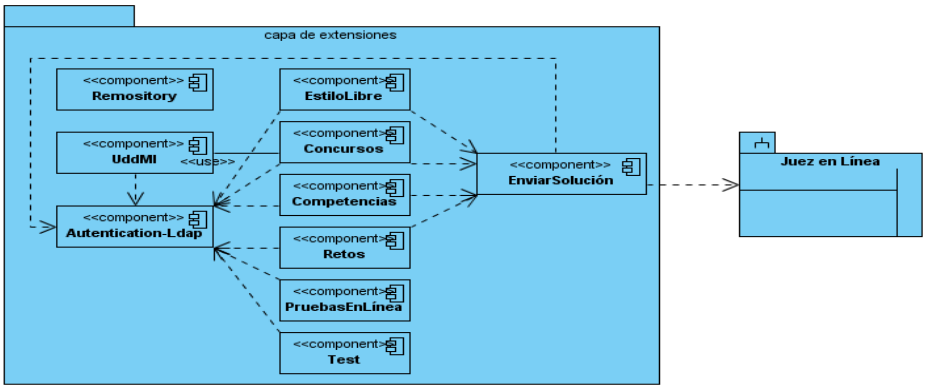
El componente *Remository* permitirá dar cumplimiento al requisito funcional **RF13**. Estará disponible en una interfaz al usuario accesible desde la opción de menú "*Sistema"*, disponible en la portada de la aplicación.

El componente *UddeIM*, posibilitará el envío de mensajes entre usuarios y administradores en ocasión de los concursos centralizados. Lo que permitirá dar cumplimiento al **RF27**.

El plugin *Autentication-Ldap* permitirá la autenticación mediante directorio activo posibilitando el cumplimiento al **RNF6**.

El diagrama mostrado en la **Fig. 2.6** muestra la interacción entre los componentes del sistema y los componentes antes mencionados:

Figura 2.6: *Vista del diagrama de componentes del sistema y los componentes reutilizables.*



2.3. MODELO DEL SISTEMA.

A continuación se listan los actores del sistema definidos a partir del diagrama del dominio presentado en la **Fig. 2.1** como es el caso del subsistema *Juez en línea* y de las personas asociadas al cumplimiento de requisitos funcionales definidos en el *epígrafe 2.1.1*:

Tabla 2.1: *Actores del sistema.*

Actores del sistema	Justificación
Administrador del sistema	El administrador del sistema es el encargado de gestionar cada cambio (insertar, eliminar, actualizar, etc.) realizado sobre el sistema en cualquiera de sus funcionalidades (Retos, Competencias, Estilos Libres, Pruebas en línea, Cursos, Usuarios, Noticias, Recursos, Enlaces).
Administrador de contenidos	Es el encargado de colaborar con el administrador en cuanto a la gestión de funcionalidades del sistema (Retos, Competencias, Estilo libres, Pruebas en línea, Test, Cursos, Noticias, Recursos, Enlaces)
Administrador de equipos	Es el encargado de gestionar cada cambio en los equipos que se registren para concursos centralizados. Puede realizar acciones de Insertar, Eliminar y Actualizar cada uno de ellos cuando sea necesario.
Usuario invitado	El usuario invitado solo tiene acceso a la información encontrada en la página principal.
Usuario registrado	El usuario registrado es el que puede acceder a cualquier funcionalidad en el sistema hasta donde lleguen sus privilegios, o sea, puede acceder a los distintos estilos de competencias o simplemente leer información y descargarla.
Usuario Equipo	El usuario equipo es considerado como un usuario registrado, es el que participa en los concursos centralizados y en las competencias personalizadas. Un usuario equipo representa a sus tres integrantes.
Juez de concurso	Permanece en contacto con los usuarios de manera online a través del chat o por correo.
Juez en línea	Sistema que realiza la calificación de las competencias, estilos libres, retos y concurso de los usuarios registrados y equipos.
Juez test	Sistema que realiza la calificación de los Test de los usuarios registrados.
Juez pruebas en línea	Sistema que realiza la calificación de las pruebas en línea de los usuarios registrados.

Seguidamente se comienza a describir las vistas de casos de usos del sistema desde una organización por actores:

Figura 2.7: *Vista de casos de uso del paquete Administrador de Equipos (a) y del paquete Usuario Invitado (b).*

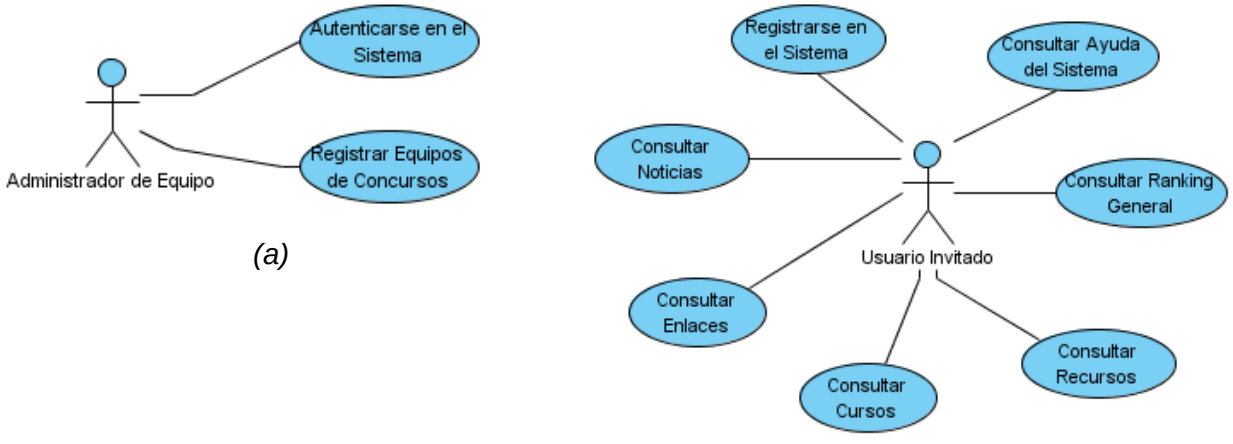


Figura 2.8: *Vista de casos de uso del paquete Usuario Registrado.*

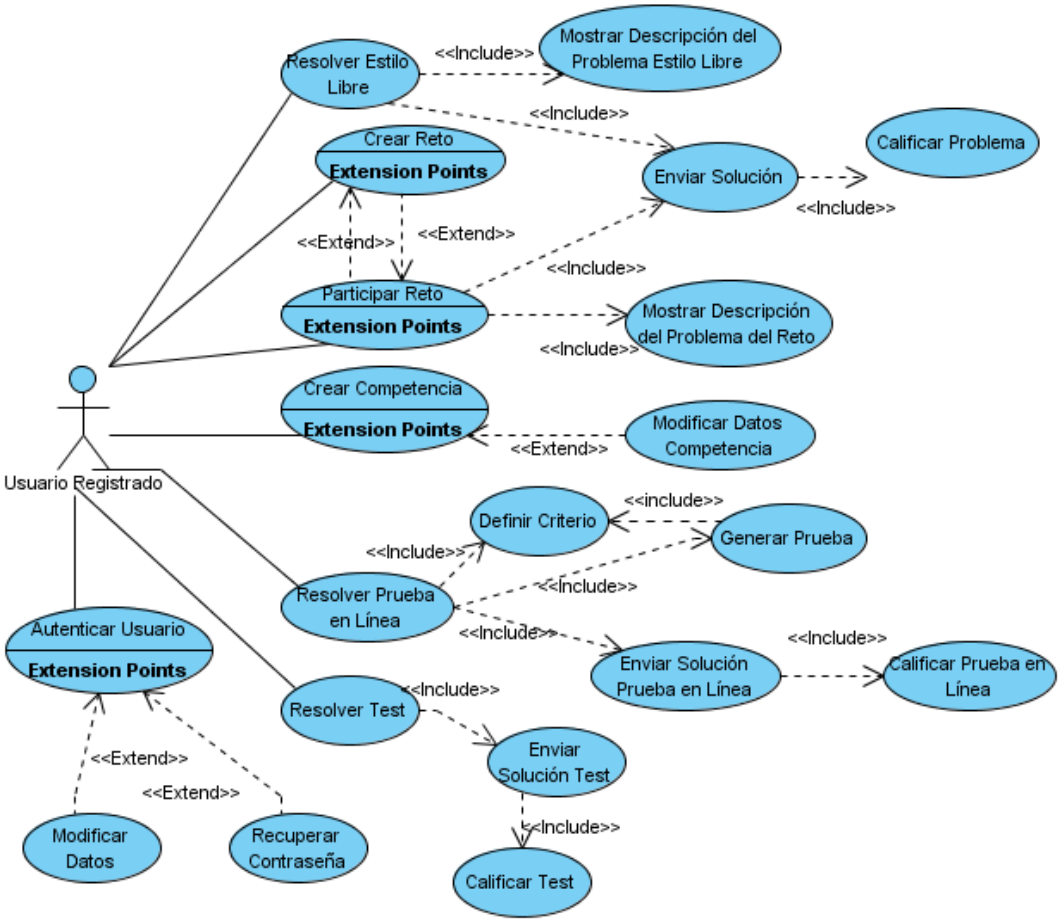


Figura 2.9: *Vista de casos de uso del paquete Usuario Equipo.*

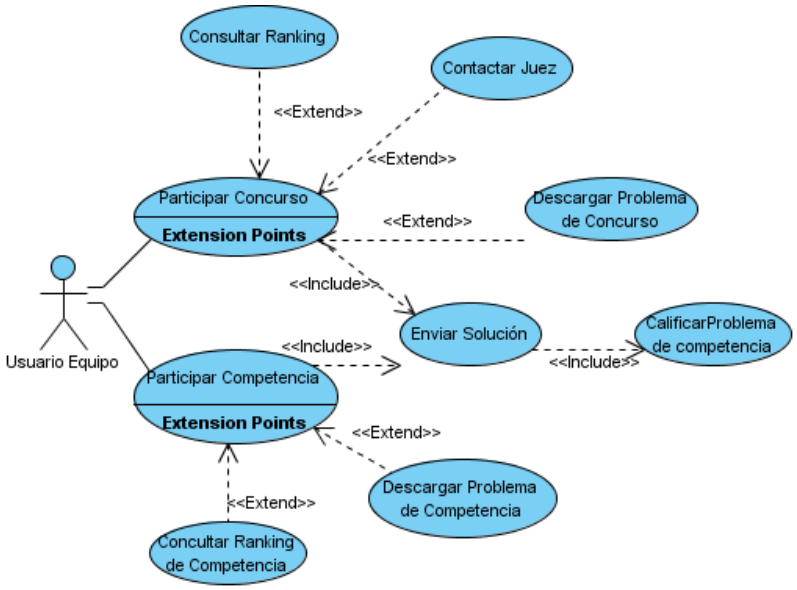
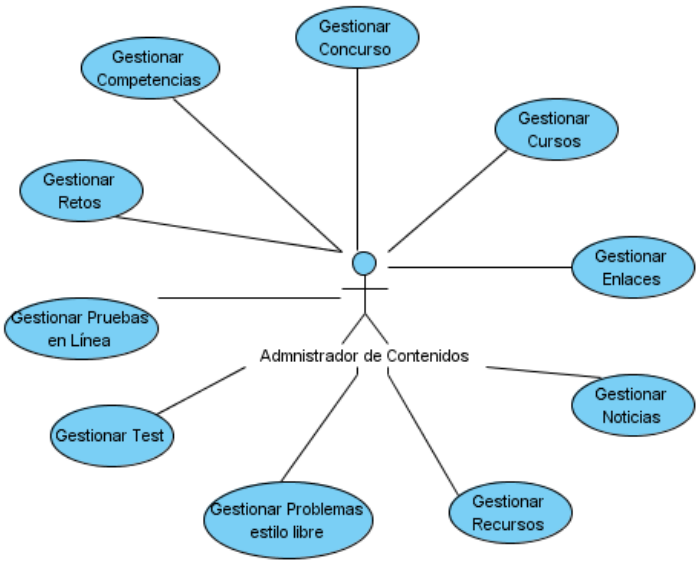


