UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Análisis y diseño del módulo “Capacitación” en el Sistema para la Gestión de la Calidad (Facultad 8)

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

Autor: Emilio Ruben Ferrera Omar.
Tutor: Ing. Ramón Enrique González Peralta.
Co-Tutora: Ing. Yadira Machado Peña.

JUNIO, 2009

Pensamiento

"En Cuba nadie ha hecho tanto en tan poco tiempo."

Fidel Castro Ruz

Declaración de autoría

Declaro ser autor de la presente tesis y reconozco a la **Universidad de las Ciencias Informáticas** los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los __ días del mes de junio del año 2009.

FIRMA
Autor: Emilio Ruben Ferrera Omar

FIRMA
Tutor: Ing. Ramón Enrique González Peralta

FIRMA
Co-tutora: Ing. Yadira Machado Peña

Agradecimientos

Me resulta extremadamente difícil agradecer en pocas palabras a tantas personas buenas que de un modo u otro aportaron algo en el logro en mi carrera profesional. Trataré de no olvidar a nadie, pues a veces pecamos irremediablemente de olvidar a alguno de esos o esas que hicieron posible uno de mis sueños.

A mi tutor y amigo, por haberme ayudado en todo lo que estuvo a su alcance, pues me apoyó desde el principio hasta el fin para que este trabajo contara hoy con la alta calidad que presenta el mismo.

Muchas GRACIAS Ramón, te agradezco mucho por tu empeño y sacrificio…

***A mi co-tutora*,** por ayudarme en la ardua tarea de la revisión de la tesis.

A personas queridas, Mario González y Dagmaris Martínez.

A Made, Mayra, Raisa, Plutín, Cheo, Walter, Quiala, Rosales, Julito, Lídice Cid, Eligio, Esther Rojas, María Luz, Luisa, porque han sido incondicional conmigo siempre.

A todos mis compañeros del grupo 8504. Y en especial a Roberto, Ochoa y Montano.

A mis compañeros de la UCI, Yeleyne, Yaimara, Raicel, Tapia, Maikel, Marlon, Carlos Pupo, Dariel, Alexei, Ariam, Maidelis, Isbel, Frank Ismauri, Andrés, Gustavo, Rosalba, Leovys, Liniet, Henry, Pablo, Yan, Ernesto, Brian, Monchy, Onna, Jesús, Héctor, Güemes, Leudis…

A mis queridos profesores, Lisandra Gibert, Arcadio, Osiris, Haydee Cruz, Lidisvey Herrero, Greisy, Yunexis Rodríguez, Surelys Veunes, Sailyn, María de los Ángeles Montero, Francisca Taño, Alcides Cabrera, Yusnier, Damir Góngora, Isabel Lombillo, Alexander Fernández, Jorge Cano, Juan José, Osmany Ferrer, Dannier Trinchet, Maylen, Celia María, Karel Ruiz, Yusi, Karen Yolanda…

A otras personas que han sido magníficas conmigo en la vida, Carlos Castellanos, Mary y Dago, Luis Bressler, Lourdes, Arturo, Claudia, Dolia y Ernesto, La Gata y familia, Franklin y Luisita.

Dedicatoria

A Caridad Omar Cardero,
porque todo lo que soy te lo debo a ti.

A la Revolución, a Fidel, a Raúl y a la UCI por darme la oportunidad de convertirme en un profesional.

A mi mamá, por ayudarme en todo momento y tener siempre Fé en mí.

A toda mi familia (Adela y familia, Magda y Molla, Urbanito y FAMILIA, Eva, Humberto Camilo y familia, Angelito y Gisel, Renato y Marinita, Fákil y Angelitín, Idalia y familia, Hassan, Bijirita), porque sin su apoyo no hubiese sido posible que hoy estuviera presentando esta memoria escrita.

A mis tres hermanitos queridos Rubencito, Yuliesita y Yuni, ustedes tres y mis sobrinos, son mi mayor tesoro en la vida. Los quiero mucho.

A mi papá Emilio Ruben Ferrera De Nacimiento, a quien debo la enseñanza de los principios por los que me rijo en la vida.

A mi segunda madre Marbe, quien me profesa un amor incondicional.

A mi abuela Gloria Cardero, quien en los últimos años de su vida me brindó mucho amor y confió siempre en la realización de mis sueños, y en los que como ella decía aún faltan por cumplir.

A Yina Aurora (novia y futura esposa), porque eres muy especial para mí.

A mi tercera madre María Elena Ramírez**,** por brindarme tanto amor y apoyo en estos inolvidables cinco años de mi carrera. Gracias.

Resumen

En medio del cambio actual de la educación cubana, es que surge la Universidad de las Ciencias Informáticas (UCI), para brindar rápida respuesta a las necesidades de formación de profesionales que impulsen la Industria Cubana del Software (ICS). En la UCI existe un gran crecimiento de la actividad productiva, donde cada día se hace más inevitable contar con personal calificado en los temas de calidad, gestión e ingeniería de software. Por eso se hace necesario construir un sistema informático que ayude a elevar aun más la capacitación de los estudiantes. El mismo debe ser capaz de brindar reportes para controlar cuales estudiantes están interesados en acreditar el perfil de calidad o conocer los cursos que ha vencido un determinado estudiante. Este sistema propuesto tendrá como premisa mantener organizada la documentación relacionada con los cursos del perfil de calidad de la Facultad 8.

Palabras Clave: Calidad de Software, Cursos, Segundo perfil, Análisis y Diseño, Sistema da Gestión.

Índice de Contenidos

Índice de Figuras

Índice de Tablas


INTRODUCCIÓN


INTRODUCCIÓN

Los sistemas educativos de todo el mundo se enfrentan actualmente al desafío de utilizar las nuevas Tecnologías de la Informática y las Comunicaciones (TICs) para proveer a sus alumnos con las herramientas y conocimientos necesarios para el siglo XXI. La educación es el punto donde confluyen poderosas fuerzas políticas, tecnológicas y educativas en constante cambio, que tendrán un efecto significativo sobre la estructura de los sistemas educativos de todo el mundo en lo que resta del siglo. Muchos países están involucrados en iniciativas que intentan transformar el proceso de enseñanza-aprendizaje, preparando a los alumnos para formar parte de la sociedad de la información y la tecnología. (DANIEL 2004)

Una de las formas en que pueden prepararse los estudiantes, para que luego de graduados puedan cumplir con las demandas reales de la sociedad es la formación de segundos perfiles profesionales paralelamente a la formación de pregrado, mediante la cual se pretende nutrir al estudiante de la teoría y la práctica necesaria para enfrentarse a problemas reales de la sociedad. Muchos son los conceptos que pretenden definir un perfil profesional, a continuación se presentan algunos y se selecciona el rector en esta investigación, para que el lector pueda entender mejor lo que más adelante será comentado:

1. Conjunto de rasgos y capacidades que, certificadas apropiadamente por quien tiene la competencia jurídica para ello, permiten que alguien sea reconocido por la sociedad como ¨tal¨ profesional, pudiéndosele encomendar tareas para las que se le supone capacitado y competente. (HAWES and CORVALÁN 2004)
2. Conjunto de capacidades y competencias que identifican la formación de una persona para asumir en condiciones óptimas las responsabilidades propias del desarrollo de funciones y tareas de una determinada profesión. Los perfiles profesionales evolucionan y cambian según la demanda ocupacional y el mercado de trabajo, por tanto son dinámicos. (GARCÍA 2001)

Teniendo en cuenta las diferentes particularidades de cada uno de los conceptos anteriormente expuestos, se ha seleccionado como apoyo en el desarrollo de la presente investigación, el segundo concepto (*Generalitat Valenciana*), ya que éste detalla los principales rasgos y peculiaridades del “perfil profesional”, añadiendo una nueva característica, fundamental en el entorno en que se desarrolla la presente investigación, que es el dinamismo del concepto.

Debido al crecimiento en la actividad productiva de la UCI, ante la necesidad de personal calificado en los temas de calidad, gestión e ingeniería de software y de investigar y automatizar procesos para el control de calidad y dar seguimiento al

proceso de desarrollo de software surge el segundo perfil de calidad de software. (PÉREZ *et al.* 2008)

El diseño de este segundo perfil se realizó teniendo en cuenta la experiencia nacional e internacional y la rica experiencia pedagógica cubana. En este diseño se tuvieron en cuenta los siguientes principios: (PÉREZ *et al.* 2008)

- El empleo de métodos de aprendizaje como: el aprendizaje problémico, justo a tiempo, por error y el auto aprendizaje.
- La integración de diferentes asignaturas permitiendo el aprendizaje simultáneo de distintos conocimientos.
- El empleo intensivo de la teleformación y la variante semipresencial, aumentando esta última a medida que el estudiante transcurre por los años superiores.
- La aplicación práctica de los elementos estudiados en los proyectos productivos, conformando un proceso académico productivo.
- La actualización constante de los contenidos, teniendo en cuenta los cambios tecnológicos, así como los avances en el campo de la ciencia pedagógica.

Se decidió que el objetivo del perfil sea dotar a los estudiantes de conceptos, métodos, técnicas y herramientas necesarias para planificar, implementar y ejecutar programas de calidad y mejoramiento de procesos de manera estructurada en los proyectos de sus facultades, logrando una visión coherente de los modelos de calidad más destacados mundialmente en la industria de software. (PÉREZ *et al.* 2008)

La Universidad de Ciencias Informáticas (UCI) a lo largo de sus períodos lectivos brinda la posibilidad de graduar al estudiante con un determinado segundo perfil según la facultad en la que este se encuentre, el perfil de calidad es, a diferencia de los demás, común para todas las facultades.

La capacitación que se realiza en la Facultad 8 con respecto a los cursos optativos para adquirir el perfil de calidad contempla varias actividades como:

- Convocar los cursos que se ofertarán y atender las solicitudes de matrícula de los estudiantes.
- Planificar un horario específico de impartición de los cursos de manera que no coincida con las demás actividades docentes.
- Brindar la posibilidad a los estudiantes de presentarse periódicamente a realizar exámenes de suficiencia.
- Impartir los cursos.
- Acreditar los cursos culminados en la Secretaría Docente.

- Mantener organizada la documentación relacionada con los cursos del perfil (clases de los mismos, programa de la asignatura, controles de asistencias, notas parciales y finales de los estudiantes).
- Brindar reportes de información a cualquier interesado.

En la actualidad resulta muy difícil llevar un control de qué estudiantes están interesados en el perfil de calidad de la Facultad 8 y de qué cursos han culminado los mismos, por tal motivo no se ofertan quizás los cursos más necesarios. Todo este proceso se realiza de manera manual, implicando de este modo gasto de mano de obra, empleo de tiempo y tardanza en la actualización y manipulación de la información referente a los cursos de calidad.

No existe conocimiento en la Facultad, por parte de los estudiantes, sobre cuáles son los cursos que conforman el perfil de calidad, ni de qué contenidos se imparten en los mismos, tampoco se conoce con claridad el momento en que el estudiante posee el perfil básico.

Resulta muy difícil elaborar un reporte relacionado con el tema del perfil de calidad en la Facultad 8, pues hay que acudir a la información contenida en los documentos y muchas veces esto toma tiempo y provoca demoras en la generación de la información según las necesidades del interesado.

Por otra parte, a pesar de contarse en la UCI con un sistema para la gestión académica (Akademos)(UCI 2005) el mismo tiene un alcance general, pues gestiona todas las asignaturas que se imparten en pregrado, pero no brinda todos los reportes necesarios para tratar el tema del perfil de calidad en la facultad, algunos de estos reportes son:

- Listado (con la suma total) de los estudiantes de la facultad 8 que han cursado el curso “X”.
- Listado (con la suma total) de los estudiantes que han cursado cierto grupo de cursos (seleccionándose en este caso todos los cursos del perfil de calidad que se quieran).

Todas estas limitaciones, problemas y necesidades que dieron origen a esta investigación afectan a todas las personas implicadas en el proceso, provocando que el trabajo sea difícil de controlar íntegramente.

Por todo lo expuesto anteriormente **el problema** a resolver quedaría definido en la siguiente interrogante: ¿Cómo contribuir a la correcta organización de las actividades relacionadas con la formación de especialistas de calidad en la facultad 8?

INTRODUCCIÓN

Este problema se enmarca en el siguiente ***objeto de estudio:*** proceso de análisis y diseño de aplicaciones para la gestión de la calidad.

El **objetivo general** de esta investigación es***:*** proponer el análisis y diseño de un sistema informático que modele el proceso relacionado con la gestión de los cursos del perfil de calidad en la facultad 8 de la Universidad de las Ciencias Informáticas.

El **campo de acción** está enfocado en el análisis y diseño de aplicaciones web para la gestión de cursos impartidos durante la formación de especialistas de calidad en la facultad 8.

Se presenta como **idea a defender:** Si se realiza el análisis y diseño de un sistema informático que modele el proceso relacionado con la gestión de los cursos del perfil de calidad en la Facultad 8 de la Universidad de las Ciencias Informáticas, se contribuirá a la correcta automatización de las actividades relacionadas con la gestión de dichos cursos, el completamiento de estos por parte de los estudiantes interesados y las matrículas a los mismos.

Para desarrollar el objetivo general se plantean los siguientes **objetivos específicos:**

1. Contribuir a la automatización de las actividades relacionadas con la gestión de los cursos del perfil de calidad en la facultad 8.
2. Realizar un levantamiento de información con alta calidad relacionado con la gestión de los cursos del perfil de calidad en la facultad 8.
3. Brindar análisis y diseño robusto para el módulo de Capacitación del sistema de Gestión de la Calidad de la Facultad 8.
4. Contribuir a la eficiencia en la gestión de los cursos del perfil de calidad de la facultad 8, así como en la generación de reportes relacionados con este tema.
5. Crear un documento que recoja todo el proceso investigativo del desarrollo del sistema informático.

Entre las **tareas a desarrollar** para darle cumplimiento a los objetivos planteados tenemos las siguientes:

- Estructurar el documento tesis.
- Elaborar el diseño teórico de la investigación.
    - Definir situación problémica, problema, objetivos, novedad científica, aportes teóricos y prácticos, impacto social.
- Revisar el estado del arte.
    - Revisión bibliográfica del tema.
    - Existencia de otras soluciones similares.

- Elaborar la propuesta de solución.
    - Realizar el análisis del Módulo Capacitación del sistema para la Gestión de la Calidad en la Facultad 8.
    - Realizar el diseño del Módulo Capacitación del sistema para la Gestión de la Calidad en la Facultad 8.

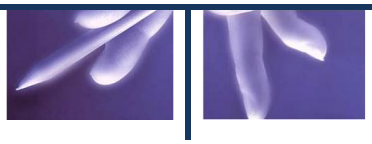
El documento se estructura en 3 capítulos fundamentales, a continuación se muestra un breve resumen de cada uno:

Capítulo I "Fundamentación Teórica": Aborda el estudio del arte del tema relacionado con la investigación que se lleva a cabo. Se destacan además las tecnologías y tendencias actuales en las cuales se apoya la propuesta del sistema informático, así como las distintas metodologías de desarrollo de software. El desarrollo de aplicaciones Web y su diseño es otro de los temas tratados en este capítulo y por último las principales herramientas de modelado de sistemas.

Capítulo II "Características del sistema": En este capítulo se realizó la descripción de las principales actividades relacionadas a la capacitación de los estudiantes de pregrado de la UCI mediante la especificación y modelación de los casos de uso del negocio y la descripción de los actores, trabajadores y entidades que intervienen en el mismo, incluyendo las reglas de negocio que deben tenerse en cuenta. Luego de este profundo estudio se crean las condiciones propicias para la identificación de los requerimientos funcionales y no funcionales validados en casos de uso de sistema.

Capítulo III "Análisis y diseño del sistema": En el presente capítulo se presentó una propuesta de sistema de gestión para la calidad de la facultad 8 del negocio modelado anteriormente. Se hizo una descripción detallada de los casos de uso del sistema utilizando para ello un lenguaje común y fácil de entender.

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA



1.1 Introducción.

En el presente capítulo se realiza un análisis de los aspectos teóricos, que fueron necesarios para la concepción de este trabajo, enfocados en cuatro perspectivas esencialmente:

- Sistemas de Gestión para la Calidad de la Facultad 8.
- La Ingeniería de Requerimientos.
- Metodologías de desarrollo de software.
- Herramientas de modelado de sistemas.

1.1.1 Sistemas Informáticos para la gestión del Conocimiento.

A lo largo de su historia el hombre ha identificado al Conocimiento como fuente generadora y a la vez como resultado de sus inventos y descubrimientos, pero en la actualidad varios autores plantean que el cambio fundamental consiste en la transición del paradigma de la Sociedad Industrial al paradigma de la Sociedad del Conocimiento; en la cual, este se concibe como el activo más importante en las organizaciones y también como producto y fuente generadora de ventajas competitivas, innovación, desarrollo e ingresos. Para ello, es necesario gestionarlo de modo consciente y planificado, potenciar su creación, transferencia, conservación y reutilización contextualizada, elevando así la capacidad creativa e innovadora del individuo en el colectivo. Todo ello, implica la toma de decisiones acertada sobre procesos de búsqueda de nuevas formas para la generación, captura, asimilación, difusión y transferencia de ese conocimiento. (GALE 2008)

La toma de decisiones en las organizaciones requiere de herramientas adecuadas para aplicar a los procesos de Gestión del Conocimiento (GC). (GALE 2008)

1.1.2 Sistemas Informáticos para la gestión de la calidad.

La calidad de los sistemas informáticos se ha convertido hoy en día en uno de los principales objetivos estratégicos de las organizaciones debido a que, cada vez más, su supervivencia depende de los sistemas informáticos para su buen funcionamiento. En la evolución experimentada por la calidad en esta área se ha pasado de un tratamiento centrado fundamentalmente en la inspección y detección de errores en los programas, a una aproximación más sistemática, dada la importancia que ha adquirido la calidad en la ingeniería de sistemas y en la ingeniería del software. En los últimos años, se han publicado diversos estándares y modelos que exponen los principios que

se deben seguir para la mejora de la calidad de productos y procesos software. Esta obra presenta los conceptos fundamentales relacionados con la calidad de los sistemas informáticos, exponiendo los aspectos más significativos relacionados con la calidad de productos y procesos software: normas ISO 90003, ISO 9126, ISO 15504, modelos como CMM, CMMI, PSP, TSP. Además, trata aspectos muy importantes para conseguir sistemas de información de calidad, como pueden ser las métricas de software, la calidad de la información o la gestión del conocimiento. A lo largo de esta obra se ha combinado el rigor científico con la experiencia práctica, proporcionando una panorámica actual y completa sobre la problemática asociada a la calidad de los sistemas informáticos. (MAGALLAN 2004)

La gestión de la calidad reduce los costes de producción de software. Cuanto antes se detecte un error en el software, menor será el coste del producto por el impacto que se ha de evitar por hecho de no haberse producido. Se puede pensar que un error cometido en etapas tempranas del ciclo de vida, por ejemplo una mala interpretación de alguna de las especificaciones que sea detectada al final del ciclo, ocasionará gran cantidad de trabajo perdido en las etapas siguientes con el consiguiente coste de resolución y de pérdida de tiempo, de especial impacto en productos cuya valía depende del tiempo de llegada al mercado. (MCLENNAN 2007)

Si se analiza el coste efectivo de la gestión de calidad se debe a cuatro factores: (MCLENNAN 2007)

1. Prevención: Se incurre en este tipo de costes cuando llevamos a cabo acciones para anticiparse a eventos antes de que ocurran. so de herramientas y técnicas avanzadas, empleo del personal cualificado más adecuado, etc.
2. Inspecciones: Se incurre en ellos cuando se mide, evalúa o inspeccionan productos, incluidos los intermedios, para ver el nivel de cumplimiento de especificaciones o seguimiento de normas, etc.
3. Internos: Son aquellos que se originan al subsanar las deficiencias detectadas antes de que el producto se haya entregado
4. Externos: Son aquellos que se originan cuando el defecto se detecta una vez entregado el producto final.

1.1.3 Técnicas de recopilación de la información.

Las técnicas de recopilación de la información, son muy importantes en la actualidad pues ayudan al analista a recopilar toda la información referente a una determinada institución y sirven de complemento para determinar las funcionalidades que implicará un sistema. (HERNÁNDEZ 1997)

Dentro de las diferentes técnicas para la Ingeniería de Requisitos (IR), se encuentran; Entrevistas y Cuestionarios, Lluvia de Ideas (Brainstorm), Revisión de documentos, Observación, entre otras. En la práctica, la técnica más apropiada para cada actividad dependerá del proyecto que esté desarrollándose. (HERNÁNDEZ 1997)

- ✓ **Lluvia de Ideas (Brainstorm).**

A esta técnica se le conoce también como tormenta de ideas y permite la libre expresión de las ideas de los participantes sin restricciones, con el propósito de producir el mayor número de datos, opiniones o soluciones sobre algún tema, pues cuantas más ideas se producen, mejores resultados se conseguirán: “la cantidad produce la calidad”; además, la producción de ideas en grupos puede ser más efectiva que la individual.

Esta técnica fue creada en el año 1941 por Alex Osborne, sirve para generar muchas ideas en un grupo. Requiere la participación espontánea de todos. Con la utilización de la "Lluvia de ideas" se alcanzan nuevas ideas y soluciones creativas e innovadoras. El clima de participación y motivación generado por la "Brainstorm" asegura mayor calidad en las decisiones tomadas por el grupo, más compromiso con la actividad y un sentimiento de responsabilidad compartido por todos. Se puede aplicar en cualquier etapa de un proceso de solución de problemas. Es fundamental para la identificación y selección de las preguntas que serán tratadas en la generación de posibles soluciones. Es muy útil cuando se desea la participación de todo el grupo.

Las etapas básicas de una sesión de "Lluvia de ideas" son las siguientes:

1. Introducción.
2. Generación de ideas.
3. Revisión de las tarjetas expuestas en el panel.
4. Análisis y selección.
5. Ordenando las ideas.

La lluvia de ideas puede responder a una estructura o no. Cuando la lluvia de ideas es desestructurada, cada persona presenta una idea a medida que se le ocurre. Este método funciona bien si los participantes son extrovertidos y se sienten cómodos entre ellos. Cuando la lluvia de ideas es estructurada, cada una de las personas aporta una idea por turno [una persona puede pasar si no tiene una idea en ese momento]. La lluvia de ideas estructurada funciona bien cuando la gente no se conoce entre sí y no es tan extrovertida: la estructura le brinda a todos una oportunidad para hablar.

El equipo en una lluvia de ideas debe estar formado por:

- El director: es la figura principal y el encargado de dirigir la sesión y debe ser un experto en pensamiento creador, pues su función es formular claramente el

problema y que todos se familiaricen con él, además estimular las ideas y siempre tratando que se cumplan las normas establecidas y no las críticas. Debe hacer que todos participen y aporten ideas, es quien concede la palabra y da por finalizada la sesión. Posteriormente, clasificará las ideas de la lista que le proporciona el secretario.

- El secretario: registra por escrito las ideas según van surgiendo. Las enumera, las reproduce fielmente, las redacta y se asegura que todos están de acuerdo con lo escrito. Por último realizará una lista de ideas.
- Los participantes: pueden ser habituales o invitados; cualquier involucrado en el proyecto entra en esta categoría. Su función es producir ideas. Conviene que entre ellos no haya diferencias jerárquicas.

Las personas que componen el equipo deben estar motivadas para solucionar el problema, y contar con un ambiente que propicie la participación de todos. Todas las ideas en principio deben tener el mismo valor, pues cualquiera de ellas puede ser la clave para la solución. Durante la celebración no deben asistir espectadores.

Entre las fases de aplicación de esta técnica, se encuentran:

- Descubrir hechos: es donde se determina el problema, delimitándolo, precisándolo y clarificándolo. Luego se plantea el problema, recogiendo las experiencias que se poseen o consultando documentación.
- Producir ideas: (es la fase de tormenta de ideas propiamente dicha).
- Descubrir soluciones: se elabora una lista preliminar de ideas, para hacer una selección de las más interesantes, la misma se realiza desechando las ideas que no tienen valor y se estudia si son válidas las que se consideran interesantes. Pueden realizarlo los mismos miembros del equipo o crear otros para esta tarea; la clasificación debe hacerse por categorías. Se presentan las ideas de forma atractiva, haciendo uso de soportes visuales.

✓ **Entrevistas y Cuestionarios.**

Esta técnica se emplea para reunir información proveniente de personas o de grupos.

La entrevista es una conversación, generalmente oral, entre dos o más personas dependiendo de si sea personal o grupal, de los cuáles, una parte es el entrevistador y la otra el entrevistado. (PLOMÉ 2000)

Durante la entrevista, el analista es quien conversa con el encuestado; el cuestionario consiste en una serie de preguntas relacionadas con varios aspectos de un sistema. Por lo común, los encuestados son usuarios de los sistemas existentes o usuarios en

potencia del sistema propuesto. En algunos casos, son gerentes o empleados que proporcionan datos para el sistema propuesto o que serán afectados por él.

Las preguntas que deben realizarse en esta técnica, deben ser preguntas de alto nivel y abstractas que pueden realizarse al inicio del proyecto para obtener información sobre aspectos globales del problema del usuario y soluciones potenciales.

Toda técnica de investigación, se sabe que no son más que instrumentos y, como todos, tienen ventajas y limitaciones que son importantes tener en cuenta. Sabemos que ninguna técnica es buena o mala por sí misma, sino en función del problema de estudio y de los objetivos que se pretenden alcanzar. (PLOMÉ 2000)

Lo importante de la entrevista es que permita ver el punto de vista de los actores. Estas pueden ser:

1. Entrevista no dirigida, abierta.
2. Entrevista dirigida, cerrada.
3. Entrevista semiestructurada o semiestandarizada.

En la presente investigación, se utilizará la entrevista dirigida, cerrada, estructurada, guiada, controlada, estandarizada. Éstas siguen un procedimiento fijado de antemano por un cuestionario o guía de la entrevista, es decir, por una serie de preguntas que el investigador prepara de antemano. Algunos autores plantean a este tipo de entrevistas y al cuestionario como prácticamente la misma cosa, solamente que se habla de entrevista en las situaciones en las que el cuestionario se aplica por el entrevistador que leerá las preguntas a quién responda. Las preguntas son presentadas exactamente como figuran en el cuestionario y en su mismo orden. No se permite al entrevistador introducir modificaciones. Las preguntas están, por lo general, cerradas, es decir, se le proporciona al entrevistado una serie de alternativas de respuesta donde debe seleccionar una u otras, ordenarlas, expresar su grado de acuerdo o desacuerdo. En este sentido, el cuestionario, como técnica está estrechamente ligado a la entrevista y no puede ser estudiado como algo aislado. (PLOMÉ 2000)

Tanto la entrevista como el cuestionario tienen como finalidad obtener información. Dentro de los tipos de cuestionarios que se encuentran, están:

1. Auto-administrado.
2. Por entrevista personal.
3. Por entrevista telefónica.
4. Auto-administrado y enviado por correo postal, electrónico o servicio de mensajería.

En este trabajo de diploma, el cuestionario a utilizar es “por entrevista personal”, donde el entrevistador aplica el cuestionario a los respondientes. Hace las preguntas y

anota las respuestas. Los encuestadores deben conocer a fondo el cuestionario, y no deben sesgar o influir en las respuestas. (PLOMÉ 2000)

1.2 Metodologías de desarrollo de software.

¿Qué metodología se debe usar para el desarrollo de un Software?

Todos en algún momento se han hecho esta pregunta cuando han tenido que desarrollar un software, pues como arquitectos de software, se debe tener un plano en que apoyarse. Todo desarrollo de software es riesgoso y difícil de controlar, pero si no se tiene en cuenta una metodología de por medio, se obtiene clientes insatisfechos con el resultado y desarrolladores aún más insatisfechos. Muchas veces se realiza el diseño de un software de manera rígida, con los requerimientos que el cliente solicitó, de forma tal que cuando el cliente en la etapa final solicita un cambio se hace muy difícil realizarlo, pues si se lleva a cabo, altera muchas cosas que no se habían previsto, y es justo éste, uno de los factores que ocasiona un atraso en el proyecto y por tanto la incomodidad del desarrollador por no cumplir con el cambio solicitado y el malestar por parte del cliente por no tomar en cuenta su pedido. Obviamente para evitar estos incidentes se debe llegar a un acuerdo formal con el cliente al inicio del proyecto, de tal manera que cada cambio o modificación no perjudique el desarrollo del mismo. (SANCHEZ 2004)

En la actualidad existe gran cantidad de metodologías orientadas al proceso de desarrollo de software, entre las más utilizadas y conocidas se pueden mencionar las siguientes: RUP (Proceso Unificado Racional), XP (Programación Extrema) y DSDM (Método de Desarrollo de Sistemas Dinámicos), las cuales se caracterizan a continuación.

1.2.1 El Proceso Unificado Racional (RUP).

RUP es una metodología basada íntegramente en el Lenguaje Unificado de Modelado (UML) la cual sirve de soporte a la misma. Esta metodología define el quién (trabajadores), cómo (actividades), cuándo (flujos de trabajo) y qué (artefactos) debe hacerse en un proyecto.

Presenta como características esenciales:

El Proceso Unificado de Rational es un proceso iterativo, que propone una comprensión incremental del problema a través de refinamientos sucesivos y un crecimiento incremental de una solución efectiva a través de varias versiones. Como parte del enfoque iterativo se encuentra la flexibilidad para acomodarse a nuevos requisitos o a cambios tácticos en los objetivos del negocio. El desarrollo bajo el

Proceso Unificado está centrado en la arquitectura. Este proceso se centra en establecer al principio una arquitectura software que guía el desarrollo del sistema. Con ello se facilita el desarrollo en paralelo, se minimiza la repetición de trabajos y se incrementa la probabilidad de reutilización de componentes y el mantenimiento posterior del sistema. Este diseño arquitectónico sirve como una sólida base sobre la cual se puede planificar y manejar el desarrollo de software basado en componentes. En RUP las actividades de desarrollo están dirigidas por los casos de uso. El Proceso Unificado pone un gran énfasis en la construcción de sistemas basada en una amplia comprensión de cómo se utilizará el sistema que se entregue. Las nociones de los casos de uso y los escenarios se utilizan para guiar el flujo de procesos desde la captura de los requisitos hasta las pruebas, y para proporcionar caminos que se pueden reproducir durante el desarrollo del sistema. (COLTELL 2003)

Divide el proceso de desarrollo en 4 fases:

Inicio: Cuya finalidad principal es alcanzar un acuerdo entre todos los interesados respecto a los objetivos del ciclo vital para el proyecto.

Elaboración: Cuyo principal propósito es el establecimiento de una línea base para la arquitectura del sistema y proporcionar una base estable para el grueso del diseño y del esfuerzo de implementación en la fase siguiente.

Construcción: Cuya finalidad principal es completar el desarrollo del sistema basado en la arquitectura de línea base.

Transición: Cuya finalidad principal es garantizar que el software está listo para entregarlo a los usuarios.

Dentro de los nueve flujos de trabajos de los que consta RUP, se encuentran:

- Disciplinas de desarrollo:

1. Modelado del negocio: Describe la estructura y la dinámica de la organización.
2. Requisitos: Describe el método basado en casos de uso para extraer los requisitos.
3. Análisis y diseño: Describe las diferentes vistas arquitectónicas.
4. Implementación: Tiene en cuenta el desarrollo de software, la prueba de unidades y la integración.
5. Pruebas: Describe los casos de pruebas, los procedimientos y las métricas para evaluación de defectos.
6. Despliegue: Cubre la configuración del sistema entregable.

- Disciplina de soporte:

7. Gestión de configuraciones: Controla los cambios y mantiene la integridad de los artefactos de un proyecto.
8. Gestión del Proyecto: Describe varias estrategias de trabajo en un proceso iterativo.

9. Entorno: Cubre la infraestructura necesaria para desarrollar un sistema.

El proceso puede ser descrito en dos dimensiones o ejes:

1. El eje horizontal demuestra los aspectos dinámicos del proceso, representa el tiempo y se expresa en términos de fases, de iteraciones, y la finalización de las fases.
2. El eje vertical representa el aspecto estático del ciclo de vida de RUP: las actividades, los flujos de trabajo, los artefactos y roles. (Ver ilustración 1)

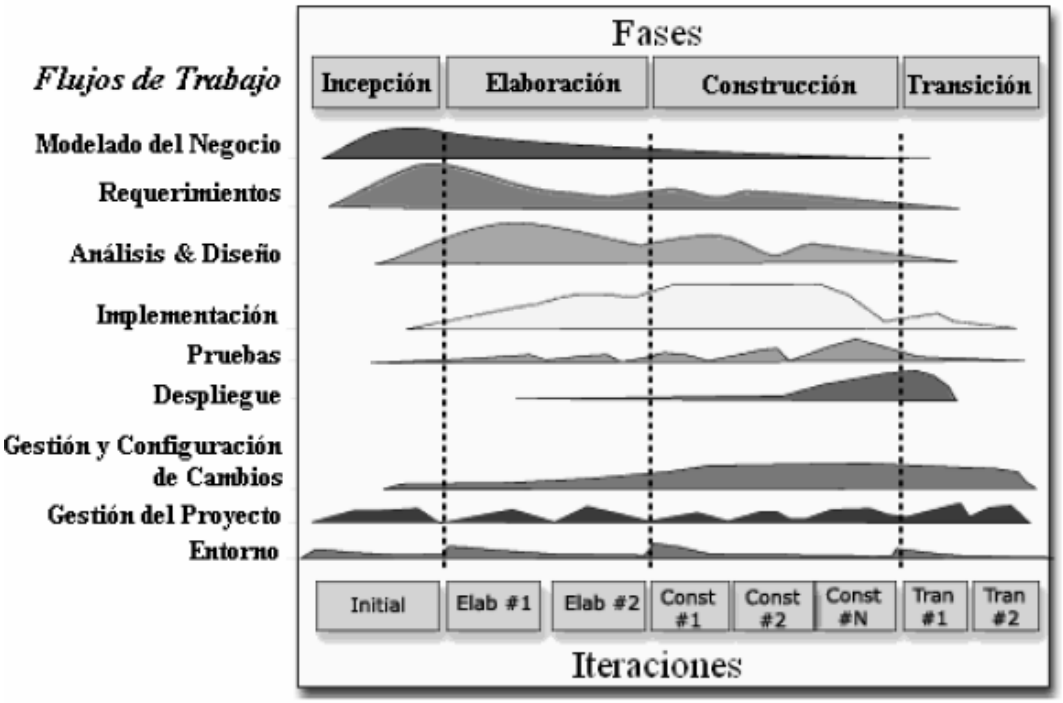


Figura 1: Flujos de trabajo, fases e iteraciones de RUP.

Los elementos del RUP son:

- Actividades: Son los procesos que se llegan a determinar en cada iteración.
- Trabajadores: Vienen hacer las personas o entes involucrados en cada proceso.
- Artefactos: Un artefacto puede ser un documento, un modelo, o un elemento de modelo.

Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software. (SANCHEZ 2004)

1.2.2 Programación Extrema (XP).

Es una metodología reciente, desarrollada por Kent Beck en 1996. La filosofía de XP es satisfacer por completo las necesidades del cliente, por eso lo integra como una parte más del equipo de desarrollo. Está diseñada para el desarrollo de aplicaciones que requieran un grupo de programadores pequeño, dónde la comunicación sea más factible que en grupos de desarrollo grandes. La comunicación es un punto importante y debe realizarse entre los programadores, los jefes de proyecto y los clientes. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. (LETELIER and PENADÉS 2007)

XP es la más destacada de los procesos ágiles de desarrollo de software. Al igual que éstos, la programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. Los defensores de esta metodología consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos. Creen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos.

La programación extrema se basa en la simplicidad, la comunicación y el reciclado continuo de código, para algunos no es más que aplicar una pura lógica. (SOLÍS 2003)

Los objetivos de XP son muy simples:

1. La satisfacción del cliente: Esta metodología trata de dar al cliente el software que él necesita y cuando lo necesita. Por tanto, debemos responder muy rápido a las necesidades del cliente, incluso cuando los cambios sean al final de ciclo de la programación.
2. Potenciar al máximo el trabajo en grupo: Tanto los jefes de proyecto, los clientes y desarrolladores, son parte del equipo y están involucrados en el desarrollo del software.

Esta metodología define cuatro variables para proyectos de software, estas son: coste, tiempo, calidad y ámbito.

XP está diseñada para grupos de pequeños programadores. Las metodologías tradicionales imponen un proceso disciplinado para tratar de hacer el trabajo predecible, eficiente y planificado. Estos métodos están orientados a documentos y se vuelven demasiado burocráticas e ineficaces. XP es más liviana y ágil y están orientadas más a las personas que a los procesos. (SOLÍS 2003)

La Programación Extrema supone: (SOLÍS 2003)

- Las personas son claves en los procesos de desarrollo.
- Los programadores son profesionales, no necesitan supervisión.
- Los procesos se aceptan y se acuerdan, no se imponen.
- Desarrolladores y gerentes comparten el liderazgo del proyecto.
- El trabajo de los desarrolladores con las personas que conocen el negocio es regular, no puntual.

El ciclo de vida ideal de XP consiste de seis fases: Exploración, Planificación de la Entrega (Release), Iteraciones, Producción, Mantenimiento y Muerte del Proyecto, las mismas se explican brevemente a continuación. (LETELIER and PENADÉS 2007)

Fase I Exploración: En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.

Fase II Planificación de la entrega: En esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días. Por otra parte, el equipo de desarrollo mantiene un registro de la velocidad de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración.

Fase III Iteraciones: Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fuercen la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para entrar en producción.

Fase IV Producción: La fase de producción requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase. Es posible que se rebaje el tiempo que toma cada iteración, de tres a una semana. Las ideas que han sido propuestas y las sugerencias son documentadas para su posterior implementación (por ejemplo, durante la fase de mantenimiento).

Fase V Mantenimiento: Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para realizar esto se requiere de tareas de soporte para el cliente. De esta forma, la velocidad de desarrollo puede bajar después de la puesta del sistema en producción. La fase de mantenimiento puede requerir nuevo personal dentro del equipo y cambios en su estructura.

Fase VI Muerte del proyecto: Es cuando el cliente no tiene más historias para ser incluidas en el sistema. Esto requiere que se satisfagan las necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema. Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura. La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo.

Esta metodología tiene varias características o ventajas las cuales se mencionan a continuación: (FOWLER 2003)

1. Comunicación: Los programadores están en constante comunicación con los clientes para satisfacer sus requisitos y responder rápidamente a los cambios de los mismos. Muchos problemas que surgen en los proyectos se deben a que después de concretar los requisitos que debe cumplir el programa no hay una revisión de los mismos, pudiendo dejar olvidados puntos importantes.
2. Simplicidad: Codificación y diseños simples y claros. Muchos diseños son tan complicados que cuando se quieren ampliar resulta imposible hacerlo y se tienen que desechar y partir de cero.
3. Realimentación (Feedback): Mediante la realimentación se ofrece al cliente la posibilidad de conseguir un sistema apto a sus necesidades ya que se le va mostrando el proyecto a tiempo para poder ser cambiado y poder retroceder a una fase anterior para rediseñarlo a su gusto.
4. Coraje: Se debe tener coraje o valentía para cumplir los tres puntos anteriores; se requiere valor para comunicarse con el cliente y enfatizar algunos puntos, a pesar de que esto pueda dar sensación de ignorancia por parte del

programador, hay que tener coraje para mantener un diseño simple y no optar por el camino más fácil, se debe confiar en que la realimentación sea efectiva.

1.2.3 Método de Desarrollo de Sistemas Dinámicos (DSDM).

Esta metodología empezó en Gran Bretaña en 1994 como un consorcio de compañías del Reino Unido. Habiendo empezado con 17 fundadores, hoy cuenta ya con más de mil miembros y ha crecido fuera de sus raíces británicas.

Tiene además principios subyacentes que incluyen una interacción activa del usuario, entregas frecuentes, equipos autorizados, pruebas a lo largo del ciclo. Como otros métodos ágiles usan ciclos de plazos cortos de entre dos y seis semanas. Hay un énfasis en la alta calidad y adaptabilidad hacia requisitos cambiantes. Es notable por tener mucha de la infraestructura de las metodologías tradicionales más maduras, al mismo tiempo que sigue los principios de los métodos ágiles. (FOWLER 2003)

DSDM define el marco para desarrollar un proceso de producción de software. Se caracteriza por ser un proceso iterativo e incremental y el equipo de desarrollo y el usuario trabajan juntos. Propone cinco fases: estudio viabilidad, estudio del negocio, modelado funcional, diseño y construcción y finalmente implementación. Las tres últimas son iterativas, además de existir realimentación a todas las fases.

1.2.4 Selección de la Metodología a utilizar.

Una vez analizadas algunas de las características fundamentales de las metodologías RUP, XP y DSDM, se puede evidenciar la particularidad de poder ser empleadas para disímiles situaciones. A continuación se realiza una comparación entre el Proceso Unificado de Desarrollo (RUP), Programación Extrema (XP) y el Método de Desarrollo de Sistemas Dinámicos (DSDM), donde se demuestra cuál de estas es más factible y fácil de aplicar al sistema informático propuesto.

XP a pesar de ser muy útil para el desarrollo de software no se ajusta al proceso actual, puesto que es una metodología flexible en cuanto a requisitos cambiantes, característica que no se ajusta al proyecto que se está desarrollando, ya que tiene bien definidos y de forma estable los requerimientos. No produce una potente documentación del sistema y está dirigida a equipos pequeños o medianos, otra de las características esenciales que tiene esta metodología es que el usuario debe convertirse en un integrante más del grupo de desarrollo. Esta metodología está diseñada para grupos pequeños de desarrolladores, lo cual puede constituir un inconveniente ya que este grupo de desarrolladores no es pequeño. RUP es una metodología que se puede adaptar a cualquier proceso de desarrollo y es iterativo e

incremental, es decir, que sus cuatro fases de desarrollo se dividen en iteraciones, obteniendo en cada una de estas un incremento del producto que se desarrolla añadiendo o mejorando las funcionalidades de la aplicación en desarrollo. Permite guiar a todos los participantes, dígase clientes, usuarios y desarrolladores, y está ampliamente disponible de forma que todos los interesados puedan comprender su papel en el desarrollo en el que se encuentra implicado. Es una metodología que se recomienda para cualquier tipo de proyecto. DSDM define el marco para desarrollar un proceso de producción de software. Se caracteriza por ser un proceso iterativo e incremental donde el equipo de desarrollo y el usuario trabajan juntos. Tiene además principios subyacentes que incluyen una interacción activa del usuario, entregas frecuentes, equipos autorizados, pruebas a lo largo del ciclo. Como otros métodos ágiles usan ciclos de plazos cortos de entre dos y seis semanas.

Teniendo en cuenta las particularidades de cada una de las metodologías anteriormente explicadas, se ha seleccionado como apoyo en el desarrollo de la aplicación propuesta la metodología RUP, ya que permite mayor productividad en equipo y la realización de mejores prácticas de software a través de plantillas y herramientas que lo guían en todas las actividades de desarrollo crítico del software.

Además RUP es una recopilación de prácticas de ingeniería de software que se están mejorando continuamente de forma regular para reflejar los cambios en las prácticas de la industria. Le ofrece al equipo de desarrollo de software un glosario de terminología y una enciclopedia de conocimiento que le ayuda a comunicar sus necesidades de forma eficaz. Además proporciona una buena base de arquitectura y gran cantidad de material con las que construir una definición de proceso, lo que permite configurar y ampliar dicha base como desee. Esto ahorrará mucho tiempo y esfuerzo que de otra manera tendría que aplicar para crear dicha definición de proceso desde cero.

1.3 Herramientas de Modelado de Sistemas.

Herramientas CASE (Ingeniería de Software Asistida por Computadora).

Realmente son las herramientas CASE el mejor método para el análisis y soluciones de software, las cuales han mejorado los aspectos claves en el desarrollo de los sistemas de información, estas herramientas han sido creadas para la automatización de procesos de análisis, diseño e implementación, brindando un sin número de componentes que hacen que los proyectos sean cada día más eficientes para los usuarios finales. Desde que se crearon éstas herramientas en 1984 hasta la

actualidad, las CASE cuentan con una credibilidad y exactitud que tienen un reconocimiento universal, siendo usadas por cualquier analista y / o programador que busca un resultado óptimo y eficaz, para cada uno de sus procesos. (VALLE 1997)

Se puede definir a una herramienta CASE como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software. (VALLE 1997)

Ventajas de la utilización de las herramientas CASE:

- Permiten el incremento en la velocidad de desarrollo de los sistemas.
- Permiten a los analistas tener más tiempo para el análisis y diseño y minimizar el tiempo para codificar y probar.
- En las etapas del proceso de desarrollo de software permiten:
    - Automatizar el dibujo de diagramas.
    - Ayudar en la documentación del sistema.
    - Ayudar en la creación de relaciones en la base de datos.
    - Generar estructuras de código.
    - Aumentan la productividad. Esto se consigue a través de la automatización de determinadas tareas, como la generación de código y la reutilización de objetos o módulos.

1.3.1 Visual Paradigm (VP).

El paradigma visual para UML Enterprise Edition es una plataforma de modelado de sistema de apoyo diseñado para arquitectos, desarrolladores, diseñadores, analistas de procesos de negocio, y modeladores de datos a fin de acelerar todo el modelo de código para el complejo proceso de desplegar las aplicaciones empresariales a través de los mejores de la raza y la galardonada tecnología de modelado visual que facilita el modelado de procesos de negocio (BPMN) visualización de UML en la última notación UML 2.1 en 13 diferentes tipos de diagramas, así como los diagramas ER, diagramas Requisito, CRC Diagramas , el análisis textual y modelado de datos (DFD y ERD). (WONG 2004)

Visual paradigma presenta varias funcionalidades tales como: (WONG 2004)

1. Potente y fácil de usar GUI: Ahora los desarrolladores pueden crear diagramas mucho más rápido que cualquier herramienta en el mercado a través de VP-UML más intuitiva de recursos centrados en la ubicación y el ratón sensible GUI.
2. Motores de generación de código de gran alcance: El avance y retroceso código motores se derivan automáticamente a partir de modelos de código o los

códigos a su vez en los modelos en cuestión de segundos, y es de fácil acceso para su elección, de 10 lenguajes de programación como Java, C + +,. NET, PHP y XML.

3. Programación de bases de datos: Adelante y la ingeniería inversa de la clase a diagramas (ERD) Diagramas Entidad-Relación y de la División de Respuesta de Emergencia a la base de datos de generación y viceversa se puede hacer el diagrama y visual, y es totalmente compatible con Hibernate. Visualizar y automatizar todo el mapeo objeto-relacional (ORM) es ahora transparente a los desarrolladores de software.
4. Excelente interoperabilidad y de apoyo más amplia IDEs: Ya sea que usted está utilizando VP-UML como una aplicación independiente o integrarse en uno de los principales IDEs Visual Studio, Eclipse, Borland JBuilder, NetBeans / Sun ONE, IntelliJ IDEA, Oracle JDeveloper, BEA WebLogic Taller), que tienen la facultad de ingeniería de ida y vuelta para mantener el modelo y el código sincronizado. Furthermore, VP-UML EE facilitates import/export for XMI of versions 1.0, 1.2 and 2.1. Además, la VP-UML EE facilita la importación / exportación de XMI de versiones 1.0, 1.2 y 2.1. La Enterprise Edition también soporta Rational Rose Erwin modelo de datos y archivo de proyecto de la importación.
5. Equipo de Apoyo al Desarrollo: Servidor de trabajo en equipo que permite a los desarrolladores de software trabajar en los mismos proyectos en paralelo.

Visual Paradigm para UML es una de las herramientas UML CASE del mercado, considerada muy completa y fácil de usar, con soporte multiplataforma y que proporciona excelentes facilidades de interoperabilidad con otras aplicaciones. Fue creada para el ciclo vital completo del desarrollo de software que lo automatiza y acelera, permitiendo la captura de requisitos, análisis, diseño e implementación. Visual Paradigm-UML también proporciona características tales como generación del código, ingeniería reversa y generación de informes.

Tiene la capacidad de crear el esquema de clases a partir de una base de datos y crear la definición de base de datos a partir del esquema de clases. Permite invertir código fuente de programas, archivos ejecutables y binarios en modelos UML al instante, creando de manera simple toda la documentación.

Está diseñada para usuarios interesados en sistemas de software de gran escala con el uso del acercamiento orientado a objeto, además apoya los estándares más recientes de las notaciones de Java y de UML. Incorpora el soporte para trabajo en

equipo, que permite que varios desarrolladores trabajen a la vez en el mismo diagrama y vean en tiempo real los cambios hechos por sus compañeros.

1.3.2 Rational Rose Enterprise Edition.

Rational Rose Enterprise Edition es el producto más completo de la familia Rational Rose. Todos los productos Rational Rose incluyen soporte Unified Modeling Language (UML). Esta herramienta es la mejor elección para el ambiente de modelado que soporte la generación de código a partir de modelos en Ada, ANSI C++, C++, CORBA, Java/J2EE, Visual C++ y Visual Basic. Como todos los demás productos Rational Rose, proporciona un lenguaje común de modelado para el equipo que facilita la creación de software de calidad más rápidamente. (BUCHWALD 2007)

Dentro de las características adicionales incluidas se encuentran: (BUCHWALD 2007)

- Soporte para análisis de patrones ANSI C++, Rose J y Visual C++ basado en "Design Patterns: Elements of Reusable Object-Oriented Software".
- Característica de control por separado de componentes modelo que permite una administración más granular y el uso de modelos.
- Soporte de ingeniería Forward y/o reversa para algunos de los conceptos más comunes de Java 1.5.
- La generación de código Ada, ANSI C ++, C++, CORBA, Java y Visual Basic, con capacidad de sincronización modelo- código configurables.
- Capacidad de análisis de calidad de código.
- El Add-In para modelado Web provee visualización, modelado y las herramientas para desarrollar aplicaciones de Web.
- Modelado UML para trabajar en diseños de base de datos, con capacidad de representar la integración de los datos y los requerimientos de aplicación a través de diseños lógicos y físicos.
- Capacidad de crear definiciones de tipo de documento XML (DTD) para el uso en la aplicación.
- Integración con otras herramientas de desarrollo de Rational.
- Capacidad para integrarse con cualquier sistema de control de versiones SCC-compliant, incluyendo a Rational ClearCase.
- Publicación web y generación de informes para optimizar la comunicación dentro del equipo.

Sistemas Operativos y Plataformas de Hardware apropiadas: (BUCHWALD 2007)

- Windows 2000.

- Windows NT.
- Windows XP.

Es una herramienta para “modelado visual”, que forma parte de un conjunto más amplio de herramientas que juntas cubren todo el ciclo de vida del desarrollo de software. Permite completar una gran parte de las disciplinas (flujos fundamentales) del proceso unificado de Rational (RUP) e incluye un conjunto de herramientas de ingeniería inversa y generación de código que nivelan el camino hasta el producto final.

El Rational Rose es una herramienta CASE basada en UML que permite crear los diagramas que se van generando durante el proceso de ingeniería en el desarrollo del software. Es completamente compatible con la metodología RUP, brinda muchas facilidades en la generación de la documentación del software que se están desarrollando, además posee un gran número de estereotipos predefinidos que viabiliza el proceso de modelación del software. Es capaz de generar el código fuente de las clases definidas en el flujo de trabajo de diseño, pero tiene la limitación de que aún hay varios lenguajes de programación que no soporta o que sólo lo hace a medias. Rose se encuentra a la cabeza en cuanto al desarrollo del Unified Modeling Language (UML), que se ha convertido en la notación estandarizada empleada en Rational Rose para especificar, visualizar y construir desarrollos de software y sistemas.

1.3.3 Selección de la herramienta de modelado a utilizar.

Se decidió utilizar como herramienta CASE para el modelado de la aplicación el Visual Paradigm, debido fundamentalmente a que es multiplataforma, y que tiene licencia de uso libre, lo que permitirá transferir los modelos obtenidos al cliente, al finalizar el desarrollo del sistema. Esta herramienta es compatible con la plataforma a utilizar en el desarrollo del módulo “Capacitación” y tiene una gran ventaja con respecto al Rational Rose ya que la Universidad de las Ciencias Informáticas posee su licencia, además que el Rational Rose es propietario.

1.4 Desarrollo de Aplicaciones Web

Las aplicaciones Web han tenido un enorme auge en los últimos años debido a su relativa facilidad de implementación y de utilización sobre las aplicaciones de escritorio. A continuación se listan algunos lenguajes del lado del cliente, así como del

lado del servidor, exponiendo sus principales características para luego hacer una correcta selección para la propuesta de la investigación.

1.4.1 Lenguajes del lado del cliente.

1. **Java Script:**

Es una de las múltiples aplicaciones que han surgido para extender las capacidades del Lenguaje HTML. Es un lenguaje script orientado a documento. Nunca podrá hacer un programa, tan sólo podrá mejorar sus páginas *Web*. (Martinez, 2001)

Las normas para poder escribir cualquier código de Java Script se basan en 5 puntos básicos y que debemos cumplir siempre. Estas normas son las siguientes: (Martinez, 2001)

1. Todo el código (sentencias) está dentro de funciones.
2. Las funciones se desarrollan entre las etiquetas <script> y </script>.
3. Las etiquetas “<script>” deben colocarse entre las etiquetas <head> y </head>.
4. Las etiquetas “<title>” no pueden estar colocadas entre las de “<*script*>”.
5. La llamada a la función se hace a través de un evento de un elemento del documento.

2. **Lenguaje HTML (Lenguaje de formato de Documentos para Hipertexto:**

Fue desarrollado originalmente por Tim Berners-Lee mientras estaba en el CERN, y fue popularizado por el navegador Mosaic desarrollado en el NCSA. Durante los años 90 ha proliferado con el crecimiento explosivo de la Web. Durante este tiempo, el HTML se ha desarrollado de diferentes maneras. La WEB depende de que los autores de páginas Web y las compañías compartan las mismas convenciones de HTML. Esto ha motivado el trabajo colectivo en las especificaciones del HTML.

Cada versión de HTML ha intentado reflejar un consenso cada vez mayor entre los interlocutores de la industria, de modo que no se desperdicien las inversiones hechas por los proveedores de contenidos y que sus documentos no dejen de ser legibles a corto plazo.

El HTML 2.0 fue desarrollado bajo los auspicios de la Internet Engineering Task Force (IETF) para codificar lo que era la práctica común a finales de 1994. HTML+ (1993) y HTML 3.0 (1995) propusieron versiones mucho más ricas de HTML. A pesar de no haber logrado nunca el consenso en las discusiones sobre estándares, estos

borradores llevaron a la adopción de un número de nuevas características. Los esfuerzos del Grupo de Trabajo HTML del World Wide Web Consortium para codificar la práctica común en 1996 condujeron a HTML 3.2 (enero de 1997). El HTML 4 desarrolla el lenguaje HTML con mecanismos para hojas de estilo, ejecución de scripts, marcos, objetos incluidos, soporte mejorado para texto de derecha a izquierda y direcciones mezcladas, tablas más ricas y mejoras en formularios, ofreciendo mejoras de accesibilidad para personas con discapacidades. El HTML 4.01 es una revisión de HTML 4.0 que corrige errores e introduce algunos cambios desde la revisión anterior.

HTML es una implementación del standard SGML (Standard Generalized Markup Language), estándar internacional para la definición de texto electrónico independiente de dispositivos, sistemas y aplicaciones. Metalenguaje para definir lenguajes de diseño descriptivos; proporciona un medio de codificar documentos hipertexto cuyo destino sea el intercambio directo entre sistemas o aplicaciones. (MARTÍN 2005)

Algunas de las fundamentales características de este lenguaje son: (MARTÍN 2005)

- Permite crear lenguajes de codificación descriptivos.
- Define una estructura de documentos jerárquica, con elementos y componentes interconectados.
- Proporciona una especificación formal completa del documento.
- No tiene un conjunto implícito de convenciones de señalización. Soporta, por tanto, un conjunto flexible de juegos de etiquetas.
- Los documentos generados por él son legibles.

Sintaxis general: (MARTÍN 2005)

- Son válidos todos las caracteres incluidos en ISO 8859-1
- El formato es libre. El formato introducido en el fichero fuente (saltos de línea, líneas en blanco, etc.) es irrelevante para el formato final del documento.

Caracteres de significado especial: (MARTÍN 2005)

1. < Marca el comienzo de una etiqueta.
2. > Marca el final de una etiqueta.
3. & Marca el comienzo de una referencia a entidad.

Un documento HTML consta de las siguientes piezas: (MARTÍN 2005)

1. Identificación SGML
2. Una etiqueta <HTML>
3. Cabecera (iniciada por la etiqueta <HEAD> y cerrada por </HEAD>)
4. Cuerpo del documento (iniciada por la etiqueta BODY y cerrada por </BODY>)

5. Una etiqueta de fin de documento </HTML>

El lenguaje HTML proporciona a los autores las herramientas para:

- Publicar documentos en línea con encabezados, textos, tablas, listas, fotos.
- Obtener información en línea a través de vínculos de hipertexto, haciendo clic con el botón de un ratón.
- Diseñar formularios para realizar transacciones con servicios remotos, para buscar información, hacer reservas, pedir productos.
- Incluir hojas de cálculo, videoclips, sonidos, y otras aplicaciones directamente en sus documentos.

Selección:

Con las características antes expuestas, se seleccionan los dos lenguajes para su utilización, ya que HTML puede utilizar algunas funcionalidades de Java Script, para la realización de una aplicación con la calidad requerida y además para validar los datos correctamente.

1.4.2 Lenguajes del lado del Servidor.

Lenguaje PHP (Procesador de hipertexto):

Creado por una gran comunidad de personas, desarrollado originalmente en el año 1994 por Rasmus Lerdorf como un CGI escrito en C que permitía la interpretación de un número limitado de comandos. Este lenguaje fue denominado Personal Home Page Tools y adquirió relativo éxito gracias a que otras personas pidieron a Rasmus que les permitiese utilizar sus programas en sus propias páginas. Dada la aceptación del primer PHP y de manera adicional, su creador diseñó un sistema para procesar formularios al que le atribuyó el nombre de FI (Form Interpreter) y el conjunto de estas dos herramientas, sería la primera versión compacta del lenguaje: PHP/FI. Actualmente PHP se encuentra en su versión 4, que utiliza el motor Zend, desarrollado con mayor meditación para cubrir las necesidades actuales y solucionar algunos inconvenientes de la anterior versión. Algunas mejoras de esta nueva versión son su rapidez, gracias a que primero se compila y luego se ejecuta, mientras que antes se ejecutaba mientras se interpretaba el código, su mayor independencia del servidor web creando versiones de PHP nativas para más plataformas y un API más elaborado y con más funciones. (ALVAREZ, MIGUEL ANGEL 2001)

“El lenguaje PHP es un lenguaje de programación de estilo clásico, de programación con variables, sentencias condicionales, bucles, funciones. No es un lenguaje de marcas como podría ser HTML, XML o WML.

Pero a diferencia de Java o Java Script que se ejecutan en el navegador, PHP se ejecuta en el servidor, por eso nos permite acceder a los recursos que tenga el servidor como por ejemplo podría ser una base de datos. El programa PHP es ejecutado en el servidor y el resultado enviado al navegador. El resultado es normalmente una página HTML pero igualmente podría ser una página WML.

Al ser PHP un lenguaje que se ejecuta en el servidor no es necesario que su navegador lo soporte, es independiente del navegador, pero sin embargo para que sus páginas PHP funcionen, el servidor donde están alojadas debe soportar PHP."(JALÓN *et al.* 1998)

La utilización de PHP permite crear páginas dinámicas además de facilitar y mejorar las prestaciones de la base de datos MySQL. (MORERA *et al.* 2001)

Este lenguaje de programación es de propósito general. Es normalmente usado como un lenguaje de script embebido en HTML para su uso en la web, pero puede también usarse como un lenguaje script para shell o hasta como un lenguaje para escribir aplicaciones con ventanas, con PHP-GTK. (MOORE *et al.* 2001)

PHP tiene dos partes: (MOORE *et al.* 2001)

1. El motor Zend: Es la parte del paquete PHP que mantiene los pedidos, los procesos de los archivos de script y maneja las variables y los recursos.
2. PHP: Implementa el 90% de la funcionalidad que ve el usuario final. Brinda un amplio rango de módulos como el soporte para MySQL, ODBC and XML.

Lenguaje ASP (Páginas Active Server):

Es un entorno para crear y ejecutar aplicaciones dinámicas e interactivas en la Web. Se puede combinar páginas HTML, secuencias de comandos y componentes ActiveX para crear páginas y aplicaciones Web interactivas.

Este lenguaje puede convertir sus secuencias de comandos en aplicaciones que realicen tareas complejas, como la recopilación y el proceso de información de pedidos para comercios electrónicos. Además, si decide convertir sus aplicaciones en componentes (módulos de software compactos que encapsulan la lógica de su aplicación), puede utilizar ASP para implementar sus componentes. (ALVAREZ, JHON 1998)

Las páginas ASP comienzan a ejecutarse cuando un usuario solicita un archivo .asp al servidor Web a través del explorador. El servidor web llama a ASP, que lee el archivo solicitado, ejecuta las secuencias de comandos que encuentre y envía los resultados al explorador del cliente. Puesto que las secuencias de comandos se ejecutan en el servidor, y no en el cliente, es el servidor el que hace todo el trabajo necesario para

generar las páginas que se envían al explorador. Las secuencias de comandos quedan ocultas a los usuarios, estos solo reciben el resultado de la ejecución en formato HTML.

Los archivos .asp son archivos de texto normales, no es necesario ningún editor especial para crearlos, puede usarse cualquier editor que genere código ascii.

Un archivo .asp puede contener texto, código HTML, código ASP o cualquier combinación de estos. Si no contiene código ASP se comporta como un archivo .html normal.

Para la implantación de un servidor Web que soporte ASP el software necesario es, si lo que estamos configurando es un servidor de alto rendimiento:

- WINDOWS NT 4.0
- IIS 4.0 (INTERNET INFORMATION SERVER 4.0) Ó IIS3.0 + ASP.EXE

ASP viene de forma nativa con dos motores de secuencia de comandos Microsoft Visual Basic Scriptig Edition (VBScript) y Microsoft JScript. Puede instalar y utilizar motores de otros lenguajes como REXX y Perl.

Algunos objetos integrados de ASP:

- Objeto Application: se utiliza para compartir información entre todos los usuarios de una aplicación.
- Objeto Request: se utiliza para tener acceso a la información que se pasa en las peticiones HTTP. Entre dicha información se incluyen los parámetros que se pasan desde los formularios HTML mediante el método POST o el método GET, cookies y certificados de cliente.
- Objeto Response: se utiliza para controlar la información que se envía al usuario. Esto incluye el envío de información directamente al explorador, la redirección del explorador a otra dirección URL o el establecimiento de valores de las *cookies*.
- Objeto Server: proporciona acceso a los métodos y las propiedades del servidor.

Selección:

Una vez analizadas algunas de las características fundamentales de los lenguajes del lado del servidor PHP y ASP, se puede concluir que debido a la particularidad y funcionalidades que cada uno de ellos brinda, se ha seleccionado PHP, ya que este lenguaje por sus características específicas es el más óptimo para aplicar a este tipo de trabajo.

Corre en la mayoría de las plataformas, utilizando el mismo código fuente, pudiendo ser compilado y ejecutado en alrededor de 25. Este lenguaje de programación es similar a la del C, por esto cualquiera con experiencia en lenguajes del estilo C podrá entender rápidamente PHP, es completamente expandible, presenta muchas interfaces distintas para cada tipo de servidor, otras de las ventajas de PHP es que puede interactuar con muchos motores de bases de datos tales como MySQL, MS SQL, Oracle, PostgreSQL, y otros muchos. PHP es Open Source, lo cual significa que el usuario no depende de una compañía específica para arreglar cosas que no funcionan, otras de las superioridades que presenta este lenguaje ante ASP es que el manejo de errores no es tan sofisticado, pero también hay que tener en cuenta que en ambientes Windows compite muy de cerca con ASP. (DONDO 1999)

Como queda evidenciado, se hace necesaria la utilización del lenguaje PHP por las razones antes mencionadas para el desarrollo de la aplicación, ya que actualmente PHP está listo para su mejor momento. No puede dejar de analizarse también con la tecnología que se va a trabajar, ya que cada lenguaje tiene su mejor funcionamiento en tecnologías determinadas, en el caso del presente sistema de gestión, el que más se apropia es el seleccionado.

1.5 Gestores de Bases de Datos.

Para tener una breve idea de que es un Sistema Gestor de Base de Datos (SGBD) es importante saber que estos constituyen un tipo de software muy específico dedicado a servir de interfaz entre las Bases de Datos, el usuario y las aplicaciones que las utilizan. El (SGBD) consiste en un conjunto de programas, procedimientos y lenguajes que nos proporcionan las herramientas necesarias para trabajar con una base de datos. Incorporar una serie de funciones que nos permita definir los registros, sus campos, sus relaciones, insertar, suprimir, modificar y consultar los datos.(M. 1997)

Dentro de los gestores de base de datos más utilizados se encuentran: MySQL, Microsoft SQL Server, Oracle, PostgreSQL, entre muchos otros que al no mencionarlos aquí en este trabajo, no dejan de ser menos importantes.

1.5.1 Gestor MySQL.

Este gestor surgió como un intento de conectar el gestor mSQL a las tablas propias de MySQL AB, usando sus propias rutinas a bajo nivel. Tras unas primeras pruebas, vieron que mSQL no era lo bastante flexible para lo que necesitaban, por lo que tuvieron que desarrollar nuevas funciones. Esto resultó en una interfaz SQL a su base de datos, con una interfaz totalmente compatible a mSQL. (PECOS)

MySQL es un servidor de base de datos particularmente rápido, robusto y fácil de usar. Permite almacenar datos de cualquier tipo. Lo único que se necesita aprender son algunos comandos del leguaje SQL (Structured Query Language) para añadir, modificar, crear y eliminar información. (MORERA *et al.* 2001)

MySQL es un gestor de base de datos sencillo de usar. También es uno de los motores de base de datos más usados en Internet, la principal razón de esto es que este potente gestor es gratis para aplicaciones no comerciales. (JALÓN *et al.* 1998)

Algunas de las características principales de este gestor de base de datos son: (JALÓN *et al.* 1998)

1. Es una base de datos relacional: Una base de datos relacional es un conjunto de datos que están almacenados en tablas entre las cuales se establecen unas relaciones para manejar los datos de una forma eficiente y segura. Para usar y gestionar una base de datos relacional se usa el lenguaje estándar de programación SQL.
2. Es Open Source: El código fuente de MySQL se puede descargar y está accesible a cualquiera, por otra parte, usa la licencia GPL de la GNU para aplicaciones no comerciales.
3. Es una base de datos muy rápida, segura y fácil de usar: Gracias a la colaboración de muchos usuarios, la base de datos se ha ido mejorando, optimizándose en velocidad.

MySQL es un sistema de gestión de bases de datos relacional. Su diseño multihilo le permite soportar una gran carga de forma muy eficiente. MySQL fue creada por la empresa sueca MySQL AB, que mantiene el copyright del código fuente del servidor SQL, así como también de la marca. Aunque MySQL es software libre, MySQL AB distribuye una versión comercial de MySQL, que no se diferencia de la versión libre más que en el soporte técnico que se ofrece, y la posibilidad de integrar este gestor en un software propietario, ya que de no ser así, se vulneraría la licencia GPL. Este gestor de bases de datos es, probablemente, el gestor más usado en el mundo del software libre, debido a su gran rapidez y facilidad de uso. Esta gran aceptación es debida, en parte, a que existen infinidad de librerías y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación, además de su fácil instalación y configuración. (PECOS)

Otras características de este gestor de bases de datos son: (PECOS)

- Aprovechamiento de la potencia de sistemas multiprocesador, gracias a su implementación multihilo.

- Soporta gran cantidad de tipos de datos para las columnas.
- Dispone de API's en gran cantidad de lenguajes (C, C++, Java, PHP, etc).
- Gran portabilidad entre sistemas.
- Soporta hasta 32 índices por tabla.
- Gestión de usuarios y passwords, manteniendo un muy buen nivel de seguridad en los datos.

Otras características que presenta MySQL son: (Wolf Kira, 2004)

- Velocidad a la hora de realizar las operaciones, lo que le hace uno de los gestores que ofrecen mayor rendimiento.
- Su bajo consumo lo hacen apto para ser ejecutado en una máquina con escasos recursos sin ningún problema.
- Las utilidades de administración de este gestor son envidiables para muchos de los gestores comerciales existentes, debido a su gran facilidad de configuración e instalación.
- Tiene una probabilidad muy reducida de corromper los datos, incluso en los casos en los que los errores no se produzcan en el propio gestor, sino en el sistema en el que está.

