UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

"FACULTAD 2"

SISTEMA PARA LA GESTIÓN DE INFORMACIÓN DE LOS CURSOS OPTATIVOS EN LA FACULTAD 2 DE LA UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

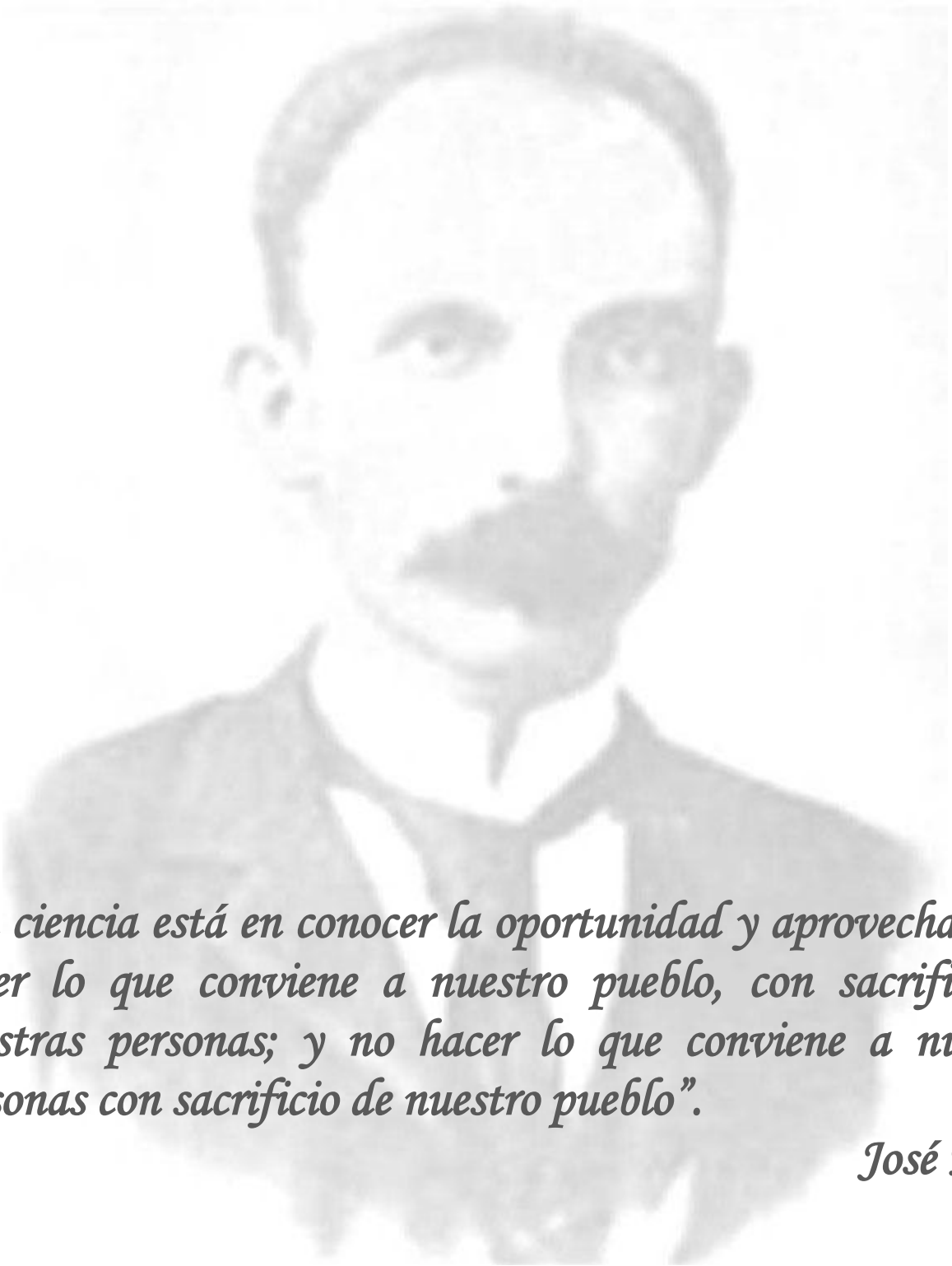
Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

AUTOR: FERNANDO MIGUEL PÉREZ GONZÁLEZ

TUTOR: ING. ARIEL DÍAZ RODRÍGUEZ
CO-TUTOR: ING. ANTONIO HERNÁNDEZ DOMÍNGUEZ

CIUDAD DE LA HABANA

JUNIO/2011



"La ciencia está en conocer la oportunidad y aprovecharla: es hacer lo que conviene a nuestro pueblo, con sacrificio de nuestras personas; y no hacer lo que conviene a nuestras personas con sacrificio de nuestro pueblo".

José Martí.

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al Centro de Telemática de la Facultad 2 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los 20 días del mes de Junio del año 2011.

_____

Fernando Miguel Pérez González

Autor

_____ _____

Ariel Díaz Rodríguez

Tutor

Antonio Hernández Domínguez

Co-Tutor

DATOS DE CONTACTO

Del Tutor:

Ing. Ariel Díaz Rodríguez (email: adrofriguez@uci.cu).

Ingeniero en Ciencias Informática graduado en la UCI en el año 2008.

Profesor Instructor.

Del Co-Tutor:

Ing. Antonio Hernández Domínguez (email: ahdominguez@uci.cu).

Ingeniero en Ciencias Informática graduado en la UCI en el año 2009.

AGRADECIMIENTOS

A mis padres Yolanda y Miguel, y mi padrastro Abelardo, por su apoyo incondicional cada vez que los necesité.

A mi novia Yanet, por estar en todo momento a mi lado, brindándome amor y cariño.

A mis tutores Ariel y Antonio, por su paciencia y entrega para culminar con éxito este trabajo de diploma.

A todos los profesores que con sus conocimientos y experiencias contribuyeron al perfeccionamiento de este material.

A mis compañeros, amigos y a todo el personal implicado en la investigación.

A los que me quieren, aprecian y acompañan hasta en los momentos finales, a todos ellos…

MUCHAS GRACIAS.

DEDICATORIA

A La Revolución Cubana.

Al Comandante en Jefe Fidel Castro Ruz.

A todos aquellos estudiantes y docentes que dedican su preciado tiempo en superarse para alcanzar una labor educativa eficiente.

RESUMEN

En la Universidad de las Ciencias Informáticas (UCI) existe constantemente un gran volumen de información relacionada con los diferentes Cursos Optativos que se imparten a los estudiantes, como parte del programa académico.

Actualmente en la Facultad 2 de la UCI, el proceso de gestión de los Cursos Optativos se realiza de forma manual, lo cual da lugar a una serie de deficiencias provocando inconformidades entre los implicados en dicho proceso.

Así surge la idea de desarrollar un Sistema de Gestión de la Información de los Cursos Optativos en la Facultad 2 de la UCI, el cual tiene como objetivo informatizar todo el proceso relacionado con la gestión de los Cursos Optativos en la Facultad 2.

En el presente documento se realiza una descripción completa de las diferentes etapas de construcción del sistema propuesto, en el cual se realiza un estudio de sistemas precedentes a nivel mundial, nacional y local, las diferentes tecnologías existentes para el desarrollo exitoso de este tipo de sistemas, seguido de las fases de análisis, diseño, implementación y prueba del software, culmina con un estudio de la factibilidad de la solución propuesta.

PALABRAS CLAVES:

Sistema de gestión, cursos optativos.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

La globalización de la economía es a la vez la globalización de la información, ello significa que más del 95% de las noticias internacionales son enviadas en cuestiones de segundos por Internet**.** En este escenario, el vertiginoso avance alcanzado por la ciencia y la técnica enriquecen considerablemente el conocimiento, a la vez que florecen nuevos conceptos y términos en las diferentes ramas del saber: una de estas ramas es precisamente la Informática.

En tanto, se reconoce por Informática, *el tratamiento racional, automático y adecuado de la información por medio del computador, para lo cual se diseñan y desarrollan aplicaciones especiales buscando seguridad e integridad*[1]. Visto así, se le concede una particular importancia a la misma porque permite a las empresas contar con la información oportuna para tomar decisiones efectivas, de ahí que las Tecnologías de las Informáticas y las Comunicaciones **(TIC)** ocupan un papel trascendental porque contribuyen a la obtención y procesamiento de mayor cantidad de datos que los medios manuales.

En Cuba, las TIC se aplican en todas las esferas de la vida, entre ellas: en el sector de la salud, los servicios, la educación, las investigaciones y la gestión económica, la cual presupone entre sus aspiraciones fundamentales, satisfacer en gran medida al cliente; así como, disminuir los gastos, aumentar la productividad, entre otros aspectos no menos importantes.

A partir de estos estudios, se confirma que en los centros educacionales del país, se imparten diferentes cursos optativos los cuales se dirigen a elevar el nivel cultural e integral de los educandos. Por ***curso optativo*** se entiende, el conjunto de conocimientos organizados de forma estructurada de cualquier materia o tema que reciben los estudiantes de manera optativa u opcional, con el objetivo de prepararlos profesionalmente y en este orden, desarrollar sus habilidades y capacidades.

[1] Biblioteca de Investigaciones. [En línea] [Citado el: 11 de junio del 2011] http://bibliotecadeinvestigaciones.wordpress.com/informatica/

Desde esta perspectiva, se subraya que los cursos optativos complementan electivamente el plan de estudio del programa académico y a su vez, contribuyen a la formación del estudiante en función de sus intereses y las necesidades del centro. Cada curso contiene en su estructura un programa analítico, el cual recoge en esencia los objetivos, temas, frecuencias, tipo de clases y otros datos de interés. La forma de evaluación es concebida a partir de una prueba de suficiencia o recibiendo el curso durante un período determinado y realizando una evaluación final como muestra de los conocimientos adquiridos.

Siendo consecuente con estos análisis, la Universidad de las Ciencias Informáticas **(UCI)** se caracteriza por ofertar una gran cantidad de cursos optativos sobre distintas materias entre las que se encuentran los siguientes: programación, diseño, gestión de calidad, matemática, ingeniería de software entre otros, en aras de contribuir a enriquecer en los estudiantes sus conocimientos, así como, formar habilidades y capacidades necesarias que los prepare para enfrentar de manera objetiva los retos y desafíos de su profesión.

Ahora bien, estos procesos se realizan de forma manual; lo que implica el empleo de tiempo y esfuerzo que atentan directamente al buen desempeño de los cursos optativos, escuchándose casi siempre las quejas por parte de la gran mayoría de los implicados.

A este hecho se suma, que la convocatoria de los cursos optativos se realiza por correo electrónico con fecha límite para los estudiantes. Posteriormente, la matrícula se debe efectuar en la Secretaría Docente perteneciente a la Facultad 2, donde los interesados acuden de forma masiva para confirmar la selección del curso optativo.

En este sentido, se subraya que el proceso se realiza de forma manual, provocando consigo conglomeración del personal, pérdida de tiempo y disgustos entre los interesados, dado a que se carece de un sistema automatizado que facilite el control eficiente de las variedades de cursos a impartir, la cantidad de estudiantes matriculados; así como, la relación de veces que los implicados han optado por dichos cursos y por este motivo no se satisface plenamente las expectativas por las que inicialmente fueron previstos estos cursos optativos.

A partir de estos argumentos, se infiere que el proceso de selección de matrícula de los cursos optativos puede ser evaluado de inestable porque varía en dependencia de las necesidades y el momento en que

han sido ofertados. Por otra parte, la coordinación del curso resulta un tanto complejo porque no favorece el intercambio directo entre los profesores y estudiantes para dar respuesta a las inquietudes que surjan.

Todas estas insuficiencias y limitaciones afectan en gran medida el buen desempeño y calidad del proceso, al no contar con un sistema automatizado que permita la gestión de los cursos optativos que se imparten en la Facultad 2; por lo que se declara la ***situación problémica*** siguiente: La gestión de los cursos optativos en la Facultad 2 al realizarse de forma manual acarrea un grupo de deficiencias e inconformidades, lo cual ocasiona demoras, gastos de tiempo y esfuerzo.

Luego de analizar la situación problémica, surge el ***problema científico*** siguiente: ¿Cómo facilitar el proceso de gestión de la información relacionada con los cursos optativos en la Facultad 2 de la Universidad de las Ciencias Informáticas?

El problema científico declarado se enmarca en el ***objeto de estudio*** siguiente: Proceso de gestión de los cursos optativos de la Universidad de las Ciencias Informáticas.

El objeto delimita el ***campo de acción*** siguiente: Proceso de gestión de los cursos optativos que se imparten en la Facultad 2 de la Universidad de las Ciencias Informáticas.

Para resolver el problema se propone el ***objetivo general*** siguiente: Desarrollar un sistema informático que gestione el proceso relacionado con los cursos optativos en la Facultad 2 de la Universidad de las Ciencias Informáticas.

Como ***objetivos específicos*** se declaran los siguientes:

- Definir el marco teórico de la investigación.
- Modelar el proceso de gestión de los cursos optativos en la Facultad 2 de la Universidad de las Ciencias Informáticas.
- Implementar el sistema de gestión de cursos optativos.
- Constatar el nivel de factibilidad del sistema propuesto.

Para dar cumplimientos a los objetivos trazados se desarrollarán las ***tareas de investigación*** siguientes:

- Realizar entrevistas para conocer cómo se desarrolla el proceso de gestión de los cursos optativos en la Facultad 2 de la Universidad de las Ciencias Informáticas.

- Identificar los problemas existentes en dicho proceso.
- Realizar un estudio de los sistemas precedentes existentes, tanto a nivel nacional como internacional.
- Identificar las herramientas, tecnologías y metodologías más adecuadas para la implementación del sistema.

Desde estos estudios se declara como ***idea a defender*** la siguiente: Con la implementación de un sistema para la gestión de información se contribuirá a mejorar en la Facultad 2 los procesos relacionados con la creación y matrícula de los cursos optativos, manteniendo segura la actualización de la información inherente a dicho proceso.

Para realizar las tareas de la investigación se emplearán los siguientes ***métodos científicos***:

Métodos teóricos:

Histórico lógico: En la primera parte de la investigación se desarrollará un estudio del estado del arte de la problemática; así como se analizarán las ventajas y desventajas de cada una de las herramientas a utilizar para el desarrollo del sistema informático para gestionar la información referente a los cursos optativos en la Facultad 2 de la Universidad de las Ciencias Informáticas.

Analítico – sintético: Se utilizará para captar y resumir varios documentos y bibliografías, los cuales servirán en gran medida para dar solución a la problemática planteada. De ellos se extraerán las ideas fundamentales y al mismo tiempo se detallará la información necesaria para el modelado correcto del negocio.

Métodos empíricos:

Entrevista: Se utilizará la entrevista como una conversación planificada con los clientes, para obtener información acerca del problema en cuestión. Su uso constituye un medio para el conocimiento real de cómo se realiza el proceso de gestión de los cursos optativos en la Facultad 2 de la Universidad de la Ciencia Informáticas. Todo esto puede influir en el posterior análisis y diseño del producto de software que se pretende implantar.

CAPÍTULO 1

FUNDAMENTACIÓN TEÓRICA

1.1 Introducción.

En la actualidad, la rama de la informática revoluciona a una velocidad increíble, razón por la que una persona podría dedicar gran parte de su vida al estudio y análisis de esta. El siguiente capítulo es el resultado de una minuciosa y breve investigación acerca de los conceptos esenciales para comprender los procesos de negocio asociados al dominio del problema, de los software existentes asociados al campo de acción, las nuevas tecnologías en este campo, las tendencias existentes en el mundo, lenguajes de implementación y una breve reseña del proceso de desarrollo de software utilizado para darle solución al problema planteado.

1.2 Estado del arte.

1.2.1 Conceptos Fundamentales.

Curso optativo.

Los cursos optativos son aquellos que se incluyen en el plan de estudio y de entre los cuales el estudiante selecciona una cantidad determinada para cursar en forma obligatoria. Los contenidos de estas asignaturas tienen como propósito ampliar y actualizar a los estudiantes sobre temas científicos relacionados con la profesión[2]. En la UCI, este concepto se adapta a las necesidades y objetivos de la universidad, de tal forma que los cursos optativos están incluidos dentro del plan de estudio del programa académico, pues cada estudiante debe haber cursado y aprobado un total de 8 cursos optativos para poder graduarse como Ingeniero en Ciencias Informáticas y que 5 de ellos respondan al perfil de la

[2] Gaceta Oficial de la República de Cuba. Reglamento para el Trabajo Docente y Metodológico en la Educación Superior. Resolución 210/07. Artículo 70. Pág. 217.

facultad a la cual pertenecen. Cada curso tiene un programa analítico que recoge los objetivos, temas, frecuencias, tipo de clases y otros datos. Estos son impartidos por profesores, en algunos casos auxiliados por estudiantes especializados en el tema. La forma de evaluación es mediante prueba de suficiencia o recibiendo el curso durante un período determinado.

1.2.2 Sistemas de gestión de información.

La Informática en la gestión de la información.

La informática es el conjunto de conocimientos científicos y técnicas que hacen posible el tratamiento automático de la información por medio de ordenadores[3]. Para ello los sistemas informáticos deben captar la información, que esta sea procesada o tratada para luego mostrar o transmitir los resultados. La informática se aplica a diversas áreas, gestión de negocio, almacenamiento de información, monitorización y control de procesos de investigación, desarrollo de juegos, diseño computarizado, aplicaciones/herramientas multimedia. La informática pretende brindar sistemas de gestión cada vez más eficientes a través de los cuales se logre la integración coherente de grandes cantidades de datos, al permitir la organización y calidad de los mismos, con lo cual se afirma que durante los últimos años con el desarrollo de herramientas y aplicaciones informáticas se han realizado grandes avances en la manipulación de grandes cantidades de datos.

Sistemas de Gestión de la Información (SGI).