Figura 2.10: *Vista de casos de uso del paquete Administrador de Contenidos.*

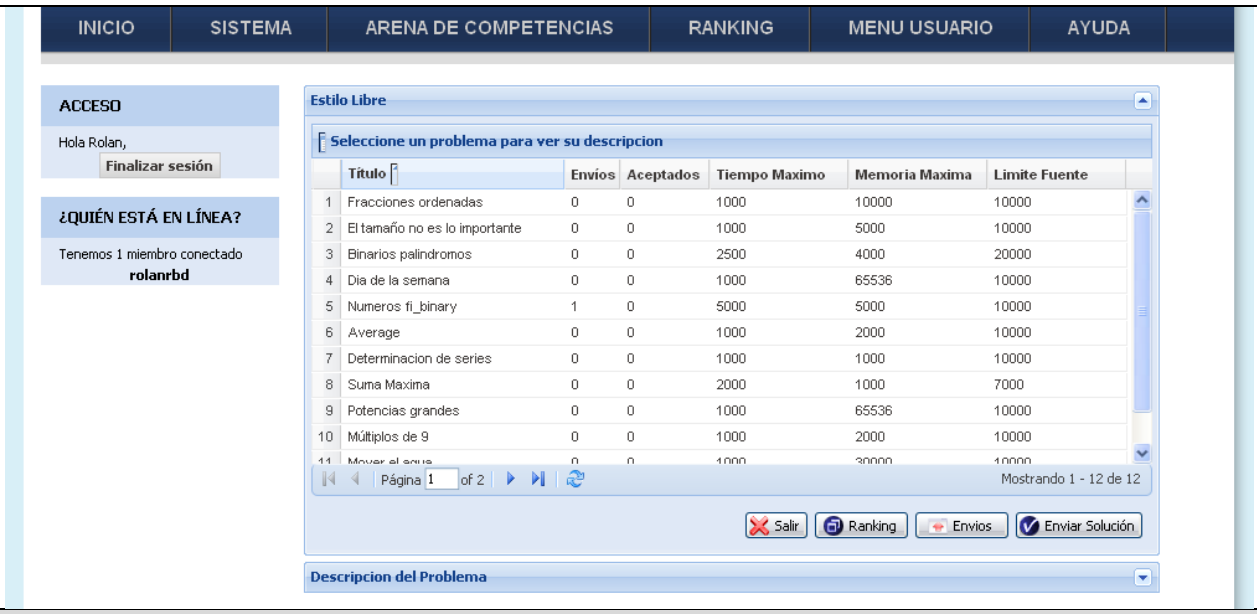


Seguidamente se procede a realizar la descripción textual de los casos de usos arquitectónicamente significativos.

Tabla 2.2: *Descripción textual del caso de uso Resolver Estilo Libre.*

Caso de Uso	
Nombre del CU	CU Resolver Estilo libre.
Actores	Usuario Registrado.
Propósito	Resolver problema seleccionado perteneciente al estilo libre de competencia.
Resumen: el CU es iniciado por los usuarios registrados cuando estos acceden a través del menú principal "*Arena de Competencia*" a la opción "*Estilo Libre*", los mismos podrán ver la descripción de los problemas, el desempeño de los usuarios registrados en cada uno de los problemas, el *Ranking* de los usuarios en este estilo de competencia, enviar una solución y salir del sistema. El CU termina con el envío de una solución o haciendo clic en el botón "*Salir*".	
Referencias	***RF17, RF25***
Precondiciones	El usuario debe estar registrado.
Postcondiciones	El usuario participa en un estilo libre de competencia.
Flujo normal de eventos	
Acciones del actor	**Respuesta del sistema**
1. El CU inicia cuando el actor *usuario registrado* selecciona la opción "*Estilo Libre*" del menú principal del sistema "*Arena de Competencias*".	2. El sistema muestra la interfaz correspondiente al *"Estilo Libre"*. 2.1 La colección de problemas con los datos necesarios para su solución (*Tiempo máximo, Memoria máxima, Límite fuente*), los envíos realizados, los aceptados. 2.2 El botón "*Salir*". 2.3 El botón "*Ranking*". 2.4 El botón "*Envíos*". 2.5 El botón *"Enviar Solución"*.
3. Si el usuario selecciona: 3.1 El botón *"Salir"* (Ver **flujo alterno 1**, "Salir del sistema"). 3.2 El botón *"Ranking"* (Ver **flujo alterno 2**, *"Mostrar Ranking"*). 3.3 El botón *"Envíos"* (Ver **flujo alterno 3**, *"Mostrar datos de envíos"*). 3.4 El botón "*Enviar Solución"* (Ver f**lujo alterno 4**, *"Enviar solución"*). 3.5 El usuario selecciona el problema que desea resolver, continuar **flujo normal de eventos**.	4. El sistema muestra una interfaz con la descripción del problema seleccionado y el botón *"Subir".*

5. El usuario presiona el botón *"Subir".*	6. El sistema resalta el problema seleccionado por el usuario, continuar **flujo normal de eventos paso 2**.
Flujo alterno 1: "*Salir del sistema*".	
Acciones del actor	**Respuesta del sistema**
1. El usuario presiona el botón *"Salir".*	2. El sistema cancela la operación y cierra la interfaz. 3. El CU termina.
Flujo alterno 2: "*Mostrar Ranking*".	
Acciones del actor	**Respuesta del sistema**
1. El usuario presiona el botón *"Ranking".*	2. El sistema muestra una interfaz con los siguientes datos: usuario, año, puntos, envíos y aceptados. 2.1 El botón *"Cerrar".*
3. El usuario presiona el botón *"Cerrar".*	4. El sistema cierra la interfaz. 5. Continuar **flujo normal de eventos paso 2**.
Flujo alterno 3: "*Mostrar datos de envíos*".	
Acciones del actor	**Respuesta del sistema**
1. El usuario presiona el botón *"Envíos".*	2. El sistema muestra una interfaz con los siguientes datos: usuario, Id, Memoria, Tiempo, Fuente, Lenguaje y Respuesta. 2.1 El botón *"Cerrar".*
3. El usuario presiona el botón *"Cerrar".*	4. El sistema cierra la interfaz. 5. Continuar **flujo normal de eventos paso 2**.
Flujo alterno 4: "*Enviar solución*".	
Acciones del actor	**Respuesta del sistema**
1. El usuario presiona el botón *"Enviar Solución".*	2. El sistema verifica si hay un problema seleccionado: 2.1 Si el problema está seleccionado el sistema muestra una interfaz con los siguientes datos: Id del problema seleccionado, listado de lenguajes de programación permitidos y cuadro de texto editable. 2.1.1 El botón *"Salir".* 2.1.2 El botón *"Someter".* 2.2 Si el problema no está seleccionado muestra el cartel de alerta *"Debe seleccionar un problema".* 2.2.1 Continuar **flujo normal de eventos paso 2**.
3. El usuario puede: 3.1 Seleccionar el botón *"Salir"* (Ver **flujo alterno 5**, *"Salir de la interfaz: Enviar solución").* 3.2 Seleccionar el botón *"Someter"* (Ver **flujo alterno 6**, *"Someter al Juez").* 3.3 Insertar: Id del problema, seleccionar el lenguaje de programación e insertar el código fuente de su propuesta de solución al problema seleccionado. Continuar **flujo alterno 4 paso 3**.	
Flujo alterno 5: "S*alir de la interfaz: Enviar solución*".	
Acciones del actor	**Respuesta del sistema**
1. El usuario presiona el botón *"Salir".*	2. El sistema cancela la operación y cierra la interfaz. 3. Continuar **flujo normal de eventos paso 2**.
Flujo alterno 6: "*Someter al Juez*".	
Acciones del actor	**Respuesta del sistema**
1. El usuario presiona el botón *"Someter".*	2. El sistema verifica los campos (Id del problema seleccionado, listado de lenguajes de programación permitidos y cuadro de texto editable): 2.1 Si alguno está vacío, muestra el cartel: *"Debe llenar campos requeridos"*. Continuar **flujo alterno 4 paso 3**. 2.2 Si lo campos no están vacíos, se envía la propuesta de solución al *Juez en Línea*. 3. CU termina.
Prioridad	Crítico.
Prototipo de Interfaz	



Puntos de Inclusión
CU Mostrar descripción del problema Estilo Libre. CU Enviar solución.

Tabla 2.3: *Descripción textual del caso de uso Resolver Prueba en Línea*.

Caso de Uso	
Nombre del CU	CU Resolver Prueba en Línea.
Actores	Usuario Registrado.
Propósito	Resolver pruebas generadas aleatoriamente dependiendo del criterio de selección.
Resumen: el CU es iniciado por los usuarios registrados cuando estos acceden a través del menú principal "*Arena de Competencia*" a la opción "*Prueba en Línea*", los mismos podrán definir el criterio (año, materia y tema), generar una prueba basada en el criterio seleccionado, responder sus preguntas, abandonar la prueba, salvar la prueba, ver el *"Ranking"* de los usuarios en este estilo de competencia y salir del sistema. El CU termina cuando el usuario registrado selecciona la opción de "*Salir*".	
Referencias	***RF17.1, RF17.2, RF17.3, RF17.4, RF17.6***
Precondiciones	El usuario debe estar registrado.
Postcondiciones	El usuario ejercita sus conocimientos a través de las pruebas en línea.
Flujo normal de eventos	
Acciones del actor	**Respuesta del sistema**
1. El CU inicia cuando el actor *usuario registrado* selecciona la opción "*Prueba en Línea*" del menú principal del sistema "*Arena de Competencias*".	2. El sistema muestra la interfaz correspondiente a *"Pruebas en Línea"*: 2.1 Los años a seleccionar. 2.1.1 Las Materias. 2.1.2 Los Temas. 2.2 El botón *"Salir".* 2.3 El botón *"Ranking".* 2.4 El botón *"Generar Prueba".*
3. El usuario selecciona: 3.1 El botón *"Salir"* (Ver **flujo alterno 1**, *"Salir de Prueba en Línea"*). 3.2 El botón "Ranking" (Ver **flujo alterno 2**, *"Mostrar Ranking"*). 3.3 El botón *"Generar Prueba"* (Ver **flujo alterno 3**, *"Generar Prueba"*). 3.4 El usuario selecciona: años, materias y temas continuar **flujo normal de eventos**.	4. El sistema muestra los años, materias y temas seleccionados. Continuar **flujo normal de eventos paso 3**.
Flujo alterno 1: "*Salir de Prueba en Línea*".	

Acciones del actor	Respuesta del sistema
1. El usuario presiona el botón *"Salir".*	2. El sistema cancela la operación y cierra la interfaz. 3. El CU termina.
Flujo alterno 2: "*Mostrar Ranking*".	
Acciones del actor	**Respuesta del sistema**
1. El usuario presiona el botón *"Ranking".*	2. El sistema muestra una interfaz con los siguientes datos: usuario, año, puntos y cantidad de pruebas realizadas. 2.1 El botón *"Cerrar".*
3. El usuario presiona el botón *"Cerrar".*	4. El sistema cierra la interfaz. Continuar **flujo normal de eventos paso 2**.
Flujo alterno 3: "*Generar Prueba*".	
Acciones del actor	**Respuesta del sistema**
1. El usuario presiona el botón *"Generar Prueba".*	2. El sistema muestra una interfaz correspondiente a las pruebas generadas, proporcionando información necesaria para la realización de prueba y las siguientes opciones: 2.1 El botón *"Abandonar Examen".* 2.2 El botón *"Siguiente".* 2.3 El botón *"Guardar Prueba".*
3. Si el usuario selecciona: 3.1 El botón *"Abandonar Examen"* (Ver **flujo alterno 4**, *"Abandonar Examen").* 3.2 El botón *"Guardar Prueba"* (Ver **flujo alterno 5**, *"Guardar Prueba").* 3.3 El botón *"Siguiente",* continuar **flujo normal de eventos**.	4. El sistema muestra una interfaz correspondiente a la pregunta (selección simple, selección múltiple y completamiento) que ha sido generada aleatoriamente dependiendo del criterio de selección y el tiempo restante para solucionar la prueba. Además muestra las siguientes opciones: el botón *"Abandonar Examen"*, el botón *"Siguiente"* y el botón *"Guardar Prueba"*. Continuar **flujo alterno 3 paso 3**.
Flujo alterno 4: "*Abandonar Examen*".	
Acciones del actor	**Respuesta del sistema**
1. El usuario presiona el botón *"Abandonar Examen"*.	2. El sistema cierra la interfaz y continua el **flujo normal de eventos paso 2**.
Flujo alterno 5: "*Guardar Prueba*".	
Acciones del actor	**Respuesta del sistema**
1. El usuario presiona el botón *"Guardar Prueba".*	2. El sistema muestra una ventana con un cuadro de texto para entrar el nombre de la prueba a guardar y además: 2.1 El botón *"Cerrar".* 2.2 El botón *"Guardar".*
3. Si el usuario presiona: 3.1 El botón *"Cerrar"* (Ver **flujo alterno 6**, *"Cerrar ventana de guardar prueba"*). 3.2 El botón *"Guardar".* Continuar **flujo normal de eventos**.	4. El sistema guarda la prueba con el nombre insertado y muestra un mensaje informando que la prueba se ha guardado con éxito y el botón *"Ok".*
5. El usuario presiona el botón "Ok".	6. El sistema muestra una interfaz con el tiempo restante para realizar la prueba en pausa y además las siguientes opciones: 6.1 El botón *"Abandonar Examen".* 6.2 El botón *"Finalizar Prueba".*
7. Si el usuario selecciona: 7.1 El botón *"Abandonar Examen",* (Ver **flujo normal de eventos paso 2**). 7.2 El botón *"Finalizar Examen"*, continuar **flujo normal de eventos**.	8. El sistema cierra la interfaz y continua el **flujo normal de eventos paso 2**.
Flujo alterno 6: "*Cerrar ventana de guardar prueba*".	
Acciones del actor	**Respuesta del sistema**
1. El usuario presiona el botón *"Cerrar".*	2. El sistema muestra una interfaz correspondiente a la pregunta (selección simple, selección múltiple y completamiento) que ha sido generada aleatoriamente y dependiendo del criterio de selección. Además muestra las siguientes opciones: el botón *"Abandonar Examen"*, el botón *"Siguiente"* y el botón *"Guardar Prueba"*. Continuar **flujo alterno 3 paso 3**.
Prioridad	Crítico.