1.5.2 Gestor PostgreSQL.

El Sistema Gestor de Bases de Datos Relacionales Orientadas a Objetos conocido como PostgreSQL, está derivado del paquete Postgre escrito en Berkeley. Con cerca de una década de desarrollo tras él, PostgreSQL es el gestor de bases de datos de código abierto más avanzado hoy en día, ofreciendo control de concurrencia multi-versión, soportando casi toda la sintaxis SQL (incluyendo subconsultas, transacciones, y tipos y funciones definidas por el usuario), contando también con un amplio conjunto de enlaces con lenguajes de programación (incluyendo C, C++, Java, Perl, Tcl y Python). (LOCKHART 1996)

PostgreSQL es un sofisticado gestor de bases de datos relacionales que soporta casi todas las construcciones SQL. En la actualidad es el más avanzado sistema de bases de datos de código abierto disponible. (MORERA *et al.* 2001)

Los sistemas de mantenimiento de Bases de Datos relacionales tradicionales (DBMS) soportan un modelo de datos que consisten en una colección de relaciones con nombre, que contienen atributos de un tipo específico. En los sistemas comerciales actuales, los tipos posibles incluyen numéricos de punto flotante, enteros, cadenas de caracteres, cantidades monetarias y fechas. Está generalmente reconocido que este

modelo será inadecuado para las aplicaciones futuras de procesado de datos. El modelo relacional sustituyó modelos previos en parte por su "simplicidad espartana". Sin embargo, como se ha mencionado, esta simplicidad también hace muy difícil la implementación de ciertas aplicaciones. Postgre ofrece una potencia adicional sustancial al incorporar los siguientes cuatro conceptos adicionales básicos en una vía en la que los usuarios pueden extender fácilmente el sistema: Clases, Herencia, Tipos y Funciones. (LOCKHART 1996)

Otras características aportan potencia y flexibilidad adicional: (LOCKHART 1996)

- Restricciones (Constraints)
- Disparadores (triggers)
- Reglas (rules)
- Integridad transaccional

Estas características colocan a Postgre en la categoría de las Bases de Datos identificadas como objeto-relacionales. Nótese que éstas son diferentes de las referidas como orientadas a objetos, que en general no son bien aprovechables para soportar lenguajes de Bases de Datos relacionales tradicionales. Postgre tiene algunas características que son propias del mundo de las bases de datos orientadas a objetos. De hecho, algunas Bases de Datos comerciales han incorporado recientemente características en las que Postgre fue pionera. Postgre ha pasado por varias revisiones importantes desde entonces. El primer sistema de pruebas fue operacional en 1987 y fue mostrado en la Conferencia ACM-SIGMOD de 1988. La versión 1, fue lanzada a unos pocos usuarios externos en Junio de 1989. La versión 2 salió en Junio de 1990. La versión 3 apareció en 1991 y añadió una implementación para múltiples gestores de almacenamiento, un ejecutor de consultas mejorado y un sistema de reescritura de reglas nuevo. En su mayor parte, las siguientes versiones hasta el lanzamiento de Postgres95 se centraron en mejorar la portabilidad y la fiabilidad. (LOCKHART 1996)

Las principales mejoras en PostgreSQL incluyen: (LOCKHART 1996)

- Los bloqueos de tabla han sido sustituidos por el control de concurrencia multi-versión, el cual permite a los accesos de sólo lectura continuar leyendo datos consistentes durante la actualización de registros, y permite copias de seguridad en caliente desde pg_dump mientras la base de datos permanece disponible para consultas.

- Se han implementado importantes características del motor de datos, incluyendo subconsultas, valores por defecto, restricciones a valores en los campos (constraints) y disparadores (triggers).
- Se han añadido funcionalidades en línea con el estándar SQL92, incluyendo claves primarias, identificadores entrecomillados, forzado de tipos cadenas literales, conversión de tipos y entrada de enteros binarios y hexadecimales.
- Los tipos internos han sido mejorados, incluyendo nuevos tipos de fecha/hora de rango amplio y soporte para tipos geométricos adicionales.
- La velocidad del código del motor de datos ha sido incrementada aproximadamente en un 20-40%, y su tiempo de arranque ha bajado el 80% desde que la versión 6.0 fue lanzada.

1.5.3 Selección del Gestor de BD a utilizar.

Tanto MySQL como PostgreSQL son magníficos para el manejo de base de datos, así que cada servidor deberá elegir cual se acomoda a sus necesidades específicas. En la presente investigación se ha seleccionado PostgreSQL como gestor de base de datos a utilizar, para ello fue necesario realizar una breve comparación de estos dos potentes gestores. En el sistema propuesto se hace necesario realizar grandes cantidades de consultas a la base de datos, además es preciso que el gestor a utilizar cuente con alta escalabilidad y pueda comprobar la integridad referencial.

- Lo mejor de PostgreSQL. (PECOS 2005)

1. Posee una gran escalabilidad. Es capaz de ajustarse al número de CPUs y a la cantidad de memoria que posee el sistema de forma óptima, haciéndole capaz de soportar una mayor cantidad de peticiones simultáneas de manera correcta.
2. Implementa el uso de rollback's, subconsultas y transacciones, haciendo su funcionamiento mucho más eficaz, y ofreciendo soluciones en campos en las que MySQL no podría.
3. Tiene la capacidad de comprobar la integridad referencial, así como también la de almacenar procedimientos en la propia base de datos, equiparándolo con los gestores de bases de datos de alto nivel.

Características que tiene PostgreSQL en desventaja. (PECOS 2005)

1. Consume gran cantidad de recursos.
2. Tiene un límite de 8K por fila, aunque se puede aumentar a 32K, con una disminución considerable del rendimiento.
3. Es de 2 a 3 veces más lento que MySQL.

- Lo mejor de MySQL. (PECOS 2005)

1. Su velocidad a la hora de realizar las operaciones, lo que le hace uno de los gestores que ofrecen mayor rendimiento.
2. Su bajo consumo lo hacen apto para ser ejecutado en una máquina con escasos recursos sin ningún problema.
3. Las utilidades de administración de este gestor son envidiables para muchos de los gestores comerciales existentes, debido a su gran facilidad de configuración e instalación.
4. Tiene una probabilidad muy reducida de corromper los datos, incluso en los casos en los que los errores no se produzcan en el propio gestor, sino en el sistema en el que está.

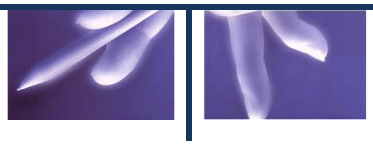
Características que tiene MySQL en desventaja. (PECOS 2005)

1. Carece de soporte para transacciones, rollback's y subconsultas.
2. El hecho de que no maneje la integridad referencial, hace de este gestor una solución pobre para muchos campos de aplicación, sobre todo para aquellos programadores que provienen de otros gestores que sí que poseen esta característica.
3. No es viable para su uso con grandes bases de datos, a las que se acceda continuamente, ya que no implementa una buena escalabilidad.

1.6 Conclusiones Parciales

Después de analizada la importancia que tiene elevar al máximo la calidad de los productos en el proceso de desarrollo del software, principalmente el educativo; y visto que en la Universidad de las Ciencias Informáticas no se cuenta con una herramienta que permita gestionarla y controlarla, se hace necesario proponer un sistema (la presente investigación) que dé respuesta a estas necesidades existentes. En este capítulo se profundizó en el conocimiento de algunos conceptos necesarios para la comprensión del trabajo. Además se realizó un análisis de las tecnologías y herramientas que serán utilizadas a lo largo del desarrollo del sistema propuesto y se fundamentaron las elecciones realizadas para la selección del lenguaje, del sistema gestor de bases de datos y de la metodología a utilizar.

CAPÍTULO II:
CARACTERÍSTICAS DEL SISTEMA



2.1 Introducción

El presente capítulo tiene como objetivo describir la propuesta de solución a la situación problémica. El mismo abordará los procesos elementales del negocio, también la realización y especificación de los casos de usos de cada proceso.

Por otra parte, también se brinda una concepción general del sistema que se ha propuesto, se muestran todas las funcionalidades y particularidades del mismo, quedan identificados los actores del sistema y se presenta el diagrama de caso de uso del sistema con sus respectivas descripciones textuales por cada caso de uso existente.

2.2 Problema y situación problémica

Actualmente en la Universidad de las Ciencias Informáticas resulta sumamente difícil controlar los estudiantes que están interesados en el perfil de calidad de la Facultad 8 y de qué cursos han culminado los mismos, trayendo esto consigo una total desinformación y por tal motivo no se ofertan quizás los cursos más necesarios. Generalmente todo este proceso se realiza de manera manual, lo que trae como consecuencia un gran gasto de mano de obra, empleo de tiempo y tardanza en la actualización y manipulación de la información relacionada con los cursos de calidad. Además de esto, los estudiantes de la facultad no tienen conocimiento sobre cuales son los cursos que conforman el perfil de calidad, mucho menos qué contenidos se imparten en los mismos, tampoco se sabe con claridad el momento exacto en que el estudiante posee el perfil básico. Otro de los problemas existentes en la facultad referente al perfil de calidad es lo difícil que resulta elaborar un reporte relacionado con el tema del perfil de calidad, sencillamente porque hay que auxiliarse en la información existente en los distintos documentos que posee el perfil de calidad y la mayoría de las veces esto toma mucho tiempo, provocando demoras en la obtención de la información según las necesidades del interesado.

La UCI cuenta con un sistema para la gestión académica [Akademos], el mismo tiene un alcance general, ya que gestiona todas las asignaturas que pertenecen a pregrado, pero una gran desventaja del mismo es que no ofrece todos los reportes necesarios para tratar el tema del perfil de calidad. Todas estas restricciones, dificultades y necesidades que dieron comienzo a esta investigación afectan a todos los individuos involucrados en el proceso.

2.3 Objeto de automatización

Los procesos que serán objeto de automatización cuentan con diferentes actividades, las cuales se mencionan a continuación:

- Guardar el profesor disponible para impartir un curso del perfil.
- Crear el grupo del curso del perfil a capacitar.
- Registrar las evaluaciones del desempeño de los estudiantes en el curso.
- Controlar la asistencia de los estudiantes pertenecientes a cursos.
- Verificar el estado de completamiento del perfil de un estudiante.
- Informar los estudiantes que fueron seleccionados para recibir capacitación de un curso.

2.4 Información que se maneja

El sistema manipulará abundante información, toda relacionada a los cursos del perfil de calidad de la Facultad 8, dicha información servirá de base para todas aquellas personas interesadas en la adquisición de conocimientos de calidad, sean capaces de nutrirse al cien por ciento. La misma debe estar disponible en un formato legible para que pueda ser utilizada de manera positiva y eficaz. Fundamentalmente se manejará información referente a la capacitación del perfil de calidad con el objetivo de orientar, capacitar y evaluar a los interesados.

2.5 Propuesta de sistema

Este trabajo tiene como propuesta de sistema, el análisis y diseño de un Sistema para la Gestión de la Calidad del módulo de “Capacitación”. Dentro de este sistema se encuentran además cuatro módulos, estos son: Administración, Auditorías y Revisiones, Pruebas y Planificación.

2.6 Modelo de negocio

El modelamiento del negocio en la etapa de concepción de un proyecto de desarrollo de software es una de las actividades más importantes, y que muchas veces no se lleva a cabo con la profundidad necesaria, provocando esto que no haya una total comprensión de los procesos a informatizar y un falso sentido de entendimiento entre los clientes y el equipo de desarrollo respecto al trabajo a realizar.

Para que un sistema informático satisfaga las necesidades de los clientes, es necesario que se cumplan de manera eficiente los objetivos del Flujo de Trabajo Modelamiento del Negocio.

Estos objetivos son:

1. Entender los problemas actuales en la organización o empresa para identificar los aspectos a mejorar.
2. Comprender la estructura y el dinamismo de la organización o empresa para la cual se va a desarrollar el sistema software.
3. Estudiar el impacto que pueden producir los cambios a nivel organizativo.
4. Asegurar que los clientes, usuarios finales, desarrolladores y otros involucrados tienen una visión común de la organización considerada.
5. Obtener los requisitos del sistema software.
6. Entender como el sistema software encaja en la organización.

Para conseguir estos objetivos, el flujo de trabajo de Modelado del Negocio consta de las siguientes etapas:

- Evaluar el estado del Negocio.
- Análisis del Negocio.
- Identificar Procesos de Negocio.
- Definir y Refinar los Procesos de Negocio.
- Diseño de la Realización de los Procesos de Negocio.
- Evaluación.

2.6.1 Actores y trabajadores del negocio

Tabla 1: Actores del negocio y sus descripciones.

Actor	Descripción
1. Interesado.	Puede ser el asesor de calidad, el coordinador de quinto año, el vicedecano docente, el jefe de Ingeniería y Práctica Profesional o bien todos aquellos profesores responsables del grupo de calidad de la facultad que necesitan estar al tanto de la formación de sus estudiantes.
2. Responsable de Práctica Profesional.	Puede ser el asesor de calidad, el coordinador de quinto año, el vicedecano docente o bien el jefe del departamento de Ingeniería de Software y Práctica Profesional, quien solicita que se preparen cursos del perfil de calidad.

Tabla 2: Trabajadores del negocio y sus descripciones.

Trabajador	Descripción
1. Coordinador del perfil de calidad.	Es el principal responsable de la preparación y ejecución de los cursos del perfil de calidad de la facultad.
2. Profesor de cursos del perfil.	Es el encargado de llevar el control de la asistencia y las evaluaciones de los estudiantes en los cursos del perfil.
3. Planificador.	Es la persona que se encarga de planificar los cursos del perfil.

2.6.2 Diagrama de casos de uso del negocio

El modelado de caso de uso está compuesto básicamente por dos elementos: los diagramas de casos de uso y las descripciones de los mismos.

Los diagramas de casos de usos muestran el comportamiento del sistema a partir de los usuarios que interactúan con el sistema. En otras palabras, describe gráficamente quien utiliza el sistema y la forma en que los usuarios esperan interaccionar con él.

A continuación en la Figura 2 se muestra el diagrama correspondiente al los casos de uso del negocio estudiado.

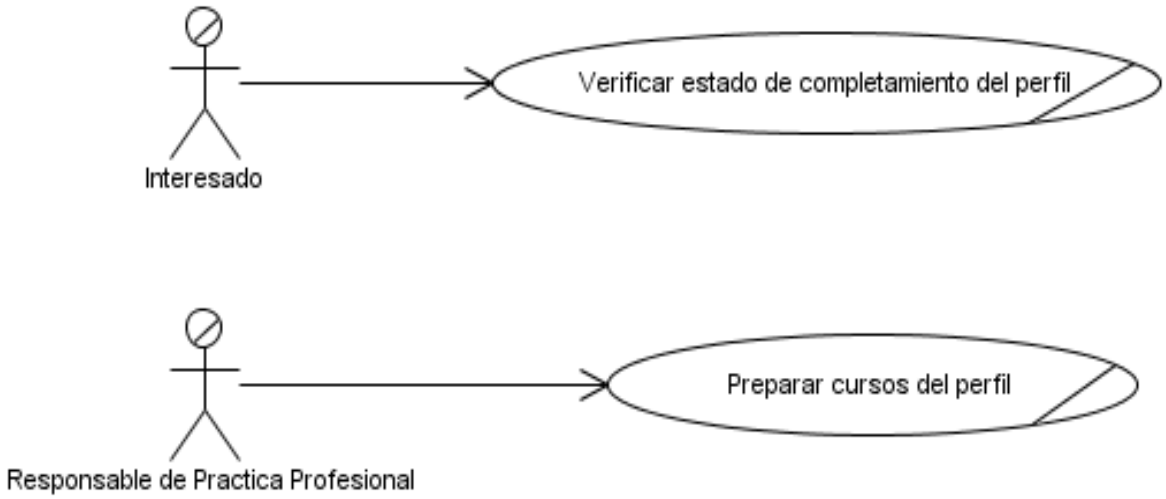


Figura 2: Diagrama de Casos de Uso del Negocio.

2.6.3 Descripciones textuales

Tabla 3: Descripción literal del Caso de Uso del Negocio “Confeccionar grupo del curso del perfil".

Caso de uso del negocio	*Preparar cursos del perfil*
Actores	1. Responsable de Práctica Profesional.
Trabajadores	1. Coordinador del perfil de calidad. 2. Planificador. 3. Profesor de cursos del perfil.
Resumen	El caso de uso se inicia cuando el actor solicita que se preparen cursos del perfil de calidad. El coordinador busca el profesor que esté disponible para impartir el curso, luego busca los estudiantes que integrarán el curso y realiza la matrícula del mismo. Inmediatamente el coordinador del perfil de calidad registra en un “Documento de Control de Cursos del Perfil de Calidad” el grupo del curso del perfil que ya está preparado y le envía al planificador los datos del mismo. El planificador planifica el curso en el horario. El profesor imparte el curso y luego el coordinador del perfil registra las notas finales en el control de cursos, terminando así el caso de uso.
Casos de uso asociados	
Precondiciones	
Acción del actor	*Respuesta del proceso de negocio*
1. El caso de uso se inicia cuando el actor solicita que se preparen cursos del perfil de calidad.	
	2. El coordinador del perfil busca el profesor que esté disponible para impartir el curso.
	3. Luego el coordinador del perfil busca los estudiantes que integrarán el curso y realiza la matrícula del mismo.

	4. Inmediatamente el coordinador del perfil de calidad registra en un "Documento de Control de Cursos del Perfil de Calidad" el grupo del curso del perfil que ya está preparado y le envía al planificador los datos del mismo.
	5. El planificador asigna los locales y turnos de clases en que debe desarrollarse el curso del perfil.
	6. El planificador envía un correo al coordinador del perfil, informando el horario final con los respectivos locales.
	7. El coordinador del perfil envía un correo al profesor encargado de impartir el curso con el objetivo de informar el horario con la ubicación del local.
	8. El profesor imparte el curso, controlando en el mismo la asistencia y evaluaciones de los estudiantes en el "Registro de Asistencia y Evaluaciones".
	9. El profesor envía un correo al coordinador del perfil, enviándole en el mismo las notas finales de los estudiantes y notificándole además que ha terminado el curso.
	10. El coordinador del perfil registra que el curso ha terminado, actualizando su "Documento de Control de Cursos del Perfil de Calidad".
.	11. El caso de uso termina.
Flujos alternos	
2. a No existe profesor disponible para impartir el curso.	
	2. a.1 El coordinador espera que haya un profesor disponible para impartir el curso del perfil.
	2. a.2 Una vez que aparece el profesor disponible para impartir el curso, el coordinador del perfil sigue el flujo normal de los eventos.
Mejoras propuestas	

Tabla 4: Descripción literal del Caso de Uso del Negocio "Verificar estado de completamiento del perfil".

Caso de uso del negocio	*Verificar estado de completamiento del perfil*

Actores	1. Interesado.
Trabajadores	1. Coordinador del perfil de calidad.
Resumen	El caso de uso se inicia cuando el actor solicita verificar el estado de completamiento del perfil. El coordinador del perfil busca en un "Documento de Control de Cursos del Perfil de Calidad" todos los cursos que han culminado los estudiantes, para ver si estos cursos complementan el perfil de calidad. El coordinador del perfil informa al interesado, el estado de completamiento del perfil según los datos consultados. El caso de uso termina.
Casos de uso asociados	
Acción del actor	*Respuesta del proceso de negocio*
1. El caso de uso se inicia cuando el actor solicita verificar el estado de completamiento del perfil.	
	2. El coordinador busca en un "Documento de Control de Cursos del Perfil de Calidad" todos los cursos que han culminado los estudiantes.
	3. El coordinador informa al interesado el estado de completamiento del perfil.
	4. El caso de uso termina.
Flujos alternos.	
Mejoras propuestas	

2.6.4 Diagramas de actividades

Un diagrama de actividades puede considerarse como un caso especial de un diagrama de estados en el cual casi todos los estados son estados acción (identifican una acción que se ejecuta al estar en él) y casi todas las transiciones evolucionan al término de dicha acción (ejecutada en el estado anterior). Un diagrama de actividades puede dar detalle a un caso de uso, un objeto o un mensaje en un objeto. Permiten representar transiciones internas al margen de las transiciones o eventos externos. A continuación se muestran los diagramas de actividades del presente trabajo.

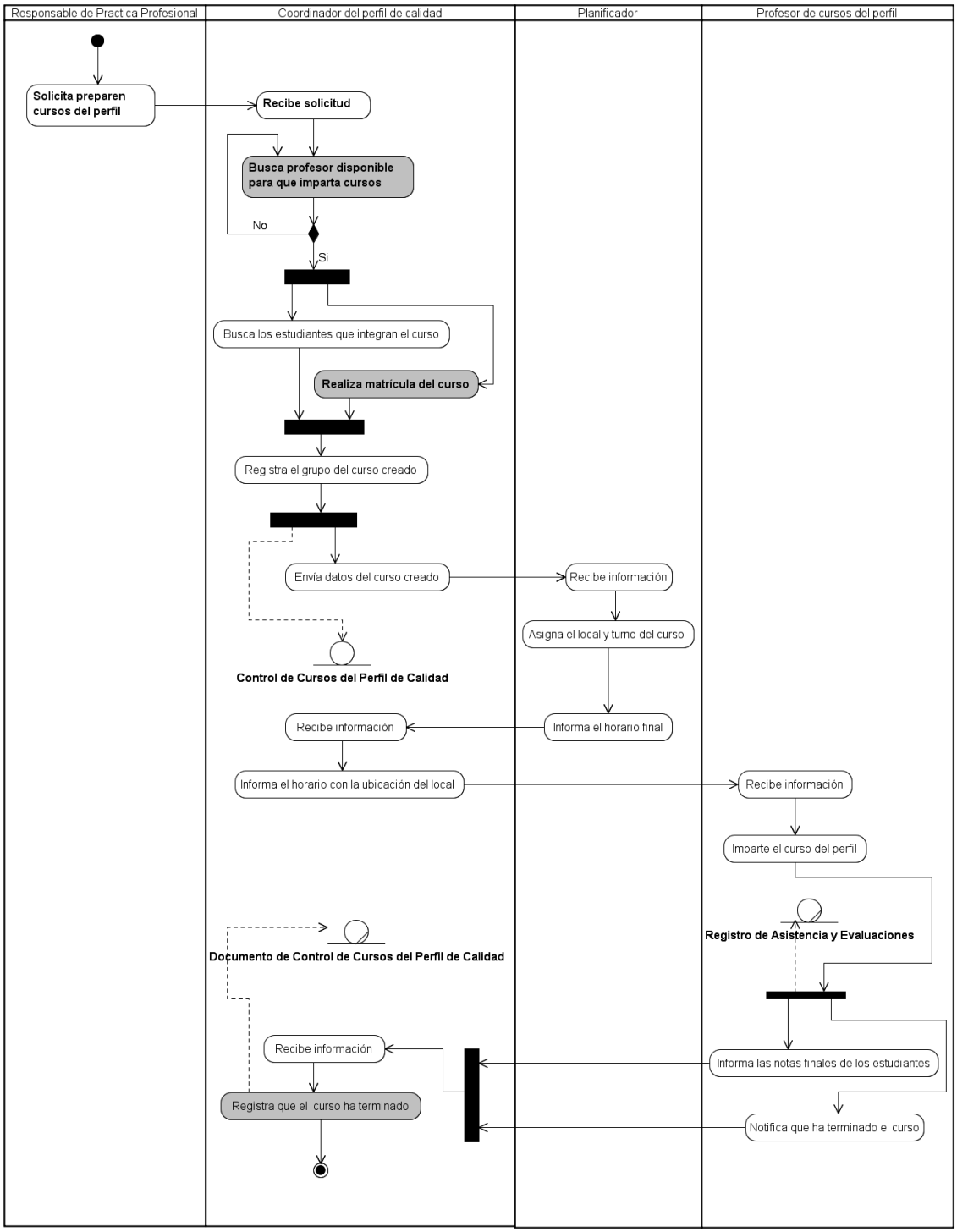


Figura 3: Diagrama de actividades “Preparar cursos del perfil”.

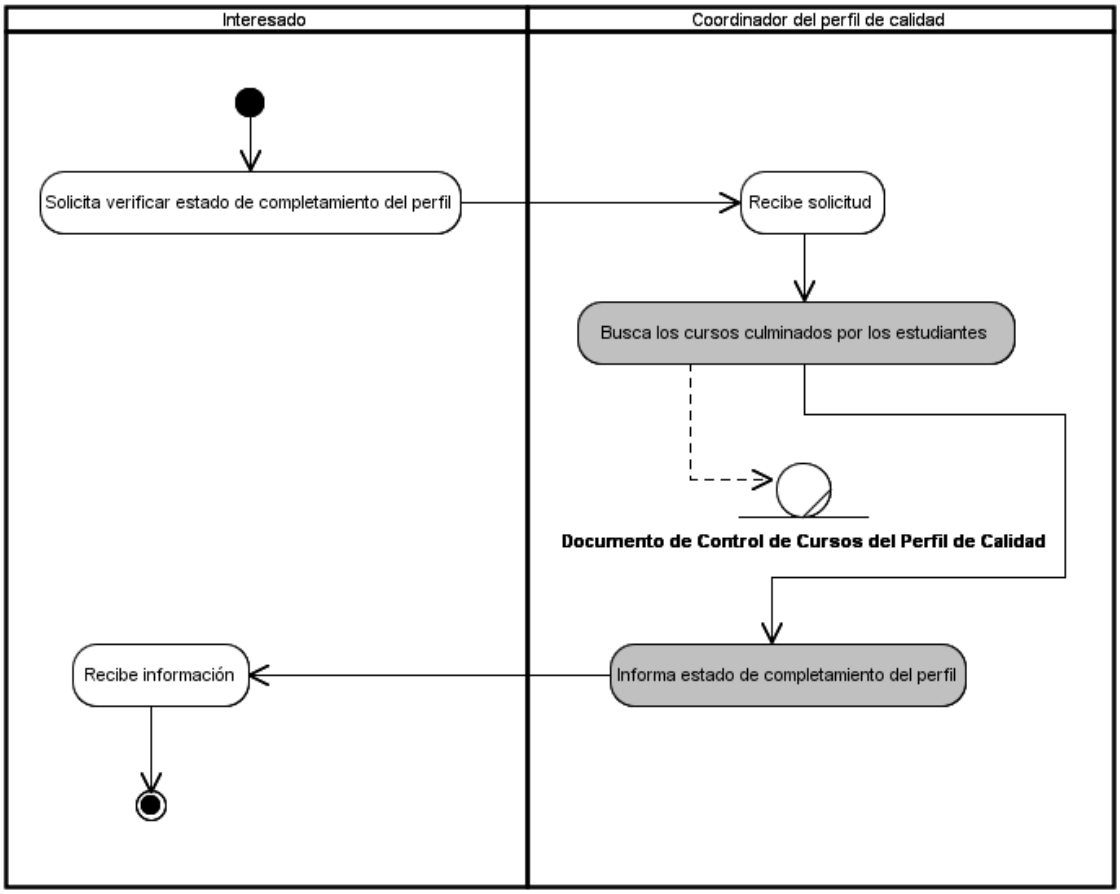


Figura 4: Diagrama de actividades “Verificar estado de completamiento del perfil”.

2.6.5 Diagrama de clases del modelo de objetos

Figura 5: Modelo de objetos del negocio.

2.7 Especificación de los requisitos del sistema

2.7.1 Ingeniería de Requerimientos.

La Ingeniería de Requerimientos (IR) cumple un papel primordial en el proceso de producción de software, ya que enfoca un área fundamental: la definición de lo que se desea producir. Su principal tarea consiste en la generación de especificaciones correctas que describan con claridad, sin ambigüedades, en forma consistente y compacta, el comportamiento del sistema; de esta manera, se pretende minimizar los problemas relacionados al desarrollo de sistemas que gestionan la calidad.

Resumiendo, se puede decir que (IR) es un proceso que ofrece diferentes alternativas para recopilar los requerimientos declarados por los clientes y modelar lo que el sistema va a realizar mediante especificaciones completas y consistentes del comportamiento del mismo.

2.7.2 Requerimientos.

La IEEE Standard Glossary of Software Engineering Terminology define un requerimiento como:

1. Condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo.
2. Condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente.
3. Una representación documentada de una condición o capacidad como en 1 o 2. (IEEE)

Existen dos grandes clasificaciones de los requerimientos:

2.7.3 Requerimientos Funcionales:

Son capacidades o condiciones que el sistema debe cumplir.

En la realización de los casos de uso del negocio, se obtienen las actividades que serán objeto de automatización. Estas actividades no son exactamente los requerimientos funcionales, pero sí son el punto de partida para identificar qué debe hacer el sistema. Los requerimientos funcionales se mantienen invariables sin importar con qué propiedades o cualidades se relacionen.

Dentro de los requerimientos funcionales del sistema propuesto se encuentran:

RF1- Gestionar programa de la asignatura (disciplina, curso optativo, semestre, año, duración total, distribución de horas por temas, objetivos, descripción de los temas, sistema de habilidades, sistema de evaluación de la asignatura, bibliografía, programa elaborado por).

RF1.1- Agregar nuevo programa de la asignatura (P1). (Enviar a los implicados)

RF1.2- Modificar programa de la asignatura (P1). (Depende de las necesidades del proyecto).

RF1.3- Eliminar programa de la asignatura (P1). (Enviar a los implicados)

RF1.4- Exportar a extensión .pdf el programa de la asignatura (P1).

RF1.5- Mostrar el programa de la asignatura (P1).

RF2- Consultar estructura del programa de la asignatura (P1).

RF3- Gestionar asignatura del perfil de calidad.

RF3.1- Agregar una nueva asignatura del perfil de calidad. (Nombre, año, duración, semestre, distribución de horas por temas, profesor, fecha de creación).

RF3.2- Modificar dato de la asignatura del perfil de calidad. (Guardar fecha de actualización).

RF3.3- Eliminar asignatura del perfil de calidad.

RF3.4- Mostrar asignatura del perfil de calidad.

RF4- Consultar datos de la asignatura del perfil de calidad.

RF5- Verificar estado de completamiento del perfil.

RF5.1- Exportar a extensión .pdf estado de completamiento del perfil.

RF6- Gestionar profesor (nombre, apellidos, imparte curso actualmente, facultad, número de solapín).

RF6.1- Agregar un nuevo profesor.

RF6.2- Eliminar profesor.

RF6.3- Modificar profesor.

RF6.4- Mostrar profesor.

RF7- Conocer datos de un profesor.

RF8- Gestionar estudiante (nombre, apellidos, año, facultad, número de solapín, usuario).

RF8.1- Agregar un nuevo estudiante.

RF8.2- Eliminar estudiante.

RF8.3- Modificar estudiante.

RF8.4- Mostrar estudiante.

RF9- Gestionar evaluaciones de los estudiantes del curso.

RF9.1- Agregar evaluación del estudiante.

RF9.2- Modificar evaluación del estudiante.

RF9.3- Eliminar evaluación del estudiante.

RF9.4- Mostrar evaluación del estudiante.

RF10- Conocer evaluación del estudiante.

RF11- Gestionar registro de asistencia.

RF11.1- Agregar asistencia al estudiante.

RF11.2- Modificar asistencia al estudiante.

RF11.3- Eliminar asistencia al estudiante.

RF11.4- Mostrar asistencia del estudiante.

RF12- Conocer asistencia del estudiante.

RF13- Gestionar curso.

RF13.1- Agregar estudiante al curso.

RF13.2- Agregar profesor al curso.

RF13.3- Agregar asignatura al curso.

RF13.4- Cancelar matrícula del curso.

RF13.5- Cambiar profesor del curso.

RF13.6- Mostrar matrícula del curso.

RF13.7- Imprimir registro de control de asistencia y evaluaciones.

RF13.8- Mostrar curso.

RF14- Consultar datos de los cursos. (Estudiantes acreditados, estudiantes pendientes).

RF14.1- Exportar a extensión .pdf datos de los cursos.

RF15- Gestionar planificación de horario de cursos.

RF15.1- Insertar curso al horario.

RF15.2- Modificar curso al horario.

RF15.3- Eliminar curso al horario.

RF15.4- Mostrar horario.

RF16- Consultar datos del horario.

RF17- Generar evaluación de Práctica Profesional (PP). (Trayectoria del estudiante y una nota del profesor para promediarla y dar la nota final).

RF18- Consultar evaluación de Práctica Profesional (PP).

2.7.4 Requerimientos no funcionales:

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al

producto atractivo, usable, rápido o confiable; por ejemplo, pudiera desearse que el sistema responda dentro de un intervalo de tiempo especificado o que obtenga los resultados de los cálculos con un nivel de precisión dado.

Existen múltiples categorías para clasificar los requerimientos no funcionales, siendo las siguientes representativas de un conjunto de aspectos que se deben tener en cuenta, aunque no limitan a la definición de otros.

- Restricciones en el diseño y la implementación.

Este tipo de requerimiento especifica o restringe la codificación o construcción de un sistema, son restricciones que han sido ordenadas y deben ser cumplidas estrictamente. Ejemplos de ellas son:

1. Estándares requeridos.
2. Lenguajes de programación a ser usados para la implementación.
3. Uso obligatorio de ciertas herramientas de desarrollo.
4. Restricciones en la arquitectura o el diseño.

- Requerimientos de apariencia o interfaz externa.

Este tipo de requerimiento describe la apariencia del producto. Es importante destacar que no se trata del diseño de la interfaz en detalle sino que especifican cómo se pretende que sea la interfaz externa del producto.

- Requerimientos de Seguridad.

Este es quizás el tipo de requerimiento más difícil, que provocará los mayores riesgos si no se maneja correctamente. La seguridad puede ser tratada en tres aspectos diferentes, estos son:

1. Confidencialidad: La información manejada por el sistema está protegida de acceso no autorizado y divulgación.
2. Integridad: La información manejada por el sistema será objeto de cuidadosa protección contra la corrupción y estados inconsistentes, de la misma forma será considerada igual a la fuente o autoridad de los datos. Pueden incluir también mecanismos de chequeo de integridad y realización de auditorías.
3. Disponibilidad: Los usuarios autorizados se les garantizará el acceso a la información y que los dispositivos o mecanismos utilizados para lograr la seguridad no ocultarán o retrasarán a los usuarios para obtener los datos deseados en un momento dado.

Dentro de los requerimientos no funcionales del sistema propuesto, se encuentran:

1. Usabilidad.

- El sistema podrá ser usado por todas aquellas personas de los proyectos de la facultad 8, fundamentalmente integrantes del grupo de calidad de la facultad y calidad UCI.
- La aplicación deberá poseer una interfaz y navegación asequibles, y funcionales tanto para usuarios expertos como para los que no tienen conocimientos profundos de informática.
- Mantener informado al usuario del resultado de las operaciones.

2. Rendimiento.

- Operaciones normales en dos hilos.
- Respuestas rápidas en segundos al usuario.
- Debe estar disponible las 24 horas del día.

3. Portabilidad.

- El sistema debe ser Multiplataforma. Haciendo énfasis en el sistema operativo Linux.

4. Seguridad.

- Se permitirá la autenticación, la cual será una contraseña de acceso para los usuarios autorizados por el administrador. El sistema informático debe garantizar que la información sea vista únicamente por quien tenga permiso para esto. Él sistema debe permitir solo el acceso a cada uno de sus servicios al personal requerido para esto.
- El sistema informará quien es el usuario correspondiente con la sección que esté activa.
- Confiabilidad: Garantía de un tratamiento adecuado de las excepciones y validación de las entradas del usuario.
- Disponibilidad: El sistema debe permitirle al usuario conectarse desde cualquier PC de la red.
- Integridad: La información manejada por el sistema será objeto de cuidadosa protección contra la corrupción de datos.

5. Apariencia o interfaz externa.

- El sistema debe poseer una interfaz Web amigable y sencilla, fácil para la interacción del usuario. Con colores en su interfaz que distingan a la facultad, sin saturación de colores ni imágenes.
- El sistema proporcionará claridad y correcta organización de la información, permitiendo la interpretación correcta e inequívoca de esta, facilitando el uso de la aplicación.

- El sistema implementará la ejecución de acciones de una manera rápida, minimizando los pasos a ejecutar en cada proceso.
- En la interfaz de presentación se mostrará el nombre del producto en este caso: “Capacitación de Calidad”.

