UNIVERSIDAD DE ORIENTE
NÚCLEO DE ANZOÁTEGUI
ESCUELA DE INGENIERÍA Y CIENCIAS APLICADAS
DEPARTAMENTO DE COMPUTACIÓN Y SISTEMAS

"DESARROLLO DE UN SISTEMA AUTOMATIZADO BAJO ENTORNO WEB PARA EL CONTROL DE LA PROGRAMACIÓN ACADÉMICA EN LA UNIVERSIDAD DE ORIENTE NÚCLEO DE ANZOÁTEGUI"

REAZLIZADO POR:

Afonso R., Mariela A.

Segnini R., Jesus E.

Trabajo de grado presentado como requisito parcial para optar al Título de
INGENIERO EN COMPUTACIÓN

Barcelona, Junio de 2009

UNIVERSIDAD DE ORIENTE

NÚCLEO DE ANZOÁTEGUI

ESCUELA DE INGENIERÍA Y CIENCIAS APLICADAS

DEPARTAMENTO DE COMPUTACIÓN Y SISTEMAS

"DESARROLLO DE UN SISTEMA AUTOMATIZADO BAJO ENTORNO WEB PARA EL CONTROL DE LA PROGRAMACIÓN ACADÉMICA EN LA UNIVERSIDAD DE ORIENTE NÚCLEO DE ANZOÁTEGUI"

JURADO CALIFICADOR:

M.Sc. Lenys Ordaz

Asesor Académico

Ing. Rhonald Rodríguez

Jurado Principal

Ing. Claudio Cortínez

Jurado Principal

Barcelona, Junio de 2009

RESOLUCIÓN

De acuerdo al Artículo Nº 44 Del Reglamento de Trabajo de Grado

“Los trabajos de grado son de exclusiva propiedad de la Universidad y sólo podrán ser utilizados para otros fines con el conocimiento del Consejo de Núcleo respectivo, quién lo participará al Consejo Universitario”.

DEDICATORIAS

A Diosito que siempre está presente en todo lo que hago, a él debo la dicha de tener una familia tan maravillosa, mis logros y todo lo que soy.

A mi mamita hermosa por ser mi mejor amiga y darme apoyo incondicional solo el que una verdadera madre sabe dar, tan preocupada y dedicada, gracias a ella soy una persona de bien, es la madre más maravillosa del mundo.

A mi papito por consentirme y amarme tanto, sus consejos juntos con los de mi mami son los que me hicieron la gran persona que soy, siempre hizo hasta lo imposible por velar por mi bienestar y el de toda la familia.

A mi hermanita la más linda Maribel que a pesar de ser menor que yo me abrió los ojos ante el mundo, siempre apoyándome y deseándome lo mejor.

A mi novio Jesús Eduardo por su amor, su compañía, por apoyarme siempre y ayudarme a lograr mis metas.

Mariela A. Afonso R.

Este trabajo de grado está dedicado principalmente a mi mama Yolmar de la Coromoto Rodríguez, que desde el cielo te sientas orgullosa.

A Dios, por esta maravillosa vida, por lo bueno, y por lo malo, pues de todo he aprendido algo.

A mi padre Carlos Eduardo Segnini por su gran dedicación a sus hijos, gracias a su enorme esfuerzo tuvimos lo justo para que nunca nos faltara nada, tus ojos cansados siempre fueron motivo de empeño para seguir luchando por alcanzar mis metas.

A mi linda hermana Gabriela Segnini, quien desde su nacimiento ha llenado de alegrías mi vida.

A mi hermano Carlos Segnini, por ser una guía durante mi carrera.

A mi bella novia Mariela, por apoyarme en todo cuanto he necesitado.

Jesús E. Segnini R.

AGRADECIMIENTOS

A Diosito le doy gracias por todos mis logros y todo lo que soy, realmente soy feliz.

A la Universidad de Oriente por darme la oportunidad de estudiar a nivel superior y tratarme de la mejor manera posible haciendo realidad este logro.

A mis padres y hermanas por su apoyo y comprensión en todo momento.

A mi novio por ser incondicional por estar siempre a mi lado y ayudarme en todo lo que he necesitado.

A mi asesora Lenys Ordaz por apoyarme desde el comienzo con este proyecto, por ofrecerme toda la información relacionada con el sistema actual, además de estar pendiente del desarrollo del proyecto en todo momento.

A Carlos y Lanz por ofrecerme información sobre la base de Datos que administran en el Centro de Computación Académica.

A mis amigos: Freya, Luisana, Wilfredo, Juan Luis, Rima, Manuel, Feres, Vanessa, Carmen, Luis Miguel, Luis Eduardo, Luis Carlos y Rosnuel por todos los maravillosos momentos que vivimos juntos.

A los profesores del Departamento de Computación y Sistemas por sus enseñanzas.

A todos los profesores jefes de departamento y directores de escuela de la Universidad de Oriente, Núcleo de Anzoátegui por su atención y ayuda durante las entrevistas.

A todos mis familiares: abuelos, tíos, primos que de alguna manera me han apoyado.

A mis compañeros de clases por su apoyo y compartir esta meta.

Gracias a todos.

Mariela A. Afonso R.

Agradezco a Dios por ser mi principal guía, en especial en los momentos más difíciles cuando solo en sus palabras pude encontrar consuelo.

A mí querida madre Yolmar de la Coromoto Rodríguez por nunca rendirse, por haberme dado la vida, y sobre todo por su amor incondicional.

Gracias a mi bella hermana Gabriela Segnini, por ser mi mayor alegría y fuente inagotable de admiración.

A mi hermano Carlos Segnini por cada valioso consejo y por haber marcado el camino para mi éxito dejando suficientes señales a mi nombre para no perderme en los trayectos obscuros.

Agradezco a mi querida novia Mariela Afonso, que llegó justo a tiempo para detener mi caída y enseñarme a caminar de nuevo cuando había olvidado cómo hacerlo, por darle nuevo sentido a la palabra amor y ser tan comprensiva.

Gracias a la Sra. Luisa, el Sr. Manuel y Maribel, mi otra familia, quienes me abrieron las puertas de su casa y de sus corazones.

A mi asesora Lennys Ordaz, siempre atenta y dispuesta a ayudarnos.

Agradezco a mis profesores, por haber compartido sus conocimientos.

A mis compañeros de clases, mis amigos, sin los cuales no hubiese podido llegar hasta este punto.

Gracias a todos!.

Jesús E. Segnini R.

RESUMEN

El siguiente proyecto de investigación se basa en el desarrollo de un sistema automatizado para el control de la Programación Académica (SACPA) para la Universidad de Oriente, Núcleo de Anzoátegui. El software se encarga de proporcionar una interfaz agradable y de fácil manejo en entorno web a los diferentes departamentos académicos de la Institución y a los directores de escuela para ingresar y administrar la Programación Académica que elaboran durante cada periodo académico. Ademas de permitir consultas por parte de los estudiantes y los profesores. Esta información relacionada con la programación académica es guardada directamente en la base de datos ubicada en el Centro de Computación Académica que es el administrador del sistema, lo que mejora la comunicación entre este centro y los departamentos. Este sistema garantiza información confiable ya que uno de sus principales objetivos es validar los datos ingresados, además permite a los usuarios consultar un mapa de aulas con el fin de mejorar la planificación de su programación. El software se elaboró utilizando el lenguaje de programación PHP versión 5.2.0, la técnica de programación AJAX, el servidor Apache versión 5.0 y el motor de base de datos MySQL versión 5.0. Este proyecto se construyó siguiendo el Proceso Unificado de Desarrollo de Software, la herramienta WebML y el Lenguaje de Modelado UML.

CONTENIDO

CAPITULO I. PLANTEAMIENTO DEL PROBLEMA

1.1 Introducción

La tecnología a nivel mundial, siempre ha sido parte importante dentro de la sociedad y la educación, donde las universidades forman parte integral de este proceso, en el ámbito de la información y la comunicación, las instituciones utilizan sistemas y recursos para el desarrollo y difusión digitalizada de la información. El desarrollo de aplicaciones Web como complemento al proceso de aprendizaje se convierte en un recurso imprescindible en nuestros días. En la actualidad, se incorporan estas nuevas tecnologías al proceso educativo en sus distintos niveles con la finalidad de variar y flexibilizar las oportunidades de aprender sin restricciones de lugar, tiempo y atendiendo a las diferencias individuales y de grupo. La Universidad de Oriente**,** específicamente el Núcleo de Anzoátegui se perfila en concordancia con estos nuevos avances en el área de la ciencia y la tecnología, por medio de la incorporación de nuevos proyectos enmarcados al desarrollo de aplicaciones usando plataforma Web, que permitirán dar a conocer y satisfacer las necesidades de información a toda la comunidad universitaria.

En el año de 1995 por Resolución CU-08/95 del Consejo Universitario, la Universidad de Oriente en el Núcleo de Anzoátegui funda la Dirección de Servicios de Computación con dos dependencias, el Centro de Computación Académica y el Centro de Computación Administrativa. El Centro de Servicios de Computación Académica de la Universidad de Oriente del Núcleo de Anzoátegui, permite apoyar a todas las dependencias académicas que lo conforman tales como: Decanato, Coordinaciones, Escuelas, Departamentos, Dirección de Bienestar Estudiantil y por supuesto la principal dependencia que es el Departamento de Admisión y Control de

Estudios (D.A.C.E), a través de sistemas de información académicos automatizados y prestación diaria de soporte técnico. A su vez, éste centro se encarga de planificar, organizar, coordinar y supervisar diferentes servicios que ofrece a estas dependencias. También el resguardo de los datos estudiantiles, la programación académica, emisión de diferentes informes impresos, presentación anual de las actividades del departamento a entidades superiores y cumplir con los deberes y atribuciones que señalan los reglamentos y resoluciones de la institución.

El personal administrativo que labora en este centro de computación puede tomar decisiones a la hora de planificar, coordinar y desarrollar nuevos proyectos en beneficio de la comunidad universitaria. Actualmente el personal de este centro está compuesto por un jefe del centro de computación académica y tres analistas de sistemas. El jefe del centro, se encarga de planificar, organizar, coordinar y supervisar todas las actividades que se desenvuelven en él, así como también delegar funciones al resto de sus empleados como son los analistas de sistemas, que se encargan de analizar, desarrollar e instalar nuevos sistemas de información y mantener los ya existentes, todo enmarcado dentro de las necesidades de la institución.

En la presente investigación se desarrolla un sistema que permitirá automatizar el control de la programación académica de la Universidad de Oriente – Núcleo de Anzoátegui.

1.1.1 Problema

El Centro de Servicios de Computación Académica, es el que lleva a cabo el proceso de inscripción de los estudiantes regulares de la universidad por cada periodo académico, en el cual participan ciertos elementos que son considerados

claves dentro del sistema. Evidentemente, estos elementos que forman parte del sistema del proceso de inscripción están enmarcados por la información suministrada por las distintas direcciones de escuelas, coordinación académica y D.A.C.E, como se muestra en la figura 1.1.

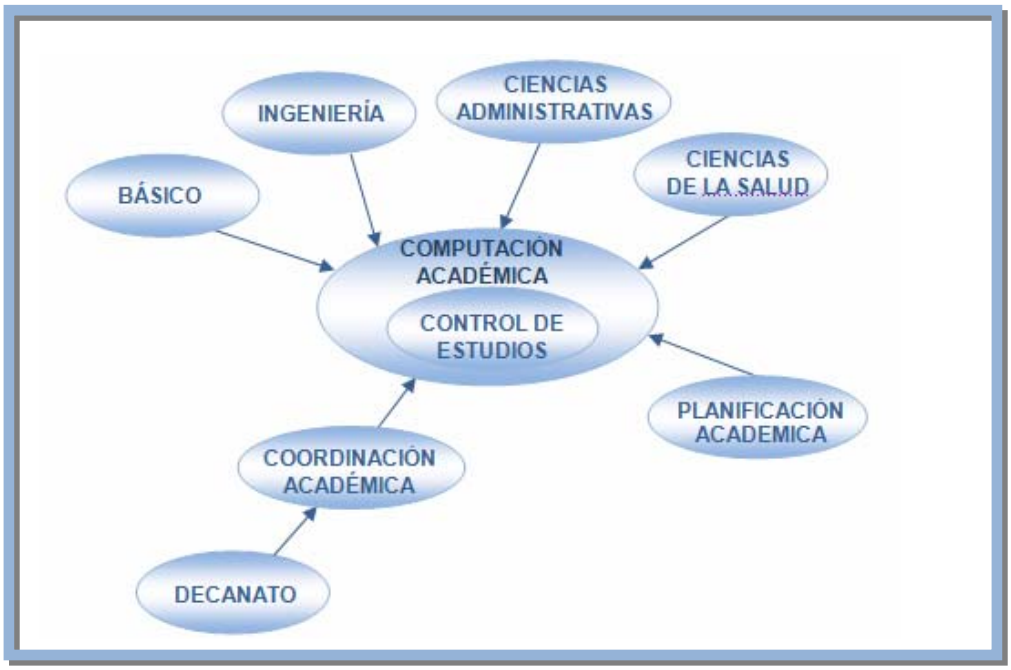


Figura 1.1. Elementos relacionados con el Proceso de Inscripción. (Fuente: Centro de Computación Académica)

Previamente al desarrollo de las inscripciones regulares, en lo referente a datos e información se procesan las preactas con las notas de cada materia para poder actualizar la data en los servidores, se prepara la generación de las citas de inscripción (hora y día) por estudiante, cabe destacar la carga de la programación académica de materias y secciones es suministrada por las diferentes Direcciones de Escuelas de la Institución, que a su vez la recibe de los departamentos.

El Centro de Servicios de Computación Académica es un ente de desarrollo,

implantación y mantenimiento de los sistemas de información académicos automatizados que existen actualmente en este núcleo, la eficiencia operativa de esta dependencia estaría siendo afectada por la ausencia de aplicaciones Web para todo el entorno del proceso de inscripción de los estudiantes regulares de la Universidad de Oriente del Núcleo de Anzoátegui, a pesar de que ya se han realizado varias aplicaciones Web, entre las cuales se encuentran: el sistema de carga de notas o preactas, el sistema de asociación de preactas al profesor, el sistema de preinscripción de estudiantes regulares, el sistema de preinscripción de nuevos ingresos, faltando el sistema para el control de la programación académica, para apoyar de manera efectiva el proceso de inscripción de estudiantes regulares. Esta carencia se estima como una debilidad que se agudiza durante la jornada de inscripción debido a que el mecanismo existente no aporta la respuesta adecuada para los usuarios y el personal al frente del sistema, esto afecta al proceso de inscripción de forma directa. Se presenta una situación problemática caracterizada por:

El aumento de la población estudiantil a lo largo de varios años, como se muestra en la tabla 1.1 lo que ha ocasionado un colapso durante las inscripciones debido a la insuficiencia de espacio físico (planta física), que no cubre la demanda de estudiantes que asisten al proceso.

Año	Matrícula
1996	12397
1997	13661
1998	13931
1999	13733
2000	13402

Tabla 1.1 Matrícula Estudiantil (Centro de Servicios de Computación Académica, 2007) 1/2

Año	Matrícula
2001	13113
2002	14185
2003	14495
2004	15999
2005	16819
2006	17428
2007	17476
2008	19327

Tabla 1.1 Matrícula Estudiantil (Centro de Servicios de Computación Académica, 2007) 2/2

Existen fallas muy frecuentes en la programación académica, suministrada por las distintas direcciones (Ingeniería, Ciencias de la Salud, Ciencias Administrativas y la Dirección de Apoyo a la Docencia Áreas Interdisciplinarias) de la institución, en cuanto a la cantidad de secciones por asignaturas planificadas que no satisfacen la demanda de cupos ya que frecuentemente deben ser modificados. Entre estas fallas podemos mencionar:

La escuela de Ciencias de la Salud realiza una programación académica usando un editor de texto y el Centro de computación Académica es el que se encarga de revisar y cargar la programación en el formato requerido por el sistema

para su posterior uso en las inscripciones, esto puede generar errores, ya que una sola persona se encarga de adaptar esta programación académica.

Las otras tres direcciones: Ingeniería, Ciencias Administrativas y la Dirección de Apoyo a la Docencia Áreas Interdisciplinarias (Básico), procesan la programación académica de sus departamentos usando un programa con una base de datos local que no se encuentra interconectada en cada dirección, este sistema está desarrollado en Visual Basic con una base de datos en Access, donde la programación académica es cargada por una sola persona que la mayoría de las veces comete errores, bien sea de trascripción o simplemente cargando la programación que ya viene con errores desde los departamentos, por lo general quien carga la programación en el sistema son las secretarias de estas direcciones.

Al no estar interconectadas las bases de datos de las direcciones pueden existir choques de aulas, ya que las tres direcciones manejan todas las aulas independientemente una de otras.

También se puede dar el caso de que choquen materias pertenecientes a un mismo semestre, esto porque al momento de verificar la programación lo pasan por alto.

La presente investigación se basa fundamentalmente en solventar las dificultades que se presentan con la programación académica en el proceso de inscripción de los estudiantes regulares, con una propuesta que se inicia con un sistema que permita cargar la programación académica, empleando tecnología Web, esto con el fin de proporcionar un excelente servicio a la comunidad universitaria y mejorar los procesos de inscripción de estudiantes regulares y de planificación en los diferentes departamentos.

Con la data que genera este sistema se proporciona información veraz a todos los entes involucrados, debido a que podrán acceder al mismo y obtener información acerca de la cantidad de estudiantes planificados por materias, para así poder tomar decisiones de aperturas de nuevas secciones a corto plazo.

El sistema permitirá a los departamentos realizar la carga de la programación académica independientemente uno de otros sin tener que hacerlo en las direcciones, simplemente desde cada departamento usando la Intranet existente o mediante el uso de Internet solo en caso de ser necesario, se accederá al sistema a través de la pagina del Centro de Computación Académica con previa identificación.

Además de esto se podrá consultar un mapa de aulas por parte de cada uno de los departamentos académicos, con la finalidad de que entre departamentos se puedan facilitar aulas en los horarios que les quedan disponibles.

En función de las anteriores consideraciones se puede formular la siguiente interrogante: *¿Cuáles serán los criterios y procedimientos a seguir para desarrollar un sistema basado en tecnología Web a objeto de minimizar las deficiencias detectadas en los actuales procesos de carga de la programación académica de la Universidad de Oriente - Núcleo de Anzoátegui y la influencia sobre las entidades académicas?*

1.1.2 Propósito

El fin de este trabajo es el desarrollo del Sistema para el control de la Programación Académica para estudiantes regulares en un entorno Web utilizando

tecnologías de Base de Datos, lenguajes de cuarta generación y una arquitectura Cliente/Servidor que funcione sobre la plataforma de red existente en el Núcleo.

Este trabajo se realizará aplicando la disciplina de Ingeniería de Software, la cual se puede obtener mediante la combinación de métodos completos para todas las fases de desarrollo de software.

En el sistema a desarrollar se utilizará el Lenguaje Unificado para Modelado UML, ya que este lenguaje define una notación y un proceso para construir Sistemas de Software complejos y ofrece un rico conjunto de métodos lógicos y físicos con los cuales se puede razonar sobre diferentes aspectos del sistema.

En vista de lo anterior, se plantea el desarrollo de un sistema automatizado, el cual será concebido con mecanismos que permitan la creación, mantenimiento, respaldo y seguridad de la información que se maneja en el área de Pregrado, que incluye los datos personales y académicos de los profesores, los pensum de estudios, las aulas disponibles, y horarios para las diferentes las materias que se dictan, con sus respectivas secciones, además de la generación de los reportes necesarios.

Esta implementación se esquematiza a los fines de proveer beneficios que establezcan un control y organización del sistema actual, permitiendo la estandarización de los diferentes procesos que se realizan en cada una de las direcciones de escuela, y así mejorar la comunicación de éstas con el Centro de Computación Académica. Las Bases de Datos, son como depósitos para los datos, que son almacenados unificadamente y que se pueden compartir por varios usuarios. Esto permite reducir la inconsistencia, reforzar los estándares, crear restricciones de seguridad y de integridad de la información.

La programación aplicada en este proyecto empleará la herramienta PHP 5, lenguaje de cuarta generación que trabaja en el contexto de una arquitectura Cliente/Servidor, donde las aplicaciones sugieren una forma de procesamiento cooperativo entre bancos de computadores personales y un servidor de base de datos central.

1.1.3 Importancia

La importancia de esta investigación usando tecnología Web radica en proporcionar soluciones a corto o mediano plazo ante la imperiosa necesidad de mejorar tanto las actividades relacionadas con el proceso de inscripción de los estudiantes regulares como la toma de decisiones de todos los departamentos con relación a la programación académica en la Universidad de Oriente - Núcleo de Anzoátegui.

La implantación y utilización de este sistema traerá consigo una notable mejora en la relación Tiempo-Trabajo en cuanto a la programación académica, que permitirá un mejor desempeño en la funcionalidad del Sistema de Admisión y Control de Estudios de de la Universidad de Oriente Núcleo de Anzoátegui.

1.1.4 Alcance

El Sistema automatizará todos los procesos referentes a la programación académica de la Universidad de Oriente, Núcleo de Anzoátegui, produciendo un mayor rendimiento tiempo/trabajo.

Con la realización de este proyecto se pretende elaborar una herramienta de gestión para solventar la problemática asociada a la programación académica de los estudiantes regulares de la Universidad de Oriente, Núcleo de Anzoátegui, este sistema utiliza plataforma Web y se desarrolló mediante la herramienta WebML, el lenguaje de programación PHP 5, el lenguaje de consultas SQL y el motor de base de datos MySQL 5.

La herramienta WebML se utiliza como herramienta de análisis y diseño del sistema. Para modelar los escenarios que conforman el sistema a través de los esquemas estáticos y dinámicos que permiten adaptar la investigación sin problemas, se usa el Lenguaje Unificado para Modelado (UML) como metodología.

En términos prácticos, la presente investigación propone que el sistema bajo plataforma Web tome una perspectiva integral, global, permitiendo que directivos de la academia y estudiantes (ya que estos tendrán acceso para consultar la información relacionada con la programación académica) se involucren dentro de nuevas tecnologías y satisfagan sus necesidades.

1.1.5 Justificación

Desde el punto de vista práctico el desarrollo del sistema se orienta a la mejora del servicio prestado por el Centro de Servicios de Computación Académica para el proceso de inscripción de alumnos regulares y sirve de soporte informático a las actividades realizadas dentro de la institución relacionadas con este proceso, logrando de esta manera la eficiencia en el manejo de la información.

Con la innovación del sistema en plataforma Web, los departamentos pueden realizar la carga de la información relacionada con la programación

académica a través del portal desde la página principal del Centro de Servicios de Computación Académica de la Universidad de Oriente del Núcleo de Anzoátegui, facilitando el acceso a los usuarios. Cada departamento es responsable de su programación debido a que tendrá control total sobre el sistema, ingresando data, realizando consultas entre ellas la del mapa de aulas que les será de mucha ayuda al momento de planificar o modificar la programación. Los departamentos tienen acceso al sistema en línea desde cualquier sala de computación de esta institución (Intranet) y desde Internet.

Para el Centro de Servicios de Computación Académica el proyecto de investigación se presenta como una alternativa a objeto de elevar su calidad de gestión en beneficio de la comunidad universitaria.

1.1.6 Originalidad

Este es uno de los primeros proyectos que se desarrollará en la Universidad de Oriente – Núcleo de Anzoátegui mediante Software Libre.

Además este proyecto es uno de los pocos que será implementado en la universidad, gracias al Centro de Computación Académica.

1.2 Objetivos

1.2.1 Objetivo General

Desarrollar un sistema basado en tecnología Web, para el control de la programación académica de los estudiantes regulares de la Universidad de Oriente, Núcleo de Anzoátegui.

1.2.2 Objetivos Específicos

- Diagnosticar la situación actual del proceso de la programación académica de estudiantes regulares del núcleo de Anzoátegui en la Universidad de Oriente.
- Identificar los requerimientos del sistema a desarrollar partiendo del análisis de la información recopilada del sistema actual.
- Definir la estructura del sistema, identificando los modelos de datos, comportamientos y relaciones.
- Estudiar la base de datos existente.
- Desarrollar módulos de programación del sistema de información propuesto utilizando herramientas de desarrollo computarizado.
- Elaborar la documentación del usuario y de mantenimiento del Software.
- Instalar el software.

CAPITULO II

Marco Teórico

CAPITULO II. MARCO TEORICO

2.1 Antecedentes de la investigación

A continuación se establecen los antecedentes más próximos al problema. Entre los trabajos realizados están:

- **(Martha Elena García León y Víctor Julio Mujica Vargas, Junio 2005). "Desarrollo de un Software para la Automatización de las actividades Administrativas del Departamento de Computación y Sistemas de la Universidad de Oriente – Núcleo de Anzoátegui".** El Proyecto se elaboró siguiendo los lineamientos del proceso unificado de desarrollo de software el objetivo primordial fue agilizar la gestión administrativa del departamento de computación y sistemas a través de la automatización de de las tareas administrativas llevadas a cabo. [1]

- **(Dr. Sánchez David, 2007)** "**Aplicación Web. Sistema Web del Estudiante – Universidad del Zulia**". Este sistema se centra en la idea de prestar un mejor servicio e interacción entre los procesos académicos de la dirección docente y el alumnado de la Universidad. Los estudiantes tendrán una inscripción interna por facultades que comienza con una pre-inscripción por carrera. Existen otros proyectos basados en tecnología Web tales como: Sitio Web "La Web del Estudiante", Aplicación Web "Sistema de Admisión Web a LUZ" y Material Multimedia Interactivo e Informativo para la Web. [2]

- **(Colmenares Giancarlo, 2.004). "Sistema de Inscripción vía Web".** El sistema de Inscripción vía Web fue realizado por el ingeniero Giancarlo Colmenares en la Universidad Experimental del Táchira (UNET). El primer

sistema fue desarrollado por el profesora Marisol Mantilla en el II período del 2004, pero fue rediseñado totalmente por Giancarlo período 3 del 2004. Al finalizar cada semestre y una vez entregadas las notas definitivas a los estudiantes, se les genera la oferta académica y esta es publicada automáticamente en la página Web. Allí el estudiante puede visualizar las materias ofertadas y el horario que posee cada una de las secciones para que cuadren el horario. El orden de inscripción es calculado de acuerdo al índice académico del índice más alto al más bajo y la activación de la inscripción se realiza cada 13 segundos. En la página de la inscripción como tal el estudiante puede cuadrar el horario y sólo verá las secciones en las cuales queden cupos disponibles. Los choques de horario son marcados con rojo. [3]

2.2 Universidad de Oriente

La Universidad de Oriente fue creada el 21 de noviembre de 1958, mediante el Decreto Ley Nro. 459 dictado por la Junta de Gobierno presidida por el Dr. Edgard Sanabria, siendo Ministro de Educación el Dr. Rafael Pizani.

La conformación de la Universidad de Oriente en cinco Núcleos, obedece a su filosofía de regionalización de la Educación Superior, estructurándose cada uno de ellos de acuerdo a la vocación de los estados en que se asientan y a las facilidades ofrecidas por los mismos. De este modo fue considerada la vocación hacia las principales actividades económicas y de producción de cada entidad federal y sus implicaciones:

- La agricultura, la ganadería, los hidrocarburos y el turismo, en Anzoátegui.
- La agricultura, la ganadería, la minería y el turismo, en Bolívar.
- La agricultura, la ganadería, los hidrocarburos y el turismo, en Monagas.

- El comercio, la pesca y el turismo, en Nueva Esparta.
- La agricultura, la pesca y el turismo, en Sucre.

Todo ello sin excluir otras posibilidades locales de desarrollo, tanto económico como social. Ver figura 2.1.

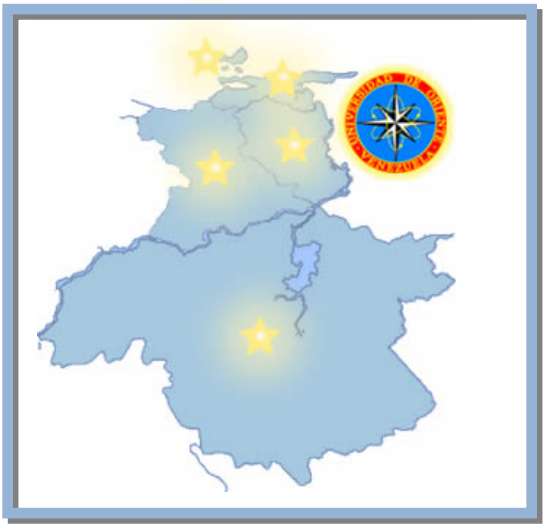


Figura 2.1. Ubicación de los Núcleos de la universidad de Oriente

El 20 de febrero de 1960, por Resolución del Consejo Universitario se crea en Barcelona, el Núcleo de Anzoátegui de la Universidad de Oriente, respondiendo a las exigencias regionales de profesionales y técnicos. Este núcleo inicia sus actividades docentes el 12 de febrero de 1963, con la apertura de las carreras de Ingeniería Eléctrica, Ingeniería Mecánica, Ingeniería Industrial e Ingeniería Química.

En el segundo semestre de 1974 se reestructura el Núcleo de Anzoátegui, creándose las Escuelas de Ingeniería y Ciencias Aplicadas, la Escuela de Ciencias Administrativas, la Escuela de Medicina y la Unidad de Estudios Básicos.

Actualmente se dictan 14 carreras en su sede de Barcelona. La extensión de Anaco se creó para ofrecer las carreras de Contaduría Pública, Administración, Ingeniería Industrial e Ingeniería de Sistemas. A continuación podemos en la figura 2.2 ver el mapa del Núcleo de Anzoátegui de la Universidad de Oriente.

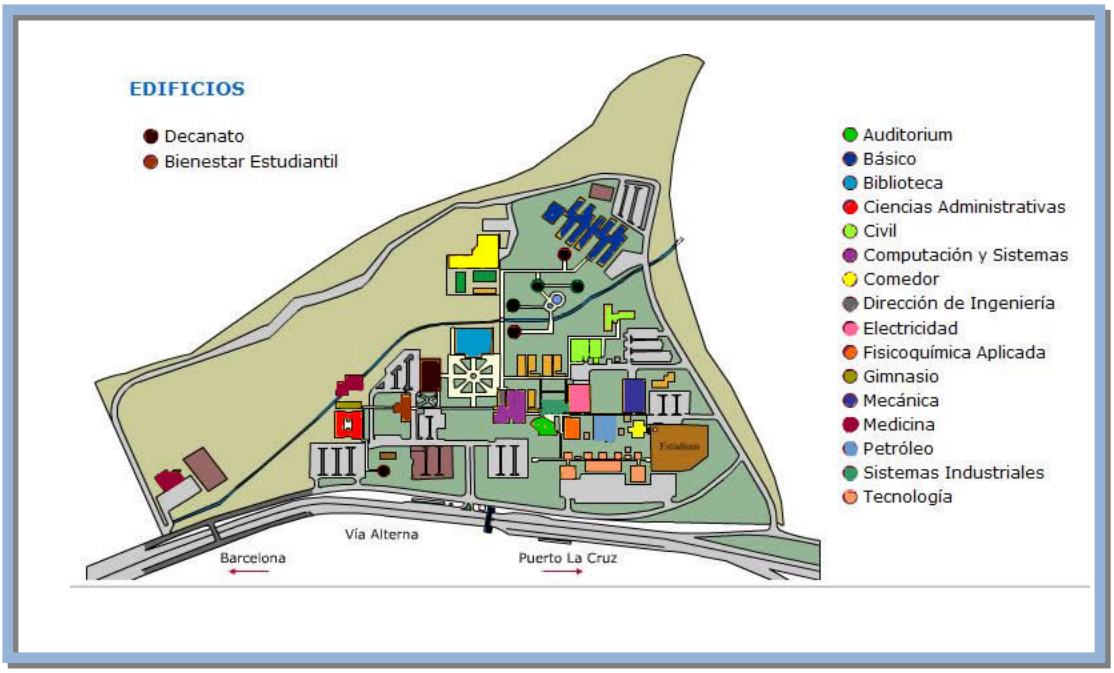


Figura 2.2. Mapa de la universidad de Oriente – Núcleo de Anzoátegui

Dentro de la universidad de Oriente se encuentra en Centro de computación académica el cual tiene como misión, Planificar, Coordinar y Supervisar todas aquellas actividades relacionadas con el análisis, diseño, administración, mantenimiento y actualización de sistemas de información y la red académica (Intranet) de la comunidad universitaria. Además de autorizar y supervisar el desarrollo de proyectos que involucren el uso de la red o plataforma Teleinformática en el área académica.

Tiene como objetivo apoyar a las dependencias académicas del núcleo como: departamento de Admisión, Control de Estudios, Dirección de Bienestar Estudiantil y Oficina de Planificación, entre otros; mediante el mantenimiento del Sistema Automatizado Académico-Administrativo de Control de Estudios, así como la impresión de diferentes listados y el resguardo de los datos estudiantiles (ver figura 2.3).

Las funciones del centro de computación académica son las siguientes:

- Organizar, promover, coordinar y supervisar los diferentes servicios que presta este departamento.
- Presentar el Informe Anual de Actividades.
- Realizar reuniones periódicas con el fin de promover y evaluar los diferentes servicios que presta el Departamento.
- Realizar reuniones con el Director de Servicios de Computación a fin de coordinar, evaluar y proyectar los diferentes servicios que se prestan.
- Cumplir con los deberes y atribuciones que señalan los Reglamentos y Resoluciones de la Institución. [4]

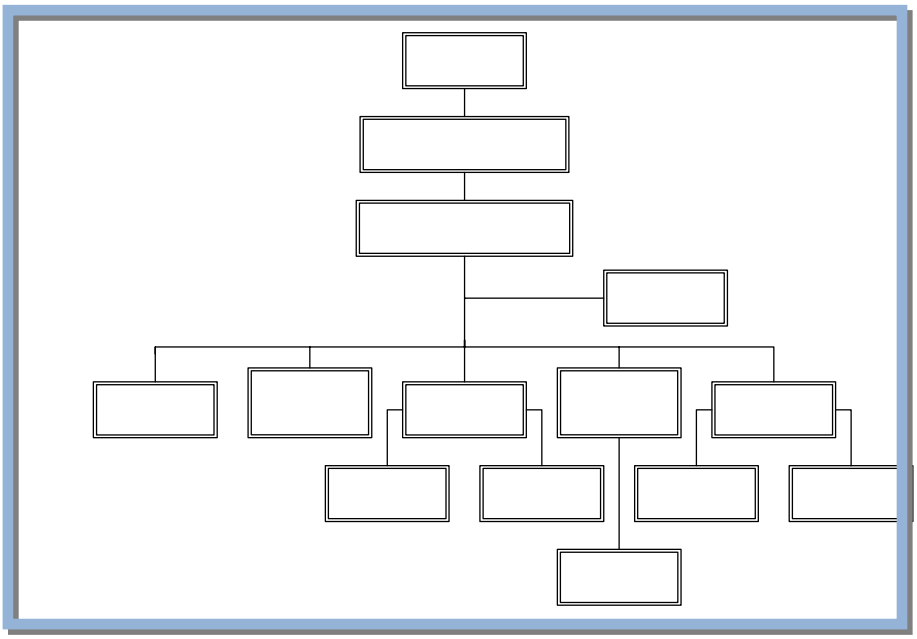


Figura 2.3. Organigrama Centro de Computación Académica

2.3 Sistemas de Informacion

2.3.1 Definición [5]

Un sistema, es un conjunto de componentes que interactúan entre sí para lograr un objetivo común. Una organización es un sistema, donde sus componentes: mercadotecnia, manufactura, ventas, investigación, embarques, contabilidad y personal; trabajan juntos para crear utilidades que beneficien tanto a los empleados como a los accionistas de la compañía. Cada uno de estos componentes es a su vez un sistema. Todo sistema depende en mayor ó menor medida, de una entidad abstracta denominada sistema de información. Este sistema es el medio por el cual los datos fluyen de una persona o departamento hacia otros y puede ser cualquier cosa, desde la comunicación interna entre los diferentes componentes da la organización y líneas telefónicas hasta sistemas de cómputo que generan reportes periódicos para varios

usuarios. Los sistemas de información proporcionan servicio a todos los demás sistemas de una organización y enlazan todos sus componentes en forma tal que estos trabajen con eficiencia para alcanzar el mismo objetivo.

2.3.2 Actividades básicas de los sistemas de información [5]

Según (Peralta, 1997) el sistema de información realiza cuatro actividades básicas: entrada, almacenamiento, procesamiento y salida de información:

- **Entrada de Información:** es el proceso mediante el cual el Sistema de Información toma los datos que requiere para procesar la información. Las entradas pueden ser manuales o automáticas.
- **Almacenamiento de información:** el almacenamiento es una de las actividades o capacidades más importantes que tiene una computadora, ya que a través de esta propiedad el sistema puede recordar la información guardada en la sección o proceso anterior.
- **Procesamiento de Información:** es la capacidad del Sistema de Información para efectuar cálculos de acuerdo con una secuencia de operaciones preestablecida. Estos cálculos pueden efectuarse con datos introducidos recientemente en el sistema o bien con datos que están almacenados.
- **Salida de Información:** la salida es la capacidad de un Sistema de Información para sacar la información procesada o bien datos de entrada al exterior. Las unidades típicas de salida son las impresoras, terminales, diskettes.

2.3.3 Análisis y diseño del sistema [5]

El análisis y diseño del sistema se refiere al proceso de explorar la situación de la organización con el propósito de mejorarla con métodos y procedimientos más adecuados, en el caso de este proyecto es utilizando el *Lenguaje de Modelado Unificado* (UML) de manera novedosa.

El análisis de sistemas, es el proceso de clasificación e interpretación de hechos, diagnósticos de problemas y empleo de la información para recomendar mejoras al sistema. El análisis especifica que es lo que el sistema debe hacer.

El diseño del sistema, es el proceso de planificar, reemplazar ó complementar un sistema organizacional existente. Es como los planos de un edificio: especifica todas las características del producto terminado, es decir, establece como alcanzar el objetivo.

2.3.4 Desarrollo del diseño del sistema [5]

Es un proceso que consta de las siguientes actividades:

2.3.4.1 Investigación preliminar

La solicitud para recibir ayuda de un sistema de información puede originarse por varias razones; sin importar cuales sean éstas, el proceso se inicia siempre con la petición de una persona (administrador, empleado, ó especialista en sistemas). Cuando se formula la solicitud, comienza la primera actividad: la investigación preliminar. Esta tiene tres partes: Aclaratoria de la solicitud, estudio de factibilidad y aprobación de la solicitud.

- **Aclaración de la solicitud:** Muchas solicitudes que provienen de empleados y usuarios no están formulados de manera clara. Por consiguiente antes de considerar cualquier investigación de sistemas, la solicitud de proyecto debe examinarse para determinar con precisión lo que el solicitante desea. Si el solicitante pide ayuda sin saber qué es lo que está mal ó donde se encuentra el problema, la aclaración del mismo se vuelve más difícil. En cualquier caso, antes de seguir adelante, la solicitud del proyecto debe estar claramente planteada.

    - **Estudio de factibilidad:** El sistema solicitado debe ser factible en tres aspectos:
    - ✓ ***Factibilidad técnica*:** El trabajo para el proyecto, ¿puede realizarse con el equipo actual, la tecnología existente de software y el personal disponible?, si se necesita una nueva tecnología ¿cuál es la posibilidad de desarrollarla?
    - ✓ ***Factibilidad económica:*** Al crear el sistema, ¿los beneficios que se obtienen serán suficientes para aceptar los costos?, ¿los costos asociados con la decisión de no crear el sistema son tan grandes que se debe aceptar el proyecto?
    - ✓ ***Factibilidad operacional*:** Si se desarrolla e implanta, ¿será utilizado el sistema?, ¿existirá cierta resistencia al cambio por parte de los usuarios?

    - **Aprobación de la solicitud:** No todos los proyectos solicitados son deseables o factibles. Los que son deben incorporarse a los planes. Muchas organizaciones desarrollan sus planes para sistemas de información con el mismo cuidado con el que planifican nuevos productos y programas de fabricación o la expansión de las

instalaciones. Después de aprobar la solicitud del proyecto se estima su costo, el tiempo necesario para terminarlo y las necesidades de personal; con esta información se determina donde ubicarlo dentro de la lista existente de proyectos.

2.3.4.2 Determinación de los requerimientos del sistema

El aspecto fundamental del análisis del sistema es comprender todas las facetas importantes de la parte de la organización que se encuentra bajo estudio. Los analistas, al trabajar con los empleados y administradores, deben estudiar los procesos de la organización para dar respuesta a las siguientes preguntas claves:

1. ¿Qué es lo que se hace?
2. ¿Cómo se hace?
3. ¿Con que frecuencia se presenta?
4. ¿Qué tan grande es el volumen de transacciones o de decisiones?
5. ¿Cuál es el grado de eficiencia con el que se efectúan las tareas?
6. ¿Existe algún problema?
7. Si existe un problema, ¿qué tan serio es?
8. Si existe un problema, ¿cuál es la causa que lo origina?

Para contestar estas preguntas, el analista conversa con varias personas para reunir detalles relacionados con los procesos de la organización, sus opiniones sobre por qué ocurren las cosas, las soluciones que proponen y sus ideas para cambiar el proceso. Se emplean cuestionarios para obtener esta información cuando no es posible entrevistar, en forma personal, a los miembros de grupos grandes dentro de la organización. Asimismo, las investigaciones detalladas requieren el estudio de manuales y reportes, la observación en condiciones reales de las actividades del trabajo y, en algunas ocasiones, muestras de formas y documentos con el fin de comprender el proceso en su totalidad.

Conforme se reúnen los detalles, los analistas estudian los datos sobre requerimientos con la finalidad de identificar las características que debe tener el nuevo sistema, incluyendo la información que deben producir los sistemas junto con características operacionales tales como controles de procesamiento, tiempo de respuesta y métodos de entradas y salidas.

2.3.4.3 Diseño del sistema

El diseño de un sistema de información produce los detalles que establecen la forma en la que el sistema cumplirá con los requerimientos identificados durante la fase de análisis. Los especialistas en sistemas se refieren, con frecuencia, a esta etapa como *diseño lógico* en contraste con la de desarrollo de software a la que denominan *diseño físico.*

Los analistas de sistemas comienzan el proceso de diseño identificando los reportes y demás salidas que debe producir el sistema. Hecho lo anterior se determinan con toda precisión los datos específicos para cada reporte y salida.

El diseño de un sistema también indica los datos de entrada, aquellos que serán calculados y los que deben ser almacenados. Los documentos que contienen las especificaciones de diseño representan a éste de muchas maneras (diagramas, tablas, y símbolos especiales). La información detallada del diseño se proporciona al equipo de programación para comenzar la fase de desarrollo de software.

En este trabajo de investigación el diseño del sistema de información se efectuará utilizando para ello el *Lenguaje de Modelado Unificado* (UML), herramienta novedosa para el desarrollo de sistemas.

2.4 INGENIERÍA DE SOFTWARE

2.4.1 Definición [6]

La Ingeniería de Software es el establecimiento y uso de principios de la ingeniería para obtener económicamente un software confiable y que funcione de modo eficiente en máquinas reales.

Es un enfoque sistemático del desarrollo, operación, mantenimiento y retiro del software. Es decir, se considera que es la rama de la ingeniería que aplica los principios de la ciencia de la computación y las matemáticas para lograr soluciones costo-efectivas (eficaces en costo o económicas) a los problemas de desarrollo de software, es decir, permite elaborar consistentemente productos correctos, utilizables y costo-efectivos.

La Ingeniería de Software es una tecnología estratificada en cuatro partes:

- **Un enfoque de calidad**: cualquier enfoque de la ingeniería (incluido el de la ingeniería de software) debe estar sustentado en un compromiso de calidad. Como por ejemplo: La Gestión de Calidad Total, Sigma Seis y otros enfoques similares que fomentan una cultura de mejora continua del proceso, y esta cultura es la que al final conduce al desarrollo de enfoques muy efectivos para la ingeniería de software.
- **El proceso**: el proceso del software forma la base para el control de la gestión de los proyectos de software y establece el contexto en el cual se aplican los métodos técnicos, se generan los productos del trabajo (modelos, documentos, datos, reportes, formatos, etcétera), se

establecen los fundamentos, se asegura la calidad, y el cambio se maneja de manera apropiada.

- **Los métodos**: los métodos abarcan un amplio espectro de tareas que incluyen la comunicación, el análisis de requisitos, el modelado del diseño, la construcción del programa, la realización de pruebas y el soporte. Los métodos de la ingeniería del software se basan en un conjunto de principios básicos que gobiernan cada área de la tecnología e incluye actividades de modelado y otras técnicas descriptivas.

- **Las herramientas**: las herramientas de la ingeniería de software proporcionan el soporte automatizado o semi-automatizado para el proceso y los métodos. Cuando las herramientas se integran de forma que la información que cree una de ellas pueda usarla otra, se dice que se ha establecido un sistema para el soporte del desarrollo del software.

2.4.2 Objetivos de la Ingeniería de Software [6]

- Poseer la capacidad para procesar transacciones con rapidez y eficiencia.
- Mejorar la calidad de los productos de software.
- Llevar a cabo el seguimiento de los costos de mano de obra, bienes y gastos generales.
- Aumentar la productividad y trabajo de los ingenieros del software.
- Facilitar el control del proceso de desarrollo de software.

- Ampliar la comunicación y facilitar la integración de funciones individuales.
- Suministrar a los desarrolladores las bases para construir software de alta calidad en una forma eficiente.
- Definir una disciplina que garantice la producción y el mantenimiento de los productos software desarrollados en el plazo fijado y dentro del costo estimado.

2.4.3 Objetivos de la Ingeniería de Software en los Proyectos de Sistemas [6]

Para que los objetivos se cumplan las empresas emprenden proyectos por las siguientes razones:

- **Capacidad:** las actividades de la organización están influenciadas por la capacidad de ésta para procesar transacciones con rapidez y eficiencia. Los sistemas de información mejoran esta capacidad en tres formas:

✓ ***Aumentan la velocidad de procesamiento*:** Los sistemas basados en computadoras pueden ser de ayuda para eliminar la necesidad de cálculos tediosos y comparaciones repetitivas. Un sistema automatizado puede ser de gran utilidad si lo que se necesita es un procesamiento acelerado.

✓ ***Aumento en el volumen*:** La incapacidad para mantener el ritmo de procesamiento no significa el abandono de los procedimientos existentes. Quizá éstos resulten inadecuados para satisfacer las demandas actuales. En estas situaciones el analista de sistemas considera el impacto que tiene la introducción de procesamiento computarizado, si el sistema existente es manual. Es poco probable que únicamente el aumento de la velocidad sea la respuesta. El tiempo de procesamiento por transacción aumenta si se

considera la cantidad de actividades comerciales de la empresa junto con su patrón de crecimiento.

- ✓ ***Recuperación más rápida de la información*:** las organizaciones almacenan grandes cantidades de datos, por eso, debe tenerse en cuenta donde almacenarlos y como recuperarlos cuando se los necesita. Cuando un sistema se desarrolla en forma apropiada, se puede recuperar en forma rápida la información.