La definición de Sistema para la Gestión de Información, desde el punto de vista de la lógica de Marketing, resulta particularmente interesante. En efecto, sugiere que sea el propio sistema (y no el humano) el sujeto de la gestión. Una definición más apropiada podría ser la de "Sistema de Soporte a la Gestión de Información" ya que, en realidad, *son las estrategias de comunicación las que realmente llevan a gestionar información de forma efectiva; los sistemas informáticos pueden a lo sumo proporcionar las herramientas necesarias, o bien incluir servicios de soporte a la toma de decisiones por lo que a la Gestión de Información se refiere*[4]. Muchos sistemas proporcionan diferentes niveles de acceso dependiendo del usuario. El acceso a los Sistemas para la Gestión de Información se realiza generalmente a través del

[3] Diccionario de la Real Academia Española [En línea] [Citado el: 7 de junio del 2011] http://buscon.rae.es/draeI/SrvltConsulta?TIPO_BUS=3&LEMA=informática
[4] Cultura Gráfica – Chile. [En línea] [Citado el: 11 de junio del 2011] http://www.culturagrafica.cl/chile/?p=115

navegador web, y a veces se requiere el uso de File Transfer Protocol **(FTP)** para subir contenidos. *Los sistemas de gestión de información están concebidos para capturar, actualizar, integrar, consultar y analizar información pertinente a la organización a la cual pertenecen*[5].

Beneficios de los sistemas de gestión de la información.

Entre los principales beneficios que proporcionan los sistemas de gestión de información se encuentran:

- Velocidad de procesamiento de la información y obtención de resultados.
- Consultas de la información.
- Productividad del trabajo.
- Exactitud y consistencia de los datos.
- Interrelación de las áreas con el sistema.
- Seguridad de la información.

1.2.3 Actualidad del trabajo.

Actualmente, en todo el mundo, existen mecanismos y software que permiten gestionar información docente y controlar mediante estrategias trazadas y planificadas las asignaturas que conforman el plan de estudio de cualquier carrera o cursos optativos que se brindan u ofertan a estudiantes para permitir su capacitación. Los software de gestión de información académica son muy utilizados en cualquier centro de enseñanza gracias a las comodidades que ellos proporcionan al claustro de profesores y a los estudiantes.

A Nivel Mundial.

En el mundo, existen diferentes universidades en las cuales se imparten cursos optativos a los estudiantes para complementar su preparación como futuros profesionales, pero en ninguno de los casos a lo largo de la investigación, se pudo comprobar la existencia de algún software encargado de la gestión de la información de estos cursos optativos.

[5] Biblioteca Virtual en Salud de Cuba [En Línea] [Citado el: 11 de junio del 2011] http://bvs.sld.cu/revistas/aci/vol14_4_06/aci11406.htm

En la Facultad de Informática de la Universidad Nacional de la Plata el alumno debe cursar 5 Optativas de ½ punto entre 4to y 5to año, elegidas con una distribución por Área de 3-1-1 (es decir 3 materias de un área y 1 de cada una de las otras dos) entre el Área de Fundamentos, el Área Arquitectura, Sistemas Operativos y Redes y el Área Algoritmos y Lenguajes, la información se gestiona de forma manual[6].

En la Universidad Tecnológica Equinoccial de Ecuador los cursos optativos que se ofrecen a los alumnos son sobre temas especiales de informática y computación, relacionados con las carreras profesionales que se dictan en las distintas Facultades. Todos los cursos optativos tienen un costo que es fijado en función del número de horas y del costo unitario establecido para cada año lectivo por las autoridades de la Universidad, para aprobar los cursos optativos dictados, el alumno debe obtener un promedio mayor o igual al 70% de la calificación máxima establecida y acreditar una asistencia no menor al 70% de las clases dictadas[7]. Todo este proceso es de forma manual.

A Nivel Nacional.

En el sistema de universidades del Ministerio de Educación Superior **(MES)** se ha investigado que existen cursos optativos de diferentes materias por cada carrera y que para gestionar este proceso a la hora de almacenar la información de los cursos que se imparten y las notas obtenidas por los estudiantes que cursan los mismos, no existe ningún software con estos fines. Se cuenta con algunos para la Gestión Académica como es el caso de *GESTACAD 1.1: Gestión Académica Universitaria de la Universidad de Matanzas*[8].

En la UCI.

En la Universidad de las Ciencias Informáticas se cuenta con un Sistema de Gestión Académica denominado Akademos que por ser para la gestión a nivel general permite el almacenamiento de los datos, por facultad, de los resultados obtenidos en las pruebas realizadas, la matrícula estudiantil, asistencia en clases y causas que describen la situación del ausente, resultados de los cursos optativos,

[6] Facultad de Informática | UNLP. [En línea] [Citado el: 18 de febrero del 2011.] http://www.info.unlp.edu.ar/lic_informatica_optativas2011.

[7] Universidad Tecnológica Equinoccial | UTE. [En línea] [Cita el: 18 de febrero del 2011] http://www.ute.edu.ec/Default.aspx?idSeccion=113&idCategoria=165

[8] Calderín Delgado, Yanoski. GESTACAD. Sistema para la Gestión Académica. [En línea] [Citado el 10 de febrero del 2011] http://www.posgrados.frc.utn.edu.ar/congreso/trabajos/7.doc.

entre otras; este último no se adapta a las necesidades específicas de la facultad puesto que no permite que se realicen de forma rápida la matrícula de los estudiantes y el proceso de gestión de los cursos optativos, además de que los permisos para acceder a la información de forma inmediata son restringidos y en muchos casos, nulos.

1.3 Descripción del tipo de aplicación.

Existen dos tipos de aplicaciones informáticas: Aplicaciones de Escritorio (o Desktop como se les conoce en el lenguaje informático) y Aplicaciones Web.

La aplicación Desktop no resulta la solución más eficiente para el problema que se presenta, pues centraliza la gestión de la información a una o sólo algunas estaciones de trabajo donde se hace necesario instalar todas las herramientas que requiere para su funcionamiento, limitando el uso del software a aquellos usuarios que tengan acceso a dichas computadoras.

Por otra parte, una aplicación web ofrece facilidades que pudieran ser la solución a los problemas planteados al inicio del trabajo de diploma. Primeramente, permite consultar la información contenida desde cualquier estación de trabajo conectada a la Intranet de la Institución. También, las actualizaciones o mejoras que se le realicen a la aplicación resultan transparentes al usuario final pues se realizan solamente en la estación que actúa como servidor de aplicaciones web independiente completamente de la estación cliente. Para acceder a la aplicación sólo se necesita de un navegador web como cliente ligero y la conexión puede ser desde cualquier máquina que esté conectada a la red.

1.4 Tecnologías, herramientas y metodología a utilizar.

Como ***propuesta de solución*** se tiene desarrollar una aplicación web que gestione la información relacionada con los cursos optativos en la Facultad 2. Para el desarrollo del sistema se investigaron varias tecnologías que se utilizan en el mundo, de las cuales fueron elegidas algunas según sus ventajas y facilidades para dar solución al problema.

1.4.1 Lenguajes de programación para la web.

Actualmente existen diferentes lenguajes de programación para desarrollar en la web, estos han ido surgiendo debido a las tendencias y necesidades de las plataformas. Desde los inicios de Internet, fueron surgiendo diferentes demandas por los usuarios y se dieron soluciones mediante lenguajes estáticos. A medida que pasó el tiempo, las tecnologías fueron desarrollándose y surgieron nuevos problemas a dar solución. Esto dio lugar a desarrollar lenguajes de programación dinámicos para la web, que permitieran interactuar con los usuarios y utilizaran sistemas de Bases de Datos[9].

Lenguaje HyperText Markup Language (HTML).

Desde el surgimiento de internet se han publicado sitios web gracias al lenguaje HTML. Es un lenguaje estático para el desarrollo de sitios web (acrónimo en inglés de HyperText Markup Language, en español Lenguaje de Marcas Hipertextuales). Desarrollado por el World Wide Web Consortium ***(W3C)****. Los archivos pueden tener las extensiones (htm, html)*[10]**.**

Ventajas[11]**:**

- Lenguaje sencillo que permite describir hipertexto.
- Texto presentado de forma estructurada y agradable.
- No necesita de grandes conocimientos cuando se cuenta con un editor de páginas Web o WYSIWYG.
- Archivos pequeños.
- Despliegue rápido.
- Soportado por la totalidad de los navegadores.

[9] Maestros de la Web. [En Línea] [Citado el 11 de junio de 2011] http://www.maestrosdelweb.com/principiantes/los-diferentes-lenguajes-de-programacion-para-la-web/
[10] Ibídem.
[11] Ibídem.

Lenguaje JavaScript.

JavaScript es un lenguaje de scripting basado en objetos, utilizado para acceder a objetos en aplicaciones. Principalmente, se utiliza integrado en un navegador web permitiendo el desarrollo de interfaces de usuario mejoradas y páginas web dinámicas[12].

Ventajas[13]**:**

- Lenguaje de scripting seguro y fiable.
- Los script tienen capacidades limitadas, por razones de seguridad.
- El código JavaScript se ejecuta en el cliente.

Una de las excelentes prácticas de programación en la arquitectura Cliente-Servidor es la validación de los datos tanto en el lado del cliente como en el lado del servidor. JavaScript es considerado un sencillo y completo lenguaje de programación para la web, el cual brinda dentro de sus principales funcionalidades la posibilidad de validar los datos entrados por los usuarios en el lado del cliente, evitando de esta forma que lleguen al servidor un grupo de datos erróneos. Es soportado además por la casi totalidad de los navegadores web.

Language Hypertext Pre-processor (PHP).

Es un lenguaje de programación utilizado para la creación de aplicaciones web. PHP es un lenguaje de script interpretado en el lado del servidor utilizado para la generación de páginas web dinámicas, embebidas en páginas HTML y ejecutadas en el servidor. PHP no necesita ser compilado para ejecutarse. Para su funcionamiento necesita tener instalado Apache o Internet Information Server ***(IIS)*** *con las librerías de PHP. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características específicas. Los archivos cuentan con la extensión (.php)*[14].

Ventajas[15]**:**

[12] Ibídem.
[13] Ibídem.
[14] Ibídem.
[15] Ibídem.

- Baja complejidad para aprenderlo.
- Se caracteriza por ser un lenguaje de rápida interpretación.
- Soporta los aspectos de la orientación a objeto como son clases y herencia.
- Es un lenguaje multiplataforma.
- Capacidad de conexión con la mayoría de los gestores de base de datos: MySQL, PostgreSQL, Oracle, MS SQL Server, entre otros.
- Capacidad de expandir su potencial utilizando módulos.
- Posee una fuerte documentación, la cual incluye descripciones y ejemplos de cada una de sus funciones.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Incluye gran cantidad de funciones.
- No requiere definición de tipos de variables.

PHP es el lenguaje seleccionado porque además de las ventajas que proporciona, brinda una gran seguridad. Además PHP no obliga a quien lo usa a seguir una determinada metodología a la hora de programar, y aunque esté dirigido a alguna en particular, el programador puede aplicar en su trabajo cualquier técnica de programación y/o desarrollo que le permita escribir código ordenado, estructurado y manejable.

1.4.2 Frameworks para PHP.

En la actualidad existen más de 40 frameworks disponibles en la comunidad de PHP[16]*,* por lo que se hace cada día más difícil decidir qué Framework funciona mejor, sobre todo porque cada uno de ellos ofrece diferentes funcionalidades.

Symfony.

Symfony 1.0.14 es el framework seleccionado para el desarrollo del sistema puesto que es un completo framework diseñado para optimizar el desarrollo de las aplicaciones web mediante algunas de sus principales características. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el

[16] Comunidad Hispana de PHP. [En línea] [Citado el 18 de febrero del 2011.] http://www.php-hispano.net/foros/frameworks/24047-listado-de-frameworks-php

tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web.

Symfony está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y Microsoft SQL Server. Se puede ejecutar tanto en plataformas Linux como en plataformas Windows[17]. Symfony se diseñó para que se ajustara a los siguientes requisitos:

- Fácil de instalar y configurar en la mayoría de plataformas.
- Independiente del sistema gestor de bases de datos.
- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Basado en la premisa de "*convenir en vez de configurar*", en la que el desarrollador sólo debe configurar aquello que no es convencional.
- Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- Preparado para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- Código fácil de leer que incluye comentarios de phpDocumentor y que permite un mantenimiento muy sencillo.
- Fácil de extender, lo que permite su integración con las librerías de otros fabricantes.

1.4.3 Object-Relational Mapping (ORM).

Las bases de datos son relacionales. PHP 5 y Symfony están orientados a objetos. Para acceder de forma efectiva a la base de datos desde un contexto orientado a objetos, es necesaria una interfaz que traduzca la lógica de los objetos a la lógica relacional. Esta interfaz se llama ORM (object-relational mapping o

[17] Potencier, Fabien, Zaninotto, François. *Symfony la guía definitiva.* [PDF]. www.librosweb.es. pág. 12.

mapeo de objetos a bases de datos), y está formada por objetos que permiten acceder a los datos y que contienen en sí mismos el código necesario para hacerlo[18].

Propel.

Propel es un ORM para PHP que facilita la labor de desarrollo de aplicaciones web, gracias a la capa que transforma el tratamiento de la Base de Datos mediante objetos, con la que se puede realizar las diferentes acciones de obtener, insertar, eliminar y modificar datos. *Propel, que también es un proyecto de software libre, es una de las mejores capas de abstracción de objetos/relacional disponibles en PHP 5. Está completamente integrado en Symfony*[19].

1.4.4 Sistema Gestor de Base de Datos.

PostgreSQL.

PostgreSQL es un sistema gestor de base de datos objeto-relacional (Object Relational Database Manager System), basado en POSTGRES Versión 8.4.2, desarrollado en la Universidad de California, en el Departamento de Ciencias de la Computación de Berkeley. Postgres es pionero en muchos conceptos que sólo estuvieron disponibles en algunos sistemas de bases de datos comerciales mucho más tarde. PostgreSQL es un descendiente del "código abierto". Soporta gran parte del SQL estándar y muchas modernas funcionalidades como:

- Consultas complejas.
- Llaves foráneas.
- Disparadores.
- Vistas.
- Integridad transaccional.
- Control de versionado concurrente. Estrategia de almacenamiento que permite trabajar con grandes volúmenes de datos.

[18] Ibídem – pág. 197.

[19] Ibídem – pág. 22.

Cumple completamente con las características ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad) para realizar transacciones seguras, es multiplataforma. Posee interfaces nativas para lenguajes como ODBC, JDBC, C, C++, PHP, PERL, TCL, ECPG; PYTHON y RUBY, además de traer soporte para la herencia y la seguridad de la capa de dispositivo de transportación de datos. Además, PostgreSQL puede ser personalizado por el usuario en muchas formas, según sus necesidades.

Debido a la liberación de la licencia, PostgreSQL puede ser utilizado, modificado y distribuido por cualquiera gratuitamente, para cualquier propósito ya sea con fines privados, comerciales o académicos. Por todo lo antes mencionado es que PostgreSQL es el servidor de base de datos seleccionado.

1.4.5 Metodologías de desarrollo de software.

Rational Unified Process (RUP).

RUP es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado (UML), constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos[20]. RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización, utiliza UML para definir los modelos de software y se divide en 4 fases: Inicio, Elaboración, Construcción y Transición.

Inicio: Tiene como objetivo determinar la visión del proyecto.
Elaboración: En esta etapa el objetivo es determinar la arquitectura óptima.
Construcción: El objetivo es obtener la capacidad operacional inicial.
Transición: Obtener el reléase del producto.

En RUP se han agrupado las actividades en grupos lógicos definiéndose 9 flujos de trabajo principales, los 6 primeros son conocidos como flujos de ingeniería y los tres últimos como flujos de apoyo. Tiene tres características esenciales: dirigido por casos de uso, centrado en la arquitectura e iterativo e incremental.

Atendiendo a que el proceso de gestión de cursos optativos en la universidad y específicamente en la facultad está propenso a sufrir constantes cambios, es seleccionada RUP como metodología de desarrollo del sistema propuesto por ser de tipo robusta y generar una gran documentación. Precisamente esta gran

[20] Quispe Carita, Vilma, Metodología RUP (Rational Unified Process). [PDF]. Puno-Perú. 2011.

documentación simplificaría en gran medida el proceso de realizar algún tipo de cambio al sistema desde el punto de vista de su funcionamiento.

1.4.6 Lenguaje de Modelado.

Unified Modeling Language (UML).

El Lenguaje Unificado de Modelado es un lenguaje gráfico para especificar, construir, visualizar, documentar las partes o artefactos que son información utilizada y originada mediante un proceso de software. Es un lenguaje estándar de modelado orientado a objetos.

Los principales beneficios de UML son:

- Modelar sistemas utilizando conceptos orientados a objetos.
- Establecer conceptos y artefactos ejecutables.
- Encaminar el desarrollo del escalamiento en sistemas complejos de misión crítica.
- Crear un lenguaje de modelado utilizado tanto por humanos como por máquinas.
- Mejor soporte a la planeación y al control de proyectos.
- Alta reutilización y minimización de costos.

Es importante resaltar que UML es un lenguaje para especificar y no para describir métodos o procesos. Se utiliza para definir un sistema de software, detallar los artefactos en el sistema, documentar y construir. Es el lenguaje en el que está descrito el modelo. Se puede aplicar en una gran variedad de formas para dar soporte a una metodología de desarrollo de software tal como RUP, pero no especifica en sí mismo qué metodología o proceso usar.

1.4.7 Herramientas Case.

Visual Paradigm.

Visual Paradigm es una herramienta para visualizar y diseñar elementos de software, para ello utiliza UML y ofrece una gama de facilidades para el modelado de aplicaciones. Está orientada a la creación de diseños usando el paradigma de programación orientada a objetos. Tiene dentro de sus características que es portable y posee gran facilidad de uso. Su diseño se centra en casos de uso y se enfoca al negocio

que genera un software de mayor calidad. También tiene disponibilidad en múltiples plataformas y usa un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación. Soporta el ciclo de vida completo del desarrollo de software y es libre.