6. Soporte.

- La aplicación tendrá un periodo de tiempo en prueba para ver su funcionamiento y que cumpla con lo requerido dándole mantenimiento para luego brindar servicio de instalación.
- Documentación necesaria para el uso o desarrollo de las funcionalidades de la herramienta.

7. Requisitos para la documentación de usuarios en línea y ayuda del sistema.

- La aplicación tiene que llevar un manual de ayuda.

8. Interfaz.

- Las interfaces deben ser lo más entendible posible.
- Todas las interfaces tienen que tener una forma de cerrarse o retornar al paso anterior.
- Ninguna interfaz puede ir cargada de muchos colores evitando la desconcentración del usuario.

9. Requisitos Legales, de Derecho de Autor y otros.

- Es un deber proteger la información por parte de las personas que tienen derecho de administración u otros que pongan en peligro la integridad y seguridad del sistema.

10. Confiabilidad.

- Está prohibida la diseminación pública de la información por cualquier usuario a cualquier nivel.

11. Políticos-culturales.

- La herramienta sólo podrá ser utilizada dentro de la Universidad de las Ciencias Informáticas.
- Las funcionalidades del sistema no deben contener palabras en otros idiomas.
- Se deben poner las palabras lo más acorde a su entendimiento.

2.7.5 Modelación del sistema

La modelación de sistemas muestra la forma en que el sistema tiene que funcionar. Al diagramar las relaciones que hay entre las actividades del sistema, la modelación de sistemas facilita la comprensión de las relaciones entre las diversas actividades y el

impacto que tienen entre sí. Muestra los procesos como parte de un gran sistema cuyo objetivo es responder a una necesidad específica del cliente. La modelación de sistemas es muy útil cuando se necesita contar con un panorama general, dado que ilustra la forma en que se interrelacionan los servicios directos y auxiliares, de dónde provienen los insumos críticos y la forma prevista en que los productos o los servicios responderán a las necesidades de la comunidad.

2.7.6 Actores del sistema

Tabla 5: Actores del sistema y sus descripciones textuales.

Actor	Descripción
1. Coordinador del perfil de calidad.	Es la persona que se encarga de buscar profesores y estudiantes disponibles para la capacitación del perfil de calidad y gestionar la asignatura.
2. Profesor de cursos del perfil.	Es el encargado de llevar el control de la asistencia y las evaluaciones de los estudiantes.
3. Planificador.	Es la persona que se encarga de planificar los cursos del perfil.
4. Líder del proyecto de calidad.	Persona que está al frente del proyecto de calidad y se encarga de organizar los diferentes equipos dentro del proyecto para garantizar el cumplimiento del los objetivos.
5. Estudiante.	Es el interesado de recibir los cursos de capacitación con el objetivo de cumplimentar el perfil de calidad.
6. Usuario.	Esta persona puede ser un interesado en consultar información referente a los cursos del perfil de calidad. La misma puede ser: Planificador, Estudiante, Profesor de cursos del perfil, Líder del proyecto de calidad o el Coordinador del perfil de calidad.
7. Consultor de datos del horario.	Esta persona puede ser un interesado en consultar información referente al horario docente. La misma puede ser: Profesor de cursos del perfil, Estudiante, Planificador o el Coordinador del perfil de calidad.
8. Consultor de evaluación de Práctica Profesional.	Esta persona puede ser un interesado en consultar información referente a las evaluaciones de la asignatura Práctica Profesional. La misma puede ser:

	Profesor de cursos del perfil, Estudiante, Líder del proyecto de calidad.
9. Consultor de asistencia a curso.	Esta persona puede ser un interesado en consultar información referente a la asistencia al curso. La misma puede ser: Profesor de cursos del perfil o Estudiante.
10. Consultor de evaluaciones de curso.	Esta persona puede ser un interesado en consultar información referente a evaluaciones de un estudiante. La misma puede ser: Profesor de cursos del perfil, Estudiante o el Coordinador del perfil de calidad.
11. Consultor de datos del profesor.	Esta persona puede ser un interesado en consultar información referente a los datos de un profesor del perfil de calidad. La misma puede ser: Coordinador del perfil de calidad o Líder del proyecto de calidad.
12. Consultor de estructura del programa de asignatura.	Esta persona puede ser un interesado en consultar información referente a la estructura del programa de una asignatura. La misma puede ser: Profesor de cursos del perfil, Coordinador del perfil de calidad o Líder del proyecto de calidad.
13. Consultor de estado de completamiento de perfil.	Esta persona puede ser un interesado en consultar información referente al estado de completamiento del perfil de un estudiante. La misma puede ser: Profesor de cursos del perfil, Coordinador del perfil de calidad o Líder del proyecto de calidad.
14. Consultor de datos de curso.	Esta persona puede ser un interesado en consultar información referente a los datos del curso. La misma puede ser: Profesor de cursos del perfil, Coordinador del perfil de calidad, Estudiante o Líder del proyecto de calidad.

2.7.7 Diagrama de casos de uso del sistema

En el siguiente diagrama de caso de uso del sistema se representan las funcionalidades principales del sistema para ganar en espacio y mejorar la calidad de las imágenes mostradas, con esta representación se tiene una visión general del sistema a nivel de Casos de Uso.

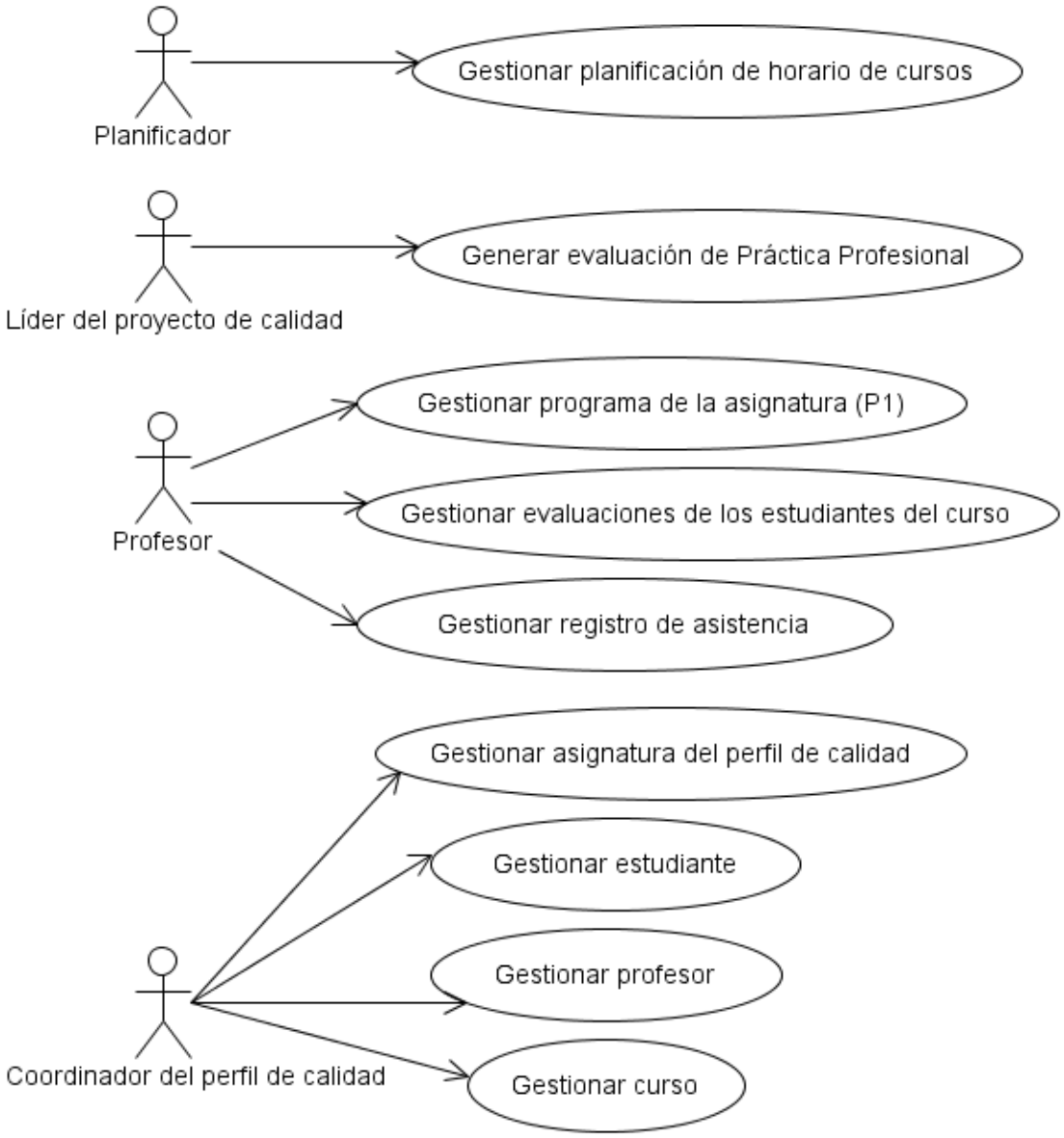


Figura 6: Diagrama de casos de uso del sistema.

En la figura anterior no se encuentran los siguientes casos de uso, por lo que se plantea al inicio de este epígrafe, no obstante más adelante se podrá encontrar la realización de los mismos:

- CU Verificar estado de completamiento del perfil.
- CU Consultar datos de los cursos.
- CU Consultar datos del horario.
- CU Consultar evaluación de Práctica Profesional.
- CU Conocer asistencia del estudiante.
- CU Conocer evaluación del estudiante.
- CU Conocer datos de un profesor.

- CU Consultar datos de la asignatura del perfil de calidad.
- CU Consultar estructura del programa de la asignatura (P1).

2.8 Descripción textual de los casos de uso del sistema

Tabla 6: Descripción de alto nivel del Caso de Uso del Sistema “Gestionar Programa de la asignatura (P1)”.

Caso de Uso:	Gestionar programa de la asignatura (P1).
Actores:	Profesor de cursos del perfil. (inicia)
Resumen:	El caso de uso se inicia cuando se solicita insertar, modificar, eliminar, mostrar o exportar a extensión .pdf, el programa de la asignatura seleccionada. En cada uno de los casos el sistema permite realizar las acciones correspondientes, siempre que los datos solicitados sean correctos, de lo contrario se muestran mensajes específicos según el error cometido.

Tabla 7: Descripción de alto nivel del Caso de Uso del Sistema “Gestionar asignatura del perfil de calidad”.

Caso de Uso:	Gestionar asignatura del perfil de calidad.
Actores:	Coordinador del perfil de calidad. (inicia)
Resumen:	El caso de uso se inicia cuando el actor solicita insertar, modificar, eliminar o mostrar una asignatura del perfil de calidad. En cada uno de los casos el sistema permite realizar las acciones correspondientes, siempre que los datos solicitados sean correctos, de lo contrario se muestran mensajes específicos según el error cometido.

Tabla 8: Descripción de alto nivel del Caso de Uso del Sistema “Verificar estado de completamiento del perfil”.

Caso de Uso:	Verificar estado de completamiento del perfil.
Actores:	Consultor de estado de completamiento de perfil (inicia)
Resumen:	El caso de uso se inicia cuando el actor solicita verificar el estado de completamiento del perfil. En este caso el sistema le muestra todos los cursos que el estudiante tiene acreditado hasta la fecha, en caso de no tener ningún curso acreditado se mostrará un mensaje específico informando que el estudiante no cuenta con cursos acreditados.

Tabla 9: Descripción de alto nivel del Caso de Uso del Sistema “Gestionar profesor”.

Caso de Uso:	Gestionar profesor.
Actores:	Coordinador del perfil de calidad. (inicia)
Resumen:	El caso de uso se inicia cuando el actor solicita agregar, modificar, mostrar o eliminar un profesor de cursos del perfil de calidad. En cada uno de los casos el sistema permite realizar las acciones correspondientes, siempre que los datos solicitados sean correctos, de lo contrario se muestran mensajes específicos según el error cometido.

Tabla 10: Descripción de alto nivel del Caso de Uso del Sistema “Gestionar estudiante”.

Caso de Uso:	Gestionar estudiante.
Actores:	Coordinador del perfil de calidad. (inicia)
Resumen:	El caso de uso se inicia cuando el actor solicita agregar, modificar o eliminar un estudiante. En cada uno de los casos el sistema permite realizar las acciones correspondientes, siempre que los datos solicitados sean correctos, de lo contrario se muestran mensajes específicos según el error cometido.

Tabla 11: Descripción de alto nivel del Caso de Uso del Sistema “Gestionar evaluaciones de los estudiantes del curso”.

Caso de Uso:	Gestionar evaluaciones de los estudiantes del curso.
Actores:	Profesor de cursos. (inicia)
Resumen:	El caso de uso se inicia cuando el actor solicita agregar, modificar, mostrar o eliminar evaluaciones de un estudiante. En cada uno de los casos el sistema permite realizar las acciones correspondientes, siempre que los datos solicitados sean correctos, de lo contrario se muestran mensajes específicos según el error cometido.

Tabla 12: Descripción de alto nivel del Caso de Uso del Sistema “Gestionar registro de asistencia”.

Caso de Uso:	Gestionar registro de asistencia.
Actores:	Profesor de cursos del perfil. (inicia)
Resumen:	El caso de uso se inicia cuando el actor solicita agregar, modificar,

	mostrar o eliminar una asistencia del estudiante. En cada uno de los casos el sistema permite realizar las acciones correspondientes, siempre que los datos solicitados sean correctos, de lo contrario se muestran mensajes específicos según el error cometido.

Tabla 13: Descripción de alto nivel del Caso de Uso del Sistema “Gestionar curso”.

Caso de Uso:	Gestionar curso.
Actores:	Coordinador del perfil de calidad. (inicia)
Resumen:	El caso de uso se inicia cuando el actor solicita agregar estudiantes al curso, agregar un profesor al curso, agregar asignatura a impartir al curso, cancelar matrícula de estudiantes al curso, cambiar profesor del curso, mostrar matrícula del curso o imprimir registro de control de asistencia y evaluaciones del curso. En cada uno de los casos el sistema permite realizar las acciones correspondientes, siempre que los datos solicitados sean correctos, de lo contrario se muestran mensajes específicos según el error cometido.

Tabla 14: Descripción de alto nivel del Caso de Uso del Sistema “Consultar datos de los cursos”.

Caso de Uso:	Consultar datos de los cursos.
Actores:	Consultor de datos de curso. (inicia)
Resumen:	El caso de uso se inicia cuando el actor solicita consultar información referente a los cursos del perfil de calidad. En este caso el sistema muestra toda la información referente a los cursos del perfil de calidad, de no existir algún curso creado se muestra un mensaje específico informando que no existen cursos.

Tabla 15: Descripción de alto nivel del Caso de Uso del Sistema “Generar evaluación de Práctica Profesional”.

Caso de Uso:	Generar evaluación de Práctica Profesional.
Actores:	Líder del proyecto de calidad. (inicia)
Resumen:	El caso de uso se inicia cuando el actor solicita generar una evaluación final de la asignatura del estudiante práctica profesional

	(PP). Para ello, el profesor en los diferentes cursos del perfil de calidad, debe haber evaluado al estudiante en distintas modalidades, como lo son por ejemplo, seminarios, talleres, evaluaciones sistemáticas y preguntas escritas, luego al finalizar el curso, se promedia entre las evaluaciones del estudiante en su trayectoria a lo largo del curso y la nota final que le otorga el profesor al estudiante una vez culminado el curso. Generándose de esta forma la evaluación final de la asignatura PP.

Tabla 16: Descripción de alto nivel del Caso de Uso del Sistema “Consultar datos del horario”.

Caso de Uso:	Consultar datos del horario.
Actores:	Consultor de datos del horario. (inicia)
Resumen:	El caso de uso se inicia cuando el actor solicita consultar datos del horario docente. Para ello, el usuario busca en el sistema en el módulo de capacitación la opción “planificación de cursos” donde se detalla ahí el local, fecha y hora de los diferentes cursos del perfil a desarrollar. En el caso que no exista cursos en el horario se mostrará un mensaje informando que no existen cursos en el horario.

Tabla 17: Descripción de alto nivel del Caso de Uso del Sistema “Consultar evaluación de Práctica Profesional”.

Caso de Uso:	Consultar evaluación de Práctica Profesional
Actores:	Consultor de evaluación de Práctica Profesional. (inicia)
Resumen:	El caso de uso se inicia cuando el actor solicita consultar evaluación de la asignatura Práctica Profesional. Para ello, el usuario busca en el sistema en el módulo de Capacitación, información referente a la nota final de la asignatura Práctica Profesional. En el caso que la nota aún no esté generada se mostrará un mensaje informando que el estudiante aún no cuenta con la nota final de la asignatura.

Tabla 18: Descripción de alto nivel del Caso de Uso del Sistema “Conocer asistencia del estudiante”.

Caso de Uso:	Conocer asistencia del estudiante
Actores:	Consultor de asistencia a curso. (inicia)

Resumen:	El caso de uso se inicia cuando el actor solicita conocer la asistencia que del estudiante a cursos del perfil de calidad. Para ello, el actor busca en el sistema en el módulo de Capacitación, donde podrá encontrar información referente a la asistencia del estudiante. En el control de registro de asistencia del profesor, donde solo podrá conocer el estado de asistencia del estudiante seleccionado.

Tabla 19: Descripción de alto nivel del Caso de Uso del Sistema “Conocer evaluación del estudiante”.

Caso de Uso:	Conocer evaluación del estudiante
Actores:	Consultor de evaluaciones de curso. (inicia)
Resumen:	El caso de uso se inicia cuando el actor solicita conocer el estado de evaluación de un estudiante. Para ello, el actor busca en el sistema el módulo de Capacitación, donde podrá encontrar información referente a las evaluaciones del estudiante seleccionado. En el control de registro de evaluaciones del profesor, donde solo podrá conocer el estado de evaluaciones del mismo.

Tabla 20: Descripción de alto nivel del Caso de Uso del Sistema “Conocer datos de un profesor”.

Caso de Uso:	Conocer datos de un profesor
Actores:	Consultor de datos del profesor. (inicia)
Resumen:	El caso de uso se inicia cuando el actor solicita conocer datos de un profesor. Para ello, el actor busca en el sistema en el módulo de Capacitación, donde podrá encontrar información referente a los profesores de cursos del perfil de calidad de la Facultad 8. En caso de no existir profesores de cursos, se mostrará un mensaje informando que no existen profesores disponibles para impartir cursos.

Tabla 21: Descripción de alto nivel del Caso de Uso del Sistema “Consultar datos de la asignatura del perfil de calidad”.

Caso de Uso:	Consultar datos de la asignatura del perfil de calidad

Actores:	Consultor de datos de asignatura. (inicia)
Resumen:	El caso de uso se inicia cuando el actor solicita consultar datos de una asignatura del perfil de calidad. Para ello, el usuario busca en el sistema el módulo de Capacitación, donde podrá encontrar información referente a las asignaturas que conforman el perfil de calidad.

Tabla 22: Descripción de alto nivel del Caso de Uso del Sistema "Consultar estructura del programa de la asignatura (P1)".

Caso de Uso:	Consultar estructura del programa de la asignatura (P1)
Actores:	Consultor de estructura del programa de la asignatura. (inicia)
Resumen:	El caso de uso se inicia cuando el actor solicita consultar estructura del programa de la asignatura del perfil de calidad. Para ello, el actor busca en el sistema el módulo de Capacitación, donde podrá encontrar información referente a las asignaturas que conforman el perfil de calidad y la estructura de la misma.

Tabla 23: Descripción de alto nivel del Caso de Uso del Sistema "Gestionar planificación de horario de cursos".

Caso de Uso:	Gestionar planificación de horario de cursos
Actores:	Planificador. (inicia)
Resumen:	El caso de uso inicia cuando el planificador consulta la solicitud de capacitación para poder realizar diferentes acciones como insertar, mostrar, modificar o eliminar cursos al horario docente. En este caso el sistema muestra toda la información referente a los cursos del perfil de calidad con que cuenta el horario, de no existir algún curso en el mismo se muestra un mensaje específico informando que no existen cursos en el horario. Además el sistema permitirá imprimir o exportar a extensión .pdf la planificación del horario.

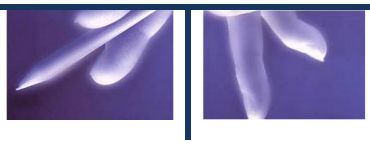
2.9 Conclusiones Parciales

A modo de conclusiones parciales del capítulo, se puede decir que el principal objetivo que persigue el mismo es abordar la situación problémica que dio origen a esta investigación y las principales características del sistema. También se estudiaron todos los procesos del negocio que forman parte de esta investigación. No puede dejar de mencionarse la realización del análisis crítico de los procesos propios presentes en

el campo de acción. Además quedaron definidos claramente los trabajadores y actores que intervienen en el negocio y sistema, así como las diferentes actividades y entidades con que interactúan los mismos, se desarrolló también el diagrama de clases del modelo de objetos.

En el subtema referente al sistema se definieron los requerimientos funcionales y no funcionales. También se describieron los actores que interactúan directamente con el sistema, los casos de uso a automatizar y se mostró el diagrama de caso de uso del sistema.

CAPÍTULO III:
ANÁLISIS Y DISEÑO DEL SISTEMA



3.1 Introducción

En el presente capítulo se traducen los requerimientos a una especificación que describe cómo implementar el sistema. El análisis consiste en obtener una visión del sistema, de modo que sólo se interesa por los requerimientos funcionales. El diseño es un refinamiento del análisis que tiene en cuenta los requerimientos no funcionales, o sea cómo cumple el sistema sus objetivos. El diseño debe ser suficiente para que el sistema pueda ser implementado sin ambigüedades.

3.2 Análisis

El proceso de análisis consiste en obtener una visión del sistema que permita entender qué hay que hacer, por lo que sólo se interesa por los requisitos funcionales. Se identifican las clases que describen la realización de los Casos de Uso, los atributos y las relaciones entre ellas, necesarias para la construcción del modelo de análisis.

3.2.1 Diagrama de Clases del Análisis

En este epígrafe se definen las siguientes clases del análisis:

1. Clase Entidad: Modelan información que posee larga vida y que es a menudo persistente.
2. Clase Controladora: Coordinan la realización de uno o unos pocos casos de uso coordinado las actividades de los objetos que implementan la funcionalidad del caso de uso.
3. Clase Interfaz: Modelan la interacción entre el sistema y sus actores.

A continuación se muestran los diagramas de clases del análisis para cada uno de los casos de uso del sistema del Módulo Capacitación. ´

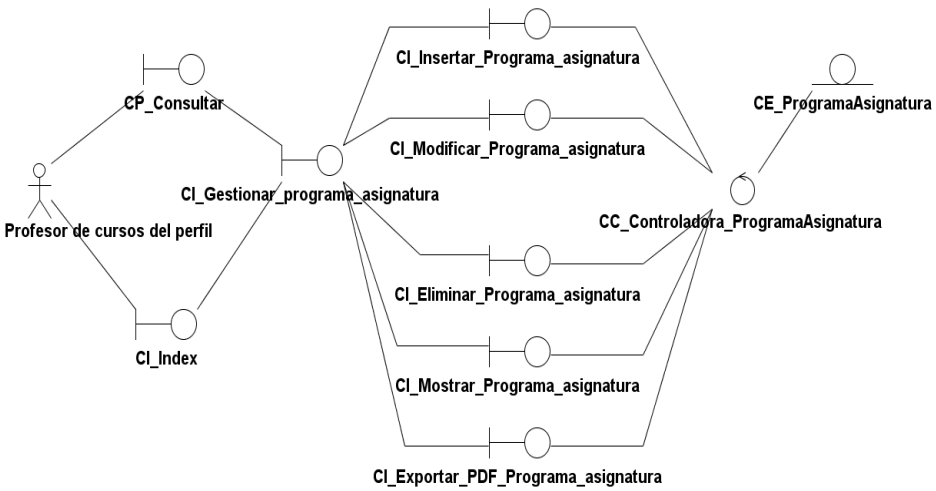


Figura 7: DCA CU “Gestionar programa de la asignatura”.

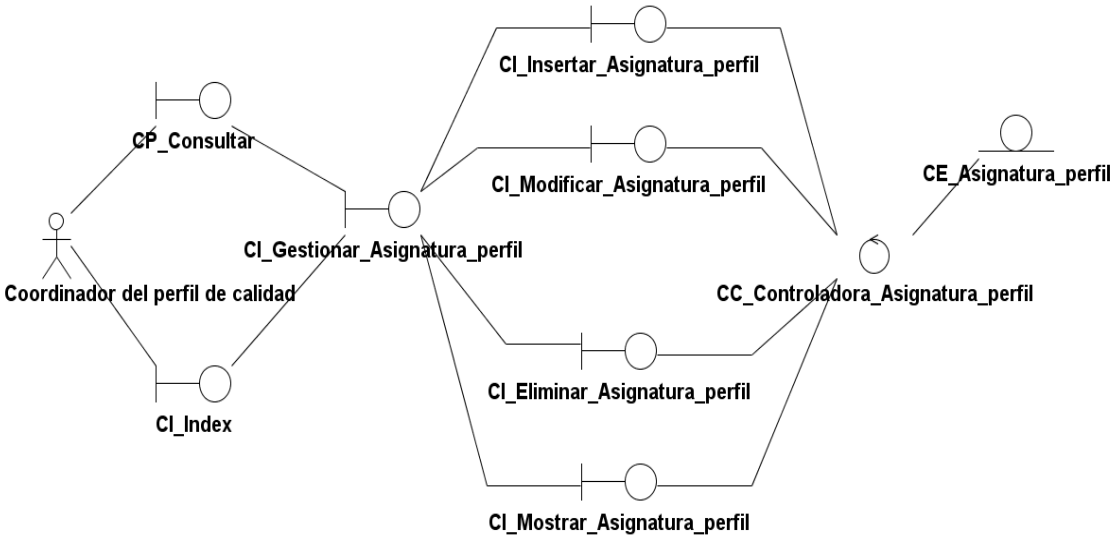


Figura 8: DCA CU "Gestionar asignatura del perfil de calidad".

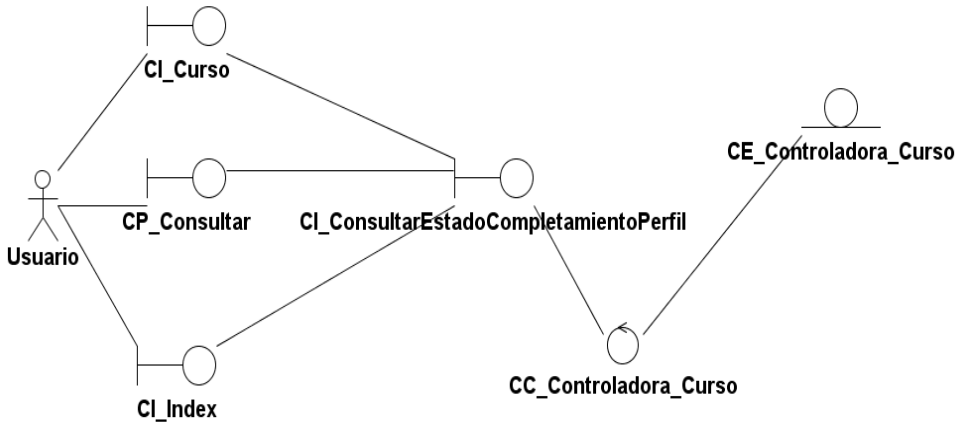


Figura 9: DCA CU "Verificar estado de completamiento del perfil".

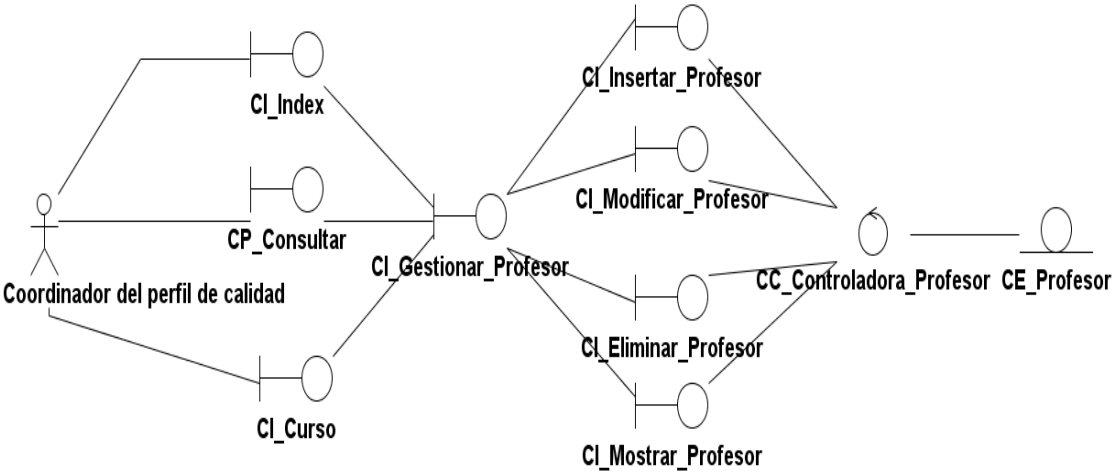


Figura 10: DCA CU "Gestionar Profesor".

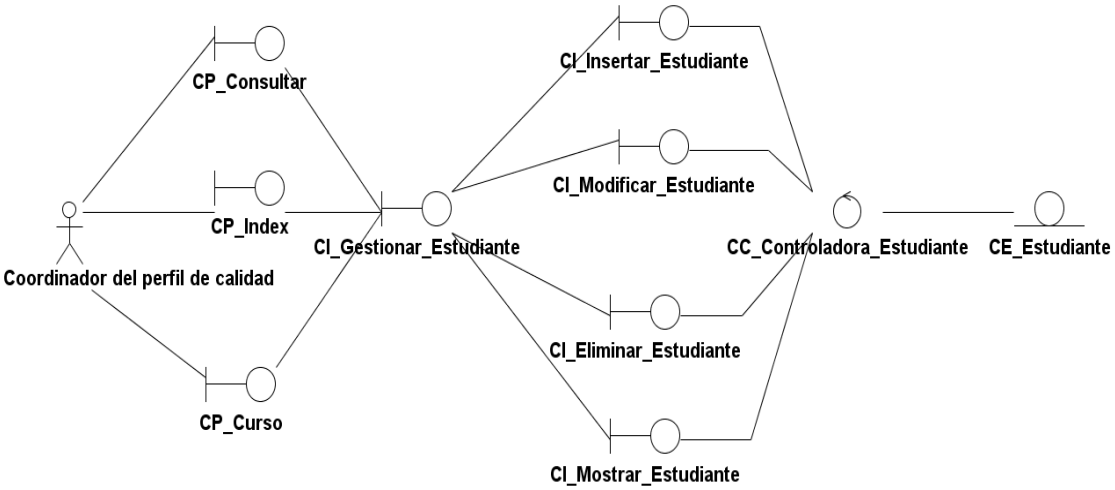


Figura 11: DCA CU “Gestionar Estudiante”.

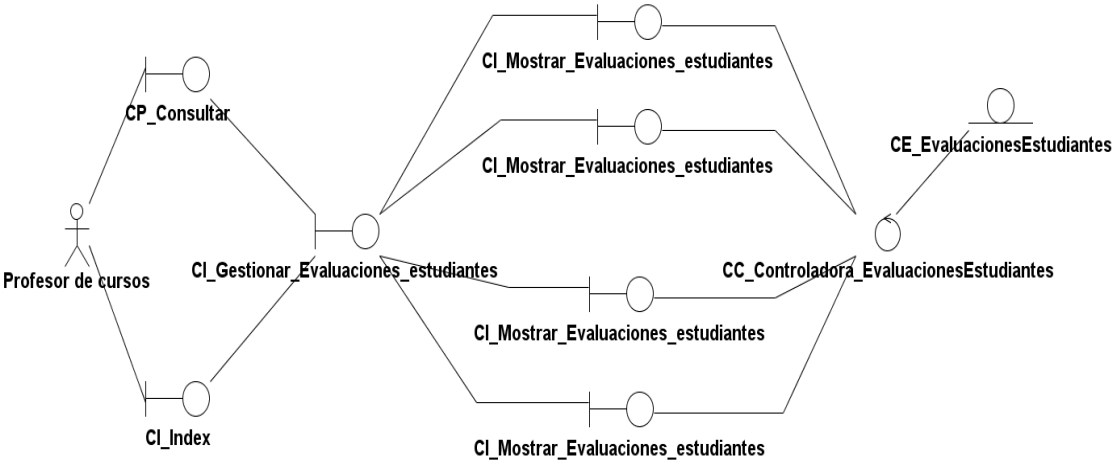


Figura 12: DCA CU “Gestionar evaluaciones de los estudiantes del curso.”.

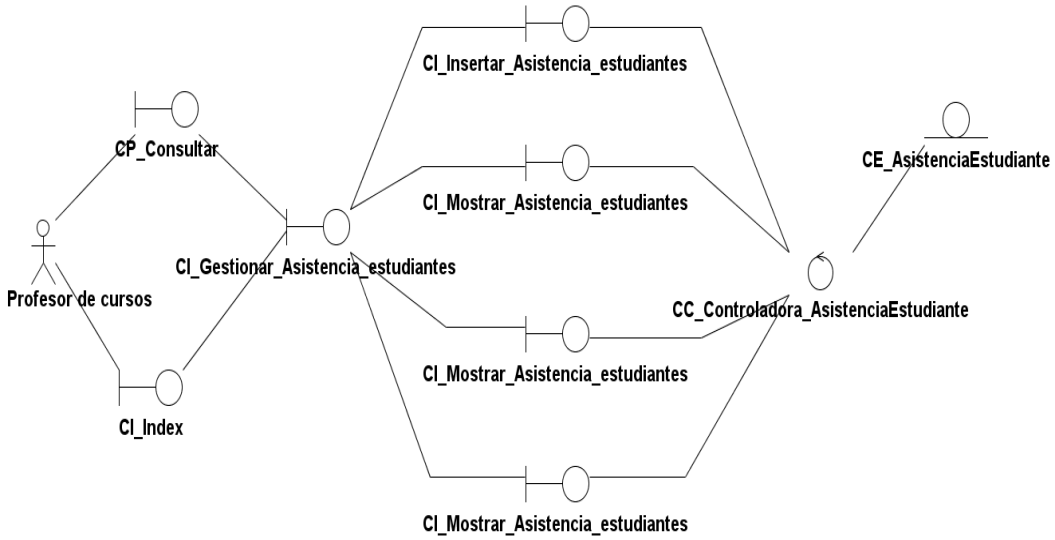


Figura 13: DCA CU “Gestionar registro de asistencia”.