- **Costo:** esta capacidad mejora de la siguiente forma:
- ✓ ***Vigilancia de los costos*:** para determinar si la compañía evoluciona en la forma esperada, de acuerdo con lo presupuestado, se debe llevar a cabo el seguimiento de los costos de mano de obra, bienes y gastos generales. La creciente competitividad del mercado crea la necesidad de mejores métodos para seguir los costos y relacionarlos con la productividad individual y organizacional.
- ✓ ***Reducción de costos*:** los diseños de sistemas ayudan a disminuir los costos, ya que toman ventaja de las capacidades de cálculo automático y de recuperación de datos que están incluidos en procedimientos de programas en computadora. Muchas tareas son realizadas por programas de cómputo, lo cual deja un número muy reducido de éstas para su ejecución manual, disminuyendo al personal.

- **Control:** esta capacidad posee las siguientes funciones para mejorar el sistema:
- ✓ ***Mayor seguridad de información*:** algunas veces el hecho de que los datos puedan ser guardados en una forma adecuada para su lectura por medio de una máquina, es una seguridad difícil de alcanzar en un medio ambiente donde no existen computadoras. Para aumentar la seguridad, generalmente se

desarrollan sistemas de información automatizados. El acceso a la información puede estar controlado por un complejo sistema de contraseñas, limitado a ciertas áreas o personal, si está bien protegido, es difícil de acceder.

- ✓ ***Menor margen de error (mejora de la exactitud y la consistencia)*:** esto se puede lograr por medio del uso de procedimientos de control por lotes, tratando de que siempre se siga el mismo procedimiento. Cada paso se lleva a cabo de la misma manera, consistencia y exactitud: por otra parte se efectúan todos los pasos para cada lote de transacciones. A diferencia del ser humano, el sistema no se distrae con llamadas telefónicas, ni olvidos e interrupciones que sufre el ser humano. Si no se omiten etapas, es probable que no se produzcan errores.

- **Comunicación:** la falta de comunicación es una fuente común de dificultades que afectan tanto a cliente como a empleados. Sin embargo, los sistemas de información bien desarrollados amplían la comunicación y facilitan la integración de funciones individuales.
- ✓ ***Interconexión (aumento en la comunicación)*:** muchas empresas aumentan sus vías de comunicación por medio del desarrollo de redes para este fin, dichas vías abarcan todo el país y les permiten acelerar el flujo de información dentro de sus oficinas y otras instalaciones que no se encuentran en la misma localidad. Una de las características más importantes de los sistemas de información para oficinas es la transmisión electrónica de información, como por ejemplo, los mensajes y los documentos.
- ✓ ***Integración de áreas en las empresas*:** con frecuencia las actividades de las empresas abarcan varias áreas de la organización, la información que surge en un área se necesita en otra área, por ejemplo. Los sistemas de información ayudan a comunicar los detalles del diseño a los diferentes grupos, mantienen las especificaciones esenciales en un sitio de fácil acceso y calculan factores

tales como el estrés y el nivel de costos a partir de detalles proporcionados por otros grupos.

2.4.4 Proceso de Ingeniería de Software [6]

El proceso de Ingeniería del Software sigue etapas según las que las necesidades del usuario que luego son traducidas en requerimientos del software, estos requerimientos son transformados en diseño y el diseño es implementado en código, el código seguidamente es probado, documentado y certificado para uso operativo. Concretamente define quién está haciendo qué, cuándo hacerlo y cómo alcanzar un cierto objetivo. A este proceso también se le llama el ciclo de vida del software que comprende cuatro grandes fases:

- **Concepción**: define el alcance del proyecto y desarrolla un caso de negocio.
- **Elaboración**: define un plan del proyecto, especifica las características y fundamenta la arquitectura.
- **Construcción**: crea el producto.
- **Transición**: transfiere el producto a los usuarios.

2.4.5 Fases de la Ingeniería de Software [6]

La ingeniería de software está dividida en las siguientes fases:

- **Análisis y Especificación de los Requerimientos:** esta fase consiste en analizar, entender y registrar el problema que el patrocinante está tratando de resolver. Los requerimientos son una descripción de las necesidades o deseos de un producto. Un conjunto de requerimientos en estado de madurez, debe

presentar una serie de características tanto individualmente como en grupo, así se tiene que un requerimiento debe ser:

- ✓ ***Necesario*:** si su omisión provoca una deficiencia en el sistema a construir, y además su capacidad, características físicas o factor de calidad no pueden ser reemplazados por otras capacidades del producto o proceso.
- ✓ ***Conciso*:** si es fácil de leer y entender. Su redacción debe ser simple y clara para aquellos que vayan a consultarlo en un futuro.
- ✓ ***Completo*:** si no necesita ampliar detalles en su redacción, es decir, si se proporciona la información suficiente para su comprensión.
- ✓ ***Consistente*:** si no es contradictorio con otro requerimiento.
- ✓ ***No* ambiguo:** cuando tiene una sola interpretación.

- **Diseño:** consiste en crear una solución que satisfaga las especificaciones definidas en la fase de análisis. En esta fase se crea la interfaz de usuario y la estructura del software.

- **Codificación:** en esta fase se lleva a cabo la implementación física de las bases de datos, de los programas (codificación), prueba de unidad o procedimientos separados y prueba de subsistemas o integración.

- **Prueba del Sistema:** los grupos de procedimientos de unidades probadas como elementos separados en la fase previa son comprobados como sistema durante la integración.

- **Instalación:** corresponde a la fase de puesta en marcha de la aplicación.

- **Mantenimiento del Sistema y Mejoras:** esta fase se corresponde al mantenimiento y mejoras futuras que deben contemplarse a la hora de

implantar una aplicación informática, una vez que el sistema está en funcionamiento**.**

2.5 Programación orientada a objetos

2.5.1 Definición [7]

La programación orientada a objeto (POO) tiene la ventaja de ser un paradigma natural donde se pueden programar sistemas. Los seres humanos perciben el mundo como si estuviera formado por objetos: mesas, sillas, computadoras, coches, cuentas bancarias, partidos de fútbol, etc. También es un instinto humano intentar organizar estos objetos disponiéndolos de una forma concreta, optando por destacar determinadas características de algunos objetos que los destacan de otros.

Los niveles de dichas categorías y los métodos de clasificación de objetos en el mundo son infinitos. La forma en que las personas clasifican las cosas depende, en gran medida, de lo que deseen hacer con ellas y las características que más les llamen la atención. A la vez que se agrupan los objetos atendiendo a esquemas de clasificación, también se tiende a resaltar determinados atributos de objetos mostrando su preferencia sobre otros.

Esta idea de crear jerarquías de objetos relacionados se utiliza en la programación orientada a objetos. En 1960, los investigadores ya observaron que muchas de las entidades del modelo de programas de computadoras se podían nombrar y que se podían describir sus propiedades y comportamiento. Se dieron cuenta de que los programas estaban relacionados con cuentas bancarias, matrices, archivos y usuarios; en definitiva, algo parecido a los objetos en el mundo real.

La programación orientada a objetos se puede describir rápidamente como la identificación de objetos importantes, su organización en jerarquías, la adición de atributos a los objetos que describen las características relevantes en el contexto del problema y de la adición de las funciones (métodos) a los objetos para realizar las tareas necesarias en el objeto. Los detalles son algo más complejo, pero lo fundamental es que se trata de un proceso simple y natural.

No obstante, que sea un proceso simple y natural no significa que sea sencillo, ya que un conjunto de objetos se podría clasificar de muchas formas distintas. La clave es la posibilidad de identificar los atributos importantes de objetos y formar abstracciones y jerarquías idóneas. Incluso en el contexto de un dominio problemático, a veces resulta bastante difícil determinar los niveles correctos de abstracción y jerarquías de clasificación correctas. Simplemente, la decisión de la clase o grupo al que un objeto pertenece puede ser una tarea bastante difícil.

2.5.2 Características de los Lenguajes Programación Orientada a Objetos [7]

Los lenguajes de programación orientada a objetos (como C++, C# y Java) se caracterizan por tres conceptos clave: encapsulación, herencia y polimorfismo, que son compatibles con este aspecto natural de identificación y clasificación de objetos.

2.5.2.1 Encapsulación

La encapsulación facilita la compresión de los grandes programas; la ocultación de datos les hace más eficaces. Los objetos pueden interactuar con otros objetos sólo a través de los atributos y métodos del objeto que se muestran públicamente. Cuantos más atributos y métodos se muestren públicamente, más difícil será modificar la clase sin que ello afecte al código que utiliza la clase. Una variable oculta se podría cambiar de un long a un double, sin que ello afecte al

código que utilicen los objetos creados (instanciados) de esa clase. El programador solo se debería preocupar por los métodos en la clase que han tenido acceso a esa clase, en lugar de por todos los sitios en el programa que un objeto ha instanciado desde los que se puede llamar a la clase.

2.5.2.2 Herencia

Proporciona dos ventajas evidentes a los programadores. La primera, y más importante, es que permite crear jerarquías que expresen las relaciones entre los diferentes tipos. Imagine que tiene dos clases, *CuentaAhorro* y *CuentaCorriente*, que proceden de la clase principal Cuenta. Si tiene una función que necesite una clase Cuenta como argumento, podrá pasarle una *CuentaAhorro* o una *CuentaCorriente*, ya que ambas clases son de tipos de Cuenta. Cuenta es una clasificación general, mientras que *CuentaCorriente* y *CuentaAhorro* son tipos más específicos.

La segunda ventaja es que las clases pueden heredar características más generales de las clases superiores dentro de la jerarquía. En lugar de desarrollar nuevas clases a partir de cero, las clases nuevas podrán heredar las funciones de las clases existentes y, a continuación, modificar o ampliar esta función. La clase principal desde la que la nueva clase hereda las propiedades se conoce como la clase básica y la nueva clase se denomina la clase derivada.

2.5.2.3 Polimorfismo

Polimorfismo significa, fundamentalmente, que las clases pueden tener el mismo comportamiento, pero implementarse de distintas maneras. Esto resulta muy útil en términos de programación, ya que permite trabajar con tipos de objetos genéricos cuando lo que interesa no es cómo implementa cada clase las funciones.

2.5.3 Clases y Objetos [7]

Como su propio nombre lo indica, la programación orientada a objetos está relacionada con objetos. Un objeto se compone de datos que describen al objeto y las operaciones que se pueden realizar en el objeto. No obstante, cuando se crea un programa, en realidad se declaran y definen clases, no objetos.

Una clase es un tipo definido por el usuario, encapsula tanto los datos como los métodos que funcionen en esos datos. Una clase es algo muy parecido a una plantilla, que se utiliza para crear (instanciar) objetos.

2.5.4 Ventajas de la Programación Orientada a Objetos en el Ciclo de Vida de Desarrollo del Software [7]

Son tres las ventajas clave de la programación orientada a objetos: compresión, reutilización del código y posibilidad de ampliación. La división del código en clases contribuye a imponer una estructura a medida que crecen los programas. La idea es unir los programas orientados a objetos de clases escritas previamente, así como realizar las modificaciones necesarias para admitir los nuevos requisitos mediante la herencia para derivar nuevas clases a partir de las ya existentes. La creación de sistemas a partir de componentes que se pueden volver a utilizar conduce, naturalmente, a una mayor productividad, que suele ser la ventaja más citada de todos los métodos orientados a objetos.

Las características de la programación orientada a objetos también proporcionan varias ventajas. La encapsulación facilita la escala de pequeños sistemas a grandes sistemas. En buena medida, independientemente del tamaño del sistema, el programador simplemente está creando objetos.

Por último, la posibilidad de ocultar datos también genera sistemas más seguros. El estado de objetos sólo se puede modificar mediante métodos expuestos públicamente; esto aumenta la posibilidad de predecir el comportamiento del objeto.

2.6 Proceso unificado de desarrollo de software

2.6.1 Definición [8]

El proceso Unificado es un proceso de desarrollo de software. Un proceso de desarrollo de software es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema de software. Sin embargo, el proceso Unificado es más que un simple proceso; es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyecto.

El proceso Unificado está basado en componentes, lo cual quiere decir que el sistema software en construcción está formado por componentes de software interconectados a través de interfaces bien definidas.

El proceso Unificado utiliza el Lenguaje Unificado de Modelado (UML) para preparar todos los esquemas de un sistema software; de hecho UML es una parte esencial del proceso Unificado**.**

El proceso unificado de desarrollo nace en vista de la necesidad de crear un estándar para el desarrollo de software con la finalidad de satisfacer la creciente

demanda de sistemas más grandes, modernos y poderosos y que a su vez permita disminuir el tiempo de desarrollo de los mismos. Además este proceso permite minimizar los riesgos de tener que codificar componentes que no serán utilizados correctamente o simplemente no son terminados. A través de la historia se han desarrollado varios modelos de proceso de software (paradigmas de desarrollo) cada uno con sus ventajas, desventajas y utilidad en algunos tipos de proyectos y problemas. Al igual que cualquier notación, el proceso unificado actúa como un modelo que puede adaptarse a cualquier tipo de proyecto (grande y pequeño).

2.6.2 Principios Básicos del Proceso Unificado de Desarrollo de Software [8]

Los verdaderos aspectos definitorios del Proceso Unificado se resumen en tres frases claves: dirigidos por casos de uso, centrados en la arquitectura, iterativo e incremental. Esto es lo que hace único al proceso Unificado. A continuación se explican los principios básicos del Proceso Unificado de Desarrollo de Software:

2.6.2.1 Proceso Unificado Dirigido por Casos de Uso

Cuando un usuario utiliza el sistema, se lleva a cabo una interacción que consiste en una secuencia de acciones (tanto por parte del usuario como del sistema) que proporcionan al usuario un resultado de valor importante para él. Un caso de uso es precisamente una interacción de ese tipo, que deriva en que el sistema debe incluir un fragmento de funcionalidad para proporcionar al usuario el resultado de valor esperado.

Por tanto, los casos de uso representan los requisitos funcionales del sistema, es decir, las necesidades de los usuarios y de la organización en donde se desplegará

el sistema. Dicho con otras palabras, el modelo de casos de uso responde a la pregunta ¿qué debe hacer el sistema para satisfacer a sus usuarios?

De nada sirve desarrollar un sistema muy avanzado tecnológicamente pero que no cumple con las expectativas de los usuarios; un producto software se crea para dar servicio a sus usuarios. Durante todo el desarrollo hay que tener en cuenta los casos de uso, esto es, los casos de uso guían todo el proceso de desarrollo. La forma en que esto se lleva a cabo es desarrollando el sistema incrementalmente, añadiendo poco a poco fragmentos de funcionalidad para realizar un conjunto de casos de uso, y demostrando la validez del sistema, comprobando que la funcionalidad desarrollada realmente realiza esos casos de uso. En definitiva, los casos de uso proporcionan el hilo conductor para avanzar en el desarrollo del software.

La captura de requisitos tiene dos objetivos: Encontrar los verdaderos requisitos y representarlos de un modo adecuado para los usuarios, clientes y desarrolladores. Los verdaderos requisitos son aquellos que cuando se implementen añadirán el valor esperado para los usuarios. Normalmente un sistema tiene muchos usuarios. Cada tipo de usuarios se representa por un actor. Los actores utilizan el sistema interactuando con los casos de usos. El modelo de casos de usos está compuesto por todos los actores y casos de uso de un sistema.

2.6.2.2 Un Proceso Centrado en la Arquitectura

La arquitectura del software sirve para poder contemplar el futuro sistema desde varios puntos de vista antes de que se construya y durante la construcción El concepto de arquitectura software incluye los aspectos estáticos y dinámicos más significativos del sistema, es decir, especifica la forma, estructura y comportamiento del sistema desde diversos puntos de vista, todos ellos a un nivel de detalle que

permita tener una idea global clara sin perderse en detalles insignificantes (que, precisamente, no influyen en la arquitectura del sistema).

De esta forma quedan cubiertos los dos aspectos fundamentales del sistema: ¿Qué hace el sistema para cada usuario? (casos de uso); ¿Cómo es el sistema y cómo funciona para realizar esos casos de uso? (arquitectura de software). Ambas especificaciones se complementan y evolucionan en paralelo. Por un lado, los casos de uso deben encajar en la arquitectura del sistema cuando se lleven a cabo (adaptar los casos de uso a lo que la arquitectura permite); por otro lado, la arquitectura debe permitir la realización de todos los casos de uso (adaptar la arquitectura a lo requerido por los casos de uso).

En la práctica, la arquitectura se diseña para permitir realizar los casos de uso, pero pueden existir limitaciones externas que obliguen a redefinir el modelo de casos de uso, dando lugar a un ciclo de realimentación. Pero existen otros parámetros que influyen en la arquitectura: la plataforma y tecnología de destino, componentes reutilizables disponibles, sistemas heredados y requisitos no funcionales (ver figura 2.4).

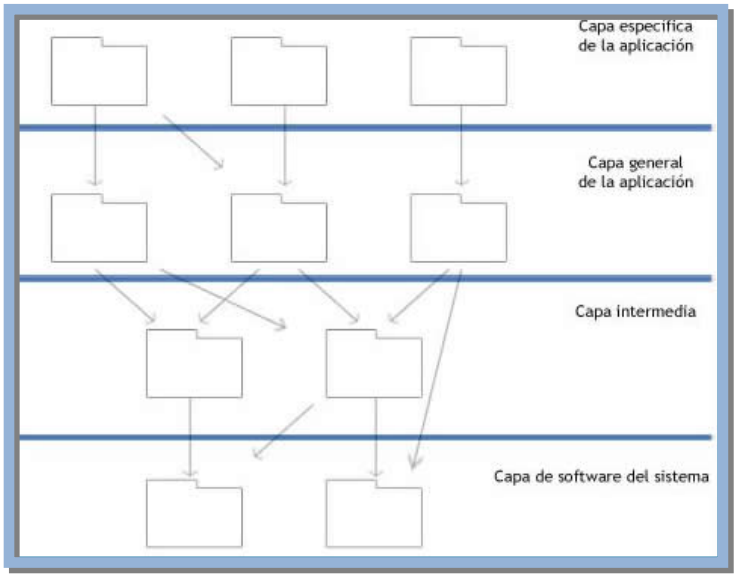


Figura 2.4. Proceso Centrado en la Arquitectura

2.6.3 Vida del Proceso Unificado [8]

El proceso unificado se repite a lo largo de una serie de ciclos que constituyen la vida de un sistema. Cada ciclo consta de cuatro fases: inicio, elaboración, construcción y transición.

En la figura 2.5 se puede visualizar las diferentes actividades: Requisitos, Análisis, Diseño, Implementación y Prueba que tienen lugar sobre las 4 fases: Inicio, Elaboración, Construcción y Transición.

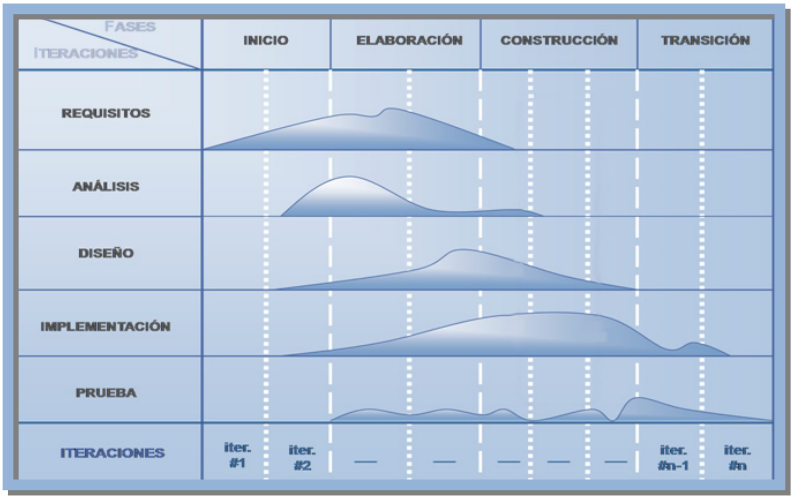


Figura 2.5. Las actividades: Requisitos, Análisis, Diseño, Implementación y Prueba tienen lugar sobre las 4 fases: Inicio, Elaboración, Construcción y Transición.

Cada ciclo produce una nueva versión del sistema, y cada versión es un producto preparado para su entrega; consta de un cuerpo de código fuente incluido en componentes que puede compilarse y ejecutarse, además de manuales y otros

productos asociados. Sin embargo, el producto terminado no sólo debe ajustarse a las necesidades de los usuarios, sino también a las de todos los interesados, es decir, toda la gente que trabajará con el producto. El producto terminado incluye los requisitos, casos de uso, especificaciones no funcionales y casos de prueba. Incluye el modelo de la arquitectura y el modelo visual (artefactos modelados con el UML).

2.6.4 Descripción de las fases del proceso unificado de desarrollo del software [8]

2.6.4.1 Fase de Inicio

En esta fase se desarrolla una descripción del producto final a partir de una buena idea y se presenta el análisis de negocio para el producto. Esencialmente esta fase responde a las siguientes preguntas:

1. ¿Cuáles son las principales funciones del sistema para sus usuarios más importantes?
2. ¿Cómo podría ser la arquitectura del sistema?
3. ¿Cuál es el plan de proyecto y cuánto costará desarrollar el producto?

La respuesta a la primera pregunta se encuentra en un modelo de casos de uso simplificado que contenga los casos de uso más críticos. Una vez obtenidos, la arquitectura es provisional, y consiste típicamente en un simple esbozo que muestra los subsistemas más importantes. En esta fase, se identifican y priorizan los riesgos más importantes, se planifica en detalle la fase de elaboración, y se estima el proyecto de manera aproximada.

2.6.4.2 Fase de Elaboración

Durante esta fase se especifican en detalle la mayoría de los casos de uso del producto y se diseña la arquitectura del sistema. La relación entre la arquitectura del sistema y el propio sistema es primordial. Por tanto la arquitectura se expresa en forma de vistas de todos los modelos del sistema, los cuales juntos representan al sistema entero. Esto implica que hay vistas arquitectónicas del modelo de casos de uso, del modelo de análisis, del modelo de diseño, del modelo de implementación y modelo de despliegue. La vista del modelo de implementación incluye componentes para probar que la arquitectura es ejecutable. Durante esta fase del desarrollo, se realizan los casos de uso más críticos que se identificaron en la fase de inicio. El resultado de esta base es la línea base de la arquitectura.

2.6.4.3 Fase de Construcción

En esta fase se crea el producto, la línea base de la arquitectura crece hasta convertirse en el sistema completo. La descripción evoluciona hasta convertirse en un producto preparado para ser entregado a la comunidad de usuarios. El grueso de los recursos requeridos se emplea durante esta fase del desarrollo. Sin embargo la arquitectura del sistema es estable, aunque se pueden describir formas mejores de estructurar el sistema. Al final de esta fase, el producto contiene todos los casos de uso que la dirección y el cliente han acordado para el desarrollo de esta versión. Sin embargo, puede que no esté completamente libre de defectos. Muchos de estos defectos se descubrirán y solucionarán durante la fase de transición.

2.6.4.4 Fase de Transición

Durante esta fase se cubre el período durante el cual el producto se convierte en versión beta. En la versión beta un número reducido de usuarios con experiencia prueba el producto e informa de defectos y deficiencias. Esta fase conlleva

actividades como la fabricación, formación del cliente, el proporcionar una línea de ayuda y asistencia, y la corrección de los defectos que se encuentren tras la entrega.

2.6.5 Flujos del Proceso Unificado de Desarrollo de Software [8]

Los flujos fundamentales: Requisitos, Análisis, Diseño, Implementación y Pruebas, se distinguen entre los flujos de trabajos fundamentales e iterativos. Dichos flujos no ocurren una sola vez, en el Proceso Unificado de Modelado, como sucede teóricamente en el modelo en cascada; sino que se repiten más bien en cada iteración, una y otra vez, como flujos de trabajos iterativos. Cada repetición, sin embargo, se diferencia en los detalles que se enfrentan o asuntos centrales de cada iteración.

2.6.5.1 Requisitos

El esfuerzo principal en la fase de requisitos es desarrollar un modelo del sistema que se va a construir, y la utilización de los casos de uso en una forma adecuada de crear este modelo. Esto es debido a que los requisitos funcionales se estructuran de forma natural mediante casos de uso, y a que la mayoría de los otros requisitos no funcionales son específicos de un solo caso de uso, y pueden tratarse en el contexto de ese caso de uso. Así, los requisitos de un sistema se capturan en forma de:

- Un modelo del negocio o un modelo del dominio para establecer el contexto del sistema.
- Un modelo de casos de uso que capture los requisitos funcionales, y los no funcionales que son específicos de casos de uso concretos. El modelo de casos de uso se realiza mediante una descripción general, un conjunto de diagramas, y una descripción detallada de cada caso de uso.

- Un conjunto de esbozos de interfaces de usuario y de prototipos de cada actor, que
- representan el diseño de las interfaces de usuario.
- Una especificación de requisitos adicionales para los requisitos que son genéricos y no específicos de un caso de uso en particular.

Estos resultados son un buen punto de partida para los siguientes flujos de trabajos: análisis, diseño, implementación y prueba. Los casos de uso dirigirán el trabajo a lo largo de estos flujos de trabajo iteración por iteración. Para cada caso de uso del modelo de casos de uso identificaremos una realización de caso de uso correspondiente en las fases de análisis y diseño y un conjunto de casos de prueba en la fase de pruebas. Por tanto, los casos de uso enlazaran directamente los diferentes flujos de trabajo.

2.6.5.2 Análisis

Durante el análisis, estudiamos los requisitos que se describen en la captura de requisitos, refinándolos y estructurándolos. El resultado del flujo de trabajo del análisis es el modelo de análisis, que es un modelo del objeto conceptual que analiza los requisitos mediante su refinamiento y estructuración. El modelo de análisis incluye los siguientes elementos:

- Paquetes del análisis y paquetes de servicio, y sus dependencias y contenidos. Los paquetes de análisis pueden aislar los cambios en un proceso del negocio, el comportamiento de un actor, o en conjunto de casos de uso estrechamente relacionados. Los paquetes de servicios aislarán los cambios en determinados servicios ofrecidos por el sistema, y construyen un elemento esencial para construir pensando en la reutilización durante el análisis.

- Clases de análisis, sus responsabilidades, atributos, relaciones y requisitos especiales. Cada una de las clases de control, entidad e interfaz aislaran los cambios al comportamiento y la información que representan. Un cambio en la interfaz de usuario o en una interfaz de comunicación normalmente se ubica en una o más clases de entidad; un cambio en el control, coordinación, secuencia, transacciones y a veces en la lógica del negocio compleja, que implica a varios objetos (de interfaz y/o de entidad) normalmente se ubica en una o más clases de control.

- Realizaciones de casos de uso-análisis que describen como se refinan los casos de uso en términos de colaboraciones dentro del modelo de análisis y de sus requisitos especiales. La realización de casos de uso aislaran os cambios en los casos de uso, debido a que si cambia un caso de uso, debe cambiarse también su realización.

- La vista arquitectura del modelo de análisis, incluyendo sus elementos significativos para la arquitectura. La vista de la arquitectura aislara los cambios de la arquitectura.

- El modelo de análisis se considera la entrada fundamental para las actividades de diseño subsiguientes. Cuando utilizamos el modelo de análisis con esa intención, conservamos en todo lo posible la estructura que define durante el diseño del sistema, mediante el tratamiento de la mayor parte de los requisitos no funcionales y otras restricciones relativas al entorno de la implementación. Más en concreto, el modelo de análisis influirá en el modelo de diseño de las siguientes maneras:

- ✓ Los paquetes del análisis y los paquetes de servicios tendrán una influencia fundamental en los subsistemas de diseño y en los subsistemas de servicios, respectivamente, en las capas específicas y generales de la aplicación. En muchos casos tendremos una traza uno a uno entre paquetes y los correspondientes subsistemas.
- ✓ Las clases del análisis servirán como especificaciones al diseñar las clases. Se requieren diferentes tecnologías y habilidades al diseñar clases del análisis con diferentes estereotipos: por ejemplo, el diseño de las clases de entidad normalmente requiere el uso de tecnologías de bases de datos, mientras que el diseño de clases de interfaz normalmente requiere el uso de tecnologías de interfaz de usuario. Sin embargo, las clases del análisis y sus responsabilidades atributos y relaciones sirven como una entrada lógica par la creación de las correspondientes operaciones, atributos y relaciones de las clases de diseño. Además, la mayoría de los requisitos especiales recogidos sobre una clase del análisis serán tratados por las clases de diseño correspondientes cuando se tienen en cuenta tecnologías como las de bases de datos y de interfaces de usuarios.
- ✓ Las realizaciones de casos de uso análisis tienen dos objetivos principales. Uno es ayudar a crear especificaciones más precisas para el caso de uso. En lugar de detallar cada caso en el modelo de casos de uso con diagramas de estado o diagramas de actividad. La descripción de un caso de uso mediante una colaboración entre clases del análisis da como resultado una especificación formal completa de los requisitos del sistema. Las realizaciones de casos de uso-análisis también sirven como entrada al diseño de los casos de uso.

La vista de la arquitectura del modelo de análisis se utiliza como entrada en la creación de la vista de la arquitectura del modelo de diseño. Es muy probable que los elementos de las diferentes vistas (de los diferentes modelos) tengas trazas entre ellos. Esto es debido a que la relación de relevancia para la arquitectura tiende a fluir suavemente a lo largo de los diferentes modelos mediante diferencias de traza.

2.6.5.3 Diseño

En el diseño se modela el sistema y se encuentra la forma para que soporte los requisitos que se suponen. El principal resultado del diseño es el modelo de diseño. Se esfuerza en conservar la estructura del sistema impuesta por el modelo de análisis, y que sirve como esquema para la implementación. El modelo de diseño incluye los siguientes elementos:

- Subsistemas del diseño y subsistemas de servicio y sus dependencias, interfaces y contenidos. Los subsistemas del diseño de las dos capas superiores (las capas específicas de la aplicación) se obtienen a partir de los paquetes del análisis algunas de las dependencias entre subsistemas del diseño se obtienen a partir de las correspondientes dependencias entre paquetes del análisis. Algunas de las interfaces se obtienen a partir de las clases del análisis.
- Clases del diseño, incluyendo las clases activas, y sus operaciones, atributos y requisitos de implementación. Algunas clases del diseño relevantes para la arquitectura se obtienen a partir de las clases del análisis relevante para la arquitectura. Algunas clases activas se obtienen a partir de clases del análisis se utilizan como especificaciones al obtener las clases de diseño.
- Realizaciones de casos de uso-diseño, que describen como se diseñan los casos de uso en términos de colaboraciones dentro del modelo de diseño. En general, al obtener las realizaciones de caso de uso-diseño utilizamos las realizaciones de caso de uso-análisis como especificaciones.

- La vista arquitectónica del modelo de diseño, incluyendo esos elementos significativos de cara a la arquitectura.

2.6.5.4 Implementación

El resultado principal de la implementación es el modelo de la implementación, el cual incluye los siguientes elementos:

- Subsistemas de implementación y sus dependencias, interfaces y contenidos.
- Componentes, incluyendo componentes ficheros y ejecutables, y las dependencia entre ellos. Los componentes son sometidos a pruebas de unidad.
- Las vista de la arquitectura del modo de implementación, incluyendo sus elementos arquitectónicamente significativos.

La implementación también produce como resultado un refinamiento de la vista de la arquitectura de modelo de despliegue, donde los componentes ejecutables son asignados a nodos. El modelo de implementación es la entrada principal de las etapas de pruebas que siguen a la implementación, más concretamente, durante la etapa de prueba cada construcción generada durante la implementación es sometida a pruebas de implementación, y posiblemente también a pruebas del sistema.

2.6.5.5 Pruebas

El resultado principal de la prueba es el modelo de prueba, el cual describe como ha sido probado el sistema. El modelo reprueba incluye:

- Casos de prueba, que especifican que probar en sistema.
- Procedimientos de prueba, que especifican como realizar los casos de pruebas.
- Componentes de pruebas, que automatizan los procedimientos de pruebas.

La prueba da también como resultado un plan de prueba, evaluaciones de las pruebas realizadas y los defectos que pueden ser pasados como entrada a flujos de trabajos anteriores, como el diseño y la implementación.

2.6.6 Flujos de Trabajos para una Iteración [8]

Cada iteración está formada por cinco grupos de flujos de trabajos fundamentales, estos flujos se adaptan según sea la fase para la cual se está desarrollando. En la figura 2.6, se observa cómo, para una iteración de cualquier fase, coexisten los cinco flujos de trabajos.

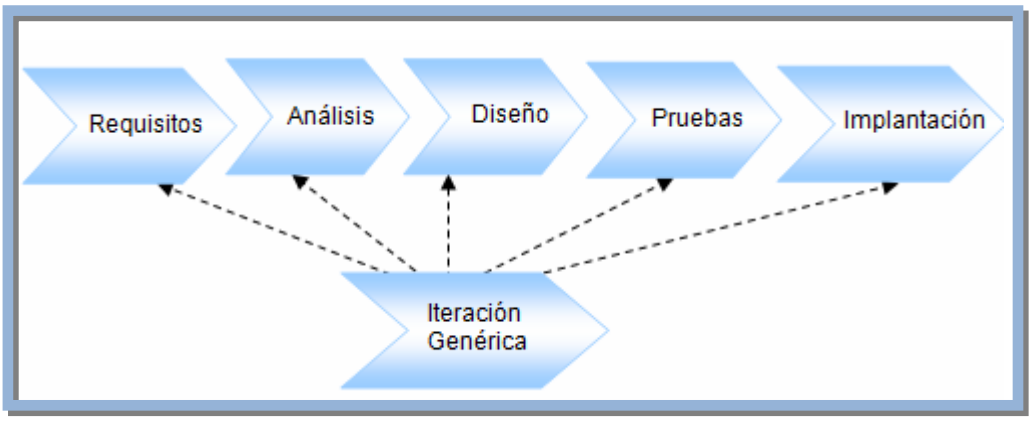


Figura 2.6. Los cinco (5) flujos de trabajo Requisitos, Análisis, Diseño, Pruebas e Implantación.

2.6.7 Ventajas del Proceso Unificado de Desarrollo de Software [8]

- La Iteración controlada acelera el ritmo de esfuerzo de desarrollo en su totalidad debido a que los desarrolladores trabajan de manera más eficiente para obtener resultados claros a corto plazo.
- La iteración controlada reconoce una realidad que a menudo ignora que las necesidades del usuario y sus correspondientes requisitos no pueden definirse completamente al principio. Mediante proceso unificado racional los requisitos se adquieren y refinan en sucesivas iteraciones.
- En cada fase del proceso unificado se genera un producto; así mismo el desarrollo de cada fase puede contener algunos de los siguientes modelos: requisitos, análisis, diseño, implementación, despliegue y pruebas.
- Los modelos son ideales porque mediante éstos se representa el trabajo evolutivo que desarrollan los ingenieros de software; y no incluye un producto final, como lo es la versión Beta del sistema; es decir no se requiere esperar la culminación del trabajo para observar un producto.
- Los modelos juntos representan al sistema como un todo.
- El proceso unificado está basado en componentes; y utiliza el estándar del lenguaje unificado de modelado para llevar a cabo sus modelos.

2.6.8 Desventaja del Proceso Unificado de Desarrollo de Software [8]

A pesar de sus características de adaptabilidad y flexibilidad, el Proceso Unificado no permite representar en su totalidad los elementos representativos de los agentes, los cuales son: habilidad social, autonomía, reactividad, pro-actividad, nociones mentales, adaptabilidad o aprendizaje, veracidad, racionalidad**.**

2.7 LENGUAJE UNIFICADO DE MODELADO (UML)

2.7.1 Definición [9]

El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual de propósito general que es usado para especificar, visualizar, construir y documentar los artefactos de un sistema software (Jacobson I., Booch G. y Rumbaugh J, 2000). El UML captura información acerca de la estructura estática y el comportamiento dinámico de un sistema. Un sistema es modelado como una colección de objetos discretos que interactúan para desempeñar un trabajo que en última instancia beneficia a un usuario externo. La estructura estática define el tipo de objetos importantes para un sistema y su implementación, así como la relación entre los objetos. El comportamiento dinámico define la historia de los objetos sobre el tiempo y la comunicación entre objetos para cumplir metas.

Partiendo del hecho que las diferencias entre los métodos disminuyen y que la guerra de métodos no hace progresar la tecnología de objetos, Jim Rumbaugh y Grady Booh decidieron a finales de 1.994, unificar sus trabajos en un método único: El Método Unificado (The Unified Method). Un año más tarde se les unió Ivan Jacobson. Los tres autores fijaron cuatro objetivos:

1) Representar sistemas completos (más allá de un solo programa) por conceptos de objetos.
2) Establecer una relación explícita entre los conceptos y los artefactos ejecutables.
3) Tener en cuenta los factores de escala inherentes a los sistemas complejos y críticos.
4) Crear un lenguaje de modelado utilizable tanto por los humanos como por las máquinas.

2.7.2 Modelos [9]

Un modelo es la representación en un cierto medio de algo en el mismo u otro medio. El modelo captura los aspectos importantes del ente que será modelado desde un cierto punto de vista, simplificando u omitiendo el resto.

- **Modelo de Casos de Uso:** El modelo de casos de uso permite que los desarrolladores de software y los clientes lleguen a un acuerdo sobre los requisitos. Contiene actores, casos de uso y sus relaciones.
- **Modelo de Análisis:** El modelo de análisis se representa mediante un sistema de análisis que denota el paquete de más alto nivel del modelo. La utilización de otros paquetes de análisis es por tanto una forma de organizar el modelo de análisis en partes más manejables que representan abstracciones de subsistemas y posiblemente capas completas del diseño del sistema.
- **Modelo de Diseño:** El modelo de diseño es un modelo de objetos que describe la realización física de los casos de usos centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar. Además, el modelo de diseño sirve de abstracción de la implementación del sistema y es, de ese modo, utilizada como una entrada fundamental de las actividades de implementación.
- **Modelo de Despliegue:** El modelo de despliegue es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos del cómputo. El modelo de despliegue se utiliza como entrada fundamental en las actividades de diseño e implementación debido a que la distribución del sistema tiene una influencia principal en su diseño.
- **Modelo de Implementación:** El modelo de implementación describe cómo los elementos del modelo de diseño, como las clases, se implementan en términos de componentes, como ficheros de código fuente, ejecutables, entre

otros. 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.

- **Modelo de Prueba:** El modelo de prueba describe principalmente cómo se prueban los componentes ejecutables (como las construcciones) en el modelo de implementación con pruebas de integración y de sistema. El modelo de pruebas puede describir también cómo han de ser probados aspectos específicos del sistema.

2.7.3 Diagramas [9]

Un diagrama es la representación gráfica de un conjunto de elementos, usualmente representado como un grafo conectado de vértices (elementos) y arcos (relaciones).

- **Diagramas de Casos de Usos:** Un diagrama de caso de uso es un diagrama que muestra un conjunto de casos de uso, actores y sus relaciones. Los mismos sirven para especificar la funcionalidad y el comportamiento de un sistema mediante su interacción con los usuarios y/u otros sistemas. O lo que es igual, un diagrama que muestra la relación entre los actores y los casos de uso dentro de un sistema. A continuación se explican los elementos del diagrama de casos de usos con detalle:
  - ✓ ***Casos de Usos:*** Un caso de uso describe un conjunto de secuencias, donde cada una representa la interacción de los elementos externos al sistema (Actores) con el mismo sistema. Un caso de uso representa un requisito funcional de un sistema; por ejemplo un caso de uso imprescindible para

un colegio es la inscripción de sus alumnos. Los casos de uso son representados por una elipse que contiene el nombre del caso de uso. El nombre es una cadena de texto y cada caso de uso debe tener uno diferente que lo identifique y distinga de otros casos de uso.

- ✓ ***Actor (es):*** Un actor es la representación de un conjunto coherente de roles que los usuarios de los casos de uso juegan cuando interactúan con estos. Por lo general, representan el papel desempeñado por una persona, un dispositivo, un objeto e incluso otro sistema que interactúa con el sistema propuesto. Debido a que los actores representan usuarios del sistema, ellos ayudan a delimitar el sistema y proporcionan un panorama más claro de lo que el sistema debe hacer. Los casos de uso son desarrollados de acuerdo a las necesidades de los actores. Un actor puede ser representado con un rectángulo con el estereotipo actor. El estereotipo estándar es un icono representado por una persona dibujada con líneas. Existen varias relaciones estándares entre casos de uso o entre actores y casos de uso: asociación, generalización y uso.
- ✓ ***Relación de Dependencia:*** Es una conexión de uso que indica que cualquier cambio en un elemento puede afectar a otro elemento que la utiliza, pero no necesariamente de modo inverso. Esta es representada por una flecha, de línea no continua, orientada hacia el elemento del que se depende.
- ✓ ***Relación de Generalización:*** Es la relación que existe entre un elemento general y un caso específico de ese mismo elemento. La generalización significa que los objetos hijos se pueden emplear en cualquier lugar donde pueda aparecer el padre, más no a la inversa. Esta relación se representa por una flecha continua con punta vacía orientada hacia el padre.

- ✓ ***Relación de Asociación:*** Se entiende por la relación estructural que indica que los objetos de un elemento están conectados con los objetos de otro. Una asociación se representa por una línea continua que conecta las clases.

Durante el análisis del sistema, los diagramas de casos de uso son utilizados para capturar los requerimientos del sistema y entender que es lo que el sistema supuestamente hace.

Durante el diseño del sistema, los diagramas de casos de uso definen el comportamiento del sistema implementado y especifica la semántica de un mecanismo. En la siguiente figura 2.7 se muestran los componentes de los diagramas de casos de uso.

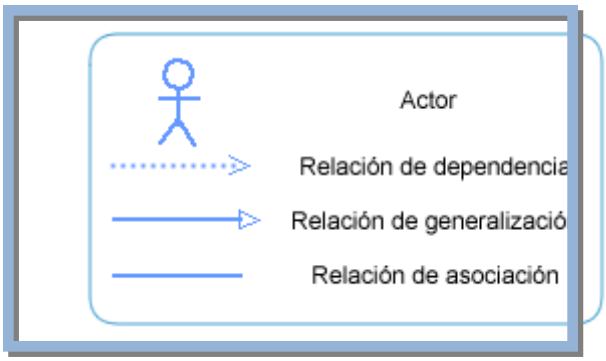


Figura 2.7. Elementos de diagramas de casos de uso (Jacobson I., Booch G, Rumbaugh J, 2000).

- **Diagramas de Clases:** Es un diagrama que muestra un conjunto de clases, interfaces y colaboraciones y las relaciones entre éstos. Los diagramas de clases muestran el diseño de un sistema desde un punto de vista estático. También se puede decir que es un diagrama que muestra una colección de elementos (estáticos) declarativos.

Un diagrama de clases también puede contener interfaces, paquetes, relaciones e instancias y se representa gráficamente mediante una colección de nodos y arcos. En la siguiente figura se muestran los elementos de un diagrama de clases, en la figura 2.8 se muestran los elementos de los diagramas de clases y objetos.

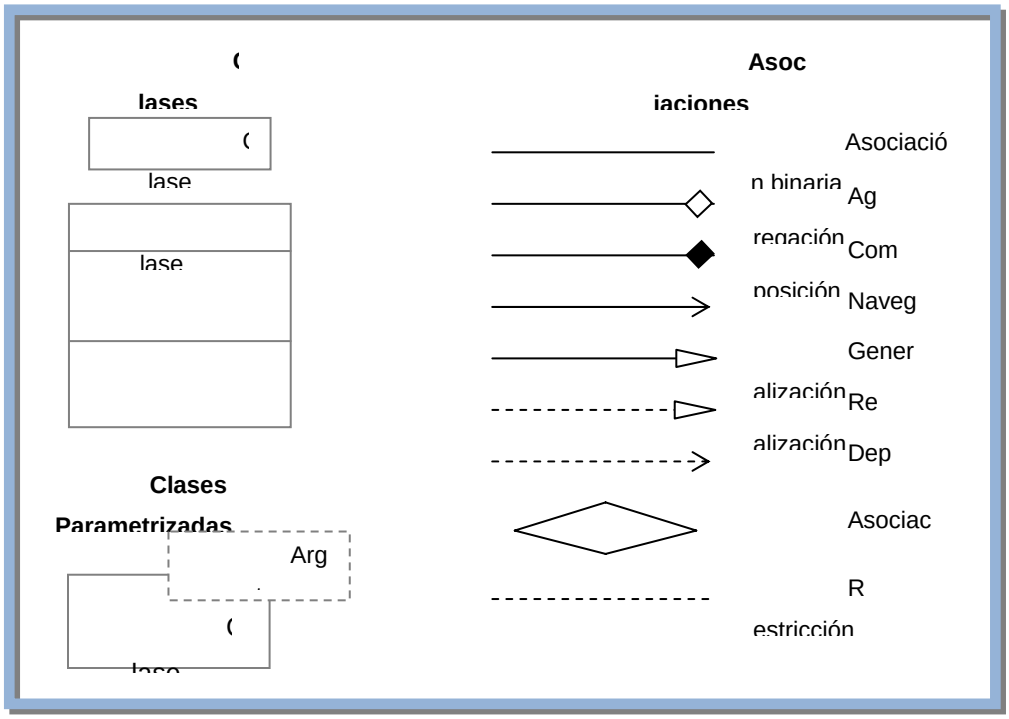


Figura 2.8. Elementos del Diagrama de Clases y Objetos (Jacobson I., Booch G, Rumbaugh J, 2000).

Por lo general, un diagrama de clases está compuesto por: clases, interfaces, colaboraciones y relaciones de dependencia, generalización y asociación.

- ***Clase:*** Una clase describe un conjunto de objetos con estructura similar, su comportamiento y sus relaciones. Representa un concepto dentro del sistema a ser modelado; las clases tienen estructura de datos, comportamiento y relaciones con otros elementos. Una clase es dibujada como un rectángulo sólido con tres compartimentos separados por líneas horizontales. El primer compartimiento contiene el nombre de la clase y otras propiedades generales de la clase. El segundo compartimiento contiene los atributos de la clase. El tercer compartimiento contiene la lista de operaciones de la clase.
- ***Interface:*** Una interface utiliza un tipo para describir el comportamiento visible de una clase, de un componente o de un paquete. Una interface es un estereotipo de un tipo. (Muller P, 1997) Pueden ser mostradas con un rectángulo que se identifica con la palabra clave "interface" y que además tiene compartimentos. La lista de operaciones soportadas por la interface se coloca en el compartimiento de operaciones. El comportamiento de sus atributos puede ser omitido debido a que siempre estará vacío. Una interface también puede ser representada con un círculo con el nombre de la interface colocado debajo del símbolo. El círculo puede estar conectado por una línea sólida con las clases que implementan la interfaz. Las operaciones no se muestran en la notación que utiliza el círculo. Una clase que usa o requiere las operaciones que provee una interface, se conecta a ésta a través de una línea de segmentos con una flecha sólida que apunta a dicha interface.
- ***Relación de dependencia:*** Una relación de dependencia se establece entre clases cuando un cambio en el elemento independiente del modelo puede requerir un cambio en el elemento dependiente. Durante el análisis del sistema, los diagramas de clases son utilizados para mostrar las partes comunes y las responsabilidades de las entidades que proveen el

comportamiento del sistema. Durante el diseño del sistema, capturan la estructura de la clase que forma la arquitectura del sistema.

- ✓ ***Relación de asociación:*** Las relaciones de asociación permiten especificar que objetos van a estar asociados con otro objeto mediante un calificador. El calificador es un atributo o conjunto de atributos de una asociación que determina los valores que indican cuales son los que se asocian. Las asociaciones representan relaciones estructurales entre clases y objetos. Una asociación simboliza una información cuya duración de vida no es relevante respecto a la dinámica general de los objetos instancias de las clases asociadas. (Booh, Rumbaugh, Jacobson, 1999).
- ✓ ***Relación de generalización:*** La generalización entre dos clases consiste en considerar a una clase como superclase y a la otra como la subclase. Puede haber más de una clase que se comporte como subclase.