Por todas estas razones se decide utilizar Visual Paradigm como Herramienta Case para el desarrollo del sistema propuesto.

1.5 Conclusiones.

En este capítulo se han introducido conceptos indispensables para la comprensión del proceso de distribución de la información, con el uso de las tecnologías actuales. La mejor solución al problema de gestión de los cursos optativos en la Facultad 2 es la implementación de un sistema basado en web. Se hace necesaria la construcción de un software que ayude a controlar el proceso de gestión de los Cursos Optativos en la Facultad 2 de la Universidad de las Ciencias Informáticas. También se detallaron las condiciones y problemas que rodean al objeto de estudio; y a través de los conceptos y definiciones planteados, se determinaron las situaciones específicas que rodean al problema y en base a esto se obtuvieron los objetivos generales y específicos para el presente trabajo**.**

CAPÍTULO 2

CARACTERÍSTICAS DEL SISTEMA

2.1 Introducción.

En el siguiente capítulo se describen las reglas del negocio asociadas al dominio del problema, así como los flujos de trabajo, ofreciendo una visión general del funcionamiento de la entidad. Se hace referencia a los requisitos funcionales y no funcionales que debe cumplir el sistema que se propone, lo que permite hacer una concepción general del mismo. Además, nos permite identificar mediante un Diagrama de Casos de Uso, las relaciones de los actores que interactúan con el sistema, reflejándose las acciones que se ejecutan o llevan a cabo.

2.2 Reglas del negocio.

En la Universidad de las Ciencias Informáticas se tiene como característica que cada estudiante tiene que graduarse como Ingeniero en Ciencias Informáticas y con un perfil en dependencia de la facultad donde desarrolla la carrera. Para poder aprobar el perfil, el estudiante debe recibir varios cursos básicos y optativos que le tributan a este, y otros que le permiten aumentar su conocimiento y desarrollar habilidades y capacidades, y de forma general, todos tributan al promedio final. En relación con esto, el negocio comienza por la necesidad de que los estudiantes tengan acreditados y aprobados cierto número de cursos optativos. Se envía por correo electrónico a todos los estudiantes una convocatoria de los cursos optativos que se ofertarán, donde se explica cómo será el procedimiento de la matrícula y a quién estará dirigido cada curso en dependencia del año que cursa, se envía los datos del profesor que lo impartirá, la capacidad y la fecha de inicio. Durante un plazo determinado el estudiante pasa con su solapín por la Secretaría Docente y se matricula en un curso según el que desee y las condiciones que debe cumplir para esto, todo depende de la necesidad de la Facultad y la estrategia que se tome en este sentido. Una vez vencido el plazo, se conforman los grupos de cada curso y se le entrega o envía por vía

electrónica al que lo impartirá, quien se encargaría de informar a sus estudiantes sobre todo lo referente al curso hasta su culminación.

Una vez vencida la fecha límite de la matrícula, el profesor que impartirá el curso, recibe el Acta de Examen e Informe de Evaluación Final con el listado de los estudiantes matriculados.

El negocio termina cuando culmina el curso optativo, el profesor conforma el Acta de evaluaciones finales de los estudiantes a los cuales se les impartió el curso, es entregada el acta en Secretaría Docente y se registra en el Sistema de Gestión Académica (Akademos) la calificación alcanzada por cada estudiante matriculado en el curso.

2.2.1 Descripción de los procesos del negocio.

En cualquier universidad nacional o extranjera se gestiona la información inherente a las materias académicas que recibe el estudiante mediante los diferentes procesos y estrategias que se lleven a cabo en la dirección general. La gestión de los cursos optativos en la Facultad 2 de la Universidad de las Ciencias Informáticas es el campo de acción, y sobre él se basa la construcción del Modelo del Negocio: reflejo de lo que desde el punto de vista real ocurre. En dicho proceso participan diferentes actores y trabajadores del negocio que toman diferentes roles para lograr la gestión de toda la información y que interactúan con los Casos de Uso del Negocio.

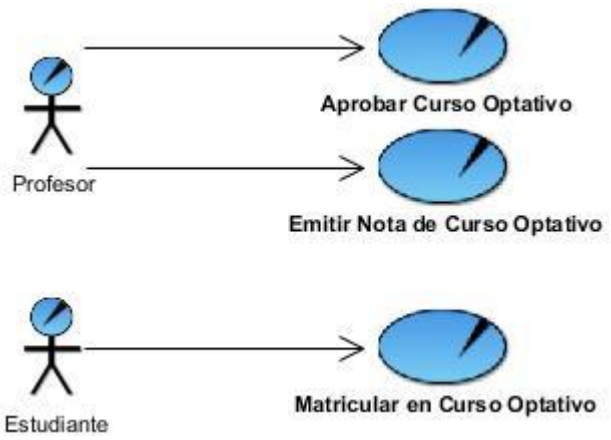
Tabla 1: Descripción de los Actores del Negocio.

Actores del Negocio	Descripción
Estudiante	El estudiante es quien inicia la acción Matricular Estudiante y al mismo tiempo es el principal beneficiado con el resultado de dicho proceso del negocio.
Profesor	Es quien inicia el proceso de negocio Aprobar Curso Optativo y a la vez es el beneficiado en la aprobación o no del Curso Optativo, es además el encargado de Emitir la Nota Final de los estudiantes matriculados en el curso.

Tabla 2: Descripción de los Trabajadores del Negocio.

Trabajadores del Negocio	Descripción
Jefe de Departamento	Es el encargado de aprobar o rechazar la realización de un determinado curso optativo, emite la información a superiores y al profesor. No se beneficia en ningún momento de las acciones ejecutadas en los procesos de negocio, sino que se limita a ejecutarlas.
Secretaria Docente	Es el encargado de registrar los cursos optativos, orienta al estudiante en la matrícula, toma los datos del estudiante y envía las evaluaciones de los estudiantes a la aplicación: Akademos. No se beneficia en ningún momento de las acciones ejecutadas en los procesos de negocio, sino que se limita a ejecutarlas.
Jefe de Cursos Optativos	Es el encargado de llenar el expediente de proyecto de cada estudiante. No se beneficia en ningún momento de las acciones ejecutadas en los procesos de negocio, sino que se limita a ejecutarlas.

Figura 1. Diagrama de Casos de Uso del Negocio.



2.3 Especificación de los requisitos de software.

Los requisitos son capacidades y condiciones con las cuales debe cumplir el sistema ya sean funcionales o no funcionales[21].

2.3.1 Relación de los Requisitos Funcionales (RF).

Los requisitos funcionales indican el comportamiento del sistema. Posteriormente estos requisitos son modelados a través del diagrama de casos de uso del sistema[22].

RF1: Gestionar Curso Optativo.

Permite insertar, actualizar y eliminar un Curso Optativo.

RF1.1 Registrar Curso Optativo.

RF1.2 Actualizar Curso Optativo.

RF1.3 Eliminar Curso Optativo.

RF2: Mostrar Curso Optativo.

Permite mostrar los Cursos Optativos que se ofertan.

RF3: Gestionar matrícula.

Permite realizar y eliminar la matrícula del Curso Optativo.

RF3.1: Registrar matrícula.

RF3.2: Eliminar matrícula.

RF4: Mostrar Pre-requisitos del Curso Optativo.

Permite mostrar los pre-requisitos que deben cumplirse para la matriculación en un curso optativo.

RF5: Mostrar resultado de Curso Optativo.

Permite a cada estudiante visualizar la evaluación obtenida en cada curso optativo.

---

[21] Jacobson, Ivar, Booch, Grady y Rumbaugh, James. *El proceso Unificado de Desarrollo de Software.* 1era edición. Madrid : Pearson Educación, S.A. - Addison Wesley, 2000. 84-7829-036-2. pág. 105.

[22] Ibídem – pág. 109.

RF6: Autenticar usuario.

Cada usuario debe autenticarse para poder acceder a algunas de las secciones que le brinda la aplicación.

RF7: Gestionar Usuario.

Se registrará usuarios dentro del sistema y se le asignaran permisos para que puedan interactuar con las diferentes funcionalidades, además se podrá eliminar cualquiera de estos o modificar los permisos.

RF 7.1 Registrar usuario.

RF7.2 Actualizar usuario.

RF7.3 Eliminar usuario.

2.3.2 Relación de los Requisitos No Funcionales.

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener[23], o sea, características que hagan al producto atractivo, rápido, usable o confiable. Están estrechamente vinculados a los requisitos funcionales, puesto que una vez que está definido lo que el sistema debe hacer, es necesario especificar cómo ha de hacerlo. Pueden llegar a marcar la diferencia entre un producto bien aceptado por los clientes y usuarios y uno de poca o ninguna calidad y aceptación.

- **Apariencia o Interfaz externa.**

  Se establece un plantilla general para toda la aplicación, de forma tal que la información sólo varía en la parte central evitando que los usuarios se pierdan durante la navegación.

  No se utilizan tecnologías de frames puesto que no son soportados por algunos navegadores web.

  Cada página no debe exceder los 500 kb en imágenes.

- **Usabilidad.**

  Garantizar un acceso fácil y rápido a los usuarios. El subsistema podrá ser usado por cualquier persona que posea conocimientos básicos en el manejo de una computadora y de un ambiente Web en sentido general.

[23] Ibídem – pág. 110.

- **Rendimiento.**

  La aplicación debe tener un alto nivel de respuesta. El tiempo de demora para páginas estática no excederá los 3 segundos. Las páginas dinámicas con alto contenido de información no deberán demorar más de 5 segundos en mostrarse completamente.

- **Portabilidad.**

  La aplicación debe ser multiplataforma (Windows/Linux).

- **Seguridad.**

  Se debe garantizar que a cada usuario sólo sea mostrada la información que le compete y antes de mostrar dicha información verificar que la credencial que posee es la adecuada para poder acceder a esa sesión del sistema. Cualquier usuario puede visualizar la información general dentro del sistema, pero sólo los usuarios autenticados y que ocupen los roles de Administrador, Secretario o Profesor del sistema podrán realizar las acciones específicas dentro del mismo. De igual forma la matrícula de los cursos optativos sólo estará habilitada para los estudiantes de la Facultad 2.

- **Software.**

  Se utilizará como lenguaje de desarrollo PHP5, como framework PHP Symfony 1.0.14 y como SGBD Postgres 8.4.2. El servidor web para la publicación del sistema será Apache 2.2 y los usuarios para poder acceder al mismo deberán tener instalado en las estaciones cliente algún navegador web.

- **Hardware.**

  El servidor donde se publica el sistema debe poseer las siguientes características:

  - 5 GB de capacidad en Disco Duro.
  - Procesador Pentium IV Genuino Hypertrading 3.0 GHz o superior.
  - 2 GB de memoria RAM como mínimo.

  Se requiere además que el servidor posea una tarjeta de red y que en la estación cliente de la Secretaria Docente exista una impresora.

- **Confiabilidad.**
  Tiene un alto grado de confiabilidad, la información para ser almacenada en la base de datos es codificada con la funcionalidad UTF8_ENCODE y de igual manera en el momento de mostrar la información a los usuarios esta es decodificada con la funcionalidad UTF8_DECODE.

- **Ayuda y documentación en línea.**
  Debe contar con un manual de usuario, que permite la aclaración de cualquier duda al interactuar con la aplicación.

- **Administración.**
  La administración estará organizada por paquetes de programación. Alguna modificación en la programación debe hacerse en el paquete correspondiente sin afectar al resto.

2.4 Definición de los Procesos del sistema.

2.4.1 Descripción del sistema propuesto.

El sistema a desarrollar contará de una interfaz sencilla y fácil de usar, diseñada para usuarios no adiestrados en el campo de la informática, con el objetivo de simplificar la interacción de los mismos con el sistema, este se implementará en PHP y se diseñará en forma de ventanas, o sea mediante elementos desplegables por donde se puedan seleccionar las opciones deseadas sin necesidad de la entrada excesiva de datos. Se realizará además un mapeo de roles para asignar a cada rol participante únicamente las funcionalidades que le corresponden evitando así pérdidas de información e inseguridad de la misma, respetando siempre la disponibilidad de la información que a cada rol concierne, por ejemplo:

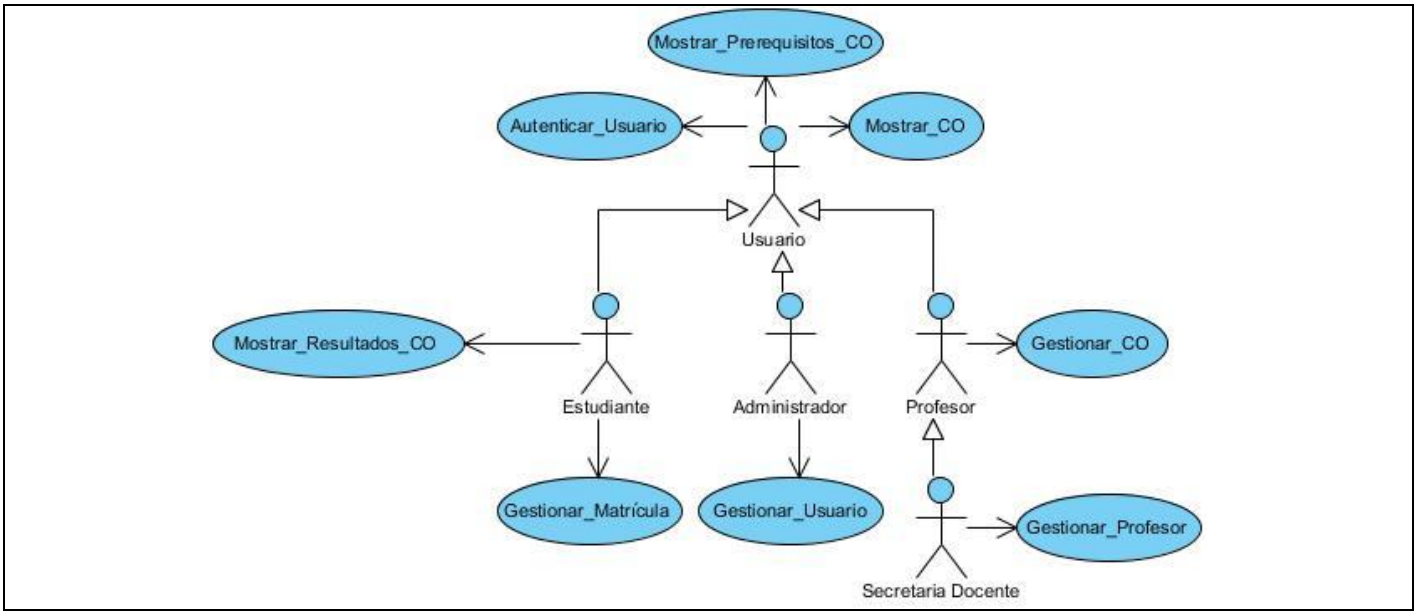
Los estudiantes sólo tendrán el acceso a la información necesaria para matricularse o darse baja de un cursos optativo, y solamente tendrán visibles aquellos cursos optativos que hayan sido previamente aprobados y habilitados.

Definición de los actores del sistema.

Tabla 3: Descripción de los Actores del Sistema.

Actor	Descripción
Estudiante	Es el encargado de realizar, o eliminar la matricula al curso optativo. Podrá visualizar los resultados obtenidos en los cursos optativos que ha recibido.
Profesor	Es el encargado de adicionar, modificar y eliminar los cursos optativos. Se ocupa de definir la nota de cada uno de los estudiantes una vez finalizado el curso optativo.
Secretaria Docente	Es el encargado de adicionar al sistema los profesores que tendrán privilegios sobre el mismo y que más tarde podrán gestionar la información de los cursos optativos. Puede además gestionar la información de los cursos optativos directamente.
Administrador	Es el encargado de registrar, actualizar y eliminar los usuarios del sistema, estableciendo los permisos para cada uno de estos y las diferentes acciones que podrán ejecutar dentro del sistema.

Figura 2. Diagrama de Casos de Uso del Sistema.



2.5 Conclusiones.

En el presente capítulo se comenzó a desarrollar la propuesta de solución, y a través de los procesos de negocio y el levantamiento de requisitos, se pudo obtener un listado con las funciones que debe tener el sistema y se describieron paso a paso todas las acciones del actor del sistema con los casos de uso con los que interactúa. A partir de esto se comenzará a desarrollar la posterior etapa que sería el análisis y diseño.

CAPÍTULO 3

ANÁLISIS Y DISEÑO

3.1 Introducción.

El flujo de trabajo de análisis y diseño tiene un papel protagónico en la fase de elaboración. Su principal objetivo es traducir los requisitos a una especificación que describa cómo implementar el sistema. Con el análisis se obtiene una visión del sistema enmarcada en el *qué hace*, de modo que sólo se interesa por los requisitos funcionales y el diseño es un refinamiento del análisis que tiene en cuenta los requisitos no funcionales, o sea el *cómo* cumple el sistema sus objetivos. El resultado final más importante en este flujo de trabajo será el modelo de diseño y otro producto importante es la documentación de la arquitectura del software, que captura varias visiones arquitectónicas del sistema.

3.2 Análisis.

Para crear una aplicación de software hay que describir el problema y las necesidades o requerimientos: en qué consiste el conflicto y que debe hacerse. El Análisis se centra en una investigación del problema, no en la manera de definir una solución[24].

[24] Larman, Craig. *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* 2da. México : Pearson Education, 2003. pág. 6. 8420534382.

3.2.1 Diagramas de Clases del Análisis (DCA).

A continuación se modelan solamente los DCA de los CUS Críticos.

Figura 3. Diagrama de Clases del Análisis. CU Autenticar Usuario.