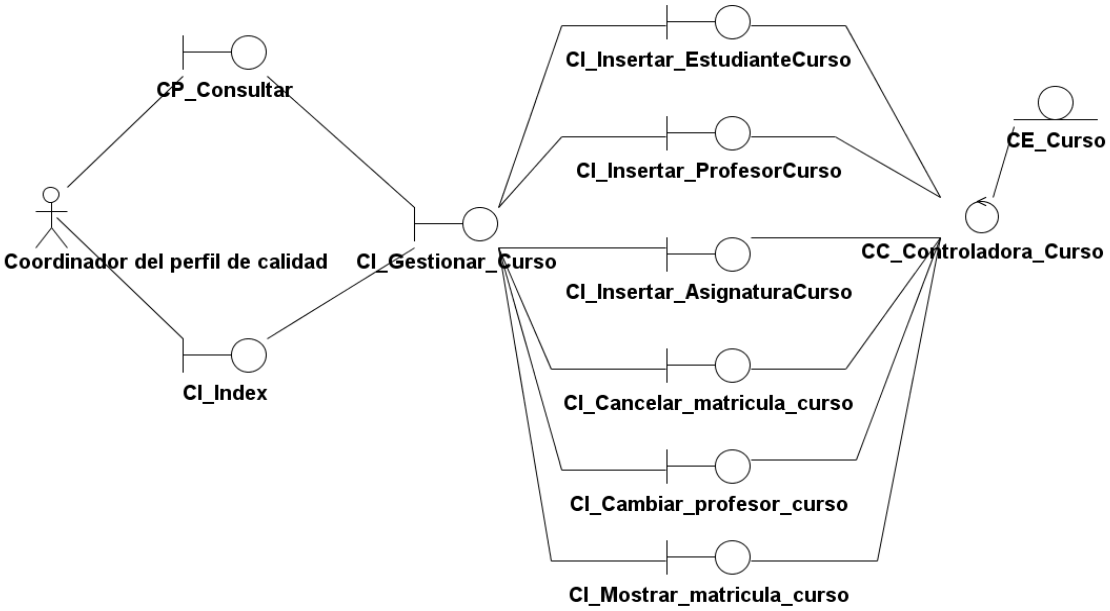


Figura 14: DCA CU “Gestionar curso”.

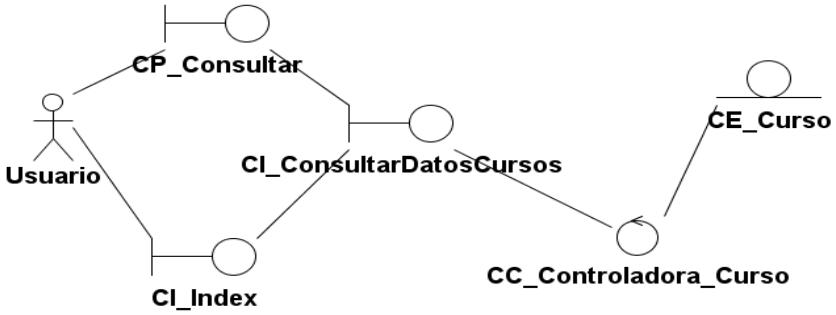


Figura 15: DCA CU “Consultar datos de cursos”.

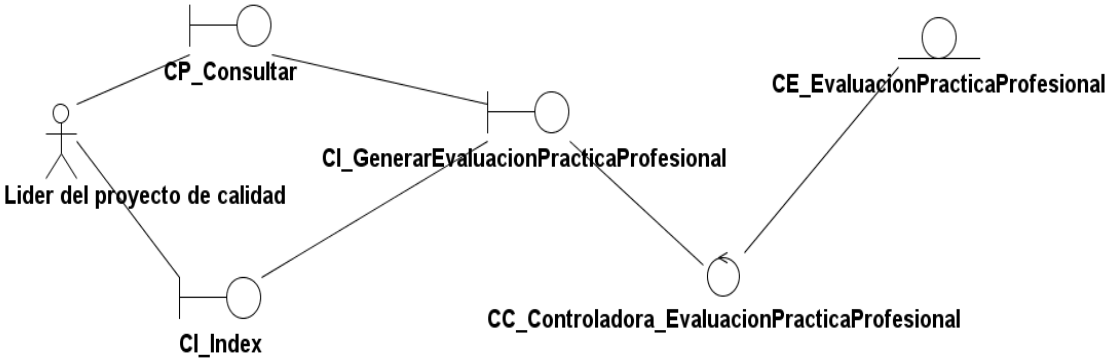


Figura 16: DCA CU “Generar evaluación de Práctica Profesional”.

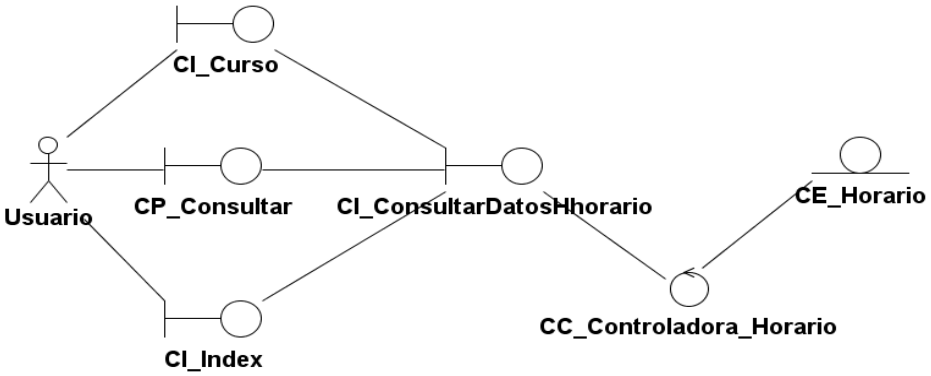


Figura 17: DCA CU "Consultar datos del horario".

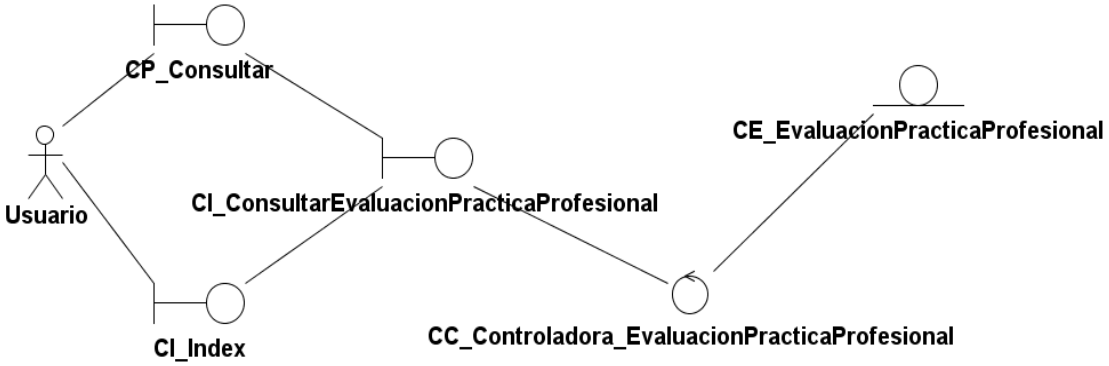


Figura 18: DCA CU "Consultar evaluación de Práctica Profesional".

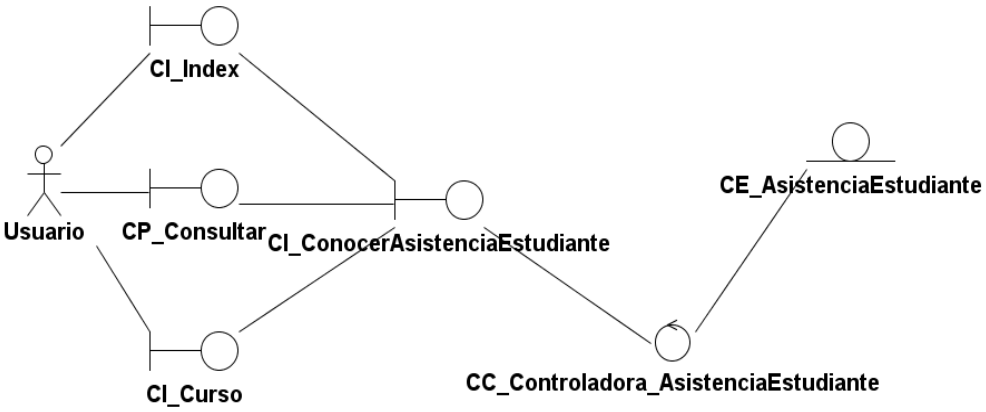


Figura 19: DCA CU "Conocer asistencia del estudiante".

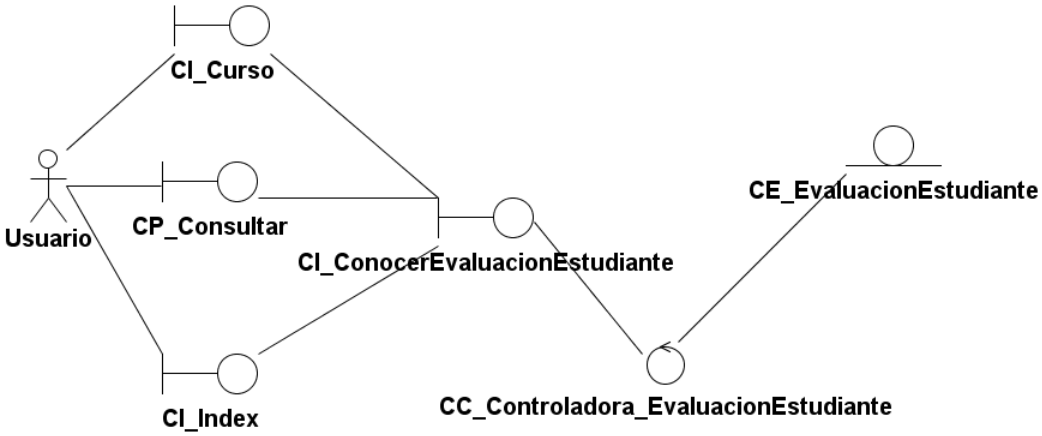


Figura 20: DCA CU “Conocer evaluación del estudiante”.

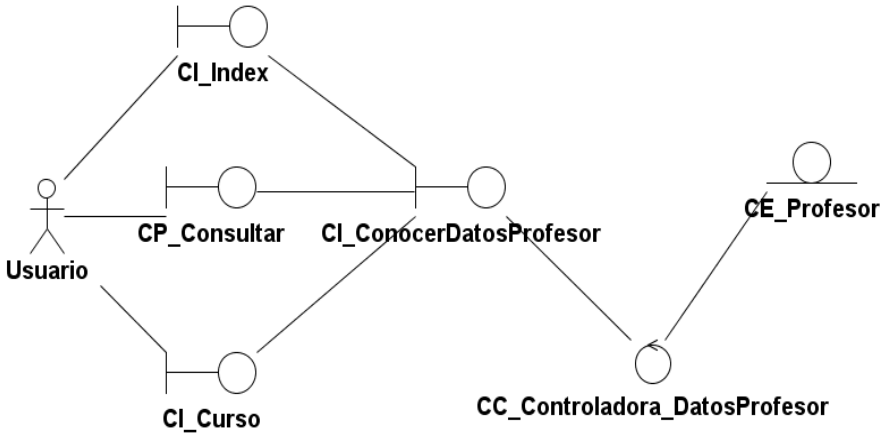


Figura 21: DCA CU “Conocer datos de un profesor”.

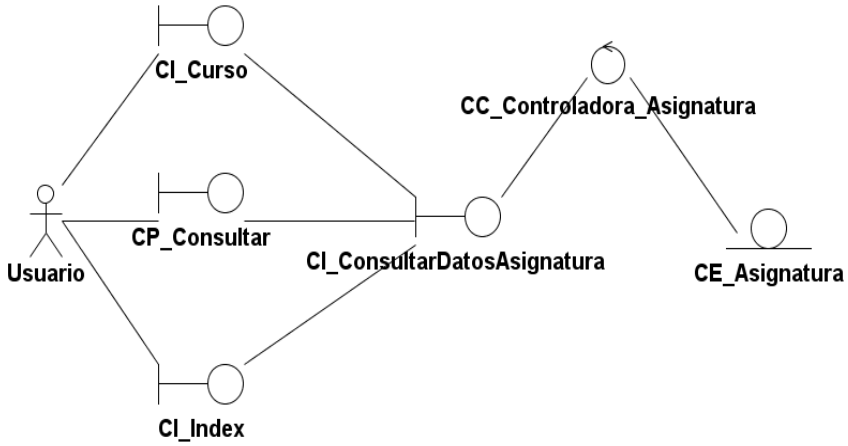


Figura 22: DCA CU “Consultar datos de la asignatura del perfil de calidad”.

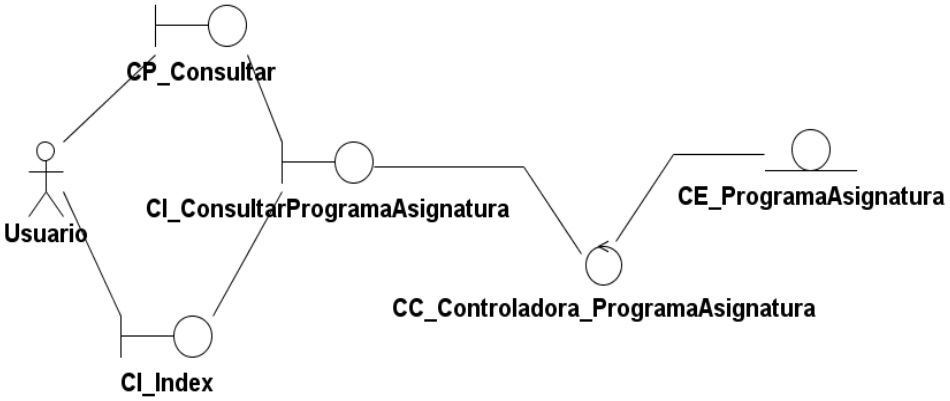


Figura 23: DCA CU "Consultar estructura del programa de la asignatura".

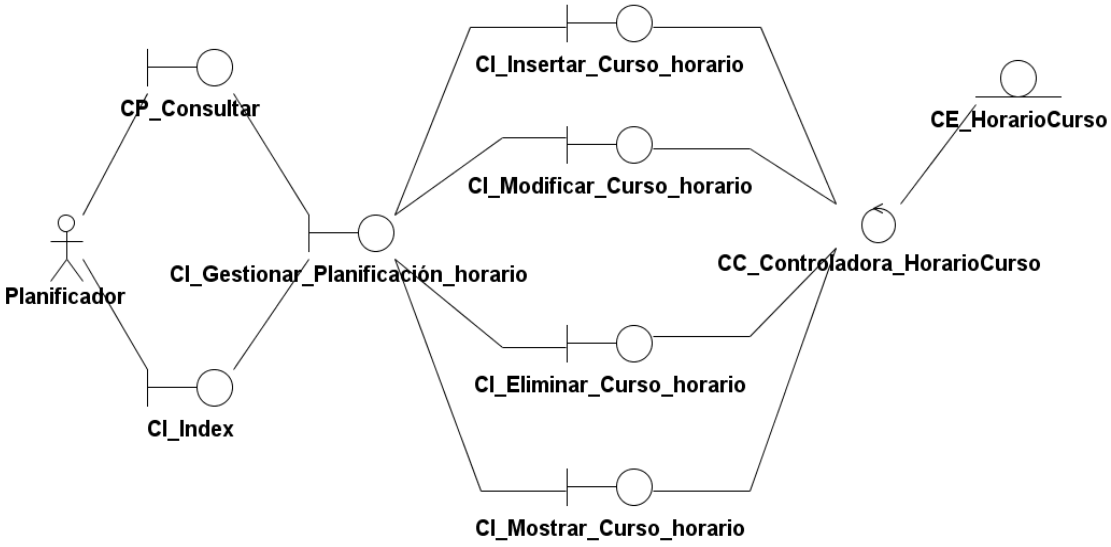


Figura 24: DCA CU "Gestionar planificación de horario de cursos".

3.3 Arquitectura y patrones utilizados.

Los patrones de diseño se definen como un conjunto de soluciones a problemas que generalmente se encuentran en el diseño de un programa. Cada patrón explica cómo resolver un determinado problema, bajo determinadas circunstancias, y las ventajas y desventajas de su uso.

Los patrones ayudan a tener un lenguaje común con otros programadores (si se trabaja en grupos de programadores).

3.3.1 Patrones de diseño de software (Grasp).

Es importante usar patrones para el desarrollo de software, ya que estos proporcionan importantes características, las cuales se mencionan a continuación: (PANTOJA 2005)

Producción de desarrollo de software más resistente al cambio.

- Establece problemas Parejas-Solución.
- Ayudan a especificar interfaces.
- Reutilización del código.
- Uso de Documentación estándar.

Los Patrones de Software para la Asignación General de Responsabilidad (o patrones GRASP por sus siglas en ingles), describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades, además estos constituyen una descripción de un problema y la solución a la que se le da un nombre. (LARMAN 2006)
Al momento de especificar un patrón se debe tener en cuenta los siguientes elementos:

- Descripción.
- Escenario de Uso.
- Solución concreta.
- Las consecuencias de utilizar este patrón.
- Ejemplos de implementación.
- Lista de patrones relacionados.

Existen varios patrones GRASP, los seleccionados por sus características para el presente trabajo fueron Bajo Acoplamiento, Alta Cohesión, Experto, Creador, Controlador:

1. Bajo Acoplamiento: El acoplamiento es una medida de fuerza con que un elemento esta a, tiene conocimiento de, confía en, otros elementos. Este patrón es un principio que asigna la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades (validaciones, seguridad, etc.). Un error muy común es asignarle demasiada responsabilidad y alto nivel de acoplamiento con el resto de los componentes del sistema. (MORA 2003)
Solución: Asignar una responsabilidad para mantener bajo acoplamiento.
Problema: ¿Cómo dar soporte a una dependencia escasa y a un aumento de la reutilización?
El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo (o débil) acoplamiento no depende de muchas otras.
Una clase con alto (o fuerte) acoplamiento recurre a muchas otras. Este tipo de clases no es conveniente: presentan los siguientes problemas:

- Los cambios de las clases afines ocasionan cambios locales.
- Son más difíciles de entender cuando están aisladas.

- Son más difíciles de reutilizar porque se requiere la presencia de otras clases de las que dependen.

Debe haber pocas dependencias entre las clases. Si todas las clases dependen de todas ¿Cuál software podemos extraer de un modo independiente y reutilizarlo en otro proyecto? Para determinar el nivel de acoplamiento de clases, son muy buenos los diagramas de colaboración de UML. Uno de los principales síntomas de un mal diseño y alto acoplamiento es una herencia muy profunda. Siempre hay que considerar las ventajas de la delegación respecto de la herencia. (LARMAN 2006)

Beneficios:

- No se afectan por cambios de otros componentes.
- Fáciles de entender por separado.
- Fáciles de reutilizar.

2. Alta Cohesión: La cohesión es una medida de la fuerza con la que se relacionan las clases y el grado de focalización de las responsabilidades de un elemento. Cada elemento de nuestro diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable, una clase con baja cohesión hace muchas cosas no relacionadas o hace demasiado trabajo. (MORA 2003)

Solución: Asignar una responsabilidad de modo que la cohesión siga siendo alta.

Problema: ¿Cómo mantener la complejidad dentro de límites manejables?

Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme.

Una clase con baja cohesión hace muchas cosas no afines o un trabajo excesivo. No conviene este tipo de clases pues presentan los siguientes problemas:

- Son difíciles de comprender.
- Son difíciles de reutilizar.
- Son difíciles de conservar.
- Son delicadas: las afectan constantemente los cambios.

Las clases con baja cohesión a menudo representan un alto grado de abstracción o han asumido responsabilidades que deberían haber delegado a otros objetos.

Cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto – identificable. (LARMAN 2006)

Beneficios:

- Mejoran la claridad y facilidad con que se entiende el diseño
- Se simplifica el mantenimiento y las mejoras de funcionalidad

- A menudo se genera un bajo acoplamiento
- Soporta mayor capacidad de reutilización.

3. Creador: Asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad. El propósito fundamental de este patrón es encontrar un creador que debemos conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento. (MORA 2003)

Solución: Asignarle a la clase B la responsabilidad de crear una instancia de clase A en uno de los siguientes casos:

- B agrega los objetos A.
- B contiene los objetos A.
- B registra las instancias de los objetos A.
- B utiliza específicamente los objetos A.
- B es un creador de los objetos A.

Problema: ¿Quién debería ser responsable de crear una nueva instancia de alguna clase?

4. Controlador: Un controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. (MORA 2003)

Solución: Asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase que represente una de las siguientes opciones:

- El “sistema” global (controlador de fachada).
- La empresa u organización global (controlador de fachada).
- Algo en el mundo real que es activo y que pueda participar en la tarea (controlador de tareas).
- Un manejador artificial de todos los eventos del sistema de un caso de uso, generalmente denominados “manejador <NombreCasodeUso> (controlador de casos de uso).

Problema: ¿Quién debería encargarse de atender un evento del sistema?

Un evento del sistema es un evento de alto nivel generado por un actor externo; es un evento de entrada externa. Se asocia a operaciones del sistema: las que emite en respuesta a los eventos del sistema.

Beneficios

- Mayor potencial de los componentes reutilizables. Garantiza que la empresa o los procesos de dominio sean manejados por la capa de los objetos del dominio y no por la de la interfaz.

- Reflexionar sobre el estado del caso de uso. A veces es necesario asegurarse de que las operaciones del sistema sigan una secuencia legal o poder razonar sobre el estado actual de la actividad y las operaciones en el caso de uso subyacente.

5. Experto: Experto es un patrón que se usa más que cualquier otro al asignar responsabilidades; es un principio básico que suele ser útil en el diseño orientado a objetos. (MORA 2003)

Problema: ¿Cuál es el principio fundamental en virtud del cual se asignan las responsabilidades en el diseño orientado a objetos?

- Un modelo de clase puede definir docenas y hasta cientos de clases de software, y una aplicación tal vez requiera el cumplimiento de cientos o miles de responsabilidades.
- Si estas se asignan en forma adecuada, los sistemas tienden a ser más fáciles de entender, mantener y ampliar, y se nos presenta la oportunidad de reutilizar los componentes en futuras aplicaciones.

Solución: Asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad.

Beneficios:

- Se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Esto soporta un bajo acoplamiento, lo que favorece al hecho de tener sistemas más robustos y de fácil mantenimiento.
- El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clase “sencillas” y más cohesivas que son más fáciles de comprender y de mantener. Así se brinda soporte a una alta cohesión.

3.3.2 Patrón de arquitectura: Modelo Vista Controlador.

Para el diseño de aplicaciones con sofisticadas interfaces se utiliza el patrón de diseño Modelo-Vista-Controlador. La lógica de una interfaz de usuario cambia con más frecuencia que los almacenes de datos y la lógica de negocio. (LAGO 2007b)

El uso de este patrón es favorable ya que establece una separación entre los distintos tipos de componentes que presenta un sistema; permitiendo implementarlos a cada uno por separado. Además el mismo brinda la posibilidad de que si uno de los componentes tiene un mal funcionamiento internamente pueda sustituirse sin que se

vean afectados los demás. También permite que no haya alguna dependencia directa entre la Vista y el Modelo, esto es de gran beneficio ya que brinda un mayor soporte a los cambios realizados en las diferentes interfaces existentes; que tienden a cambiar con más frecuencia que la lógica del negocio. Es importante mencionar que con el uso de este patrón se puede añadir nuevas vistas sin afectar el Modelo, es decir, múltiples páginas en la aplicación web pueden emplear los mismos elementos del Modelo.

Elementos del patrón Modelo-Vista-Controlador utilizado en este trabajo:

- Modelo: datos y reglas de negocio.
- Vista: muestra la información del modelo al usuario.
- Controlador: gestiona las entradas del usuario.

El modelo es el responsable de:

- Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
- Define las reglas de negocio (la funcionalidad del sistema).
- Lleva un registro de las vistas y controladores del sistema.
- Si se está ante un modelo activo, notificará a las vistas los cambios que en los datos pueda producir un agente externo.

El controlador es responsable de:

- Recibe los eventos de entrada.
- Contiene reglas de gestión de eventos.

Las vistas son responsables de:

- Reciben datos del modelo y los muestra al usuario.
- Tienen un registro de su controlador asociado.
- Pueden dar el servicio de “Actualización ()”, para que sea invocado por el controlador o por el modelo.

3.3.3 PATRÓN FAÇADE (FACHADA).

El patrón façade (o Facade, o en español Fachada) trata de simplificar la interface entre dos sistemas o componentes de software ocultando un sistema complejo detrás de una clase que hace las veces de pantalla o fachada. La idea principal es la de ocultar todo lo posible la complejidad de un sistema, el conjunto de clases o componentes que lo forman, de forma que solo se ofrezca un (o unos pocos) punto de entrada al sistema tapado por la fachada. Una ventaja más de usar una clase fachada para comunicar las dos partes o componentes, es la de aislar los posibles cambios que se puedan producir en alguna de las partes. Si se cambia, por ejemplo, el medio

de comunicación o de almacenamiento de una de las partes, la otra, que por ejemplo hace la presentación, no tiene porque enterarse, y viceversa. (GORUS 2002)

Con este patrón ofrecemos un acceso sencillo y desacoplamos al máximo nuestro sistema cliente (el que accede a la fachada) de los sistemas ocultos. (SÁNCHEZ 2006)

Algunas ventajas que disfrutaremos en el propio desarrollo son: (SÁNCHEZ 2006)

- Facilitamos la utilización y comprensión (acompañando la interfaz con una documentación mínima) de los sistemas ocultados
- Los clientes se olvidan de toda la complejidad del negocio, sólo les importa los resultados obtenidos.
- Este patrón de diseño se puede implementar como una interfaz o como una clase abstracta, sin detallar los detalles de implementación (es justo lo que queremos ocultar).
- Es habitual que se implemente la fachada utilizando también el patrón Singleton (sólo se necesita una fachada).
- Como vemos es un patrón de diseño muy simple e intuitivo, que nace del sentido común y experiencia. Los beneficios que nos ofrece son enormes en comparación con su coste de implementación.

El valor que añade esta clase es el ofrecer a los clientes una forma única y simplificada de acceder a los servicios más generales del subsistema. Para ello, los clientes enviarán mensajes solo a la fachada, y esta se encargará de poner en funcionamiento la maquinaria del subsistema para conseguir el objetivo pretendido y devolver al cliente los resultados.

3.3.4 Patrón de acceso a datos (DAO).

El problema que viene a resolver este patrón es el de contar con diversas fuentes de datos (base de datos, archivos, servicios externos, etc). De tal forma que se encapsula la forma de acceder a la fuente de datos. Este patrón surge históricamente de la necesidad de gestionar una diversidad de fuentes de datos, aunque su uso se extiende al problema de encapsular no sólo la fuente de datos, sino además ocultar la forma de acceder a los datos. Se trata de que el software cliente se centre en los datos que necesita y se olvide de cómo se realiza el acceso a los datos o de cuál es la fuente de almacenamiento. (LAGO 2007a)

Las aplicaciones pueden utilizar el API JDBC para acceder a los datos de una base de datos relacional. Este API permite una forma estándar de acceder y manipular datos en una base de datos relacional. El API JDBC permite a las aplicaciones J2EE utilizar

sentencias SQL, que son el método estándar para acceder a tablas y vistas. La idea de este patrón es ocultar la fuente de datos y la complejidad del uso de JDBC a la capa de presentación o de negocio. Un DAO define la relación entre la lógica de presentación y empresa por una parte y por otra los datos. El DAO tiene un interfaz común, sea cual sea el modo y fuente de acceso a datos. (LAGO 2007a)

Algunas características de este útil patrón: (LAGO 2007a)

- No es imprescindible, pero en proyectos de cierta complejidad resulta útil que el DAO implemente un interfaz. De esta forma los objetos cliente tienen una forma unificada de acceder a los DAO.
- El DAO accede a la fuente de datos y la encapsula para los objetos clientes, entendiendo que oculta tanto la fuente como el modo (JDBC) de acceder a ella.
- El TransferObject encapsula una unidad de información de la fuente de datos. El ejemplo sencillo es entenderlo como un "bean de tabla", es decir, como una representación de una tabla de la base de datos, por lo que se representa las columnas de la tabla como atributos del TransferObject.
- Permite la transparencia.
- Permite una migración más fácil.
- Reduce la complejidad del código de los objetos de negocio.
- Centraliza todos los accesos a datos en una capa independiente.
- Añade una capa extra.

Cuando se realiza una aplicación de cierto alcance, surge la necesidad de obtener datos de varias fuentes distintas, como pueden ser una base de datos MySQL, PostgreSQL, Oracle, Informix, Sybase, SQL Server, BD orientada a objetos, ficheros planos, servidor LDAP, ficheros de texto, servicios externos, etc. El objetivo de este patrón es encapsular la fuente de datos, de manera que no necesitamos saber en cada momento la forma de acceso a los datos, de modo que podamos consumir los datos que maneje la aplicación sin preocuparnos de la fuente que suministra esa información, y además podemos modificar cada fuente de datos de forma independiente. (GP 2008)

3.4 Diseño.

El diseño es el centro de atención al final de la fase de elaboración y el comienzo de las iteraciones de construcción. El modelo de diseño está muy cercano al de implementación, lo que es natural para guardar y mantener el modelo de diseño a través del ciclo de vida completo del software. En el diseño se modela el sistema y

determina su forma para que soporte todos los requerimientos, incluyendo los no funcionales. Una entrada esencial en el diseño es el resultado del análisis, o sea el modelo de análisis, que proporciona una comprensión detallada de los requerimientos. Los propósitos del diseño se definen en transformar los requerimientos en un diseño de cómo el sistema debe ser, desarrollar una robusta arquitectura del sistema y adaptar el diseño para que se corresponda con el entorno de implementación, diseñando sus funcionalidades.

3.4.1 Diagramas de clases del diseño.

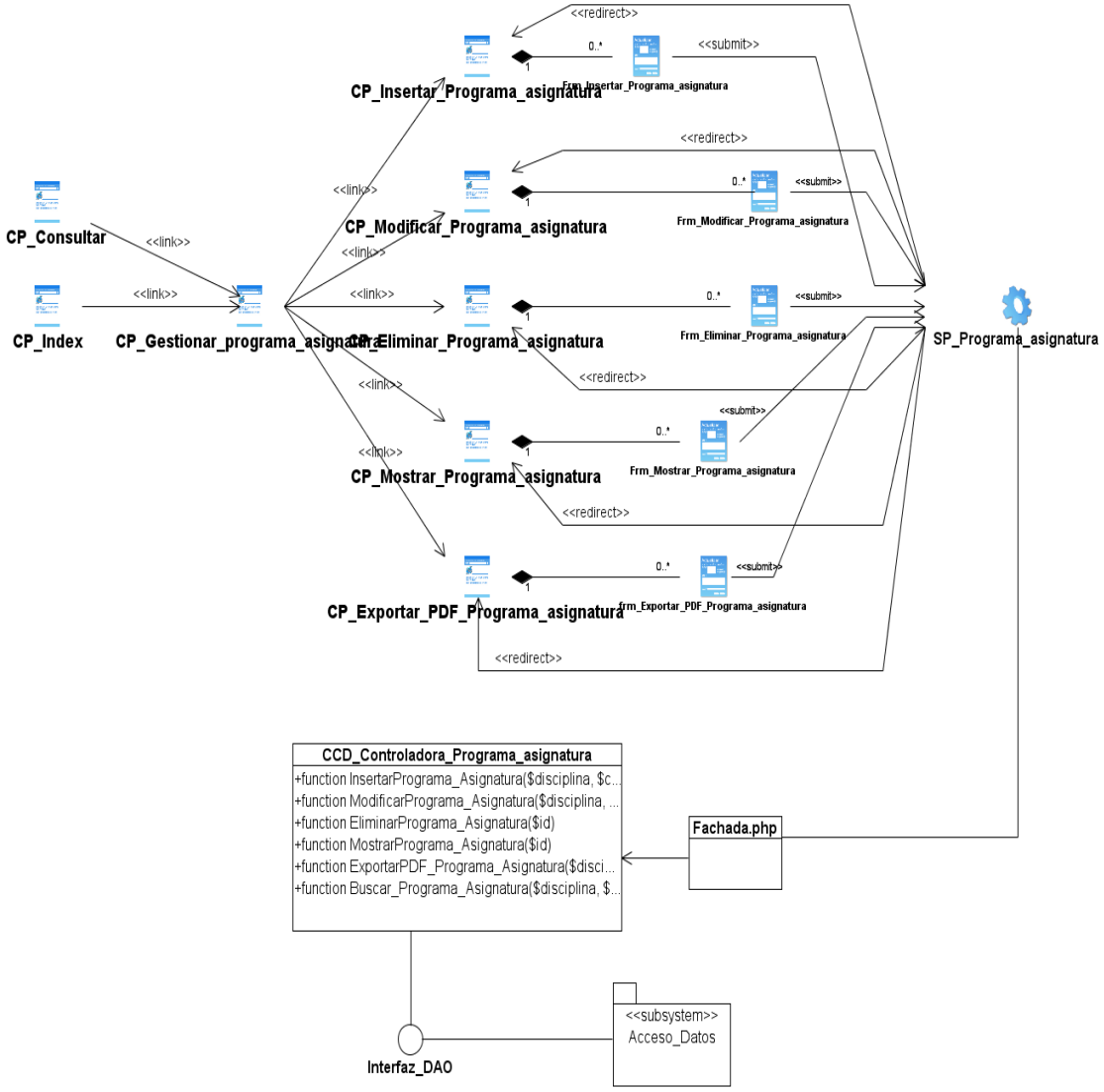


Figura 25: DCD CU “Gestionar programa de asignatura”.