- **Diagramas de Secuencia:** Los diagramas de secuencia muestran interacciones entre los objetos según un punto de vista temporal. (Booh, Rumbaugh, Jacobson, 1999) En particular, muestra los objetos que participan en la interacción y los mensajes que ellos intercambian arreglados en una secuencia de tiempo. Este diagrama no muestra la asociación entre los objetos. Representa una interacción, la cual es un conjunto de mensajes intercambiados entre objetos dentro de un sistema que afectan una operación o resultado. Estos diagramas tiene dos dimensiones: la dimensión vertical que representa el tiempo, y la dimensión horizontal que representa objetos diferentes.

Los objetos se comunican intercambiando mensajes representados por medio de flechas horizontales, orientadas del emisor del mensaje hacia el destinatario. Booh, Rumbaugh, Jacobson (1999), el ordenamiento horizontal de las líneas de vida de un objeto es arbitrario; varias etiquetas pueden ser

utilizadas (tales como marcas de tiempo, descripción de acciones durante una activación) en el margen o cerca de una activación o transición que ellos etiqueten.

La línea de vida de un objeto representa la presencia de un objeto en un momento particular. Si un objeto es creado o destruido, la línea de vida de un objeto comienza o termina en este punto. Cuando el objeto es destruido, se coloca una "X" grande al final y se envía un mensaje. Una activación muestra el período durante el cual un objeto está realizando una acción directa o se está ejecutando un procedimiento subordinado. Este representa la duración de la acción y controla la relación entre la activación y el objeto que realizó la llamada.

Un mensaje es una comunicación entre objetos que transporta información con la expectativa de que alguna acción sea realizada. La recepción de un mensaje es un tipo de evento. Un mensaje se muestra con una flecha sólida desde la línea de vida de un objeto a la línea de vida de otro objeto. En el caso de un mensaje al mismo objeto, la línea debe comenzar y terminar en el mismo símbolo del objeto.

Durante el análisis del sistema, los diagramas de secuencia indican la semántica de las interacciones primarias y secundarias y durante la fase de diseño los diagramas de secuencia muestran la semántica de los mecanismos del diseño lógico del sistema.

- ✓ **Diagrama de Colaboración:** Este diagrama es utilizado para modelar interacciones entre objetos. En el análisis de este diagrama es más importante el paso de información de un objeto a otro, constituido por los mensajes, que

en su diseño se detallan. Cada caso de uso se desarrolla como una realización de casos de uso, donde cada uno tiene un conjunto de clasificadores participantes que desempeñan diferentes roles. Cada clase de análisis representa un objeto o instancia de una clase en el diagrama de colaboración donde se establece la cooperación existente entre ellas. La secuencia en que estos objetos aparecen en un caso de uso se muestra usando números y comienza con un evento que llega a una interfaz, se sigue con un análisis de los eventos siguientes y la posible respuesta del sistema.

- ✓ **Diagrama de Objetos:** los diagramas de objetos modelan las instancias de elementos contenidos en los diagramas de clases. Un diagrama de objetos muestra un conjunto de objetos y sus relaciones en un momento concreto. Los diagramas de objetos se emplean para modelar la vista de diseño estática o la vista de procesos estática de un sistema al igual que se hace con los diagramas de clases, pero desde la perspectiva de instancias reales o prototípicas. Esta vista sustenta principalmente los requisitos funcionales de un sistema. Los diagramas de objetos permiten modelar estructuras de datos estáticas.

- ✓ **Diagrama de Estados:** se usan para representar gráficamente máquinas de estados finitos. Las Tablas de Transiciones son otra posible representación. El Lenguaje Unificado de Modelado (UML) especifica una notación estandarizada para diagramas de estado que puede utilizarse para describir clases, sistemas, subsistemas o incluso procesos de negocio. Los elementos básicos de notación que pueden usarse para componer un diagrama son:
- ✓ Círculo lleno, apuntando a un estado inicial
- ✓ Círculo hueco que contiene un círculo lleno más pequeño en el interior, indicando el estado final (si existiera)

- ✓ Rectángulo redondeado, denotando un estado. En la parte superior del rectángulo está el nombre del estado. Puede contener una línea horizontal en la mitad, debajo de la cual se indican las actividades que se hacen en el estado
- ✓ Flecha, denotando transición. El nombre del evento (si existiera) que causa esta transición etiqueta el cuerpo de la flecha. Se puede añadir una expresión de Guarda, encerrada en corchetes ([]) denotando que esta expresión debe ser cierta para que la transición tenga lugar. Si se realiza una acción durante la transición, se añade a la etiqueta después de "/". NombreDeEvento[ExpresiónGuarda]/acción Línea horizontal gruesa con x>1 líneas entrando y 1 línea saliendo o 1 línea entrando y x>1 líneas saliendo. Estas denotan Unión/Separación, respectivamente.

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

La interpretación de un diagrama de actividades depende de la perspectiva considerada: en un diagrama conceptual, la actividad es alguna tarea que debe ser realizada; en un diagrama de especificación o de implementación, la actividad es un método de una clase. Generalmente se suelen utilizar para modelar los pasos de un algoritmo.

- **Diagrama de Implementación:** este diagrama muestra los aspectos físicos del sistema. Incluye la estructura del código fuente y la implementación, en tiempo de implementación. Existen dos tipos:

✓ ***Diagramas de componentes:*** muestra la dependencia entre los distintos componentes de software, incluyendo componentes de código fuente, binario y ejecutable. Un componente es un fragmento de código software (un fuente, binario o ejecutable) que se utiliza para mostrar dependencias en tiempo de compilación.

✓ ***Diagrama de plataformas o despliegue:*** muestra la configuración de los componentes hardware, los procesos, los elementos de procesamiento en tiempo de ejecución y los objetos que existen en tiempo de ejecución. En este tipo de diagramas intervienen nodos, asociaciones de comunicación, componentes dentro de los nodos y objetos que se encuentran a su vez dentro de los componentes. Un *nodo* es un objeto físico en tiempo de ejecución, es decir una máquina que se compone habitualmente de, por lo menos, memoria y capacidad de procesamiento, a su vez puede estar formado por otros componentes.

2.8 Tecnologías web

2.8.1 Intranet [10]

Red diseñada para el procesamiento de información dentro de una compañía u organización. Entre sus usos se incluyen servicios tales como distribución de documentos, distribución de software, acceso a bases de datos y aprendizaje. Las intranets deben su nombre a que en ellas se utilizan a menudo aplicaciones asociadas a Internet, tales como páginas Web, sitios FTP, correo electrónico, grupos de noticias

y listas de distribución, a las cuales únicamente se pueden tener acceso a los terminales de la propia compañía u organización.

2.8.2 Internet [11]

Es un método de interconexión de redes de computadoras implementado en un conjunto de protocolos denominado TCP/IP y garantiza que redes físicas heterogéneas funcionen como una red (lógica) única. De ahí que Internet se conozca comúnmente con el nombre de "red de redes", pero es importante destacar que Internet no es un nuevo tipo de red física, sino un método de interconexión.

2.8.3 Aplicación Cliente - Servidor [12]

Programa compartido en toda una red. El programa se encuentra almacenado en un servidor de red y puede ser utilizado simultáneamente por más de un cliente.

2.8.4 Servidor Web [13]

Un servidor web es un programa que implementa el *protocolo HTTP* (*HyperText Transfer Protocol*). Este protocolo pertenece a la capa de aplicación del modelo OSI y está diseñado para transferir lo que llamamos hipertextos, páginas web o páginas HTML (HyperText Markup Language): textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música.

Es un programa que se ejecuta continuamente en un ordenador (también se emplea el término para referirse al ordenador que lo ejecuta), manteniéndose a la espera de peticiones por parte de un cliente (un navegador web) y que responde a

estas peticiones adecuadamente, mediante una *página web* que se exhibirá en el navegador o mostrando el respectivo mensaje si se detectó algún error.

- **Servidor web internet Information Server**

Es una serie de servicios para los ordenadores que funcionan con Windows. Originalmente era parte del *Option Pack* para Windows NT. Luego fue integrado en otros sistemas operativos de Microsoft destinados a ofrecer servicios, como Windows 2000 o Windows Server 2003. Windows XP Profesional incluye una versión limitada de IIS. Los servicios que ofrece son: FTP, SMTP, NNTP y HTTP/HTTPS.

- **Servidor web Apache**

Este servicio convierte a un ordenador en un servidor de Internet o Intranet es decir que en las computadoras que tienen este servicio instalado se pueden publicar páginas web tanto local como remotamente (servidor web).

Los Servicios de Internet Information Services (IIS) proporcionan las herramientas y funciones necesarias para administrar de forma sencilla un servidor Web seguro. Si ha pensado alojar un sitio Web y FTP (File Transfer Protocol, Protocolo de transferencia de archivos) con IIS, configure el servidor como un servidor de aplicaciones.

El servidor HTTP Apache es un servidor web HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etc.), Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual. Cuando comenzó su desarrollo en 1995 se basó inicialmente en código del popular NCSA HTTPd 1.3, pero más tarde fue reescrito por completo. Su nombre se debe a que Behelendorf

eligió ese nombre porque quería que tuviese la connotación de algo que es firme y enérgico pero no agresivo, y la tribu Apache fue la última en rendirse al que pronto se convertiría en gobierno de EEUU, y en esos momentos la preocupación de su grupo era que llegasen las empresas y "civilizasen" el paisaje que habían creado los primeros ingenieros de internet. Además Apache consistía solamente en un conjunto de parches a aplicar al servidor de NCSA. Era, en inglés, *a patchy server* (un servidor "parcheado").

El servidor Apache se desarrolla dentro del proyecto HTTP Server (httpd) de la Apache Software Foundation Apache presenta entre otras características mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que ayude en su configuración.

2.8.5 Lenguaje de Etiquetas por Hipertexto [14]

El lenguaje de marcas de hipertexto que se utiliza para documentos del World Wide Web. HTML es una aplicación de SGML que utiliza etiquetas para marcar los elementos, como texto y gráficos, en un documento para indicar como deberían visualizar los exploradores Web estos elementos al usuario y como deberían responder a las acciones del usuario, como la activación de un enlace presionando una tecla o haciendo clic con el ratón. HTML 2.0, definido por el Internet Engineering Task Force (IETF), incluye características del HTML común a todos los exploradores Web alrededor de 1995 y fue la primera versión de HTML ampliamente utilizado en el WWW.

En 1994, se propuso HTML+ para ampliar el HTML 2.0 pero nunca se implementó. HTML 3.0, que nunca se estandarizó ni fue implementado totalmente

por ningún desarrollador de exploradores Web, introdujo las tablas. HTML 3.2, el último estándar propuesto, incorpora características ampliamente implementadas en 1996.

La mayoría de los exploradores, especialmente Netscape Navigator e Internet Explorer, reconocen más etiquetas de HTML que las incluidas en el estándar actual. HTML 4, la última especificación, soporta hojas de estilo y lenguajes de script e incluye características de internacionalización y de accesibilidad.

2.8.6 Técnica de desarrollo web AJAX [15]

AJAX, acrónimo de *Asynchronous JavaScript And XML* (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

Ajax es una tecnología asíncrona, en el sentido de que los datos adicionales se requieren al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página. JavaScript es el lenguaje interpretado (scripting language) en el que normalmente se efectúan las funciones de llamada de Ajax mientras que el acceso a los datos se realiza mediante *XMLHttpRequest*, objeto disponible en los navegadores actuales. En cualquier caso, no es necesario que el contenido asíncrono esté formateado en XML.

Ajax es una técnica válida para múltiples plataformas y utilizable en muchos sistemas operativos y navegadores, dado que está basado en estándares abiertos como JavaScript y Document Object Model (DOM).

2.8.6.1 Tecnologías incluidas en AJAX

AJAX es una combinación de cuatro tecnologías ya existentes:

- XHTML (o HTML) y hojas de estilos en cascada (CSS) para el diseño que acompaña a la información.
- Document Object Model (DOM) accedido con un lenguaje de scripting por parte del usuario, especialmente implementaciones ECMAScript como JavaScript y JScript, para mostrar e interactuar dinámicamente con la información presentada.
- El objeto XMLHttpRequest para intercambiar datos de forma asíncrona con el servidor web. En algunos frameworks y en algunas situaciones concretas, se usa un objeto iframe en lugar del XMLHttpRequest para realizar dichos intercambios.

2.8.7 Aplicaciones Web [16]

El creciente desarrollo del comercio electrónico así como el desplazamiento de las distintas organizaciones hacia la Web ha traído en la actualidad una constante evolución de las aplicaciones Web. Cada día se incrementan las transacciones financieras, la transferencia de información confidencial y ejecución de procesos online, entre otros, las cuales exigen funcionalidad, confiabilidad, usabilidad y eficiencia por mencionar algunas características de calidad. Esta relevancia de la economía genera grandes desafíos en las aplicaciones Web que son los de controlar y mejorar su calidad.

Aunque las aplicaciones Web están creciendo rápidamente. La mayoría de los desarrolladores Web ponen poca atención en el análisis de requisitos, así como en las metodologías y procesos de desarrollo. Además los desarrolladores de aplicaciones confían excesivamente en el conocimiento y experticia de los desarrolladores individuales y sus prácticas de desarrollo individual más bien que en las prácticas estándar. No obstante, son las mismas metodologías de desarrollo las que no tratan de manera adecuada y profunda los atributos de calidad. Estas situaciones traen como consecuencia que los atributos de calidad de los sistemas basados en la Web tales como la funcionalidad, confiabilidad, mantenibilidad, usabilidad y portabilidad no se les da la debida consideración que se merecen durante el proceso de desarrollo.

La revisión de los modelos de calidad de la IEEE y la ISO/IEC y de esta forma tomarlos como referencia en el análisis de las metodologías. Después se revisa que es una aplicación Web así como su diferencia de las aplicaciones convencionales para poder identificar y comprender cuales son aquellas características particulares que las hacen diferentes y así saber a qué nuevos aspectos enfrentar en el tratamiento de los requisitos de calidad. También se describen los atributos de calidad de las aplicaciones Web que actualmente varios investigadores consideran en sus trabajos y se analiza cual es el punto de vista del autor, se muestran resumidamente todas aquellas propuestas que tratan los atributos de calidad así como las técnicas que utilizan y que atributos son los que considera. Por último se hace una clasificación de los atributos que trata cada metodología en base al estándar IEEE 1061 y es mediante este estándar y dicha clasificación que se construye la taxonomía general de los atributos de calidad. Con esta taxonomía se pretende mostrar a los desarrolladores de sistemas Web cuales son aquellas metodologías, técnicas, actividades y requisitos de calidad que se deben considerar en la etapa de ingeniería de requisitos para dar mayor garantía de calidad a las aplicaciones Web.

No obstante cabe destacar que hay mucho campo de investigación en el área de ingeniería de requisitos de los sistemas Web puesto que la mayoría de las metodologías no consideran los atributos de calidad y aquellas que lo hacen no lo tienen muy claro o lo hacen de forma limitada, la ingeniería Web es una disciplina que continua con su proceso de madurez.

2.8.7.1 Anatomía de una Página Web

Según Mas (2005) una página Web es superficialmente parecida a cualquier otro documento: un texto, unas imágenes, todo compuesto de una determinada manera. Una página Web es un tipo de fichero que tiene poco de particular: se trata simplemente de un fichero de texto, con una extensión .htm o .html (de hypertext markup language - lenguaje de hipertexto.) Este fichero contiene el texto más una serie de códigos que permiten dar formato a la página en el navegador: por ejemplo, distribuir en columnas, poner letras en negrita, asignar colores, rodear una imagen con texto... El programa navegador (normalmente Internet Explorer o Navigator) interpreta los códigos del html para mostrar en pantalla la información contenida y del modo que se ha especificado aquellos códigos. En la figura 2.9 se observan los elementos principales que conforman una página web.

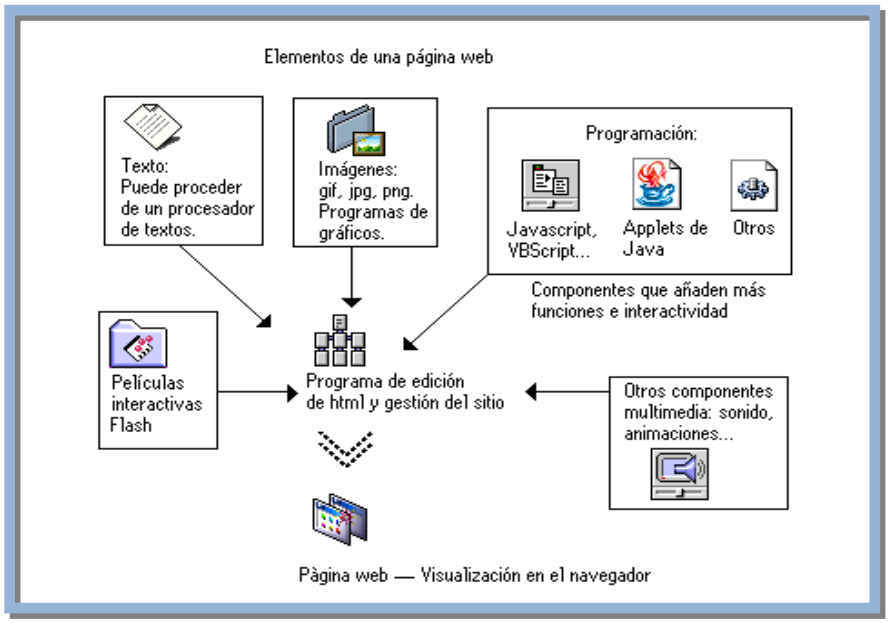


Figura 2.9 Anatomía de una página Web.

Principales componentes de la página Web típica, como se muestra en la ilustración anterior:

- **Texto.** El texto editable se muestra en pantalla con alguna de las fuentes que el usuario tiene instaladas (a veces se utiliza una tecnología de fuentes incrustadas, con lo que vemos en el monitor una fuente que realmente no se tiene, pero es poco frecuente).
- **Gráficos.** Son ficheros enlazados desde el fichero de la página propiamente dicho. Se puede hablar de dos formatos casi exclusivamente: GIF y JPG.
- **Formularios.** Son una mezcla de texto y a veces gráficos, que permiten enviar información por parte del visitante, por ejemplo, consultando un catálogo, solicitando más información, comunicando su opinión, votando en una encuesta. Existen diferentes modelos de formulario; algunos simplemente se envían por correo electrónico; otros funcionan ejecutando un programa guión en el servidor.

2.8.8 Concepto general de WebML [17]

Es un lenguaje de modelado de alto nivel para la especificación de hipertexto, el cual consiste en simples conceptos visuales para expresar un hipertexto como un conjunto de páginas compuestas de unidades de contenido y operaciones enlazadas, y para unir tales unidades y operaciones a la data que hacen referencia. WebML sigue el estilo reconocido de lenguajes de modelado conceptual como Entidad – Relación y UML: cada concepto tiene una representación gráfica y las especificaciones son diagramas. Así como el modelo Entidad – Relación construye, también los diagramas de WebML se pueden representar usando la sintaxis de UML, posiblemente con pérdida de concisión pero no del poder expresivo.

En general, WebML es un audaz e impresionante logro. Reúne elegantemente y parece ser capaz de construir cualquier cosa a través del uso apropiado de composición y la construcción de bloques. Es impresionante que WEBML ya tenga un lenguaje y modelo de diseño visual bien trabajado. Se basa en tres modelos fundamentales para el desarrollo de aplicaciones Web, como lo son:

- **Modelo de Datos:** El modelo de datos de WebML es una adaptación conveniente de los modelos conceptuales de diseño de datos que se emplea en otras disciplinas como diseño de bases de datos, ingeniería de software y representación del conocimiento. El modelo de datos de WebML es compatible con el modelo de datos Entidad – Relación usado en el diseño conceptual de bases de datos, también es compatible con los diagramas de clase UML empleados en el modelado orientado a objetos.

  El elemento fundamental del modelo de datos son las entidades, definidas como contenedores de elementos de datos, y sus relaciones definidas

como las conexiones semánticas entre entidades. Las entidades tienen propiedades, llamadas atributos, con un tipo asociado. Las entidades pueden ser organizadas de manera jerárquica y sus relaciones pueden restringirse por medio de la cardinalidad.

- **Modelo de Hipertexto:** El modelo de hipertexto especifica cómo se compone el sitio y la navegación en el sitio. La composición del sitio describe las páginas que forman parte del hipertexto y las unidades de contenido que constituyen cada página. Las páginas de un sitio Web son catalogadas como contenedores de información que es enviada al usuario. Las unidades son elementos atómicos de contenido, empleadas para publicar información descrita en el modelo de datos. WebML contiene siete (7) tipos de unidades predefinidas para desarrollar páginas Web: data, multi-data, index (y sus variantes jerárquicas y selección múltiple), entry, scroller. Cada unidad está asociada a una entidad subyacente, de la cual se obtiene el contenido. La especificación de una entidad subyacente determina el tipo de objeto del cual se deriva el contenido de la unidad.

  La navegación del sitio se realiza a través de enlaces, los cuales se definen entre unidades que se encuentran en una misma página, en diferentes páginas o entre páginas completas. La información transportada a través de los enlaces se conoce como contexto de navegación o simplemente contexto. Los enlaces que transportan información contextual se denominan enlaces contextuales mientras que los que no transportan información son conocidos como enlaces no contextuales. La información contextual generalmente es necesaria para asegurar las operaciones de computación de las unidades.

El modelo de hipertexto debería estar al nivel correcto de abstracción; la especificación del hipertexto debe ser mantenida a un nivel conceptual, el cual significa que no debe comprometer mucho el diseño y la implementación de los detalles, tales como la distribución actual de la funcionalidad entre las varias capas de la aplicación Web.

- **Modelo de Gestión de Contenido:** Las aplicaciones Web con frecuencia realizan operaciones sobre datos. Ejemplos de esto es llenar formularios de un perfil personal, la adición de ítems al carrito de compras, o la actualización de contenido publicado en la Web por medios de aplicaciones de manejo de contenidos. En todos estos casos, las acciones llevadas a cabo por medio de interfaces Web, tienen efectos secundarios, por ejemplo ellos cambian el contenido de algunos orígenes de datos conectados al sitio Web. Además de actualizar los datos, las aplicaciones Web pueden invocar programas definidos externamente, enviándoles entradas de datos, el cual depende del contenido de la página y de lo que escoja el usuario. Ejemplos de tales operaciones de propósito general es la identificación de un usuario, enviar un mail, entre otros

  Estas operaciones en WebML no afectan el modelo de datos, y requiere dos simples e intuitivas extensiones del modelo de hipertexto. La primera extensión es la notación de unidades operacionales, las cuales son usadas para expresar algún proceso ejecutado como el resultado de navegar un enlace; una unidad operacional puede denotar una manipulación de datos o la ejecución de un servicio externo genérico.

  La segunda extensión se aplica a unidades operacionales de los enlaces de salida, los cuales están distinguidos en OK-links y KO-links. OK y KO-links capturan el concepto de operación exitosa y fallida, respectivamente, y

permiten al diseñador tomar rutas alternativas después de la ejecución de una operación, dependiendo del resultado de la ejecución.

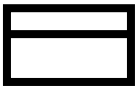
WebML incluye muchas unidades operacionales predefinidas, las cuales ofrecen la mayoría de las primitivas más usadas para actualizar instancias de las entidades y relaciones de la aplicación, creando, modificando, y borrando objetos, y conectándose y desconectándose entre relaciones, además otras operaciones útiles como ingreso de usuario, salida de usuario y envío de e-mails.

Las operaciones de manejo de contenido predefinidas pueden ser agrupadas en transacciones, las cuales son secuencias de actualizaciones ejecutadas automáticamente; cuando una transacción es especificada, o toda la secuencia de operaciones que la constituyen se ejecuta exitosamente, o la secuencia toda no se ejecuta.

2.8.8.1 Sumario de Estereotipos del Modelo de Hipertexto WebML

La herramienta de modelado WebML ofrece un amplio conjunto de notaciones y estereotipos que permiten representar la definición, estructuración e interpretación de aplicaciones web.

En la tabla 2.1 se describe cada estereotipo.

Elemento WebML	Descripción	Propiedades
Entity (Entidad) 	Una entidad es la representación de un objeto o concepto del mundo real que se describe en una base de datos.	Ninguna
ISA Hierarchy (Jerarquías ISA) 	Jerarquías Isa: Se dice A isa B si el conjunto de entidades B es una generalización del conjunto de entidades A. attrib(B) attrib(A). A hereda de B (en el mismo sentido de la programación orientada a objetos).	Ninguna
Relations… (Relaciones)	Una relación entre dos o más entidades describe alguna interacción entre las mismas.	Ninguna
Link (Enlace)	Es una conexión orientada entre dos unidades o páginas. Abstrae el concepto de ancla y permite portar información (por medio de parámetros entre unidades) Pueden ser definidos como:	Enlaces normales, automáticos y de transporte. **»** Nombre **»** Elemento origen **»** Elemento destino **»** Tipo de enlace

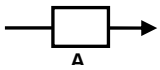
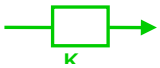
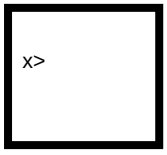
	» Enlace de transporte » Enlace automático	» Parámetros de enlace
Transport Link (Enlace de transporte) **Pa rametersPar** 	Enlace de Transporte: no funcionan como un ancla, pero son capaces de pasar parámetros.	Parámetros de enlace » Nombre » Valor origen
Automatic Link (Enlace automatico) **Pa rametersPar** 	Enlace Automático: son navegados sin la intervención del usuario.	Parámetros de enlace » Nombre » Valor origen

Tabla 2.1. Estereotipos WebML 1/7

Elemento WebML	Descripción	Propiedades
Link OK (Enlace OK) **Pa rametersPar**	Enlace OK: enlace de operación distinguida, se ejecuta en caso de que la operación haya sido exitosa	» Nombre » Elemento origen (unidad de operación)

		» Elemento destino » Parámetros de enlace
Link KO (Enlace KO) **Pa rametersPar** 	Enlace KO: enlace de operación distinguida, se ejecuta en caso de que la operación falle.	» Nombre » Elemento origen (unidad de operación) » Elemento destino » Parámetros de enlace
Window (Ventana) 		Ninguna
Web Page (Pagina Web) 	Representa la interfaz actual buscada por el usuario. Esta contiene sub-paginas AND/OR	» Nombre » Punto de referencia Contenido: unidades o subpáginas
Area (Área)	Es un contenedor de páginas o	» Nombre

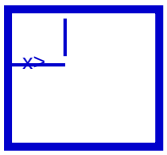
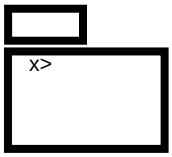
	recursivamente de sub-áreas, donde cada una puede ser usada para obtener una organización jerárquica del hipertexto	**»** Punto de referencia Contenido: páginas, sub-áreas y página por defecto o subpágina

Tabla 2.1. Estereotipos WebML 2/7

Elemento WebML	Descripción	Propiedades
Siteview (Vista del sitio) 	Representa una vista de hipertexto	**»** Nombre **»** Contenido: páginas, áreas y página inicial
Transaction (Transacción) 	Es una secuencia de operaciones ejecutadas atómicamente, lo que significa que todas las operaciones individuales se ejecutan exitosamente o toda la secuencia entera no es hecha	Ninguna

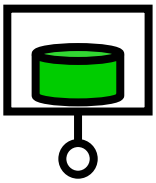
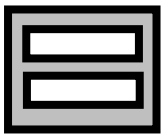
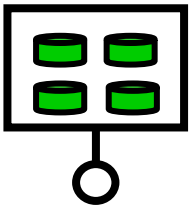
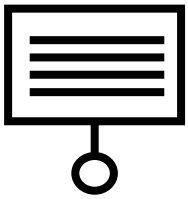
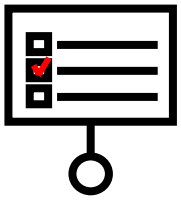
Data Unit **(Unidad de datos)** 	Publica un objeto obtenido de una entidad determinada	**»** Nombre **»** Entidad de Origen **»** Selector (opcional) Atributos incluidos
Entry Unit **(Unidad de entrada)** 	Esta soporta el ingreso de datos basado en un formulario	**»** Nombre **»** Punto de referencia Contenido: unidades o subpáginas

Tabla 2.1. Estereotipos WebML 3/7

Elemento	Descripción	Propiedades

WebML		
Multidata Unit (Unidad de datos multiples) 	Presenta múltiples objetos de una entidad juntos, repitiendo la presentación de muchas unidades de dato	**»** Nombre **»** Entidad de Origen **»** Selector (opcional) **»** Atributos incluidos Cláusula de orden (opcional)
Index Unit (Unidad índice) 	Presentan múltiples objetos de una entidad como una lista	**»** Nombre **»** Entidad de Origen **»** Selector (opcional) **»** Atributos incluidos **»** Cláusula de orden (opcional)
Multichoice (Unidad índice de múltiples elecciones) 	Una variante del anterior, donde cada elemento de la lista está asociado con un “checkbox” permitiendo al usuario seleccionar múltiples objetos	**»** Nombre **»** Entidad de Origen **»** Selector (opcional) **»** Atributos incluidos Cláusula de orden (opcional)

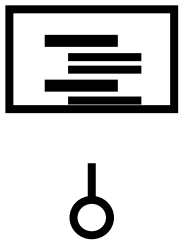
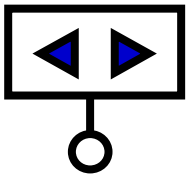
Hierarchical Index Unit (Unidad índice jerárquica) 	Una variante del “index unit”, en cual las entradas de los índices están organizadas en un árbol multi - nivel	**»** Nombre **»** Por cada nivel: **»** Entidad de Origen **»** Selector (opcional) **»** Atributos incluidos **»** Cláusula de orden (opcional)

Tabla 2.1. Estereotipos WebML 4/7

Elemento WebML	**Descripción**	**Propiedades**
Scroller Unit (Unidad de desplazamiento) 		
Create Unit (Unidad de	Habilita la creación de una nueva instancia de una entidad	**»** Nombre **»** Entidad de origen

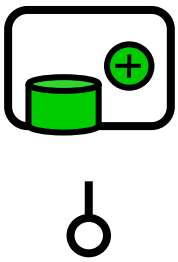
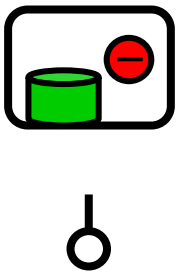
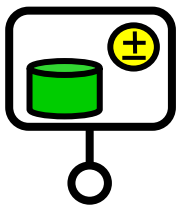
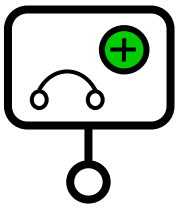
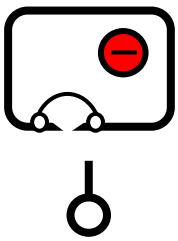
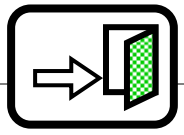
creación) 		Conjunto de agnaciones de valores
Delete Unit (Unidad de eliminación) 	Elimina uno o más objetos de una entidad dada	**»** Nombre **»** Entidad de origen **»** Selector
Modify Unit (Unidad de modificación) 	Modifica o actualiza uno o más objetos de una entidad dada	**»** Nombre **»** Entidad de origen **»** Selector **»** Conjunto de asignaciones de valores

Tabla 2.1. Estereotipos WebML 5/7

Elemento WebML	Descripción	Propiedades
Connect Unit (Unidad de conexión) 	Crea nueva instancia de una relación	» Nombre » Rol de la relación » Selector de la entidad de origen » Selector de la entidad objetivo
Disconnect Unit (Unidad de desconexión) 	Elimina una instancia de una relación	» Nombre » Rol de la relación » Selector de la entidad de origen Selector de la entidad objetivo
Login Unit (Unidad de entrada al sistema) 	Verifica la identidad de un usuario de acceda al sitio	» Nombre de Usuario » Contraseña

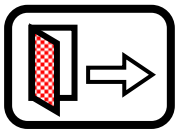
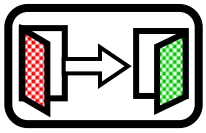
Logout Unit (Unidad de cierre de sesión) 	Lleva al usuario a la página principal sin control de acceso	**»** Ninguno

Tabla 2.1. Estereotipos WebML 6/7

Elemento WebML	**Descripción**	**Propiedades**
Change Group Unit (Unidad de cambio de grupo) 	Verifica la entidad de un usuario accediendo al sitio por medio de otro grupo de usuarios	**»** Nombre de Usuario **»** Contraseña
Sendmail	Provee la capacidad de enviar	**»** Remitente

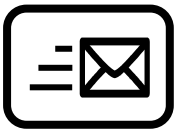
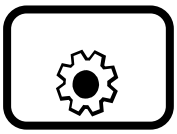
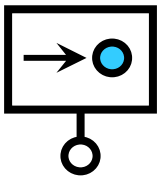
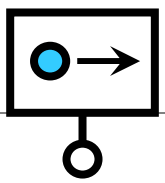
Unit (Unidad de envió de correo o mensajes) 	mensajes de email	**»** Recipiente **»** Asunto **»** Cuerpo **»** Adjuntos
Generic Operation Unit (Unidad de Operación Genérica) 	Define una operación genérica: la contribución y el producto. Por el que los parámetros deben ser definidos por el diseñador	Las propiedades son definidos por el diseñador
Set Unit (Unidad de Asignación) 	Asigna un valor a un parámetro global	**»** Parámetro global
Get Unit (Unidad de Extracción) 	Retira el valor de un parámetro global	**»** Parámetro global

Tabla 2.1. Estereotipos WebML 7/7

2.9 LENGUAJE DE PROGRAMACIÓN PHP

2.9.1 Definición [18]

Es uno de los lenguajes de lado servidor más extendido en la web y que ha sido creado por una gran comunidad de personas. El sistema fue desarrollado originalmente en el año 1994 por Rasmus Lerdorf como un CGI escrito en C que permitía la interpretación de un número limitado de comandos. El sistema fue denominado Personal Home Page Tools y adquirió relativo éxito gracias a que otras personas pidieron a Rasmus que les permitiese utilizar sus programas en sus propias páginas. Dada la aceptación del primer PHP y de manera adicional, su creador diseñó un sistema para procesar formularios al que le atribuyó el nombre de FI (Form Interpreter) y el conjunto de estas dos herramientas, sería la primera versión compacta del lenguaje: PHP/FI.

La siguiente gran contribución al lenguaje se realizó a mediados del 97 cuando se volvió a programar el analizador sintáctico, se incluyeron nuevas funcionalidades como el soporte a nuevos protocolos de Internet y el soporte a la gran mayoría de las bases de datos comerciales. Todas estas mejoras sentaron las bases de PHP versión 3.

Actualmente PHP se encuentra en su versión 5, que utiliza el motor Zend, desarrollado con mayor meditación para cubrir las necesidades actuales y solucionar algunos inconvenientes de la anterior versión. Algunas mejoras de esta nueva versión

son su rapidez -gracias a que primero se compila y luego se ejecuta, mientras que antes se ejecutaba mientras se interpretaba el código-, su mayor independencia del servidor Web -creando versiones de PHP nativas para más plataformas- y un API más elaborado y con más funciones.

El lenguaje PHP es un lenguaje de programación de estilo clásico, es decir, es un lenguaje de programación con variables, sentencias condicionales, ciclos (bucles), funciones. No es un lenguaje de marcado como podría ser HTML, XML o WML. El programa PHP es ejecutado en el servidor y el resultado enviado al navegador. El resultado es normalmente una página HTML pero igualmente podría ser una página WML.

2.10 Base de datos

2.10.1 Definición [19]

Una base de datos es una colección de datos relacionados. Por datos, se quiere decir hechos conocidos que pueden registrarse y que tienen un significado implícito. Una base de datos tiene las siguientes propiedades implícitas:

- Una base de datos representa algunos aspectos del mundo real, en ocasiones denominado minimundo o Universo del Discurso. Los cambios en el minimundo se reflejan en la base de datos.
- Una base de datos es una colección de datos lógicamente coherentes, con algunos significados inherentes. Un conjunto aleatorio de datos no puede considerarse como una base de datos.
- Las bases de datos se diseñan, construyen y pueblan con datos para un propósito específico. Está destinada a un grupo de usuarios y tiene algunas aplicaciones preconcebidas de interés para dichos usuarios.

2.10.2 Sistema De Gestión De Base De Datos (SGBD) [19]

Es una colección de programas que permiten a los usuarios crear y mantener una base de datos. Por lo tanto un SGBD es un sistema de software de propósito general que facilita los procesos de definición, construcción, y manipulación de base de datos para distintas aplicaciones. Ahora veamos descrito cada uno de estos procesos:

- **La definición**: consiste en especificar los tipos de de datos, las estructuras y restricciones para los datos que se van a almacenar en dicha base de datos.
- **La construcción**: es el proceso de almacenar los datos concretos sobre algún medio de almacenamiento controlado por el SGBD.
- **La manipulación**: incluye funciones como consultar la base de datos para recuperar unos datos específicos, actualizar la base de datos para reflejar los cambios ocurridos en el minimundo, y generar informes a partir de los datos.

2.10.3 Arquitectura de un SGBD [19]

Una arquitectura apropiada para los sistemas de base de datos es la llamada arquitectura de tres esquemas también conocida como la arquitectura ANSI/SPARC, la cual tiene como objetivo separar las aplicaciones del usuario y la base de datos física. Esta arquitectura se define en los tres niveles siguientes:

- **El nivel interno:** Tiene un esquema interno, que describe la estructura física de almacenamiento de la base de datos. El esquema interno emplea un modelo de datos físico y describe todos los detalles para su almacenamiento, así como los caminos de acceso para la base de datos.
- **El nivel conceptual:** Tiene un esquema conceptual, que describe la estructura de la base de datos completa para una comunidad de usuarios. El esquema conceptual oculta los detalles de las estructuras físicas de almacenamiento y

se concentra en describir entidades, tipos de datos, vínculos, operaciones de los usuarios y restricciones. En este nivel podemos usar un modelo de datos de alto nivel o uno de implementación.

- **El nivel externo o de vistas**: Incluye varios esquemas externos o vistas de usuario. Cada esquema externo describe la parte de la base de datos que interesa a un grupo de usuarios determinado, y oculta a ese grupo el resto de la base de datos.

2.10.4 Modelo de datos [19]

Un modelo de datos es un conjunto de conceptos que sirven para describir la estructura de una base de datos. El concepto de estructura de una base de datos hace referencia a los tipos de datos, los vínculos y las restricciones que deben cumplirse para esos datos.

La mayoría de los modelos de datos contienen un conjunto de operaciones básicas para especificar lecturas y actualizaciones de la base de datos. Los modelos de datos más utilizados son los conocidos modelos de datos de representación o de implementación y entre ellos están los más comunes que son:

- El modelo de datos relacional.
- El modelo de red.
- El modelo jerárquico.

2.10.5 Arquitectura Cliente – Servidor [19]

Se usa para caracterizar un SGBD cuando la aplicación se ejecuta físicamente en una máquina, llamada cliente, y otra, el servidor, se encarga del almacenamiento y

el acceso a los datos. Los proveedores ofrecen diversas combinaciones de clientes y servidores; por ejemplo, un servidor para varios clientes.

2.10.6 Lenguaje Estructurado de Consultas SQL [19]

El lenguaje de consulta estructurado es un sublenguaje de base de datos utilizado para la consulta, actualización y administración de bases de datos relacionales, el estándar de facto para los productos de bases de datos.

2.10.7 Sistema Manejador de Base de Datos MySQL [19]

Es un sistema manejador de bases de datos relacional de código abierto que es gratis para muchos usos. MySQL al principio enfrentó oposición debido a su falta de apoyo a construcciones básicas de SQL tales como consultas anidadas y claves externas. Últimamente, sin embargo, MySQL encontró una base de usuarios entusiastas por sus términos de licencia tan liberal, su ejecución tan vivaz y facilidad de uso. También fue ayudado en parte por una amplia variedad de tecnologías tales como PHP, Python, Java, Perl y el hecho que tenía módulos estables y muy bien documentados. MySQL no ha fallado en recompensar la lealtad de sus usuarios con la agregación de consultas anidadas y claves externas a partir de la versión 4.1.

2.11 Software libre

2.11.1 Definición [20]

El software libre es una cuestión de libertad, no de precio. Para comprender este concepto, debemos pensar en la acepción de libre como en libertad de expresión y no como en “barra libre de cerveza”.

Con software libre nos referimos a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, modificar y mejorar el software. Existen cuatro clases de libertad para los usuarios de software:

- **Libertad 0:** La libertad para ejecutar el programa sea cual sea nuestro propósito.
- **Libertad 1:** La libertad para estudiar el funcionamiento del programa y adaptarlo a tus necesidades – el acceso al código fuente es condición indispensable para esto.
- **Libertad 2:** La libertad para redistribuir copias y ayudar así a tu vecino.
- **Libertad 3:** La libertad para mejorar el programa y luego publicarlo para el bien de toda la comunidad – el acceso al código fuente es condición indispensable para esto.

Software libre es cualquier programa cuyos usuarios gocen de estas libertades, de modo que es libre redistribuir copias con o sin modificaciones, de forma gratuita o cobrando por su distribución, a cualquiera y en cualquier lugar. Gozar de esta libertad significa, entre otras cosas, no tener que pedir permiso ni pagar para ello.

Asimismo, es libre introducir modificaciones y utilizarlas de forma privada, ya sea en el trabajo o en tiempo libre, sin ni siquiera tener que mencionar su existencia. Si se decidiera publicar estos cambios, no es obligado notificarlo de ninguna forma ni a nadie en particular.

La libertad para utilizar un programa significa que cualquier individuo u organización podrán ejecutarlo desde cualquier sistema informático, con cualquier fin y sin la obligación de comunicárselo subsiguientemente ni al desarrollador ni a ninguna entidad en concreto.

La libertad para redistribuir copias supone incluir las formas binarias o ejecutables del programa y el código fuente tanto de las versiones modificadas como de las originales. La distribución de programas en formato ejecutable es necesaria para su adecuada instalación en sistemas operativos libres. No pasa nada si no se puede producir una forma ejecutable o binaria, dado que no todos los lenguajes pueden soportarlo, pero todos debemos tener la libertad para redistribuir tales formas si se encuentra el modo de hacerlo.

Para que las libertades 1 y 3 – la libertad para hacer cambios y para publicar las versiones mejoradas – adquieran significado, debemos disponer del código fuente del programa. Por consiguiente, la accesibilidad del código fuente es una condición necesaria para el software libre.

Para materializar estas libertades, éstas deberán ser irrevocables siempre que no cometamos ningún error; si el desarrollador del software pudiera revocar la licencia sin motivo, ese software dejaría de ser libre.

Sin embargo, ciertas normas sobre la distribución de software libre nos parecen aceptables siempre que no planteen un conflicto con las libertades centrales. Por ejemplo, el copyleft, grosso modo, es la norma que establece que, al redistribuir el programa, no pueden añadirse restricciones que nieguen a los demás sus libertades centrales.

Esta norma no viola dichas libertades, sino que las protege. De modo que se puede pagar o no por obtener copias de software libre, pero independientemente de la manera en que se obtenga, siempre se tendrá la libertad para copiar, modificar e incluso vender estas copias.

.

CAPITULO III. FASE DE INICIO

3.1 Introducción

A continuación se describe el funcionamiento del sistema de forma general, en esta fase se muestra la perspectiva que tiene el desarrollador mediante los diferentes diagramas y modelos, en este caso mediante el Proceso Unificado de Desarrollo de Software y el Lenguaje Unificado de Modelado UML para la representación del sistema.

El objetivo de esta fase se basa en conocer y analizar a fondo las actividades y procesos llevados a cabo en el área de estudio con el fin de describir los requerimientos principales, formulando una arquitectura aspirante del sistema, para esto es necesario la comunicación y el entendimiento eficiente por parte de los desarrolladores y los usuarios del sistema.

En esta fase también se deben identificar los posibles riesgos críticos que pueden afectar al sistema, se debe realizar una investigación con el fin de establecer una opinión racional y justificable del propósito global y la viabilidad del nuevo sistema, y decidir si vale la pena invertir tiempo en un estudio más profundo.

3.2 Contexto del sistema

Es necesario conocer, estudiar y analizar las actividades relacionadas con la programación académica de la Universidad de Oriente - Núcleo de Anzoátegui para poder comprender el contexto del sistema, mediante las siguientes herramientas:

- Realizando entrevistas y visitas a los departamentos académicos y direcciones de escuela.
- Solicitar datos e información al Centro de Computación Académica.

Todo esto con el fin de conocer el proceso que se lleva a cabo e identificar las debilidades que existen, de manera que éstas puedan ser corregidas a futuro con la automatización del sistema para el control de la programación académica. Una vez estructurada la información necesaria se consigue una aproximación del contexto del sistema representado mediante el modelo de dominio, describiendo los conceptos importantes del contexto como objeto del dominio y enlazando estos objetos entre sí.

3.2.1 Comprensión de los Requisitos

Los requisitos son capacidades y condiciones con las cuales debe estar conforme el sistema. El primer reto del trabajo de los requisitos es encontrar, comunicar y recordar lo que se necesita realmente, de manera que tenga un significado claro. Para establecer los requisitos iniciales, se consideró necesario implementar el modelo de dominio del sistema y modelo de casos de uso para obtener el modelo conceptual y así estudiar los requerimientos fundamentales del sistema.

3.2.2 Modelo de dominio del sistema

Este modelo se encarga de capturar los objetos más importantes vinculados al contexto del sistema. Estos objetos representan los eventos que tienen lugar en el entorno donde se desenvuelve el sistema.

Un modelo de dominio es una representación visual de las entidades del mundo real en un área de interés. También se les denomina modelos conceptuales,

modelo de objetos del dominio y modelos de objetos de análisis. Para la descripción del modelo de dominio deben realizarse diagramas de clases basados en el Lenguaje de Modelado UML. En la figura 3.1 se muestra el modelo de dominio.

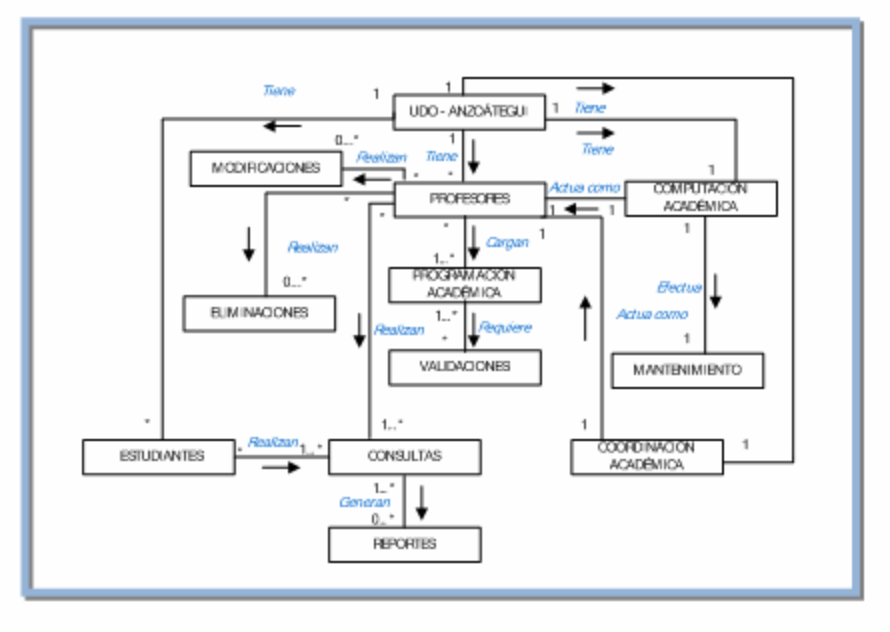


Figura 3.1. Modelo de dominio del sistema

3.2.2.1 Glosario de Términos del Modelo de Domino

- **UDO - Anzoátegui:** la Universidad de Oriente, Núcleo de Anzoátegui se define como un sistema de Educación Superior al servicio del país con objetivos comunes a las demás universidades venezolanas y del mundo, esta sede se encuentra ubicada en la Ciudad de Barcelona, Estado Anzoátegui.
- **Profesores:** los profesores que poseen privilegios para usar este sistema son los que laboran en los Departamentos Académicos, las Direcciones de Escuela y cualquier otro al que se le otorgue permisos, el sistema les ayuda en el ingreso de toda la información relacionada con la programación académica:

asignaturas, aulas, departamentos, escuelas especialidades, pensum, profesores y programación. También pueden solicitar consultas, generar reportes en formato digital o impreso, modificar campos y eliminar registros.