Prototipo de Interfaz

Puntos de Inclusión
CU Definir criterio. CU Generar prueba. CU Enviar solución Prueba en Línea.

Tabla 2.4: *Descripción textual del caso de uso Resolver Test.*

Caso de Uso	
Nombre del CU	Resolver Test.
Actores	Usuario Registrado.
Propósito	Resolver Test.
Resumen: el CU es iniciado por los usuarios registrados cuando estos acceden a través del menú principal "*Arena de Competencia*" a la opción "*Resolver Test*", los mismos podrán resolver cualquier test que esté activo, ver el *"Ranking"* de los usuarios en este estilo de competencia y salir del sistema. El CU termina cuando el usuario registrado selecciona la opción de "*Salir*".	
Referencias	***RF18.***
Precondiciones	El usuario debe estar registrado.
Postcondiciones	El usuario ejercita sus conocimientos a través de los Test.
Flujo normal de eventos	
Acciones del actor	**Respuesta del sistema**
1. El CU inicia cuando el actor Usuario Registrado selecciona la opción "*Test*" del menú principal del sistema "*Arena de Competencias*".	2. El sistema muestra la interfaz correspondiente a los *"Test".* 2.1 La colección de test con los datos necesarios para su solución (nombre, categoría, fecha de publicación, fecha de vencimiento). 2.2 El botón *"Salir".* 2.3 El botón *"Ranking".* 2.4 El botón *"Realizar Test".*
3. Si el usuario selecciona: 3.1 El botón *"Salir"* (Ver **flujo alterno 1**, *"Salir de la Interfaz"*). 3.2 El botón *"Ranking"* (Ver **flujo alterno 2**, *"Mostrar*	4. El sistema resalta el test seleccionado por el usuario, continuar **flujo normal de eventos paso 3**.

Ranking"). 3.3 El botón *"Realizar Test"* (Ver **flujo alterno 3**, *"Realizar Test"*). 3.4 El usuario selecciona el test que desea resolver, continuar **flujo normal de eventos**.	
Flujo alterno 1: "*Salir de la Interfaz*".	
Acciones del actor	**Respuesta del sistema**
1. El usuario presiona el botón *"Salir".*	2. El sistema cancela la operación y cierra la interfaz. 3. El CU termina.
Flujo alterno 2: "*Mostrar Ranking*".	
Acciones del actor	**Respuesta del sistema**
1. El usuario presiona el botón *"Ranking".*	2. El sistema muestra una interfaz con los siguientes datos: usuario, año y puntos. 2.1 El botón *"Cerrar".*
3. El usuario presiona el botón *"Cerrar".*	4. El sistema cierra la interfaz. 5. Continuar **flujo normal de eventos paso 3**.
Flujo alterno 3: "*Realizar Test*".	
Acciones del actor	**Respuesta del sistema**
1. El usuario presiona el botón "Realizar Test".	2. El sistema verifica si hay un test seleccionado: 2.1 El sistema muestra una interfaz correspondiente a la pregunta (selección simple, selección múltiple y completamiento) que ha sido generada aleatoriamente dependiendo del test seleccionado. Además muestra las siguientes opciones: 2.1.1 El botón *"Abandonar Test".* 2.1.2 El botón *"Siguiente".* 2.2 Si el test no está seleccionado muestra el cartel de alerta *"Debe seleccionar un Test".* 2.2.1 Continuar **flujo normal de eventos paso 3**.
3. Si el usuario selecciona: 3.1 El botón *"Abandonar Test"* (Ver **flujo alterno 4**, *"Abandonar Test").* 3.2 El botón *"Siguiente",* continuar **flujo normal de eventos**.	4. El sistema verifica si hay más preguntas en el test: 4.1 Si hay más preguntas el sistema muestra una interfaz correspondiente a la pregunta (selección simple, selección múltiple y completamiento) que ha sido generada aleatoriamente dependiendo del test seleccionado. Además muestra las siguientes opciones: 4.1.1 El botón *"Abandonar Test".* 4.1.2 El botón *"Siguiente",* continuar **flujo alterno 3, paso 3**. 4.2 Si no hay más preguntas muestra un *cartel "No hay más preguntas disponibles"* y el botón *"Finalizar Test".* Continuar **flujo normal de eventos**.
5. El usuario selecciona: 5.1 El botón *"Abandonar Test"* (Ver **flujo alterno 4**, *"Abandonar Test"*). 5.2 El botón *"Finalizar Test"*, continuar **flujo normal de eventos**.	6. El sistema muestra la puntuación y cierra la interfaz. 7. El CU termina.
Flujo alterno 4: "*Abandonar Test*".	
Acciones del actor	**Respuesta del sistema**
1. El usuario presiona el botón *"Abandonar Test".*	2. El sistema cierra la interfaz y continua el **flujo normal de eventos paso 2**.
Prioridad	Crítico.
Prototipo de Interfaz	

Puntos de Inclusión
CU Enviar solución Test.

Tabla 2.5: *Descripción textual del caso de uso Crear Reto*.

Caso de Uso	
Nombre del CU	Crear Reto.
Actores	Usuario Registrado.
Propósito	Retar a otro usuario registrado.
Resumen: el CU es iniciado por los usuarios registrados cuando estos acceden a través del menú principal "*Arena de Competencia*" a la opción "*Retos*", los mismos podrán: crear un *Reto*, ver el *Ranking* de los usuarios en este estilo de competencia, ver los retos que le han hecho y salir del sistema. El CU termina con el envío de una solución o haciendo clic en el botón "*Salir*".	
Referencias	***RF19, RF20***
Precondiciones	El usuario debe estar registrado.
Postcondiciones	El usuario crea su propio reto o participa en un reto.

Flujo normal de eventos	
Acciones del actor	**Respuesta del sistema**
1. El CU inicia cuando el actor U*suario Registrado* selecciona la opción "*Retos*" del menú principal del sistema "*Arena de Competencias*".	2. El sistema muestra la interfaz correspondiente a los "Retos". 2.1 Los usuarios registrados con su facultad y año. 2.2 El botón *"Salir".* 2.3 El botón *"Ranking".* 2.4 El botón *"Retar Usuario".* 2.5 La pestaña *"Invitaciones de Retos".* 2.5.1 Los datos del retador: usuario, id, año y facultad. 2.5.2 El botón *"Aceptar Reto".* 2.6 La pestaña *"Retos Pendientes".*
3. Si el usuario selecciona: 3.1 El botón *"Salir"* (Ver **flujo alterno 1**, *"Salir del sistema"*). 3.2 El botón *"Ranking"* (Ver **flujo alterno 2**, *"Mostrar Ranking"*). 3.3 De la pestaña *"Invitaciones de Retos"*, el botón *"Aceptar Reto"* (Ver **flujo alterno 3**, "*Aceptar Reto"*). 3.4 La pestaña *"Retos Pendientes"* (Ver **flujo alterno 4**, *"Retos Pendientes"*). 3.5 El botón *"Retar Usuario",* **continuar flujo normal de eventos**.	4. El sistema verifica si hay un usuario seleccionado: 4.1 Si el usuario a retar no está seleccionado, el sistema muestra un mensaje de alerta *("Debe seleccionar un usuario"*). 4.1.1 El botón *"OK".* 4.2 Si el usuario está seleccionado el sistema envía un mensaje informando que se le ha enviado una solicitud de reto al usuario seleccionado. 4.2.1 El botón *"OK".*

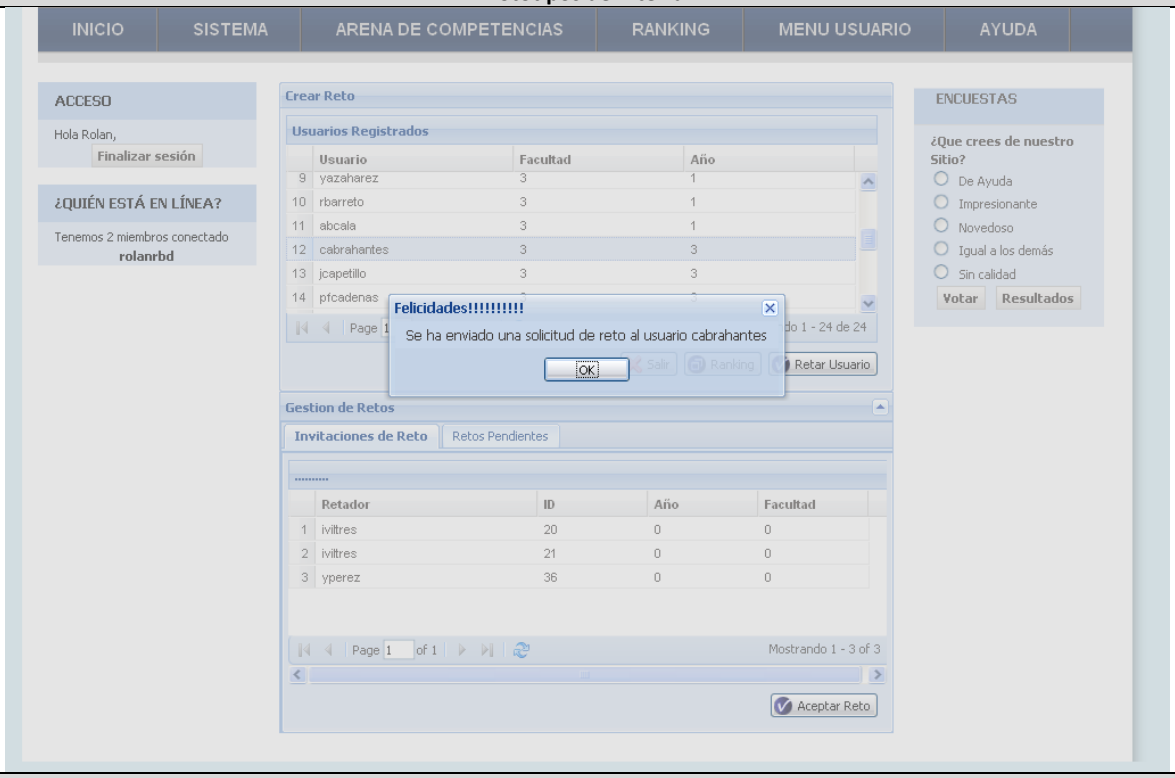
5. Si El usuario selecciona el botón *"OK",* continuar el **flujo normal de eventos paso 2**.	
Flujo alterno 1: "*Salir del sistema*".	
Acciones del actor	**Respuesta del sistema**
1. El usuario presiona el botón *"Salir".*	2. El sistema cancela la operación y cierra la interfaz. 3. El CU termina.
Flujo alterno 2: "*Mostrar Ranking*".	
Acciones del actor	**Respuesta del sistema**
1. El usuario presiona el botón *"Ranking".*	2. El sistema muestra una interfaz con los siguientes datos: usuario, año y puntos. 2.1 El botón *"Cerrar".*
3. El usuario presiona el botón *"Cerrar".*	4. El sistema cierra la interfaz. 5. Continuar **flujo normal de eventos paso 2**.
Flujo alterno 3: "*Aceptar Reto*".	
Acciones del actor	**Respuesta del sistema**
1. El usuario presiona el botón "Aceptar Reto".	2. El sistema verifica si existe alguna invitación a Reto seleccionado: 2.1 Si no hay alguna invitación a reto seleccionado muestra un cartel de alerta, *"Debe seleccionar una invitación"*, continuar **flujo normal de eventos paso 2**. 2.2 Si hay alguna invitación a reto seleccionado muestra cartel de alerta de confirmación del reto con el usuario seleccionado, continuar **flujo normal de eventos paso 2**.
Flujo alterno 4: "*Retos Pendientes*".	
Acciones del actor	**Respuesta del sistema**
1. El usuario presiona la pestaña, "Retos Pendientes".	2. El sistema muestra la siguiente interfaz correspondiente a los *"Retos".* 2.1 Los usuarios registrados con su facultad y año. 2.2 El botón *"Salir".* 2.3 El botón *"Ranking".* 2.4 El botón *"Retar Usuario".* 2.5 La pestaña *"Retos Pendientes".* 2.5.1 Los datos del reto: usuario retado, usuario retador, id del problema y el ganador en caso de haber concluido el reto. 2.5.2 El botón *"Enviar Solución".* 2.6 La pestaña *"Invitaciones de Reto".*
3. Si el usuario selecciona: 3.1 El botón *"Salir"* (Ver **flujo alterno 1**, *"Salir del sistema"*). 3.2 El botón *"Ranking"* (Ver **flujo alterno 2**, *"Mostrar Ranking"*). 3.3 El botón *"Retar Usuario".* 3.4 La pestaña *"Invitaciones a Reto"* (Ver **flujo alterno 3**, *"Aceptar Reto"*). 3.5 La pestaña *"Retos Pendientes"*. 3.6 El reto a realizar, **continuar flujo normal de eventos**. 3.7 El botón *"Enviar Solución"*, continuar (Ver **flujo alterno 5**, *"Enviar Solución"*).	4. El sistema muestra la descripción del problema asignado al reto. 4.1 El botón *"Cerrar".*
5. Si el usuario presiona el botón *"Cerrar",* continuar **flujo normal de eventos paso 2**.	
Flujo alterno 5: "*Enviar solución*".	
Acciones del actor	**Respuesta del sistema**
1. El usuario presiona el botón *"Enviar Solución".*	2. El sistema verifica si hay un problema seleccionado: 2.1 Si el problema está seleccionado el sistema muestra una interfaz con los siguientes datos: Id del problema seleccionado, listado de lenguajes de programación