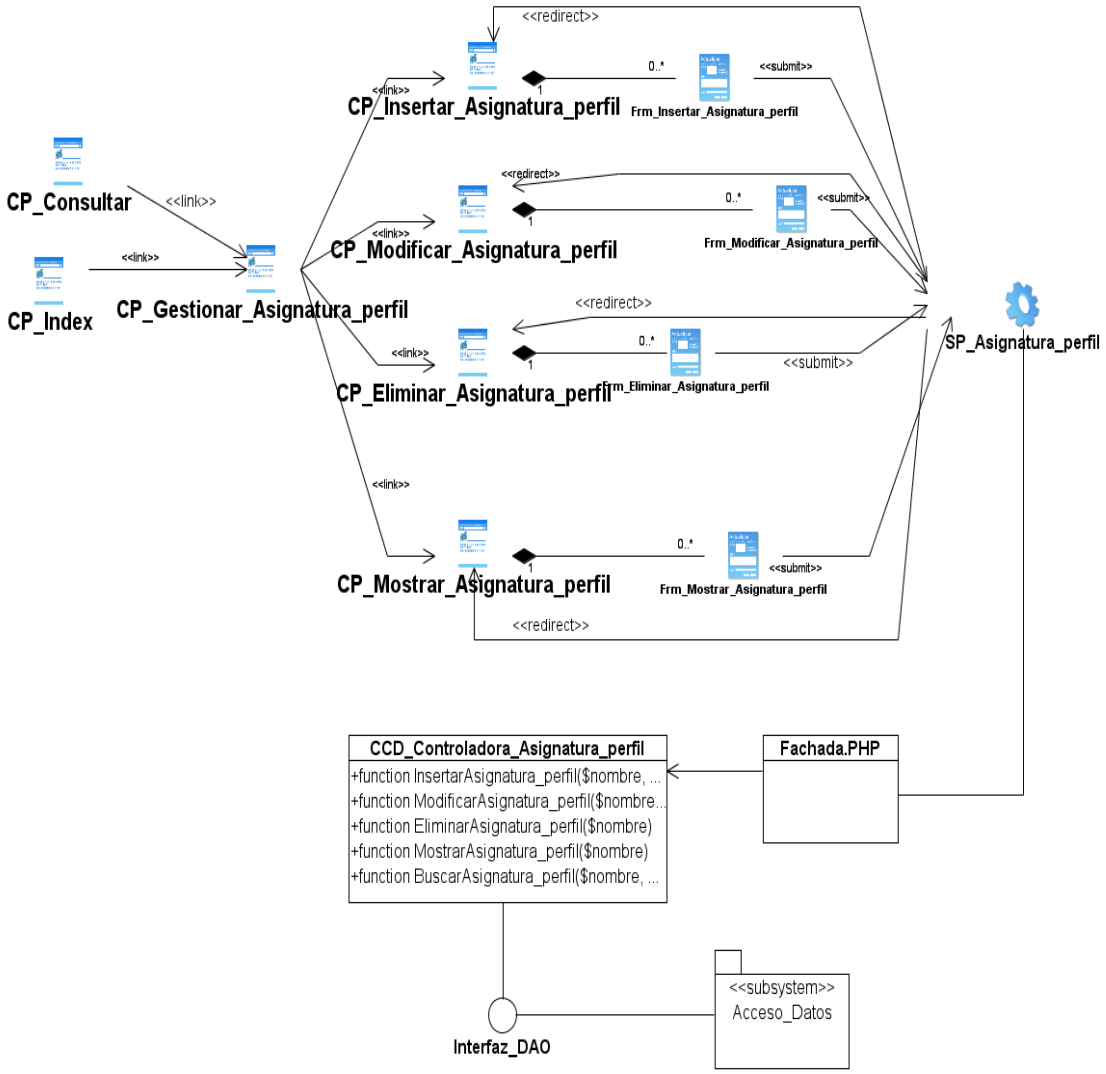


Figura 26: DCD CU “Gestionar asignatura del perfil de calidad”.

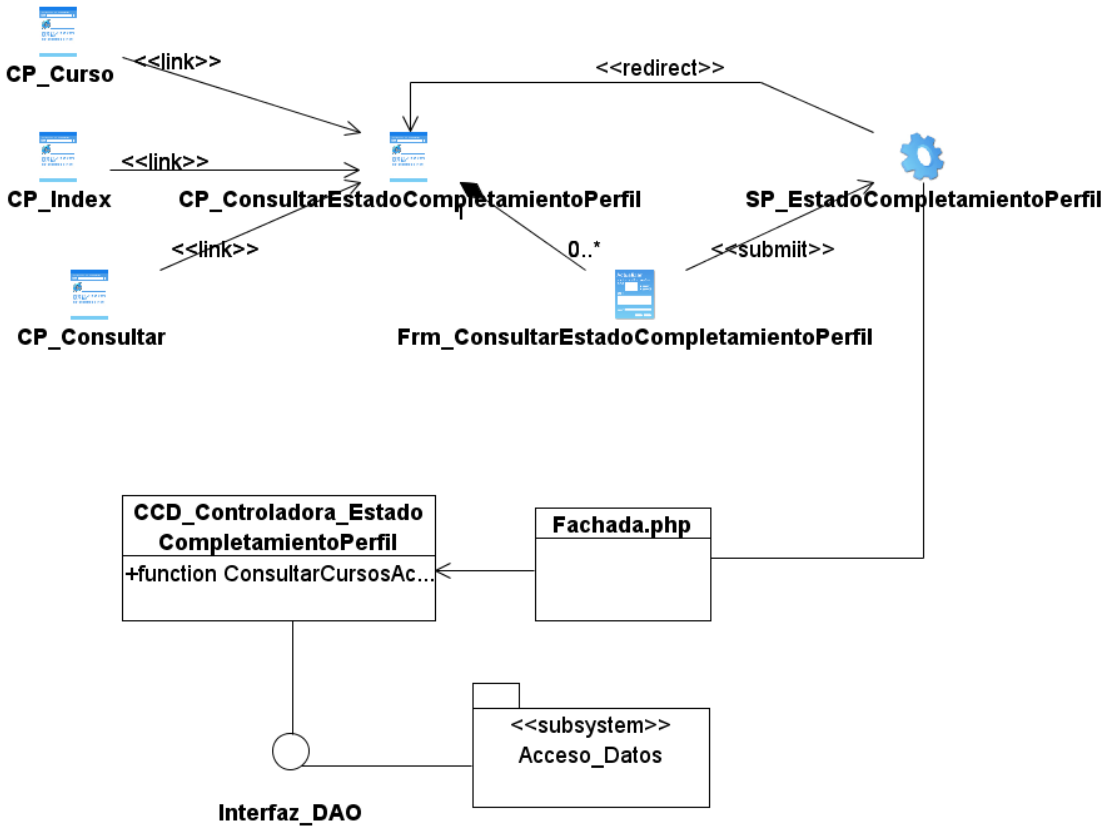


Figura 27: DCD CU “Verificar estado de completamiento del perfil”.

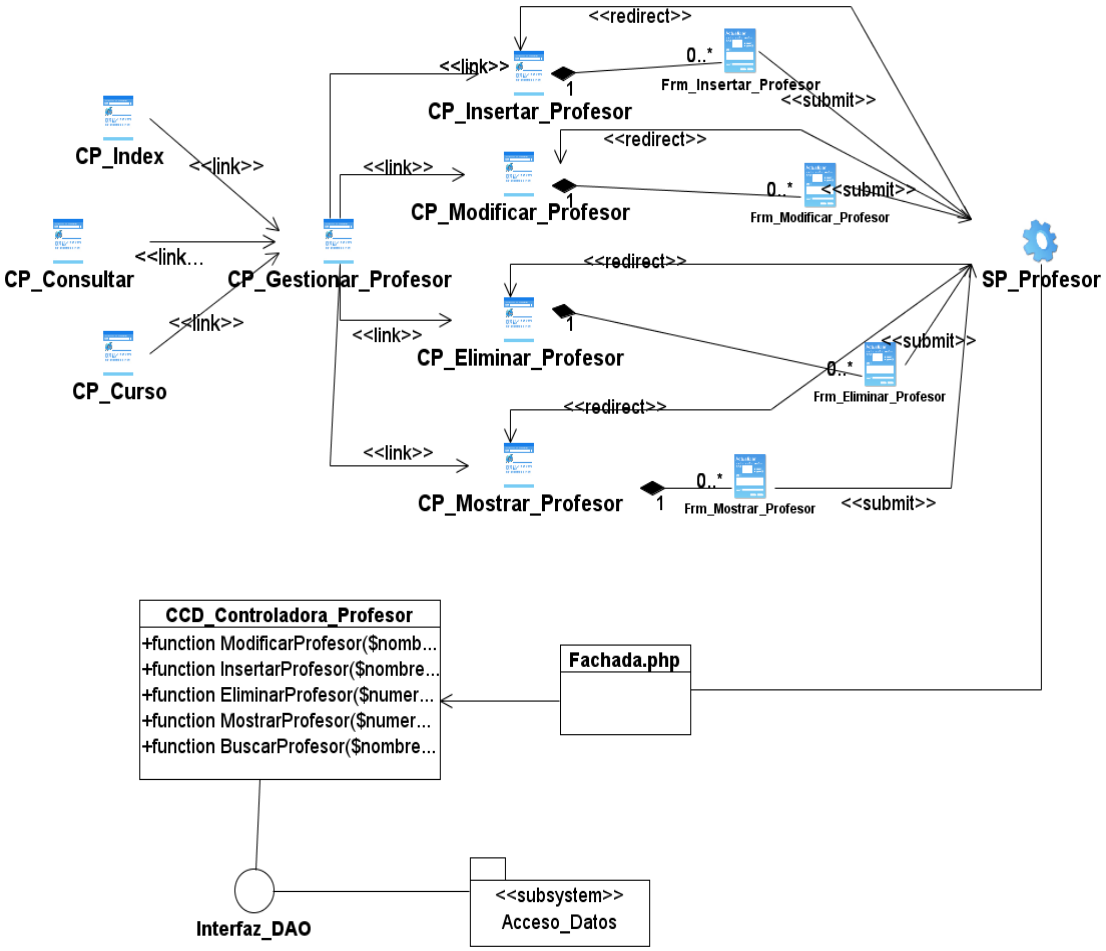


Figura 28: DCD CU “Gestionar profesor”.

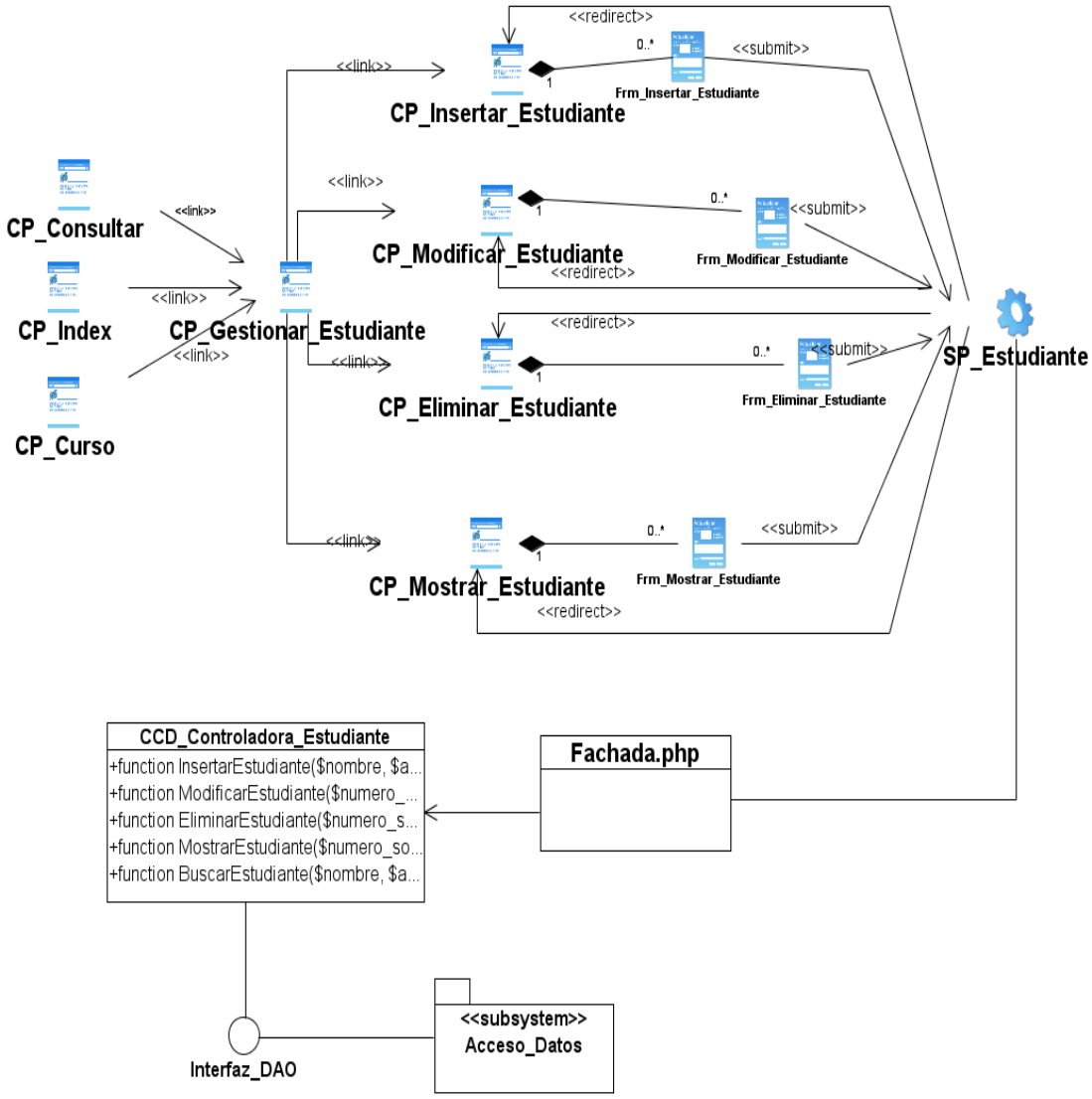


Figura 29: DCD CU “Gestionar estudiante”.

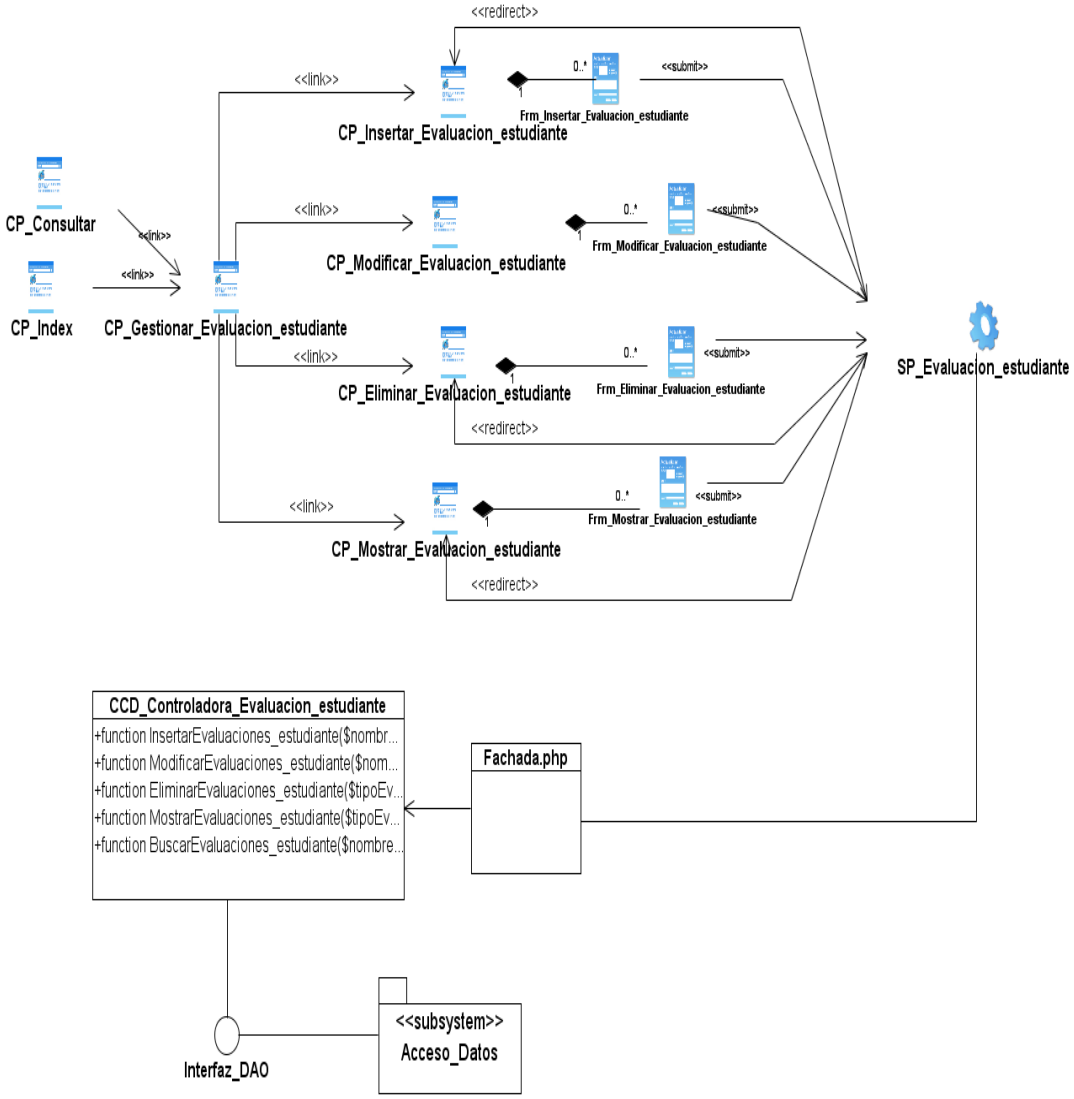


Figura 30: DCD CU “Gestionar evaluaciones de los estudiantes del curso”.

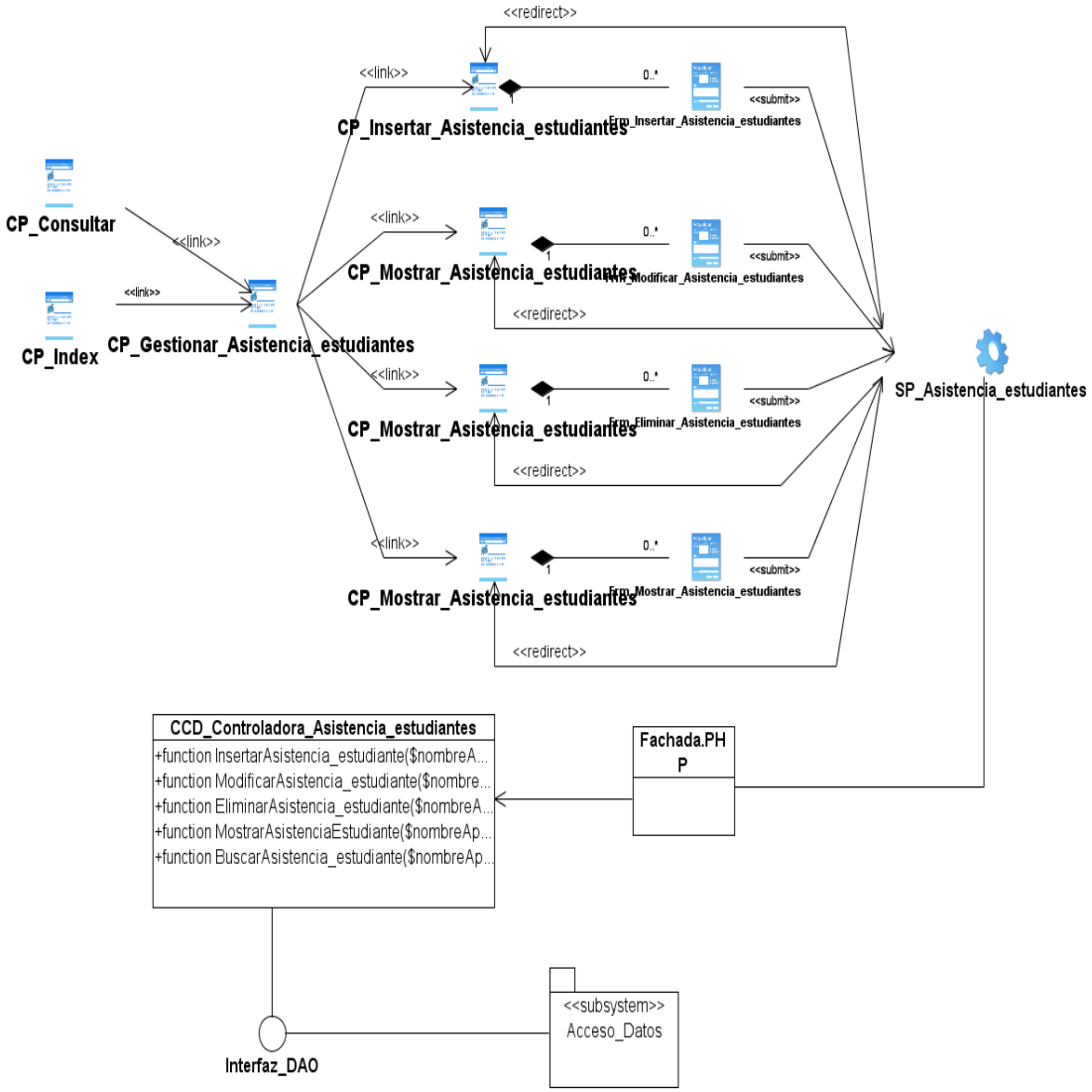


Figura 31: DCD CU “Gestionar Registro de Asistencia”.

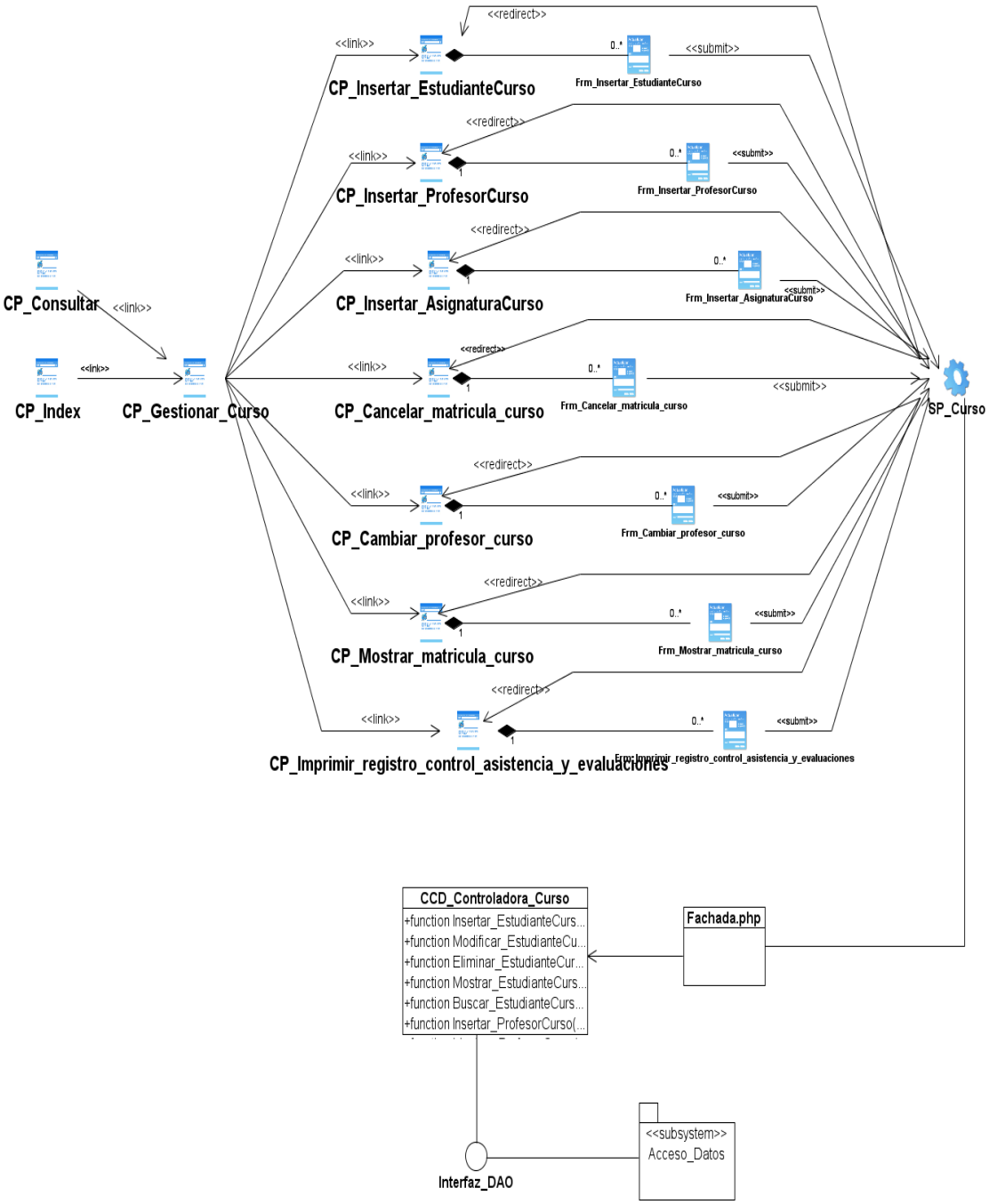


Figura 32: DCD CU “Gestionar curso”.

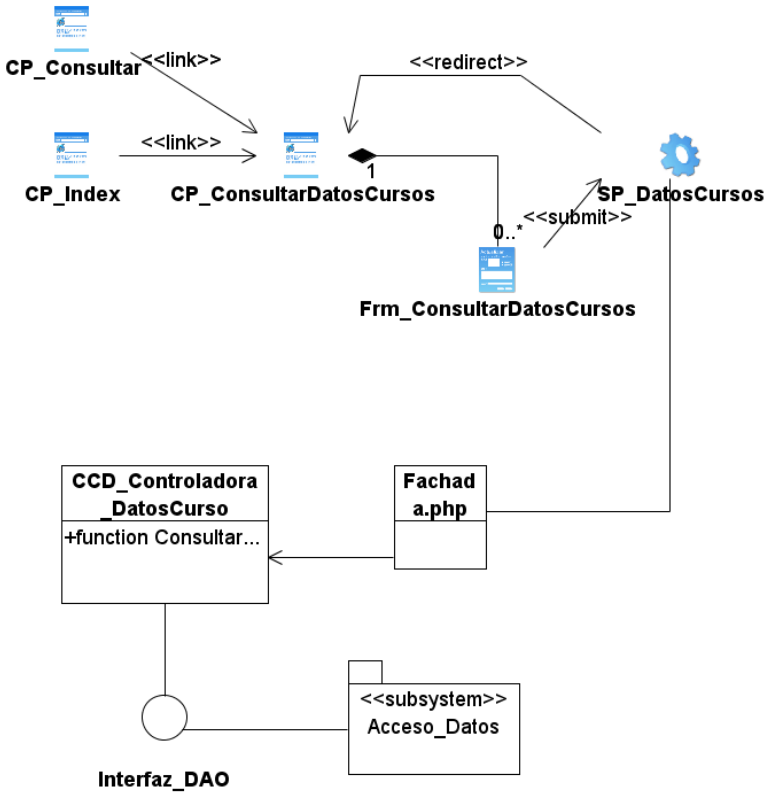


Figura 33: DCD CU "Consultar datos de los cursos".

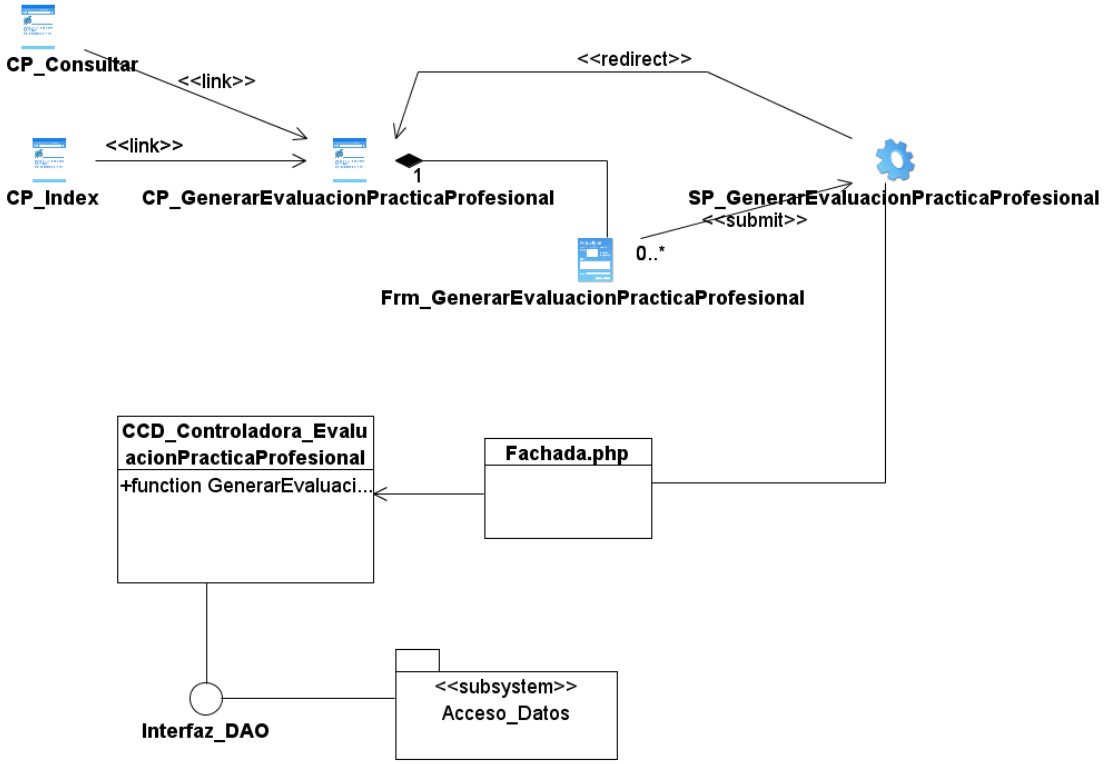


Figura 34: DCD CU "Generar evaluación de Práctica Profesional".

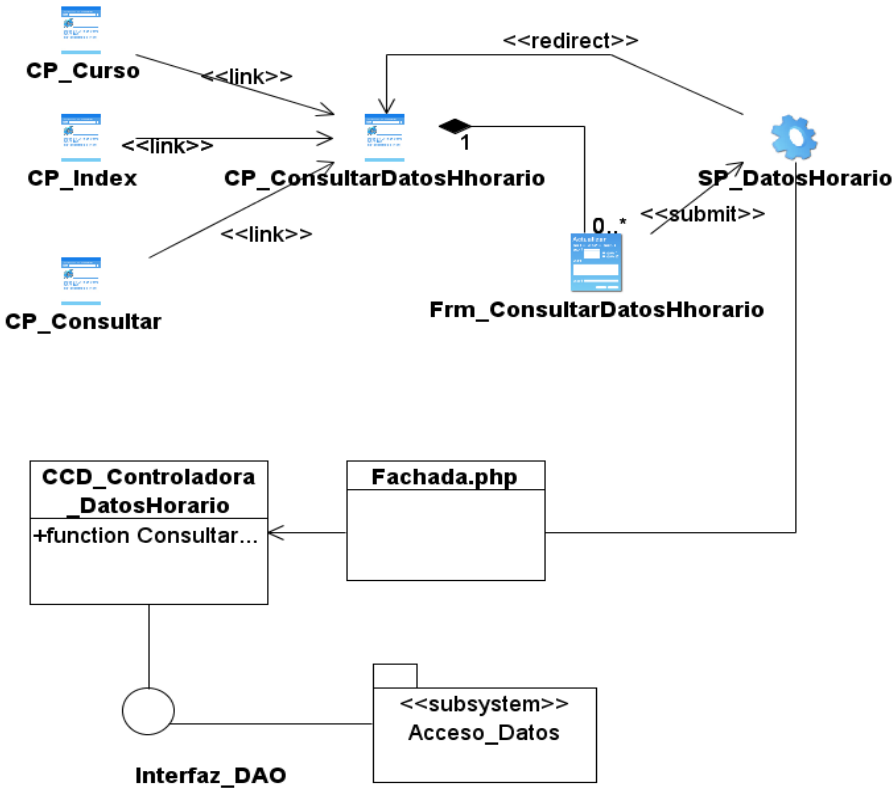


Figura 35: DCD CU "Consultar datos del horario".

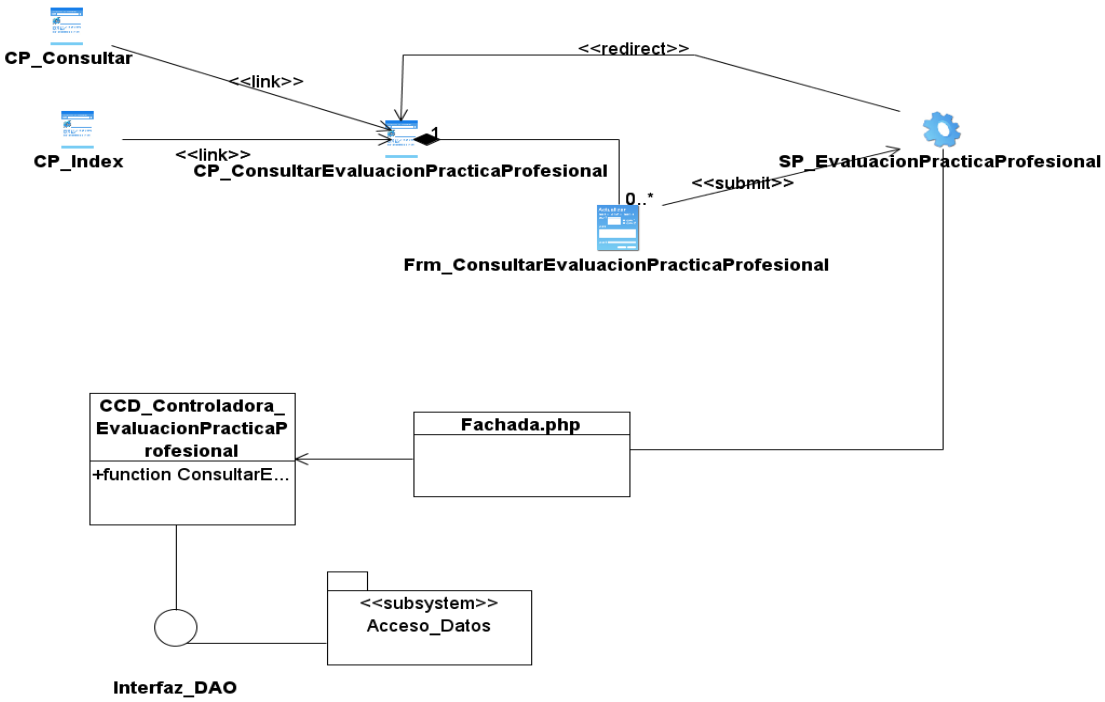


Figura 36: DCD CU "Consultar evaluación de Práctica Profesional".

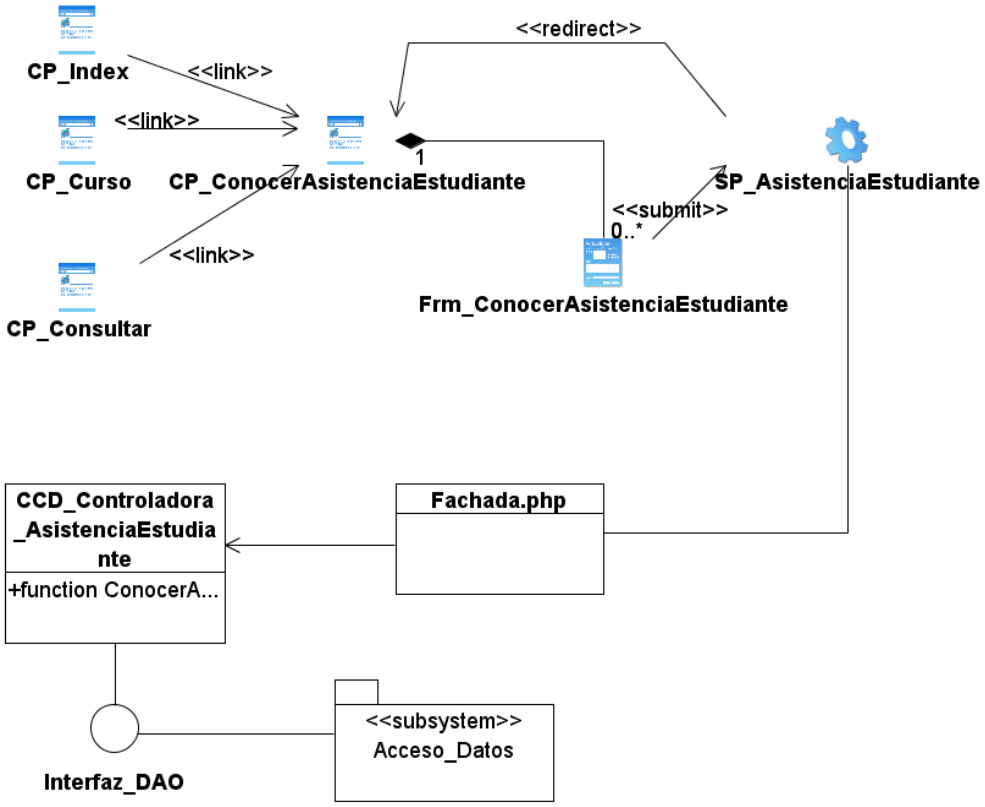


Figura 37: DCD CU "Conocer asistencia de estudiantes".