- **Estudiantes:** son usuarios que solo podrán consultar algunos datos de la programación académica el pensum y la programación, además de generar reportes en formato digital o impreso.

- **Coordinación Académica:** el personal que labora en la coordinación académica posee los mismos privilegios que los profesores para ingresar y administrar los datos de la programación académica.

- **Computación Académica:** el personal que labora en este centro se encarga de realizar el mantenimiento del sistema, creando grupos de usuarios y otorgando privilegios de acceso y limpiando la tabla programación de la base de datos realizando antes una copia de seguridad, además tiene los mismos permisos que los profesores.

- **Mantenimiento:** es una tarea que solo puede ser realizada por el personal que labora en el Centro de Computación Académica, este proceso se basa en llevar el control de los privilegios de cada grupo de usuarios, limpiar la tabla de la programación y cualquier modificación que el sistema requiera.

- **Validaciones:** la data ingresada es verificada por el sistema en relación al formato de los datos.

- **Programación académica:** se refiere a todos los datos relacionados con la programación académica: asignaturas, aulas, departamentos, escuelas

especialidades, mapa de aulas, pensum, profesores y programación. Estos datos se almacenan en el servidor ubicado en el Centro de Computación Académica, solo los usuarios con privilegios pueden ingresar y administrar estos datos.

- **Consultas:** representa la opción de poder visualizar los datos relacionados con la programación académica: asignaturas, aulas, departamentos, escuelas especialidades, mapa de aulas, pensum, profesores y programación.

- **Modificaciones:** se pueden editar campos de las asignaturas, las aulas, las escuelas, las especialidades y los profesores.

- **Eliminaciones:** se pueden borrar registros de las asignaturas, las aulas, las escuelas, las especialidades, el pensum y la programación.

- **Reportes:** se pueden generar los reportes en formato digital o impreso para la programación y el pensum.

3.3 Riesgos del sistema

Es importante considerar los riesgos en todo desarrollo de software, un riesgo es la variable del proyecto que pone en peligro el éxito del mismo. Los riegos constituyen la probabilidad de que un proyecto sufra sucesos no deseables.

Durante esta fase es importante detectar los riesgos que pueda sufrir el sistema, y tratar aquellos que podrían en un futuro afectar a la aplicación sobre todo

en las últimas fases del proceso unificado, este es el motivo por el cual se debe llevar a cabo iteraciones que examinen los riesgos.

Todos los riesgos técnicos pueden hacerse corresponder con un caso de uso o un escenario correspondiente. A continuación se muestra la lista de los riesgos del sistema:

- No poseer una definición clara de los requisitos del usuario, provoca un mal conocimiento del ámbito del sistema, es importante atenuarlo en la fase de inicio.
- Es importante conocer todas las necesidades del usuario y crear una visión clara de lo que se quiere.
- Una arquitectura que no sea lo suficientemente robusta, es considerado un riesgo crítico y debe solventarse durante la fase de inicio y elaboración.

3.4 Requisitos del sistema

Lo fundamental es guiar el desarrollo hacia un sistema correcto. La captura de requisititos muchas veces se hace complicada debido a que la mayoría de los usuarios no saben que parte de su trabajo pueden transformarse en software; con frecuencia los usuarios no saben cuáles son los requisitos ni tampoco cómo especificarlos de una forma precisa.

La técnica inmediata para identificar los requisitos del sistema se basa en los casos de usos, éstos capturan tanto los requisitos funcionales como los no funcionales que son específicos de cada caso de uso.

3.4.1 Requisitos Funcionales

- El sistema debe incluir la capacidad de administrar y gestionar los usuarios que harán uso del mismo.
- El sistema debe incorporar datos relacionados con los profesores que laboran en la Institución.
- El sistema debe incorporar datos relacionados con el personal que labora en el Centro de Computación Académica y en la Coordinación Académica.
- El sistema debe incorporar datos sobre los estudiantes de la universidad.
- El sistema debe permitir a los usuarios con permisos cargar la información relacionada con la programación académica: asignaturas, aulas, departamentos, escuelas, especialidades, pensum, profesores y programación.
- El sistema debe permitir la validación de la data ingresada en el sistema a fin de mantener el formato de los datos.
- El sistema debe facilitar al usuario, la realización de consultas de toda la información relacionada con la programación académica: asignaturas, aulas, departamentos, escuelas, especialidades, mapa de aulas, pensum, profesores y programación. Los estudiantes solo podrán consultar el pensum y la programación.
- El sistema debe permitir a los usuarios con permisos la modificación de alguno de los campos de las asignaturas, las aulas, los departamentos, las escuelas, las especialidades y los profesores.
- El sistema debe permitir a los usuarios con privilegios la eliminación de registros de las asignaturas, las aulas, los departamentos, las escuelas, las especialidades, los profesores y la programación, solo a los usuarios con privilegios de eliminación.
- El sistema debe permitir la generación de reportes ya sea en formato impreso o en formato digital.

- El sistema debe permitir al administrador del sistema el mantenimiento durante cada periodo académico, creando y administrando los permisos de usuarios, también admitiendo limpiar la tabla de la programación.

3.4.2 Requisitos No Funcionales

- El sistema debe estar constituido y estructurado como una aplicación WEB que podrá ser usada tanto a nivel local (Intranet) como a nivel externo (Internet).
- La estructura y diseño del sistema es escalable lo que significa que debe adaptarse fácilmente a cualquier cambio o mejora.
- El sistema debe poseer una interfaz amigable de fácil acceso y manejo.
- El sistema debe poseer un manual de usuario para la instalación y el correcto uso del sistema.

3.4.3 Requisitos de Software

- Sistema Operativo del Servidor: Windows 2003 Server.
- Sistema Operativo del Cliente: Microsoft Windows 2000 Professional o superior.
- Navegador de Internet: Internet Explorer 6, Mozilla Firefox 2 o superiores.
- Herramientas de Diseño Web: Editor de texto Gedit y Aptana Studio.
- Lenguaje de Programación: PHP versión 5.2.6
- Sistema de gestión de base de datos: MySQL versión 5.0.
- Servidor Web: Internet Information Server versión 5.0 o Apache 2.0

3.4.4 Requisitos de Hardware

3.4.4.1 Cliente:

- Procesador Pentium II o Superior.
- 256 MB de memória RAM o superior.
- Tarjeta Red 10/100/1000.
- Disco duro de 20 GB.

3.4.4.2 Servidor Web:

- Procesadores Pentium IV de o superior.
- 2 GB memoria RAM o superior.
- Tarjeta Red 10/100/1000.
- Disco duro de 20 GB.

3.4.5 Modelo de Casos de Uso

Es un modelo del sistema que permite capturar, crear y documentar requisitos, esto facilita a los desarrolladores de software y al cliente llegar a un acuerdo sobre lo que se quiere del sistema, y proporcionan la entrada fundamental para el análisis, diseño y las pruebas.

El modelo de casos de uso incluye los casos de usos y actores. Los casos de uso son fragmentos de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores. Los actores representan a todos aquellos que de una u otra manera interactúan con el sistema. Estos pueden ser los usuarios (personas que hacen uso del sistema) y también pueden estar representados por sistemas o dispositivos externos que se comunican con el sistema.

3.4.5.1 Identificación de Actores

Es importante realizar la identificación de los actores ya que ello permite obtener una visión del entorno externo del sistema. A continuación se describe cada uno de los actores que usaran el sistema:

Usuarios: pueden ser los siguientes tomando en cuenta que son grupos creados por el administardor del sistema:

- **Profesores (jefes de departamento):** los profesores jefes de departamento deben identificarse en el sistema para tener acceso al mismo y poder luego ingresar y administrar la información relacionada con la programación académica: asignaturas, aulas, departamentos, escuelas, especialidades, pensum, profesores y programación, teniendo la opción de consultar esta información, generar reportes en formato impreso o digital, modificar campos y eliminar registros.
- **Coordinación Académica:** el personal que labora en la coordinación tiene los mismos privilegios que los profesores.
- **Estudiantes:** los estudiantes deben identificarse en el sistema para poder visualizar el pensum y la programación, esta información es la que requieren al momento de crear sus horarios de clases, teniendo además la opción de generar reportes en formato digital o impreso.

Administrador: el personal del centro de computación académica es el encargado de administrar el sistema, crea los grupos de usuarios para otorgarle los privilegios de acceso y limpia la tabla de la base de datos relacionada con la programación realizando antes una copia de seguridad, además el administrador tiene los mismos privilegios que los profesores.

3.4.5.2 Identificación de Casos de Uso

- **Entrar al sistema:** este caso de uso permite a los diferentes usuarios ingresar al sistema verificando su existencia.

- **Ingresar programación académica:** este caso de uso es el que permite a los usuarios con privilegios ingresar la información relacionada con la programación académica: asignaturas, aulas, departamentos, escuelas, especialidades, pensum, profesores y programación.
- **Validar data:** este caso de uso permite al sistema controlar el ingreso de datos por parte de los usuarios controlando así el formato de la información y el correcto uso del sistema.

- **Consultar programación académica:** este caso de uso permite visualizar la información relacionada con la programación académica: asignaturas, aulas, departamentos, escuelas, especialidades, mapa de aulas, pensum, profesores y programación.

- **Modificar programación académica:** este caso de uso permite editar los campos de las asignaturas, las aulas, los departamentos, las escuelas, las especialidades y los profesores.

- **Eliminar programación académica:** este caso de uso permite borrar registros de las asignaturas, las aulas, los departamentos, las escuelas, las especialidades, el pensum, los profesores y la programación.

- **Generar reportes:** este caso de uso permite imprimir o guardar en formato digital el pensum y la programación.

- **Gestionar usuarios:** este caso de uso permite al personal que labora en el Centro de Computación Académica crear grupos de usuarios y otorgar privilegios de acceso.

- **Gestionar BD:** se basa en limpiar la tabla de la programación de la base de datos, realizando antes una copia de seguridad.

En la figura 3.2 se muestra el diagrama de casos de uso principal del sistema.

Figura 3.2. Diagrama de casos de uso principal del sistema

3.4.5.3 Descripción de los Casos de Uso

El diagrama de casos de uso principal permite representar una visión global del sistema, en el se muestran las funciones primordiales. A continuación se presentan detalladamente los casos de uso para una mejor comprensión del sistema:

Caso de Uso 1. Entrar al Sistema.

Actores: Usuarios y administrador.
Descripción: Permite al actor entrar al sistema.

Pre-Condición:

- Ninguna

Flujo Principal:

1. El usuario invoca al caso de uso entrar al sistema.
2. El usuario ingresa su nombre de usuario y su contraseña.
3. El sistema verifica los datos y dependiendo del grupo en el que se encuentre el sistema le proporciona los permisos de acceso.
4. Finaliza el caso de uso.

Flujo Alterno:

- El usuario puede salir o cancelar el ingreso al sistema.

Nombre del caso de uso 2: Ingresar programación académica.

Actores: Profesores y Administrador.

Descripción: Permite al actor ingresar los datos relacionados con la programación académica.

Pre-Condición:

- El usuario debe haberse identificado en el sistema.
- El usuario debe haber inicializado sesión, teniendo acceso así al menú principal.

Flujo Principal:

1. El usuario invoca cualquiera de los casos de uso siguientes: ingresar asignaturas, ingresar aulas, ingresar departamentos, ingresar escuelas ingresar especialidades, ingresar pensum, ingresar profesores, ingresar programación.
2. El sistema carga el formulario de solicitud.
3. Desde la base de datos, el sistema carga automáticamente los datos de interés.
4. El usuario debe ingresar los datos que requiere el formulario.
5. El sistema verifica la data ingresada.
6. Si el usuario está de acuerdo con los datos ingresados presiona el botón cargar y los datos se guardan en el servidor ubicado en el centro de computación académica.
7. Si el usuario no está de acuerdo presiona el botón borrar y vuelve a ingresar los datos.
8. Finaliza el caso de uso.

Flujo Alterno:

- El usuario puede salir o cancelar el ingreso de datos si así lo desea.

- El usuario puede seleccionar cualquier ítem del menú principal.

En la figura 3.3 se muestra el diagrama de casos de uso ingresar datos de programación académica.

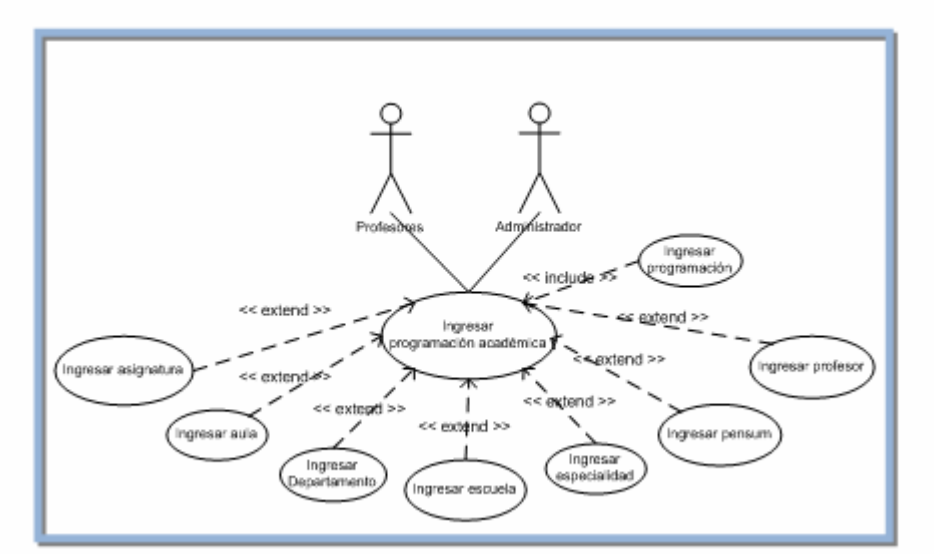


Figura 3.3. Diagrama de casos de uso ingresar programación académica

Nombre del caso de uso 3: Consultar programación académica.

Actores: Usuarios y Administrador.

Descripción: Permite al actor visualizar la información relacionada con la programación académica, por supuesto después de que estos datos se encuentran almacenados en el servidor. Los estudiantes solo podrán visualizar la programación y el pensum, la coordinación académica, los profesores y el administrador podrán visualizar todo.

Pre-Condición:

- El usuario debe haberse identificado en el sistema.
- El usuario debe haber inicializado sesión, teniendo acceso así al menú principal.

Flujo Principal:

1. El usuario invoca cualquiera de los casos de uso siguientes: consultar asignaturas, consultar aulas, consultar departamentos, consultar escuelas, consultar especialidades, consultar mapa de aulas, consultar pensum, consultar profesores, consultar programación.
2. El sistema permite visualizar los datos relacionados con el caso de uso seleccionado.
3. Finaliza el caso de uso (ver figura 3.4).

Flujo Alterno:

- El usuario puede salir o cancelar de la opción procesar.
- El usuario puede seleccionar cualquier ítem del menú principal.

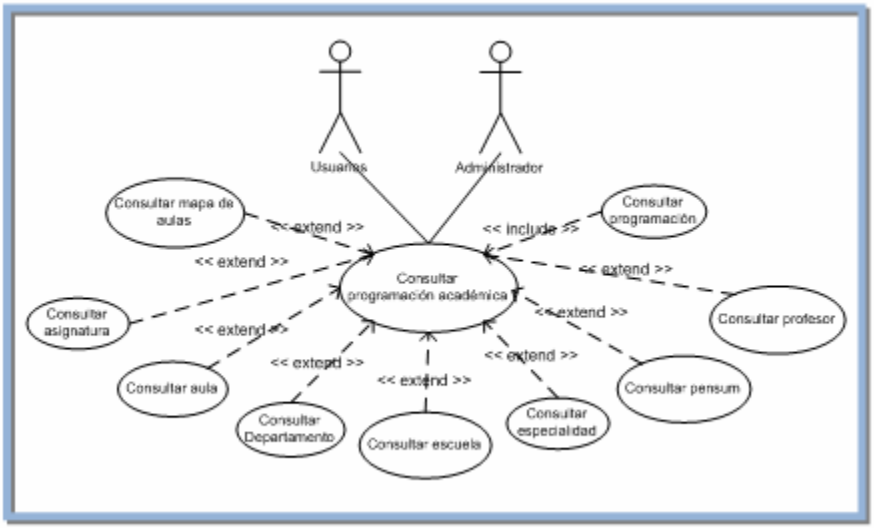


Figura 3.4. Diagrama de casos de uso consultar programación académica

Nombre del caso de uso 4: Modificar programación académica.

Actores: Profesores y Administrador.

Descripción: Permite al actor modificar los campos de algunos registros de la programación académica, por supuesto después de que la misma se encuentra almacenada en el servidor.

Pre-Condición:

- El usuario debe haberse identificado en el sistema.
- El usuario debe haber inicializado sesión, teniendo acceso así al menú principal.

Flujo Principal:

1. El usuario invoca cualquiera de los casos de uso siguientes: modificar asignaturas, modificar aulas, modificar departamentos, modificar escuelas, modificar especialidades, modificar profesores.
2. El usuario selecciona un registro.
3. El usuario selecciona la opción de editar.
4. El sistema verifica la data ingresada.
5. El sistema envía un mensaje de confirmación.
6. El sistema actualiza los cambios generados por parte del usuario los datos se guardan en el servidor ubicado en el centro de computación académica.
7. El actor puede consultar los cambios realizados.
8. Finaliza el caso de uso.

Flujo Alterno:

- El usuario puede salir o cancelar de la opción procesar.

- El usuario puede seleccionar cualquier ítem del menú principal.

En la figura 3.5 se muestra el diagrama de casos de uso modificar programación académica.

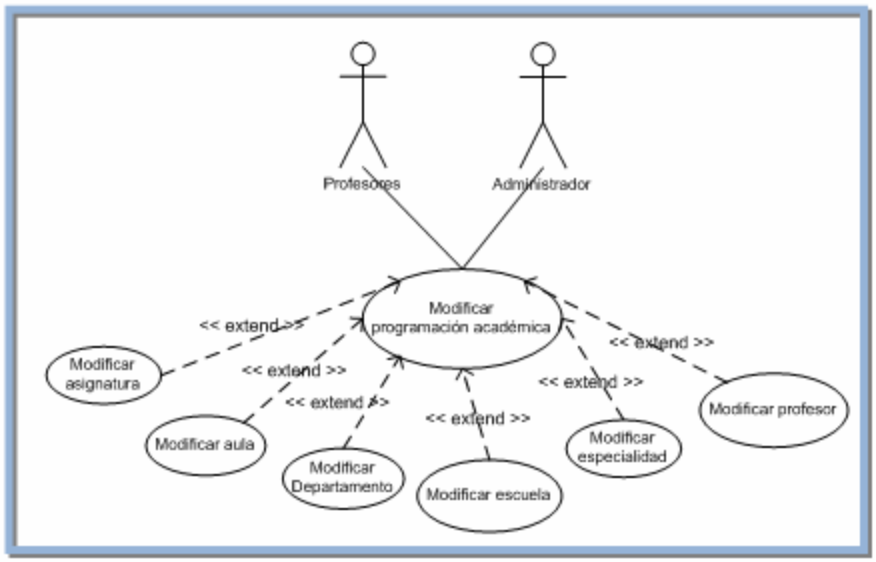


Figura 3.5. Diagrama de casos de uso modificar programación académica

Nombre del caso de uso 5: Eliminar programación académica.

Actores: Profesores y Administrador.

Descripción: Permite al actor eliminar registros de la programación académica, por supuesto después de que la misma se encuentra almacenada en el servidor.

Pre-Condición:

- El usuario debe haberse identificado en el sistema.

- El usuario debe haber inicializado sesión, teniendo acceso así al menú principal.

Flujo Principal:

1. El usuario invoca cualquiera de los casos de uso siguientes: eliminar asignaturas, eliminar aulas, eliminar departamentos, eliminar escuelas, eliminar especialidades, eliminar pensum, eliminar profesores, eliminar programación.
2. El sistema permite visualizar los datos relacionados del caso de uso seleccionado.
3. El usuario selecciona el registro que desea eliminar.
4. El usuario selecciona la opción eliminar.
5. El sistema envía un mensaje de confirmación.
6. Si la respuesta es positiva por parte del usuario los datos se actualizan en el servidor ubicado en el centro de computación académica, en caso contrario no se realiza ninguna operación.
7. El actor puede consultar los cambios realizados.
8. Finaliza el caso de uso.

Flujo Alterno:

- El usuario puede salir o cancelar de la opción procesar.
- El usuario puede seleccionar cualquier ítem del menú principal.

En la figura 3.6 se muestra el diagrama de casos de uso eliminar programación académica.

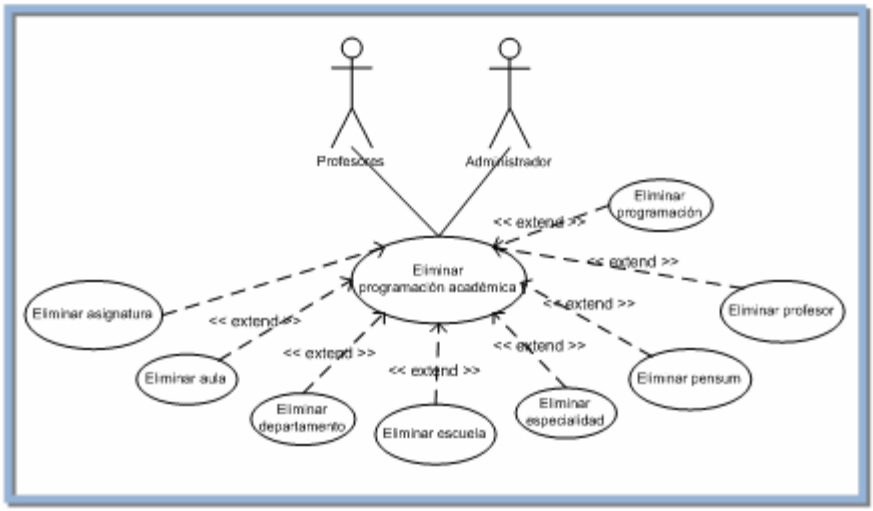


Figura 3.6. Diagrama de casos de uso eliminar la programación académica

Nombre del caso de uso 6: Generar Reportes.

Actores: Usuarios y Administrador.

Descripción: Permite al actor generar los reportes del pensum y de la programación académica.

Pre-Condición:

- El usuario debe haberse identificado en el sistema.
- El usuario debe haber inicializado sesión, teniendo acceso así al menú principal.

Flujo Principal:

1. El usuario invoca el caso de uso generar reportes.
2. El sistema genera el reporte.

3. El usuario a través del programa para visualizar PDF selecciona la opción de guardar o de imprimir.
4. Se guarda o se imprime el documento.
5. Finaliza el caso de uso.

Flujo Alterno:

- El usuario puede salir.
- El usuario puede seleccionar cualquier ítem del menú principal.

En la figura 3.7 se muestra el diagrama de casos de uso generar reportes.

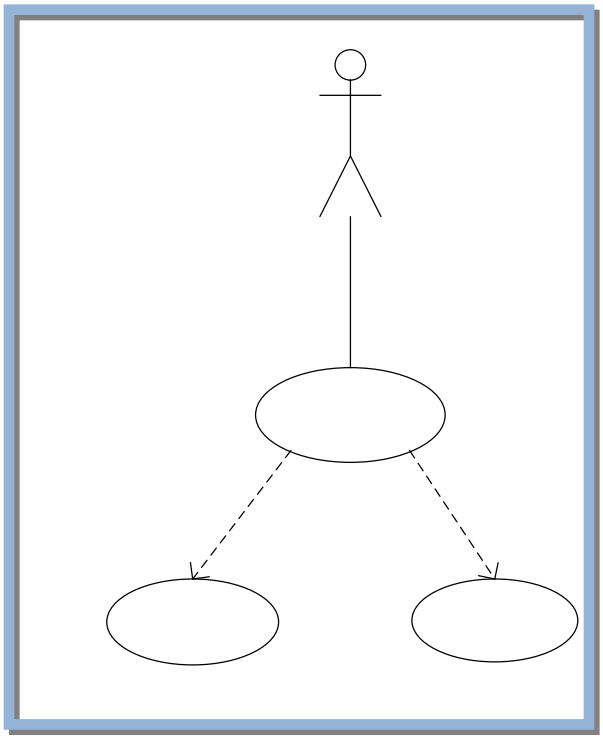


Figura 3.7. Diagrama de casos de uso generar reportes

Nombre del caso de uso 7: Gestionar usuarios.

Actores: Administrador.

Descripción: Este caso de uso permite llevar el control de los privilegios y grupos de usuarios beneficiados con el sistema.

Pre-Condición:

- El usuario debe haberse identificado en el sistema.
- El usuario debe haber inicializado sesión, teniendo acceso así al menú principal.

Flujo Principal:

1. El usuario invoca el caso de uso permisos de usuarios.
2. El actor puede crear un nuevo grupo de usuarios y otorgarle los privilegios de acceso al sistema.
3. Finaliza el caso de uso.

Flujo Alterno:

- El usuario puede salir.
- El usuario puede seleccionar cualquier ítem del menú principal.

Nombre del caso de uso 8: Gestionar BD.

Actores: Administrador.

Descripción: Este caso de uso permite eliminar la información que se encuentra almacenada en la tabla programación de la base de datos, realizando antes una copia de seguridad.

Pre-Condición:

- El usuario debe haberse identificado en el sistema.
- El usuario debe haber inicializado sesión, teniendo acceso así al menú principal.

Flujo Principal:

1. El usuario invoca el caso de uso gestionar BD.
2. El actor realiza la copia de seguridad de la tabla programación.
3. El actor elimina los datos de la tabla programación.
4. Finaliza el caso de uso.

Flujo Alterno:

- El usuario puede salir.
- El usuario puede seleccionar cualquier ítem del menú principal.

3.5 Análisis

En esta etapa de la fase de inicio se analizan los requisitos que se describieron en los diagramas de casos de uso, refinándolos y estructurándolos. El resultado de este flujo es conseguir una compresión más precisa de los requisitos y una descripción de los mismos que sea más fácil de mantener y que ayude a estructurar el sistema completo, incluyendo su arquitectura.

3.5.1 Diagramas de Clases de Análisis

Describen como se lleva a cabo y se ejecuta un caso de uso determinado en términos de las clases de análisis y de sus objetos del análisis en interacción,

centrándose en los requisitos funcionales. Forman las posibles clases del diseño, ajustándolas con los estereotipos básicos: De Interfaz, de Control y de Entidad.

3.5.1.1 Clases de Interfaz

Estas se utilizan para modelar interacciones entre el sistema y sus actores, es decir, usuarios y sistemas externos. Esta interacción a menudo implica recibir información y peticiones de los usuarios y los sistemas externos. Las clases de interfaz modelan las partes del sistema que dependen de sus actores, lo cual implica que clasifican y reúnen los límites del sistema. Por tanto, un cambio en una interfaz de usuario o una interfaz de comunicación queda normalmente aislado en una o más clases de interfaz. Las clases de interfaz representan a menudo abstracciones de ventanas, formularios, paneles, interfaces de comunicación, interfaces de impresoras, sensores, terminales y API

En la tabla 3.1 se muestra las clases de interfaz.

Clases	Definición	Tipo
:Entrar al sistema	Esta clase permite a los diferentes usuarios: estudiantes, profesores, coordinación y computación académica accesar al sistema, el actor obtiene acceso al sistema.	Interfaz
:Ingresar programación	Esta clase permite a los profesores y al administrador interactuar con el sistema, ingresando la información relacionada con la programación	

académica	académica: asignaturas, aulas, departamentos, escuelas, especialidades, pensum, profesores y programación.	Interfaz

Tabla 3.1. Clases de Interfaz 1/2

Clases	Definición	Tipo
:Consultar programación académica	Esta clase permite a los usuarios y al administrador interactuar con el sistema, consultando la información relacionada con la programación académica: asignaturas, aulas, departamentos, escuelas, especialidades, mapa de aulas, pensum, profesores y programación. Los estudiantes solo podrán consultar el pensum y la programación, la coordinación académica solo la programación.	Interfaz
:Modificar programación académica	Esta clase permite a los profesores y al administrador interactuar con el sistema, modificando la información relacionada con la programación académica: asignaturas, aulas, departamentos, escuelas, especialidades y profesores.	Interfaz
:Eliminar	Esta clase permite a los profesores y al administrador interactuar	

programación académica	con el sistema, eliminando la información relacionada con la programación académica: asignaturas, aulas, departamentos, escuelas, especialidades, pensum, profesores y programación.	Interfaz
:Generar reportes	Esta clase permite a los usuarios y al administrador interactuar con el sistema, generando un reporte digital o impreso del pensum y la programación académica.	Interfaz
:Permisos de usuarios	Esta clase permite al administrador crear grupos de usuarios y otorgarles privilegios de accesos al sistema.	Interfaz
:Gestionar BD	Esta clase permite al administrador borrar la información de la tabla programación realizando antes una copia de seguridad.	Interfaz

Tabla 3.1. Clases de Interfaz 2/2

3.5.1.2 Clases de Control

Representan coordinación, secuencia, transacciones, y control de otros objetos y se usan con frecuencia para encapsular el control de un caso de uso en concreto. Los aspectos dinámicos del sistema se modelan con las clases de control, debido a que ellas manejan y coordinan las acciones y los flujos de control principales, y delegan trabajo a otros objetos, es decir, objetos de interfaz y de entidad. En la tabla 3.2 se muestran las clases de control del sistema.

Clases	Definición	Tipo
:Verificar usuario	Esta clase permite comprobar que el usuario esta activo y tiene acceso al sistema.	Control
:Validar datos	Esta clase permite verificar el formato de los datos.	Control
:Visualizar	Esta clase permite gestionar la visualización de los datos ingresados en el sistema.	Control
:Editar	Esta clase permite gestionar la modificación de algunos campos en los registros del sistema.	Control
:Borrar	Esta clase permite gestionar la eliminación de los registros de la informacion del sistema.	Control
:Reportes	Esta clase permite gestionar los reportes de la programación y el pensum.	Control
:Privilegios	Esta clase permite gestionar los grupos de usuarios y los privilegios para cada usuario.	Control
:Gestionar	Esta clase permite vaciar la tabla de la programación realizando antes una	Control

	copia de seguridad.	

Tabla 3.2. Clases de Control

3.5.1.3 Clase de Entidad

Se utilizan para modelar información que posee una vida larga y que es a menudo persistente. Las clases de entidad modelan la información y el comportamiento asociado de algún fenómeno o concepto, como una persona, un suceso o un objeto. En la tabla 3.3 se muestran las clases de entidad del sistema.

Clases	Definición	Tipo
:Acceso al sistema	Esta clase permite generar el acceso correspondiente a cada tipo de usuario.	Entidad
:Guardar programación académica	Esta clase permite almacenar en el servidor la data ingresada en el sistema relacionada con la programación académica.	Entidad
:Ver programación académica	Esta clase permite visualizar desde el servidor la data ingresada en el sistema relacionada la programación académica.	Entidad
:Actualizar programación académica	Esta clase permite guardar los cambios realizados en la programación académica al modificar campos o eliminar registros.	Entidad
:Crear	Esta clase permite crear grupos	Entidad

grupos y permisos	de usuarios y otorgar los privilegios de acceso al sistema.	
:Control	Esta clase permite la eliminación de los registros de la tabla de la programación académica de la base de datos realizando antes una copia de seguridad.	Entidad

Tabla 3.3. Clase de entidad

3.5.2 Diagramas de Clases de Análisis para los casos de uso del sistema.

A continuación se describen los diagramas de clases de análisis para cada una de los casos de uso del sistema:

3.5.2.1 Diagrama de clases de análisis para el caso de uso entrar al sistema.

Se identifico la clase de interfaz entrar al sistema, el usuario interactúa con esta clase para solicitar el acceso al sistema mediante su usuario que es el identificador y su contraseña, todos los usuarios tienen acceso a esta clase. Mediante la clase de control verificar usuario, el actor podrá entrar a la página principal obteniendo así acceso al sistema con sus respectivos permisos (ver figura 3.8).

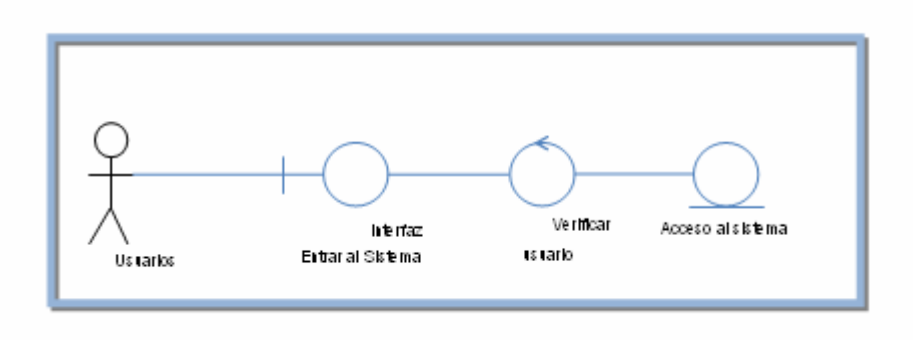


Figura 3.8. Diagrama de clases de análisis para el caso de uso entrar al sistema

3.5.2.2 Diagrama de clases de análisis para el caso de uso ingresar programación académica.

Se identifico la clase de interfaz ingresar programación académica: asignaturas, aulas, departamentos, escuelas, especialidades, pensum, profesores y programación, el usuario interactúa con esta clase cargando la información relacionada con la programación académica, solo los profesores y el administrador tienen acceso a esta clase. Toda la data ingresada es verificada por la clase de control validar datos, para corregir de esta manera cualquier error por parte de los usuarios y mantener el formato de los datos, finalmente los datos son guardados en el servidor esto gracias a la clase de entidad guardar programación académica (ver figura 3.9).

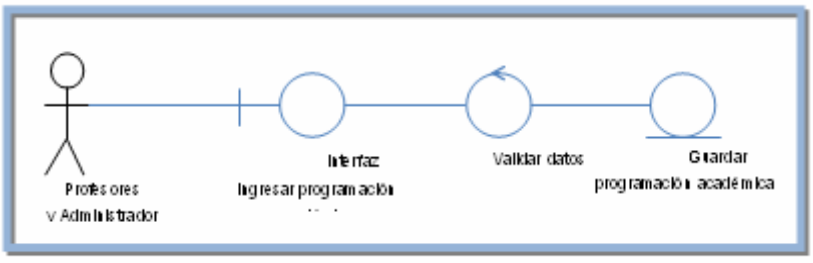


Figura 3.9. Diagrama de clases de análisis para el caso de uso ingresar programación académica

3.5.2.3 Diagrama de clases de análisis para el caso de uso consultar los datos de la programación académica.

Se identifico la clase de interfaz consultar información de la programación académica: asignaturas, aulas, departamentos, escuelas, especialidades, mapa de aulas, pensum, profesores y programación, (los estudiantes solo podrán visualizar la programación y el pensum). Tenemos la interfaz de control visualizar la cual permite generar la clase de entidad ver programación académica, (ver figura 3.10).

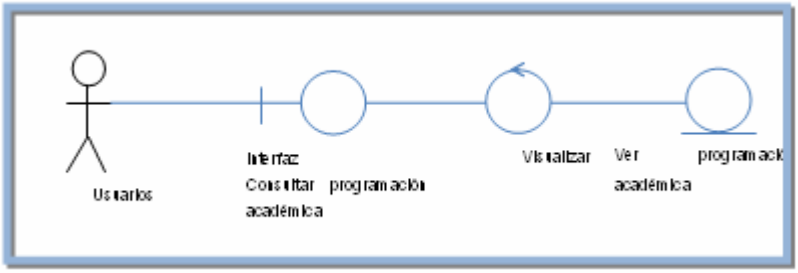


Figura 3.10. Diagrama de clases de análisis para el caso de uso consultar programación académica.

3.5.2.4 Diagrama de clases de análisis para el caso de uso modificar la programación académica.

Se identifico la clase de interfaz modificar programación académica: académica: asignaturas, aulas, departamentos, escuelas, especialidades y profesores, solo los profesores y el administrador tienen acceso esta clase. Las clase de control editar permite modificar campos, generando la clase de entidad actualizar programación académica, (ver figura 3.11).

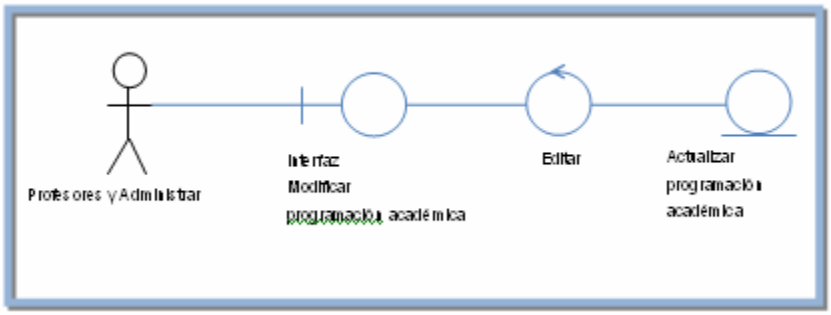


Figura 3.11. Diagrama de clases de análisis para el caso de uso modificar programación académica.

3.5.2.5 Diagrama de clases de análisis para el caso de uso eliminar programación académica.

Se identifico la clase de interfaz eliminar programación académica: académica: asignaturas, aulas, departamentos, escuelas, pensum, especialidades, profesores y programación, solo los profesores y el administrador tienen acceso esta clase. Las clase de control borrar permite eliminar registros, generando la clase de entidad actualizar programación académica, (ver figura 3.12).

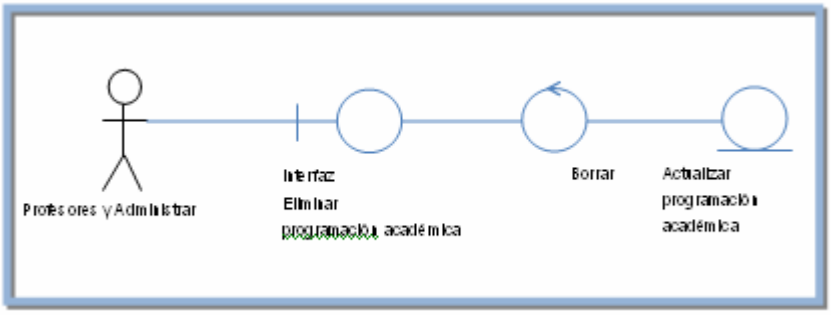


Figura 3.12. Diagrama de clases de análisis para el caso de uso eliminar programación académica.

3.5.2.6 Diagrama de clases de análisis para el caso de uso generar reportes.

Se identifico la clase de interfaz generar reportes, todos los usuarios tienen accesos a esta clase, esta clase genera reportes del pensum y la programación. Mediante la clase de control reportes permite generar un reporte en formato digital o impreso a través de la clase de entidad ver programación académica, (ver figura 3.13).

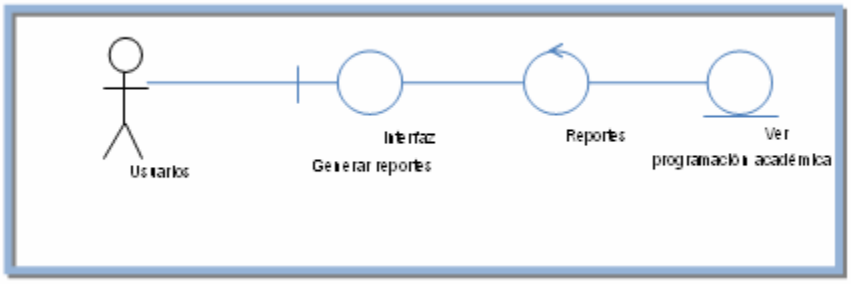


Figura 3.13. Diagrama de clases de análisis para el caso de uso generar reportes de la programación académica.

3.5.2.7 Diagrama de clases de análisis para el caso de uso permisos de usuarios.

Se identifico la clase de interfaz permisos de usuarios, solo el administrador del sistema tiene acceso a esta clase, la cual permite crear grupos de usuarios para otorgarle permisos de acceso al sistema de acuerdo al tipo de usuario. Tenemos la clase de control privilegios que se encarga de generar los permisos a los distintos tipos de usuarios y la clase de control grupos que permite crear los grupos, (ver figura 3.14).

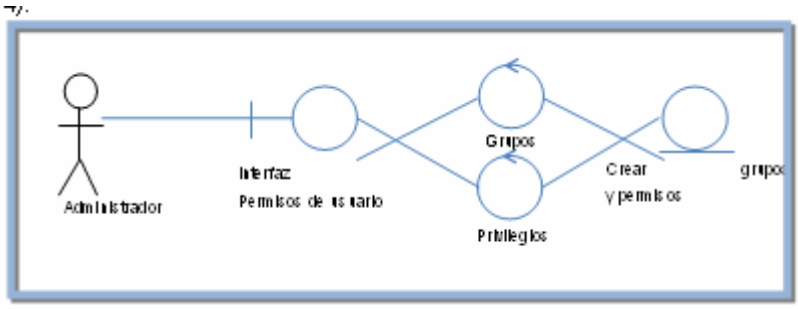


Figura 3.14. Diagrama de clases de análisis para el caso de uso permisos de usuarios

3.5.2.8 Diagrama de clases de análisis para el caso de uso gestionar BD.

Se identifico la clase de interfaz gestionar BD, solo el administrador del sistema tiene acceso a esta clase, la cual permite eliminar la información de la tabla programación de la base de datos realizando antes una copia de seguridad de la misma. Tenemos la clase de control privilegios que se encarga de generar los permisos a los distintos tipos de usuarios, (ver figura 3.15).

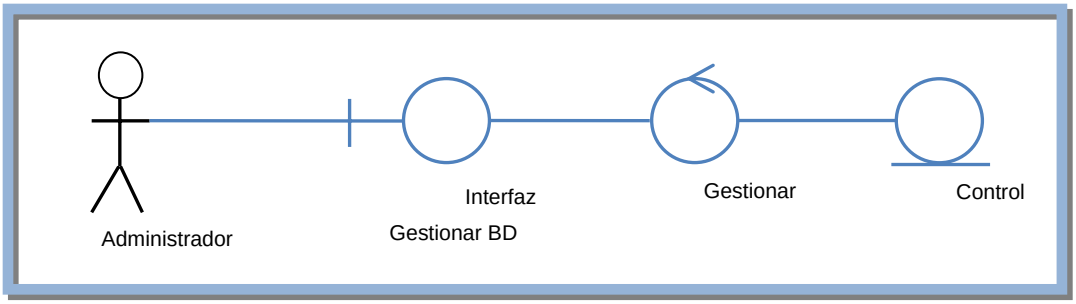


Figura 3.15. Diagrama de clases de análisis para el caso de uso gestionar BD.

3.5.2 Paquetes de Análisis

Los paquetes de análisis proveen un medio para organizar el modelo de análisis en partes manejables. Éstos pueden constar de realizaciones de casos de uso y otros paquetes del análisis. Estos paquetes deben minimizar sus dependencias.

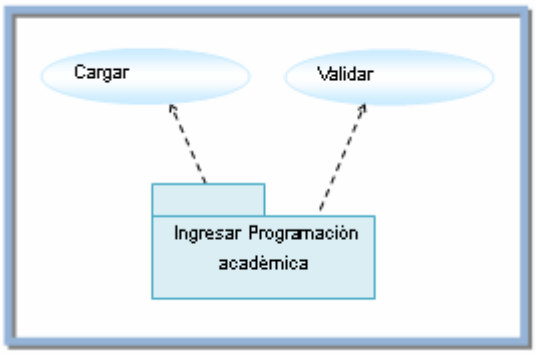


Figura 3.16. Paquete de análisis Ingresar programación académica

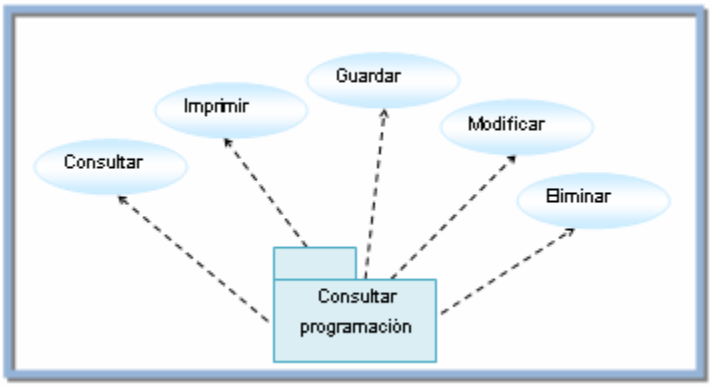


Figura 3.17. Paquete Consultar programación académica.

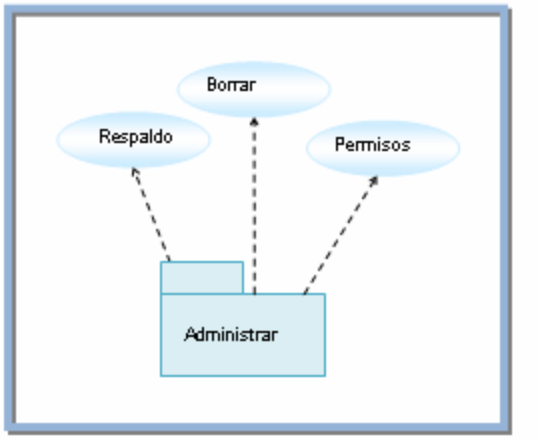


Figura 3.18. Paquete de administrar sistema

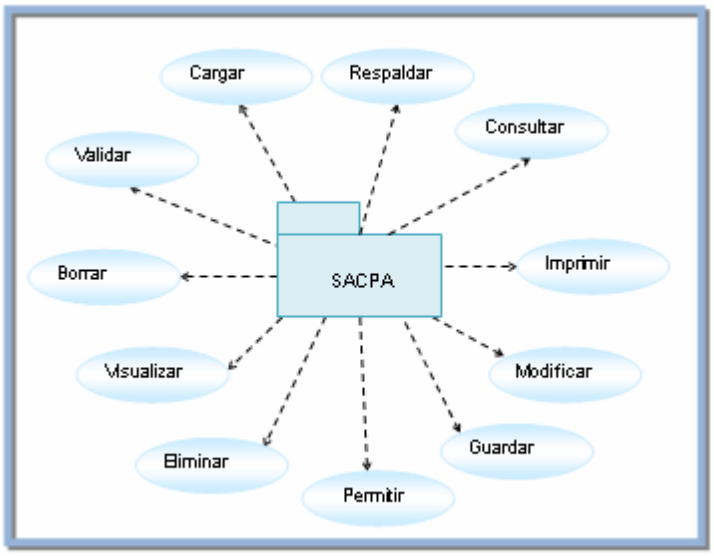


Figura 3.19. Paquete de análisis SACPA

3.6 Diseño del sistema

3.6.1 Arquitectura de una aplicación web

Toda aplicación web está comprendida por tres componentes básicos: el servidor Web, el motor de ejecución del programa y el servidor de base de datos, todos a su vez requieren de un servidor de aplicación, el cual es requerido para la ejecución de la aplicación.