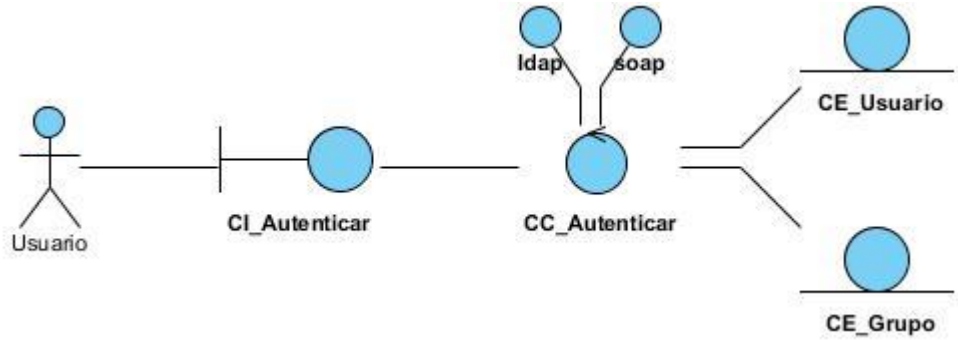


Figura 4. Diagrama de Clases del Análisis. CU Gestionar Usuario.

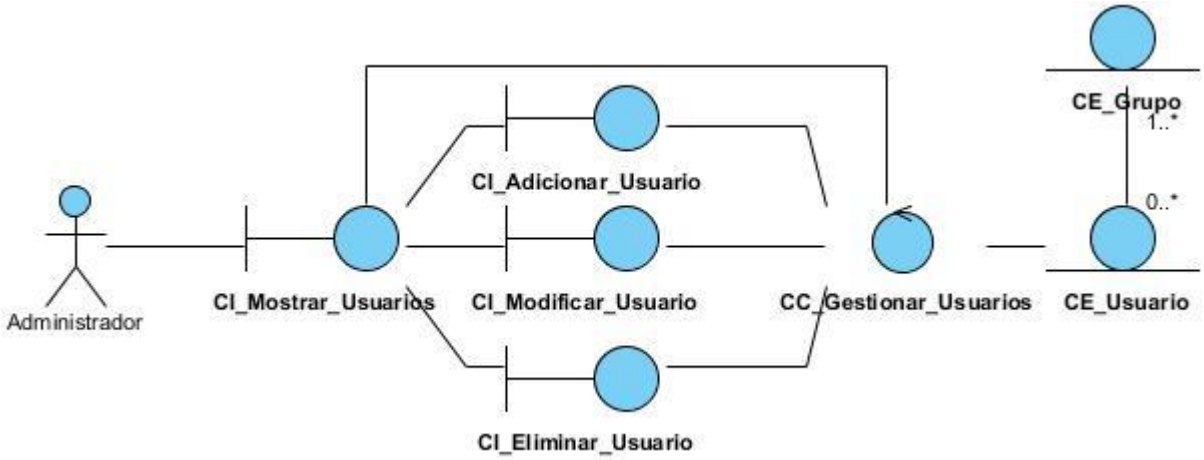
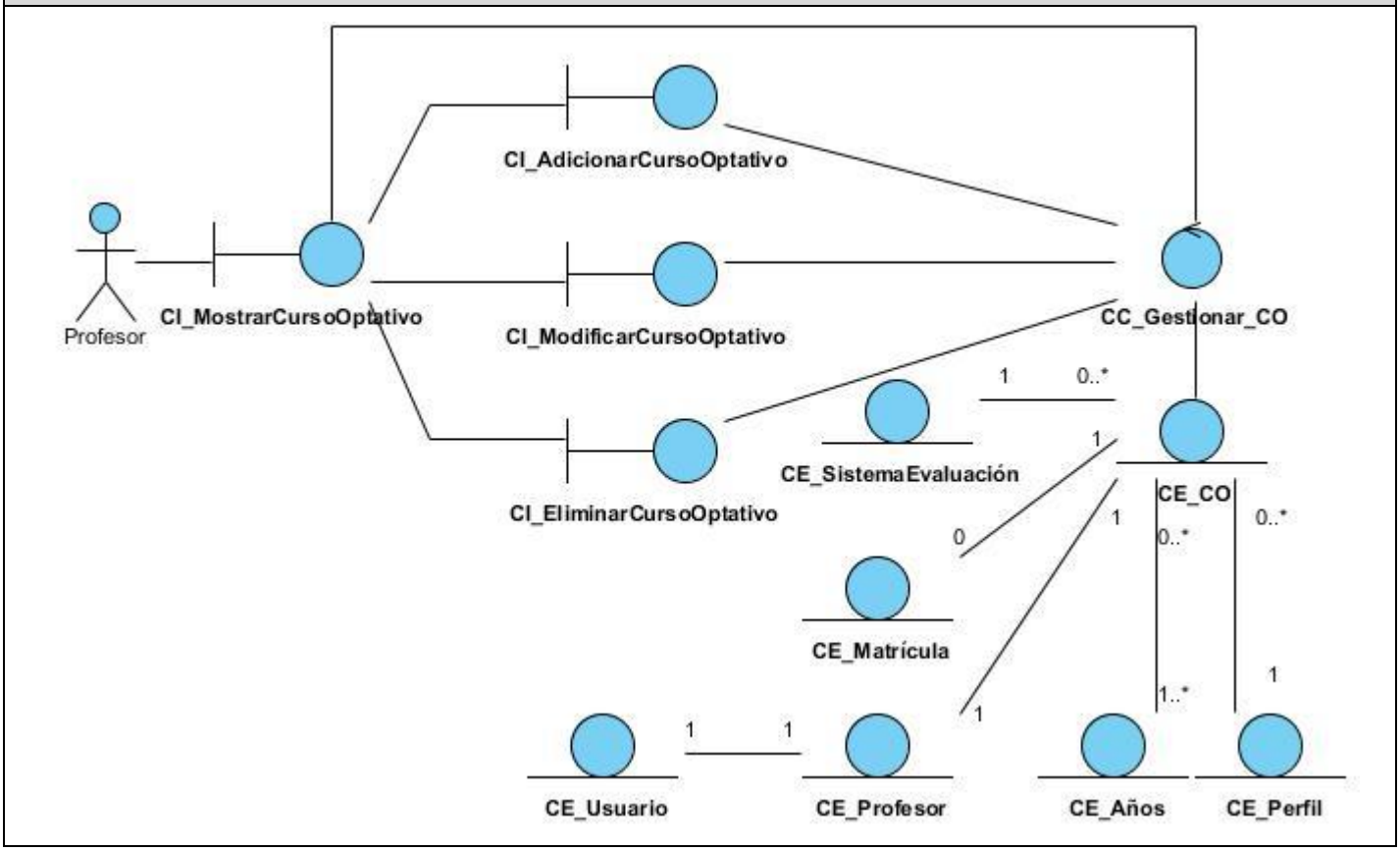


Figura 5. Diagrama de Clases del Análisis. CU Gestionar Curso Optativo.



3.3 Diseño.

El modelo de diseño es un modelo de objetos que describe la realización de los casos de uso y se utiliza como abstracción del modelo de implementación y el código fuente.[25] Es usado como una entrada inicial en las actividades de implementación y prueba.

El diseño es el centro de atención al final de la fase de elaboración y el comienzo de las iteraciones de construcción. Esto contribuye a una arquitectura estable y sólida, así como permite crear un plano del modelo de implementación.

[25] Jacobson, Ivar, Booch, Grady y Rumbaugh, James. *El proceso Unificado de Desarrollo de Software.* 1era edición. Madrid : Pearson Educación, S.A. - Addison Wesley, 2000. pág. 208. 84-7829-036-2.

En el diseño se modela el sistema y se encuentra su forma (incluida la arquitectura) para que soporte todos los requisitos, incluyendo los no funcionales y las restricciones que se le imponen.

Uno de los elementos bases del proceso de desarrollo de software es diseñar la Arquitectura de Software. Esencialmente sobre ella se sustentan todos los mecanismos de diseño y representaciones de la estructura general de la aplicación a desarrollar. La correcta definición del estilo arquitectónico a utilizar y los patrones es la raíz de lo anteriormente descrito.

3.3.1 Definición de la Arquitectura.

Una definición de arquitectura del software es:

"C*onjunto de decisiones significativas sobre la organización del sistema de software, la selección de los elementos estructurales y sus interfaces, con los que se compone el sistema, junto con su comportamiento tal como se especifica en las colaboraciones entre esos elementos, la composición de esos elementos estructurales y de comportamiento en subsistemas progresivamente más amplios y el estilo de arquitectura que guía esta organización -estos elementos y sus interfaces, sus colaboraciones y su composición*"[26].

Una idea general de la definición anterior es que para definir una arquitectura se necesita lo siguiente:

- Patrones de Diseño.
- Principios de Diseño (Patrones y estilos arquitectónicos).
- Diagramas de Clases de Diseño.
- Diagramas de Interacción.

3.3.1.1 Patrones de Diseño.

En el diseño de objetos es extremadamente importante la asignación habilidosa de responsabilidades. La decisión acerca de la asignación de responsabilidades, a menudo, tiene lugar durante la creación de los diagramas de interacción y con seguridad durante la programación. Los patrones codifican buenos consejos y principios relacionados con frecuencia con la asignación de responsabilidades.

[26] Rumbaugh, James, Jacobson, Ivar y Booch, Grady. *El lenguaje unificado de modelado. Manual de referencia.* s.l. : Addison Wesley, 2000. pág. 130.

Patrones GOF (Gang Of Four o Pandilla de los Cuatro). Son un conjunto de 23 patrones de diseño muy útiles durante el diseño de objetos. Estos se clasifican según el propósito para el que han sido definidos como: Patrones creacionales, estructurales y de comportamiento.

Dentro del grupo de Patrones estructurales se encuentra los Patrones Decorador y Fachada, dos de los cuales fueron utilizados en el desarrollo de la aplicación por lo que se describen a continuación:

Patrón Decorador.

El patrón Decorador responde a la necesidad de añadir dinámicamente funcionalidad a un Objeto. Esto nos permite no tener que crear sucesivas clases que hereden de la primera incorporando la nueva funcionalidad, sino otras que la implementan y se asocian a la primera.

La capa de la vista también puede aprovechar la separación de código, las páginas web suelen contener elementos que se muestran de forma idéntica a lo largo de toda la aplicación, por este motivo en Symfony la vista se separa en un layout y en una plantilla. Normalmente, el layout es global en toda la aplicación o al menos en un grupo de páginas, el contenido de la plantilla se integra en el layout, o si se mira desde el otro punto de vista, el layout *decora* la plantilla. Este comportamiento es una implementación del patrón de diseño llamado “decorador”.

Patrón Fachada.

Una fachada se trata de un objeto que ofrezca una sencilla interfaz que ocultará uno o varios sistemas más complejos y sus interacciones. Con este patrón se ofrece un acceso sencillo y se desacopla al sistema. Típicamente se utiliza en librerías o en sistemas diseñados en capas.

Durante el desarrollo del sistema se implementó una clase Fachada encargada de interconectar la capa de abstracción de la base de datos y la capa del modelo físico de datos. De esta forma se crearon las diferentes funcionalidades encargadas de obtener los datos solicitados por las clases controladoras y que serían mostrados luego en las diferentes vistas de usuarios.

Patrones GRASP (General Responsibility Assignment Software Patterns o Patrones Generales de Software para Asignar Responsabilidades). Describen los principios fundamentales del diseño de objetos y la asignación de responsabilidades, expresados como patrones.

Experto en Información.[27]

Problema: *¿Cuál es un principio general para asignar responsabilidades a los objetos?*

Un Modelo de Diseño podría definir cientos o miles de clases y una aplicación podría requerir que se realicen cientos o miles de responsabilidades. Durante el diseño de objetos, cuando se definen las interacciones entre los objetos, se toman decisiones sobre la asignación de responsabilidades a las clases. Si se hace bien, los sistemas tienden a ser más fáciles de entender, mantener y ampliar y existen más oportunidades para reutilizar componentes en futuras aplicaciones.

Solución: Asignar una responsabilidad al experto en información -*la clase que tiene la información necesaria para realizar la responsabilidad*.

Beneficios de su uso: Se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Esto favorece a mantener un bajo acoplamiento entre las clases, 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.

Utilización en Symfony: Este es uno de los más utilizados en el desarrollo de la aplicación, por ejemplo Propel es la librería externa que utiliza Symfony como ORM, encapsula toda la lógica de los datos y son generadas las clases con todas las funcionalidades comunes de las entidades.

Creador.[28]

Problema: *¿Quién debería ser el responsable de la creación de una nueva instancia de alguna clase?*

[27] Larman, Craig. *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* 2da. México : Pearson Education, 2003. pág. 164-168. 8420534382.
[28] Ibídem – pág. 168-170.

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

Solución: Asignar a la clase B la responsabilidad de crear una instancia de clase A si se cumple uno o más de los casos siguientes:

- B agrega objetos de A.
- B contiene objetos de A.
- B registra instancias de objetos de A.
- B utiliza más estrechamente objetos de A.
- B tiene los datos de inicialización que se pasarán a un objeto de A cuando sea creado (por tanto, B es un Experto con respecto a la creación de A).
- B es un creador de los objetos A.

Beneficios de su uso:

- Provee el bajo acoplamiento entre las clases (descrito a continuación), lo que implica menos dependencias de mantenimiento y mayores oportunidades para reutilizar. Probablemente no se incrementa el acoplamiento porque la clase creada es presumible que ya sea visible a la clase creadora, debido a las asociaciones existentes que motivaron su elección como creador.

Utilización en Symfony: En las clases Controladoras (actions.php) se encuentran las acciones definidas para el sistema y se ejecutan en cada una de ellas. En dichas acciones se crean los objetos de las clases que representan las entidades, lo que evidencia que las clases Actions son “creadoras” de dichas entidades.

Alta Cohesión.[29]

Problema: *¿Cómo mantener la complejidad manejable?*

En cuanto al diseño de objetos, la cohesión (o de manera más específica, la cohesión funcional) es una medida de la fuerza con la que se relacionan y del grado de focalización de las responsabilidades de un

[29] Ibídem – pág. 170-173.

elemento. Un elemento con responsabilidades altamente relacionadas y que no hace una gran cantidad de trabajo, tiene alta cohesión. Estos elementos pueden ser clases, subsistemas, etc.

Solución: Asignar una responsabilidad de manera que la cohesión permanezca alta.

Beneficios de su uso:

- Se incrementa la claridad y facilita la comprensión del diseño.
- Se simplifican el mantenimiento y las mejoras.
- Se soporta a menudo bajo acoplamiento.
- El grano fino de funcionalidad altamente relacionada incrementa la reutilización porque una clase cohesiva se puede utilizar para un propósito muy específico.

Utilización en Symfony: Symfony permite la organización del trabajo en cuanto a la estructura del proyecto y la asignación de responsabilidades con una alta cohesión. Un ejemplo de ello es la clase Actions, la cual está formada por varias funcionalidades que están estrechamente relacionadas, siendo esta la responsable de definir las acciones para las plantillas y colaborar con otras para realizar diferentes operaciones, instanciar objetos y acceder a las propiedades.

Bajo Acoplamiento.[30]

Problema. *¿Cómo soportar bajas dependencias, bajo impacto del cambio e incremento de la reutilización?*

El acoplamiento es una medida de la fuerza con que un elemento *está conectado a*, *tiene conocimiento de*, *confía en*, otros elementos. Un elemento con bajo (o débil) acoplamiento no depende de demasiados otros elementos. Estos elementos pueden ser clases, subsistemas, sistemas, etcétera.

Una clase con alto (o fuerte) acoplamiento confía en muchas otras clases. Tales clases podrían no ser deseables; algunas adolecen de los siguientes problemas:

- Los cambios en las clases relacionadas fuerzan cambios locales.
- Son difíciles de entender de manera aislada.
- Son difíciles de reutilizar puesto que su uso requiere la presencia adicional de las clases de las que depende.

[30] Ibídem – pág. 173-177.

Solución: Asignar una responsabilidad de manera que el acoplamiento permanezca bajo.

Beneficios de su uso:

- No afectan los cambios en otros componentes.
- Fácil de entender de manera aislada.
- Conveniente para reutilizar.

Utilización en Symfony: La clase Actions hereda únicamente de sfActions para alcanzar un bajo acoplamiento de clases. Las clases que implementan la lógica del negocio y de acceso a datos se encuentran en el modelo, las cuales no tienen asociaciones con las de la vista o el controlador, lo que proporciona que la dependencia en este caso sea baja.

Controlador.[31]

Problema: *¿Quién debe ser el responsable de gestionar un evento de entrada al sistema?*

Un evento del sistema de entrada es un evento generado por un actor externo. Se asocian con operaciones del sistema como respuesta a los eventos del sistema, tal como se relacionan los mensajes y los métodos.

Solución: Asignar la responsabilidad de recibir o manejar un mensaje de evento del sistema a una clase que representa una de las siguientes opciones:

- Representa el sistema global, dispositivo o subsistema *(controlador de fachada).*
- Representa un escenario de caso de uso en el que tiene lugar el evento del sistema, a menudo denominado <NombreDelCasoDeUso>Manejador, <NombreDelCasoDeUso>Coordinador o <NombreDelCasoDeUso>Sesion *(controlador de sesión o de caso de uso).*

Informalmente, una sesión es una instancia de una conversación con un actor. Las sesiones pueden tener cualquier duración, pero se organizan a menudo en función de los casos de uso (sesiones de casos de uso).

[31] Ibídem – pág. 177-185.