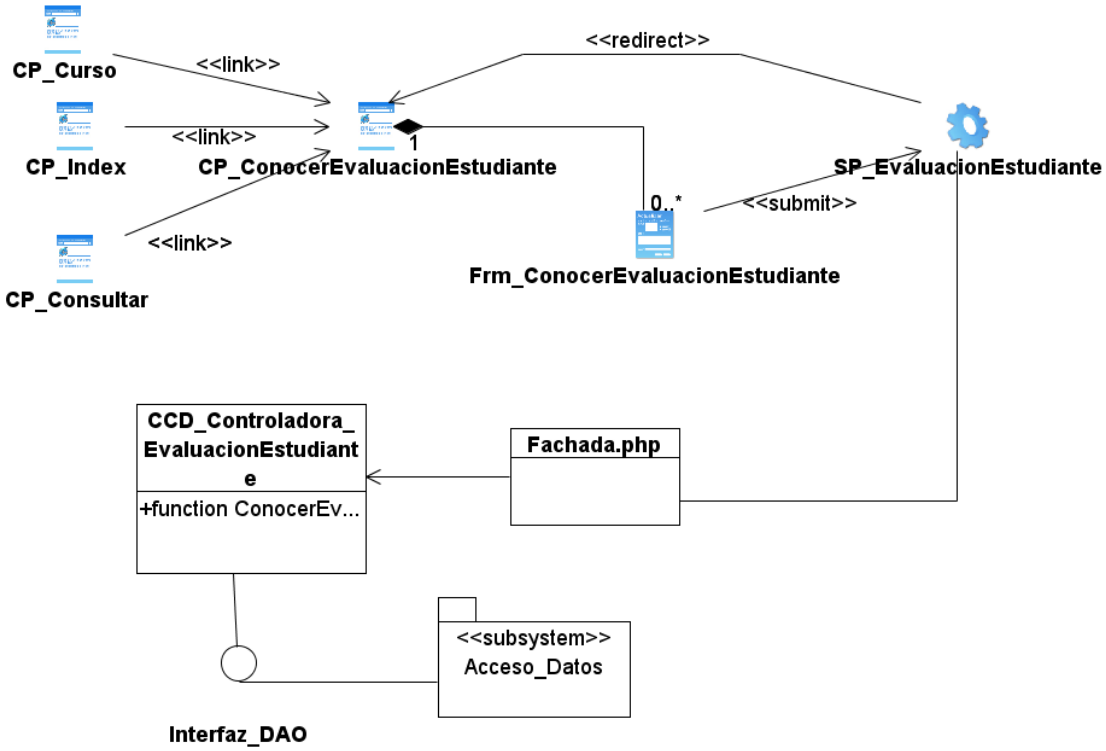


Figura 38: DCD CU "Conocer evaluación del estudiante".

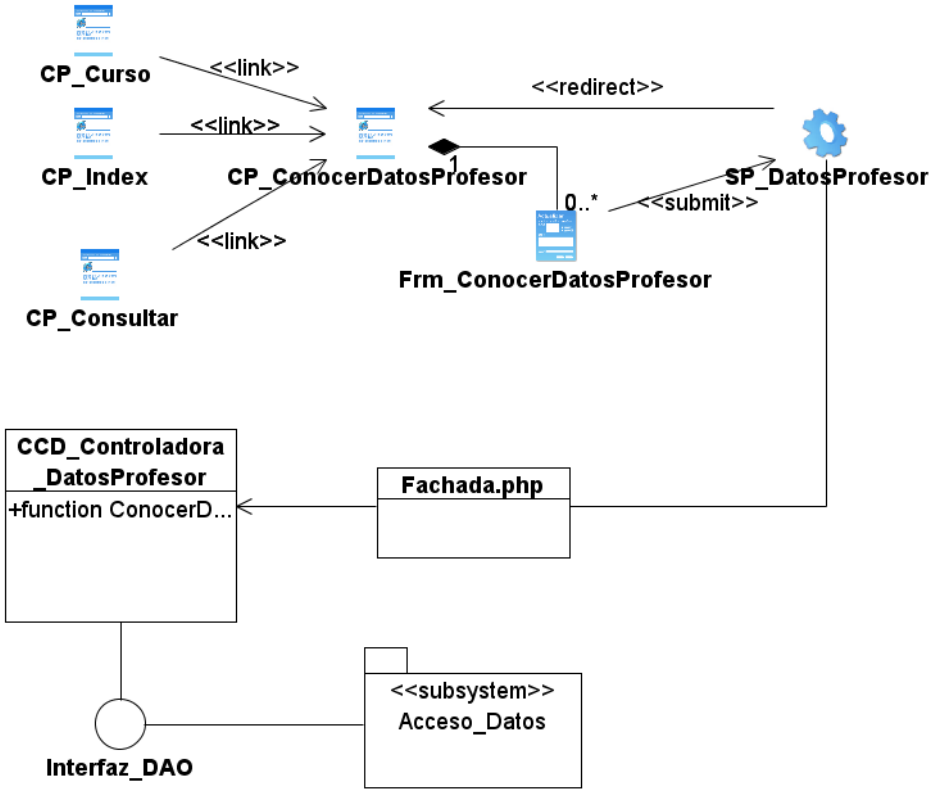


Figura 39: DCD CU "Conocer datos de un profesor".

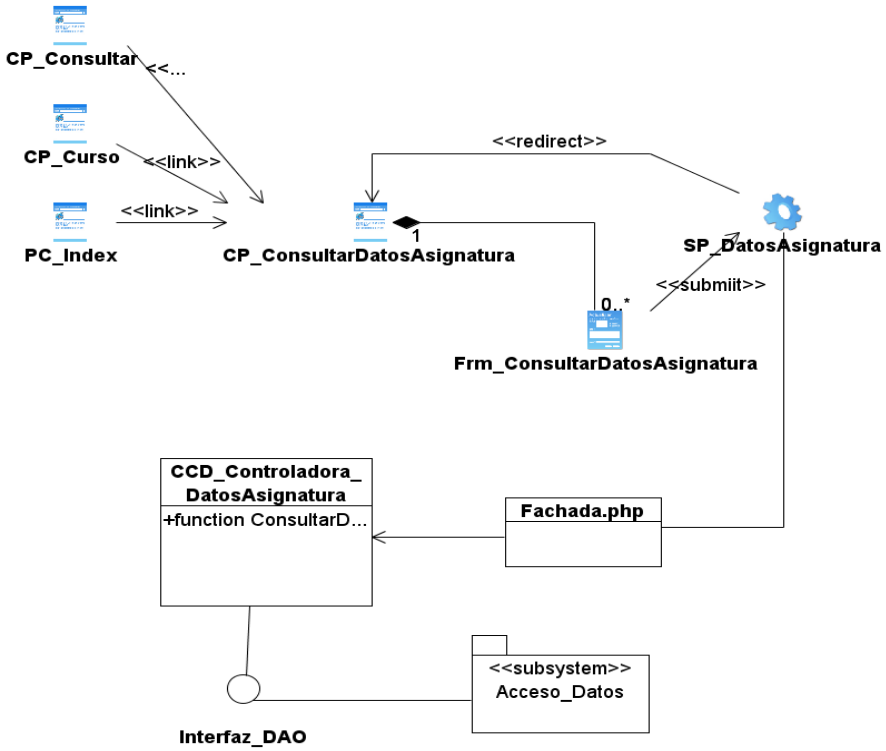


Figura 40: DCD CU "Consultar datos de la asignatura del perfil de calidad".

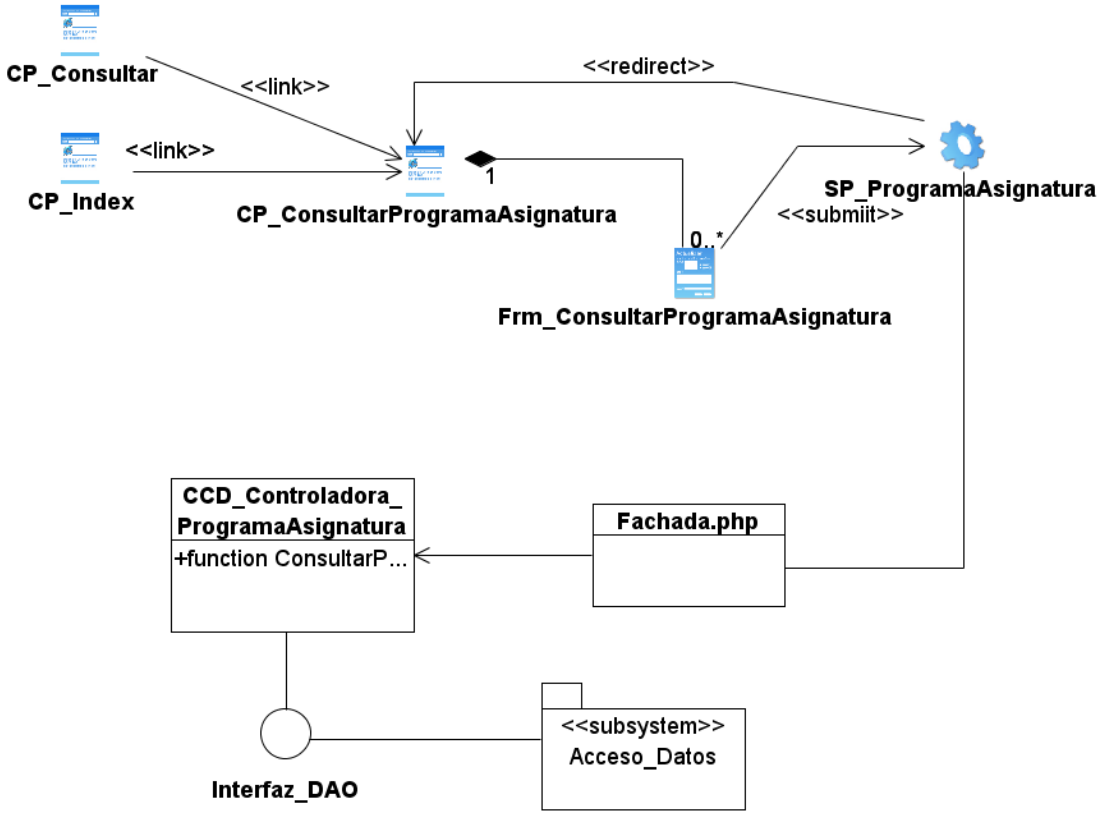


Figura 41: DCD CU “Consultar estructura del programa de la asignatura”.

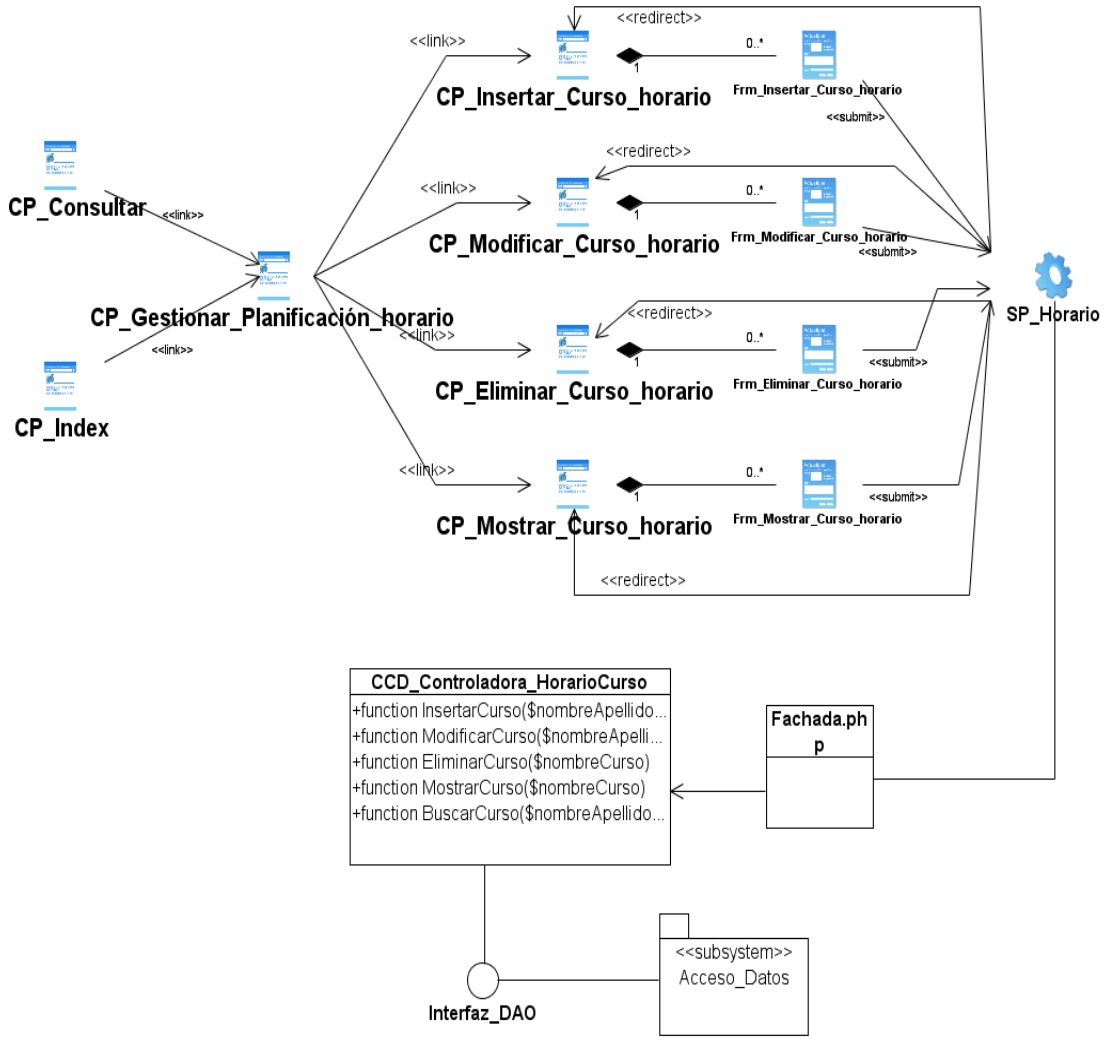


Figura 42: DCD CU “Gestionar planificación de horario de cursos”.

3.4.2 Subsistema de Acceso a Datos.

Ver Anexos [Anexo 1].

3.4.3 Descripción de las clases del diseño.

Para brindar una panorámica más detallada de los atributos y métodos representados, se hace una descripción de cada una de las clases existentes por separado. Ver Anexos [Anexo 2].

3.4.4 Diagrama de Clases Persistentes.

Un diagrama de clases muestra un conjunto de clases, interfaces, y colaboraciones y sus relaciones. Gráficamente un diagrama de clase es una colección de vértices y arcos. Un diagramas de clase persistentes es justo un tipo de diagrama que muestra

un conjunto de objetos deben ser almacenados en algún repositorio como una base de datos relacional. Ver Anexos [Anexo 3].

3.4.5 Diseño de la Base de Datos.

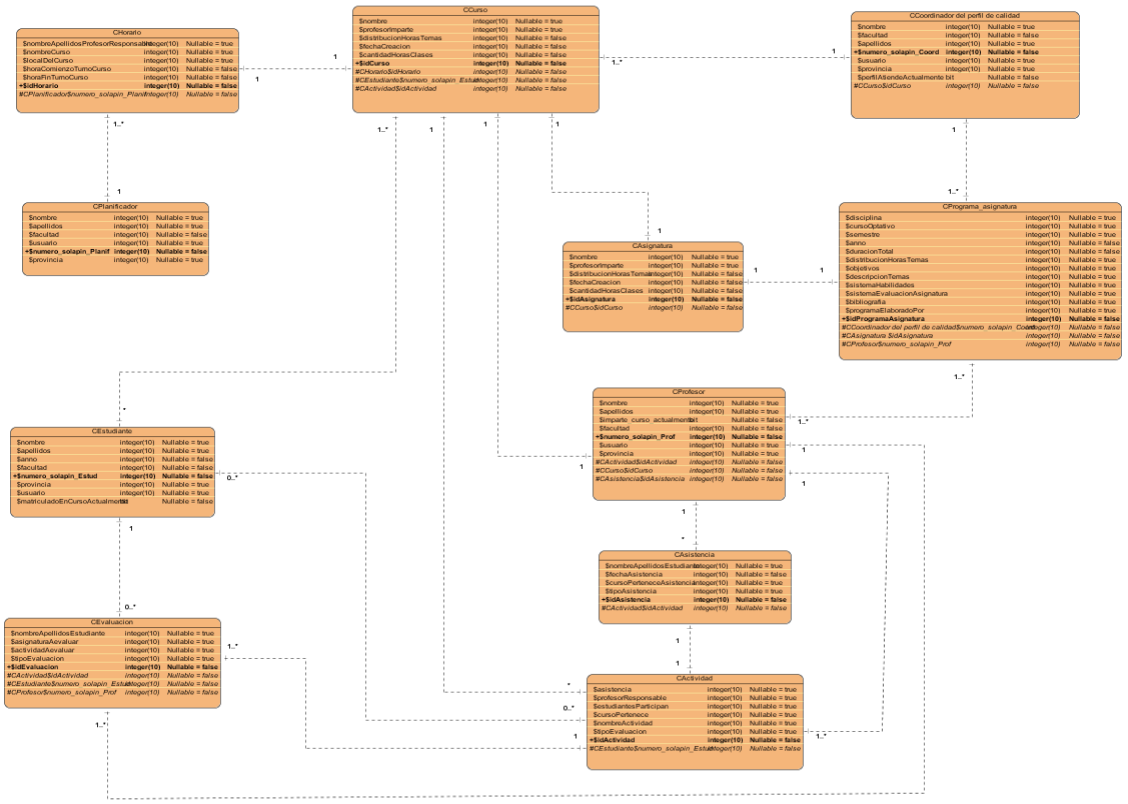


Figura 43: Diagrama Entidad Relación de la Base de Datos.

3.4.6 Descripción de las tablas de la Base de Datos.

Tabla 24: Tablas de la BD.

Nombre: Programa_asignatura		
Descripción: Almacena los datos de los programas de las asignaturas de los cursos del perfil (P1).		
Atributo	**Tipo**	**Descripción**
disciplina	varchar	Disciplina a la que pertenece el programa de la asignatura.
cursoOptativo	varchar	Nombre del curso optativo al que pertenece el programa de la asignatura.
semestre	varchar	Semestre actual.
anno	int	Año de la carrera.
duracionTotal	int	Duración total del de la asignatura.

distribucionHorasTemas	varchar	Distribución de horas por temas del programa de la asignatura.
objetivos	varchar	Objetivos del programa de la asignatura.
descripcionTemas	varchar	Descripción de los temas del programa de la asignatura.
sistemaHabilidades	varchar	Sistema de habilidades que presenta el programa de la asignatura.
sistemaEvaluacionAsignatura	varchar	Sistema de evaluación del programa de la asignatura.
bibliografia	varchar	Bibliografia de del programa de la asignatura
programaElaboradoPor	varchar	Persona que elaboró el programa de la asignatura.
nombreDelPrograma	varchar	Nombre del programa.
idProgramaAsignatura	int	Identificador del Programa Asignatura.
Nombre: Asignatura		
Descripción: Almacena datos relacionados con las asignaturas de los cursos del perfil.		
Atributo	**Tipo**	**Descripción**
nombre	varchar	Nombre de la asignatura.
profesorImparte	varchar	Profesor que imparte la asignatura.
distribucionHorasTemas	int	Distribución de horas por temas de la asignatura.
fechaCreacion	int	Fecha de creación de la asignatura del perfil
cantidadHorasClases	int	Cantidad de horas clases que tiene la asignatura del perfil.
idAsignatura	Int	Identificador de la asignatura.
Nombre: Curso		
Descripción: Almacena todos los datos de los cursos del perfil.		
Atributo	**Tipo**	**Descripción**
nombre	varchar	Nombre del curso.
profesorImparte	varchar	Nombre del profesor que imparte el curso.
distribucionHorasTemas	varchar	Distribución de horas por temas.
fechaCreacion	int	Fecha de creación del curso.
cantidadHorasClases	int	Cantidad de horas clases del curso.
idCurso	Int	Identificador del curso.
Nombre: Profesor		
Descripción: Almacena los datos de los profesores de los cursos del perfil.		
Atributo	**Tipo**	**Descripción**
nombre	varchar	Nombre del profesor.

apellidos	varchar	Apellidos del profesor.
imparte_curso_actualmente	boolean	Saber si el profesor imparte un curso actualmente.
facultad	int	Facultad a la que pertenece el profesor.
numero_solapin_Prof	int	Número de solapín del profesor.
Usuario	varchar	Usuario del profesor.
Provincia	varchar	Provincia del profesor.
Nombre: Asistencia		
Descripción: Almacena los datos de la asistencia de los estudiantes a los cursos del perfil.		
Atributo	**Tipo**	**Descripción**
nombreApellidosEstudiante	varchar	Nombre y apellidos del estudiante del cual se controla la asistencia.
fechaAsistencia	int	Fecha de la asistencia del estudiante.
cursoPerteneceAsistencia	varchar	Curso al cual pertenece la asistencia.
tipoAsistencia	varchar	Tipo de asistencia, (presente, ausente, tardanza).
idAsistencia	Int	Identificador de la asistencia.
Nombre: Evaluacion		
Descripción: Almacena los datos de las evaluaciones de los estudiantes a los cursos del perfil.		
Atributo	**Tipo**	**Descripción**
nombreApellidosEstudiante	varchar	Nombre y apellidos del estudiante del cual se controla sus evaluaciones.
asignaturaAevaluar	varchar	Nombre de la asignatura a evaluar.
actividadAevaluar	varchar	Actividad que se desea evaluar.
tipoEvaluacion	varchar	Tipo de evaluación, (buena, mala, regular).
idEvaluacion	Int	Identificador de la evaluación.
Nombre: Horario		
Descripción: Almacena los datos del horario de los cursos del perfil.		
Atributo	**Tipo**	**Descripción**
nombreApellidosProfesorResponsable	varchar	Nombre y apellidos del profesor responsable de impartir el curso.
nombreCurso	varchar	Nombre del curso que se realizará.
localDelCurso	varchar	Local donde se realizará el curso.
horaComienzoTurnoCurso	integer	Hora de comienzo del turno del curso.
horaFinTurnoCurso	int	Hora de fin del turno del curso.
idHorario	Int	Identificador del horario.

Nombre: Estudiante		
Descripción: Almacena los datos de los estudiantes de los cursos del perfil.		
Atributo	**Tipo**	**Descripción**
nombre	varchar	Nombre del estudiante.
apellidos	varchar	Apellidos del estudiante.
anno	int	Año en que se encuentra el estudiante.
facultad	int	Facultad a la que pertenece el estudiante.
numero_solapin_Estud	int	Número de solapín del estudiante.
provincia	varchar	Provincia del estudiante.
usuario	varchar	Usuario del estudiante.
matriculadoEnCursoActualmente	boolean	Saber si el estudiante está matriculado en un curso actualmente.

Nombre: Planificador		
Descripción: Almacena los datos del planificador del horario de los cursos del perfil.		
Atributo	**Tipo**	**Descripción**
nombre	varchar	Nombre del planificador.
apellidos	varchar	Apellidos del planificador.
facultad	int	Facultad a la cual pertenece el planificador.
numero_solapin_Planif	int	Número de solapín del planificador.
provincia	varchar	Provincia del planificador.
usuario	varchar	Usuario del planificador.

Nombre: Coordinador del perfil de calidad		
Descripción: Almacena los datos del coordinador del perfil.		
Atributo	**Tipo**	**Descripción**
nombre	varchar	Nombre del coordinador del perfil.
apellidos	varchar	Apellidos del coordinador del perfil.
facultad	int	Facultad del coordinador del perfil.
numero_solapin_Coord	int	Número de solapín del coordinador del perfil.
provincia	varchar	Provincia del coordinador del perfil.
usuario	varchar	Usuario del coordinador del perfil.
perfilAtiendeActualmente	boolean	Saber que perfil atiende el coordinador del perfil.

Nombre: Actividad		
Descripción: Almacena los datos de las actividades que se realizan en los cursos del perfil.		
Atributo	**Tipo**	**Descripción**
nombreActividad	varchar	Nombre de la actividad.

asistencia	varchar	Asistencia a la actividad.
profesorResponsable	varchar	Profesor responsable de la actividad.
nombreCurso	varchar	Nombre del curso.
nombreEstudiantesParticipan	varchar	Nombre de los estudiantes que participan.
tipoEvaluacion	varchar	Tipo de evaluación.
idActividad	int	Identificador de la actividad.

3.4.7 Tratamiento de excepciones.

El sistema está dirigido a evitar errores, teniendo en cuenta siempre la creación de interfaces amigables para los usuarios. El tratamiento de errores se realiza al verificar las entradas de datos y al realizar las consultas a la base de datos, verificando que se logre establecer la conexión o que se logre realizar la consulta a una determinada tabla. Se valida que el usuario no deje campos vacíos, que solo tenga acceso al sistema un usuario registrado y que el acceso a un tipo de información sea sólo para aquellos que estén implicados en él y muestra después de cualquier acción realizada un mensaje de confirmación de realización de la acción o muestra el resultado de la misma.

3.4.8 Diagrama de Despliegue.

El proceso de desarrollo de sistemas ha evolucionado dramáticamente en los últimos años. En la última década esa evolución se ha visto impulsada por el paradigma de orientación a objetos, a la utilización de Internet y las redes de datos, como medio global de comunicación. Los sistemas cliente/servidor típicos pertenecen a la categoría de las aplicaciones que operan en dos niveles: la aplicación reside en la estación de trabajo del usuario, comúnmente llamada el cliente, mientras que la base de datos se encuentra alojada en un servidor de datos. En este tipo de aplicaciones, se puede observar que el esfuerzo de computación realmente lo realiza la estación de trabajo cliente, mientras que el servidor solo cumple las funciones de almacenamiento y extracción de los datos; no existe una distribución de la carga de trabajo proporcional a la capacidad de cómputo, pues los clientes, que suelen ser equipos de menor capacidad de cómputo que los servidores, realizan el trabajo más pesado.

Un diagrama de despliegue muestra las relaciones físicas entre los componentes hardware y software en el sistema final, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes software (procesos y objetos que se ejecutan en ellos). Estarán formados por instancias de los

componentes software que representan manifestaciones del código en tiempo de ejecución. (VILAS 2001)

Un diagrama de despliegue es un grafo de nodos unidos por conexiones de comunicación. Un nodo es un objeto físico en tiempo de ejecución que representa un recurso computacional, generalmente con memoria y capacidad de procesamiento. (VILAS 2001)

Los nodos se interconectan mediante soportes bidireccionales que pueden a su vez estereotiparse. Esta vista permite determinar las consecuencias de la distribución y la asignación de recursos. (ALLEGUE and BUGALETTO 2001)

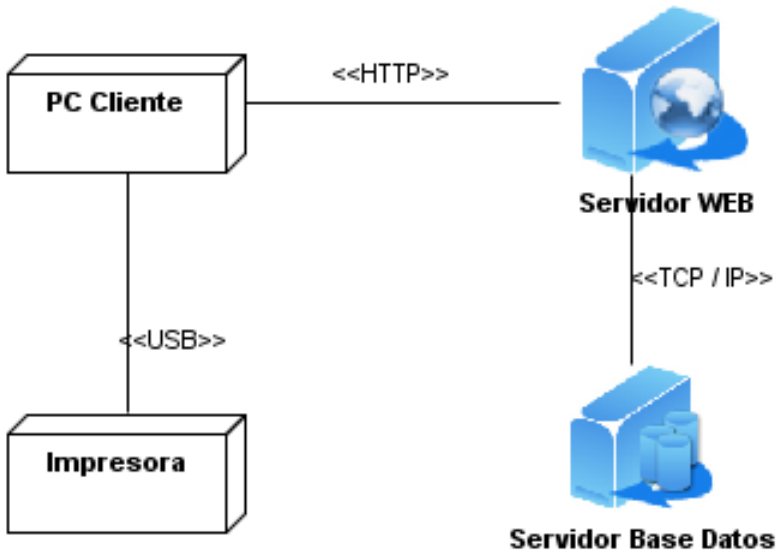


Figura 44: Diagrama de despliegue.

3.5 Conclusiones Parciales.

En este capítulo se realizó el análisis y diseño del sistema, en este último se utilizaron diagramas de clases con estereotipos Web para explicar la lógica del negocio del sistema. Se hizo una descripción de los diagramas de clases del diseño para brindar una panorámica más detallada de los atributos y métodos representados en las mismas.

Se presentó el diagrama de clases persistentes de la base de datos, que contiene la información física que se utilizó para la construcción de la aplicación, así como el modelo de entidad relación que describe la representación lógica y física de datos persistentes en el sistema. Para mejor entendimiento por parte del lector se realizó una descripción de las tablas de la Base de Datos. Se llevó a cabo el diagrama de despliegue el cual muestra las relaciones físicas entre los componentes hardware y software que presentará sistema final.

Conclusiones

Con el desarrollo del presente trabajo se logró hacer un estudio de los procesos del negocio para posteriormente hacer el análisis y diseño de una aplicación web completamente nueva, la cual brinde soporte al proceso de Capacitación del perfil de calidad de la Facultad 8.

Una vez concluida la investigación se logró cumplir los objetivos planteados:

- Se logró estudiar todo el proceso docente-educativo referente al perfil de calidad de la Facultad 8.
- Se propuso el análisis y diseño de un sistema informático para la gestión del perfil de calidad en la Facultad 8.
- Se elaboró toda la documentación referente al software que se propuso, donde se ofrece toda la información de la ingeniería de software del mismo.
- Se contribuyó a una mejor organización de la información relacionada a los cursos del perfil.
- Se contribuyó a la automatización de las actividades relacionadas con la gestión de los cursos del perfil de calidad.

Como síntesis de las principales ideas descritas a lo largo de este trabajo se consideran las siguientes:

Se seleccionó como metodología de desarrollo de software, el Proceso Unificado de Desarrollo de Software (RUP) y como herramienta de modelado el Visual Paradigm.

Se realizó el Modelado de Negocio para el proceso de capacitación de perfil de calidad y para ello se utilizaron técnicas de recopilación de la información, entre las que se encuentran; lluvias de ideas, entrevistas y cuestionarios. Para el negocio, se realizaron diferentes diagramas como lo son; de casos de uso del negocio y de actividades, además una descripción de actores y trabajadores que intervienen en los procesos del negocio, también la especificación de casos de uso del negocio y las reglas del mismo.

Se realizó una propuesta de Modelo de Sistema, realizándose los diagramas de casos de uso del sistema, también la especificación de estos casos de uso, además los modelos conceptuales y una descripción de los actores que intervienen en el sistema.

Para el desarrollo de aplicaciones Web, se propuso la utilización de los lenguajes de programación del lado del cliente, HTML y JavaScript, y del lado del servidor PHP.

Como gestor de bases de datos a utilizar, se propuso PostgreSQL, por sus características.

Se realizó una propuesta de Análisis y Diseño para el sistema, donde se realizaron los diagramas de clases del Análisis y el Diseño, los diagramas de secuencia y una descripción de las clases del Diseño.

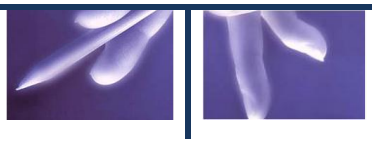
Para el diseño de la base de datos, fue necesario realizar el diagrama de entidad relación. También se hizo una descripción de las tablas de la base de datos y el diagrama de despliegue, para la realización del sistema.

Recomendaciones

Una vez concluido el desarrollo de este trabajo se recomienda:

1. Utilizar la documentación generada en la etapa de implementación del sistema propuesto.
2. Seguir perfeccionando el modelado de sistema mediante la actualización de los cambios que sean necesarios durante el proceso de implementación y pruebas.
3. Usar la documentación obtenida para ampliar las fronteras y realizar productos semejantes no sólo para la Facultad 8, sino también para todas las Universidades del país y la de otros países en los que sea necesario crear y fortalecer una adecuada cultura docente.
4. Continuar profundizando en los contenidos y formas en las que se imparten los cursos del perfil de calidad.
5. Realizar la implementación de la aplicación, para la solución real de los problemas existente en el perfil de calidad de la Facultad 8.

REFERENCIAS BIBLIOGRÁFICAS

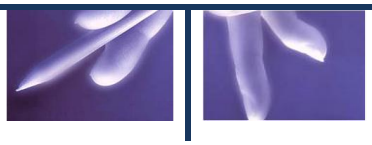


1. "Generalitat Valenciana." from http://www.recursoseees.uji.es/fichas/fc12.pdf.
2. Allegue, F. and G. Bugaletto (2001) UML = Unified Modeling Language. Lenguaje Unificado de Modelamiento. Volume, DOI:
3. Alvarez, J. (1998). "Páginas Active Server (ASP) v 0.1." from http://usuarios.tripod.es/smaug.
4. Alvarez, M. A. (2001). "DesarrolloWeb.com." from http://www.desarrolloweb.com/articulos/436.php.
5. Buchwald, E. (2007). "GSI Grupo Soluciones Innova S. A." from http://www.rational.com.ar/herramientas/roseenterprise.html.
6. Coltell, Ò. (2003). "INTRODUCCIÓN AL PROCESO UNIFICADO DE DESARROLLO DE SOFTWARE." from http://www3.uji.es/~coltell/Docs/IngSw_Apuntes/ISW_2004_TC05.pdf.
7. Daniel, P. R. A. S. J. (2004). "Las tecnologías de la información y la comunicación en la formación docente." from http://unesdoc.unesco.org/images/0012/001295/129533s.pdf.
8. Dondo, A. (1999). "PHP en Castellano." from http://www.programacion.com/php/articulo/porquephp/#porquephp_cuando.
9. Fowler, M. (2003). "La Nueva Metodología ", from http://www.programacionextrema.org/articulos/newMethodology.es.html#tth_sEc7.
10. Gale, S. (2008) Access my library. Volume, DOI:
11. García, S. (2001). "aprendemas.com." from http://microsites.aprendemas.com/Esni/P2.asp.
12. Gorus, F. (2002) JavaHispano. Volume, DOI:
13. GP, A. (2008) Sistema de Gestión de un centro de rehabilitación neurológica. Volume, DOI:
14. Hawes, G. and O. Corvalán (2004). Construcción de un perfil profesional. Talca.
15. Hernández, J. (1997) Técnicas de recopilación de la información. Volume, DOI:
16. Jalón, J. G. d., J. I. Rodríguez, et al. (1998). "Aprenda C++ como si estuviera en primero." from: http://mat21.etsii.upm.es/ayudainf/aprendainf/Cpp/manualcpp.pdf.
17. Lago, R. (2007) Patron "Data Access Object". Volume, DOI:
18. Lago, R. (2007). "Patrón "Modelo-Vista-Controlador"." from http://www.proactiva-calidad.com/java/patrones/mvc.html.