La configuración básica más utilizada, comprende el servidor web, el motor de ejecucion del programa y la base de datos instalados en un solo servidor físico es decir, una misma máquina. Los host o clientes o estaciones de trabajo, están interconectados al servidor a través de la intranet y pueden estar conectados a Internet. Si tal conexión no es suministrada, solo los usuarios dentro de la intranet corporativa pueden acceder a la aplicación (ver figura 3.20).

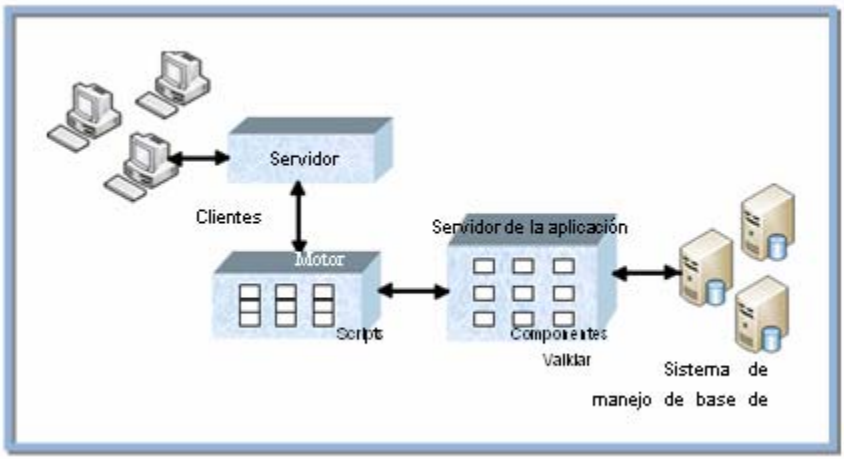


Figura 3.20. Arquitectura Básica de una aplicación Web

3.7 Evaluación de la fase de inicio

Ésta fase se realizó siguiendo explícitamente los lineamientos del proceso unificado de software. Fue un período breve, en cuanto a tiempo de realización, donde se hizo una sola iteración para lograr los objetivos de la misma. En esta iteración se trabajaron los flujos de requisito, análisis y, muy brevemente, diseño. En esta etapa se logró expresar el contexto del sistema a través del modelo de dominio, el cual fue el punto clave para iniciar la captura de requisitos para el control de la programación académica. Se definieron las clases conceptuales que representan el negocio, y se describieron para comprender este modelo conceptual. Además del modelo de dominio, se desarrollaron los casos de uso, pieza muy importante en esta fase ya que colaboró notablemente con el futuro análisis de requisitos.

El Modelo de Casos de uso de este sistema fue bastante extenso, a pesar de ser la fase de inicio, debido a la complejidad del sistema. Se definieron los actores, se relacionaron con los respectivos casos de uso, y la función que cumple cada uno en el sistema. Posteriormente, trabajamos con el flujo de análisis usando como entrada los objetos obtenidos en la captura de requisitos. Se obtuvo un esbozo del sistema usando clases de análisis y realizaciones de casos de uso, que guiaron al siguiente paso, en el flujo de diseño, a definir la arquitectura del sistema.

También se presenta la propuesta y diseño de una arquitectura robusta, estable y candidata a ser implementada en las fases venideras. Se logró establecer una visión del sistema para el control de la programación académica, como está estructurado, y un camino viable y cómodo para continuar con las próximas fases. Por último se indicaron los riesgos que puede presentar el sistema y la viabilidad del proyecto, se logró establecer que el proyecto es viable y por tanto se decidió continuar con el desarrollo del Software.

.

CAPITULO IV. FASE DE ELABORACIÓN

4.1 Introducción

El propósito de esta fase es analizar el dominio del problema, establecer los principios de la arquitectura, desarrollar el plan del proyecto y eliminar los mayores riesgos.

En esta fase se construye un prototipo de la arquitectura, que debe evolucionar en iteraciones sucesivas hasta convertirse en el sistema final. Este prototipo debe contener los Casos de Uso críticos identificados en la fase de inicio. Hasta ahora se tiene en un sentido amplio la arquitectura que se desea implementar. El sistema es complejo, ya que son muchas las variables que hay que tomar en cuenta.

En esta fase nos encontramos con un modelo de casos de uso ligeramente modificado, porque el inicial no cubría todas las expectativas.

Al concluir esta fase se tiene la información necesaria de todos los requisitos funcionales y la elaboración de un modelo de análisis y de diseño sólidos como para llevar a cabo la construcción de la arquitectura base del sistema. Cabe destacar que SACPA es una aplicación Web, por lo que se escogió como guía metodológica y técnica de modelado de gestión de contenidos WebML.

4.2 Requisitos del sistema

La captura de requisitos en la fase de elaboración se lleva a cabo con la finalidad de refinar los requisitos ya obtenidos, contemplando nuevos requisitos que fueron pasados por alto en la fase de inicio.

4.2.1 Requisitos Funcionales

El sistema debe permitir consultar las cargas de profesores y generar reportes de las mismas, esta carga proporciona información a cada uno de los profesores activos de las materias que dictaran durante el semestre académico**.**

4.2.2 Requisitos No Funcionales

En esta fase no se identificaron nuevos requisitos no funcionales debido a que los requerimientos necesarios fueron identificados en la fase de inicio.

4.2.3 Modelo de Caso de Uso

Debido a que en la fase de inicio los casos de usos no estaban completos, han sido identificados dos nuevos casos de uso, lo que permite conocer más a fondo la estructura del sistema.

4.2.3.1 Identificación de Actores

En esta fase no se identificaron nuevos actores, ya que los actores que interactúan con los nuevos casos de uso ya se encuentran identificados en la fase de inicio.

4.2.3.2 Identificación de Casos de Uso

- **Consultar cargas de profesores:** Este permite generar la carga de cada uno de los profesores activos para poder visualizarla, esta carga proporciona a los profesores información sobre las asignaturas que dictaran durante cada periodo académico.

- **Reportes de cargas:** Este permite imprimir la carga y/o guardarla en formato digital, para cada profesor y si se requiere para todos los profesores de un mismo departamento (ver figura 4.1).

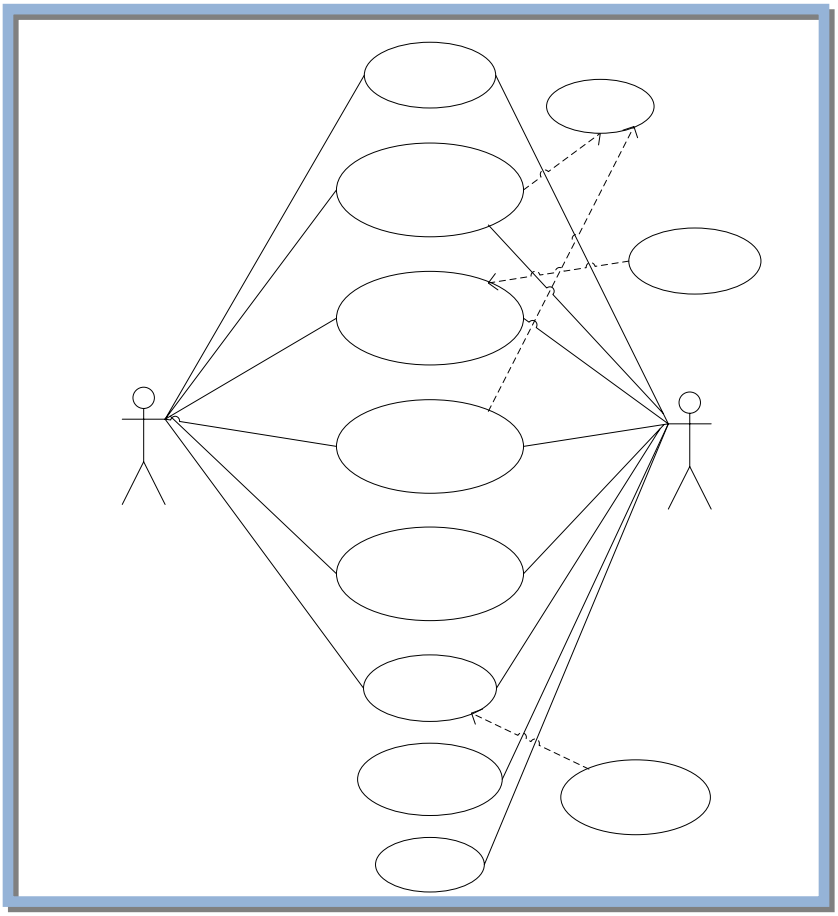


Figura 4.1. Diagrama de casos de uso principal del sistema modificado

Nombre del caso de uso: Consultar cargas de profesores.

Actores: Profesores y Administrador.

Descripción: Permite al actor visualizar la carga de los profesores la cual informa sobre las materias que dictara cada profesor durante el periodo académico.

Pre-Condición:

- El usuario debe haberse identificado en el sistema.
- El usuario debe haber inicializado sesión, teniendo acceso así al menú principal.

Flujo Principal:

1. El usuario invoca al caso de uso consultar cargas de profesores.
2. El sistema permite visualizar la carga solicitada.
3. Finaliza el caso de uso.

Flujo Alterno:

- El usuario puede salir.
- El usuario puede seleccionar cualquier ítem del menú principal

Nombre del caso de uso: Reportes de cargas.

Actores: Profesores y Administrador.

Descripción: Permite al actor imprimir una o varias cargas o si lo prefiere guardarlas en formato digital.

Pre-Condición:

- El usuario debe haberse identificado en el sistema.

- El usuario debe haber inicializado sesión, teniendo acceso así al menú principal.

Flujo Principal:

1. El usuario invoca al caso de uso Generar reportes.
2. El sistema permite realizar reportes ya sea en formato digital o impreso
3. Finaliza el caso de uso.

Flujo Alterno:

- El usuario puede salir.
- El usuario puede seleccionar cualquier ítem del menú principal

4.2.4 Validaciones

Es el proceso de comprobar la precisión de los datos; conjunto de reglas que se pueden aplicar a un control para especificar el tipo y el intervalo de datos que los usuarios pueden especificar. En este caso se valida en ingreso de los datos para mantener el formato que el sistema requiere.

4.2.5 Prototipo de Interfaz de Usuario

El Prototipo de Interfaz Usuario refleja a rasgos muy generales como será la interfaz de la aplicación, esto por medio de un esquema de las interfaces principales. Los prototipos se modelan teniendo como guía los casos de uso que se han venido estudiando a lo largo de la fase de inicio y elaboración.

4.2.5.1 Prototipo de Interfaz Entrar al sistema

En este prototipo se muestra cómo serán ubicados los bloques principales que conformaran la interfaz para entrar al sistema, esta será la primera pantalla con la que el usuario interactuará.

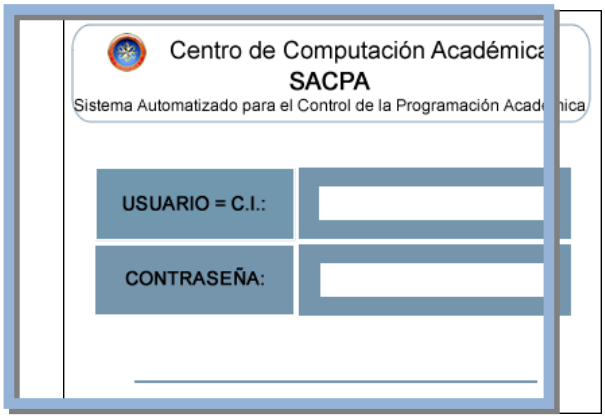


Figura 4.2. Prototipo de Interfaz para entrar al sistema

4.2.5.2 Prototipo de Interfaz Principal

En el prototipo mostrado a continuación se observan los bloques principales que van a componer la Interfaz Principal del sistema.

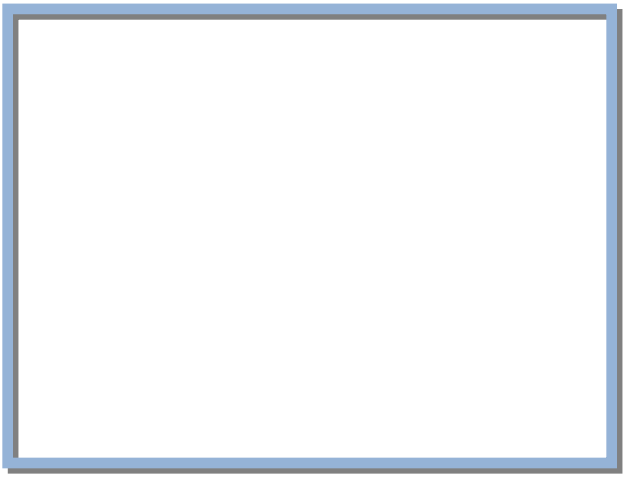


Figura 4.3. Prototipo de Interfaz Principal

4.2.6 Modelo de Hipertextos

El objetivo del modelo de hipertexto es especificar la organización de las interfaces de aplicaciones Web. Para que sean efectivas, tales especificaciones deben ser capaces de transmitir en una forma fácil e intuitiva aspectos como la división lógica de las aplicaciones en módulos de alto nivel, cada uno de los cuales incorpora un conjunto de funciones coherentes a una clase especifica de usuarios, la partición de los módulos de alto nivel en sub-módulos, para una mejor organización en aplicaciones extensas, y la topología de hipertexto actual, en términos de pagina, hecha de elementos contenidos, y enlazada para soportar la navegación e interacción por parte del usuario (ver tablas 4.1 a 4.6).

4.2.6.1 Identificación de Sites Views

- **Site View Ingresar**

Nombre del Site View	Ingresar
Descripción	Agrupa las páginas referidas al ingreso de la información relacionada con la programación académica.
Grupo de Usuarios	Profesores y Administrador.
Casos de Uso Involucrados	Ingresar Asignaturas, Ingresar Aulas, Ingresar Departamentos, Ingresar Escuelas, Ingresar Especialidades, Ingresar Pensum, Ingresar Profesores, Ingresar Programación.

Tabla 4.1 Mapa del Site View Ingresar

- **Site View Consultar**

Nombre del Site View	Consultar
Descripción	Agrupa las páginas referidas a consultas de la información relacionada con la programación académica.
Grupo de Usuarios	Usuarios y Administrador
Casos de Uso Involucrados	Consultar Asignaturas, Consultar Aulas, Consultar cargas, Consultar Departamentos, Consultar Escuelas, Consultar Especialidades, Consultar Mapa de aulas, Consultar Pensum, Consultar Profesores, Consultar Programación.

Tabla 4.2 Mapa del Site View Consultar

- **Site View Modificar**

Nombre del Site View	Modificar
Descripción	Agrupa las páginas referidas a modificaciones, de algunos campos pertenecientes a los registros de la programación académica.
Grupo de Usuarios	Profesores y Administrador
Casos de Uso Involucrados	Modificar Asignaturas, Modificar Aulas, Modificar Departamentos, Modificar Escuelas, Modificar Especialidades, Modificar Profesores.

Tabla 4.3 Mapa del Site View Modificar

- **Site View Eliminar**

Nombre del Site View	Eliminar
Descripción	Agrupa las páginas referidas eliminaciones de registros relacionados con la programación académica.
Grupo de Usuarios	Profesores y Administrador
Casos de Uso Involucrados	Eliminar Asignaturas, Eliminar Aulas, Eliminar Departamentos, Eliminar Escuelas,

	Eliminar Especialidades, Eliminar Pensum, Eliminar Profesores, Eliminar Programación.

Tabla 4.4 Mapa del Site View Eliminar

- **Site View Permisos de Usuarios**

Nombre del Site View	Permisos de usuarios
Descripción	Permite crear grupos de usuarios y asignarle permisos de acceso al sistema.
Grupo de Usuarios	Administrador.
Casos de Uso Involucrados	Permisos de usuarios.

Tabla 4.5 Mapa del Site View Permisos de usuarios

- **Site View Gestionar BD**

Nombre del Site View	Gestionar BD
Descripción	Permite eliminar la información de la tabla de la programación de la base de datos, realizando antes una copia de seguridad.
Grupo de Usuarios	Administrador.
Casos de Uso Involucrados	Gestionar BD.

Tabla 4.6 Mapa del Site View Gestionar BD

4.3 Análisis

En la fase de inicio se realizó un análisis para comprobar la factibilidad de la arquitectura, en esta fase se extiende el análisis hasta el punto de que pueda servir como base a la línea principal de la arquitectura ejecutable.

4.3.1 Identificación de Clases de Análisis

Las clases de análisis identificadas en la fase anterior correspondiente a los casos de uso: Entrar al sistema, Ingresar programación académica, Consultar programación académica, Modificar programación académica, Eliminar programación academica, Generar reportes, Permisos de usuarios y Getionar BD no fueron modificadas y siguen siendo consideradas para esta fase.

4.3.2 Diagramas de Colaboración

Un diagrama de colaboración es una forma alternativa al diagrama de secuencia de mostrar un escenario. Este tipo de diagrama muestra las interacciones entre objetos organizadas entorno a los objetos y los enlaces entre ellos. Los diagramas de secuencia proporcionan una forma de ver el escenario en un orden temporal, qué pasa primero, qué pasa después.

4.3.2.1 Diagrama de colaboración para el caso de uso Entrar al sistema

1) El usuario interactúa con la clase de interfaz: entrar al sistema, a través de esta clase el usuario solicita acceso al sistema mediante su usuario que es el identificador y su contraseña, todos los usuarios tienen acceso a esta clase.
2) Mediante la clase de control: gestionar sesión, el usuario podrá entrar a la página principal obteniendo así acceso al sistema.
3) Ahora mediante la clase de entidad: acceso al sistema, el sistema proporciona acceso a cada tipo de usuario, (ver figura 4.4).

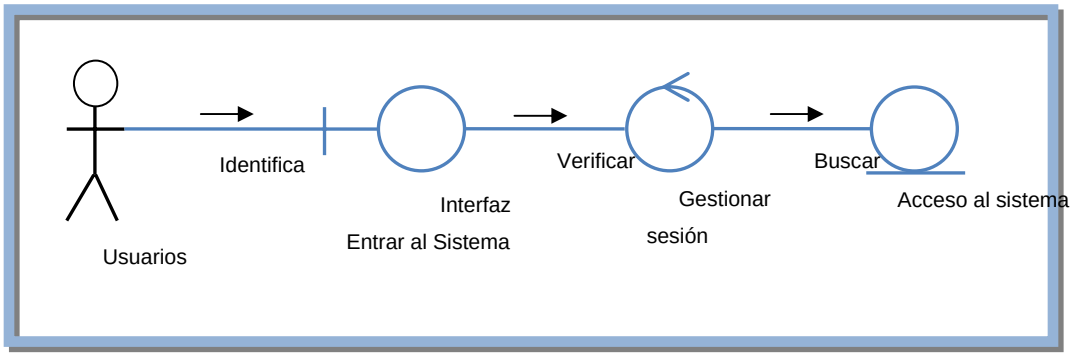


Figura 4.4. Diagrama de colaboración para el caso de uso entrar al sistema

4.3.2.2 Diagrama de colaboración para el caso de uso ingresar programación académica.

1) Se identifico la clase de interfaz: ingresar programación académica: asignaturas, aulas, departamentos, escuelas, especialidades, pensum, profesores y programación. El usuario interactúa con esta clase cargando la data de la programación académica al sistema, solo los profesores y el administrador tienen acceso a esta clase.
2) Toda la información ingresada es validada mediante la clase de control: gestionar datos, para corregir cualquier error por parte de los usuarios y mantener el formato de los datos.
3) Para guardar los datos relacionados con la programación académica en el servidor se usa la clase de entidad: guardar datos de programación académica, (ver figura 4.5).

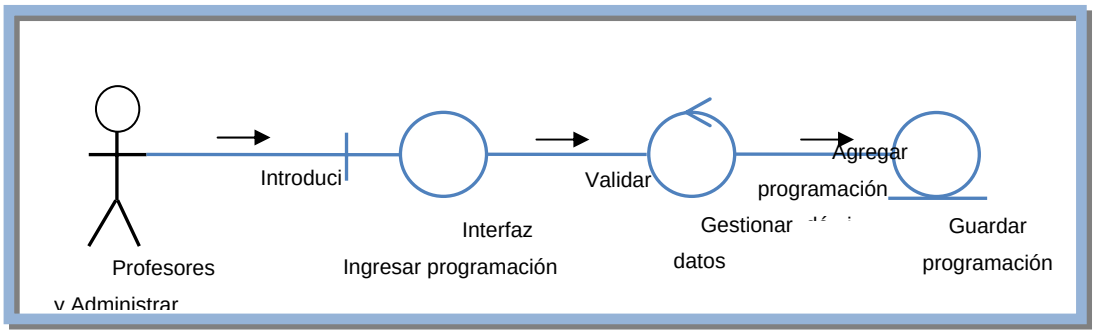


Figura 4.5. Diagrama de colaboración para el caso de uso ingresar programación académica.

4.3.2.3 Diagrama de colaboración para el caso de uso consultar programación académica.

1) Se identifico la clase de interfaz: consultar programación académica: asignaturas, aulas, cargas, departamentos, escuelas, especialidades, pensum, profesores y programación. Todos los usuarios tienen acceso a esta clase, pero los estudiantes solo podrán consultar el pensum y la programación.
2) Luego la clase de control: visualizar, permite ver los datos relacionados con la programación académica.
3) La clase entidad: ver programación académica, permite que los datos se muestren desde la base de datos, (ver figura 4.6).

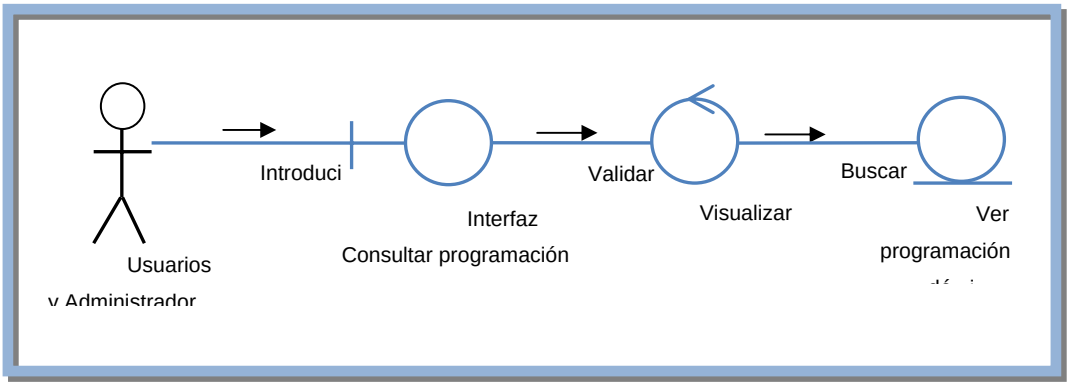


Figura 4.6. Diagrama de colaboración para el caso de uso consultar programación académica.

4.3.2.4 Diagrama de colaboración para el caso de uso modificar programación académica.

1) Se identifico la clase de interfaz modificar programación académica: asignaturas, aulas, departamentos, escuelas, especialidades, pensum y profesores, solo los profesores y el amdinistrador tiene acceso a esta clase que les permite modificar campos de los registros de la programación académica.

2) Luego la clase de control: editar, permite modificar los campos de algunos de los registros.
3) La clase entidad: actualizar programación académica, permite ver la información y guardar los cambios en el servidor, (ver figura 4.7)

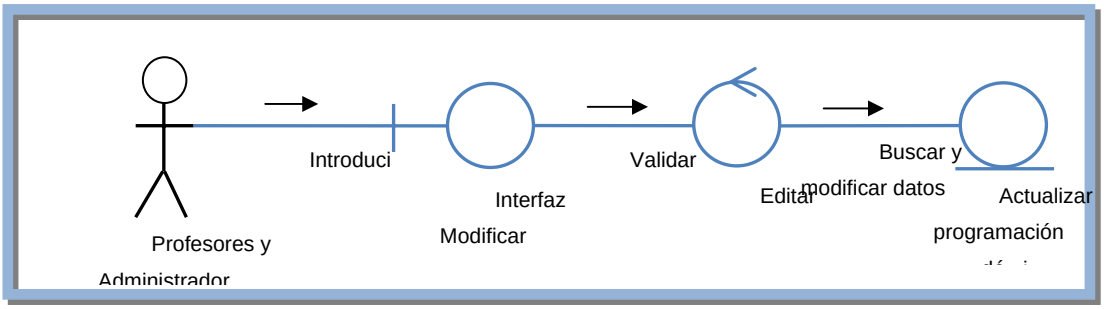


Figura 4.7. Diagrama de colaboración para el caso de uso modificar programación académica.

4.3.2.5 Diagrama de colaboración para el caso de uso eliminar programación académica.

1) Se identifico la clase de interfaz eliminar programación académica: asignaturas, aulas, departamentos, escuelas, especialidades, pensum, profesores y programación, solo los usuarios con privilegios tienen acceso a esta clase que les permite borrar registros de la programación académica.
2) Luego la clase de control: borrar, permite eliminar registros de la programación académica.
3) La clase entidad: actualizar programación académica, permite ver la información y guardar los cambios en el servidor, (ver figura 4.8)

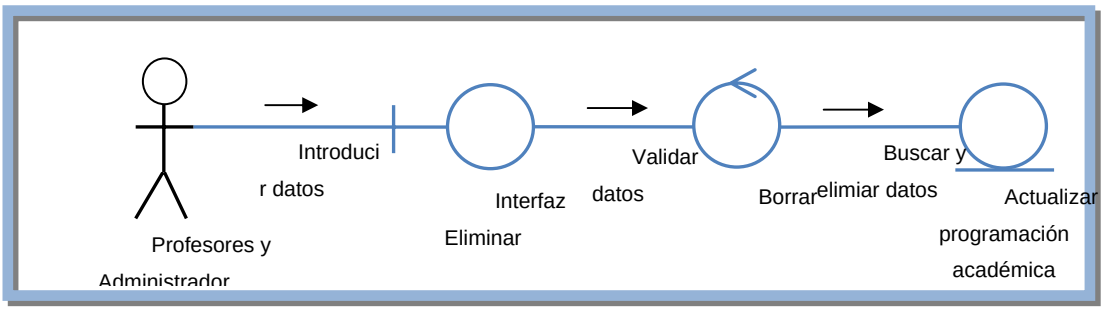


Figura 4.8. Diagrama de colaboración para el caso de uso eliminar programación académica.

4.3.2.6 Diagrama de colaboración para el caso de uso Generar reportes.

1) Se identifico la clase de interfaz: generar reportes, todos los usuarios tienen accesos a esta clase donde podrán generar un reporte impreso o en formato digital de las cargas, el pensum y la programación. Los estudiantes solo pueden generar reportes de la programación y el pensum.
2) Luego tenemos la clase de control: gestionar reportes, que permite imprimir o guardar en formato digital el documento generado por el programa para visualizar pdf.
3) Después tenemos la clase de entidad: ver programación académica, que permite cargar los datos necesarios para generar los reportes, (ver figura 4.9).

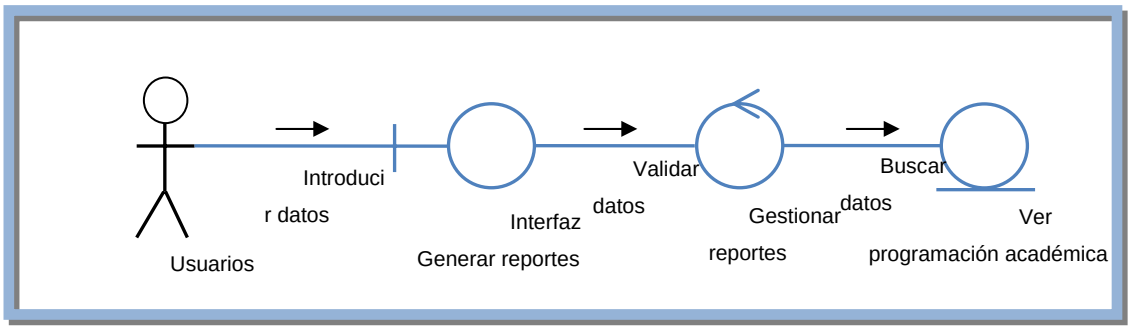


Figura 4.9. Diagrama de colaboración para el caso de uso generar reportes

4.3.2.7 Diagrama de colaboración para el caso de uso permisos de usuarios.

1) El administrador introduce datos a través de la clase de interfaz permisos.
2) Crea el grupo mediante la clase de control grupos.
3) Mediante la clase de entidad se almacena la información del grupo creado.
4) Mediante la clase de control se crean los privilegios al grupo creado.
5) Se genera la clase de entidad guardar ue permite almacenar la información.

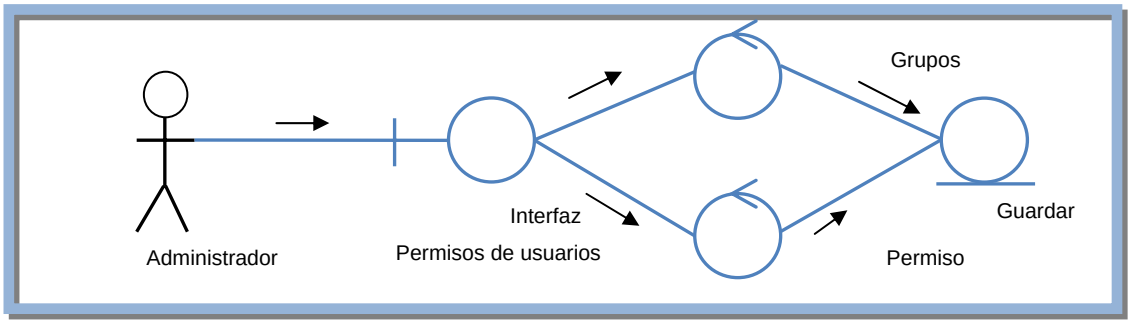


Figura 4.10. Diagrama de colaboración para el caso de uso permisos de usuarios

4.3.2.8 Diagrama de colaboración para el caso de uso gestionar BD.

1) Se identifico la clase de interfaz: gestionar BD, solo el administrador tiene acceso a esta clase, la cual permite limpiar la tabla de la programación de la base de datos, realizando antes una copia de seguridad.
2) Luego tenemos la clase de control: Gestionar tablas, que permite eliminar la información de la tabla programación de la base de datos realizando antes una copia de seguridad de esta tabla.
3) Después tenemos la clase de entidad: control, es la que permite el mantenimiento del sistema, (ver figura 4.11).

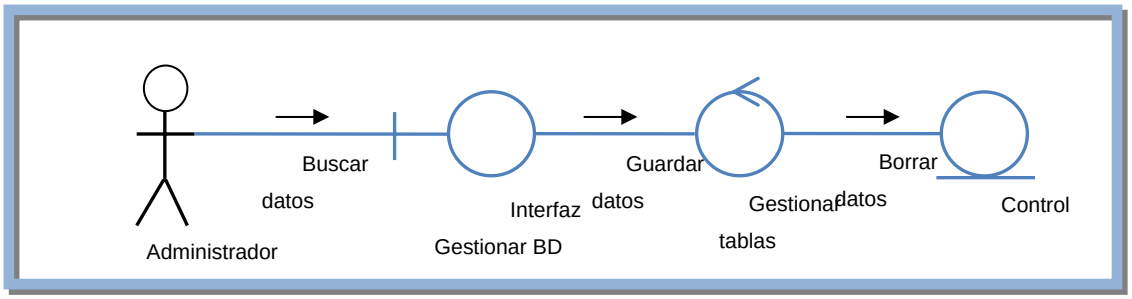


Figura 4.11. Diagrama de colaboración para el caso de uso Gestionar BD

4.4 Diseño

En el flujo de trabajo de diseño se modela el sistema y se encuentra su representación para que soporte todos los requisitos, incluyendo los requisitos no funcionales y otras restricciones. Aquí se toma como entrada el análisis de los requisitos. En el diseño se realizan actividades delicadas y cuidadosas que dan como resultado los componentes que se implementarán en la siguiente fase. Entre ellos tenemos, el diseño de la base de datos, diseño de la arquitectura, diseño de interfaces y Páginas Web.

4.4.1 Base de Datos del Sistema

Como se puede observar en los modelos de gestión diseñados en la sesión anterior, se le hacia un llamado a la base de datos del sistema, bien sea para ingresar o consultar. La base de datos para este sistema ya se encuentra diseñada, por lo tanto el sistema desarrollado debe adaptarse a la estructura de la base de datos existente, de esta forma mantener la compatibilidad con otros sistemas que actualmente utilizan dicha data.

4.4.1.1 Identificación de la Tablas de la Base de Datos del Sistema

En el estudio de la base de datos del sistema fueron identificadas las tablas necesarias para manipular toda la información requerida por el sistema SACPA, las cuales serán mostradas a continuación:

- **Tabla AULAS**

En esta tabla se almacenan los datos relacionados con las aulas de clases de la universidad.

Esquema de la tabla:
AULA varchar(6), UBICACION varchar(30).

- **Tabla DPTO**

En esta tabla se almacenan los datos relacionados con los departamentos académicos existentes en la universidad.

Esquema de la tabla:

DPTO varchar(2), NOM_DPTO varchar(30), ESCUELA varchar(1).

- **Tabla ESCUELAS**

En esta tabla se almacenan los datos relacionados con las escuelas existentes en la universidad.

Esquema de la tabla:
ESCUELA varchar(1), NOM_ESC varchar(30).

- **Tabla ESPEC**

En esta tabla se almacenan los datos relacionados con las distintas carreras que se dictan en la universidad.

Esquema de la tabla:
ESPE varchar(4), ESPECIALID varchar(30), DPTO varchar(2).

- **Tabla MATERIAS**

En esta tabla se almacenan los datos relacionados con las materias que se dictan en cada una de las carreras.

Esquema de la tabla:
CODIGO varchar(7), DESCRIP varchar(30).

- **Tabla MAPA**

En esta tabla se almacena un código que relaciona los datos necesarios para generar el mapa de aulas.

Esquema de la tabla:

Plaza varchar(50).

- **Tabla PENSUM**

Esta tabla almacena los datos relacionados con el pensum de cada una de las carreras que se dictan en la universidad.

Esquema de la tabla:

ESPE varchar(4), CODIGO varchar(7), SEMESTRE varchar(2), TIPO varchar(1), REQUI varchar(31).

- **Tabla PERIACA**

En esta tabla se almacena la información relacionada con los periodos académicos en la universidad.

Esquema de la tabla:

PERIODO varchar(3), FECHA_I date, FECHA_F date.

- **Tabla PROACH**

En esta tabla se almacena algunos detalles relacionados con la programación académica.

Esquema de la tabla:

SEQ varchar(10), ESPE varchar(4), CODIGO varchar(7), SECCION varchar(2), CUPO varchar(2).

- **Tabla PROGH**

En esta tabla se almacena el horario y las aulas de la programación académica.

Esquema de la tabla:

SEQ varchar(10), HORA_I varchar(4), HORA_F varchar(7), AULA varchar(6).

- **Tabla PROFE**

En esta tabla se almacenan los datos relacionados con los profesores que dictan las asignaturas en la Universidad.

Esquema de la tabla:

Cedula varchar(8), Nombre varchar(100), Departamento varchar(2).

4.4.1.2 Identificación de la Tablas de la Base de Datos de los usuarios del sistema.

Estas tablas fueron creadas para el control de privilegios al sistema por parte de los usuarios.

- **Tabla ESTUDIANTES**

Esta tabla almacena los estudiantes que tienen acceso al sistema.

Esquema de la tabla:

CEDULA varchar(8), CLAVE varchar(10), EMAIL varchar(120), PREGUNTA varchar(40), REPSUESTA varchar(40), NUCLEO varchar(3), ESTADO varchar(2), USUARIO varchar(12).

- **Tabla PROFESORES**

  Esta tabla almacena los profesores que tienen acceso al sistema.

  Esquema de la tabla:

  USUARIO varchar(12), PASS varchar(14), TIPO varchar(4), ESTATUS varchar(2), CEDULA varchar(8), APELLIDO varchar(20), NOMBRE varchar(20).

- **Tabla PERMISOS**

  Esta tabla almacena los permisos de cada grupo de usuarios para el acceso al sistema.

  Esquema de la tabla:

  USUARIO varchar(12), PASS varchar(14), TIPO varchar(4).

4.4.1 Modelo físico de datos de la Base de Datos del Sistema SACPA y la Base de Datos Usuarios.

En la figura 4.12 se muestra el modelo físico de datos de la Base de Datos del Sistema.

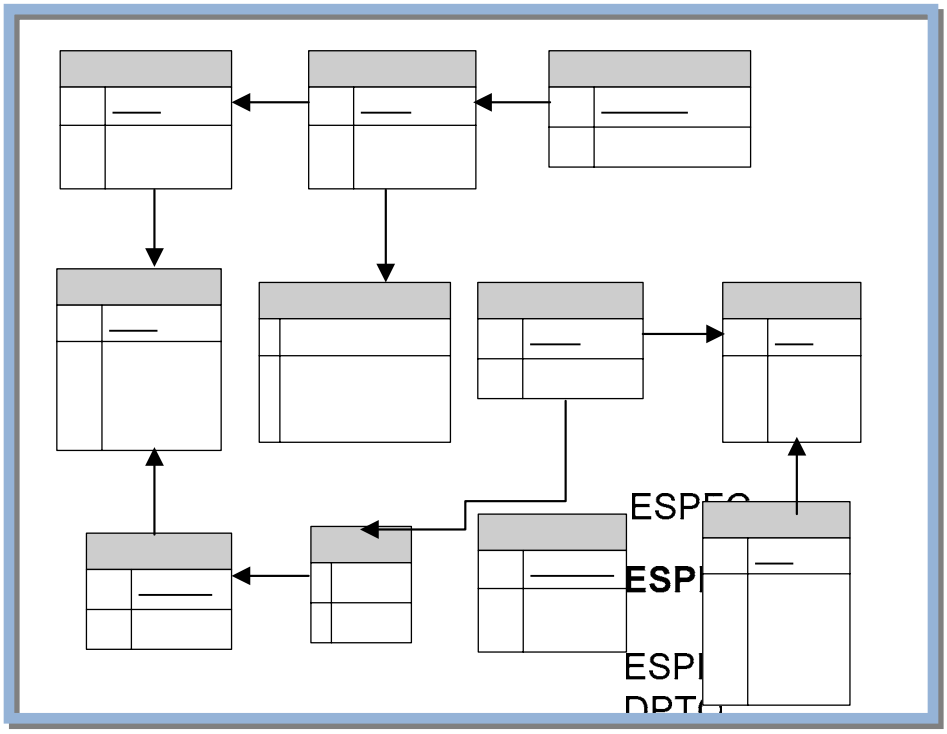
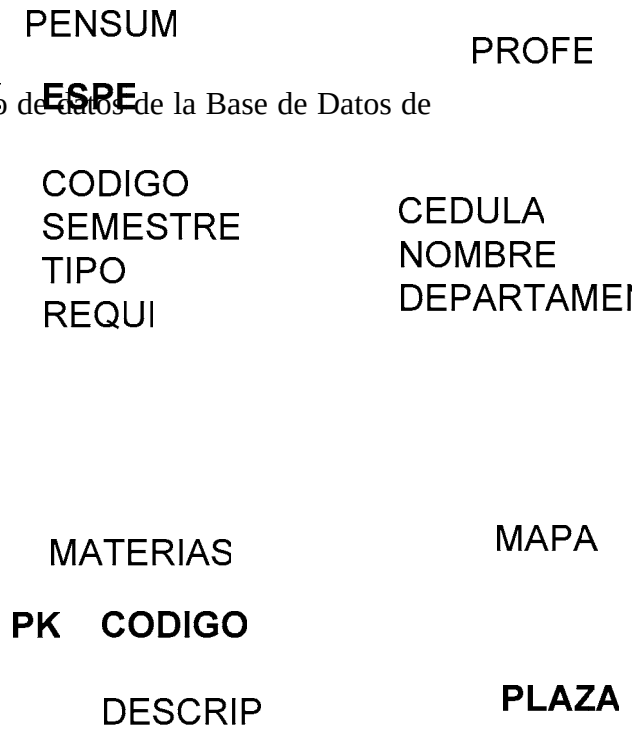


Figura 4.12. Modelo físico de datos de la Base de Datos del Sistema SACPA

En la figura 4.13 se muestra el modelo físico de datos de la Base de Datos de usuarios.



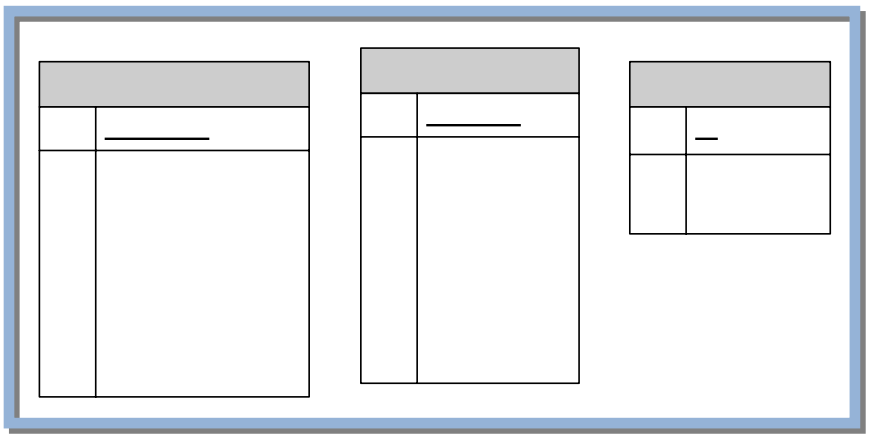


Figura 4.13. Modelo físico de datos de la Base de Datos de usuarios

4.4.3 Modelos de Gestión de Contenidos

El diseño de los modelos de gestión de contenidos es una de las actividades que consume más tiempo en esta fase. Durante esta actividad se diseñarán las páginas que integrarán los site views (las vistas del sitio) y la interacción entra las mismas. Con el diseño de las páginas se poseerá un conocimiento absoluto de cómo funcionará el sistema y de lo que se requiere específicamente.

4.4.3.1 Modelo de Gestión de Contenido de la Página de Entrar al Sistema

Esta página contiene un formulario que consta de un campo para indicar el usuario creado por computación academica y la clave, estos dos campos son indispensables para la validación de la sesión. Una vez que el usuario presiona el botón enviar, los datos son captados y validados en la base de datos, verificando el usuario y su respectiva clave, si la información ingresada pertenece a un usuario registrado en la base de datos, entonces éste puede entrar al sistema y se enviará a la página que le compete dependiendo de sus privilegios de usuario.

La acción de validar la información del usuario es representada con el estereotipo “Login”, si el usuario es un usuario no válido, el proceso de rechazo es representado con el link “KO”, si todo va bien y el usuario es validado, el proceso de aceptación es representado con el link “OK” y se procede a crear una variable de sesión del sistema, variable que identifica al usuario mientras éste hace uso de la aplicación.

En la figura 4.14 se puede observar el modelo de gestión de contenido para la Página Entrar al Sistema.

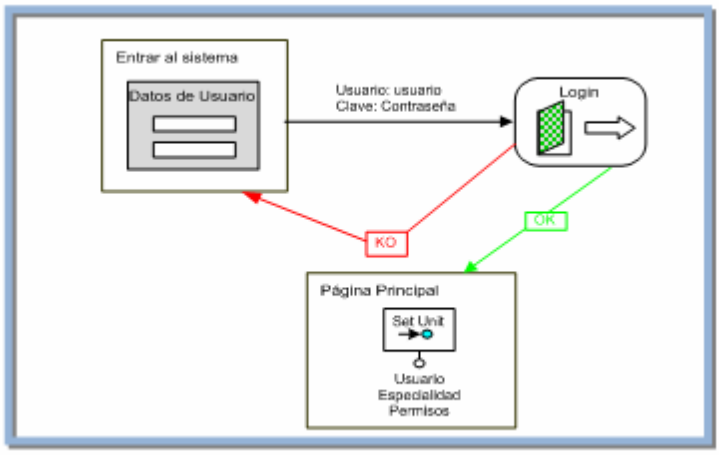


Figura 4.14. Modelo de gestión de contenido para la Página Entrar al Sistema

4.4.3.2 Modelo de Navegación y Gestión de Contenido para Páginas de Ingreso.

Debido a la gran cantidad de páginas para ingreso de información y que todas siguen un flujo parecido, solo mostraremos el flujo para ingresar asignaturas y programación.

1) **Modelo de Navegación y Gestión de Contenido para la Página de Ingreso de Asignaturas.**

Para poder acceder a la página de ingreso de asignaturas, el usuario debe pasar primero por la página principal del sistema SACPA, siendo validada su sesión dentro de la aplicación, comprobando que no ha caducado su sesión, si la sesión de usuario ha caducado, se le redirecciona a la página de inicio de sesión. Si la sesión es vigente, el usuario puede acceder a las páginas contenidas en la página principal. Para representarlo serán usadas subpages del WebML, que permiten representar una página dentro de otra.

A continuación se muestra el flujo de ingreso de asignaturas:

1. Al cargar la página, se inicia un formulario de ingreso de datos con la finalidad de que el usuario pueda ingresar una nueva asignatura. Este proceso es modelado con un “Transport Link” (enlace de transporte) como se muestra en el modelo.

2. El usuario ingresa todos los datos requeridos.

3. El usuario procede a ingresar la nueva asignatura presionando el botón cargar.

4. La información es enviada a la base de datos e ingresada en la tabla asignaturas.

5. Si se produce algún error es llamada una página de error mediante un “KO Link”, si el proceso de solicitud se lleva a cabo con éxito es llamada una página de éxito por medio de un “OK Link”.

En la Figura 4.15 se muestra el modelo de gestión de contenido para la Página Ingresar Asignaturas.

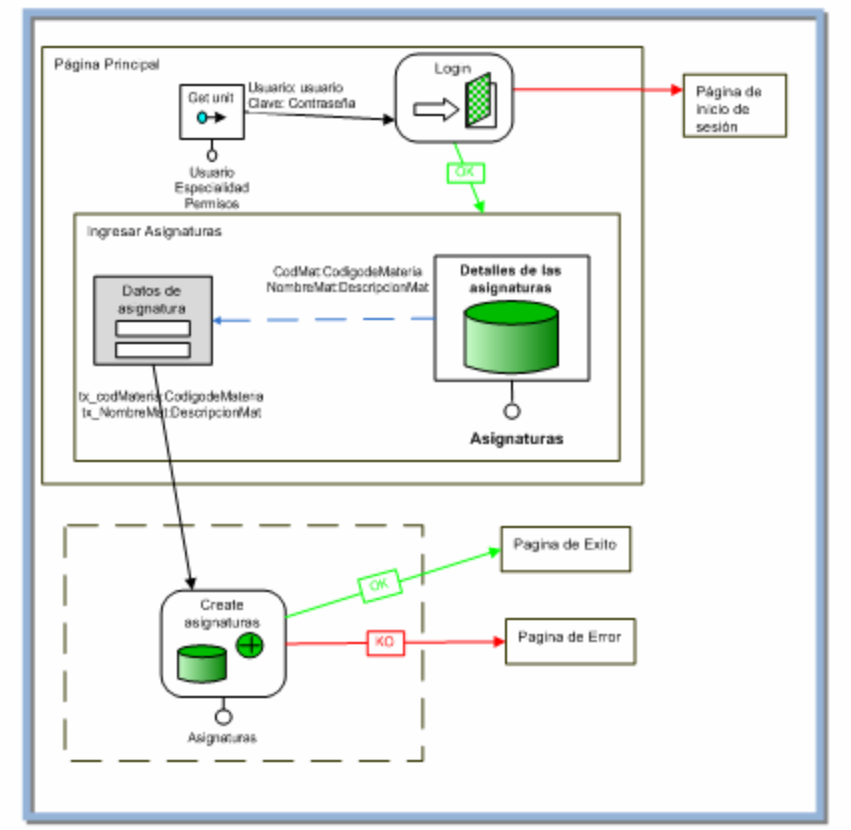


Figura 4.15. Modelo de gestión de contenido para la Página Ingresar Asignaturas

2) Modelo de Navegación y Gestión de Contenido para la Página de Ingreso de la Programación.