Beneficios de su uso:

- El Controlador recibe la solicitud del servicio desde la capa de interfaz de usuario (UI, del inglés User Interface) y coordina su realización, normalmente delegando a otros objetos.
- *Aumenta el potencial para reutilizar y* las *interfaces conectables.* Asegura que la lógica de la aplicación no se maneja en la capa de interfaz. Técnicamente, las responsabilidades de un controlador podrían manejarse en un objeto interfaz, pero la implicación de tal diseño es que el código del programa y la lógica relacionada con la realización de la lógica de la aplicación estaría embebida en los objetos ventana o interfaz. Un diseño de una interfaz como controlador reduce la oportunidad de reutilizar la lógica en futuras aplicaciones, puesto que está ligada a una interfaz particular (por ejemplo, objetos ventana) que es raramente aplicable en otras aplicaciones. En cambio, delegando la responsabilidad de una operación del sistema a un controlador ayuda a la reutilización de la lógica en futuras aplicaciones. Y puesto que la lógica no está ligada a la capa de interfaz, puede sustituirse por una interfaz nueva.
- *Razonamiento sobre el estado de los casos de uso.* A veces es necesario asegurar que las operaciones del sistema tienen lugar en una secuencia válida, o ser capaces de razonar sobre el estado actual de la actividad y operaciones del caso de uso que está en marcha.

Utilización en Symfony: Un ejemplo del patrón Controlador es fácil de encontrar en Symfony, este se puede ver desde la clase sfFrontController, sfFrontWebController, sfContex, los “actions”, el index.php del ambiente, etc. La arquitectura del framework (MVC) ayuda desde el principio pues existe una capa específicamente para los controladores, que son el núcleo del mismo. Symfony aplica el patrón “Front Controller” (Controlador frontal) y por tanto tiene una estructura bien organizada de controladores, que parte desde el “index.php” del ambiente y terminan en los “actions”. Cada clase en esta capa tiene su responsabilidad y es única, hay controladores que se encargan de la seguridad del sistema trabajando con ficheros YML, otros sólo se encargan de identificar mediante datos las clases que deben realizar determinadas tareas.

3.3.1.2 Patrones y Estilos Arquitectónicos.

Patrón Modelo Vista Controlador (MVC).

Es un estilo basado en un patrón de diseño que plantea la separación de diferentes clases en dependencia de la función que realizan de modo tal que sea posible manejar dinámicamente la forma en que se procesan solicitudes y se gestiona la manera en que se muestran resultados al usuario final. En otras palabras separa la presentación del dominio de la aplicación.

Es un principio que utilizan muchos frameworks para basar su funcionamiento, la idea de "*Don't call us, we'll call you*" (No nos llame, nosotros lo llamaremos a usted). Esa idea ha hecho que los frameworks que implementan MVC se puedan usar sencillamente implementando interfaces o extendiendo de una clase abstracta que brinda el framework.

Symfony está basado en un patrón clásico del diseño web conocido como arquitectura MVC[32], que está formado por 3 niveles:

- El modelo representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.
- La vista transforma el modelo en una página web que permite al usuario interactuar con ella.
- El controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones. El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes de, por ejemplo, el tipo de gestor de bases de datos utilizado por la aplicación. El uso de un framework que utiliza MVC obliga a dividir y organizar el código de acuerdo a las convenciones establecidas por el mismo. El código de la presentación se guarda en la vista, el código de manipulación de datos se guarda en el modelo y la lógica de procesamiento de las peticiones constituye el controlador.

[32] Potencier, Fabien, Zaninotto, François. *Symfony la guía definitiva.* [PDF]. www.librosweb.es. pág 26.

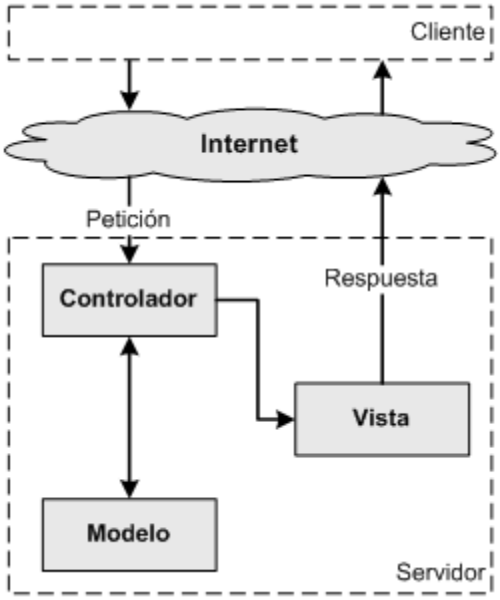


Figura 6. Representación del Patrón MVC en Symfony[33]**.**

La capa del modelo se puede dividir en la capa de acceso a los datos y en la capa de abstracción de la base de datos, para lograr que las funciones que acceden a los datos no utilicen sentencias ni consultas que dependan de una base de datos, sino que utilicen otras funciones; para realizar las consultas emplea los patrones Active Record y Active Table:

Active Record: Representa de forma *Orientada a Objetos* los datos de una Base de Datos Relacional - modelo conocido también como **ORM** o "Object-Relational Mapping" - , definiendo interfaces sencillas para acceder y manipular esos datos.

Active Table: El Active Table es un objeto que puede persistir otros objetos en medios no volátiles, como bases de datos. La función de una clase que implementa este patrón es de comunicarse con la base de datos y regresar objetos como los que tienen definidos para la aplicación.

Así, si se cambia de sistema gestor de bases de datos, solamente es necesario actualizar la capa de abstracción de la base de datos.

[33] Ibídem – pág. 27.

Patrón Front Controller.

Situándose dentro del patrón MVC, el "Front Controller" es el componente que recibe los requerimientos, los envía a los elementos encargados de procesar la lógica y luego lo envía nuevamente a la vista (esta vez incluyendo los datos obtenidos). Si no tuvieran un punto de acceso único, tendrían los siguientes problemas:

- Código duplicado.
- Manejo no centralizado de vistas.

Symfony no presenta este problema, el controlador normalmente se divide en un controlador frontal, que es único para cada aplicación, y las acciones, que incluyen el código específico del controlador de cada página. Una de las principales ventajas de utilizar un controlador frontal es que ofrece un punto de entrada único para toda la aplicación. Así, en caso de que sea necesario impedir el acceso a la aplicación, solamente es necesario editar el script correspondiente al controlador frontal.

En la siguiente figura se muestra el flujo de trabajo de Symfony:

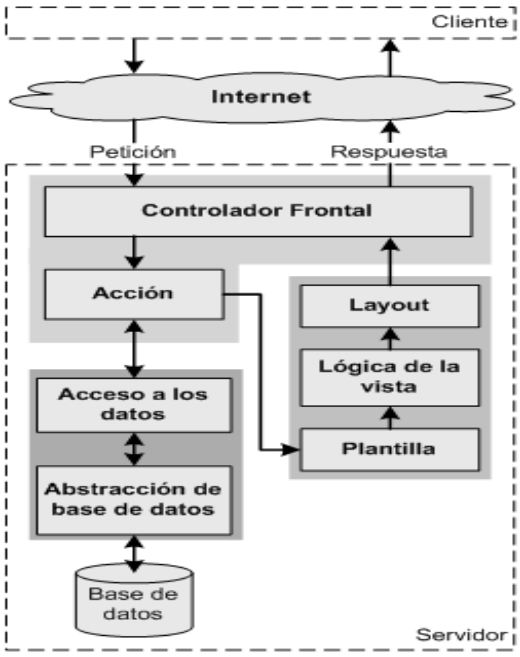


Figura 7. Flujo de trabajo de Symfony[34]**.**

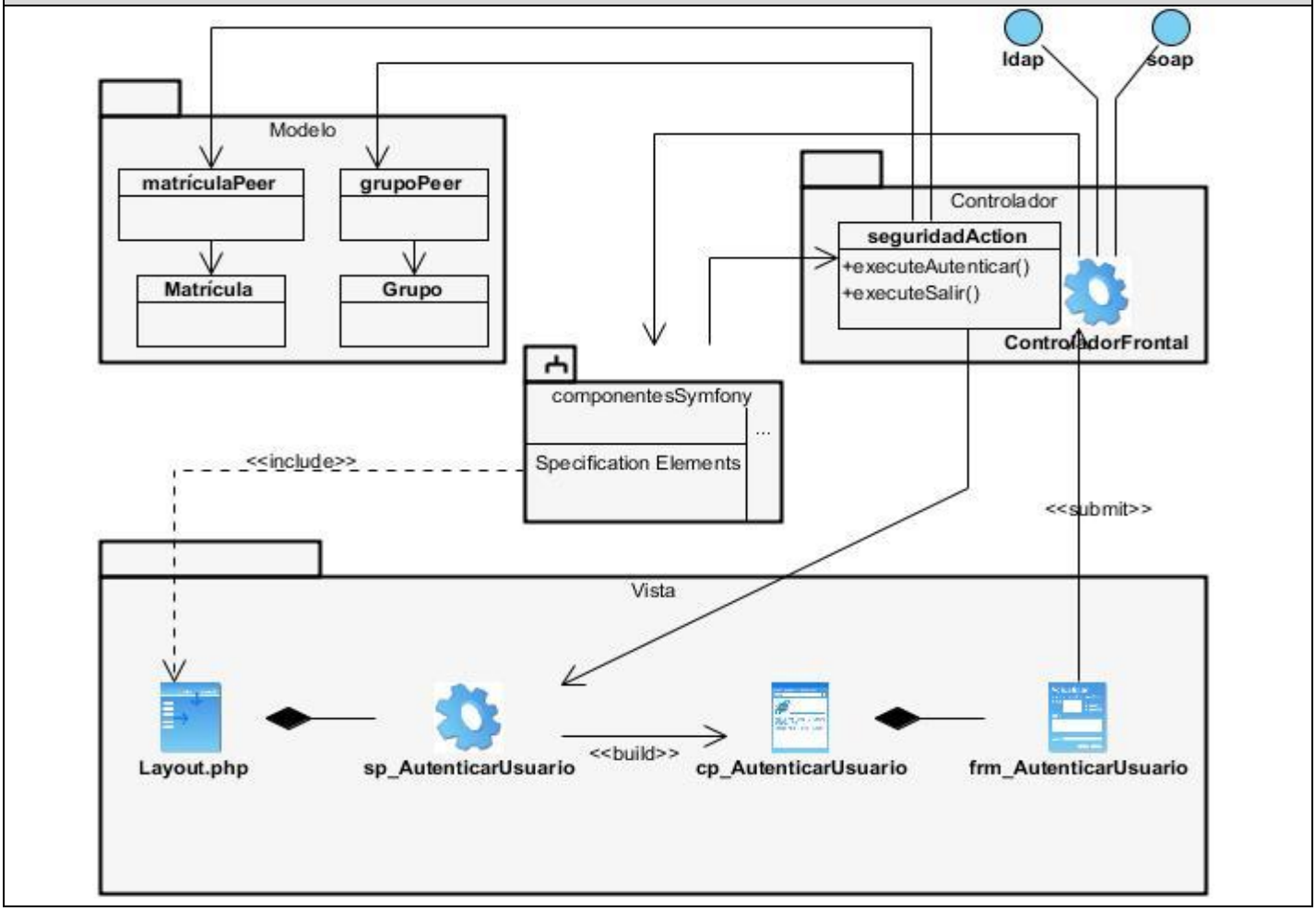
[34] Ibídem – pág. 39.

3.3.1.3 Diagramas de Clases del Diseño (DCD).

Los diagramas de clases son diagramas de estructura estática que muestran las clases del sistema y sus interrelaciones. *Los diagramas de clase son el pilar básico del modelado con UML, siendo utilizados tanto para mostrar lo que el sistema puede hacer (análisis), como para mostrar cómo puede ser construido (diseño). Los diagramas de clases son los más utilizados en el modelado de sistemas orientados a objetos*[35].

A continuación se modelan solamente los DCD de los CUS Críticos.

Figura 8. Diagrama de Clases del Diseño. CU Autenticar Usuario.



[35] Larman, Craig. *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* 2da. México : Pearson Education, 2003. pág. 224-226. 8420534382.

Figura 9. Diagrama de Clases del Diseño. CU Gestionar Usuario.

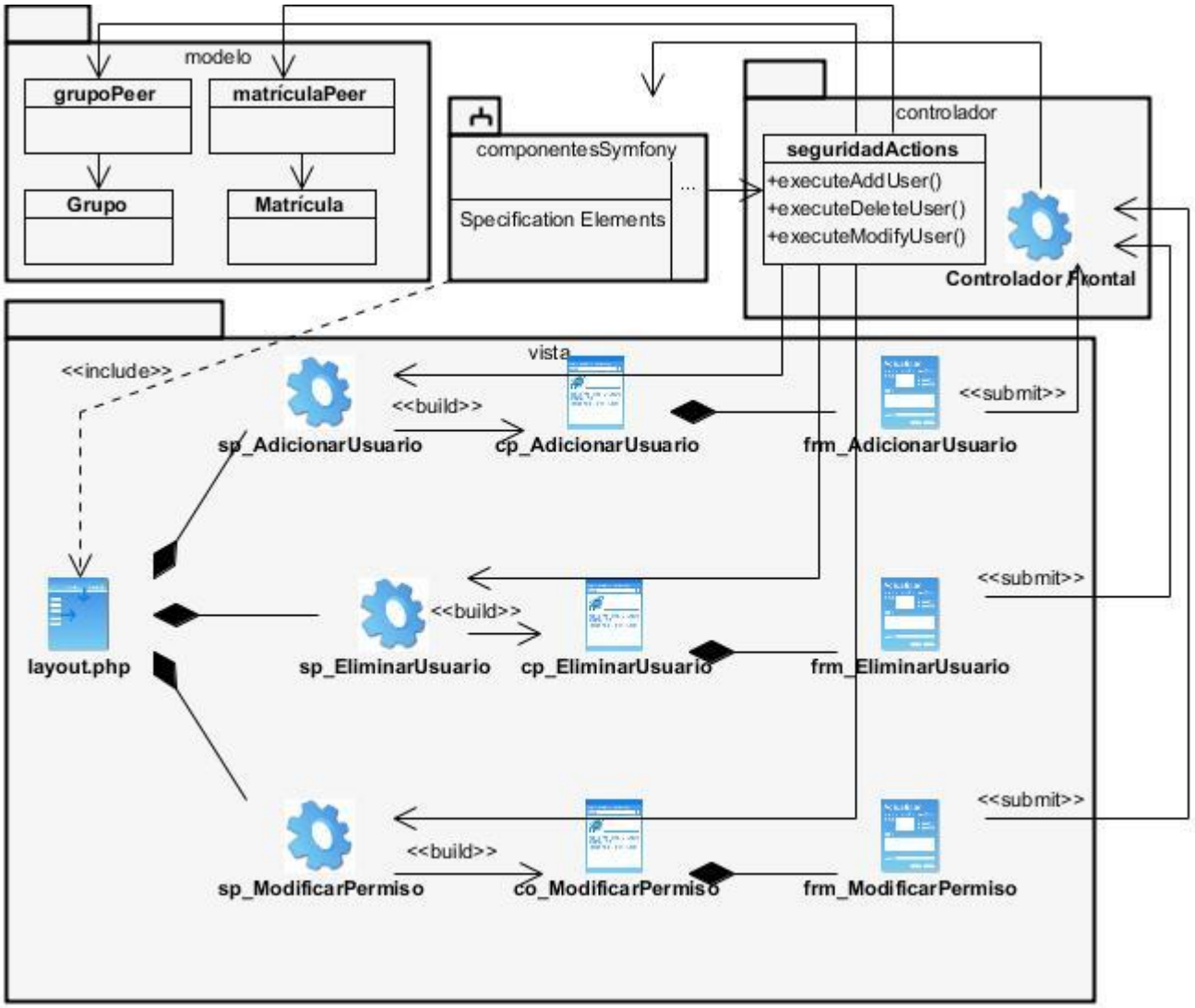
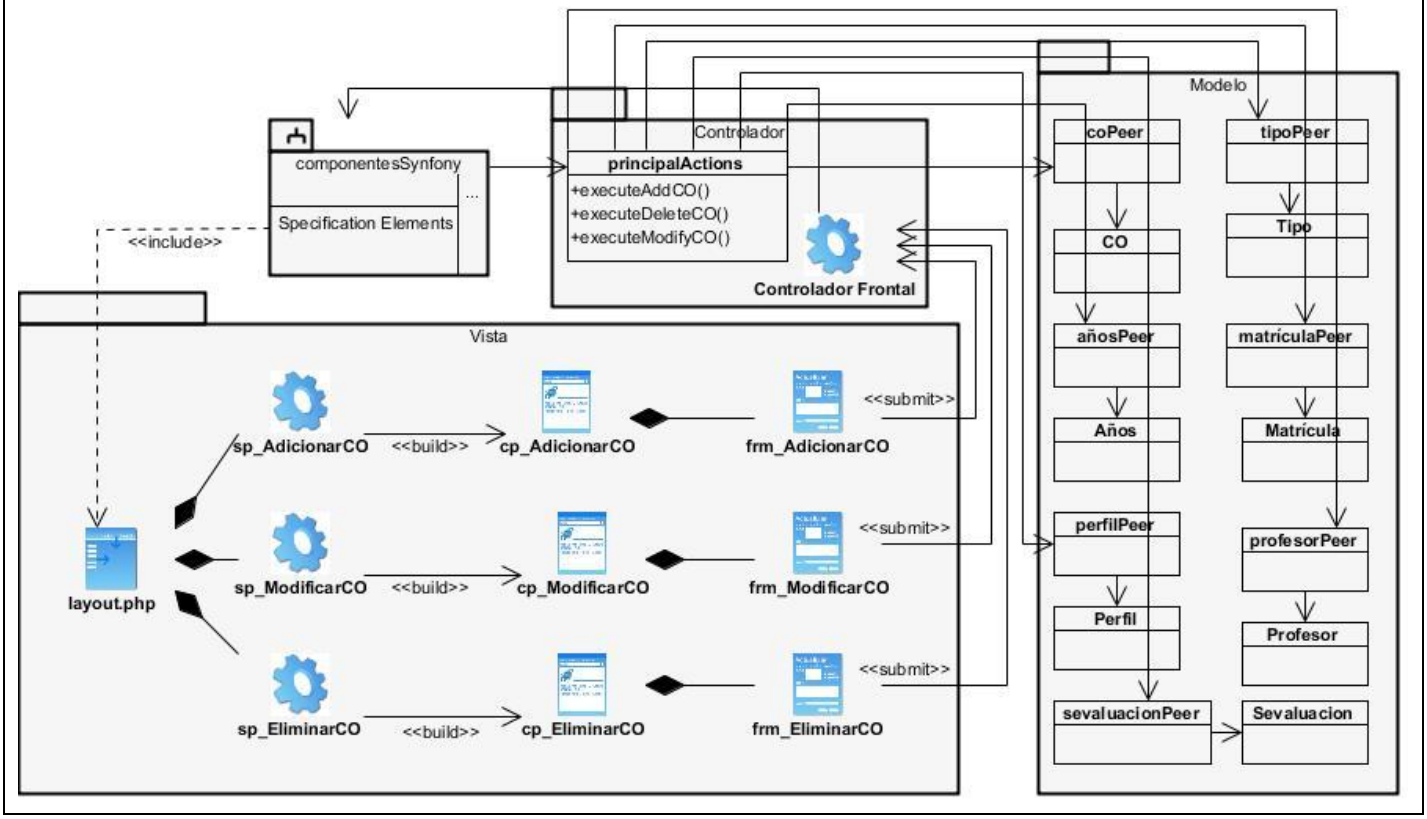


Figura 10. Diagrama de Clases del Diseño. CU Gestionar Curso Optativo.