	permitidos y cuadro de texto editable. 2.1.1 El botón *"Salir".* 2.1.2 El botón *"Someter".* 2.2 Si el problema no está seleccionado muestra el cartel de alerta *"Debe seleccionar un problema"*. 2.2.1 Continuar **flujo normal de eventos paso 2**.
3. El usuario puede: 3.1 Seleccionar el botón *"Salir",* continuar **flujo normal de eventos paso 2**. 3.2 Seleccionar el botón *"Someter"* (Ver **flujo alterno 6**, *"Someter al Juez"*). 3.3 Insertar: Id del problema, seleccionar el lenguaje de programación e insertar el código fuente de su propuesta de solución al problema seleccionado. Continuar **flujo alterno 5 paso 3**.	

Flujo alterno 6: "*Someter al Juez*".

Acciones del actor	Respuesta del sistema
1. El usuario presiona el botón *"Someter".*	2. El sistema verifica los campos (Id del problema seleccionado, listado de lenguajes de programación permitidos y cuadro de texto editable): 2.1 Si alguno está vacío, muestra el cartel: *"Debe llenar campos requeridos"*. Continuar **flujo alterno 5 paso 3**. 2.2 Si lo campos no están vacíos, se envía la propuesta de solución al *Juez en Línea.* 3. CU termina.
Prioridad	Crítico.

Prototipos de Interfaz



Puntos de Inclusión

CU Participar Reto.
CU Mostrar descripción del problema Estilo Libre.
CU Enviar solución.

2.4. MODELO DEL DISEÑO.

En el flujo de diseño del sistema se obtuvo el artefacto modelo de diseño, en el desarrollo de este flujo se tuvo presente que la solución propuesta es una aplicación web implementada en *PHP* y con el uso de la librería de *JavaScript ExtJS*, por lo que se hace necesario la utilización de los estereotipos y especificaciones pertinentes, como lo define la metodología.

Seguidamente se definen los diagramas de clases para los casos de usos arquitectónicamente significativos descritos en el *epígrafe 2.3* en las **Tablas 2.2 - 2.5**.

Figura 2.11: *Diagrama de clases del diseño correspondiente al caso de uso Resolver Estilo Libre.*

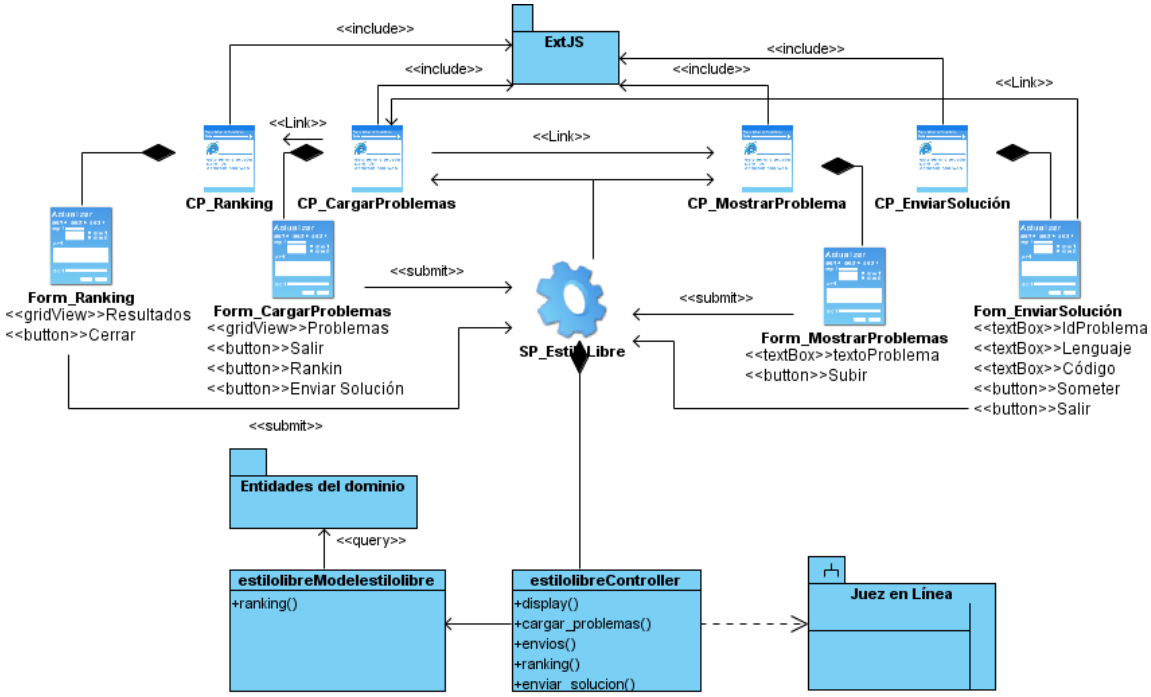


Figura 2.12: *Diagrama de clases del diseño correspondiente al caso de uso Resolver Prueba en Línea.*

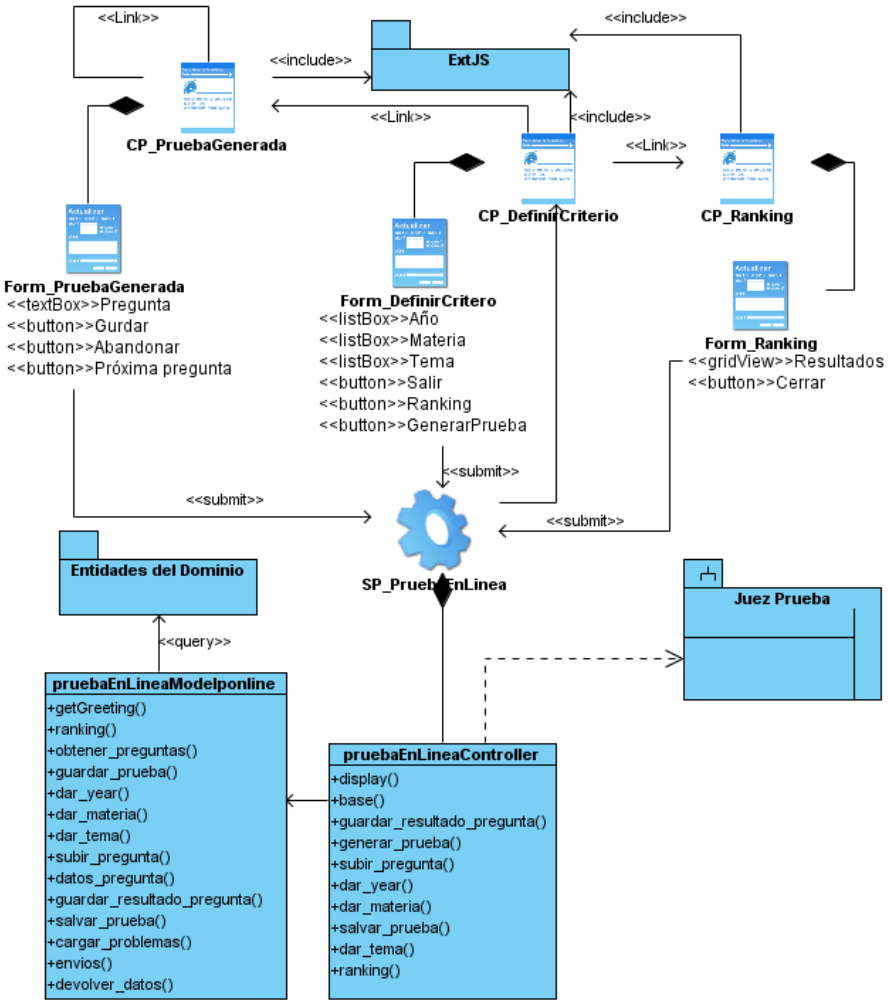


Figura 2.13: *Diagrama de clases del diseño correspondiente al caso de uso Resolver Retos.*

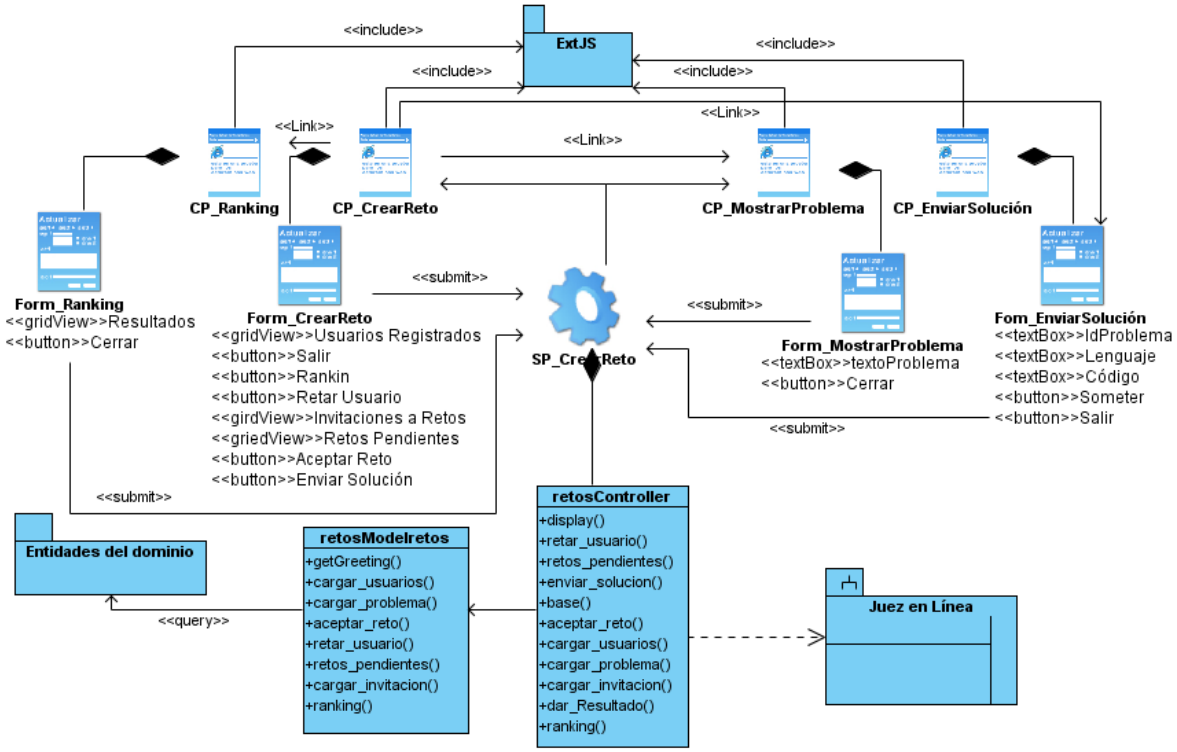
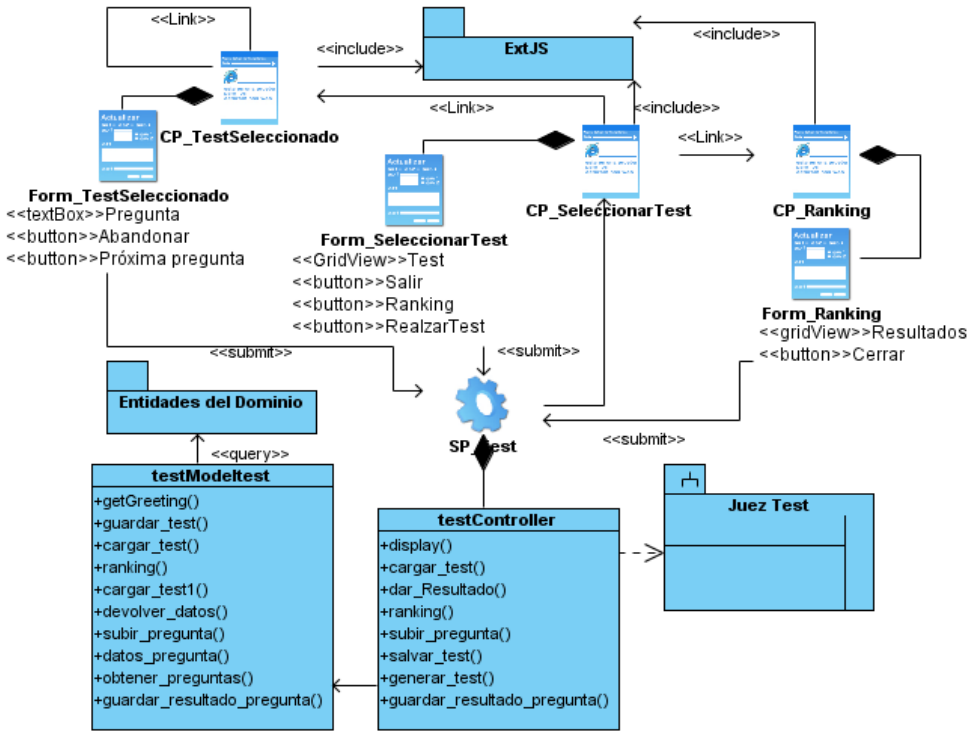