El ingreso de la programación es una de las funciones más importantes que posee el sistema, para poder acceder a la página de ingreso de la programación, el usuario debe pasar primero por la página principal del sistema SACPA, siendo validada su sesión dentro de la aplicación, comprobando que no ha caducado su sesión, si la sesión de usuario ha caducado, se le redirecciona a la página de inicio de sesión. Si la sesión es vigente, el usuario puede acceder a las páginas contenidas en la página principal para el control de la Programación Académica. Para representarlo serán usadas subpages del WebML, que permiten representar una página dentro de otra.A continuación se muestra el flujo de ingreso de la programación:

1. Al cargar la página, se inicia un formulario de ingreso de datos con la finalidad de que el usuario pueda ingresar una nueva programación. Este proceso es modelado con un “Automatic Link” (enlace automatico) como se muestra en el modelo.

2. El usuario ingresa todos los datos requeridos.

3. El usuario procede a guardar la programación presionando el botón cargar.

4. La información es enviada a la base de datos e ingresada en la tabla programación.

5. Si se produce algún error es llamada una página de error mediante un “KO Link”, si el proceso de solicitud se lleva a cabo con éxito es llamada una página de éxito por medio de un “OK Link”.

En la figura 4.16 se muestra Modelo de gestión de contenido para la Página Ingresar Programación.

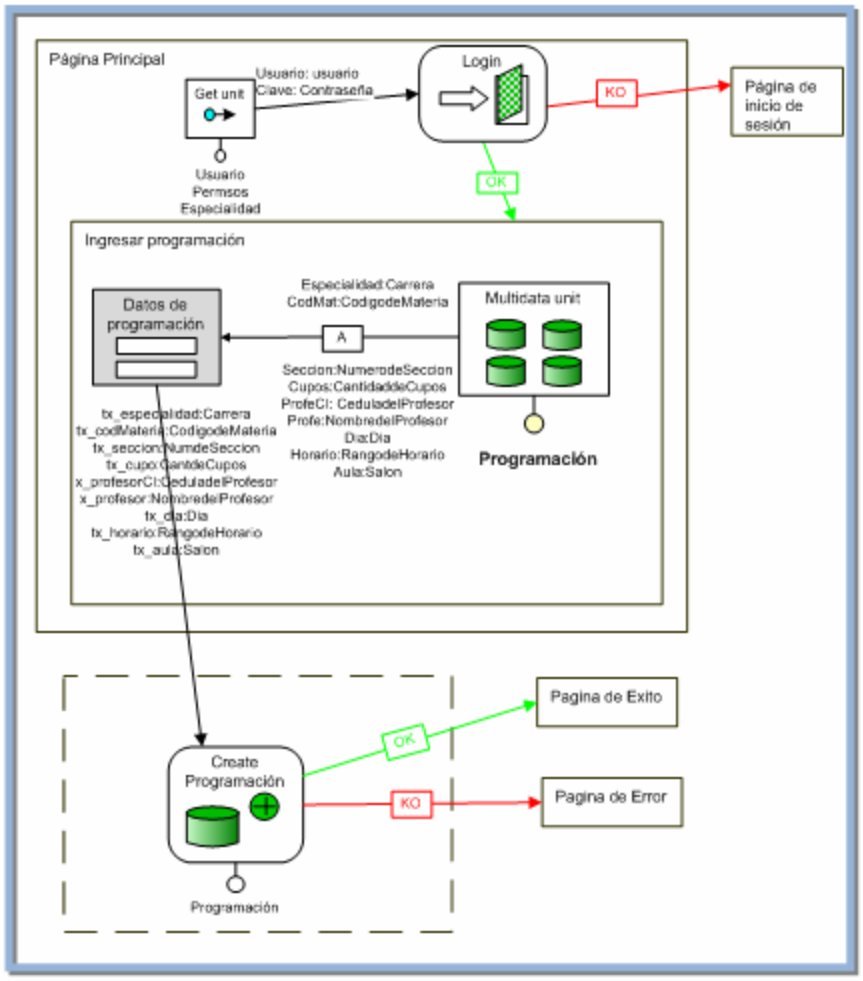


Figura 4.16. Modelo de gestión de contenido para la Página Ingresar Programación

4.4.3.3 Modelo de Navegación y Gestión de Contenido para Páginas de Consultas.

Debido a la gran cantidad de páginas para consultas de información y que todas siguen un flujo parecido solo mostraremos el flujo para consultar asignaturas y programación. En algunos casos el usuario debe llenar un campo del formulario de búsqueda.

1) **Modelo de Navegación y Gestión de Contenido para la Página Consultar Asignaturas.**

El flujo del modelo de gestión de contenidos para la página consultar asignaturas es el siguiente:

1. El usuario selecciona la opción consultar asignaturas.

2. El sistema invoca la página encargada de realizar las consultas a la base de datos de la opción seleccionada.

3. Se muestran las asignaturas cargadas en el sistema.

4. Si desea visualizar una asignatura específica se hace uso del buscador.

5. Si se produce algún error es llamada una página de error mediante un “KO Link”, si el proceso de solicitud se lleva a cabo con éxito es llamada una página de éxito por medio de un “OK Link”. (Ver figura 4.17).

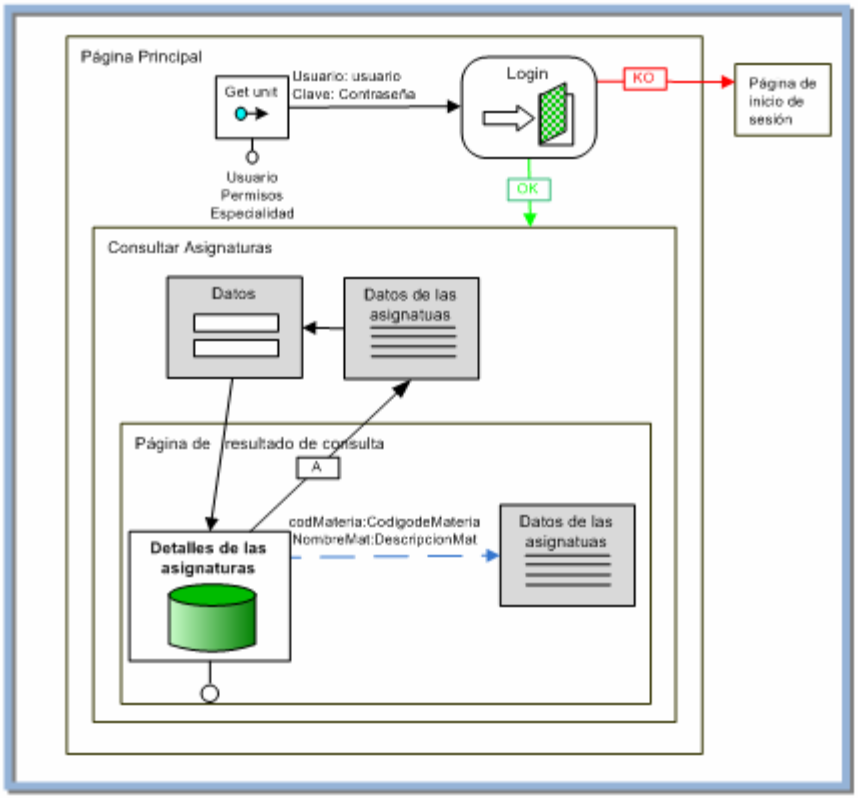


Figura 4.17. Modelo de gestión de contenido para consultar asignaturas

2) Modelo de Navegación y Gestión de Contenido para la Página Consultar Programación.

El flujo del modelo de gestión de contenidos para la página consultar programación es el siguiente:

1. El usuario selecciona la opción consultar programación.
2. El sistema invoca la página encargada de realizar las consultas a la base de datos de la opción seleccionada.

3. Se debe seleccionar la especialidad para visualizar la programación correspondiente.
4. Se muestra la programación organizada en semestres para esa especialidad.
5. Si se produce algún error es llamada una página de error mediante un “KO Link”, si el proceso de solicitud se lleva a cabo con éxito es llamada una página de éxito por medio de un “OK Link”. (Ver figura 4.18).

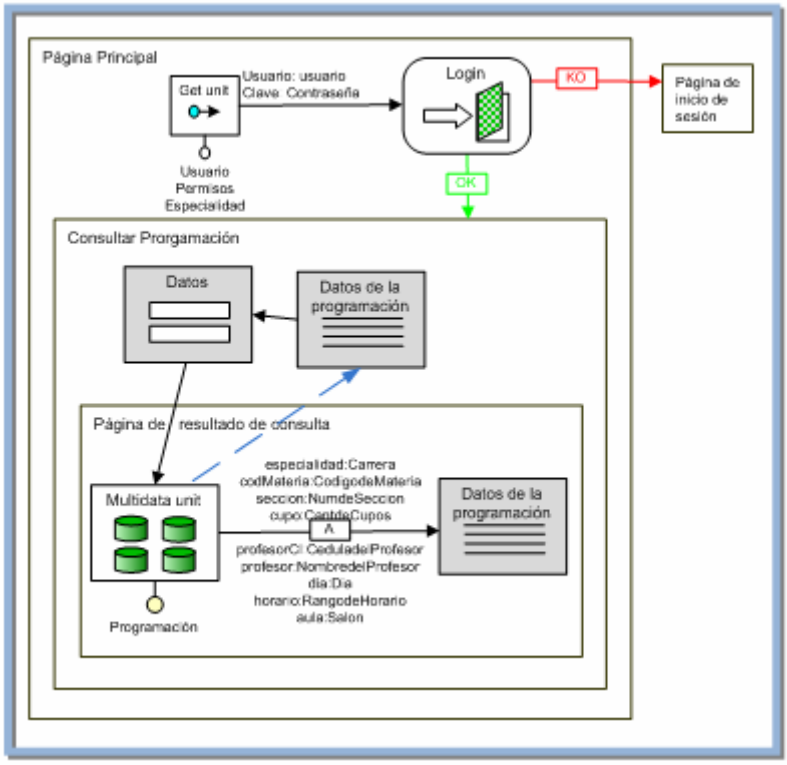


Figura 4.18. Modelo de gestión de contenido para consultar programación

4.4.3.4 Modelo de Navegación y Gestión de Contenido para Páginas de Modificaciones.

Debido a la gran cantidad de páginas para modificaciones de campos y que todas siguen un flujo parecido solo mostraremos el flujo para modificar

asignaturas ya que la programación no puede ser modificada, en caso de que se desee modificar algún campo de la programación se debe eliminar ese registro y cargarlo nuevamente.

- **Modelo de Navegación y Gestión de Contenido para la Página Modificar Asignaturas.**

En esta página es posible modificar campos de algunos registros referentes a las asignaturas.

El flujo del modelo de gestión de contenido para la modificación de campos es el siguiente:

1. El usuario visualiza el registro del campo que desea modificar.
2. El usuario pulsa el botón “modificar”.
3. El sistema permite la modificación de algunos campos del registro seleccionado.
4. El sistema emite un mensaje de confirmación.
5. Los cambios son enviados a la base de datos local.
6. Si se produce algún error es llamada una página de error mediante un “KO Link”, si el proceso de solicitud se lleva a cabo con éxito es llamada una página de éxito por medio de un “OK Link”. (Ver figura 4.19).

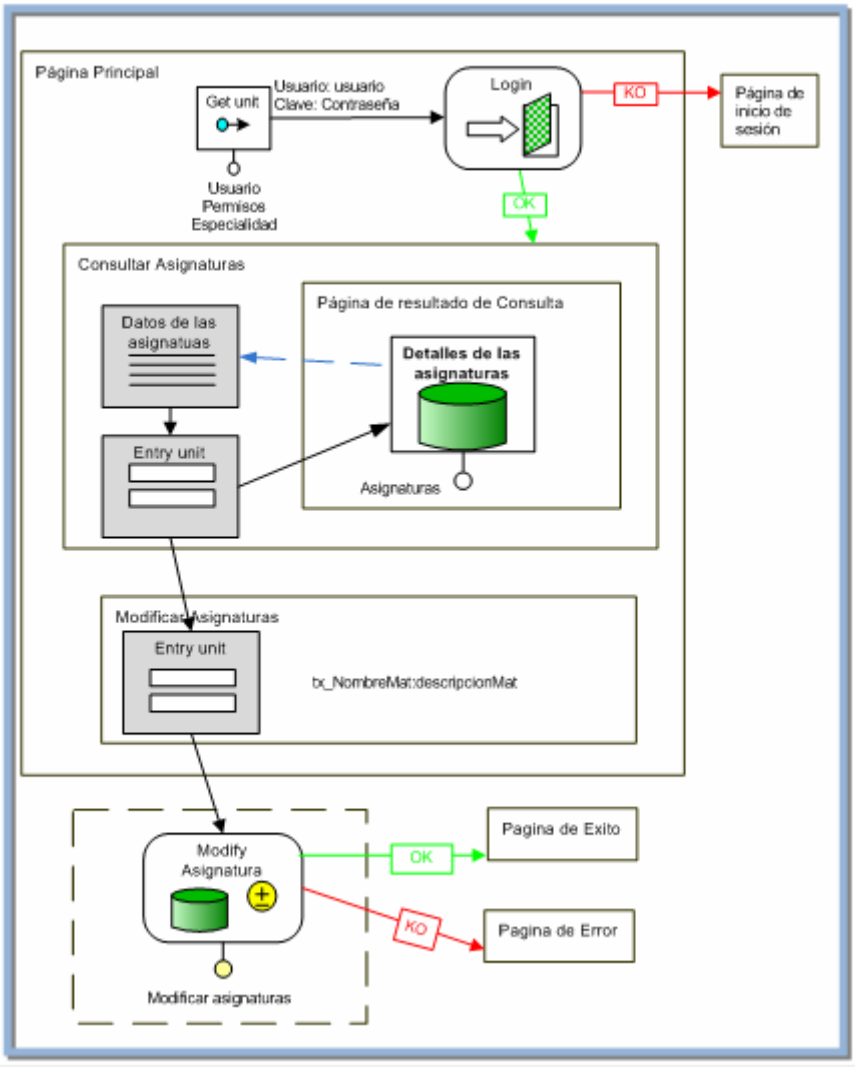


Figura 4.19. Modelo de gestión de contenido para modificar asignaturas

4.4.3.4 Modelo de Navegación y Gestión de Contenido para Páginas de Eliminaciones.

Debido a la gran cantidad de páginas para eliminaciones de registros y que todas siguen un flujo parecido solo mostraremos el flujo para eliminar asignaturas y programación.

1) **Modelo de Navegación y Gestión de Contenido para la Página Eliminar Asignaturas.**

En esta página es posible eliminar registros de las asignaturas.

El flujo del modelo de gestión de contenido para la eliminación de registros es el siguiente:

1. El usuario visualiza el registro que desea eliminar.

2. El usuario pulsa el botón “eliminar”.

3. El sistema permite la eliminación del registro seleccionado.

4. Los cambios son enviados a la base de datos local.

5. Si se produce algún error es llamada una página de error mediante un “KO Link”, si el proceso de solicitud se lleva a cabo con éxito es llamada una página de éxito por medio de un “OK Link”. (Ver figura 4.20).

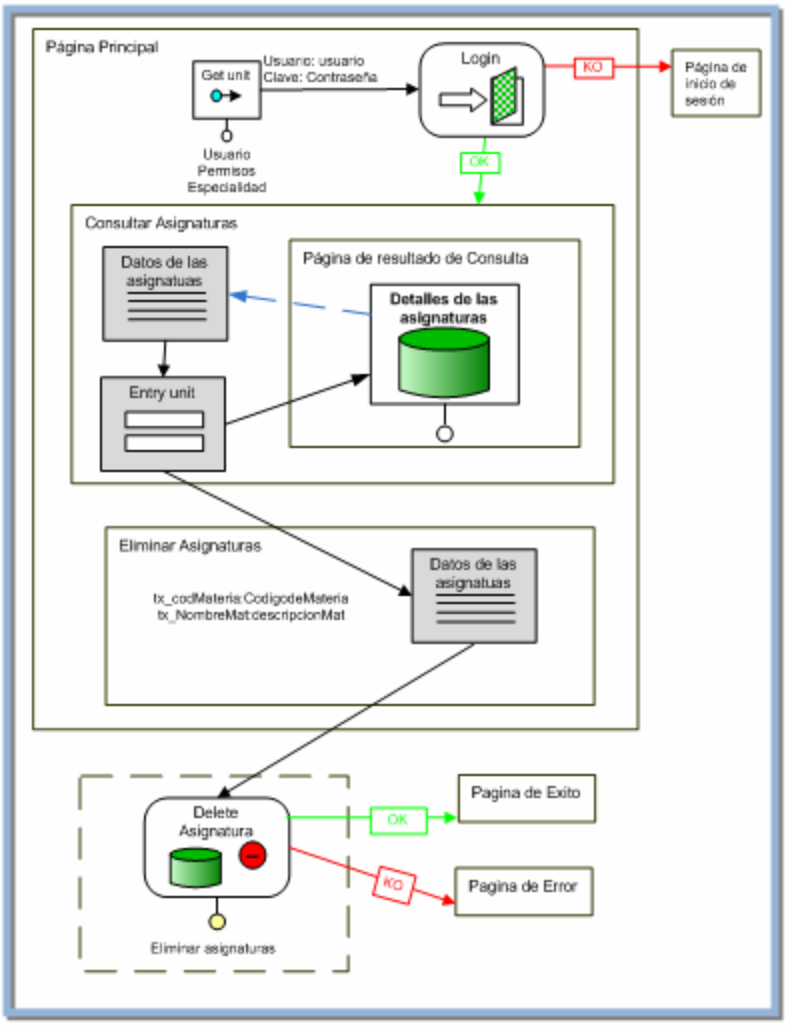


Figura 4.20. Modelo de gestión de contenido para eliminar asignaturas

2) **Modelo de Navegación y Gestión de Contenido para la Página Eliminar Programación.**

En esta página es posible eliminar registros de la programación.

El flujo del modelo de gestión de contenido para la eliminación de registros es el siguiente:

1. El usuario presiona el botón consultar programación.

2. El usuario selecciona la especialidad de la programación que desea eliminar.

3. La página que muestra los detalles de la programación es invocada de manera asíncrona.

4. Se muestran detalles sobre la programación.

5. El usuario pulsa el botón “eliminar”.
6. El sistema elimina de la base de datos el registro de la programación seleccionado.
7. Si se produce algún error es llamada una página de error mediante un “KO Link”, si el proceso de solicitud se lleva a cabo con éxito es llamada una página de éxito por medio de un “OK Link”. (Ver figura 4.21).

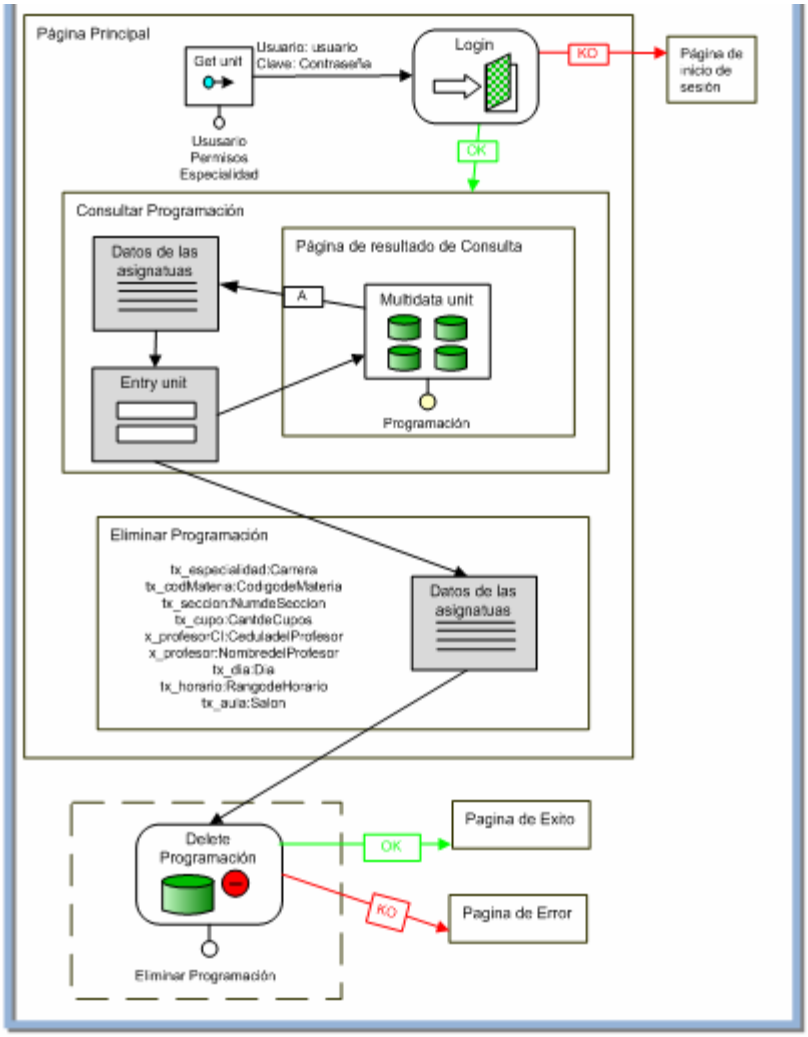


Figura 4.21. Modelo de gestión de contenido para eliminar programación

4.4.4 Diseño de la Arquitectura

En el diseño de la arquitectura, se muestra la capa específica y la capa general de la aplicación, estas están compuestas de los paquetes de análisis realizados en la fase anterior. En la capa intermedia y la capa de software del sistema se muestran los subsistemas relacionados con software prefabricado, que ejecute y proporcione

soporte a las funcionalidades requeridas por el sistema, tales como sistemas operativos, manejadores de bases de datos y componentes reutilizables. La capa específica de la aplicación está conformada por los paquetes del análisis gestión de ingreso de la programación académica y gestión de procesamiento de la programación académica; y la capa general de la aplicación se encuentra conformda por la validación de sesión, gestión de modificación y gestión de consultas.

La capa intermedia está integrada por el motor zend de PHP, que junto al manejador de bases de datos MySQL permite generar páginas WEB de contenido dinámico. Las peticiones a dichas páginas son procesadas por el servidor WEB, puediendo concebir de esta forma la ejecución del sistema SACPA. En la capa de software del sistema, el subsistema TCP/IP es el método de varias capas mediante el cual los datos se envían por la red que conecta al usuario con el servidor. El sistema operativo es el software encargado de coordinar, administrar y gestionar todos los recursos del sistema, para así dar soporte a las actividades realizadas por los demás subsistemas, el sistema manejador de base de datos MySQL, es el encargado de manejar o atender de forma transparente las solicitudes realizadas sobre la base de datos. En la figura 4.22 se muestra la vista Lógica de las Capas de la Arquitectura del Sistema.

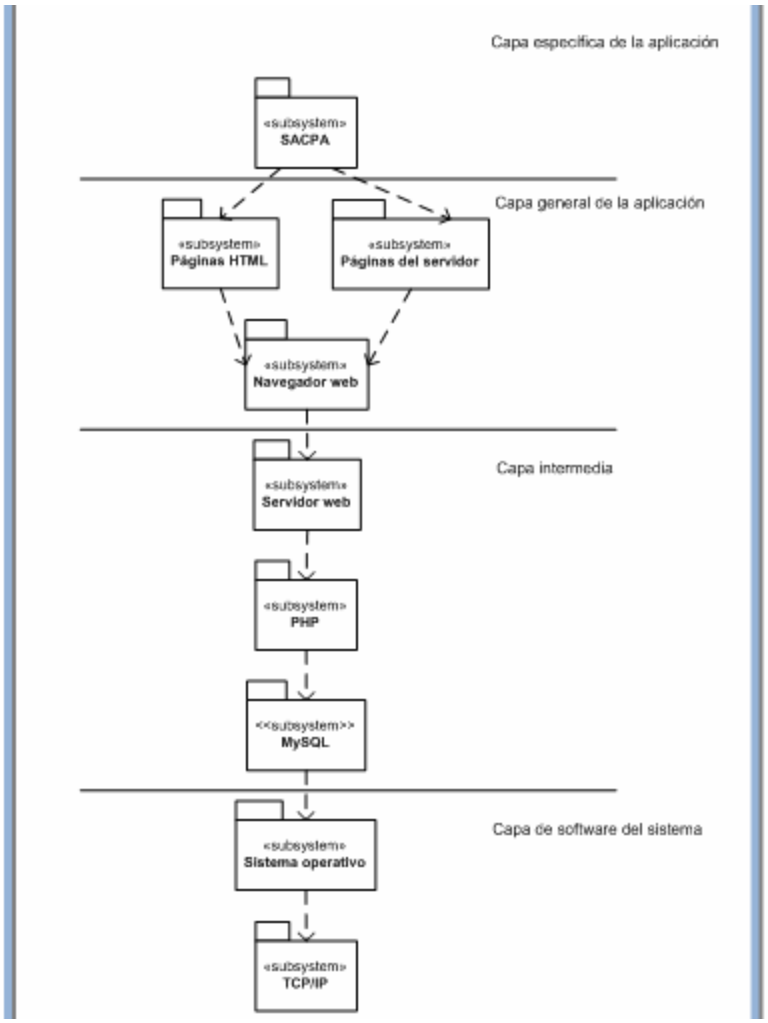


Figura 4.22. Vista Lógica de las Capas de la Arquitectura del Sistema

4.5 Implementación

A partir del diseño realizado se implementan los archivos de código fuente y ejecutables para este flujo de trabajo.

La integración del sistema se llevará a cabo implementando aquellos modelos de gestión de contenidos más importantes en la aplicación.

Es muy importante resaltar que según el Proceso Unificado para esta fase, el flujo de trabajo de implementación es el flujo primordial de todo proyecto de desarrollo de software.

4.5.1 Implementación de la Arquitectura

Basado en la vista de la arquitectura en el modelo de despliegue se identificarán los componentes necesarios para implementar los subsistemas de servicio.

4.5.2 Identificación de los Componentes de la Arquitectura

Existe un servidor donde se alojan un conjunto de páginas Web que representan el sistema como tal, que se ejecutarán a través del motor de scripts de PHP y serán distribuidas por la red mediante el servidor Web Internet Information Server/Apache mediante el protocolo TCP/IP. Adicional a estos componentes existe el servidor de base de datos MySQL que es el encargado de manejar las transacciones que solicite el cliente a través del navegador Web (ver figura 4.23).

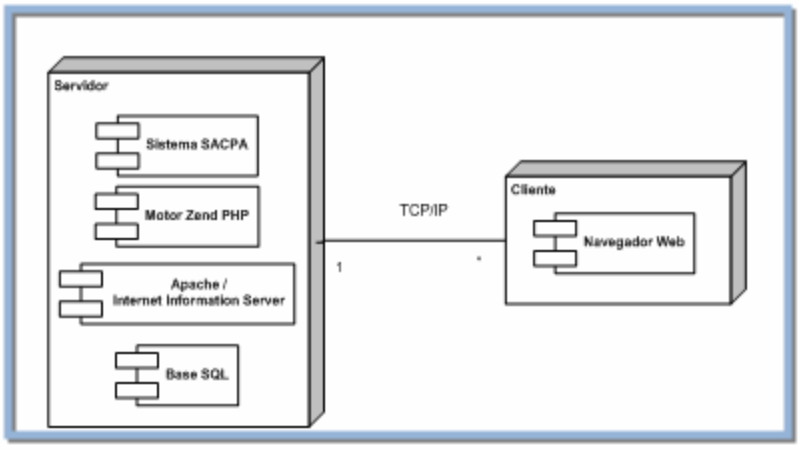


Figura 4.23 Diagrama de despliegue del Sistema

4.6 Evaluación de la fase de elaboración

Durante el desarrollo de la fase de elaboración, se lograron obtener los resultados esperados, como una arquitectura sólida y confiable para el sistema, se considera que fueron definidos la mayoría de los requisitos, los cuales fueron representados mediante el modelo casos de uso.

Se estudio la base de datos con la que trabajará el sistema y se pudieron diseñar las páginas de dicha aplicación mediante el modelo de hipertexto y modelo de gestión de contenido de WebML. Es importante mencionar que solo fue necesaria una iteración para dar por cumplidos los objetivos planteados en la fase de elaboración. .

CAPITULO V. FASE DE CONSTRUCCIÓN

5.1 Introducción

En la fase anterior se estableció la línea base de la arquitectura y se definieron nuevos casos de uso significativos. En la fase de construcción se realizará la codificación de las páginas Web que fueron diseñadas en la fase de elaboración mediante el modelo gestión de contenidos. En la implementación de estas páginas también se realizará la construcción visual de las mismas, siguiendo el prototipo de interfaz mostrado en la fase anterior, también serán realizadas las pruebas para así poder obtener una versión beta del sistema.

5.2 Implementación

El Modelo de Implementación contiene los artefactos de ejecución como el código fuente, las páginas web del servidor, entre otros. Por lo tanto, el código que se va a crear en esta fase forma parte del modelo de implementación.

Para la codificación de las páginas se usará el lenguaje de programación PHP a través del ambiente de desarrollo que ofrece Aptana Studio versión 2.0. Las operaciones de base datos serán gestionadas por el administrador de base datos MySQL versión 5.0. La integración de todos estos elementos formará la arquitectura de trabajo estable que se diseñó en la fase anterior.

5.2.1 Implementación del Modelo de Hipertexto

5.2.1.1 Computación de una Página Dinámica

El flujo de trabajo para procesar una página dinámica de un contenido guardado en una base de datos se esquematiza en la figura 5.1.

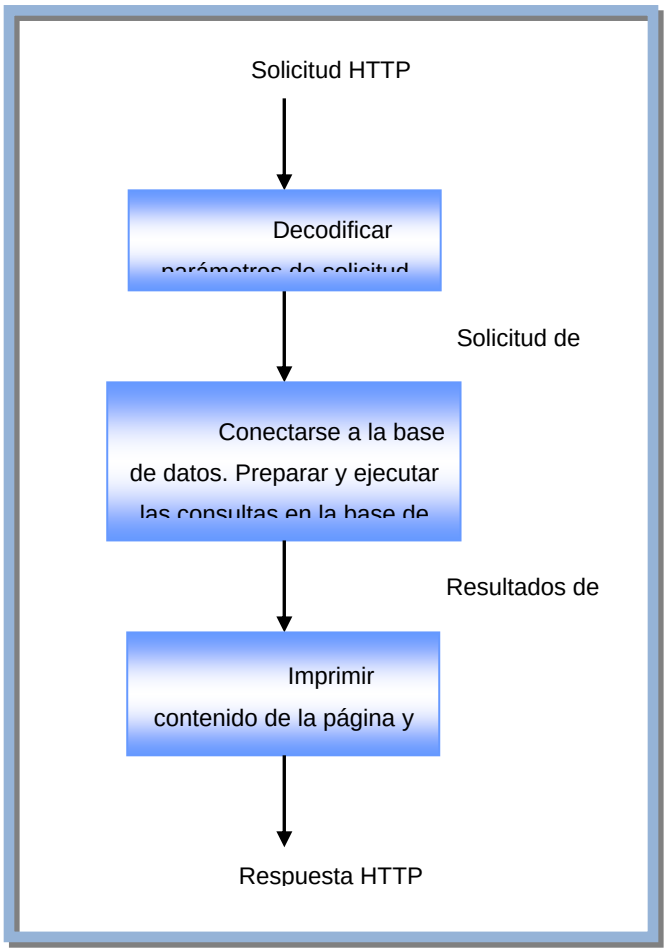


Figura 5.1. Procesamiento de una página con contenido desde una base de datos

El primer paso, es analizar la solicitud HTTP para extraer los posibles parámetros, típicamente usados por las consultas en la base de datos para sacar el contenido de la página. En el segundo paso, se establece la conexión a la base de datos y las consultas para extraer el contenido necesario para llenar la página. En la mayoría de los casos, las consultas tienen una estructura fija y requieren solo los

parámetros de entrada, en algunos casos el código fuente de la consulta debe ser ensamblado en tiempo de ejecución, justo antes de ejecutar la consulta. Luego que la consulta es enviada a la base de datos, sus resultados son transferidos a las estructuras de datos y pueden ser usadas para determinar el valor de los parámetros usados en otras consultas de extracción de contenido. Por lo tanto, la ejecución de consultas es iterada hasta que todas las consultas necesarias para retirar el contenido de la página han sido procesadas. Finalmente en el último paso, cuando todas las piezas del contenido necesario para construir la página HTML han sido extraídas, la página es producida y retornada como resultado de la petición HTTP. Específicamente, los resultados son usados para construir la parte dinámica de la página, la cual consiste típicamente de contenido (texto, imágenes, etc) y enlaces, expresados como etiquetas de enlace HTML.

5.2.2 Implementación de los Modelos de Gestión de Contenido

Aquí se codifican las operaciones definidas en el diseño del modelo de gestión de contenidos, construidos en la fase de elaboración. Las páginas diseñadas se implementarán haciendo uso del lenguaje de programación PHP y la tecnología AJAX.

- **Modelo de gestión de contenido de la pagina entrar al sistema.**

Esta es la pantalla inicial del sistema, donde los usuarios se identifican para ser autenticados por el sistema e iniciar sesión. Los datos de usuario ingresados se guardan en una variable de sesión y se buscan en la de base de datos y si hay coincidencia se le permite el acceso, de lo contrario se le niega. A continuación se presenta la vista de la interfaz en la figura 5.2 y el código fuente.

Figura 5.2. Vista de la interfaz entrar al sistema

Nombre del fichero: index.php

Descripción: Archivo principal que ejecuta el servidor al hacer la petición para entrar al sistema, en el mismo se efectúa la verificación de los usuarios, la inicialización de las variables de sesión, incluida la variable de permisos de acceso.

Código Fuente:

```
<?PHP
/*index.php: Página principal desde la que se ejecutan los llamados a las demás sub paginas*/
//Si no se ha iniciado sesión, entonces la creamos.
if(!isset($_SESSION["activo"]))
session_start();
//Arreglamos algunos problemas con la cache del deprecado IE6
header("Cache-control: private");

//Si se recibe el formulario de inicio de sesión.
if($_POST['usuario'])
{

//Incluimos el archivo declarativo de las variables de conexión a la base de datos de los usuarios.
	include "dbcon.php";

	//Inicializamos $tipousuario en null (ni profesor, ni estudiante aun)
	$tipousuario=null;
	$usu = $_POST['usuario'];
	$clave = $_POST['clave'];

//Cargamos las librerías de conexión ADODB que nos permiten utilizar el mismo código en
//más de un manejador de base de datos. Ejemplo: ORACLE y MySQL.
	include "adodb5/adodb.inc.php";
	$db = NewADOConnection("$MANEJADOR_SQL");
```

```
	$db->Connect("$LUGAR_SQL", "$USUARIO_SQL", "$CLAVE_SQL", "$BD_SQL");

//Buscamos si el usuario existe en la base de datos de los profesores.
	$resultado = $db->Execute("SELECT profesores.USUARIO, profesores.PASS,
profesores.ESTATUS,profesores.CEDULA, profesores.APELLIDO, profesores.NOMBRE,
profesores.DEPARTAMENTO, permisos.GRUPO, permisos.PERMISOS FROM profesores
LEFT JOIN permisos ON profesores.GRUPO=permisos.ID WHERE
profesores.CEDULA='".$usu."'");
	if ($resultado === false) die("La consulta a la BD fallo");

//Si no obtenemos registros entonces buscamos en la base de datos de los estudiantes.
		if($resultado->RecordCount()<1)
		{
			$resultado = $db->Execute("SELECT estudiantes.CEDULA,
estudiantes.CLAVE, estudiantes.EMAIL, estudiantes.USUARIO, permisos.GRUPO,
permisos.PERMISOS FROM estudiantes LEFT JOIN permisos ON
estudiantes.GRUPO=permisos.ID WHERE estudiantes.CEDULA='".$usu."'");
			if($resultado->RecordCount()<1)
			{

//Si no obtenemos registros, entonces el usuario no existe en el sistema.
				$mensaje= "<div align=\"center\">Usuario
desconocido.</div>";
			}
			else
			{

//Si obtenemos un registro entonces es un estudiante.
				$tipousuario="estudiante";
			}
		}
		else
		{

//Si en la primera consulta obtenemos un registro entonces el usuario es un profesor.
			$tipousuario="profesor";
		}


//Si el tipo de usuario es diferente a null
	if($tipousuario!=null)
	{
		$rs=$resultado->FetchNextObject();

//De ser un profesor
		if($tipousuario=="profesor" && $rs->PASS==$clave )
			{

//inicializamos las siguientes variables de sesión que nos acompañaran durante toda la
misma.
				$_SESSION["activo"]=true;
```

```
                                $_SESSION["login"]=$rs->CEDULA;
                                $_SESSION["nombre"]=$rs->USUARIO;
                                $_SESSION["permisos"]=$rs->PERMISOS;
                                $_SESSION["dpto"]=$rs->DEPARTAMENTO;
                        }

//De ser un estudiante
                else if($tipousuario=="estudiante" && $rs->CLAVE==$clave )
                {

//Inicializaremos estas variables de sesión.
                        $_SESSION["activo"]=true;
                        $_SESSION["login"]=$rs->CEDULA;
                        $_SESSION["nombre"]=$rs->USUARIO;
                        $_SESSION["permisos"]=$rs->PERMISOS;
                        $_SESSION["dpto"]=$rs->null;
                }
                else
                {

//Si la clave es inválida, entonces avisamos.
                        $mensaje= "<div align=\"center\">Password no valido.</div>";
                }

        }
}
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html><head>
<meta content="text/html; charset=ISO-8859-1" http-equiv="content-
type"><title>SACPA</title>
<link rel="shortcut icon" href="favicon.ico">

<!--Hoja de estilos externa-->
<link href="estilos.css" type="text/css" rel="stylesheet">

<!--funciones javascript externas-->
<script src="clienthint.js"></script>

</head>
<body>
<table id="principal" style="text-align: left; width: 100%; height: 100%;" border="1"
cellpadding="2" cellspacing="0">
<tbody>
<tr>
<td style="background-color: rgb(81, 105, 121);" colspan="2" rowspan="1"
background="fondlog.gif">
<img style =" border:0px;" alt="Sistema Automatizado Para el Control de la Programacion
Academica" title="Sistema Automatizado Para el Control de la Programacion Academica"
src="logo.png">
```

```
</td>
</tr>
<tr>
<td style="background-color: rgb(81, 105, 121); width:87%;"><span style="color: rgb(252,
252, 252);"><? if($_SESSION["login"])
{
echo '<a href="index.php?sec=inst">INICIO</a> | ';
}?> <?PHP if(!isset($_SESSION["activo"]) || $_GET['sec']=="salir") echo"HOLA
INVITADO, PUEDES <a href=\"index.php?sec=entrar\">ENTRAR AL SISTEMA</a>"; else
echo "USUARIO:".$_SESSION["nombre"]." <a href=\"index.php?sec=salir\">SALIR DEL
SISTEMA</a> | " ?> - <a class="ayuda"
href="javascript:mostrarOcultar('ayuda');">AYUDA</a></span></td>
<td style="text-align: center; background-color: rgb(81, 105, 121);">
<div style="margin-left: 0px; width: 80%;" id="navmenu" left="" align=""><span style="font-
weight: bold; color: rgb(252, 252, 252);"> 
    <span style="text-decoration: underline;">MENU</span></span></div>
</td>
</tr>
<tr>
<td class ="taf" colspan="1" rowspan="1">
<?PHP

//Dependiendo de la existencia o no de una sesión activa procedemos a permitir el incluir
//las respectivas secciones del site, eligiéndolas a través de un parámetro pasado por
//GET (sec)
if(!isset($_SESSION["activo"]))
{
        include("log.php");
}
else{
        switch($_GET["sec"])
        {
                case log:
                include ("log.php");
                break;
                case salir:
                include ("salir.php");
                break;
                case inst:
                include ("inst.php");
                break;
                case menu:
                include ("menu.php");
                break;
                case aasig:
                if($_SESSION["permisos"][0])
                {
                include ("asignaturas.php");
                break;
                }
                case aaula:
```

```
include ("aulas.php");
break;
case adpto:
include ("departamentos.php");
break;
case aescu:
include ("escuelas.php");
break;
case aespe:
include ("especialidades.php");
break;
 case aprof:
include ("profesores.php");
break;
case aprog:
include ("programacion.php");
break;
case apens:
include ("pensum.php");
break;
default:
case vasig:
include ("vasign.php");
break;
case vaula:
include ("vaulas.php");
break;
case vdepa:
include ("vdepar.php");
break;
case vescu:
include ("vescue.php");
break;
case vespe:
include ("vespec.php");
break;
case vpens:
include ("vpensu.php");
break;
case vprofe:
include ("vprofe.php");
break;
case vprogr:
include ("vprogr.php");
break;
case vcarg:
include ("vcarga.php");
break;
case maula:
include ("maulas.php");
break;
```

```
                    case agrup:
                    include ("grupos.php");
                    break;
                    case lbdad:
                    include ("limpiar_bd.php");
                    break;
                    default:
                    include ("log.php");
                    break;

          }
}
?>
<br>
<br>
<br>
</td>
<td style="width: 160px; color: rgb(69, 69, 116); background-color: rgb(203, 216, 224);
vertical-align: top;">
<?PHP

//Incluimos el menú a la derecha de la página
include("menu.php");

?>
</td>
</tr>
<tr>
<td style="background-color: rgb(81, 105, 121); text-align: center;" colspan="2"
rowspan="1"><span style="color: rgb(253, 253, 253);">Tesistas : Mariela Afonso
y Jes&uacute;s Segnini</span></td>
</tr>
</tbody>
</table>
<br>
</body></html>
//Formulario para entrar al sistema
<?php
if(basename($_SERVER['PHP_SELF'])=="log.php")
{
echo "Intento de hacking bloqueado.";
exit;
}
?>
<BR>
<BR>
<div align="center" id="ayuda" style="display: none"><table border="0" cellspacing="2"
cellpadding="8" bdcolor="black" width="420px">
<tr bgcolor="#FF6600" align="center"><td><b><font color="white"><img
src="imagenes/s_notice.png" border="0"></img>  SACPA: Sistema de
ayuda</font></b></td></tr>
```

```
<tr bgcolor="yellow" align="left"><td> Para iniciar sesi&oacute;n en el sistema debe introducir su nombre de usuario (Nro. de c&eacute;dula) y su clave, de no hacerlo no podra acceder a las diferentes secciones que conforman el sistema SACPA.</td></tr>
</table></div>

<div align=center><form method="post" action="index.php?sec=inst" name="entrar">

<table style="text-align: left; width: 420px;" border="0" cellpadding="2" cellspacing="2">
<tbody>
<tr style="color: white; font-weight: bold;" align="center">
<td style="background-color: rgb(81, 105, 121);" colspan="3" rowspan="1"><img
src="imagenes/ensacpa.png" border="0"></img></td>
</tr>
<tr>
<td style="width: 411px; background-color: rgb(203, 216, 224); ">Usuario</td>
<td rowspan="1" style="text-align: right; width: 58px; background-color: rgb(203, 216,
224);"><input maxlength="12" size="13" name="usuario"></td>
<td style="width: 203px; background-color: rgb(203, 216, 224);">(c&eacute;dula)</td>
</tr>
<tr>
<td style="width: 411px; background-color: rgb(203, 216, 224);">Clave</td>
<td rowspan="1" style="text-align: right; width: 58px; background-color: rgb(203, 216,
224);"><input maxlength="12" size="13" name="clave" type="password"></td>
<td style="width: 203px;background-color: rgb(203, 216, 224);">Olvido su clave?</td>
</tr>
<tr>

<td rowspan="1" colspan="3" style="text-align: center; background-color: rgb(203, 216,
224);"><button value="Enviar" name="Enviar" style="width: 55px; height: 22px;" id="enviar"
type="submit">Entrar</button> <button type="reset" value="Borrar" name="Borrar"
style="width: 55px; height: 22px;">Borrar</button></td>
</tr>
</tbody>
</table>
<br>
</form></div>
<?
if($mensaje && $_POST);
echo $mensaje;
?>
```

- **Modelo de gestión de contenido de la pagina de bienvenida al sistema.**

Esta página muestra un mensaje de bienvenida, el estado de sesión del usuario, compuesto por el nombre de usuario y un resumen de la cantidad de registros que se encuentran en la base de datos. Los datos del usuario se extraen a través de

variables de sesión definidas en la página de inicio de sesión. A continuación se presenta la vista de la interfaz en la figura 5.3 y el código fuente.

Figura 5.3. Vista de la interfaz de bienvenida al sistema

Nombre del fichero: inst.php

Descripción: Archivo que muestra un resumen de la cantidad de registros de la información del sistema SACPA.