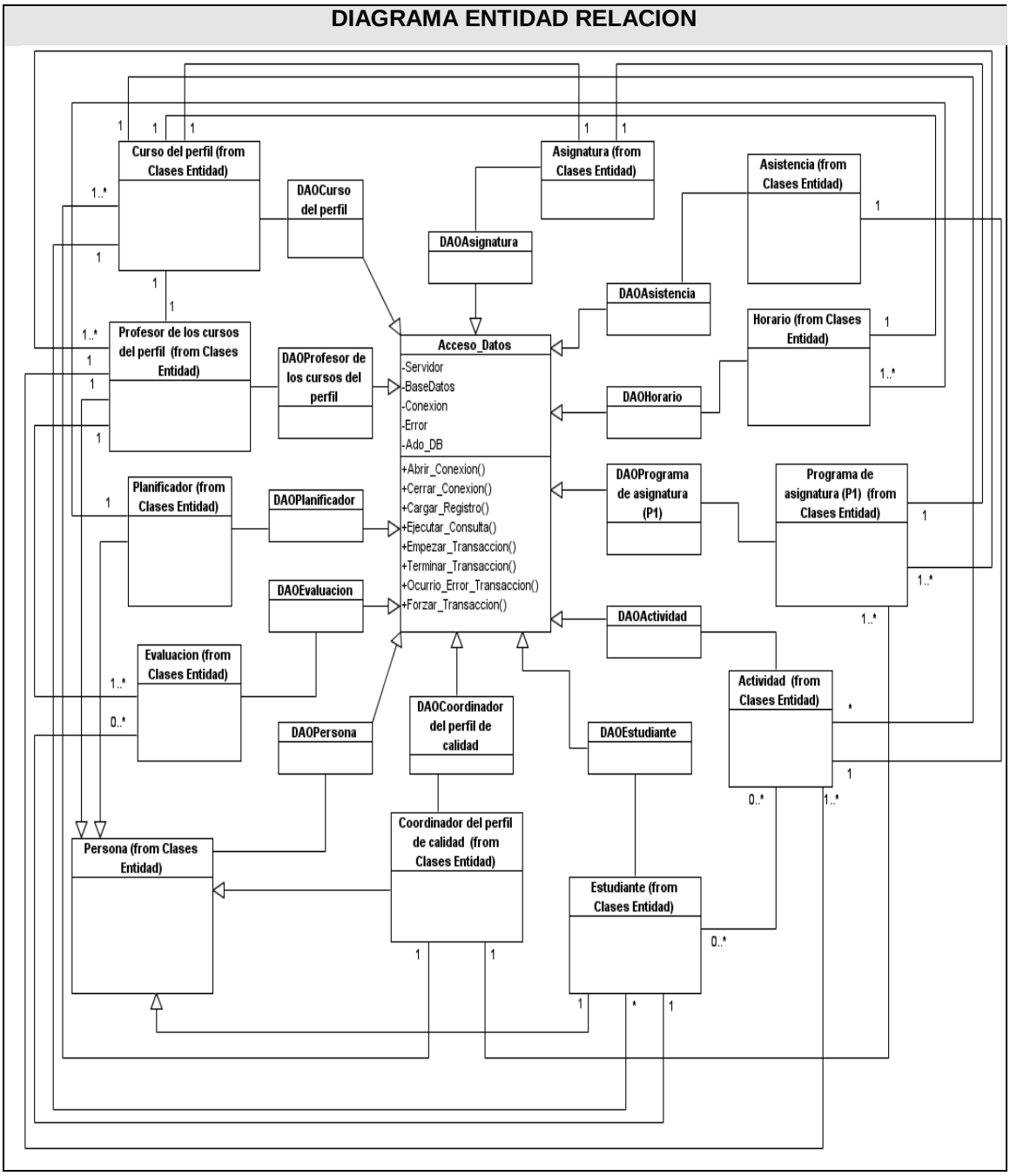
19. Larman, C. (2006). "El mundo informatico y tu que mundo vives." from http://jorgesaavedra.wordpress.com/2006/08/17/patrones-grasp-craig-larman/.
20. Letelier, P. and M. C. Penadés. (2007). "Métodologías ágiles para el desarrollo de software:
21. eXtreme Programming (XP) ", from http://www.willydev.net/descargas/masyxp.pdf.
22. Lockhart, T. (1996). "Manual del usuario de PostgreSQL." from https://forja.rediris.es/docman/view.php/312/454/Postgres-User.pdf.
23. M., R. V. (1997). "Monogrfías.com." from http://www.monografias.com/trabajos13/metomt/metomt.shtml.
24. Magallan, D. F. (2004) CALIDAD DE LOS SISTEMAS INFORMATICOS Volume, DOI:
25. Martín, P. R. (2005). "HTML." from http://www.asptutor.com/zip/cbhtml.pdf.
26. Mclennan, M. (2007) Kynetia Software for business solutions. Volume, DOI:
27. Moore, J., S. Fox, et al. (2001). "Documentación PHP-GTK."
28. Mora, R. C. (2003). "Patrones Grasp." from http://sophia.javeriana.edu.co/~lcdiaz/ADOO2006-3/grasp_cpaternostro-lvargas-jviafara.pdf.
29. Morera, G., B. Berciano, et al. (2001). "Virtualpyme." from http://www.virtualpyme.com/desarrollo.htm.
30. Pantoja, D. H. B. D. G. V. J. S. V. G. A. E. Q. (2005). "Patrones Grasp." from http://www.google.com.cu/search?hl=es&q=patrones+de+dise%C3%B1o+de+software+%28Bustamante%2C+Valencia%2C+Valencia%2C+Asakawa%2C+%26+Pantoja%29&meta=.
31. Pecos, D. "PostGreSQL vs. MySQL." from http://www.netpecos.org/docs/mysql_postgres/index.html.
32. Pecos, D. (2005) PostGreSQL vs. MySQL. Volume, DOI:
33. Pérez, Y. F., A. F. Estrada, et al. (2008). "Diseño de un segundo perfil de calidad de software."
34. Plomé, A. (2000) Entrevistas y cuestionarios: técnicas para la elaboración de preguntas y recolección de respuestas en investigación. Volume, DOI:
35. Sánchez, J. C. (2006) Lycka Bonita. Volume, DOI:
36. Sanchez, M. M. (2004). "Metodologías De Desarrollo De Software." from http://www.informatizate.net.
37. Solís, M. C. (2003). "Una explicación de la programación extrema (XP)."

38. UCI, D. d. i. (2005). "AKADEMOS (Gestión Académica)." from http://akademos/akademos/default.aspx.
39. Valle, J. (1997). "Herramientas CASE para BD."
40. Vilas, A. F. (2001) Diagrama de despliegue. Volume, DOI:
41. Wong, R. (2004). "Calidad construir aplicaciones más rápido, mejor y más barato", from http://66.102.1.101/translate_c?hl=es&sl=en&u=http://www.visual-paradigm.com/product/vpuml/enterpriseedition.jsp&prev=/search%3Fq%3DVisual%2Bparadigma-UML.%26hl%3Des%26sa%3DG&usg=ALkJrhiHWtg4azQfAMdx_bTPI8PbpeTBbw.

ANEXOS



Anexo 1. Diagramas de clases del diseño del Subsistema de Acceso a Datos.



Anexo 2. Descripción de las clases del diseño.

➢ Clases Controladoras.

Nombre: Gestionar profesor	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	InsertarProfesor($nombre, $apellidos, $imparte_curso_actualmente, $facultad, $numero_solapin)
	ModificarProfesor($nombre, $apellidos, $imparte_curso_actualmente, $facultad, $numero_solapin)
	Eliminar Profesor($nombre, $apellidos, $imparte_curso_actualmente, $facultad, $numero_solapin)
	Mostrar Profesor($nombre, $apellidos, $imparte_curso_actualmente, $facultad, $numero_solapin)
	Buscar Profesor($nombre, $apellidos, $imparte_curso_actualmente, $facultad, $numero_solapin)
	Actualizar Profesor($nombre, $apellidos, $imparte_curso_actualmente, $facultad, $numero_solapin)
Descripción:	Controla todo lo referente a los profesores de cursos del perfil de calidad.

Nombre: Gestionar estudiante	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	InsertarEstudiante($nombre, $apellidos, $anno, $facultad, $numero_solapin)
	ModificarEstudiante($nombre, $apellidos, $anno, $facultad, $numero_solapin)
	EliminarEstudiante($nombre, $apellidos, $anno, $facultad, $numero_solapin)
	MostrarEstudiante($nombre, $apellidos, $anno, $facultad, $numero_solapin)
	BuscarEstudiante($nombre, $apellidos, $anno, $facultad, $numero_solapin)

	ActualizarEstudiante($nombre, $apellidos, $anno, $facultad, $numero_solapin)
Descripción:	Controla todo lo referente a los estudiantes que reciben los cursos del perfil de calidad.

Nombre: Gestionar evaluaciones de estudiantes	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	InsertarEvaluaciones_estudiante($nombreApellidosEstudiante, $asignaturaAevaluar, $actividadAevaluar, $tipoEvaluacion)
	ModificarEvaluaciones_estudiante($nombreApellidosEstudiante, $asignaturaAevaluar, $actividadAevaluar, $tipoEvaluacion)
	EliminarEvaluaciones_estudiante($nombreApellidosEstudiante, $asignaturaAevaluar, $actividadAevaluar, $tipoEvaluacion)
	MostrarEvaluaciones_estudiante($nombreApellidosEstudiante, $asignaturaAevaluar, $actividadAevaluar, $tipoEvaluacion)
	BuscarEvaluaciones_estudiante($nombreApellidosEstudiante, $asignaturaAevaluar, $actividadAevaluar, $tipoEvaluacion)
	ActualizarEvaluaciones_estudiante($nombreApellidosEstudiante, $asignaturaAevaluar, $actividadAevaluar, $tipoEvaluacion)
Descripción:	Controla todo lo referente a las evaluaciones obtenidas por los estudiantes en el transcurso de los cursos del perfil de calidad.

Nombre: Gestionar asistencia de estudiante	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	InsertarAsistencia_estudiante($nombreApellidosEstudiante, $fechaAsistencia, $tipoAsistencia, $cursoPerteneceAsistencia)
	ModificarAsistencia_estudiante($nombreApellidosEstudiante, $fechaAsistencia, $tipoAsistencia, $cursoPerteneceAsistencia)
	EliminarAsistencia_estudiante($nombreApellidosEstudiante, $fechaAsistencia, $tipoAsistencia, $cursoPerteneceAsistencia)
	MostrarAsistenciaEstudiante($nombreApellidosEstudiante, $fechaAsistencia, $tipoAsistencia, $cursoPerteneceAsistencia)
	BuscarAsistencia_estudiante($nombreApellidosEstudiante,

	$fechaAsistencia, $tipoAsistencia, $cursoPerteneceAsistencia)
	ActualizarAsistencia_estudiante($nombreApellidosEstudiante, $fechaAsistencia, $tipoAsistencia, $cursoPerteneceAsistencia)
Descripción:	Controla todo lo referente a la asistencia de los estudiantes en los cursos del perfil de calidad.

Nombre: Gestionar asignatura del perfil	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	InsertarAsignatura_perfil($nombre, $profesorImparte, $distribucionHorasTemas, $fechaCreacion, $cantidadHorasClases)
	ModificarAsignatura_perfil($nombre, $profesorImparte, $distribucionHorasTemas, $fechaCreacion, $cantidadHorasClases)
	EliminarAsignatura_perfil($nombre, $profesorImparte, $distribucionHorasTemas, $fechaCreacion, $cantidadHorasClases)
	MostrarAsignatura_perfil($nombre, $profesorImparte, $distribucionHorasTemas, $fechaCreacion, $cantidadHorasClases)
	BuscarAsignatura_perfil($nombre, $profesorImparte, $distribucionHorasTemas, $fechaCreacion, $cantidadHorasClases)
	ActualizarAsignatura_perfil($nombre, $profesorImparte, $distribucionHorasTemas, $fechaCreacion, $cantidadHorasClases)
Descripción:	Controla todo lo referente a las asignaturas que se imparten a los estudiantes en los cursos del perfil de calidad.

Nombre: Gestionar programa de asignatura del perfil	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	InsertarPrograma_Asignatura($disciplina, $cursoOptativo, $semestre, $anno, $duracionTotal, $distribucionHorasTemas, $objetivos, $descripcionTemas, $sistemaHabilidades, $sistemaEvaluacionAsignatura, $bibliografia, $programaElaboradoPor)
	ModificarPrograma_Asignatura($disciplina, $cursoOptativo, $semestre, $anno, $duracionTotal, $distribucionHorasTemas, $objetivos, $descripcionTemas, $sistemaHabilidades, $sistemaEvaluacionAsignatura, $bibliografia, $programaElaboradoPor)

	EliminarPrograma_Asignatura($disciplina, $cursoOptativo, $semestre, $anno, $duracionTotal, $distribucionHorasTemas, $objetivos, $descripcionTemas, $sistemaHabilidades, $sistemaEvaluacionAsignatura, $bibliografia, $programaElaboradoPor)
	MostrarPrograma_Asignatura($disciplina, $cursoOptativo, $semestre, $anno, $duracionTotal, $distribucionHorasTemas, $objetivos, $descripcionTemas, $sistemaHabilidades, $sistemaEvaluacionAsignatura, $bibliografia, $programaElaboradoPor)
	ExportarPDF_Programa_Asignatura($disciplina, $cursoOptativo, $semestre, $anno, $duracionTotal, $distribucionHorasTemas, $objetivos, $descripcionTemas, $sistemaHabilidades, $sistemaEvaluacionAsignatura, $bibliografia, $programaElaboradoPor)
	Actualizar_Programa_Asignatura($disciplina, $cursoOptativo, $semestre, $anno, $duracionTotal, $distribucionHorasTemas, $objetivos, $descripcionTemas, $sistemaHabilidades, $sistemaEvaluacionAsignatura, $bibliografia, $programaElaboradoPor)
	Buscar_Programa_Asignatura($disciplina, $cursoOptativo, $semestre, $anno, $duracionTotal, $distribucionHorasTemas, $objetivos, $descripcionTemas, $sistemaHabilidades, $sistemaEvaluacionAsignatura, $bibliografia, $programaElaboradoPor)
Descripción:	Controla todo lo referente al programa de las asignaturas que se imparten a los estudiantes en los cursos del perfil de calidad.

Nombre: Gestionar planificación del horario	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	InsertarCurso($nombreApellidosProfesorResponsable, $nombreCurso)
	ModificarCurso($nombreApellidosProfesorResponsable, $nombreCurso)
	EliminarCurso($nombreApellidosProfesorResponsable, $nombreCurso)
	MostrarCurso($nombreApellidosProfesorResponsable, $nombreCurso)
	BuscarCurso($nombreApellidosProfesorResponsable, $nombreCurso)
	ActualizarCurso($nombreApellidosProfesorResponsable, $nombreCurso)
Descripción:	Controla todo lo referente a la planificación del horario docente de los cursos del perfil de calidad.

Nombre: Gestionar curso

Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Insertar_EstudianteCurso($nombre, $apellidos, $anno, $facultad, $numeroSolapin)
	Modificar_EstudianteCurso($nombre, $apellidos, $anno, $facultad, $numeroSolapin)
	Eliminar_EstudianteCurso($nombre, $apellidos, $anno, $facultad, $numeroSolapin)
	Actualizar_EstudianteCurso($nombre, $apellidos, $anno, $facultad, $numeroSolapin)
	Mostrar_EstudianteCurso($nombre, $apellidos, $anno, $facultad, $numeroSolapin)
	Buscar_EstudianteCurso($nombre, $apellidos, $anno, $facultad, $numeroSolapin)
	Insertar_ProfesorCurso($nombre, $apellidos, $imparte_curso_actualmente, $facultad, $numeroSolapin)
	Mostrar_ProfesorCurso($nombre, $apellidos, $imparte_curso_actualmente, $facultad, $numeroSolapin)
	Modificar_ProfesorCurso($nombre, $apellidos, $imparte_curso_actualmente, $facultad, $numeroSolapin)
	Eliminar_ProfesorCurso($nombre, $apellidos, $imparte_curso_actualmente, $facultad, $numeroSolapin)
	Actualizar_ProfesorCurso($nombre, $apellidos, $imparte_curso_actualmente, $facultad, $numeroSolapin)
	Buscar_ProfesorCurso($nombre, $apellidos, $imparte_curso_actualmente, $facultad, $numeroSolapin)
	Insertar_AsignaturaCurso($nombre, $profesorImparte, $distribucionHorasTemas, $fechaCreacion, $cantidadHorasClases)
	Modificar_AsignaturaCurso($nombre, $profesorImparte, $distribucionHorasTemas, $fechaCreacion, $cantidadHorasClases)
	Mostrar_AsignaturaCurso($nombre, $profesorImparte, $distribucionHorasTemas, $fechaCreacion, $cantidadHorasClases)
	Eliminar_AsignaturaCurso($nombre, $profesorImparte, $distribucionHorasTemas, $fechaCreacion, $cantidadHorasClases)
	Buscar_AsignaturaCurso($nombre, $profesorImparte, $distribucionHorasTemas, $fechaCreacion, $cantidadHorasClases)
	Actualizar_AsignaturaCurso($nombre, $profesorImparte,

	$distribucionHorasTemas, $fechaCreacion, $cantidadHorasClases)
	Cancelar_matricula_curso($nombreApellidosEstudiante)
	Cambiar_ProfesorCurso($nombreApellidosEstudiante)
	Mostrar_MatriculaCurso($nombreApellidosEstudiante, $nombreApellidosProfesorResponsableCurso)
	Imprimir_RegistroControlAsistenciaEvaluaciones($nombre)
Descripción:	Controla todo lo referente a los curso del perfil de calidad.

- **Clases Entidad.**

Nombre: Programa_asignatura	
Tipo de clase: Entidad	
Atributo	**Tipo**
$disciplina	
$cursoOptativo	
$semestre	
$anno	
$duracionTotal	
$distribucionHorasTemas	
$objetivos	
$descripcionTemas	
$sistemaHabilidades	
$sistemaEvaluacionAsignatura	
$bibliografia	
$programaElaboradoPor	
$nombreDelPrograma	
Nombre:	Programa_asignatura($Edisciplina, $EcursoOptativo, $Esemestre, $Eanno, $EduracionTotal, $EdistribucionHorasTemas, $Eobjetivos, $EdescripcionTemas, $EsistemaHabilidades, $EsistemaEvaluacionAsignatura, $Ebibliografia, $EprogramaElaboradoPor, $EnombreDelPrograma)
	Get_ Disciplina()
	Get_CursoOptativo()
	Get_Semestre()
	Get_Anno()
	Get_DuracionTotal()
	Get_DistribucionHorasTemas()
	Get_Objetivos()

	Get_DescripcionTemas()
	Get_SistemaHabilidades()
	Get_SistemaEvaluacionAsignatura()
	Get_Bibliografía()
	Get_ProgramaElaboradoPor()
	Get_NombreDelPrograma()
	Set_ Disciplina()
	Set_CursoOptativo()
	Set_Semestre()
	Set_Anno()
	Set_DuracionTotal()
	Set_DistribucionHorasTemas()
	Set_Objetivos()
	Set_DescripcionTemas()
	Set_SistemaHabilidades()
	Set_SistemaEvaluacionAsignatura()
	Set_Bibliografía()
	Set_ProgramaElaboradoPor()
	Set_NombreDelPrograma()
Descripción:	Controla toda la información relacionada a la estructura del programa de una asignatura de cursos del perfil de calidad.

Nombre: Asignatura	
Tipo de clase: Entidad	
Atributo	**Tipo**
$nombre	
$profesorImparte	
$distribucionHorasTemas	
$fechaCreacion	
$cantidadHorasClases	
Nombre:	Asignatura($Enombre, $EprofesorImparte, $EdistribucionHorasTemas, $EfechaCreacion, $EcantidadHorasClases)
	Get_Nombre()
	Get_ProfesorImparte()
	Get_DistribucionHorasTemas()
	Get_FechaCreacion()

	Get_CantidadHorasClases()
	Set_Nombre()
	Set_ProfesorImparte()
	Set_DistribucionHorasTemas()
	Set_FechaCreacion()
	Set_CantidadHorasClases()
Descripción:	Controla la información relacionada a las asignaturas de cursos del perfil de calidad.

Nombre: Curso	
Tipo de clase: Entidad	
Atributo	**Tipo**
$nombre	
$profesorImparte	
$distribucionHorasTemas	
$fechaCreacion	
$cantidadHorasClases	
Nombre:	Curso($Enombre, $EprofesorImparte, $EdistribucionHorasTemas, $EfechaCreacion, $EcantidadHorasClases)
	Get_Nombre()
	Get_ProfesorImparte()
	Get_DistribucionHorasTemas()
	Get_FechaCreacion()
	Get_CantidadHorasClases()
	Set_Nombre()
	Set_ProfesorImparte()
	Set_DistribucionHorasTemas()
	Set_FechaCreacion()
	Set_CantidadHorasClases()
Descripción:	Controla la información relacionada a cursos del perfil de calidad.

Nombre: Profesor	
Tipo de clase: Entidad	
Atributo	**Tipo**
$nombre	
$apellidos	
$imparte_curso_actualmente	

$facultad		
$numero_solapin		
$usuario		
$provincia		
Nombre:	Profesor($Enombre, $Eapellidos, $Eimparte_curso_actualmente, $Efacultad, $Enumero_solapin, $Eusuario, $Eprovincia)	
	Get_Nombre()	
	Get_Apellidos()	
	Get_Imparte_curso_actualmente()	
	Get_Facultad()	
	Get_Numero_solapin()	
	Get_Usuario()	
	Get_Provincia()	
	Set_Nombre()	
	Set_Apellidos()	
	Set_Imparte_curso_actualmente()	
	Set_Facultad()	
	Set_Numero_solapin()	
	Set_Usuario()	
	Set_Provincia()	
Descripción:	Controla la información relacionada a profesores de cursos del perfil de calidad.	

Nombre: Asistencia		
Tipo de clase: Entidad		
Atributo		**Tipo**
$nombreApellidosEstudiante		
$fechaAsistencia		
$cursoPerteneceAsistencia		
$tipoAsistencia		
Nombre:	Asistencia($EnombreApellidosEstudiante, $EfechaAsistencia, $EcursoPerteneceAsistencia, $EtipoAsistencia)	
	Get_NombreApellidosEstudiante()	
	Get_FechaAsistencia()	
	Get_CursoPerteneceAsistencia()	
	Get_TipoAsistencia()	
	Set_NombreApellidosEstudiante()	

	Set_FechaAsistencia()
	Set_CursoPerteneceAsistencia()
	Set_TipoAsistencia()
Descripción:	Controla la información relacionada a la asistencia de los estudiantes de cursos del perfil de calidad.

Nombre: Evaluacion	
Tipo de clase: Entidad	
Atributo	**Tipo**
$nombreApellidosEstudiante	
$asignaturaAevaluar	
$actividadAevaluar	
$tipoEvaluacion	
Nombre:	Evaluacion($EnombreApellidosEstudiante, $EasignaturaAevaluar, $EactividadAevaluar, $EtipoEvaluacion)
	Get_NombreApellidosEstudiante()
	Get_AsignaturaAEvaluar()
	Get_ActividadAEvaluar()
	Get_TipoEvaluacion()
	Set_NombreApellidosEstudiante()
	Set_AsignaturaAEvaluar()
	Set_ActividadAEvaluar()
	Set_TipoEvaluacion()
Descripción:	Controla la información relacionada a las evaluaciones de los estudiantes de cursos del perfil de calidad.

Nombre: Horario	
Tipo de clase: Entidad	
Atributo	**Tipo**
$nombreApellidosProfesorResponsable	
$nombreCurso	
$localDelCurso	
$horaComienzoTurnoCurso	
$horaFinTurnoCurso	
Nombre:	Horario($EnombreApellidosProfesorResponsable, $EnombreCurso, $ElocalDelCurso, $EhoraComienzoTurnoCurso, $EhoraFinTurnoCurso)
	Get_NombreApellidosProfesorResponsable()

	Get_NombreCurso()
	Get_LocalDelCurso()
	Get_HoraComienzoTurnoCurso()
	Get_HoraFinTurnoCurso()
	Set_NombreApellidosProfesorResponsable ()
	Set_NombreCurso()
	Set_LocalDelCurso()
	Set_HoraComienzoTurnoCurso()
	Set_HoraFinTurnoCurso()
Descripción:	Controla la información relacionada al horario docente de los cursos del perfil de calidad.

Nombre: Estudiante	
Tipo de clase: Entidad	
Atributo	**Tipo**
$nombre	
$apellidos	
$anno	
$facultad	
$numero_solapin	
$provincia	
$usuario	
$matriculadoEnCursoActualmente	
Nombre:	Profesor($Enombre, $Eapellidos, $Eanno, $Efacultad, $Enumero_solapin, $Eprocincia, $Eusuario, $EmatriculadoEnCursoActualmente)
	Get_Nombre()
	Get_Apellidos()
	Get_Anno()
	Get_Facultad()
	Get_Numero_solapin()
	Get_Provincia()
	Get_Usuario()
	Get_MatriculadoEnCursoActualmente()
	Set_Nombre()
	Set_Apellidos()
	Set_Anno()

	Set_Facultad()
	Set_Numero_solapin()
	Set_Provincia()
	Set_Usuario()
	Set_MatriculadoEnCursoActualmente()
Descripción:	Controla la información relacionada a estudiantes de cursos del perfil de calidad.

Nombre: Planificador	
Tipo de clase: Entidad	
Atributo	**Tipo**
$nombre	
$apellidos	
$facultad	
$numero_solapin	
$provincia	
$usuario	
Nombre:	Profesor($Enombre, $Eapellidos, $Efacultad, $Enumero_solapin, $Eprocincia, $Eusuario)
	Get_Nombre()
	Get_Apellidos()
	Get_Facultad()
	Get_Numero_solapin()
	Get_Provincia()
	Get_Usuario()
	Set_Nombre()
	Set_Apellidos()
	Set_Facultad()
	Set_Numero_solapin()
	Set_Provincia()
	Set_Usuario()
Descripción:	Controla la información relacionada a planificadores del horario de los cursos del perfil de calidad.

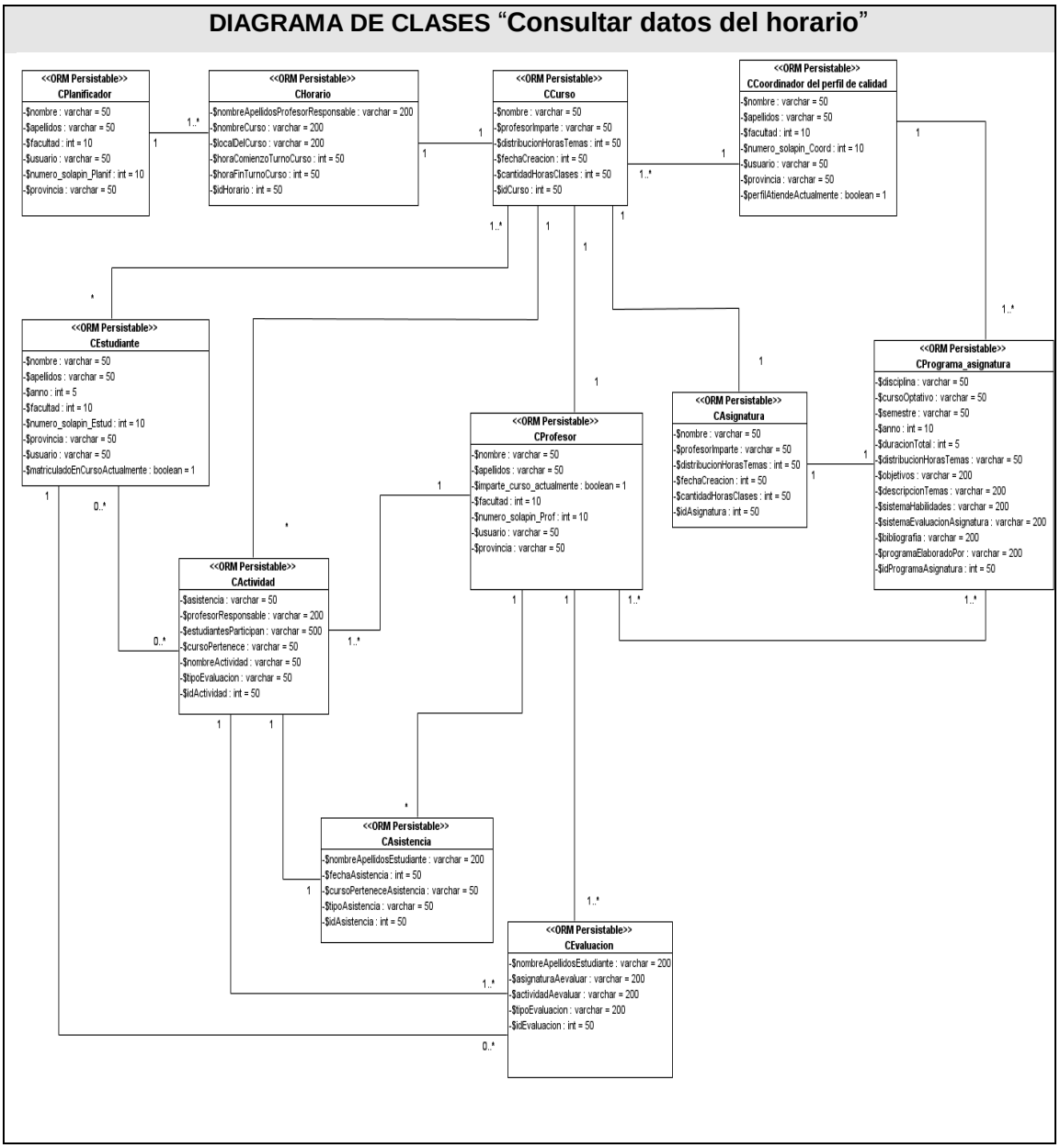
Nombre: Coordinador del perfil de calidad	
Tipo de clase: Entidad	
Atributo	**Tipo**

$nombre		
$apellidos		
$facultad		
$numero_solapin		
$provincia		
$usuario		
$perfilAtiendeActualmente		
Nombre:	Profesor($Enombre, $Eapellidos, $Efacultad, $Enumero_solapin, $Eprocincia, $Eusuario, $EperfilAtiendeActualmente)	
	Get_Nombre()	
	Get_Apellidos()	
	Get_Facultad()	
	Get_Numero_solapin()	
	Get_Provincia()	
	Get_Usuario()	
	Get_PerfilAtiendeActualmente()	
	Set_Nombre()	
	Set_Apellidos()	
	Set_Facultad()	
	Set_Numero_solapin()	
	Set_Provincia()	
	Set_Usuario()	
	Set_PerfilAtiendeActualmente()	
Descripción:	Controla la información relacionada a cursos del perfil de calidad.	

Nombre: Actividad		
Tipo de clase: Entidad		
Atributo		**Tipo**
$nombreActividad		
$asistencia		
$profesorResponsable		
$nombreCurso		
$nombreEstudiantesParticipan		
$tipoEvaluacion		
Nombre:	Profesor($EnombreActividad, $Easistencia, $EprofesorResponsable, $EnombreCurso, $EnombreEstudiantesParticipan, $EtipoEvaluacion)	
	Get_NombreActividad()	
	Get_Asistencia()	

	Get_ProfesorResponsable()
	Get_NombreCurso()
	Get_NombreEstudiantesParticipan()
	Get_TipoEvaluacion()
	Set_NombreActividad()
	Set_Asistencia()
	Set_ProfesorResponsable()
	Set_NombreCurso()
	Set_NombreEstudiantesParticipan()
	Set_TipoEvaluacion()
Descripción:	Controla la información relacionada a las actividades que se realizan en los cursos del perfil de calidad.

Anexo 3. Diagrama de Clases Persistentes.



Glosario de términos

Actor Abstracción de las entidades externas a un sistema, subsistemas o clases que interactúan directamente con el sistema. Un actor participa en un caso de uso o en conjunto coherente de casos de usos para llevar a cabo un propósito global.

Artefacto: Los artefactos o entregables son aquellos productos tangibles que son producidos, modificados y usados por las actividades. Incluyen modelos, elementos del modelo, código fuente y ejecutables.

Capítulo: Se le llama capítulo a la primera división que tiene el contenido. El contenido está agrupado en capítulos y estos a su vez en epígrafes.

Casos de Usos: Se utilizan para modelar cómo un sistema o negocio funciona actualmente, o cómo los usuarios desean que funcione.

Diagrama: Representación gráfica de una colección de elementos de modelado.

GRASP: Acrónimo de General Responsability Assignment Software Patterns (Patrones de Software para la asignación General de Responsabilidad), describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades.

Interfaz del usuario: Es la forma en que los usuarios pueden comunicarse con una computadora, y comprende todos los puntos de contacto entre el usuario y el equipo.

Metodología: Se refiere a los métodos de investigación en una ciencia.

Modelamiento del Negocio: Comprender el funcionamiento de la organización. Garantizar que los clientes, usuarios finales, desarrolladores y las otras partes interesadas tengan un entendimiento común de la organización. Derivar los requisitos de software necesarios para apoyar los procesos de la organización.

Patrones: Son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos.

Proceso: Conjunto de actividades o eventos que se realizan o suceden con un determinado fin.

Plan de Captura de Requisitos: Describe la estrategia a seguir durante la captura de requisitos, incluye las etapas, los entregables y los participantes.

Proceso de Negocio: Un proceso de negocio es un conjunto de tareas relacionadas lógicamente llevadas a cabo para lograr un resultado de negocio definido. Dos de las características importantes de los procesos son:

1. Tienen clientes (internos o externos).
2. Cruzan fronteras organizacionales, es decir, operan entre subunidades organizacionales.

Los procesos pueden definirse con base en tres dimensiones: Entidades, objetos y actividades.

Sistema Informático: Es la síntesis de hardware, software y de un soporte humano. Un sistema informático típico emplea un ordenador que usa dispositivos programables para almacenar, recuperar y procesar datos.

Tecnologías de la Información y las Comunicaciones (TIC): Conjunto de avances tecnológicos que proporciona la informática, las telecomunicaciones y las tecnologías audiovisuales, que comprenden los desarrollos relacionados con los ordenadores, Internet, la telefonía, los "mas media", las aplicaciones multimedia y la realidad virtual. Estas tecnologías básicamente proporcionan información, herramientas para su proceso y canales de comunicación.

UML: Lenguaje Unificado de Modelado. Es el lenguaje de modelado de sistema de software más conocido en la actualidad.