Figura 2.14: *Diagrama de clases del diseño correspondiente al caso de uso Resolver Test.*



Como se puede observar en los diagramas de clases de uso del sistema, representados en las **Figuras 2.11 – 2.14** se ha tenido en cuenta que el desarrollo de cada uno de estos componentes

para el *CMS Joomla*, están basados en el patrón arquitectónico MVC, tal y como se definió en el *epígrafe 2.2*.

Para lograr los objetivos planteados en la definición de esta investigación y darle cumplimiento a los requisitos funcionales identificados y descritos en el *epígrafe 2.1.1* se hace necesario controlar la persistencia de un gran volumen de información, como puede ser: registro de usuario, colección de problemas, preguntas, invitaciones a retos, etc. Seguidamente se presenta el diagrama de clases persistentes y el diagrama del modelo de datos.

Figura 2.15: *Diagrama de clases persistentes.*

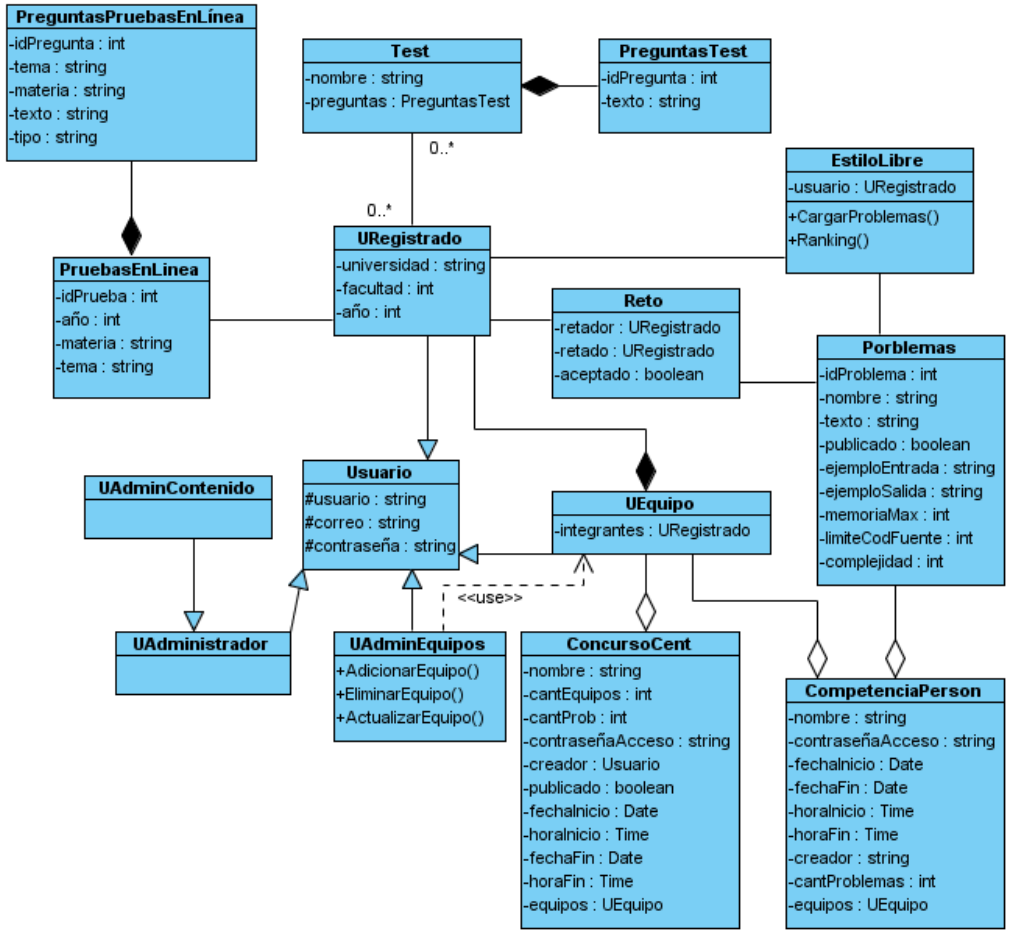
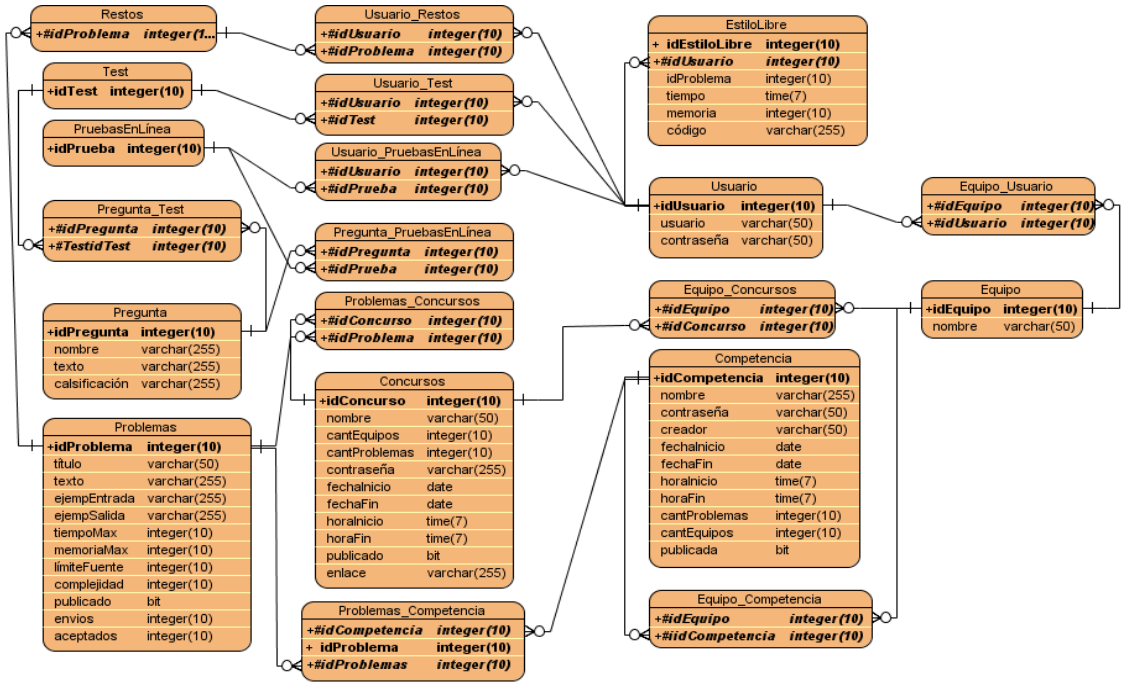


Figura 2.16: *Diagrama del modelo de datos.*



2.5. VALIDACIÓN DEL SISTEMA.

En el presente epígrafe se realiza una validación de la solución propuesta mediante diferentes tipos de pruebas realizadas por especialistas de la entidad Albet - Calisoft. Se incluyen las revisiones técnicas y funcionales.

2.5.1. Revisiones técnicas.

Las revisiones técnicas forman parte de las llamadas evaluaciones estáticas, las cuales buscan, como su nombre lo indica, faltas sobre el sistema en reposo. Estudian los distintos modelos que componen el sistema de software buscando posibles faltas en los mismos, por lo que estas técnicas se pueden aplicar, tanto a requisitos como a modelos de análisis, diseño y código.

El resultado obtenido muestra que todas las no conformidades fueron resueltas y que las no conformidades detectadas fueron de redacción y técnicas, en ninguno de los casos consideradas como críticas; siendo liberado el software en la tercera iteración.

Figura 2.17: *Porcentaje de las no conformidades detectadas por clasificación de la primera iteración.*

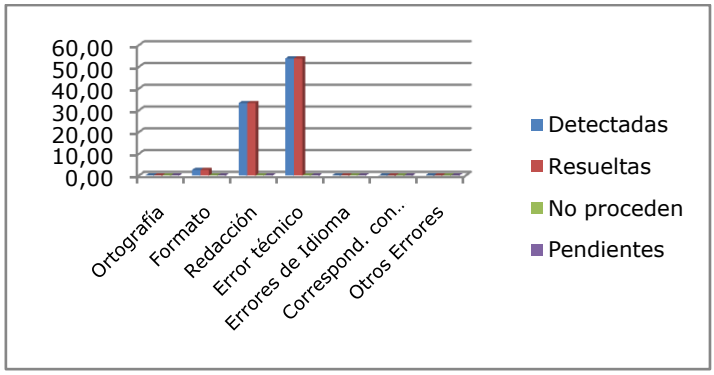
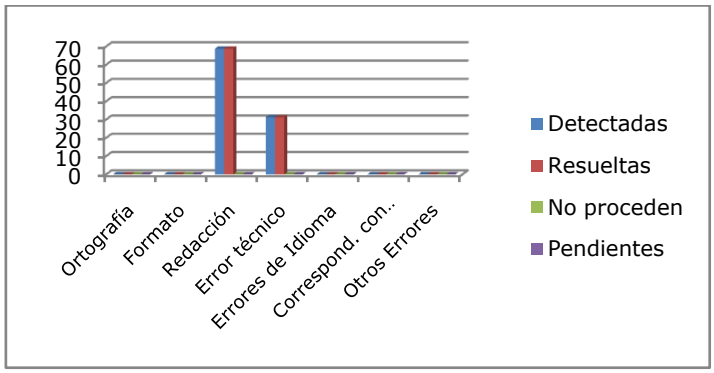


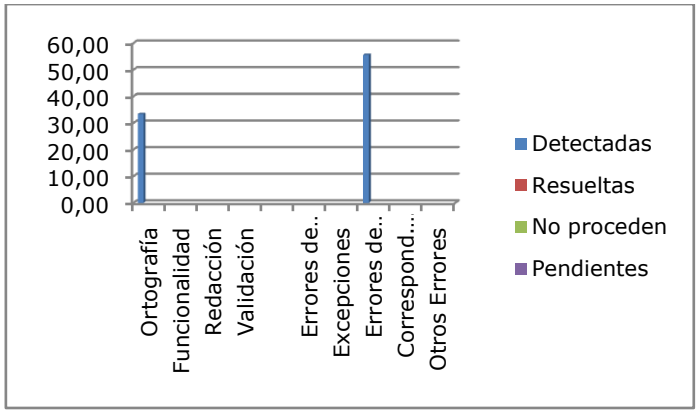
Figura 2.18: *Porcentaje de las no conformidades detectadas por clasificación de la segunda iteración.*



2.5.2. Revisiones funcionales.

Las pruebas funcionales forman parte de las llamadas evaluaciones dinámicas o pruebas de software. Estas se definen como una actividad en la cual un sistema o uno de sus componentes se ejecutan en circunstancias previamente especificadas. De las mismas se identificaron solamente errores ortográficos y de idioma, ver la **Fig. 2.19**, al no usar la codificación adecuada (utf8_spanish2_ci) y olvidar traducir las etiquetas predefinidas en idioma inglés en las librerías de Javascript, ExtJS.