Código Fuente:

```
<?PHP
if(basename($_SERVER['PHP_SELF'])=="inst.php")
{
echo "Intento de hacking bloqueado.";
exit;
}
require_once("dbcond.php");
require_once("adodb5/adodb.inc.php");

$db = NewADOConnection("$MANEJADOR_SQLD");
$db->Connect("$LUGAR_SQLD", "$USUARIO_SQLD", "$CLAVE_SQLD", "$BD_SQLD");
$resultado = $db->Execute("SELECT * FROM MATERIAS");
if ($resultado === false) die("La consulta a la BD fallo");
$totalmate=$resultado->RecordCount();
$resultado = $db->Execute("SELECT * FROM AULAS");
if ($resultado === false) die("La consulta a la BD fallo");
$totalaul=$resultado->RecordCount();
```

```
$resultado = $db->Execute("SELECT * FROM DPTO");
if ($resultado === false) die("La consulta a la BD fallo");
$totaldpto=$resultado->RecordCount();
$resultado = $db->Execute("SELECT * FROM ESCUELAS");
if ($resultado === false) die("La consulta a la BD fallo");
$totalescu=$resultado->RecordCount();
$resultado = $db->Execute("SELECT * FROM ESPEC");
if ($resultado === false) die("La consulta a la BD fallo");
$totalespe=$resultado->RecordCount();
$resultado = $db->Execute("SELECT * FROM PROFE");
if ($resultado === false) die("La consulta a la BD fallo");
$totalprof=$resultado->RecordCount();
?>
<BR>
<BR>
<div align="center" id="ayuda" style="display: none"><table border="0" cellspacing="2"
cellpadding="8" bdcolor="black" width="420px">
<tr bgcolor="#FF6600" align="center"><td><b><font color="white"><img
src="imagenes/s_notice.png" border="0"></img>  SACPA: Sistema de
ayuda</font></b></td></tr>
<tr bgcolor="yellow" align="left"><td>  En esta secci&oacute;n se presenta un resumen de la
informaci&oacute;n que actualmente maneja el sistema. Puede acceder a otras secciones
seleccionandola desde el menu ubicado en la parte derecha de su monitor.</td></tr>
</table></div>

<div align=center>
<table style="text-align: left; width: 420px;" border="0" cellpadding="2" cellspacing="2">
<tbody>
<tr style="color: white; font-weight: bold;" align="center">
<td style="background-color: rgb(81, 105, 121);" colspan="2" rowspan="1"><img
src="imagenes/resumen.png" border="0"></img></td>
</tr>
<tr>
<td rowspan="1" style="text-align: left; background-color: rgb(203, 216,
224);"><b>Asignaturas</b></td>
<td rowspan="1" style="text-align: center; background-color: rgb(203, 216,
224);"><?=$totalmate?></td>
</tr>
<tr>
<td rowspan="1" style="text-align: left; background-color: rgb(203, 216,
224);"><b>Aulas</b></td>
<td rowspan="1" style="text-align: center; background-color: rgb(203, 216,
224);"><?=$totalaul?></td>
</tr>
<tr>
<td rowspan="1" style="text-align: left; background-color: rgb(203, 216,
224);"><b>Departamentos</b></td>
<td rowspan="1" style="text-align: center; background-color: rgb(203, 216,
224);"><?=$totaldpto?></td>
</tr>
<tr>
```

```
<td rowspan="1" style="text-align: left; background-color: rgb(203, 216,
224);"><b>Escuelas</b></td>
<td rowspan="1" style="text-align: center; background-color: rgb(203, 216,
224);"><?=$totalescu?></td>
</tr>
<tr>
<td rowspan="1" style="text-align: left; background-color: rgb(203, 216,
224);"><b>Especialidades</b></td>
<td rowspan="1" style="text-align: center; background-color: rgb(203, 216,
224);"><?=$totalespe?></td>
</tr>
<tr>
<td rowspan="1" style="text-align: left; background-color: rgb(203, 216,
224);"><b>Profesores</b></td>
<td rowspan="1" style="text-align: center; background-color: rgb(203, 216,
224);"><?=$totalprof?></td>
</tr>
<tr>
<td rowspan="1" colspan="2" style="text-align: center; background-color: rgb(203, 216,
224);">Para mas informaci&oacute;n contacte con el departamento de computaci&oacute;n
acad&eacute;mica</td>
</tr>
</tbody>
</table>
<br>
</form></div>
```

- **Modelo de gestión de contenido de la pagina de finalización de sesión.**

Esta página se encarga de establecer el punto de finalización de la sesión de un usuario. Se identifica al usuario por medio de la extracción de las variables de sesión previamente creadas. Luego se almacena en la base de datos la información relacionada con finalización de sesión y el sistema redirecciona al usuario a la página para entrar al sistema. A continuación se presenta la vista de la interfaz en la figura 5.4 y el código fuente.

Figura 5.4. Vista de la interfaz de finalización de sesión

Nombre del fichero: salir.php

Descripción: Archivo que termina la sesión del usuario.

Código Fuente:

```
<?php
if(basename($_SERVER['PHP_SELF'])=="salir.php")
{
        echo "Intento de hacking bloqueado.";
        exit;
}
if(!isset($_SESSION)){
header("location: index.php");
} else {
session_unset("SESSION");
session_destroy();
include("cla_bas.php");
$base= new base();
echo"<br><br>";
$base->mensajero("<div align=\"center\">Su sessi&oacute;n se ha cerrado
exitosamente.<BR> <a class=\"az\" href=\"index.php?sec=log\"></div>");
include("log.php");
}?>
```

- **Modelo de gestión de contenido de las páginas ingresar asignaturas.**

Es un grupo de páginas está conformado por ocho (8) enlaces, que se encargan de permitir el ingreso de los datos relacionados con la programación académica: asignaturas, aulas, departamentos, escuelas, especialidades, pensum, profesores y programación. Para efectos de practicidad, sólo se colocarán dos (2) ejemplos de este tipo de páginas: asignaturas y progrmación, debido a la masiva cantidad de páginas que se tendrían que representar.

Página ingresar asignaturas

A continuación se presenta la vista de la inetrfaz ingresar asignaturas en la figura 5.5 y el código fuente de la misma.

Figura 5.5. Vista de la interfaz ingresar asignaturas

Nombre del fichero: asignaturas.php

Descripción: Se manejan todos los formularios referentes a la carga y manipulación de las asignaturas.

Código Fuente:

```
<?PHP
//Verificamos que el archivo no sea llamado individualmente.
if(basename($_SERVER['PHP_SELF'])=="asignaturas.php")
{
	echo "Intento de hacking bloqueado.";
	exit;
}
include "cla_asi.php";
$asignatura=new asignaturas();

//Si venimos del formulario para agregar, entramos a la rutina de
protección/validación/inserción del registro.
if($_POST['agregar']=="paso2")
{
	$codigo=$asignatura->proteger($_POST["codigo"]);
```

```
		$descripcion=$asignatura->proteger($_POST["descripcion"]);
		if($asignatura->validar("agregar",$codigo,$descripcion))
		{
			$asignatura->ingresar($codigo,$descripcion);
		}
		else
		{
			$asignatura->errordevalidacion();
		}
}
//Fin de la protección/validación/inserción del registro.
//Formulario para agregar asignaturas
if(!$_GET['actualizar'] && !$_POST['actualizar'])
{
echo '
<BR>
<div align="center" id="ayuda" style="display: none"><table border="0" cellspacing="2"
cellpadding="8" bdcolor="black" width="420px">
<tr bgcolor="#FF6600" align="center"><td><b><font color="white"><img
src="imagenes/s_notice.png" border="0"></img>  SACPA: Sistema de
ayuda</font></b></td></tr>
<tr bgcolor="yellow" align="left"><td>  Puede agregar una nueva Asignatura llenando los
campos correspondientes. Para ello debe respetar las siguientes normativas:<br>
1) El campo <b>C&oacute;digo</b> debe contener 7 d&iacute;gitos num&eacute;ricos.
Ejemplo: 0123456<br>
2) El campo <b>Descripci&oacute;n</b> se refiere al nombre de la asignatura, debe
contener por lo menos 5 caracteres alfanum&eacute;ricos. Ejemplo: Matem&aacute;ticas
I.</td></tr>
</table></div>
<div align=center><form method="post" action="index.php?sec=aasig" name="entrar">
<input type="hidden" name="agregar" value="paso2">
<table style="text-align: left; width: 420px;" border="0" cellpadding="2" cellspacing="2">
<tbody>
<tr style="color: white; font-weight: bold;" align="center">
<td style="background-color: rgb(81, 105, 121);" colspan="3" rowspan="1"><img
src="imagenes/casig.png" border="0"></img></td>
</tr>
<tr>
<td style="width: 411px; background-color: rgb(203, 216, 224); ">Codigo</td>
<td colspan="2" rowspan="1" style="text-align: left; width: 58px; background-color: rgb(203,
216, 224);"><input maxlength="7" size="8" name="codigo"></td>
</tr>
<tr>
<td style="width: 411px; background-color: rgb(203, 216, 224);">Descripcion</td>
<td colspan="2" rowspan="1" style="text-align: left; width: 58px; background-color: rgb(203,
216, 224);"><input maxlength="30" size="31" name="descripcion"></td>
</select></td>
</tr>
<tr>
<td rowspan="1" colspan="4" style="text-align: center; background-color: rgb(203, 216,
224);"><button value="Enviar" name="Enviar" style="width: 55px; height: 22px;" id="enviar"
```

```
type="submit">Cargar</button> <button type="reset" value="Limpiar" name="Borrar"
style="width: 55px; height: 22px;">Borrar</button></td>
</tr>
</tbody>
</table>
<br>
</form></div>';
}
//fin del formulario para agregar asignaturas
        ?>
```

Nombre del fichero: cla_asi.php

Descripción: Declaración de la clase asignatura desde la cual se manejan todos los métodos que manipulen la información referente a las asignaturas (aquí se mencionan solo los referentes a ingresar asignaturas)

Código Fuente:

```
<?php
/*Clase asignaturas: Esta clase formará los objetos contentivos de los datos
referentes a las asignaturas, así como sus respectivos métodos*/

//Verificamos que el archivo no sea llamado individualmente.
if(basename($_SERVER['PHP_SELF'])=="cla_asi.php")
{
    echo "Intento de hacking bloqueado.";
    exit;
}
include("cla_bas.php");

//Heredamos de la clase base
class asignaturas extends base{

//Constructor de la clase
public function __construct(){
        $this->error="";
        $this->proceso=new procesos();
}
//Destructor de la clase
public function __destruct(){}
/*Este método permite ingresar una nueva materia a la BD, recibe como parametros 2
cadenas alfa-nÃºmericas
que representan el codigo de materia y la descripciÃ³n de la misma.*/
public function ingresar($codasi,$desasi)
{
        if($_SESSION["permisos"][0])
        {
```

```
                $this->proceso-
>insertar_simple("asignaturas",$codasi,$desasi,null,null,null,null);
                echo "<div align='center'><b>Gracias!, La asignatura se ha agregado con
exito</b></div>";
        }
        else
        {
                $this->error="Error 2000: Ud. no dispone de privilegios para agregar
asignaturas.";
                $this->errordevalidacion();
        }
}
/*Este método se encarga de la ejecuciÃ³n de los procesos correspondientes a
validaciones*/
public function validar($tipo,$codigo,$descripcion)
{
        //Verificamos que el usuario actual posea permisos necesarios.
        if($_SESSION["permisos"][0]) //inicio de if sessiones
        {
                $ok=true;
                echo $tipos;
                if($tipo=="agregar")
                {
                        if(!$this->repetida($codigo))
                        {
                                $this->error=$this->error."-Ya existe el codigo de
materia.<br>";
                                $ok =false;
                        }

                }
        //verificamos que el código sea de 7 digitos
        if(strlen($codigo)!=7)
        {
                $this->error=$this->error."-La longitud del codigo debe ser de 7 digitos
num&eacute;ricos<br>";
                $ok=false;
        }
        //Verificamos que la descripción sea minimo de 5 caracteres
        if(strlen($descripcion)<5)
        {
                $this->error=$this->error."-La descripcion debe ser de al menos 5
caracteres.<br>";
                $ok=false;
        }
        $patron = "^[[:digit:]]+$";
        //chequeamos que el campo cédula solo contenga números
        if(!eregi($patron, $codigo))
        {
                $this->error=$this->error."-El c&oacute;digo solo puede contener
n&uacute;meros.<br>";
```

```
                $ok=$false;
        }
return $ok;
}//fin if sessiones

else//nicio else sessiones
{
        $this->error="Error 2000: Ud. no dispone de privilegios para agregar asignaturas.";
        }//fin else sessiones} } ?>
```

1) Página ingresar programación

A continuación se presentan las vistas de las dos interfaces para ingresar la programación y su respectivo código fuente.

En la figura 5.6 se muestra la vista de la interfaz ingresar programación donde se ingresa el código y nombre de la asignatura, además la sección, los cupos y el nombre del profesor.

En la figura 5.7 se muestra una segunda pantalla de la interfaz ingresar programación donde se cargan los horarios y el aula.

Figura 5.6. Vista de la interfaz ingresar programación primera pantalla

Figura 5.7. Vista de la interfaz ingresar programación segunda pantalla

Nombre del fichero: programacion.php

Descripción: Se manejan todos los formularios referentes a la carga y manipulación de la programación.

Código Fuente:

```
<?php
//Evitamos que el fichero sea llamado individualmente.
if(basename($_SERVER['PHP_SELF'])=="programacion.php")
{
	echo "Intento de hacking bloqueado.";
	exit;
}
include "cla_prog.php";

$error=0;
$error2=0;

//Instanciamos un nuevo objeto de la clase programación.
$programa=new
programacion($LUGAR_SQLD,$USUARIO_SQL,$CLAVE_SQLD,$BD_SQLD);

//Si venimos de los formularios de ingreso de programación.
if($_POST["paso1"]||$_POST["paso2"])
{

//Si estamos creando una nueva sesión.
	if($_POST["paso1"])
	{
		if($programa->validar1($programa-
>proteger($_POST["especialidad"]),$programa->proteger($_POST["codigomat"]),$programa-
```

```
>proteger($_POST["seccion"]),$programa->proteger($_POST["cupo"]),$programa-
>proteger($_POST["profesor"]))
                        {
                                $error=0;
                        }
                        else
                        {
                                $error=1;
                                $programa->errordevalidacion();
                        }
                }
                //Si estamos agregando horarios a la seccià³n creada anteriormente.
                if($_POST["paso2"])
{
                        if($programa->validar2($programa->proteger($_POST["SEQ"]),$programa-
>proteger($_POST["dia"]),$programa->proteger($_POST["horai"]),$programa-
>proteger($_POST["horaf"]),$programa->proteger($_POST["aula"])))
                        {
                                $error2=0;
                        }
                        else
                        {
                                $error2=1;
                                $programa->errordevalidacion();
                        }
                }

}

//Mostramos los formularios para agregar nueva programación
if((!$_POST["paso1"] && !$_POST["paso2"] && !$_POST["paso3"]) || $error==1)
{
//Verificamos los permisos de usuario. Si son solo para el departamento
//cargamos solo las especialidades de ese departamento en el formulario
//desde el cual se crean nuevas secciones para la programación.
if($_SESSION["permisos"][17] && !$_SESSION["permisos"][16])
        $espe=$programa->proceso-
>consulta_t_simple("especialidadesd",$_SESSION["dpto"]);

//Si los permisos son globales entonces se muestran todas las especialidades.
else if($_SESSION["permisos"][16])
        $espe=$programa->proceso->consulta_t_simple("especialidadesd",null);
?>
<BR>
<BR>
<!MENSAJE DE AYUDA!>
<div align="center" id="ayuda" style="display: none"><table border="0" cellspacing="2"
cellpadding="8" bdcolor="black" width="420px">
<tr bgcolor="#FF6600" align="center"><td><b><font color="white"><img
src="imagenes/s_notice.png" border="0"></img>  SACPA: Sistema de
ayuda</font></b></td></tr>
```

```
<tr bgcolor="yellow" align="left"><td>  Puede agregar la PROGRAMACI&Oacute;N de su Departamento llenando los campos correspondientes. Para ello debe respetar las siguientes normativas:<br>
1) seleccione de la lista la <b>Especialidad</b> a la que pertenecer&aacute; esta Programaci&oacute;n. Ejemplo: INGENIER&Iacute;A EN COMPUTACI&Oacute;N.<br>
2) El campo <b>Asignatura</b> debe contener por lo menos 3 caracteres alfanum&eacute;ricos. Ejemplo: Matem&aacute;ticas I.<br>
3) El campo <b>Secci&oacute;n</b> debe contener 2 d&iacute;gitos num&eacute;ricos. Ejemplo: 01.<br>
4) El campo <b>Cupo</b> debe contener por lo menos 2 d&iacute;gitos num&eacute;ricos y un valor igual o mayor a 15. Ejemplo: 15.<br>
5) El campo <b>C.I Profesor</b> debe contener 8 d&iacute;gitos num&eacute;ricos. Ejemplo: 12345678.<br>
</td></tr>
</table></div>
<! MENSAJE DE AYUDA!>

<div align=center><form method="post" action="index.php?sec=aprog" name="preforma" id="forma">
<input type="hidden" value="true" id="paso0" name="paso0">
<table style="text-align: left; width: 420px;" border="0" cellpadding="2" cellspacing="2">
<tbody>
<tr style="color: white; font-weight: bold;" align="center">
<td style="background-color: rgb(81, 105, 121);" colspan="3" rowspan="1"><img src="imagenes/caprog.png" border="0"></img></td>
</tr>
<td style="width: 411px; background-color: rgb(203, 216, 224);">Especialidad</td>
        <td colspan="2" rowspan="1" style="text-align: left; width: 58px; background-color: rgb(203, 216,
224);"><select id = "1133" name="especialidad" onchange="this.form.submit()">
<OPTION VALUE="null" SELECTED="true">Seleccione una especialidad</OPTION>
<?php
if($_POST["paso0"]==true)
{
while ($res=$espe->FetchNextObject())
        {
                if($_POST["especialidad"]==$res->ESPE)
                {
                        echo "<OPTION VALUE=\"".$res->ESPE."\" SELECTED=\"true\">";
                }
                else
                {
                        echo "<OPTION VALUE=\"".$res->ESPE."\">";
                }
                echo $res->ESPECIALID;
                echo "</OPTION>";
                }
        }
else
{
```

```
		while ($res=$espe->FetchNextObject())
		{
			echo "<OPTION VALUE=\"".$res->ESPE."\">";
			echo $res->ESPECIALID;
			echo "</OPTION>";
		}
}

if($_POST["paso0"]==true)
{
		$asig=$programa->proceso-
>consulta_t_simple("asig_x_espe",$_POST["especialidad"]);
}
else
{
		$asig=$programa->proceso->consulta_t_simple("asig_x_espe",null);
}

?>

</select></td>
</tr>
</form>
<form method="post" action="index.php?sec=aprog" name="entrar" id="form1">
<input type="hidden" value="true" id="paso1" name="paso1">
<input type="hidden" value="<?=$_POST['especialidad']?>" id="especialidad"
name="especialidad">

<tr>
<td style="width: 411px; background-color: rgb(203, 216, 224);">Asignatura</td>
<td colspan="2" rowspan="1" style="text-align: left; width: 58px; background-color: rgb(203,
216, 224);">
<input id="1122" name="codigomat" type="text" size="8" maxlength="7"
onblur="showHint(this.value,document.getElementById(1133).value);"
onkeyup="showHint(this.value,document.getElementById(1133).value);"
autocomplete="off"><BR>
<span id="txtHint"></span><br>
<select name="1123" id="asignatura"
onchange="document.getElementById(1122).value=entrar.asignatura.value;
showHint(this.value,document.getElementById(1133).value);">
<OPTION VALUE="null">---</option>


<?PHP
while ($res=$asig->FetchNextObject())
		{
			echo "<OPTION VALUE=\"".$res->CODIGO."\">";
			echo $res->CODIGO." - ".$res->DESCRIP;
			echo "</OPTION>";
		}
?>
```

```
</select></td>
</tr>
<tr>
<td style="width: 411px; background-color: rgb(203, 216, 224); ">Seccion</td>
<td colspan="2" rowspan="1" style="text-align: left; width: 58px; background-color: rgb(203,
216, 224);"><input maxlength="2" size="3" name="seccion"
onblur="ponerCeros(this,2);"></td>
</tr>
<tr>
<td style="width: 411px; background-color: rgb(203, 216, 224); ">Cupo</td>
<td colspan="2" rowspan="1" style="text-align: left; width: 58px; background-color: rgb(203,
216, 224);"><input maxlength="2" size="3" name="cupo" onblur="ponerCeros(this,2);">[00 =
secci&oacute;n bloqueada]</td>
</tr>
<tr>
<td style="width: 411px; background-color: rgb(203, 216, 224); ">C.I Profesor</td>
<td colspan="2" rowspan="1" style="text-align: left; width: 58px; background-color: rgb(203,
216, 224);"><input maxlength="8" size="9" name="profesor"
onblur="ponerCeros(this,8);"></td>
</tr>
<tr>
<td rowspan="1" colspan="4" style="text-align: center; background-color: rgb(203, 216,
224);"><button value="Enviar" name="Enviar" style="width: 55px; height: 22px;" id="enviar"
type="submit">Cargar</button> <button type="reset" value="Limpiar" name="Borrar"
style="width: 55px; height: 22px;">Borrar</button></td>
</tr>
</tbody>
</table>
<br>
</form></div>
<?PHP
}
//Si recibimos datos de los formularios y tenemos permisos, entonces se puede agregar
//nuevos horarios y eliminar anteriores en la sección actual en trabajo.
if($_SESSION["permisos"][16] || $_SESSION["permisos"][17] ) //inicio control session
{
        if(($_POST["paso1"] && $error==0 )|| $_POST["paso2"] || $_POST["paso3"])
        {
                if(($_POST["paso1"]) && $error==0)
                {
                $programa->ingresar1($programa-
>proteger($_POST["especialidad"]),$programa->proteger($_POST["codigomat"]),$programa-
>proteger($_POST["seccion"]),$programa->proteger($_POST["cupo"]),$programa-
>proteger($_POST["profesor"]));
                $ultimo=$programa->ultimo;
                }


        if(($_POST["paso3"]))
        {
```

```
                        $programa->borrahora($programa->proteger($_POST["SEQ"]),$programa
->proteger($_POST["DIA"]),$programa->proteger($_POST["HORAI"]),$programa->proteger($_POST["HORAF"]),$programa->proteger($_POST["AULA"]),$programa->proteger($_POST["MATERIA"]));


                        $programa->borramapa($programa->proteger($_POST["AULA"]),$programa->proteger($_POST["DIA"]),$programa->proteger($_POST["HORAI"]),$programa->proteger($_POST["HORAF"]),$programa->proteger($_POST["MATERIA"]),$programa->proteger($_POST["SECCION"]));

                }
                if($_POST["SEQ"])
                        $ultimo=$programa->proteger($_POST["SEQ"]);
                if($_POST["paso2"])
                {
                        if($error2==0)
                        {
                                $programa->ingresar2($programa->proteger($_POST["SEQ"]),$programa->proteger($_POST["dia"]),$programa->proteger($_POST["horai"]),$programa->proteger($_POST["horaf"]),$programa->proteger($_POST["aula"]),$programa->proteger($_POST["materia"]),$programa->proteger($_POST["SECCION"]));
                        }
                }
                $horario=$programa->proceso->consulta_t_simple("horario",$ultimo);
                $aulas=$programa->proceso->consulta_t_simple("aulas",null);
                $mat=$programa->proceso->consulta_t_simple("programa",$ultimo);
                $mat=$mat->FetchNextObject();
                $nespe=$programa->proceso->consulta_t_simple("especialidades",$mat->ESPE);
                $nespe=$nespe->FetchNextObject();
                $nmat=$programa->proceso->consulta_t_simple("materias",$mat->CODIGO);
                $nmat=$nmat->FetchNextObject();
                $dprofe=$programa->proceso->consulta_t_simple("profesores",$mat->CEDULA);
                $dprofe=$dprofe->FetchNextObject();
?>

<! MENSAJE DE AYUDA!>

<div align="center" id="ayuda" style="display: none"><table border="0" cellspacing="2" cellpadding="8" bdcolor="black" width="420px">
<tr bgcolor="#FF6600" align="center"><td><b><font color="white"><img src="imagenes/s_notice.png" border="0"></img>  SACPA: Sistema de ayuda</font></b></td></tr>
<tr bgcolor="yellow" align="left"><td> PROGRAMACI&Oacute;N: Ingrese el horario de la asignatura que esta cargando. Para ello debe respetar las siguientes normativas:<br>
1) Seleccione de la lista el <b>D&iacute;</b> que se dictar&aacute; la asignatura, si la materia se dicta durante m&aacute;s de un d&iacute;a cargue este dia con todos sus datos y
```

```
luego los dem&aacute;s d&iacute;as. Ejemplo: Lunes.<br>
2) Seleccione de la lista el <b>Horario</b> en que se dictar&aacute; la asignatura, tome en cuenta que la hora desde debe ser menor que la hora hasta. Ejemplo: Desde las: 7:00am Hasta las 8:00am.<br>
3) Seleccione de la lista el <b>Aula</b> en que se dictar&aacute; la asignatura. Ejemplo: A-1.
</td></tr>
</table></div>
<! MENSAJE DE AYUDA!>

<div align=center>
<table style="text-align: left; width: 420px;" border="0" cellpadding="2" cellspacing="2">
<tbody>
<tr style="color: white; font-weight: bold;" align="center">
<td style="background-color: rgb(81, 105, 121);" colspan="4" rowspan="1">Horarios</td>
</tr>
<tr>


<td style="width: 411px; background-color: rgb(203, 216, 224);" colspan="4">
<?php
echo "<B>Asignatura:</B> ".$nmat->DESCRIP." [".$mat->CODIGO."]<BR><B>Especialidad:</B> ".$nespe->ESPECIALID." [".$mat->ESPE."]<BR><B>Seccion:</B> ".$mat->SECCION."<BR><B>Cupo:</B> ".$mat->CUPO."<BR><B>Profesor:</B> ".$dprofe->NOMBRE." [".$mat->CEDULA."] ";
?>

</td>
</tr>

<tr>
<td style="width: 411px; background-color: rgb(203, 216, 224);"><B>Dia</B></td>
<td style="width: 411px; background-color: rgb(203, 216, 224);"><B>Hora</B></td>
<td style="width: 411px; background-color: rgb(203, 216, 224);"><B>Aula</B></td>
<td style="width: 40px; background-color: rgb(203, 216, 224);"><B>Borrar</B></td>
</tr>

<?
while ($row=$horario->FetchNextObject())
        {
                if($programa->evitachoque($row->HORAI,$row->HORAF,$row->DIA,$row->AULA)>1)
                        $color="rgb(240, 210, 224)";
                else
                        $color="rgb(203, 216, 224)";
                echo "<tr><td style=\"width: 411px; background-color: ".$color."\">".$row->DIA."</td>";
                echo "<td style=\"width: 411px; background-color: ".$color."\">".$row->HORAI." a ".$row->HORAF."</td>";
                echo "<td style=\"width: 411px; background-color: ".$color."\">".$row->AULA."</td>";
```

```
                    echo "<td style=\"width: 40px; background-color: ".$color." \"> <form
method=\"post\" action=\"index.php?sec=aprog\"><input type=\"image\"
src=\"imagenes/b_drop.png\" alt=\"Borrar hora\">"
."<input type=\"hidden\" value=\"true\" id=\"paso3\" name=\"paso3\">"
."<input type=\"hidden\" value=\"".$ultimo."\" id=\"SEQ\" name=\"SEQ\">"
."<input type=\"hidden\" value=\"".$row->DIA."\" id=\"DIA\" name=\"DIA\">"
."<input type=\"hidden\" value=\"".$row->HORAI."\" id=\"HORAI\" name=\"HORAI\">"
."<input type=\"hidden\" value=\"".$row->HORAF."\" id=\"HORAF\" name=\"HORAF\">"
."<input type=\"hidden\" value=\"".$row->AULA."\" id=\"AULA\" name=\"AULA\">"
."<input type=\"hidden\" value=\"".$row->AULA."\" id=\"aula\" name=\"aula\">"
."<input type=\"hidden\" value=\"".$mat->CODIGO."\" id=\"MATERIA\" name=\"MATERIA\">"
."<input type=\"hidden\" value=\"".$mat->SECCION."\" id=\"SECCION\"
name=\"SECCION\">"
."</form></td></tr>";
            }
?>
<form method="post" action="index.php?sec=aprog" name="entrar" id="form1">
<input type="hidden" value="true" id="paso2" name="paso2">
<input type="hidden" value="<?=$ultimo?>" id="SEQ" name="SEQ">
<input type="hidden" value="<?=$mat->CODIGO?>" id="materia" name="materia">
<input type="hidden" value="<?=$mat->SECCION?>" id="SECCION" name="SECCION">
<tr style="color: white; font-weight: bold;" align="center">
<td style="background-color:rgb(81, 105, 121);" colspan="4" rowspan="1">Agregar
Horas</td>
</tr>

<tr>
<td style="width: 411px; background-color: rgb(203, 216, 224); ">Dia</td>
<td colspan="3" rowspan="1" style="text-align: left; width: 58px; background-color: rgb(203,
216, 224);">

<select name="dia" id="dia">

<option value="Lunes">Lunes</option>
<option value="Martes">Martes</option>
<option value="Miercoles">Miercoles</option>
<option value="Jueves">Jueves</option>
<option value="Viernes">Viernes</option>
<option value="Sabado">Sabado</option>
</select>
</td>
</tr>

<tr>
<td style="width: 411px; background-color: rgb(203, 216, 224); ">Hora</td>
<td colspan="1" rowspan="1" style="text-align: left; width: 58px; background-color: rgb(203,
216, 224);">Desde las

<select name="horai" id="horai">

<?PHP
```

```
for($hora=7;$hora<=12;$hora++)
	for($minutos=0;$minutos<60;$minutos+=15)
	{
		$hor=str_pad($hora,2,"0",STR_PAD_LEFT);
		$min=str_pad($minutos,2,"0",STR_PAD_LEFT);
		if($hora==12)
			echo"<option value='".$hor.":".$min." PM'>".$hor.":".$min."
PM</option>";
		else
			echo"<option value='".$hor.":".$min." AM'>".$hor.":".$min."
AM</option>";
	}

for($hora=1;$hora<=9;$hora++)
	for($minutos=0;$minutos<60;$minutos+=15)
	{
		$hor=str_pad($hora,2,"0",STR_PAD_LEFT);
		$min=str_pad($minutos,2,"0",STR_PAD_LEFT);
		echo"<option value='".$hor.":".$min." PM'>".$hor.":".$min." PM</option>";
	}
?>
</select>
</td>
<td colspan="2" rowspan="1" style="text-align: left; width: 58px; background-color: rgb(203,
216, 224);">hasta las
<select name="horaf" id="horaf">

<?php

for($hora=8;$hora<=12;$hora++)
	for($minutos=0;$minutos<60;$minutos+=15)
	{
		$hor=str_pad($hora,2,"0",STR_PAD_LEFT);
		$min=str_pad($minutos,2,"0",STR_PAD_LEFT);
		if($hora==12)
			echo"<option value='".$hor.":".$min." PM'>".$hor.":".$min."
PM</option>";
		else
			echo"<option value='".$hor.":".$min." AM'>".$hor.":".$min."
AM</option>";
	}

for($hora=1;$hora<=10;$hora++)
	for($minutos=0;$minutos<60;$minutos+=15)


{
		$hor=str_pad($hora,2,"0",STR_PAD_LEFT);
		$min=str_pad($minutos,2,"0",STR_PAD_LEFT);
		echo"<option value='".$hor.":".$min." PM'>".$hor.":".$min." PM</option>";
	}
```

```
?>
</select>
</td>
</tr>
<tr>
<td style="width: 411px; background-color: rgb(203, 216, 224); ">Aula</td>
<td colspan="3" rowspan="1" style="text-align: left; width: 58px; background-color: rgb(203,
216, 224);">
<select name="aula" id="aula">
<?PHP
while ($res=$aulas->FetchNextObject())
        {
                if($_POST["aula"]==$res->AULA)
                        echo "<OPTION VALUE=\"".$res->AULA."\" selected=\"true\">";
                else
                        echo "<OPTION VALUE=\"".$res->AULA."\">";
                echo $res->AULA." - ".$res->UBICACION;
                echo "</OPTION>";
        }
?>
</select>
</td>
</tr>
<tr>
<td rowspan="1" colspan="4" style="text-align: center; background-color: rgb(203, 216,
224);"><button value="Enviar" name="Enviar" style="width: 55px; height: 22px;" id="enviar"
type="submit">Cargar</button> <button type="reset" value="Limpiar" name="Borrar"
style="width: 55px; height: 22px;">Borrar</button></td>
</tr>
</tbody>
</table>
<br>
</form></div>

<?
}
}// fin control sesión
?>
```

Nombre del fichero: gethint.php

Descripción: Archivo utilizado por la tecnologia AJAX para realizar la consulta de las materias para el pensum y la programación.

Código fuente:

```
<?php
header("Cache-Control: no-cache, must-revalidate");
 // Date in the past
header("Expires: Mon, 26 Jul 1997 05:00:00 GMT");
```

```
include "dbcond.php";

include "adodb5/adodb.inc.php";

//get the q parameter from URL
$q=$_GET["q"];
$b=$_GET["b"];


$db = NewADOConnection("$MANEJADOR_SQLD");
$db->Connect("$LUGAR_SQLD", "$USUARIO_SQLD", "$CLAVE_SQLD", "$BD_SQLD");
if($b=="null")
{
	$resultado = $db->Execute("SELECT * FROM MATERIAS WHERE CODIGO IN
(SELECT CODIGO FROM PENSUM WHERE CODIGO LIKE '".$q."%')");
	//$resultado = $db->Execute("SELECT * FROM MATERIAS left join PENSUM on
MATERIAS.CODIGO=PENSUM.CODIGO WHERE PENSUM.CODIGO LIKE '".$q."'");
}
else
{
	$resultado = $db->Execute("SELECT * FROM MATERIAS WHERE CODIGO IN
(SELECT CODIGO FROM PENSUM WHERE ESPE = '".$b."' AND CODIGO LIKE
'".$q."%')");
	//echo "SELECT * FROM MATERIAS WHERE CODIGO IN (SELECT CODIGO
FROM PENSUM WHERE ESPE = '".$b."' AND CODIGO LIKE '".$q."%')";
	//$resultado = $db->Execute("SELECT * FROM MATERIAS left join PENSUM on
MATERIAS.CODIGO=PENSUM.CODIGO WHERE PENSUM.ESPE = '".$b."' AND
PENSUM.CODIGO LIKE '".$q."'");
}
if ($resultado === false) die("La consulta a la BD fallo");

echo '<table id="tabla1" style="text-align: left; width: 300px; border-style: solid; border-
width:0; padding:1; border-color: white; border-spacing:1;"><tbody>';


if($resultado->RecordCount()<1)
	{
		$mensaje= "No hay coincidencia";
	}
else while($rs=$resultado->FetchNextObject())
	{
echo "<tr align=\"left\">
<td  width =\"50px\" class =\"mano\" style=\" color: white; background-color: rgb(81, 105,
121); -moz-border-radius:3; \" rowspan=\"1\" onclick=\"entrar.codigomat.value='".$rs-
>CODIGO."'; showHint(entrar.codigomat.value,null); \">".$rs->CODIGO."</td>
<td class =\"mano\" style=\"color: black; background-color: rgb(203, 216, 224); -moz-border-
radius:3; \" rowspan=\"1\" onclick=\"entrar.codigomat.value='".$rs->CODIGO."';
showHint(entrar.codigomat.value,null); \">".$rs->DESCRIP."</td>
</tr>";
	}
```

```
echo '</tbody></table>';

?>
```

Nombre del fichero: clienthint.js

Descripción: Contiene todas las funciones javascript utilzadas en el sistema sacpa.

Código fuente:

```
var xmlHttp

function showHint(str,sel)
{
if (str.length<3)
 {
 document.getElementById("txtHint").innerHTML="";
 return;
 }
xmlHttp=GetXmlHttpObject();
if (xmlHttp==null)
 {
 alert ("Para obtener mejores resultados actualice su explorador a Firefox o IE 6 o superior");
 return;
 }
var url="gethint.php";
url=url+"?q="+str;
url=url+"&b="+sel;
url=url+"&sid="+Math.random();
xmlHttp.onreadystatechange=stateChanged;
xmlHttp.open("GET",url,true);
xmlHttp.send(null);
}

function stateChanged()
{
if (xmlHttp.readyState==4)
{
document.getElementById("txtHint").innerHTML=xmlHttp.responseText;
}
}

function GetXmlHttpObject()
{
var xmlHttp=null;
try
 {
 // Firefox, Opera 8.0+, Safari
```

```
  xmlHttp=new XMLHttpRequest();
  }
catch (e)
  {
  // Internet Explorer
  try
    {
    xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");
    }
  catch (e)
    {
    xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
    }
  }
return xmlHttp;
}

function confirmar ( mensaje ) {
return confirm( mensaje );
}

function ponerCeros(obj,num) {
  while (obj.value.length<num && obj.value.length>0 )
    obj.value = '0'+obj.value;
}

function mostrarOcultar(id){
mostrado=0;
elem = document.getElementById(id);
if(elem.style.display=='block')mostrado=1;
elem.style.display='none';
if(mostrado!=1)elem.style.display='block';
}
```

Nombre del fichero: cla_prog.php

Descripción: Declaración de la clase programación desde la cual se manejan todos los métodos que ejecutan y manipulan información referente a la programación y reportes (aquí se mencionan solo los referentes a ingresar programación)

Código fuente:

```
<?php
/*Clase programación: Esta clase formará los objetos contentivos de los datos
referentes a la programación académica, así como sus respectivos métodos*/

//Evitamos que el fichero sea llamado individualmente.
if(basename($_SERVER['PHP_SELF'])=="cla_prog.php")
```

```
{
	echo "Intento de hacking bloqueado.";
	exit;
}

//Incluimos la clase padre
include("cla_bas.php");
class programacion extends base{

//Constructor de la clase
public function __construct(){
	$this->error="";
	$this->proceso=new procesos();
}

//Destructor de la clase
public function __destruct(){}

//Este metodo permite ingresar una nueva materia a la BD
public function ingresar1($codespe,$codasig, $seccion, $cupo, $ciprof)
{
	if($_SESSION["permisos"][16] || $_SESSION["permisos"][17] )
	{
		$ultimo=$this->proceso->ultimo_simple("programacion");
		$ultimo++;
		$this->ultimo=$ultimo;
		$this->proceso-
>insertar_simple("programacion1",$ultimo,$codespe,$codasig,$seccion,$cupo,$ciprof);
		echo "<div align='center'><b>Gracias!, La programacion se ha agregado con
exito.<br>Ahora introduzca el horario correspondiente</b></div>";

	}
	else
	{
		$this->error="Error 2014: Ud. no dispone de privilegios para cargar los datos
de la programación para esta especialidad.";
		$this->errordevalidacion();
	}
}

//Ingreso de los horarios de una sección determinada.
public function ingresar2($_seq, $_dia, $_horai, $_horaf, $_aula,$_materia,$_seccion)
{
	if($_SESSION["permisos"][16] || $_SESSION["permisos"][17] )
	{
		$this->proceso-
>insertar_simple("programacion2",$_seq,$_dia,$_horai,$_horaf,$_aula,null);
		$inicio=$this->posihora($_horai);
		$final=$this->posihora($_horaf);
		$_dia=strtoupper($_dia);
		if($_aula!="S/A")
```

```
                    {
                    $this->ocupar($inicio,$final,$_dia,$_aula);
                    $horarios=$this->rango($_horai,$_horaf);
                    for($t=0;$t<count($horarios);$t++)
                    {
                    $this->proceso-
>insertar_simple("mapa",$_aula."."$_dia."."$horarios[$t]."."$_materia."."$_seccion,null,null,
null,null,null);
                    }
                    }

          }
else
          {
                    $this->error="Error 2014: Ud. no dispone de privilegios para programar el
semestre de esta especialidad.";
                    $this->errordevalidacion();
          }
}

//validamos que no exista la misma seccion de la misma materia en la base de datos
public function repetida($codasig,$seccion)
{
          $ok=true;
          $resultado=$this->proceso->progxmatsec($codasig,$seccion);
          if($resultado->RecordCount()>0)
          {
                    $ok = false;
          }
          return $ok;
}


//validamos que la cedula del profesor exista en la base de datos
public function existe1($ciprofe)
{
          $ok=false;
          $resultado=$this->proceso->consulta_t_simple("profesores",$ciprofe);
          if($resultado->RecordCount()>0)
          {
                    $ok = true;
          }
          return $ok;
}

//validamos que la secuencia exista en la bd
public function existe2($seq)
{
          $ok=false;
          $resultado=$this->proceso->consulta_t_simple("programa",$seq);
          if($resultado->RecordCount()>0)
```

```
		{
			$ok = true;
		}
		return $ok;
}


//validamos congruencia de las horas
public function checkhora($horaei,$horaef)
{
		$ok=true;
		$horai=(int)substr($horaei,0,2);
		//Convertimos las horas y los minutos a numeros
		$minutosi=(int)substr($horaei,3,2);

		if($horaei{6}=="P")	//Si horai es PM, sumamos 12, si no, nada.
		{
			if ($horai!=12)
			$horai=$horai+12;
		}

		$horaf=(int)substr($horaef,0,2);
		//Convertimos las horas y los minutos a numeros
		$minutosf=(int)substr($horaef,3,2);
		if($horaef{6}=="P")	//Si horai es PM, sumamos 12, si no, nada.
		{
			if ($horaf!=12)
				$horaf=$horaf+12;
		}

if($horai>$horaf)
{
		$ok = false;
		return $ok;
}

if($horai==$horaf)
{
		if($minutosi>=$minutosf)
		{
			$ok = false;
			return $ok;
		}

}

		return $ok;
}

//Borra una hora en especifico de la tabla de programacion en curso
public function borrahora($seq,$dia,$horai,$horaf,$aula,$materia)
```

```
{

        $cadena[0]=$seq;
        $cadena[1]=$dia;
        $cadena[2]=$horai;
        $cadena[3]=$horaf;
        $cadena[4]=$aula;
        $cadena[5]=$materia;
        $inicio=$this->posihora($horai);
        $final=$this->posihora($horaf);
        $dia=strtoupper($dia);
        if($aula!="S/A")
                $this->desocupar($inicio,$final,$dia,$aula);
        $this->proceso->eliminar_complejo("hora_x_datos",$cadena);
}
//Borra una hora en especifico de la tabla de programacion en curso
public function borramapa($aula,$dia,$horai,$horaf,$materia,$seccion)
{

if($aula!="S/A")
                {
                        $horarios=$this->rango($horai,$horaf);
                        for($t=0;$t<count($horarios);$t++)
                        {

        $cadena=$aula."-".$dia."-".$horarios[$t]."-".$materia."-".$seccion;
                                //echo $cadena;
                                $this->proceso-
>eliminar_simple("mapeo2",$cadena,null,null,null,null,null);
                        }

                }
}

//Nos avisa si existe un choque de aulas.
public function evitachoque($horai,$horaf,$dia,$aula)
{
        $hi=$this->posihora($horai);
        $hf=$this->posihora($horaf);
        $cadena=$this->proceso->horaxdia(strtoupper($dia),$aula);
        return $this->choque($hi,$hf,$cadena);
}

//Valida los datos de creaciÃ³nd e una nueva sesiÃ³n.
public function validar1($codespe,$codasig,$seccion,$cupo,$ciprofe)
{
//Validamos los permisos.
        if($_SESSION["permisos"][16] || $_SESSION["permisos"][17]) //inicio de if sessiones
        {
                $ok=true;
                //chequeamos que se seleccione una especialidad
```

```
                    if(strlen($codespe)==null)
                    {
                            $this->error=$this->error."-Debe seleccionar una especialidad de la
lista.<br>";
                            $ok=false;
                    }

            //Verificamos que no estÃ© repetida
            if(!$this->repetida($codasig,$seccion))
            {
                    $this->error=$this->error."-Secci&oacute;n repetida en la asignatura
seleccionada.<br>";
                    $ok =false;
            }

            //verificamos que el codigo sea de 7 digitos
            if(strlen($codasig)!=7)
            {
                    $this->error=$this->error."-La longitud del c&oacute;digo debe ser de 7
d&iacute;gitos.<br>";
                    $ok=false;
            }

            //Chequeamos que el cupo sea nÃºmerico
            if(!is_numeric($cupo))
            {
                    $this->error=$this->error."-El campo cupo solo debe contener
n&uacute;meros.<br>";
            $ok=false;
            }

            //Chequeamos que el codigo sea nÃºmerico
            if(!is_numeric($codasig))
            {
                    $this->error=$this->error."-El campo c&oacute;digo solo debe contener
n&uacute;meros.<br>";
                    $ok=false;
            }

            //Chequeamos que la cedula sea nÃºmerica
            if(!is_numeric($ciprofe))
            {
                    $this->error=$this->error."-El campo c&eacute;dula solo debe contener
n&uacute;meros.<br>";
                    $ok=false;
            }

            //Chequeamos que la cedula pertenezca a un profesor existente en la bd
            if(!$this->existe1($ciprofe))
            {
                    $this->error=$this->error."-La c&eacute;dula del profesor no existe en la base
```

```
de datos.<br>";
                $ok=false;
        }

return $ok;

        }//fin if sessiones
        else//nicio else sessiones
        {
                $this->error="Error 2000: Ud. no dispone de privilegios para agregar
programaci&oacute;n.";
                //$this->errordevalidacion();

}//fin else sessiones
}

//Se valida la entrada de los horarios para una seccion.
public function validar2($seq,$dia,$horai,$horaf,$aula)
{
        if($_SESSION["permisos"][16] || $_SESSION["permisos"][17]) //inicio de if sessiones
        {
                $ok=true;

                //Chequeamos que la secuencia exista en la base de datos
                if(!$this->existe2($seq))
                {
                        $this->error=$this->error."-Hubo un error con la secuencia, por favor
cargue la programacion nuevamente.<br>";
                        $ok=false;
                }

                //Chequeamos que la horaf sea mayor a la horai
                if(!$this->checkhora($horai,$horaf))
                {


$this->error=$this->error."-La hora final debe ser mayor a la hora de inicio.<br>";

                        $ok=false;
                }

                //Verificamos que no exista choque en las aulas, de ser asi, procedemos a
avisar.
                if($this->evitachoque($horai,$horaf,$dia,$aula)>1)
                        echo "<div align=\"center\"><b>Alerta: Choque de aulas
detectado</b></div></br>";
                return $ok;

        }//fin if sessiones
        else//nicio else sessiones
        {
```

```
                    $this->error="Error 2000: Ud. no dispone de privilegios para agregar
programaci&oacute;n.";
                    //$this->errordevalidacion();
            }//fin else sessiones

}

}

?>
```

- **Modelo de gestión de contenido de las paginas consultar**

Este grupo de páginas está conformado por diez (10) páginas, construidas con la finalidad de mostrar la información almacenada en la base de datos, en ellas existe la posibilidad, los estudiantes solo podrán consultar la programación y el pensum. Para efectos de practicidad, sólo se colocarán dos (2) ejemplos de este tipo de páginas: asignaturas y programación, debido a la masiva cantidad de páginas que se tendrían que representar.

1) **Página consultar asignaturas**

A continuación se presenta la vista de la interfaz para consultar asignaturas en la figura 5.8 y el código fuente de la misma.

Figura 5.8. Vista de la interfaz consultar asignaturas

Nombre del fichero: vasign.php

Descripción: Archivo para visualizar las asignaturas.