3.3.1.4 Diagramas de Interacción.

Un diagrama de interacción muestra gráficamente cómo los objetos interactúan a través de mensajes para realizar las tareas.[36] No son sólo importantes para modelar los aspectos dinámicos de un sistema, sino también para construir sistemas ejecutables por medio de ingeniería directa e inversa. Existen dos tipos de diagramas de interacción, secuencia y colaboración.

3.3.2 Modelo de Datos.

El Modelo de Datos describe las representaciones lógicas y físicas de datos persistentes utilizados por una aplicación. En los casos en que la aplicación utilizará un sistema de gestión de bases de datos relacionales (RDBMS), el modelo de datos también incluye elementos de modelo para procedimientos

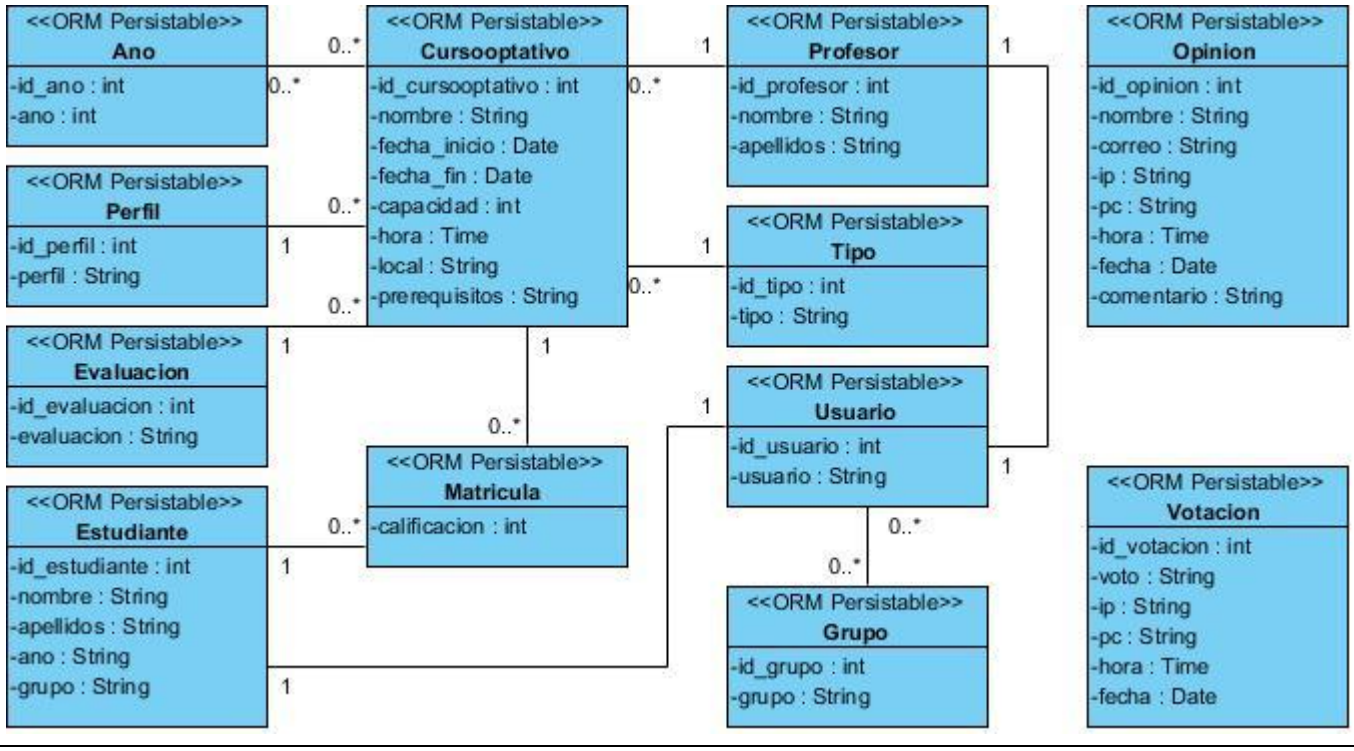
[36] Ibídem – pág. 148.

almacenados, desencadenantes, restricciones, etc. que definen la interacción de los componentes de la aplicación con RDBMS[37]*.*

3.3.2.1 Modelo Lógico de Datos.

Todas las clases identificadas en el dominio del diseño no son clases persistentes. La persistencia es la capacidad de un objeto de mantener su valor en el espacio y en el tiempo. El diagrama de clases persistentes tiene como entrada el diagrama de clases del diseño, de donde se seleccionan las clases que van a persistir y se confecciona el mismo. A continuación se muestra el diagrama de clases persistentes.

Figura 11. Diagrama de Clases Persistentes.

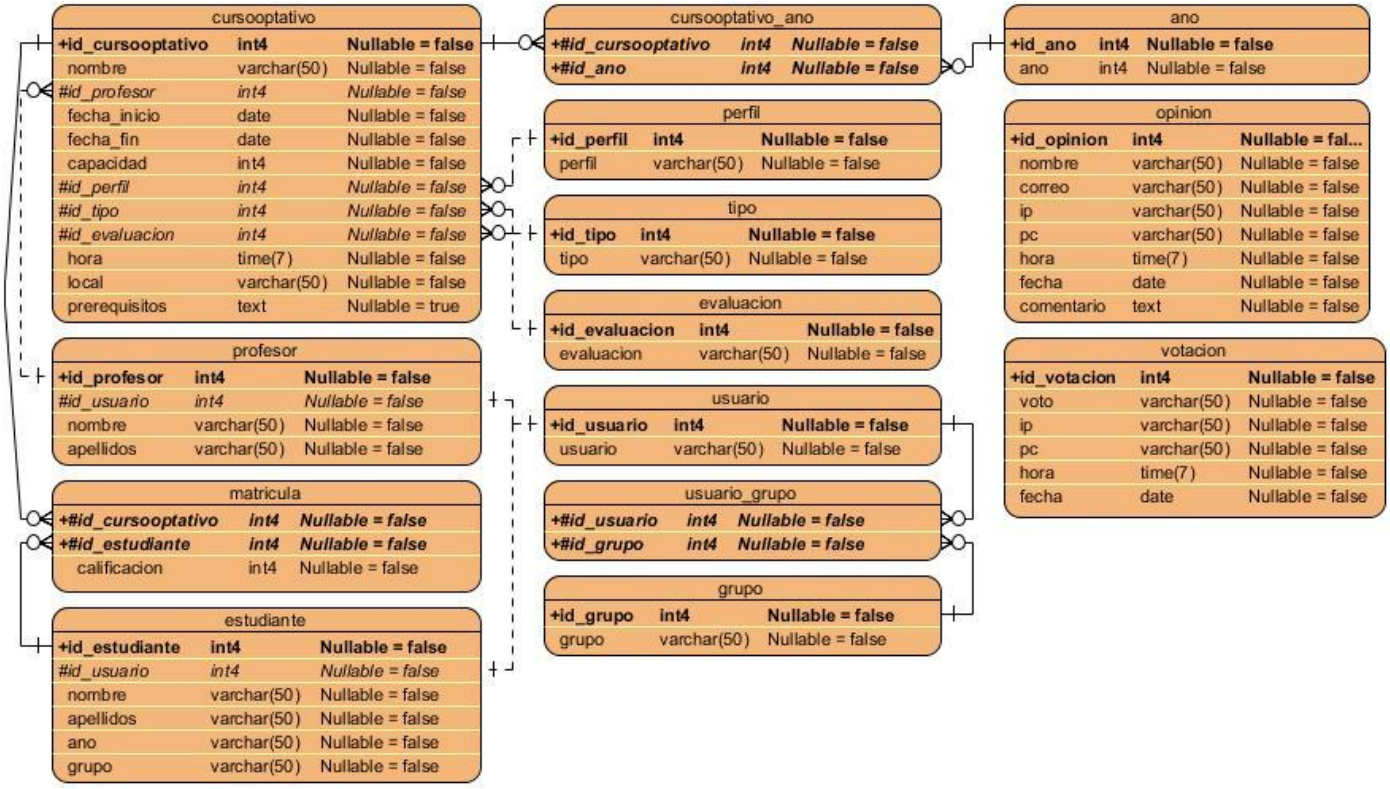


3.3.2.2 Modelo Físico de Datos.

El modelo físico de datos se desarrolló a partir de la base del conjunto de clases persistentes y sus asociaciones en el modelo de diseño, la herramienta utilizada para la realización de este modelo fue Visual Paradigm for UML 6.0 Enterprise Edition generándose posteriormente para PostgreSQL.

[37] *Ayuda en español del Rational Unified Process.* s.l. : IBM Corporation, 2006.

Figura 12. Diagrama Entidad Relación.



3.4 Conclusiones.

En el presente capítulo se realizó el análisis y diseño del sistema donde fueron relacionados los requisitos funcionales y los no funcionales. Se ha llevado a cabo la descripción de las clases y demás elementos necesarios para la implementación. Se obtuvieron los diagramas de clases del análisis y del diseño así como se definieron, a partir del mismo, cuáles son las clases que serán persistentes. A partir de esto, se construyó el modelo de datos.

CAPÍTULO 4

IMPLEMENTACIÓN Y PRUEBA

4.1 Introducción.

En este capítulo se presentará el modelo de implementación, donde se describe cómo los elementos del modelo de diseño, se implementan en términos de componentes, como ficheros de código fuente, ejecutables, etc. El modelo de implementación describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización, disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados y cómo dependen los componentes unos de otros.

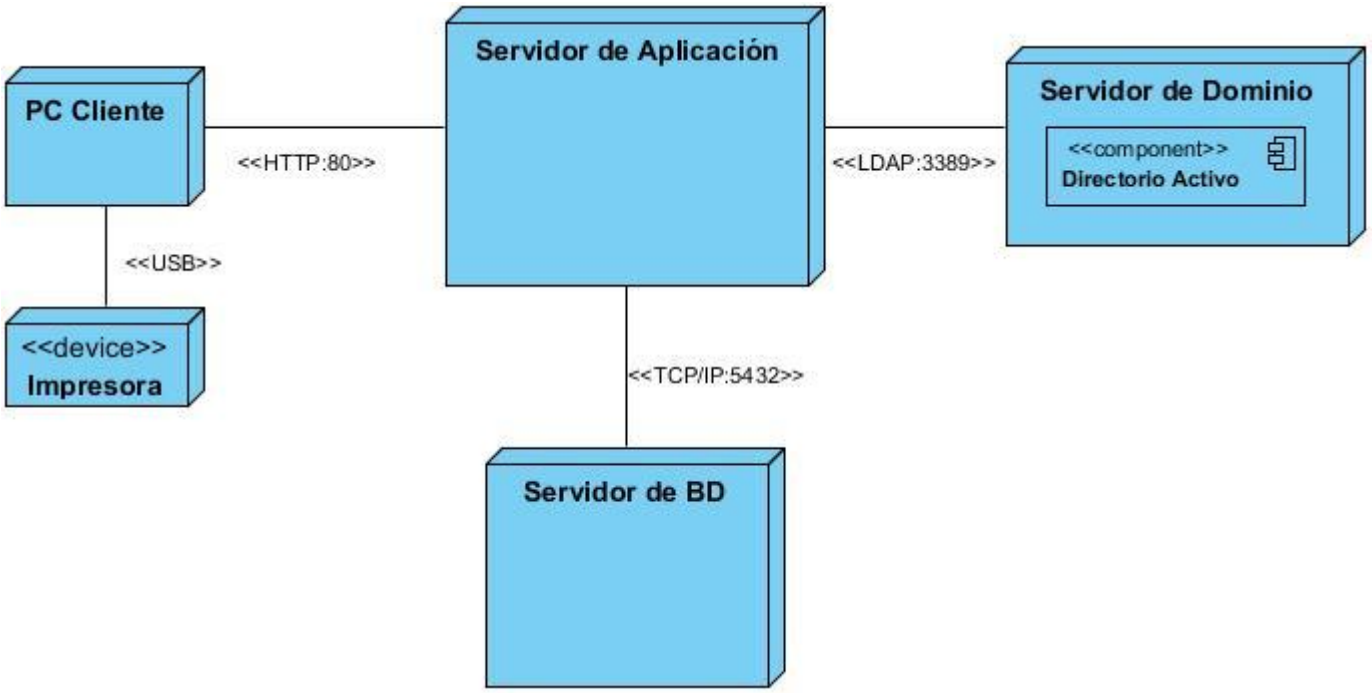
4.2 Despliegue.

4.2.1 Diagrama de Despliegue.

Un diagrama de despliegue es utilizado para representar la distribución física de un sistema. El mismo muestra la configuración de los nodos de procesamiento en tiempo de ejecución, los links de comunicación entre ellos y las instancias de los componentes y objetos que residen en ellos[38].

[38] Jacobson, Ivar, Booch, Grady y Rumbaugh, James. *El proceso Unificado de Desarrollo de Software.* 1era edición. Madrid : Pearson Educación, S.A. - Addison Wesley, 2000. pág. 217. 84-7829-036-2.

Figura 13. Diagrama de Despliegue.



4.2.2 Descripción de los Nodos del Diagrama de Despliegue.

Tabla 4: Descripción de los Nodos del Diagrama de Despliegue.

Recurso	Breve descripción de su uso.
Impresora	Impresora conectada a la estación cliente de la Secretaria Docente para realizar la impresión de las Actas de Evaluación Final de los Cursos Optativos.
PC Cliente.	Diferentes ordenadores que interactúan con la aplicación beneficiándose de los servicios que esta brinda.
Servidor de Aplicación.	Contiene los componentes necesarios para publicar la aplicación utilizando el Servidor Web Apache.
Servidor de Base de Datos.	Contiene publicada la Base de Datos correspondiente a la aplicación utilizando el Servidor de Base de Datos PostgreSQL.

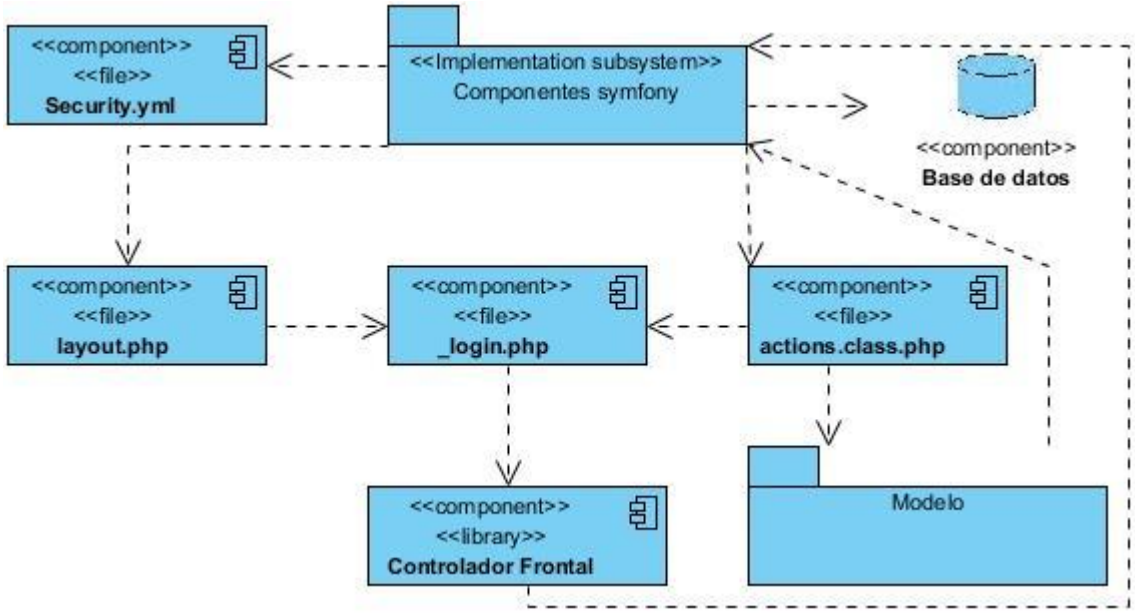
4.2.3 Diagramas de Componentes (DC).

Un componente es el empaquetamiento físico de los elementos de un modelo, como son las clases en el modelo de diseño.[39] Algunos estereotipos de componentes son los siguientes:

- <<executable>> Es un programa que puede ser ejecutado en un nodo.
- <<file>> Es un fichero que contiene código fuente o datos.
- <<library>> Es una librería estática o dinámica.
- <<table>> Es una tabla de la base de datos.
- <<document>> Es un documento.

A continuación se modelan solamente los DC de los CUS Críticos.