Figura 2.19: *Porcentaje de las no conformidades detectadas en las pruebas funcionales en la primera iteración.*



2.5.3. Evaluación por criterio de expertos.

Al concluir la primera iteración de la aplicación informática desarrollada como resultado de la solución propuesta, la misma fue sometida a una valoración por criterio de expertos, para lo cual se aplicó una encuesta (*Ver instrumento aplicado en el Anexo B*) a un total de once expertos con una experiencia que variaba entre los 6 y 8 años como profesores de las asignaturas de la disciplina de TPC.

Una síntesis biográfica de los expertos encuestados puede ser observada en el *Anexo C*. La muestra de profesores entrevistados estaba conformada como se describe a continuación: dos profesores instructores para un 18.18%, ocho profesores asistentes para un 72.72%, un profesor auxiliar para un 9.09%, dos másteres en ciencias para un 18.18%, un doctor en ciencias para un 9.09%.

El coeficiente de competencia de los expertos obtenido fue de un 36.36% medio y un 63.63% alto y el coeficiente general fue alto con un valor de 0.82*u* (*Ver* **Tabla D.1** *del Anexo D*), un conocimiento promedio de 7.27*u*, un coeficiente de conocimiento de 0.73*u* y un coeficiente de argumentación de 0.92*u*.

La **Tabla 2.6** muestra el criterio final obtenido en cada uno de los aspectos al ser sometidos a la consideración de los encuestados (*Ver las* **Tabla D.2 – Tabla D.5** *del Anexo D*). En el 90.90% de los aspectos se obtuvo un criterio de muy adecuado y solo en el 9.09% se obtuvo un criterio de bastante adecuado. Lo que permite validar la hipótesis planteada.

Tabla 2.6: *Criterio obtenido para cada uno de los aspectos.*

No	Aspectos	Criterio
1	Calidad de la investigación.	Muy adecuado
2	Novedad Científica.	Muy adecuado
3	Aporte Científico.	Bastante Adecuado
4	Necesidad del empleo de la nueva aplicación.	Muy adecuado
5	Satisfacción de las necesidades como medio de enseñanza para el proceso de enseñanza-aprendizaje.	Muy adecuado
6	Contribución al proceso de desarrollo de habilidades con un enfoque sistémico.	Muy adecuado
7	Facilidades de utilización como medio de enseñanza-aprendizaje.	Muy adecuado
8	Posibilidad de uso en varias asignaturas de la disciplina.	Muy adecuado
9	Posibilidades de extensión de la aplicación mediante la inclusión de otros estilos de competencias y funcionalidades.	Muy adecuado
10	Contribución al proceso de enseñanza-aprendizaje y desarrollo de habilidades en las técnicas de programación en la Universidad de las Ciencias Informáticas.	Muy adecuado
11	Posibilidades de utilización de la aplicación informática.	Muy adecuado

Fuente: *Elaboración propia.*

Otros datos que contribuyen a corroborar la validación de la hipótesis se enumeran a continuación:

1. El 82% considera que calidad de la investigación es excelente y 18% que es bueno.
2. El 45% considera que la novedad científica del trabajo es excelente, el 18% que es bueno y el 36% considera que es aceptable.
3. Para el 27% de los encuestados el aporte es excelente, otro 27% considera que es bueno, un 18% lo considera como aceptable, otro 18% lo considera cuestionable y un 9.1% como malo.
4. El 82% de los encuestados considera que es muy necesario el uso de la nueva aplicación, un 9.1% considera que es necesario y el otro 9.1% que es poco necesario.
5. El 67% se mostró totalmente de acuerdo sobre la satisfacción de las necesidades como medio de enseñanza para el proceso de enseñanza-aprendizaje y el otro 33% se mostró de acuerdo.
6. El 55% se mostró totalmente de acuerdo con respecto a la contribución al proceso de desarrollo de habilidades con un enfoque sistémico de la nueva aplicación, otro 18% se mostró como muy de acuerdo y el 27% restante se mostró de acuerdo.
7. El 64% considera que las facilidades de utilización como medio de enseñanza-aprendizaje es excelente, un 18% que es bueno, un 9.1% que es aceptable y el 9.1% restante que es cuestionable.

8. El 82% considera que las posibilidades de uso en varias asignaturas de la disciplina son excelentes y el otro 18% que es bueno.
9. El 73% considera que las posibilidades de extensión de la aplicación mediante la inclusión de otros estilos de competencias y funcionalidades son excelentes y el otro 27% considera que es bueno.
10. El 64% de los encuestados considera que la contribución al proceso de enseñanza-aprendizaje y desarrollo de habilidades en las técnicas de programación es excelente, un 18% considera que es bueno y el 18% restante que es aceptable.
11. El 82% considera que las posibilidades de utilización de la aplicación informática es excelente, el 9.1% que es bueno y el otro 9.1% que es aceptable.
12. El 100% de los encuestados considera que no es necesario suprimir ninguna de las características presentes en la aplicación, un 22% considera que es necesario mejorar la interfaz gráfica, otro 11% que es necesario mejorar la aleatoriedad de las pruebas en línea y el 56% sugirió funcionalidades que deberían ser adicionadas.

De forma general el 36% de los encuestados considera que la investigación es excelente y presenta una alta novedad científica, con aplicabilidad y resultados relevantes; otro 55% considera que la investigación es buena, tiene novedad científica y resultados destacados, solo el 9.1% considera que el resultado es aceptable, suficientemente bueno con reservas.

CONCLUSIONES DEL CAPÍTULO.

- La especificación de los requisitos funcionales y no funcionales permitió definir los casos de usos y la arquitectura base del sistema.
- Se definió una arquitectura estable que permitirá la evolución de la aplicación sin afectar los componentes existentes.
- La utilización de la metodología *RUP* permitió obtener cada uno de los artefactos necesarios para desarrollar una aplicación informática ajustada a los principios definidos y concluir satisfactoriamente la primera iteración.
- Se comprobó, mediante la valoración del criterio de expertos, la pertinencia de la investigación realizada y aplicabilidad de los resultados obtenidos.

CONCLUSIONES GENERALES.

Al finalizar el presente trabajo, se considera cumplido el objetivo definido y validada la hipótesis del la cual se partió. A este planteamiento se puede arribar a partir de las siguientes conclusiones:

- El estudio de las aplicaciones informáticas educativas en el área de las TPC permitió definir su clasificación en términos de sus características y el diagrama de Bejamín Bloom, que facilitará la selección de la aplicación informática más adecuada en dependencia de la actividad docente a realizar.
- La arquitectura definida basada en: capas, componentes, objetos y MVC para el desarrollo del sistema permitirá la evolución de la aplicación de forma rápida y sencilla sin afectar los componentes existentes.
- Se desarrolló la subcapa lógica de estilos de competencias permitiendo satisfacer los principales requisitos funcionales del sistema.
- Se desarrolló la aplicación informática *"Coliseo Virtual"* que apoyará al proceso de aprendizaje y contribuirá la asimilación conocimientos y el desarrollo de habilidades en las técnicas de programación con un enfoque sistémico, validado mediante el criterio de expertos.
- La validación de la solución propuesta mostró la pertinencia de la investigación y veracidad de la hipótesis planteada.

RECOMENDACIONES.

- Incluir funcionalidades que le permita a los profesores supervisar el desempeño de los estudiantes y definir rutas de aprendizaje personalizadas.
- Desarrollar un módulo que permita la integración con el entorno virtual de aprendizaje de la universidad, el *CMS Moodle*.
- Incluir funcionalidades de ayuda en la resolución de los problemas algorítmicos.
- Extender las funcionalidades de la aplicación informática *"Coliseo Virtual"*, mediante la inclusión de otros estilos de competencias.

REFERENCIAS BIBLIOGRÁFICAS.

Achour, Mehdi, et al. 2008. *Manual de PHP.* 2008.

ACM and IEEE. 2008. Computer Science Curriculum 2008: An Interim Revision of CS 2001. s.l. : ACM and IEEE, 2008. p. 102.

ACM, AIS and IEEE. 2005. Computing Curricula 2005 - The Overview Report. s.l. : ACM and IEEE, 2005. p. 62.

Ala-Mutka, Kirsti. 2005. *Codewitz Needs Analysis Literature Study PROBLEMS IN LEARNING AND TEACHING PROGRAMMING.* Institute of Software Systems, Tampere University of Technology. Finland : s.n., 2005.

Almarcegui Aguerri, Fernando, Bartolozzi Castilla, Michael and Cañadas López, Miguel Ángel. 2009. Comparación y elección de un Sistema de Gestión de Contenidos de software libre. 2009. pp. 3-45.

Alonso, Francisco A. Cano. 2007. Los esfuerzos cubanos en la introducción y uso de las TIC en el Sistema Nacional de Educación. *XII Congreso de Informática en la Educación .* Habana : Informática 2007, 2007.

Álvarez de Zayas, Carlos. 1992. *La escuela en la vida.* 1ra. Ciudad de la Habana : Pueblo y Educación, 1992. p. 170. 959-07-0015-2.

—. 1999. *La Pedagogía como Ciencia.* 1ra. Habana : Pueblo y Educación, 1999. p. 254. Vol. 1. 959-258-037-5.

Aptana. 2009. Aptana. [Online] 2009. http://www.aptana.com/.

Arora, Ashish and Gambardella, Alfonso. 2005. *From Underdogs to Tigers: The Rise and Growth of the Software Industry in Brazil, China, India, Ireland, and Israel.* Oxford : Oxford University Press, 2005. 9780199275601.

Awad, M. A. 2005. *A Comparison between Agile and Traditional Software Development Methodologies.* Computer Science and software Engineering, University of Western Australia. 2005. p. 84.

Ben-Ari, M., Ragonis, N. and Levy, R.B.B. 2002. *A Vision of Visualization in Teaching Object-Oriented Programming.* Department of Science Teaching, Weizmann Institute of Science. Rehovot, Israel : s.n., 2002.

Booch, Grady. 1998. *Object Oriente Analysis an Desing with applications.* 2da. California, Estados Unidos : Addison-Wesley, 1998. p. 542.

Booch, Grady, Jacobson, Ivar and Rumbauch, James. 1999. *El Lenguaje Unificado de Modelado.* 1ra. Universidad de Murcia, España : Addison-Wesley, 1999. p. 419.

Chapman, James. 2007. Project Management Principles & Training. *Principle Based Project Management Information and Training Site.* [Online] 2007. http://www.hyperthot.com/project.htm.

Ciudad Ricardo, Febe A. and Herrera Martínez, Yosnel. 2007. DoMet como propuesta para la modelación de entornos organizacionales complejos y difusos. *UCIENCIA.* Habana : Universidad de las Ciencias Informáticas, 2007. p. 11.

Clayberg, Eric and Rubel, Dan. 2006. *Eclipse: Building Commercial-Quality Plug-ins.* 2006. p. 864.

Comisión de Carrera. 2009. *Modelo del Profesional y Objetivos de la carrera de Ingeniería en Ciencias Informáticas.* Dirección de Formación, Universidad de las Ciencias Informáticas. Habana : s.n., 2009. p. 5.

—. 2002. *Plan de Estudio A.* Dirección de Formación, Universidad de las Ciencias Informáticas. Habana : s.n., 2002. p. 5.

—. 2003. *Plan de Estudio C.* Dirección de Formación, Universidad de las Ciencias Informáticas. Habana : s.n., 2003. p. 5.

—. 2004. *Plan de Estudio D.* Dirección de formación, Universidad de las Ciencias Informáticas. Habana : s.n., 2004. p. 5.

—. 2006. *Plan de Estudio E.* Dirección de Formación, Universidad de las Ciencias Informáticas. Habana : s.n., 2006. p. 5.

—. 2008. *Plan de Estudio F.* Dirección de Formación, Universidad de las Ciencias Informáticas. Habana : s.n., 2008. p. 5.

—. 2009. *Plan de Estudio G.* Dirección de Formación, Universidad de las Ciencias Informáticas. Habana : s.n., 2009. p. 5.

Converse, Tim, Park, Joyce and Morgan, Clark. 2004. *PHP5 and MySQL Bible.* s.l. : Wiley Publishing, Inc, 2004. p. 1083.

—. 2003. *PHP5 and MySQL Bible.* s.l. : Wiley Publishing, Inc, 2003. p. 1083.

Conway, Mattew. 1997. *Alice: Easy-to-Learn 3D Scripting for Novices.* 1997. p. 242.

Dann, W. P., Cooper, S. and Pausch, R. 2006. *Learning to Program With Alice.* s.l. : Prentice Hall, 2006.

Deitel, H. M. and Deitel, P. J. 2005. *C++ How to Program.* 5ta. s.l. : Prentice Hall, 2005. p. 1536.

Encinosa, Lázaro J. Blanco. 2002. *Apuntes para una historia de la Informática en Cuba.* 2002.

Enterprise Architect. 2009. UML tools for software development and modelling - Enterprise Architect UML modeling tool. [Online] 2009. http://www.sparxsystems.com.au/.

Fiallo Gómez, Marnie and García Alfonso, Navil. 2006. Un millón de egresados de los Joven Club de Computación. *Granma Internacional.* 2006.

Fuentes González, Homero Calixto and Álvarez Valiente, Ilsa Bernardina. 2003. *Introducción a la didáctica. La conducción del proceso de enseñanza-aprendizaje en la educación superior.* Santiago de Cuba : s.n., 2003. p. 106.

Gallego Vázquez, José Antonio. 2003. *Desarrollo web con PHP y MySQL.* Madrid, España : Anaya Multimedia, 2003.

—. 2003. *Desarrollo web con PHP y MySQL.* Madrid, España : Anaya Multimedia, 2003.

Gilfillan, Ian. 2004. *La Biblia de MySQL.* s.l. : Anaya Multimedia, 2004. p. 841.

González García, Gerardo. 2008. ¿Qué es Software educativo, cómo se clasifica y cuáles son sus características? *Profesor Interactivo.* [Online] 2008. http://profesorinteractivo.blogia.com/.

Graells, P. M and Virtuales, T. 1999. Criterios para la clasificación y evaluación de espacios web de interés educativo. *Educar.* 1999. p. 95.

Henriksen, P. and Kölling, M. 2004. *greenfoot: combining object visualisation with interaction.* s.l. : ACM New York, NY, USA, 2004. pp. 73-82.

Hernández, Michael J. 2003. *Database Design for Mere Mortals. A Hands-On Guide to Relational Database Design.* s.l. : Addison Wesley, 2003. p. 672.

Jacobson, Ivar, Booch, Grady and Rumbaugh, James. 1999. *El proceso unificado del desarrollo de software.* 1ra. México : Addison-Wesley, 1999. p. 464.

Jiménez Fernández, Juan Carlos y Rodríguez Serrano, María Natacha. 2008. Joven Club de Computación, un aliado del proceso pedagógico en Cuba. Habana : FORBES, 2008. Vol. 7ma, pág. 20.

Joomla Spanish. 2009. Joomla Spanish. [Online] 2009. http://www.joomlaspanish.org/.

Kiczales, G. J, et al. 2002. *Aspect-oriented programming. US 6467086B1* United States, 2002.

Kline, Kevin E. 2004. *SQL in a nutshell a destopk quick reference.* United States of America : O'Reilly, 2004. p. 711.

Koala Project. 2002. DynamicJava. 2002.

Korth, Henry F. and Silberschatz, Abraham. 2006. *Fundamentos de Bases de Datos.* 5ta. s.l. : Mc.Graw Hill, 2006. p. 944.

Kroenke, David M. 1996. *Procesamiento de Bases de Datos. Fundamentos, diseño e instrumentación.* México : Prentice Hall, 1996.

Külling, M. and Henriksen, P. 2005. *Game programming in introductory courses with direct state manipulation.* University of Kent. s.l. : ACM New York, NY, USA, 2005. pp. 59-63.

Kurland, D. Midian, et al. 1986. A study of the development of programming ability and thinking skills in high school students. s.l. : Educational Computer Research, 1986. Vol. 2, pp. 429-458.

Larman, Craig. 1999. *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* México : Prentice Hall, 1999. p. 536.

LeBlanc, Joseph. 2007. *Learning Joomla! 1.5 Extension Development Creating Modules, Components, and Plug-Ins with PHP.* BIRMINGHAM - MUMBAI : Packt Publishing Ltd, 2007. p. 171.

López, T. 1979. *Resultados principales del desarrollo de los medios de computación en Cuba.* 1979.

Maloney, J., et al. 2004. Scratch: a sneak preview education. 2004. pp. 104-109.

Marquès, Pere. 2009. Clasificación del software educativo. *La Uno - Educación y TIC. El Blog de Educación y Nuevas Tecnologías de la Escuela Nº 1 de Neuquén Capital.* [Online] 2009. http://escuelauno-nqn.blogspot.com/.

—. 1996. *El software educativo.* Universidad Autónoma de Barcelona. 1996.

Meyer, Bertrand. 1999. *Construcción de software orientado a objetos.* 2da. Madrid, España : Prentice Hall, 1999. p. 1248. Vol. 1.

Monteagudo, Pedro L. 2003. *Software Educativo para el Adiestramiento en Ruidos Respiratorios.* CECAM. 2003. Tesis presentada en opción al Título Académico de Master en Informática en Salud.

Moreno, A. and Myller, N. 2003. *Producing an Educationally Effective and Usable Tool for Learning, The Case of the Jeliot Family.* University of Joensuu. 2003.

Myller, Niko. 2004. *The Fundamental Design Issues of Jeliot 3.* Department of Computer Science, University of Joensuu. s.l. : Computer Science, 2004. p. 92, Master's Thesis.

Pears, Arnold, et al. 2007. A survey of literature on the teaching of introductory programming. Dundee, Scotland : ACM, 2007. pp. 204-223.

Portuondo Padrón, Roberto and Basulto Morales, Carlos. 2000. *Introducción a la didáctica. La didáctica como ciencia.* 1ra. Camagüey : s.n., 2000. p. 77.

Pressman, Roger S. 2005. Ingeniería de Software un enfoque práctico. 6ta s.l. : McGraw-Hill, 2005. p. 980.

—. 1998. *Ingeniería de Software. Un enfoque práctico.* 4ta. Madrid, España : s.n., 1998. p. 600. Vol. 2.

Rational Rose. 2009. IBM Rational Software. [Online] 2009. http://www-01.ibm.com/software/rational/.

Rational Unified Process. 2006. Ayuda de Rational. *Ciclo vital de RUP.* [Aplicación web]. s.l. : IBM Corporation, 2006.

Reina Quintero, Antonia Mª. 2000. *Visión General de la Programación Orientada a Aspectos.* Universidad de Sevilla. Sevilla, España : s.n., 2000. p. 42, Interno. SI-2000-11.

Reynoso, Carlos Billy. 2005. MSDN webcast No.2. Profundizando en Estilos de Arquitectura de Software. [Online] 2005. http://www.microsoft.com/spanish/msdn/latam/mediacenter/webcast/architect.aspx.

Robins, Anthony, Rountree, Janet and Rountree, Nathan. 2003. Learning and teaching programming: A review and discussion. s.l. : Computer Science Education, 2003. Vol. 13, pp. 137-172.

Ruenda, Carlos A. 2007. *LoroEDI – Entorno de Desarrollo Integrado de Loro.* Universidad Autónoma de Manizales. Colombia : s.n., 2007. p. 37, Manual de Usuario.

Suarez-Inclan Rivero, Yadilka. 2008. *Plan de Perfeccionamiento de la Disciplina de Programación.* Departamento de Técnicas de Programación de Computadoras, Universidad de las Ciencias Informáticas. Habana : s.n., 2008.

Suarez-Inclank Rivero, Yadilka. 2009. *Resultados de la 4ta convocatoria del Examen Final de la Disciplina de Técnicas de Programación de Computadoras.* Técnicas de Programación de Computadoras, Universidad de las Ciencias Informáticas. Habana : s.n., 2009. Resumen de resultados del EFDTPC.

Talízina, N. F. 2000. *La formación de las habilidades del pensamiento matemático.* Universidad Autónoma de San Luis Potosí : UASLP, 2000. p. 280. 9687674881, 9789687674889.

Terry, P.D. 1996. *Compilers and Compiler Generators an introduction with C++.* s.l. : International Thomson Computer Press, 1996. p. 435.

Triana Cordoví, Juan. 2002. El desempeño de la economía cubana en el primer semestre de 2002. Habana, Economía y Desarrollo : s.n., 2002. Vol. 131, p. 14.

Tsai, Alice Y. H. 1990. *Sistemas de Bases de Datos. Administración y uso.* México : Prentice Hall, 1990.

Ulloa. 2002. *Desarrollo de habilidades profesionales.* Universidad de Camaguey. Camagüey : s.n., 2002. p. 87.

Visual Paradigm. 2009. UML and Business Process modeling tool for software development project - Visual Paradigm for UML. [Online] 2009. http://www.visual-paradigm.com/product/vpuml/index.jsp.

Zapata, Carlos Mario, et al. 2009. Aproximación a una ontología para lenguajes de modelado gráfico. *Revista de ingeniería.* Habana : s.n., 2009. pp. 16-24.

Zilberstein Toruncha, José, Portela Falguera, Rolando and Mcpherson Sayú, Margarita. 1999. *Didácticas Integradora de las Ciencias vs Didáctica Tradicional. Experiencia Cubana.* Ciudad de la Habana : Academia, 1999. p. 33.

ANEXOS

ANEXO A: INSTRUMENTO APLICADO PARA LA VALIDACIÓN MEDIANTE CRITERIO DE EXPERTOS.

Estimado Colega:

Usted ha sido seleccionado por su calificación científico-técnica, sus años de experiencia y los resultados alcanzados en su labor profesional, como experto para evaluar los resultados técnicos de la investigación: *"Coliseo Virtual"* Aplicación informática de apoyo al aprendizaje de las técnicas de programación. Le agradecemos que ofrezca sus ideas y criterios sobre las bondades, deficiencias e insuficiencias que presenta la aplicación en su futura aplicación práctica.

Marque con una X, según su criterio, sus conocimientos sobre el tema:

1	2	3	4	5	6	7	8	9	10

Antes de proceder a la valoración de los criterios nos gustaría que autoevalúe sus conocimientos basado en los siguientes indicadores:

No. Pregunta	FUENTES DE ARGUMENTACION	Grado de influencia de cada una de las fuentes en sus criterios.		
		A (alto)	M (medio)	B (bajo)
P1	Análisis teóricos realizados por usted.			
P2	Su experiencia obtenida.			
P3	Trabajos de autores nacionales.			
P4	Trabajos de autores extranjeros.			
P5	Su propio conocimiento del estado del problema en el extranjero.			
P6	Su intuición.			

Los aspectos sobre los cuales debe emitir su criterio deben ser evaluados según la siguiente escala:

Escala	Código	Escala	Código
Excelente	E	Totalmente de acuerdo	TA
Bueno	B	Muy de acuerdo	MA
Aceptable	A	De acuerdo	A
Cuestionable	C	Parcialmente de acuerdo	PA
Malo	M	En desacuerdo	D

Aspectos		Calificación				
Criterio de mérito científico		**E**	**B**	**A**	**C**	**M**
1.	Calidad de la investigación.					
2.	Novedad Científica.					
3.	Aporte Científico.					
Criterio de implantación		**TA**	**MA**	**A**	**PA**	**D**
4.	Necesidad del empleo de la nueva aplicación.					
5.	Satisfacción de las necesidades como medio de enseñanza para el proceso de enseñanza-aprendizaje.					
6.	Contribución al proceso de desarrollo de habilidades con un enfoque sistémico.					
Criterios de Generalización		**E**	**B**	**A**	**C**	**M**
7.	Facilidades de utilización como medio de enseñanza-aprendizaje.					
8.	Posibilidad de uso en varias asignaturas de la disciplina.					
9.	Posibilidades de extensión de la aplicación mediante la inclusión de otros estilos de competencias y funcionalidades.					
Criterios de impacto		**E**	**B**	**A**	**C**	**M**
10.	Contribución al proceso de enseñanza-aprendizaje y desarrollo de habilidades en las técnicas de programación en la Universidad de las Ciencias Informáticas.					
11.	Posibilidades de utilización de la aplicación informática.					

Valoración Final:

Elementos a Suprimir:

Elementos a Mejorar:

Elementos a Añadir:

Categoría Final de la Investigación:

___**Excelente:** *Alta novedad científica, con aplicabilidad y resultados relevantes.*

____**Bueno:** *Novedad Científica. Resultados Destacados.*

____**Aceptable:** *Suficientemente bueno con reservas.*

____**Cuestionable:** *No tiene relevancia científica.*

____ **Malo:** *No aplicable.*

Nota: *Sus criterios y opiniones se operarán de forma anónima. Le agradecemos de antemano su valiosa colaboración y estamos seguros que su valiosa experiencia y señalamientos críticos contribuirán a perfeccionar la aplicación informática "Coliseo Virtual", tanto en su concepción teórica como en su futura aplicación.*

ANEXO B: SÍNTESIS BIOGRÁFICA DE LOS EXPERTOS ENCUESTADOS.