Figura 14. Diagrama de Componentes. CU Autenticar Usuario.



[39] *The Unified Modeling Language for Object-Oriented Development.* s.l. : Rational Software Corp, 1997. pág. 228.

Figura 15. Diagrama de Componentes. CU Gestionar Usuario.

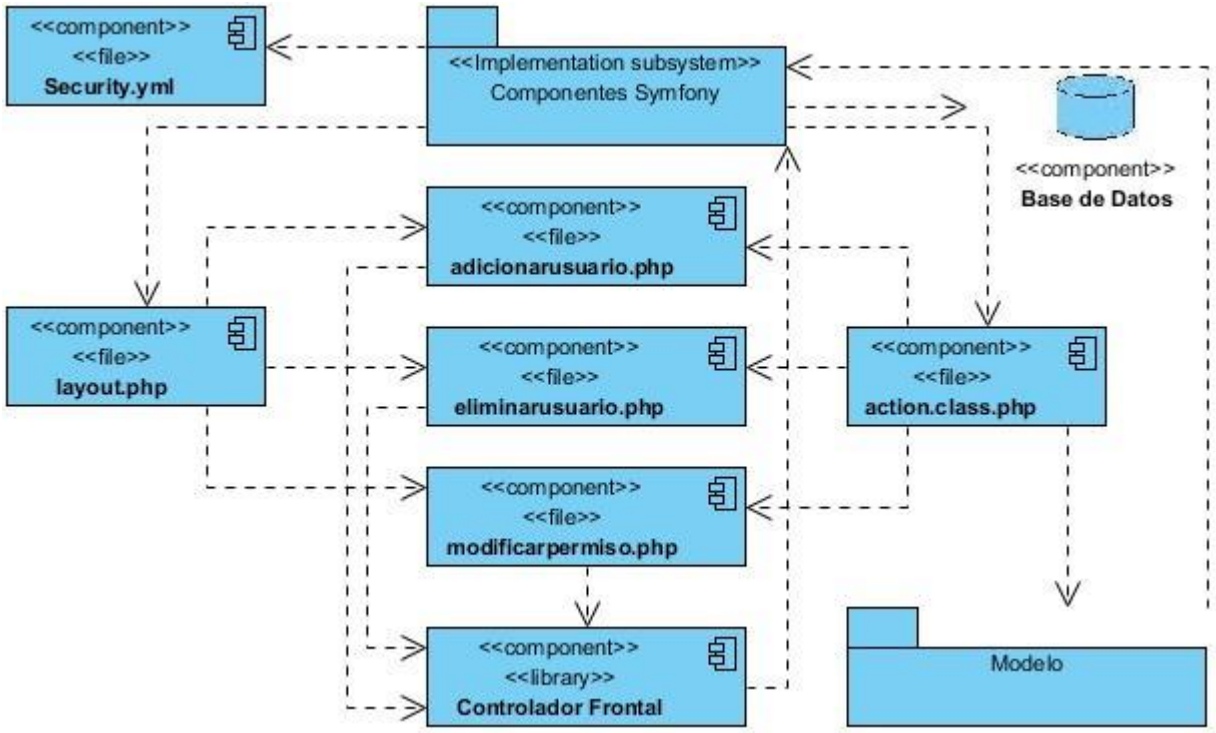
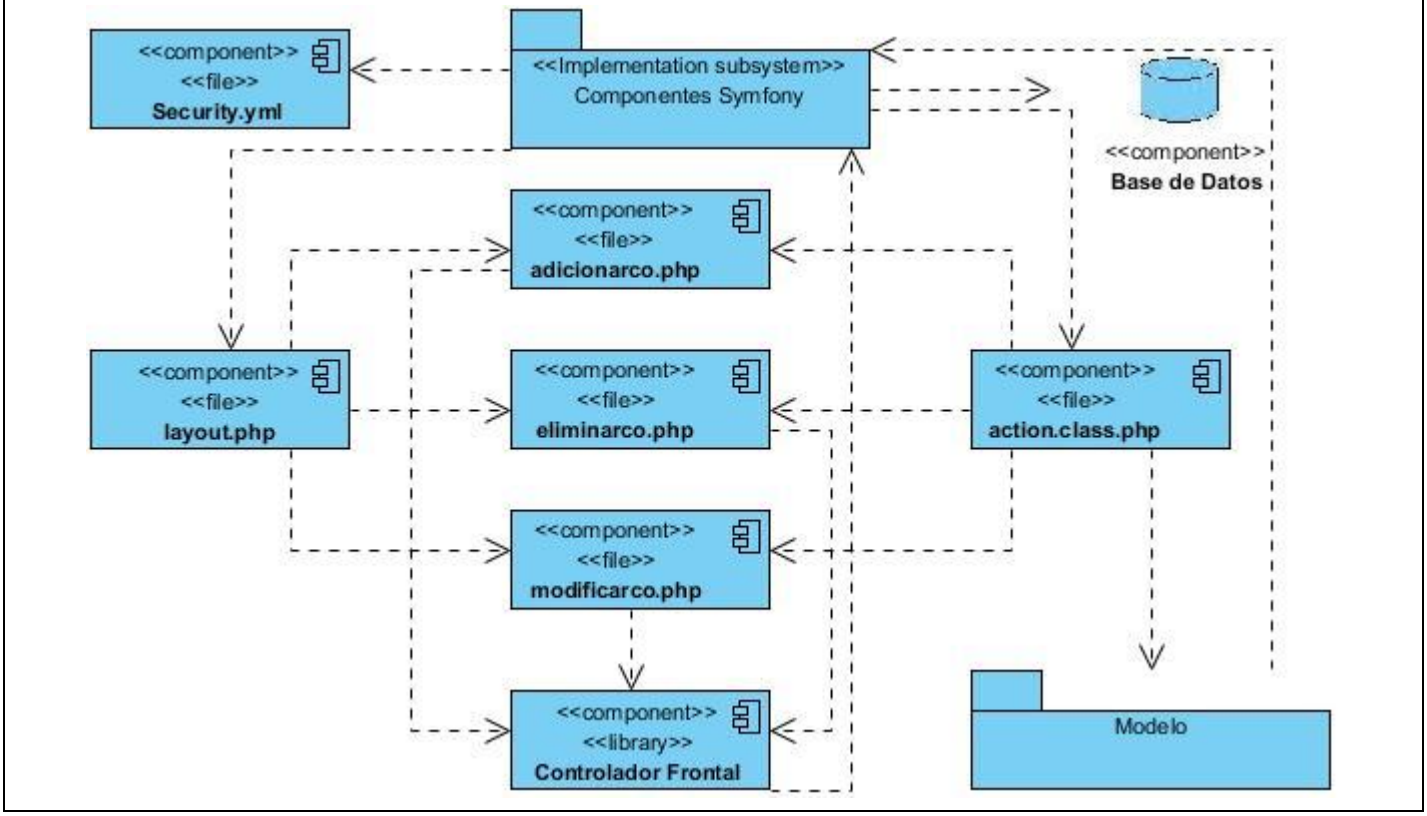


Figura 16. Diagrama de Componentes. CU Gestionar Curso Optativo.



4.3 Modelo de Prueba.

La prueba del software es un conjunto de actividades que se lleva a cabo sistemáticamente, que puede planificarse por adelantado y ejecutar una vez construido el código para la revisión final de las especificaciones, del diseño y de la codificación del software[40]. Cualquier producto de ingeniería puede ser probado mediante una de estas técnicas:

1. Conociendo la funcionalidad específica para la cual fue diseñado el producto, se pueden llevar a cabo pruebas que demuestren que cada función es completamente operativa. **Pruebas de caja negra.**

[40] Jacobson, Ivar, Booch, Grady y Rumbaugh, James. *El proceso Unificado de Desarrollo de Software.* 1era edición. Madrid : Pearson Educación, S.A. - Addison Wesley, 2000. 84-7829-036-2. pág. 281.

2. Conociendo el funcionamiento del producto se pueden desarrollar pruebas que aseguren que “todas las piezas encajen”, o sea, que la operación interna se ajusta a las especificaciones y que todos los componentes internos se han comprobado de forma adecuada. **Pruebas de caja blanca.**

Para el sistema desarrollado se le realizaron Pruebas de Caja Negra, la cual se centra fundamentalmente en los *requerimientos funcionales* del software.

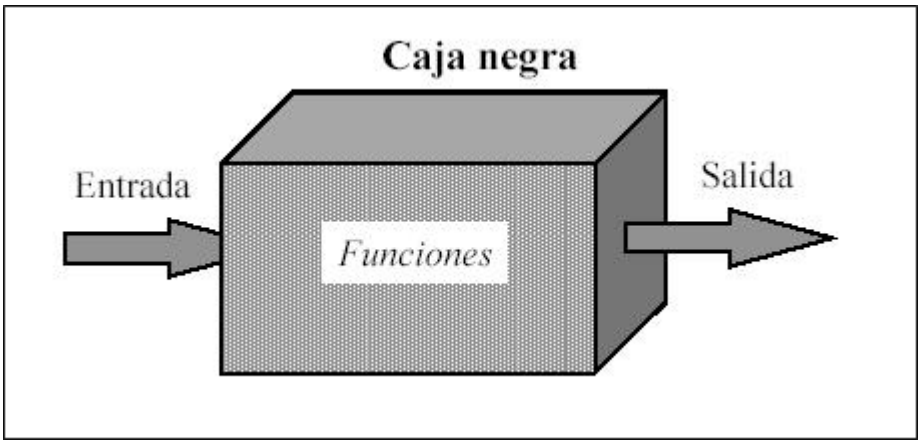


Figura 17. Representación de la Prueba de Caja Negra.

Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos.

A continuación se realizan los Casos de Prueba a los CUS Críticos.

4.3.1 Casos de Prueba.

Tabla 5: Caso de Prueba. Autenticar Usuario.

Condiciones de Entrada	Casos Válidos	Casos No Válidos
El usuario debe introducir los datos de usuario y contraseña.	El usuario introduce los datos (usuario y contraseña) y son datos válidos de algún usuario del dominio UCI.	El usuario no introduce alguno de los datos o que los datos introducidos no corresponden a algún usuario válido en el dominio UCI.

Caso de Uso: Autenticar Usuario.
Caso de Prueba: Autenticar Usuario.
Entrada: Usuario: fmperez. Contraseña: **********.
Resultado: El sistema autentica el usuario.
Condiciones: Ambos campos deben estar llenos y corresponder a un usuario válido en el dominio UCI.

Caso de Uso: Autenticar Usuario.
Caso de Prueba: Autenticar Usuario.
Entrada: Usuario: fmperez. Contraseña:
Resultado: El sistema muestra un mensaje de Error.
Condiciones: Ambos campos deben estar llenos y corresponder a un usuario válido en el dominio UCI.

Caso de Uso: Autenticar Usuario.
Caso de Prueba: Autenticar Usuario.
Entrada: Usuario: asdfghj. Contraseña: **********.
Resultado: El sistema muestra un mensaje de Error.
Condiciones: El campo usuario debe corresponder a un usuario válido del dominio UCI.

Tabla 6: Caso de Prueba. CUS Gestionar Usuario. Escenario Adicionar Usuario.

Condiciones de Entrada	Casos Válidos	Casos No Válidos
El administrador debe introducir el usuario que desea adicionar.	Que el administrador introduzca un usuario válido.	Que el administrador no introduzca el campo usuario o introduzca un usuario no válido.
El administrador debe especificar el grupo al cual pertenecerá el usuario introducido.	Que el administrador establezca el grupo al cual pertenece el nuevo usuario.	Que el administrador no especifique ningún grupo para el nuevo usuario.

Caso de Uso: Gestionar Usuario.
Caso de Prueba: Adicionar Usuario.
Entrada: Usuario: jabatista. Grupo: Secretario.
Resultado: El sistema adiciona el usuario.
Condiciones: Ambos campos deben estar llenos y corresponder a un usuario válido en el dominio UCI.

Caso de Uso: Gestionar Usuario.
Caso de Prueba: Adicionar Usuario.
Entrada: Usuario: jabatista. Grupo: -- Seleccione --.
Resultado: El sistema muestra un mensaje de Error.
Condiciones: Debe especificar a el grupo al que pertenece el usuario

Caso de Uso: Gestionar Usuario.
Caso de Prueba: Adicionar Usuario.
Entrada: Usuario: asdfghj. Grupo: -- Administrador --.
Resultado: El sistema muestra un mensaje de Error.
Condiciones: El campo usuario debe corresponder a un usuario válido del dominio UCI.

Tabla 7: Caso de Prueba. CUS Gestionar Curso Optativo. Escenario Adicionar Curso Optativo.

Condiciones de Entrada	Casos Válidos	Casos No Válidos
El usuario debe introducir el nombre, capacidad, cantidad de horas, la hora y el local del curso optativo.	Que el usuario introduzca el nombre, capacidad (número), cantidad de horas (número), la hora y el local del curso optativo.	Que el usuario no introduzca alguno de los parámetros o que introduzca un valor no válido para alguno de estos.
El usuario debe especificar el/los año(s) para el/los cual(es) estará habilitado el curso optativo.	Que el usuario selecciones al menos un año para el cual estará disponible el curso optativo.	Que el usuario no especifique ningún año para el cual estará destinado el curso optativo.
El usuario debe seleccionar el profesor, el perfil, el tipo y el sistema de evaluación del curso optativo.	Que el usuario selecciones el profesor, el perfil, el tipo y el sistema de evaluación del curso optativo.	Que el usuario no selecciones alguno de estos campos.
El usuario debe especificar la fecha de inicio y de fin del curso optativo.	Que el usuario especifique que la fecha de inicio y de culminación del curso optativo.	Que el usuario no especifique alguna de las fechas o que alguna de ellas no sea válida.

Caso de Uso: Gestionar Curso Optativo.
Caso de Prueba: Adicionar Curso Optativo.
Entrada: Nombre: Tecnologías modernas de programación web. Profesor: ahdominguez. Fecha de Inicio: 29/06/2011. Fecha de Fin: 11/07/2011. Capacidad: 30. Años: 3ro, 4to, 5to. Perfil: TLM. Tipo: Básico. Sistema de Evaluación: Evaluación Final. Cantidad de Horas: 10. Hora: 8.00 am. Local: Laboratorio 204. Pre requisitos (opcional):
Resultado: El sistema adiciona el curso optativo.
Condiciones: Todos los datos para adicionar el curso optativo deben estar llenos y dentro de los rangos establecidos.

Caso de Uso: Gestionar Curso Optativo.
Caso de Prueba: Adicionar Curso Optativo.
Entrada: Nombre: Tecnologías modernas de programación web. Profesor: ahdominguez. Fecha de Inicio: 29/06/2011. Fecha de Fin: 27/06/2011. Capacidad: 30. Años: 3ro, 4to, 5to. Perfil: TLM. Tipo: Básico. Sistema de Evaluación: Evaluación Final. Cantidad de Horas: 10. Hora: Local: Laboratorio 204. Pre requisitos (opcional):
Resultado: El sistema muestra un mensaje de Error.
Condiciones: Los datos para adicionar el curso optativo no pueden estar vacíos y la fecha de culminación del curso optativo no puede ser inferior a la fecha de inicio.

4.4 Conclusiones.

En el presente capítulo se ha llevado a cabo la descripción de cómo las clases y los objetos se organizan en términos de componentes. Se explicó cómo está estructurada la aplicación físicamente, a través del diagrama de despliegue y los diagramas de componentes. En los procesos de pruebas, se encuentran el diseño de los Casos de Pruebas que demuestran hasta cierto punto como se debe probar el sistema. Con el diseño de los elementos principales de ambos flujos se cumplen los objetivos trazados para el desarrollo de este capítulo.

CAPÍTULO 5

FACTIBILIDAD DEL SISTEMA

5.1 Introducción.

Es importante evaluar la factibilidad de un proyecto antes de su elaboración, para conocer si es conveniente llevarlo a cabo. La viabilidad y el análisis de riesgo están relacionados de muchas maneras. Si el riesgo del proyecto es alto, la viabilidad de producir software de calidad se reduce. En el presente capítulo se hace un estudio de la factibilidad, los beneficios y el costo del sistema propuesto.

5.2 Método de estimación: Puntos por Casos de Uso.

Una vez determinados los casos de uso que guiarán el desarrollo del software, se puede predecir una estimación del tiempo de duración del proyecto mediante el análisis de Puntos de Casos de Uso.

La estimación mediante el análisis de Puntos de Casos de Uso se trata de un método propuesto inicialmente por Gustav Karner en el año 1993 y que *se utiliza para la estimación del tiempo de desarrollo de un proyecto mediante la asignación de "pesos" a un cierto número de factores que lo afectan*[41], para finalmente, contabilizar el tiempo total estimado para el proyecto a partir de esos factores.

Este método resulta muy efectivo para estimar el esfuerzo requerido en el desarrollo de los primeros casos de uso de un sistema, si se sigue una aproximación iterativa como el Proceso Unificado de Software. En este tipo de aproximación, los primeros casos de uso a desarrollar son los que ejercitan la mayor parte de la arquitectura del software y los que a su vez ayudan a mitigar los riesgos más significativos.

[41] Peralta, Mario. *Estimación del esfuerzo basada en casos de uso.* [PDF] Buenos Aires, Argentina : s.n., 2004. pág. 6-7.

A continuación se detallan los pasos a seguir para la realización de este método.

5.2.1 Cálculo de Puntos de Casos de Uso sin ajustar.

El cálculo de Puntos de casos de Uso sin ajustar se calcula mediante la siguiente ecuación:

UUCP = UAW + UUCW

Donde:

UUCP: Puntos de Casos de Uso sin ajustar

UAW: Factor de Peso de los Actores sin ajustar

UUCW: Factor de Peso de los Casos de Uso sin ajustar

Factor de Peso de los Actores sin ajustar (UAW).

Este valor se calcula mediante un análisis de la cantidad de actores presentes en el sistema y la complejidad de cada uno de ellos. Los criterios a tener en cuenta se detallan a continuación.

Tabla 8: Factor de Pesos de los Actores sin ajustar.

Tipo de Actor	Descripción	Factor de Peso
Simple	Otro sistema que interactúa con el sistema a desarrollar mediante una interfaz de programación.	1
Medio	Otro sistema que interactúa con el sistema a desarrollar mediante un protocolo o una interfaz basada en texto.	2
Complejo	Una persona que interactúa con el sistema mediante una interfaz gráfica.	3

En el sistema propuesto va a interactuar un actor, los usuarios, que constituye un actor complejo ya que es una persona que interactúa con el sistema mediante una interfaz gráfica, por lo tanto el factor de peso sería 3. Luego el factor de peso de los actores sin ajustar se calcula mediante la siguiente ecuación:

UAW = Σ Cantidad Actores * Factor de Peso.

UAW = 1 * 3.

UAW = 3.

Factor de Peso de los Casos de Uso sin ajustar (UUCW).

El valor del Factor de Peso de los Casos de Uso sin ajustar se calcula mediante un análisis de la cantidad de Casos de Uso presentes en el sistema y la complejidad de cada uno de ellos. La complejidad de los Casos de Uso se establece teniendo en cuenta la cantidad de transacciones efectuadas en el mismo, donde una transacción es una secuencia de actividades atómica, es decir, se efectúa la secuencia de actividades completa, o no se efectúa ninguna de las actividades de la secuencia. Los criterios se muestran en la siguiente tabla:

Tabla 9: Factor de Pesos de los Casos de Uso sin ajustar.

Tipo de Caso de Uso	Descripción	Factor de Peso
Simple	El caso de Uso contiene de 1 a 3 transacciones.	5
Medio	El caso de Uso contiene de 4 a 7 transacciones.	10
Complejo	El caso de Uso contiene más de 8 transacciones.	15

Para la realización de la aplicación se tienen en cuenta 8 casos de uso, los cuales están de 1 a 3 transacciones, por lo que todos los casos de uso tienen un Tipo “Simple” con un “Factor de Peso” igual a 5, entonces el factor de peso de los casos de uso sin ajustar sería:

UUCW = Σ Cantidad CU * Factor de Peso.
UUCW = 8 * 5.
UUCW = 40.

Por tanto los Puntos de Casos de Uso sin ajustar sería:

UUCP = UAW + **UUCW**.
UUCP = 3 + 40.
UUCP = 43.

5.2.2 Cálculo de Puntos de Casos de Uso ajustados.

Ya obtenidos los Puntos de casos de uso sin ajustar, se debe ajustar este valor mediante la siguiente ecuación:

UCP = UUCP * TCF * EF.

Donde:

UCP: Puntos de Casos de Uso ajustados.

UUCP: Puntos de Casos de Uso sin ajustar.

TCF: Factor de complejidad técnica.

EF: Factor de ambiente.

El Factor de Complejidad Técnica **(TCF)** se calcula mediante la cuantificación de un conjunto de factores que determinan la complejidad técnica del sistema. Cada factor se cuantifica en un valor desde 0 a 5, donde 0 sería un aporte irrelevante y 5 sería un aporte muy relevante.

Tabla 10: Factor de Complejidad Técnica.

Factor	Descripción	Peso (P)	Valor Asignado (VA)	P * VA
T1	Sistema distribuido.	2	0	0
T2	Objetivos de performance o tiempo de respuesta.	1	4	4
T3	Eficiencia del usuario final.	1	4	4
T4	Procesamiento interno complejo.	1	3	3
T5	El código debe ser reutilizable.	1	3	3
T6	Facilidad de instalación.	0.5	3	1.5
T7	Facilidad de uso.	0.5	2	1
T8	Portabilidad.	2	0	0
T9	Facilidad de cambio.	1	1	1
T10	Concurrencia.	1	0	0
T11	Incluye objetivos especiales de seguridad.	1	3	3
T12	Provee acceso directo a terceras partes.	1	0	0
T13	Se requieren facilidades especiales de entrenamiento a los usuarios.	1	1	1
TOTAL				**21.50**

TCF = 0.6 + 0.01 * Σ (Peso * Valor asignado).

TCF = 0.6 + 0.01 * 21.5.

TCF = 0.815 ≈ 0.82.

El Factor de Ambiente **(EF)** está relacionado con las habilidades y entrenamiento del grupo de desarrollo que realiza el sistema. Cada factor se cuantifica con un valor desde 0 a 5, donde 0 sería un aporte irrelevante y 5 sería un aporte muy relevante.

Tabla 11: Factor de Ambiente.

Factor	Descripción	Peso (P)	Valor Asignado (VA)	P * VA
E1	Familiaridad con el modelo del proyecto utilizado.	1.5	2	3
E2	Experiencia en la aplicación.	0.5	2	1
E3	Experiencia en orientación a objetos.	1	3	3
E4	Capacidad del analista líder.	0.5	1	0.5
E5	Motivación.	1	5	5
E6	Estabilidad de los requerimientos.	2	4	8
E7	Personal part - time.	-1	1	-1
E8	Dificultad del lenguaje de programación.	-1	2	-2
TOTAL				**17.5**

EF = 1.4 - 0.03 * Σ (Peso* Valor Asignado).

EF = 1.4 – 0.03 * 17.5.

EF = 0.875.

Calculando los puntos de Caso de Uso Ajustados quedaría:

UCP = UUCP * TCF * EF.

UCP = 43 * 0.82 * 0.875.

UCP = 30.85.

5.2.3 Cálculo del Esfuerzo.

Que está dado por la siguiente ecuación:

E = UCP * CF.

Donde:

E: Esfuerzo estimado en horas hombres.

UCP: Punto de casos de usos ajustados.

CF: Factor de conversión.

Para calcular el **Factor de conversión (CF)** se tiene en cuenta:

Se contabilizan cuántos factores de los que afectan al Factor de Ambiente están por debajo del valor medio (3), para los factores de E1 a E6.
Se contabilizan cuántos factores de los que afectan al Factor de Ambiente están por encima del valor medio (3), para los factores E7 y E8.
Como el total contabilizado es 2 se utiliza el **factor de conversión 20 horas-hombre/Punto de Casos de Uso.**
E = UCP * CF.
E = 30.85 ***** 20.
E = 617 Horas-Hombre.

5.2.4 Cálculo del Esfuerzo Total del Proyecto.

Tabla 12: Distribución del Esfuerzo entre las diferentes actividades del módulo.

Actividad	Porcentaje	Esfuerzo
Análisis	10.00 %	205.67
Diseño	25.00 %	514.17
Implementación	30.00 %	617.00
Prueba	25.00 %	514.17
Sobrecarga (otras actividades)	10.00 %	205.67
Total	**100.00 %**	**2056.67**

El proyecto requiere de **2056.67** horas-hombre para su desarrollo. Trabajando 8 horas diarias se obtiene aproximadamente 257.08 días para el desarrollo del proyecto, es decir, para un equipo de 1 personas trabajando 26 días al mes el proyecto tiene una duración de 9.89 meses-hombre.

5.3 Beneficios tangibles e intangibles.

El Sistema de Gestión de los Cursos Optativos para la Facultad 2 es un software desarrollado fundamentalmente para beneficio de la Secretaría Docente, los Profesores y los Alumnos que se ven

involucrados en este proceso académico, el cual aporta una considerable agilización y eficiencia en este ámbito, contando con un sistema informático encargado de la gestión de dicho proceso.

Además de los beneficios tangibles que proporciona la implementación de este producto de software se generan los siguientes beneficios intangibles:

- Simplificación del proceso de publicación de los cursos optativos.
- Mejor divulgación de los cursos que se ofertan.
- Elimina la conglomeración de los estudiantes en el momento de matricularse.
- Elimina la entrada de datos erróneos de los estudiantes al matricularse.
- Disminución del tiempo y el esfuerzo que invierte la Secretaria Docente en la creación de los grupos y la emisión de las calificaciones de los estudiantes que reciben los cursos optativos.
- Generación automática de las Actas de los cursos optativos.

5.4 Análisis de costos y beneficios.

Salario medio de la fuerza de trabajo: $ 100 MN.
Cantidad de trabajadores: 1.

Como se tiene 2056.67 horas-hombre según lo estimado, se divide este valor entre 8 horas (cantidad real de horas que le dedica cada trabajador al desarrollo del sistema) para calcular la cantidad de días de trabajo-hombre que se necesitan para realizar la aplicación. Al realizar esta operación matemática se obtiene que el sistema de gestión necesita de 257.08 días de trabajo-hombre. Esta estimación se divide entre los 26 días laborales que tiene cada mes y se obtiene que la duración total de la implementación de la aplicación es 9.89 meses-hombre. El costo final del módulo sería:

Costo Total = Salario medio * Cantidad de trabajadores * Duración proyecto (meses)
Costo total del módulo = $ 100 * 1 * 9.89 = $ 989.00 MN.

La solución propuesta aporta varios beneficios tanto tangibles como intangibles, los cuales fueron descritos en el epígrafe anterior. El despliegue de esta aplicación en la Facultad 2 de la UCI contribuirá a hacer del Proceso de Gestión de los Cursos Optativos un proceso mucho ágil y eficiente. Por otra parte esta aplicación puede ser perfectamente una plantilla adaptable a cada una de las facultades de la

universidad permitiendo aportar con esta solución en el proceso de gestión de los cursos optativos del resto de las facultades. También es válido destacar que la totalidad de la tecnología utilizada para el desarrollo de la aplicación es libre, así como se utilizó el lenguaje de programación PHP para garantizar un producto de software multiplataforma. Por todo lo antes planteado se puede concluir que la implementación del Sistema de Gestión de los Cursos Optativos es Factible.

5.5 Conclusiones.

En el presente capítulo se efectuó el estudio de factibilidad correspondiente al desarrollo del sistema mediante el Método de Estimación Puntos por Casos de Uso. El mismo permitió llegar a la conclusión de que resultará factible implementar el Sistema de Gestión de la Información de los Cursos Optativos en la Facultad 2 de la UCI, debido a que el costo de producción, según la estimación realizada, es mucho menor que el costo actual de estos software en el mercado internacional.

CONCLUSIONES GENERALES

Se concluye el presente trabajo de diploma dando cumplimiento al objetivo propuesto para la realización del mismo. Mediante el presente documento, se ilustra al lector, acerca del proceso de desarrollo del Sistema de Gestión de la Información de los Cursos Optativos en la Facultad 2 de la Universidad de Ciencias Informáticas.

Además esta investigación arrojó los siguientes resultados:

- Se desarrolló una aplicación capaz de gestionar los cursos optativos y el proceso de matrícula de los estudiantes en los mismos.
- Se describieron los procesos, lo cual brindó una mayor claridad a la hora de implementar el sistema.
- Se seleccionaron las herramientas más adecuadas para el desarrollo del sistema, luego de haber realizado un estudio de las tendencias actuales en el desarrollo de aplicaciones web.
- Se realizó el levantamiento de requisitos funcionales y no funcionales, a partir de los cuales se definieron y describieron cada una de las funcionalidades que posee el sistema.
- Con la descripción de los Casos de Prueba se comprobó el correcto funcionamiento del sistema así como la fiabilidad y el rendimiento de la aplicación.
- Se pusieron en práctica los conocimientos adquiridos durante la carrera en las disciplinas de Ingeniería de Software y Programación, Metodología de la Investigación, entre otras.
- Se logró determinar la factibilidad y la estimación del esfuerzo y costo del sistema utilizando el método de estimación Puntos por Casos de Uso, con lo que se demostró que es factible implementar el mismo.

RECOMENDACIONES

Tomando como base la investigación realizada y la experiencia acumulada durante la realización del presente trabajo de diploma, se proponen las siguientes recomendaciones:

Se implementen nuevas funcionalidades que aporten comodidad, facilidad y resuelvan nuevas peticiones de los implicados según necesidades o interés de los mismos como son: un mecanismo para recoger la asistencia y las calificaciones de las evaluaciones sistemáticas de los estudiantes en los cursos optativos, así como archivar los mismos para tener un control de los curso que ha pasado cada estudiante y la nota obtenida en los mismos.

Se mantenga constante la actualización y mantenimiento del Sistema de Gestión de tal modo que pueda brindar cada una de las funcionalidades implementadas de la forma más óptima posible.

La Facultad conserve el documento como consulta y guía para aquellos que darán mantenimiento al sistema.

Poner en explotación el sistema y que forme parte del Portal de la Facultad 2, pudiendo de esta forma mantener su publicación en todo momento, accesible por el estudiantado y los profesores desde cualquier lugar de la universidad.

Que se expanda la solución informática a cada una de las facultades de la universidad logrando informatizar la gestión de los cursos optativos no sólo en la Facultad 2, sino en toda la universidad.

BIBLIOGRAFÍA

1. Thayer, R.H. y Dorfman, M. *Software Requeriments Engineering.* 2da edición. s.l. : IEEE Computer Society Press, 1997.

2. Jacobson, Ivar, Booch, Grady y Rumbaugh, James. *El proceso Unificado de Desarrollo de Software.* 1era edición. Madrid : Pearson Educación, S.A. - Addison Wesley, 2000. 84-7829-036-2.

3. Rumbaugh, James, Jacobson, Ivar y Booch, Grady. *El lenguaje unificado de modelado. Manual de referencia.* s.l. : Addison Wesley, 2000.

4. *The Unified Modeling Language for Object-Oriented Development.* s.l. : Rational Software Corp, 1997.

5. Andrade Fonseca, Roberto. *Programación de funciones en PL/pgSQL para PostgreSQL.* [PDF] s.l. : ABL Consultores, S.A. de C.V., 2002.

6. Equipo de desarrollo de PostgreSQL. *Tutorial de PostgreSQL.* [PDF] [ed.] Thomas Lockhart. s.l. : Postgres Global Development Group, 1999.

7. Brandenbaugh, Jerry. *Programación de aplicaciones Web con JavaScript.* Madrid : Anaya Multimedia S.A., 2000. 84-415-1070-9.

8. Mercer, Dave. *Fundamentos de programación en XML.* s.l.: Osborne McGraw-Hill, 2001. 958-4-0297-4.

9. James Garrett, Jesse. Ajax: A New Approach to Web Applications. [En línea] [Citado el: 24 de abril de 2011.] http://www.adaptivepath.com/ideas/essays/archives/000385.php.

10. The Apache Software Foundation. [En línea] [Citado el: 20 de marzo de 2011.] http://www.apache.org/.

11. Larman, Craig. *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* 2da. México : Pearson Education, 2003. 8420534382.

12. OMG. Portal Web Object Management Group. *Unified Modeling Language.* [En línea] 8 de enero de 2009. http://www.uml.org/.

13. Álvarez Romero, Eduardo y Pueyo, Daniel. *Integration Definition For Funcion Modeling (IDEF0).* 2004-2005.

14. Portal Web del Visual Paradigm. [En línea] http://www.visual-paradigm.com/.

15. Sitio Oficial de PHP. [En línea] 2011. http://www.php.net/.

16. Manual Online de Doctrine PHP. [En línea] [Citado el: 1 de marzo de 2011.] http://www.doctrine-project.org/documentation.

17. Breit, Kevin, House, Henry y Samson, Judith. Manual de Día. [En línea] 2000. http://projects.gnome.org/dia/doc/dia-manual.pdf.

18. *Ayuda en español del Rational Unified Process.* s.l. : IBM Corporation, 2006.

19. Yanoski Calderín Delgado, GESTACAD. Sistema para la Gestión Académica. [En línea] [Citado el 10 de febrero del 2011.] http://www.posgrados.frc.utn.edu.ar/congreso/trabajos/7.doc.

20. Peralta, Mario. *Estimación del esfuerzo basada en casos de uso.* [PDF] Buenos Aires, Argentina : s.n., 2004.

21. Potencier, Fabien, Zaninotto, François. *Symfony la guía definitiva.* [PDF]. www.librosweb.es.

22. Lellelid, Hans. *Propel - Guía de usuario.* [PDF]. Propel project. 2004.

23. Gaceta Oficial de la República de Cuba. Reglamento para el Trabajo Docente y Metodológico en la Educación Superior. Resolución 210/07. [PDF]. Cuba. 2007.

GLOSARIO

CUN: Caso de uso del negocio.

CUS: Caso de uso del sistema.

DCA: Diagrama de Clases del Análisis.

DCD: Diagrama de Clases del Diseño.

HTML: HyperText Markup Language. Lenguaje usado para escribir documentos para servidores World Wide Web.

HTTP: HyperText Transfer Protocol. Protocolo de Transferencia de Hipertextos. Modo de comunicación para solicitar páginas web.

Herramientas CASE: Herramientas utilizadas para el desarrollo de proyectos de Ingeniería de Software.

Hardware: Componentes electrónicos, tarjetas, periféricos y equipo que conforman un sistema de computación; se distinguen de los programas (software) porque son tangibles.

Microsoft: Compañía que manufactura los sistemas de operación DOS y Windows.

PHP: Hypertext Preprocessor. Es un ambiente script del lado del servidor que permite crear y ejecutar aplicaciones web dinámicas e interactivas.

PostgreSQL: es un Sistema de Gestión de Bases de Datos Objeto-Relacionales (ORDBMS) libre.

RUP: Rational Unified Process. Metodología para el desarrollo de Software.

Software: Programas de sistema, utilerías o aplicaciones expresados en un lenguaje de máquina.

SGBD: Sistema de Gestión de Bases de Datos. Es el software que permite la utilización y/o la actualización de los datos almacenados en una (o varias) base(s) de datos por uno o varios usuarios desde diferentes puntos de vista y a la vez.

UML: Unified Modeling Language. Es una notación estándar para modelar objetos del mundo real como primer paso en el desarrollo de programas orientados a objetos. Es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema de software.

WEB (WWW): Red de documentos HTML intercomunicados y distribuidos entre servidores del mundo entero.