- *Lic. Yadilka Suárez-Inclan Rivero:* profesora asistente Licenciada en Ciencia de la Computación, siete años impartiendo las asignaturas de la disciplina de TPC, durante los cursos 2004 – 2005 y 2005 – 2006 fue asesora de la asignatura de Introducción a la Programación y Programación 1, vicedecana de formación durante el curso 2006 – 2007 y se desempeña desde el curso 2007 – 2008 como Jefa del Departamento Docente Central de TPC.

- *DrC. Rafael Arturo Trujillo Rasúa:* profesor auxiliar Licenciado en Ciencia de la Computación y Doctor en Informática, ocho años impartiendo las asignaturas de la disciplina de TPC. En la actualidad se desempeña como asesor, labor que realiza desde el curso 2004 - 2005. Ha impartido las asignaturas de Introducción a la Programación, Programación 1, Programación 2 y Programación 4.

- *Msc. Yuniesky Coca Bergolla:* profesor asistente Licenciado en Ciencia de la Computación y Máster en Informática Aplicada. Siete años impartiendo las asignaturas de la disciplina de TPC. Ha impartido las asignaturas de Introducción a la Programación, Programación 1, Programación 2, Programación 4, Gráfico por computadora e Inteligencia Artificial. Ha sido jefe de proyecto y en la actualidad se desempeña como jefe de Departamento de Técnicas de Programación desde el curso 2007 – 2008.

- *Msc. Yoel Caisés Almaguer:* profesor asistente Licenciado en Ciencia de la Computación y Máster en Ciencia de la Computación. Siete años de experiencia impartiendo las asignaturas de la disciplina de TPC. Ha impartido las asignaturas de Introducción a la Programación, Programación 1, Programación 2, Programación 4 y Técnicas de Compilación.

- *Lic. Carlos Luis Milian del Valle*: profesor asistente Licenciado en Ciencia de la Computación, ocho años impartiendo las asignaturas de la disciplina de TPC, durante los cursos 2004 – 2005, 2005 – 2006, 2006 - 2007 fue asesor de la asignatura de Introducción a la Programación y Programación 1. En la actualidad se desempeña desde el curso 2007 – 2008 como Jefe del Departamento Docente Central de Inteligencia Artificial.

- *Lic. Karel Osorio Ramirez:* profesor asistente Licenciado en Ciencia de la Computación. Seis años de experiencia impartiendo las asignaturas de la disciplina de TPC. Ha impartido las asignaturas de Introducción a la Programación, Programación 1, Programación 2 y Programación 4. En la actualidad se desempeña como asesor desde el curso 2007 – 2008.

- *Lic. José Albert Cruz Almaguer:* profesor asistente Licenciado en Ciencia de la Computación, seis años impartiendo las asignaturas de la disciplina de TPC, durante los cursos 2006 – 2007, 2007 – 2008 y 2008 - 2009 se desempeña como asesor de la asignatura de Introducción a la Programación y Programación 1. En la actualidad se desempeña como profesor de las asignaturas de Introducción a la Programación y Programación 1.

- *Lic. David Silva Barreras:* profesor asistente Licenciado en Ciencia de la Computación, siete años impartiendo las asignaturas de la disciplina de TPC. En el curso 2004 – 2005 se desempeña como vicedecano de producción, durante el curso 2005 – 2006 y 2006 – 2007 se desempeña como jefe de proyecto. Ha impartido las asignaturas de Introducción a la Programación, Programación 1, Programación 2 y Programación 4. En la actualidad se

desempeña como profesor de las asignaturas de Introducción a la Programación y Programación 1.

- *Lic. Arismayda Dorado Risco:* profesora instructor Licenciada en Ciencia de la Computación, seis años impartiendo las asignaturas de la disciplina de TPC. Ha impartido la asignatura de Introducción a la Programación, Programación 1, Programación 2 y Programación 4. En la actualidad se desempeña como profesora de Preparación para el Examen Final de Disciplina.
- *Ing. Ernesto Medina Delgado:* profesor instructor Ingeniero en Ciencias Informáticas, tres años impartiendo las asignaturas de la disciplina de TPC. Ha impartido las asignaturas de Introducción a la Programación, Programación 1, Arquitectura y Patrones y Seguridad Informática. Ha sido jefe de proyecto y en la actualidad se desempeña desde el curso 2008 – 2009 como Jefe del Polo de Informática Jurídica.
- *Lic. Lester Rodríguez Vallejo:* profesor asistente Licenciado en Ciencia de la Computación, siete años impartiendo las asignaturas de la disciplina de TPC. Ha impartido las asignaturas de Introducción a la Programación, Programación 1 y Programación 2. En la actualidad se desempeña como jefe de Departamento de Técnicas de Programación desde el curso 2008 – 2009.

ANEXO C: RESULTADOS DEL MÉTODO DE EVALUACIÓN POR CRITERIO DE EXPERTO.

Tabla D.1: *Competencia de los expertos sobre el tema.*

No	Experto	Conocimiento	P1	P2	P3	P4	P5	P6	Karg	Kcon	Kcom	Competencia
1	1	6	0.2	0.5	0.05	0.05	0.05	0.05	0.9	0.6	0.75	Medio
2	2	8	0.2	0.5	0.05	0.05	0.05	0.05	0.9	0.8	0.85	Alto
3	3	8	0.2	0.5	0.05	0.05	0.05	0.05	0.9	0.8	0.85	Alto
4	4	7	0.2	0.5	0.05	0.05	0.05	0.05	0.9	0.7	0.8	Alto
5	5	6	0.2	0.5	0.05	0.05	0.05	0.05	0.9	0.6	0.75	Medio
6	6	9	0.3	0.5	0.05	0.05	0.05	0.05	1	0.9	0.95	Alto
7	7	8	0.3	0.5	0.05	0.05	0.05	0.05	1	0.8	0.9	Alto
8	8	6	0.2	0.5	0.05	0.05	0.05	0.05	0.9	0.6	0.75	Medio
9	9	6	0.2	0.5	0.05	0.05	0.05	0.05	0.9	0.6	0.75	Medio
10	10	8	0.2	0.5	0.05	0.05	0.05	0.05	0.9	0.8	0.85	Alto
11	11	8	0.2	0.5	0.05	0.05	0.05	0.05	0.9	0.8	0.85	Alto
Total		**7.2727**	**0.22**	**0.50**	**0.05**	**0.05**	**0.05**	**0.05**	**0.92**	**0.73**	**0.82**	**Alto**

- *Karg*: coeficiente de argumentación del conocimiento.
- *Kcon*: coeficiente de conocimiento.
- *Kcom*: coeficiente de competencia

Tabla D.2: *frecuencias absolutas.*

No	Elementos	C1	C2	C3	C4	C5	Total
1	Calidad de la investigación.	9	2	0	0	0	11
2	Novedad Científica.	5	2	4	0	0	11
3	Aporte Científico.	3	3	2	2	1	11
4	Necesidad del empleo de la nueva aplicación.	9	1	1	0	0	11
5	Satisfacción de las necesidades como medio de enseñanza para el proceso de enseñanza-aprendizaje.	6	2	3	0	0	11
6	Contribución al proceso de desarrollo de habilidades con un enfoque sistémico.	6	5	0	0	0	11
7	Facilidades de utilización como medio de enseñanza-aprendizaje.	7	2	1	1	0	11
8	Posibilidad de uso en varias asignaturas de la disciplina.	9	2	0	0	0	11
9	Posibilidades de extensión de la aplicación mediante la inclusión de otros estilos de competencias y funcionalidades.	8	3	0	0	0	11
10	Contribución al proceso de enseñanza-aprendizaje y desarrollo de habilidades en las técnicas de programación en la Universidad de las Ciencias Informáticas.	7	2	2	0	0	11
11	Posibilidades de utilización de la aplicación informática.	9	1	1	0	0	11

Tabla D.3: *frecuencias acumuladas.*

No	Aspectos	C1	C2	C3	C4	C5
1	Calidad de la investigación.	9	11	11	11	11
2	Novedad Científica.	5	7	11	11	11
3	Aporte Científico.	3	6	8	10	11
4	Necesidad del empleo de la nueva aplicación.	9	10	11	11	11
5	Satisfacción de las necesidades como medio de enseñanza para el proceso de enseñanza-aprendizaje.	6	8	11	11	11
6	Contribución al proceso de desarrollo de habilidades con un enfoque sistémico.	6	11	11	11	11
7	Facilidades de utilización como medio de enseñanza-aprendizaje.	7	9	10	11	11
8	Posibilidad de uso en varias asignaturas de la disciplina.	9	11	11	11	11
9	Posibilidades de extensión de la aplicación mediante la inclusión de otros estilos de competencias y funcionalidades.	8	11	11	11	11
10	Contribución al proceso de enseñanza-aprendizaje y desarrollo de habilidades en las técnicas de programación en la Universidad de las Ciencias Informáticas.	7	9	11	11	11
11	Posibilidades de utilización de la aplicación informática.	9	10	11	11	11

Tabla D.4: *frecuencias relativas.*

No	Aspectos	C1	C2	C3	C4	C5
1	Calidad de la investigación.	0.8181	0.9999	0.9999	0.9999	0.9999
2	Novedad Científica.	0.4545	0.6364	0.9999	0.9999	0.9999
3	Aporte Científico.	0.2727	0.5455	0.7273	0.9091	0.9999
4	Necesidad del empleo de la nueva aplicación.	0.8181	0.9091	0.9999	0.9999	0.9999
5	Satisfacción de las necesidades como medio de enseñanza para el proceso de enseñanza-aprendizaje.	0.5454	0.7273	0.9999	0.9999	0.9999
6	Contribución al proceso de desarrollo de habilidades con un enfoque sistémico.	0.5454	0.9999	0.9999	0.9999	0.9999
7	Facilidades de utilización como medio de enseñanza-aprendizaje.	0.6363	0.8182	0.9091	0.9999	0.9999
8	Posibilidad de uso en varias asignaturas de la disciplina.	0.8181	0.9999	0.9999	0.9999	0.9999
9	Posibilidades de extensión de la aplicación mediante la inclusión de otros estilos de competencias y funcionalidades.	0.7272	0.9999	0.9999	0.9999	0.9999
10	Contribución al proceso de enseñanza-aprendizaje y desarrollo de habilidades en las técnicas de programación en la Universidad de las Ciencias Informáticas.	0.6363	0.8182	0.9999	0.9999	0.9999
11	Posibilidades de utilización de la aplicación informática.	0.8181	0.9091	0.9999	0.9999	0.9999

Tabla D.5: *Puntos de corte.*

No	Aspectos	C1	C2	C3	C4	Suma	P	N-P	Criterio
1	Calidad de la investigación.	0.91	3.72	3.72	3.72	12.07	3.02	-1.22	Muy adecuado
2	Novedad Científica.	-0.11	0.35	3.72	3.72	7.67	1.92	-0.12	Muy adecuado
3	Aporte Científico.	-0.60	0.11	0.60	1.34	1.45	0.36	1.43	Bastante Adecuado
4	Necesidad del empleo de la nueva aplicación.	0.91	1.34	3.72	3.72	9.68	2.42	-0.62	Muy adecuado
5	Satisfacción de las necesidades como medio de enseñanza para el proceso de enseñanza-aprendizaje.	0.11	0.60	3.72	3.72	8.16	2.04	-0.24	Muy adecuado
6	Contribución al proceso de desarrollo de habilidades con un enfoque sistémico.	0.11	3.72	3.72	3.72	11.27	2.82	-1.02	Muy adecuado
7	Facilidades de utilización como medio de enseñanza-aprendizaje.	0.35	0.91	1.34	3.72	6.31	1.58	0.22	Muy adecuado
8	Posibilidad de uso en varias asignaturas de la disciplina.	0.91	3.72	3.72	3.72	12.07	3.02	-1.22	Muy adecuado
9	Posibilidades de extensión de la aplicación mediante la inclusión de otros estilos de competencias y funcionalidades.	0.60	3.72	3.72	3.72	11.76	2.94	-1.14	Muy adecuado
10	Contribución al proceso de enseñanza-aprendizaje y desarrollo de habilidades en las técnicas de programación en la Universidad de las Ciencias Informáticas.	0.35	0.91	3.72	3.72	8.70	2.17	-2.17	Muy adecuado
11	Posibilidades de utilización de la aplicación informática.	0.91	1.34	3.72	3.72	9.68	2.42	-0.62	Muy adecuado
	Suma	**4.45**	**20.43**	**35.41**	**38.53**	**98.81**			
	Punto de Corte	**0.40**	**1.86**	**3.22**	**3.50**				