Código Fuente:

```
<?php
if(basename($_SERVER['PHP_SELF'])=="vasign.php")
{
echo "Intento de hacking bloqueado.";
exit;
}
include "dbcond.php";
include "cla_asi.php";

$asignatura=new
asignaturas($LUGAR_SQLD,$USUARIO_SQLD,$CLAVE_SQLD,$BD_SQLD);

if($_GET['busqueda'])
$asignatura->consultar($_GET['busqueda']);
else
$asignatura->consultar(null);
?>
```

Nombre del fichero: cla_asi.php

Descripción: Declaración de la clase asignatura desde la cual se manejan todos los métodos que manipulen la información referente a las asignaturas (aquí se mencionan solo los referentes a consultar asignaturas)

Codigo fuente:

```
/*Muestra todas las asignaturas de forma tabulada*/
public function consultar($palabra)
        {
//Verificamos la existencia de permisos para poder consultar las asignaturas.
                if($_SESSION["permisos"][1] || $_SESSION["permisos"][0])
//inicio sesiones
                {


                        if($_GET['sub']=="borrar" && $_GET['cod'])
                        {
                                $this->eliminar($_GET["cod"]);
                        }
                        if($palabra=="")
                        {
                                $extender=null;
                                $paginar=$this->proceso-
>consultar_todos("asignaturas",$_GET["ob"],null,null);
```

```
                                        $pagistro="index.php?sec=vasig";
                                        include("paginador.php");
                                        $todos=$this->proceso-
>consultar_todos("asignaturas",$_GET["ob"],null,$pagisql);

                                }
                                else
                                {
                                        $palabra=$this->proteger($palabra);
                                        $extender="&busqueda=".$palabra."";
                                        $paginar=$this->proceso-
>busqueda_todos("asignaturas",$_GET["ob"],$palabra,null);
                                        $pagistro="index.php?sec=vasig".$extender."";
                                        include("paginador.php");
                                        $todos=$this->proceso-
>busqueda_todos("asignaturas",$_GET["ob"],$palabra,$pagisql);
                                }
//Iniciamos la muestra por pantalla de la tabla HTML que contiene las asignaturas.
                        $pasar=null;
                        if($_GET['ob'])
                        {
                                $pasar="&ob=".$_GET['ob'];
                        }
echo '
<BR>
<div align=center>

<table style="text-align: left; width: 55%;" border="0" cellpadding="2" cellspacing="2">
<tbody>
<tr style="color: white; font-weight: bold;" align="center">
<td style="background-color: rgb(81, 105, 121);" colspan="4" rowspan="1"><img
src="imagenes/conasi.png" border="0"></img></td>
</tr>

<form method="get" action="index.php" name="buscar">
<input type="hidden" name ="sec" value="vasig">
<tr style="color: white; font-weight: bold;" align="left">
<td style="background-color: rgb(81, 105, 121);" colspan="4" rowspan="1"><input
name="busqueda" value ='".$palabra."'><button style="width: 75px; height: 22px;"
id="buscar" type="submit">Buscar</button></td>
</tr>
</form>

<tr>
<td style="width: 411px; background-color: rgb(203, 216, 224); "><b><a class="az"
href="index.php?sec=vasig&ob=codi'.$exportar.$extender.'">Codigo</a></b></td>
<td style="width: 411px; background-color: rgb(203, 216, 224); "><b><a class="az"
href="index.php?sec=vasig&ob=desc'.$exportar.$extender.'">Descripcion</a></b></td>
<td style="width: 50px; background-color: rgb(203, 216, 224); "><b
class="az">Editar</b></td>
<td style="width: 50px; background-color: rgb(203, 216, 224); "><b
```

```
class="az">Borrar</b></td>
</tr>
';
        while ($res=$todos->FetchNextObject())

{
                echo "<tr>";
                echo "<td style=\"width: 411px; background-color: rgb(203, 216, 224);
\">".$res->CODIGO."</td>";
                echo "<td style=\"width: 411px; background-color: rgb(203, 216, 224);
\">".$res->DESCRIP."</td>";
                if($_SESSION["permisos"][0])

{
                        echo "<td style=\"width: 50px; background-color: rgb(203, 216, 224);
\"><a href=\"index.php?sec=aasig&actualizar=paso1&cod=".$res->CODIGO."&des=".$res-
>DESCRIP."\"><img src=\"imagenes/b_edit.png\" border=\"0\"></img></a></td>";
                        echo "<td style=\"width: 50px; background-color: rgb(203, 216, 224);
\"><a href=\"index.php?sec=vasig".$pasar.$exportar.$extender."&sub=borrar&cod=".$res-
>CODIGO."\" onclick=\"return confirmar('Se eliminara la asignatura ".$res->DESCRIP." y por
ende podria ser afectada la programaci&oacute;n actual  desea proseguir? ')\"><img
src=\"imagenes/b_drop.png\" border=\"0\"></img></a></td>";
                }

        Else

        {
                echo "<td style=\"width: 50px; background-color: rgb(203, 216, 224); \"><img
src=\"imagenes/b_editn.png\" border=\"0\"></img></td>";
                echo "<td style=\"width: 50px; background-color: rgb(203, 216, 224); \"><img
src=\"imagenes/b_dropn.png\" border=\"0\"></img></td>";
        }

        echo "</tr>";

        }
        echo "<tr style=\"color: white; font-weight: bold;\" align=\"center\">";
        echo "<td style=\"background-color: rgb(81, 105, 121);\" colspan=\"4\"
rowspan=\"1\">".$navegacion."<br></td>";
echo "</tr>";
        echo'
</tbody>
</table>
<br>
</form></div>';

//Finalizamos la construcción de la tabla HTML con los datos de las asignaturas.
        }
//fin llave sessiones
//Si el usuario no tiene permisos para ver la asignatura se despliega el siguiente error.
        else //else sessiones
```

```
        {
                $this->error="Error 2001: Ud. no dispone de privilegios para ver las
asignaturas.";
                $this->errordevalidacion();
        }
//fin else sessiones
}
```

2) Pagina consultar programación

A continuación se presentan las vistas de la interfaz para consultar programación y el código fuente.

En la figura 5.9 se muestra la primera pantalla donde se debe seleccionar la especialidad de la programación que se desea consultar.

En la figura 5.10 se muestran los resultados de la programación para esa especialidad, el sistema carga desde la base de datos toda la información relacionada con la porgramacion planificada por los jefes de departamentos de la Universidad.

Figura 5.9. Vista de la interfaz consultar programación primera pantalla

Figura 5.10. Vista de la interfaz consultar programación segunda pantalla

Nombre del fichero: vprogr.php

Descripción: Archivo para visualizar la programación.

Código Fuente:

```
<?PHP
if(basename($_SERVER['PHP_SELF'])=="vprogr.php")
{
echo "Intento de hacking bloqueado.";
exit;
}
include "cla_prog.php";

$programa=new programacion();
$programa->consultar();
```

Nombre del fichero: cla_prog.php

Descripción: Declaración de la clase programación desde la cual se manejan todos los métodos que ejecutan y manipulan información referente a la programación y reportes (aquí se mencionan solo los referentes a consultar programación)

Código Fuente:

```
////////////////////////////////////////////
////////////////////////////////////////////
////////////////////////////////////////////
////Muestra la programacion por especialidad/////
////////////////////////////////////////////
////////////////////////////////////////////
////////////////////////////////////////////

public function consultar()
{
if($_SESSION["permisos"][16] || $_SESSION["permisos"][17]  || $_SESSION["permisos"][18])
{

if($_GET['sub']=="borrar" && $_GET['cod'] && $_GET["cod"] && $_GET["secc"])
{
$this->eliminar($_GET["seq"],$_GET["cod"],$_GET["secc"]);
}
$pasar=null;
if($_GET['ob'])
{
$pasar="&ob=".$_GET['ob'];
}


//PDF
include('pdf/class.ezpdf.php');
$pdf =& new Cezpdf('LETTER','landscape');
$pdf->selectFont('pdf/fonts/Courier.afm');
$pdf->ezText("\n\n",12);
//$pdf->ezImage('imagenes/logoudo.jpg', 0, '60' , 'none', 'center');
$pdf->ezText("<b>U N I V E R S I D A D   D E   O R I E N T
E</b>\n",12,array('justification'=>'center'));
$pdf->ezText("<b>N U C L E O   D E   A N Z O A T E G U
I</b>\n",12,array('justification'=>'center'));
//FPDF

$espec=null;
$donde=null;
$resto=null;

$espec=$this->proceso->consulta_t_simple("especialidades",null);

$periodo=$this->proceso->consulta_t_simple("periodo",null);
$periodo=$periodo->FetchNextObject();
$fe=substr($periodo->FECHA_I,0,4);


echo '
```

```
<BR>
<BR>
<div align=center><form method="get" action="index.php" name="buscar">
<input type="hidden" id="sec" name="sec" value="vprogr">
<table style="text-align: left; width: 75%;" border="0" cellpadding="2" cellspacing="2">
<tbody>
<tr style="color: white; font-weight: bold;" align="center">
<td style="background-color: rgb(81, 105, 121);" colspan="10" rowspan="1"><img
src="imagenes/conprog.png" border="0"></img></td>
</tr>
<tr style="color: white; font-weight: bold;" align="center">
<td style="background-color: rgb(81, 105, 121);" colspan="10"
rowspan="1"><b>V&aacute;lido para el per&iacute;odo acad&eacute;mico '.$periodo-
>PERIODO.' del '.$fe.'</b></td>
</tr>

<tr>
<td style="width: 411px; background-color: rgb(203, 216, 224);">Especialidad</td>
<td colspan="9" rowspan="1" style="text-align: left; width: 58px; background-color: rgb(203,
216, 224);"><select name="codespe" onchange="this.form.submit()">
';


while ($res=$espec->FetchNextObject())
        {
                if($_GET["codespe"]==$res->ESPE)
                        echo "<OPTION VALUE=\"".$res->ESPE."\" selected=\"selected\">";
                else
                        echo "<OPTION VALUE=\"".$res->ESPE."\">";
                        echo $res->ESPECIALID;
                        echo "</OPTION>";
        }

echo'


</select></td>
</tr>
';

if($_GET["codespe"])
{
        $espe=$_GET["codespe"];
        $prim=$this->proceso->consulta_t_simple("especialidades",$espe);
        $prim=$prim->FetchNextObject();
}
else
{
        $prim=$this->proceso->consulta_t_simple("especialidades",null);
        $prim=$prim->FetchNextObject();
        $espe=$prim->ESPE;
```

```
}
$donde="where p.ESPE=".$espe."";
$resto="&codespe=".$espe;
$pagilimite=9999; // Un numero muy grande para que no requiera otra pagina.
$pagistro="index.php?sec=vprogr";

//PDF
$pdf->ezText("<b> P R O G R A M A C I O N    A C A D E M I C
A</b>\n\n",12,array('justification'=>'center'));
$pdf->ezText("<b>[".$espe."] - ".$prim-
>ESPECIALID."</b>\n\n",12,array('justification'=>'center'));
$pdf->ezText("<b>PERIODO ACADEMICO ".$periodo->PERIODO." DEL
".$fe."</b>\n\n",12,array('justification'=>'center'));
//EPDF

for ($i=1;$i<=20;$i++)
{
        unset($data);
        $paginar=$this->proceso->consultar_todos("programacion",$_GET["ob"], $espe."
and P4.SEMESTRE=".$i."",null);
        //Incluimos nuestro fichero de paginacion
        include("paginador.php");
        $todos=$this->proceso->consultar_todos("programacion",$_GET["ob"],$espe." and
P4.SEMESTRE=".$i."",$pagisql);

if($todos->RecordCount()>0)
{
        if($i==15)
        {
                //PDF
                $pdf->ezText("<b>ELECTIVAS</b>\n",12,array('justification'=>'center'));
                //EPDF
                echo '<tr style="color: white; font-weight: bold;" align="center"><td
style="background-color: rgb(81, 105, 121);" colspan="11"
rowspan="1">ELECTIVAS</td><tr>';
        }
else
{
        //PDF
        $pdf->ezText("<b>SEMESTRE ".$i."</b>\n",12,array('justification'=>'center'));
        //EPDF
        echo '<tr style="color: white; font-weight: bold;" align="center"><td
style="background-color: rgb(81, 105, 121);" colspan="11" rowspan="1">SEMESTRE
'.$i.'</td><tr>';
}

echo'
<tr>
<td style="width: 411px; background-color: rgb(203, 216, 224); "><B><a class="az"
href="index.php?sec=vprogr&ob=codi'.$resto.'">C&oacute;digo</a></B></td>
<td style="width: 411px; background-color: rgb(203, 216, 224); "><B><a class="az"
```

```
href="index.php?sec=vprogr&ob=prer'.$resto.'">Pre-requisitos</a></B></td>
<td style="width: 411px; background-color: rgb(203, 216, 224); "><B><a class="az"
href="index.php?sec=vprogr&ob=asig'.$resto.'">Asignatura</a></B></td>
<td style="width: 200px; background-color: rgb(203, 216, 224); "><B><a class="az"
href="index.php?sec=vprogr&ob=secc'.$resto.'">Seccion</a></B></td>
<td style="width: 300px; background-color: rgb(203, 216, 224); "><B class="az"
>D&iacute;a</B></td>
<td style="width: 900px; background-color: rgb(203, 216, 224); "><B class="az"
>Horario</B></td>
<td style="width: 411px; background-color: rgb(203, 216, 224); "><B
class="az">Aula</B></td>
<td style="width: 411px; background-color: rgb(203, 216, 224); "><B><a class="az"
href="index.php?sec=vprogr&ob=cupo'.$resto.'">Cupo</a></B></td>
<td style="width: 700px; background-color: rgb(203, 216, 224); "><B><a class="az"
href="index.php?sec=vprogr&ob=prof'.$resto.'">Profesor</a></B></td>
<td style="width: 50px; background-color: rgb(203, 216, 224); "><B class="az"
>Borrar</B></td>
</tr>
';
$aux=0;

while (!is_null($todos) && $res=$todos->FetchNextObject())
        {
                $aux++;
                if($aux%2==0)
{
                        $rgb="211, 211, 211";
                }
                else
                {
                        $rgb="230, 230, 230";
                }
                $resultado=$this->proceso->consulta_t_simple("horario",$res->SEQ);
                $row=$resultado->FetchNextObject();


echo "<tr>
<td style=\"width: 411px; background-color: rgb(".$rgb."); \">".$res->CODIGO."</td>
<td style=\"width: 411px; background-color: rgb(".$rgb."); \">".$res->REQUI."</td>
<td style=\"width: 411px; background-color: rgb(".$rgb."); \">".$res->DESCRIP."</td>";
if ($res->CUPO=="00")
        echo "<td style=\"width: 300px; background-color: rgb(".$rgb."); \">".$res-
>SECCION."*</td>";
else
        echo "<td style=\"width: 300px; background-color: rgb(".$rgb."); \">".$res-
>SECCION."</td>";
echo "<td style=\"width: 200px; background-color: rgb(".$rgb."); \">".$row->DIA."</td>
<td style=\"width: 900px; background-color: rgb(".$rgb."); \">".$row->HORAI." a ".$row-
>HORAF."</td>
<td style=\"width: 411px; background-color: rgb(".$rgb."); \">".$row->AULA."</td>";
if ($res->CUPO=="00")
```

```
		echo "<td style=\"width: 411px; background-color: rgb(".$rgb."); \">40</td>";
else
		echo "<td style=\"width: 411px; background-color: rgb(".$rgb."); \">".$res-
>CUPO."</td>";
echo "<td style=\"width: 700px; background-color: rgb(".$rgb."); \">".$res->NOMBRE."</td>
<td style=\"width: 50px; background-color: rgb(".$rgb."); \"><a
href=\"index.php?sec=vprogr".$pasar.$exportar.$resto."&sub=borrar&seq=".$res-
>SEQ."&cod=".$res->CODIGO."&secc=".$res->SECCION."\" onclick=\"return confirmar('Se
eliminara de la programacion la materia ".$res->DESCRIP." seccion ".$res->SECCION."
desea proseguir?')\"><img src=\"imagenes/b_drop.png\" border=\"0\"></img></a></td>
</tr>";


//pdf
if ($res->CUPO=="00")
$data[] = array('codigo'=>".$res->CODIGO.",'prerequisitos'=>".$res->REQUI.",
'descripcion'=>".$res->DESCRIP.", 'seccion'=>".$res->SECCION."*', 'dia'=>".$row->DIA.",
'horario'=>".$row->HORAI.". a ".$row->HORAF."., 'aula'=>".$row->AULA.", 'cupo'=>'40',
'profesor'=>".$res->NOMBRE.");
else
$data[] = array('codigo'=>".$res->CODIGO.",'prerequisitos'=>".$res->REQUI.",
'descripcion'=>".$res->DESCRIP.", 'seccion'=>".$res->SECCION.", 'dia'=>".$row->DIA.",
'horario'=>".$row->HORAI.". a ".$row->HORAF."., 'aula'=>".$row->AULA.", 'cupo'=>".$res-
>CUPO.", 'profesor'=>".$res->NOMBRE.");
//epdf

		while($row=$resultado->FetchNextObject())
		{
			echo "<tr>
				<td style=\"width: 411px; background-color: rgb(".$rgb.");
\"> </td>
				<td style=\"width: 411px; background-color: rgb(".$rgb.");
\"> </td>
				<td style=\"width: 300px; background-color: rgb(".$rgb.");
\"> </td>
				<td style=\"width: 300px; background-color: rgb(".$rgb.");
\"> </td>
				<td style=\"width: 200px; background-color: rgb(".$rgb."); \">".$row-
>DIA."</td>
				<td style=\"width: 900px; background-color: rgb(".$rgb."); \">".$row-
>HORAI." a ".$row->HORAF."</td>
				<td style=\"width: 411px; background-color: rgb(".$rgb."); \">".$row-
>AULA."</td>
				<td style=\"width: 411px; background-color: rgb(".$rgb.");
\"> </td>


		<td style=\"width: 700px; background-color: rgb(".$rgb."); \"> </td>
				<td style=\"width: 50px; background-color: rgb(".$rgb.");
\"> </td>
				</tr>";
```

```
//pdf
$data[] = array('codigo'=>'','prerequisitos'=>'', 'descripcion'=>'', 'seccion'=>'', 'dia'=>''.$row-
>DIA.'', 'horario'=>''.$row->HORAI.'. a '.$row->HORAF.'.', 'aula'=>''.$row->AULA.'', 'cupo'=>'',
'profesor'=>'');
//epdf
                    }          }
//pdf
$titles = array('codigo'=>'CODIGO','prerequisitos'=>'P.REQ.', 'descripcion'=>'ASIGNATURA',
'seccion'=>'SECCION', 'dia'=>'DIA', 'horario'=>'HORARIO', 'aula'=>'AULA', 'cupo'=>'CUPO',
'profesor'=>'PROFESOR');

//$opciones= array('showHeadings'=>0,'shaded'=>0,'showLines'=>0);
$pdf-
>ezTable($data,$titles,'',array('showHeadings'=>1,'shaded'=>1,'showLines'=>1,'shadeCol'
=>array(0.9,0.9,0.9),'titleFontSize' => 9,'FontSize' => 8,'width' => 700,
'cols'=>array('codigo'=>array('width'=>60),'prerequisitos'=>array('width'=>60),'descripcion'=>a
rray('width'=>170),'seccion'=>array('width'=>55,'justification'=>'center'),'dia'=>array('width'=>7
0),'horario'=>array('width'=>140),'aula'=>array('width'=>50,'justification'=>'center'),'cupo'=>arr
ay('width'=>35,'justification'=>'center'),'profesor'=>array('width'=>100))));
$pdf->ezText("\n\n",12);
//EPDF }}
```

- **Modelo de gestión de contenido de las paginas modificar**

Este grupo de páginas está conformado por seis (6) páginas, construidas con la finalidad de modificar los campos de los registros relacionados con la porgamación académica. Solo los profesores y el administrador pueden modificar datos. Para efectos de practicidad, sólo se colocará un ejemplo de este tipo de páginas, debido a la masiva cantidad de páginas que se tendrían que representar.

A continuación se presenta la vista de la interfaz para modificar asignaturas en la figura 5.11 y el código fuente de la misma.

Figura 5.11. Vista de la interfaz modificar asignaturas

Nombre del fichero: asignaturas.php

Descripción: Este archivo muestra el formulario para editar las asignaturas.

Código Fuente:

```
//Formulario html de ediciÃ³n de asignaturas
echo '
<BR>
<div align="center" id="ayuda" style="display: none"><table border="0" cellspacing="2"
cellpadding="8" bdcolor="black" width="420px">
<tr bgcolor="#FF6600" align="center"><td><b><font color="white"><img
src="imagenes/s_notice.png" border="0"></img>  SACPA: Sistema de
ayuda</font></b></td></tr>
<tr bgcolor="yellow" align="left"><td>  Modificando datos de una Asignatura:<br>
1) El campo <b>C&oacute;digo</b> No puede ser editado, si desea modificarlo debe
eliminar el registro y cargarlo nuevamente.<br>
2) El campo <b>Descripci&oacute;n</b> se refiere al nombre de la asignatura, debe
contener por lo menos 5 caracteres alfanum&eacute;ricos. Ejemplo: Matem&aacute;ticas
I.</td></tr>
</table></div>
<div align=center><form method="post" action="index.php?sec=aasig" name="entrar">
<input type="hidden" name="actualizar" value="paso2">
<table style="text-align: left; width: 420px;" border="0" cellpadding="2" cellspacing="2">
<tbody>
<tr style="color: white; font-weight: bold;" align="center">
<td style="background-color: rgb(81, 105, 121);" colspan="3" rowspan="1"><img
src="imagenes/ediasig.png" border="0"></img></td>
</tr>
<tr>
<input type="hidden" maxlength="7" size="8" name="codigo" value="'.$codigo.'">
<td style="width: 411px; background-color: rgb(203, 216, 224); ">Codigo</td>
<td colspan="2" rowspan="1" style="text-align: left; width: 58px; background-color: rgb(203,
216, 224);"><input maxlength="7" size="8" name="codigo2" value="'.$codigo.'"
disabled></td>
```

```
</tr>
<tr>
<td style="width: 411px; background-color: rgb(203, 216, 224);">Descripcion</td>
<td colspan="2" rowspan="1" style="text-align: left; width: 58px; background-color: rgb(203,
216, 224);"><input maxlength="30" size="31" name="descripcion"
value="'.$descripcion.'"></td>
</select></td>
</tr>
<tr>
<td rowspan="1" colspan="4" style="text-align: center; background-color: rgb(203, 216,
224);"><button value="Enviar" name="Enviar" style="width: 80px; height: 22px;" id="enviar"
type="submit">Actualizar</button> <button type="reset" value="Limpiar"
name="Borrar" style="width: 55px; height: 22px;">Borrar</button></td>
</tr>
</tbody>
</table>
<br>
</form></div>';
//final de codigo html de edición de asignaturas
}
```

Nombre del fichero: cla_asig.php

Descripción: Declaración de la clase asignatura desde la cual se manejan todos los métodos que manipulen la información referente a las asignaturas (aquí se mencionan

solo los referentes a editar asignaturas).

Código Fuente:

```
/*Este método nos permite eliminar asignaturas de la base de datos, recibe como Único
parametro
una cadena contentiva del codigo de materia*/
public function eliminar($codasi)
        {
//Verificamos que el usuario actual posea permisos necesarios.
                if($_SESSION["permisos"][0])
                {
                        $codasi=$this->proteger($codasi);
                        $resultado=$this->proceso-
>consulta_t_simple("horario_x_codigo",$codasi);
                        while($rs=$resultado->FetchNextObject())
//Eliminamos la ocupacion del aula
                        {
                                $inicio=$this->posihora($rs->HORAI);
                                $final=$this->posihora($rs->HORAF);
                                $dia=strtoupper($rs->DIA);
                                if($rs->AULA!="S/A")
```

```
                                        $this->desocupar($inicio,$final,$dia,$rs->AULA);
                                                }
                                $this->proceso-
>eliminar_simple("asignaturas",$codasi,null,null,null,null,null);
//Eliminamos: la materia.
                                $this->proceso->eliminar_complejo("horario_x_materia",$codasi);
//los horarios con dicha materia.
                                $this->proceso->eliminar_complejo("programa_x_materia",$codasi);;
// la programacion de la materia.
                                $this->proceso->eliminar_complejo("pensum_x_materia",$codasi);; //
la materia del pensum.
                                $this->proceso->eliminar_complejo("mapa_x_materia",$codasi);;
// el mapa de aulas ocupado por la materia.
                                echo "<br>";
                                $this->mensajero("Se ha borrado exitosamente la asignatura.<br>");
                                                }
                                else
                        {
                                $this->error="Error 2000: Ud. no dispone de privilegios para eliminar
asignaturas.";
                                $this->errordevalidacion();                     }
                }
```

- Modelo de gestión de contenido de las páginas eliminar

Este grupo de páginas está conformada por ocho (8) páginas, construidas con la finalidad de eliminar los registros relacionados con la programación académica. Solo los profesores y el administrador puede eliminar datos. Para efectos de practicidad, sólo se colocarán dos (2) ejemplos de este tipo de páginas, debido a la masiva cantidad de páginas que se tendrían que representar.

1) Página eliminar asignaturas

A continuación se presenta la vista de la interfaz para eliminar asignaturas en la figura 5.12 y el código fuente de la misma.

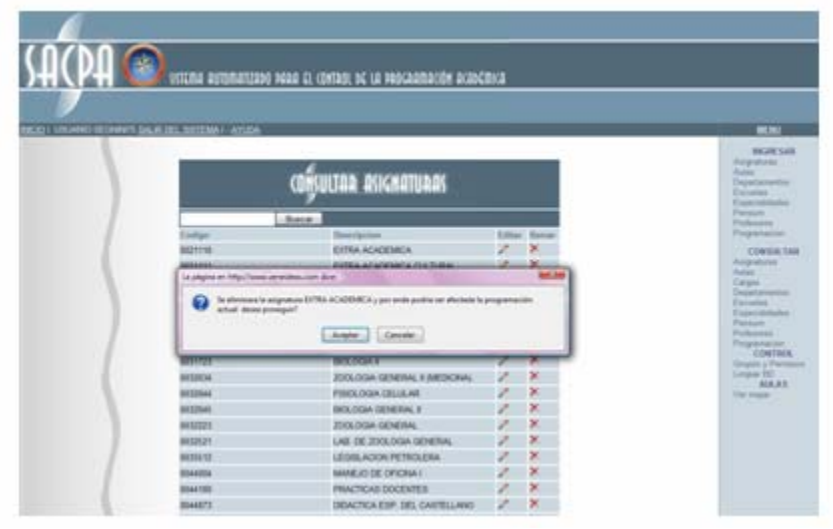


Figura 5.12. Vista de la interfaz eliminar asignaturas

Nombre del fichero: cla_prog.php

Descripción: Declaración de la clase asignatura desde la cual se manejan todos los métodos que manipulen la información referente a las asignaturas (aquí se mencionan solo los referentes a eliminar asignaturas).

Código Fuente:

```
/*Este método nos permite eliminar asignaturas de la base de datos, recibe como Único
parametro
una cadena contentiva del codigo de materia*/
public function eliminar($codasi)
        {
//Verificamos que el usuario actual posea permisos necesarios.
                if($_SESSION["permisos"][0])
                {
                        $codasi=$this->proteger($codasi);
                        $resultado=$this->proceso-
>consulta_t_simple("horario_x_codigo",$codasi);
                        while($rs=$resultado->FetchNextObject())
//Eliminamos la ocupacion del aula
                        {
                                $inicio=$this->posihora($rs->HORAI);
                                $final=$this->posihora($rs->HORAF);
                                $dia=strtoupper($rs->DIA);
                                if($rs->AULA!="S/A")
                                $this->desocupar($inicio,$final,$dia,$rs->AULA);}
                        $this->proceso-
```

```
>eliminar_simple("asignaturas",$codasi,null,null,null,null,null);
//Eliminamos: la materia.
                        $this->proceso->eliminar_complejo("horario_x_materia",$codasi);
//los horarios con dicha materia.
                        $this->proceso->eliminar_complejo("programa_x_materia",$codasi);;
// la programacion de la materia.
                        $this->proceso->eliminar_complejo("pensum_x_materia",$codasi);; //
la materia del pensum.
                        $this->proceso->eliminar_complejo("mapa_x_materia",$codasi);;
// el mapa de aulas ocupado por la materia.
                        echo "<br>";
                        $this->mensajero("Se ha borrado exitosamente la asignatura.<br>");
}
                        else                {
                        $this->error="Error 2000: Ud. no dispone de privilegios para eliminar
asignaturas.";
                        $this->errordevalidacion();                }
}
```

1) Página eliminar programación

A continuación se presenta la vista de la interfaz para eliminar programación en la figura 5.13 y el código fuente de la misma.

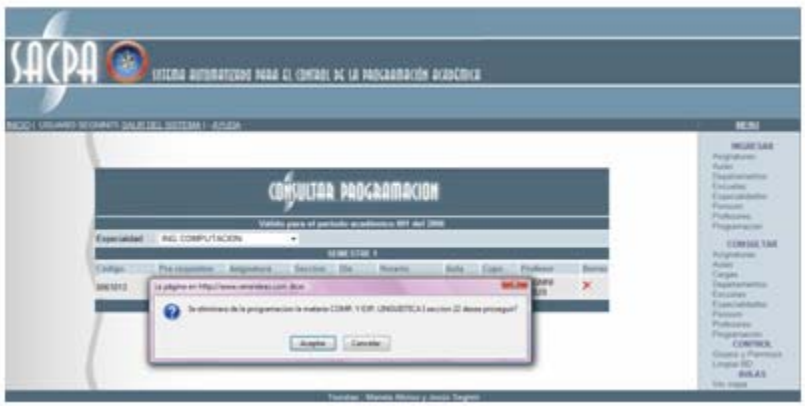


Figura 5.13. Vista de la interfaz eliminar programación

Nombre del fichero: cla_prog.php

Descripción: Declaración de la clase programación desde la cual se manejan todos los métodos que manipulen la información referente a la programación (aquí se mencionan solo los referentes a eliminar programación).

Código Fuente:

```
public function eliminar($seq,$codigo,$seccion)
	{
		if($_SESSION["permisos"][0])
		{
			$resultado=$this->proceso->consulta_t_simple("horario",$seq);
			while($rs=$resultado->FetchNextObject())
	//Eliminamos la ocupacion del aula
			{
				$inicio=$this->posihora($rs->HORAI);
				$final=$this->posihora($rs->HORAF);
				$dia=strtoupper($rs->DIA);
				if($rs->AULA!="S/A")
					$this->desocupar($inicio,$final,$dia,$rs->AULA);

			}

			$this->proceso-
>eliminar_simple("programacion",$seq,null,null,null,null,null); // Eliminamos la prog.
			$this->proceso-
>eliminar_simple("horarios",$seq,null,null,null,null,null); // Eliminamos sus horarios.
			$this->proceso-
>eliminar_simple("mapeo",$codigo,$seccion,null,null,null,null);


			echo "<br>";
			$this->mensajero("Se ha borrado exitosamente la
programaci&oacute;n seleccionada.<br>");

		}

		else

		{
			$this->error="Error 2014: Ud. no dispone de privilegios para eliminar
la asignatura de la programacion.";
			$this->errordevalidacion();
		}

	}
public function eliminard($dia,$horai,$horaf,$aula)

	{
```

```
$inicio=$this->posihora($horai);
$final=$this->posihora($horaf);
$vdia=strtoupper($dia);
$this->desocupar($inicio,$final,$vdia,$aula);
}
```

- **Modelo de gestión de contenido de las paginas reportes**

Este grupo de páginas está conformado por tres (3) páginas, construidas con la finalidad de imprimir o guardar en formato digital la porgramacion, las cargas d elos porfesores y el pensum. Para efectos de practicidad, sólo se colocará un (1) ejemplo de este tipo de páginas: programación.

En la figura 5.14 se muestra la interfaz y el código fuente de la vista de la interfaz generar reportes.

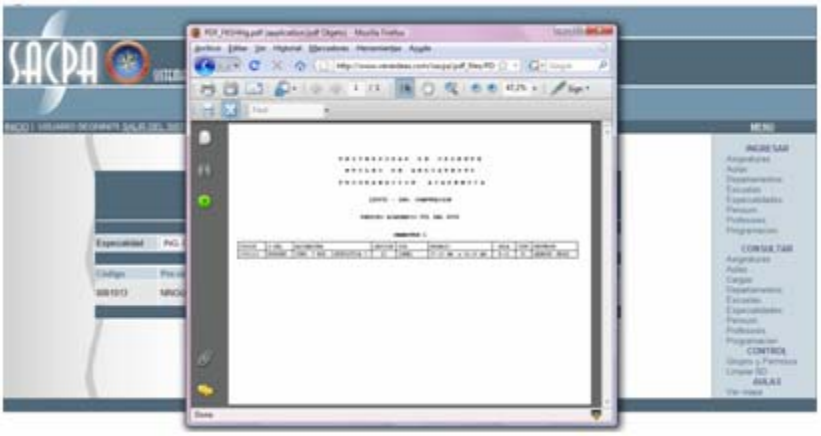


Figura 5.14. Vista de la interfaz Generar Reportes

Nombre del fichero: cla_prog.php

Descripción: Este archivo permite generar los reportes.

Código Fuente:

```
public function eliminar($seq,$codigo,$seccion)
{
```

```
                if($_SESSION["permisos"][0])

                {

                        $resultado=$this->proceso->consulta_t_simple("horario",$seq);

                        while($rs=$resultado->FetchNextObject())

                                //Eliminamos la ocupacion del aula
                        {

                                $inicio=$this->posihora($rs->HORAI);
                                $final=$this->posihora($rs->HORAF);
                                $dia=strtoupper($rs->DIA);
                                if($rs->AULA!="S/A")
                                        $this->desocupar($inicio,$final,$dia,$rs->AULA);

                        }

                        $this->proceso-
>eliminar_simple("programacion",$seq,null,null,null,null,null); // Eliminamos la prog.
                        $this->proceso-
>eliminar_simple("horarios",$seq,null,null,null,null,null); // Eliminamos sus horarios.
                        $this->proceso-
>eliminar_simple("mapeo",$codigo,$seccion,null,null,null,null);


                        echo "<br>";
                        $this->mensajero("Se ha borrado exitosamente la
programaci&oacute;n seleccionada.<br>");
                }

                else

{

                        $this->error="Error 2014: Ud. no dispone de privilegios para eliminar
la asignatura de la programacion.";
                        $this->errordevalidacion();

                }

        }
public function eliminard($dia,$horai,$horaf,$aula)
        {
        $inicio=$this->posihora($horai);
        $final=$this->posihora($horaf);
        $vdia=strtoupper($dia);
        $this->desocupar($inicio,$final,$vdia,$aula);
```

}

- **Modelo de gestión de contenido de la página de permisos de usuarios.**

A esta página solo tiene acceso el administrador, está diseñada para crear grupos de usuarios y otorgarle permisos de acceso a las distintas secciones del sistema.

A continuación se presenta la vista de la interfaz en la figura 5.15 y el código fuente de la misma.

Figura 5.15. Vista de la interfaz Grupos y Permisos

Nombre del fichero: grupos.php

Descripción: Manejo general de formularios y funciones que permiten manejar los grupos de usuarios y asignación de permisos de acceso al sistema.

Código Fuente:

```
<?PHP
if(basename($_SERVER['PHP_SELF'])=="grupos.php")
```

```
{
echo "Intento de hacking bloqueado.";
exit;
}

if($_SESSION["permisos"][21])
{

include "dbcon.php";
include "adodb5/adodb.inc.php";

if($_POST['agregar'])
{

$permi='';
if($_POST['asigtot'])
$permi=$permi.'1';
else
$permi=$permi.'0';

if($_POST['asigver'])
$permi=$permi.'1';
else
$permi=$permi.'0';

if($_POST['aulatot'])
$permi=$permi.'1';
else
$permi=$permi.'0';

if($_POST['aulaver'])
$permi=$permi.'1';
else
$permi=$permi.'0';

if($_POST['depatot'])
$permi=$permi.'1';
else
$permi=$permi.'0';

if($_POST['depaver'])
$permi=$permi.'1';
else
$permi=$permi.'0';

if($_POST['escutot'])
$permi=$permi.'1';
else
$permi=$permi.'0';

if($_POST['escuver'])
```

```
$permi=$permi.'1';
else
$permi=$permi.'0';

if($_POST['espetot'])
$permi=$permi.'1';
else
$permi=$permi.'0';

if($_POST['espever'])
$permi=$permi.'1';
else
$permi=$permi.'0';

if($_POST['penstot'])
$permi=$permi.'1';
else
$permi=$permi.'0';

if($_POST['pensdep'])
$permi=$permi.'1';
else
$permi=$permi.'0';

if($_POST['pensver'])
$permi=$permi.'1';
else
$permi=$permi.'0';

if($_POST['proftot'])
$permi=$permi.'1';
else
$permi=$permi.'0';

if($_POST['profdep'])
$permi=$permi.'1';
else
$permi=$permi.'0';

if($_POST['profver'])
$permi=$permi.'1';
else
$permi=$permi.'0';

if($_POST['progtot'])
$permi=$permi.'1';
else
$permi=$permi.'0';

if($_POST['progdep'])
$permi=$permi.'1';
```

```
else
$permi=$permi.'0';

if($_POST['progver'])
$permi=$permi.'1';
else
$permi=$permi.'0';
if($_POST['cargtot'])
$permi=$permi.'1';
else
$permi=$permi.'0';
if($_POST['cargdep'])
$permi=$permi.'1';
else
$permi=$permi.'0';
if($_POST['admitot'])
$permi=$permi.'1';
else
$permi=$permi.'0';

if($_POST['mapatot'])
$permi=$permi.'1';
else
$permi=$permi.'0';
$permi=$permi."00000000000"; //permisos aun sin utilizar

//Insertamos el nuevo grupo de usuarios co sus respectivos permisos
$db = NewADOConnection("$MANEJADOR_SQL");
$db->Connect("$LUGAR_SQL", "$USUARIO_SQL", "$CLAVE_SQL", "$BD_SQL");
$resultado = $db->Execute("INSERT INTO permisos (ID,GRUPO,PERMISOS)
                           VALUES ('','".$_POST['grupo']."', '".$permi."')");

if ($resultado === false) die("La consulta a la BD fallooo");
else
echo "<div align=\"center\"><b>El grupo se ha agregado con &eacute;xito</b></div>";
}
else
if($_POST['actualizar'])
{
$permi='';
if($_POST['easigtot'])
$permi=$permi.'1';
else
$permi=$permi.'0';
if($_POST['easigver'])
$permi=$permi.'1';
else
$permi=$permi.'0';
if($_POST['eaulatot'])
$permi=$permi.'1';
else
```

```
$permi=$permi.'0';
if($_POST['eaulaver'])
$permi=$permi.'1';
else
$permi=$permi.'0';
if($_POST['edepatot'])
$permi=$permi.'1';
else
$permi=$permi.'0';
if($_POST['edepaver'])
$permi=$permi.'1';
else
$permi=$permi.'0';
if($_POST['eescutot'])
$permi=$permi.'1';
else
$permi=$permi.'0';
if($_POST['eescuver'])
$permi=$permi.'1';
else
$permi=$permi.'0';
if($_POST['eespetot'])
$permi=$permi.'1';
else
$permi=$permi.'0';
if($_POST['eespever'])
$permi=$permi.'1';
else
$permi=$permi.'0';
if($_POST['epenstot'])
$permi=$permi.'1';
else
$permi=$permi.'0';
if($_POST['epensdep'])
$permi=$permi.'1';
else
$permi=$permi.'0';
if($_POST['epensver'])
$permi=$permi.'1';
else
$permi=$permi.'0';
if($_POST['eproftot'])
$permi=$permi.'1';
else
$permi=$permi.'0';
if($_POST['eprofdep'])
$permi=$permi.'1';
else
$permi=$permi.'0';
if($_POST['eprofver'])
$permi=$permi.'1';
```

```
else
$permi=$permi.'0';

if($_POST['eprogtot'])
$permi=$permi.'1';
else
$permi=$permi.'0';
if($_POST['eprogdep'])
$permi=$permi.'1';
else
$permi=$permi.'0';
if($_POST['eprogver'])
$permi=$permi.'1';
else
$permi=$permi.'0';
if($_POST['ecargtot'])
$permi=$permi.'1';
else
$permi=$permi.'0';
if($_POST['ecargdep'])
$permi=$permi.'1';
else
$permi=$permi.'0';
if($_POST['eadmitot'])
$permi=$permi.'1';
else
$permi=$permi.'0';
if($_POST['emapatot'])
$permi=$permi.'1';
else
$permi=$permi.'0';
$permi=$permi."00000000000"; //permisos aun sin utilizar
//actualizamos los permisos del grupo
$db = NewADOConnection("$MANEJADOR_SQL");
$db->Connect("$LUGAR_SQL", "$USUARIO_SQL", "$CLAVE_SQL", "$BD_SQL");
$resultado = $db->Execute("UPDATE permisos SET PERMISOS='".$permi."' where
ID='".$_POST['idgrupo']."'");
if ($resultado === false) die("La consulta a la BD fallooo");
else
echo "<div align=\"center\"><b>El grupo se ha actualizado con &eacute;xito</b></div>";
}
$fallo=false;
//Consultamos todos los grupos existentes
$db = NewADOConnection("$MANEJADOR_SQL");
$db->Connect("$LUGAR_SQL", "$USUARIO_SQL", "$CLAVE_SQL", "$BD_SQL");
$grupose = $db->Execute("SELECT * FROM permisos ORDER BY GRUPO");
if ($grupose === false)
{
$fallo=true;
};
?>
```

```
<BR>
<div align="center" id="ayuda" style="display: none"><table border="0" cellspacing="2"
cellpadding="8" bdcolor="black" width="420px">
<tr bgcolor="#FF6600" align="center"><td><b><font color="white"><img
src="imagenes/s_notice.png" border="0"></img>  SACPA: Sistema de
ayuda</font></b></td></tr>
<tr bgcolor="yellow" align="left"><td>  Grupos de usuarios y sus privilegios:<br>
Desde esta secci&oacute;n se puede agregar grupos de usuarios y sus privilegios,
as&iacute; como modificar los permisos en grupos ya existentes.</td></tr>
</table></div>
<div align=center><form method="post" action="index.php?sec=agrup" name="entrar"
id="1">
<table style="text-align: center; width: 740px" border="0" cellpadding="2" cellspacing="2" >
<tbody>
<tr>
<td>
<input type="hidden" name="agregar" value="true">
<table style="text-align: left; width: 420px;" border="0" cellpadding="2" cellspacing="2">
<tbody>
<tr style="color: white; font-weight: bold;" align="center">
<td style="background-color: rgb(81, 105, 121);" colspan="3" rowspan="1"><img
src="imagenes/car_gru.png" border="0"></img></td>
</tr>
<tr style="color: white; font-weight: bold;" align="center">
<td style="background-color: rgb(81, 105, 121);" colspan="3" rowspan="1">AGREGAR
GRUPO</td>
</tr>
<tr>
<td style="width: 411px; background-color: rgb(203, 216, 224); ">Nombre</td>
<td colspan="2" rowspan="1" style="text-align: left; width: 58px; background-color: rgb(203,
216, 224);"><input maxlength="30" size=33" name="grupo"></td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224); "><b>Permisos
Asignaturas</b></td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224);">Control total
<input type="checkbox"  name =asigtot /> Ver<input type="checkbox"  name =asigver />
</td>
</td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224); "><b>Permisos
Aulas</b></td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224);">Control total
<input type="checkbox"  name =aulatot /> Ver<input type="checkbox"  name =aulaver />
</td>
</td>
```

```
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224); "><b>Permisos
Departamentos</b></td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224);">Control total
<input type="checkbox"  name =depatot /> Ver<input type="checkbox"  name =depaver />
</td>
</td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224); "><b>Permisos
Escuelas</b></td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224);">Control total
<input type="checkbox"  name =escutot /> Ver<input type="checkbox"  name =escuver />
</td>
</td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224); "><b>Permisos
Especialidades</b></td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224);">Control total
<input type="checkbox"  name =espetot /> Ver<input type="checkbox"  name =espever />
</td>
</td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224); "><b>Permisos
Pensum</b></td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224);">Control total
<input type="checkbox"  name =penstot /> Total DPTO <input type="checkbox"  name
=pensdep /> Ver<input type="checkbox"  name =pensver /> </td>
</td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224); "><b>Permisos
Profesores</b></td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224);">Control total
<input type="checkbox"  name =proftot /> Total DPTO <input type="checkbox"  name
=profdep /> Ver<input type="checkbox"  name =profver /> </td>
</td>
</tr>
```

```
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224); "><b>Permisos
Programacion</b></td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224);">Control total
<input type="checkbox"  name =progtot /> Total DPTO <input type="checkbox"  name
=progdep /> Ver<input type="checkbox"  name =progver /> </td>
</td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224); "><b>Permisos de
Cargas</b></td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224);">Total DPTO
<input type="checkbox"  name =cargtot> Carga individual <input type="checkbox"  name
=cargdep> </td>
</td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224);
"><b>Administracion</b></td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224);">Control
administrativo <input type="checkbox"  name = admitot> </td>
</td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224); "><b>Mapa de
aulas</b></td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224);"> Ver mapa <input
type="checkbox"  name = mapatot > </td>
</td>
</tr>
<tr>
<td rowspan="1" colspan="4" style="text-align: center; background-color: rgb(203, 216,
224);"><button value="Enviar" name="Enviar" style="width: 55px; height: 22px;" id="enviar"
type="submit">Cargar</button> <button type="reset" value="Limpiar" name="Borrar"
style="width: 55px; height: 22px;">Borrar</button></td>
</tr>
</tbody>
</table>
</form>
</td>
<td>
<table style="text-align: left; width: 420px;" border="0" cellpadding="2" cellspacing="2">
<tbody>
```

```
<tr style="color: white; font-weight: bold;" align="center">
<td style="background-color: rgb(81, 105, 121);" colspan="3" rowspan="1"><img
src="imagenes/adm_gru.png" border="0"></img></td>
</tr>
<tr style="color: white; font-weight: bold;" align="center">
<td style="background-color: rgb(81, 105, 121);" colspan="3" rowspan="1">EDITAR
GRUPO</td>
</tr>
<tr>
<td style="width: 100px; background-color: rgb(203, 216, 224); ">Grupos</td>
<td colspan="2" rowspan="1" style="text-align: left; width: 310px; background-color: rgb(203,
216, 224);">
<form method="post" action="index.php?sec=agrup" name="edicion" id="2">
<input type="hidden" name="seleccion" value="true">
<select name="grupos" onchange="edicion.submit()">
<option value="null">Seleccione un grupo</option>
<?php
while($rs=$grupose->FetchNextObject())
{
echo '<option value='.$rs->ID.'>'.$rs->GRUPO.'</option>';
}
if($_POST['seleccion'] && $_POST['grupos']!="null")
{
$db = NewADOConnection("$MANEJADOR_SQL");
$db->Connect("$LUGAR_SQL", "$USUARIO_SQL", "$CLAVE_SQL", "$BD_SQL");
$edicion = $db->Execute("SELECT * FROM permisos WHERE ID='".$_POST["grupos"]."'");
$ind=0;
$rs=$edicion->FetchNextObject();
if($rs->PERMISOS[0]==1)
$asigtot="checked";
else
$asigtot="";
if($rs->PERMISOS[1]==1)
$asigver="checked";
else
$asigver="";
if($rs->PERMISOS[2]==1)
$aulatot="checked";
else
$aulatot="";
if($rs->PERMISOS[3]==1)
$aulaver="checked";
else
$aulaver="";
if($rs->PERMISOS[4]==1)
$depatot="checked";
else
$depatot="";
if($rs->PERMISOS[5]==1)
$depaver="checked";
else
```

```
$depaver="";
if($rs->PERMISOS[6]==1)
$escutot="checked";
else
$escutot="";
if($rs->PERMISOS[7]==1)
$escuver="checked";
else
$escuver="";
if($rs->PERMISOS[8]==1)
$espetot="checked";
else
$espetot="";
if($rs->PERMISOS[9]==1)
$espever="checked";
else
$espever="";
if($rs->PERMISOS[10]==1)
$penstot="checked";
else
$penstot="";
if($rs->PERMISOS[11]==1)
$pensdep="checked";
else
$pensdep="";
if($rs->PERMISOS[12]==1)
$pensver="checked";
else
$pensver="";
if($rs->PERMISOS[13]==1)
$proftot="checked";
else
$proftot="";
if($rs->PERMISOS[14]==1)
$profdep="checked";
else
$profdep="";
if($rs->PERMISOS[15]==1)
$profver="checked";
else
$profver="";
if($rs->PERMISOS[16]==1)
$progtot="checked";
else
$progtot="";
if($rs->PERMISOS[17]==1)
$progdep="checked";
else
$progdep="";
if($rs->PERMISOS[18]==1)
$progver="checked";
```

```
else
$progver="";
if($rs->PERMISOS[19]==1)
$cargtot="checked";
else
$cargtot="";
if($rs->PERMISOS[20]==1)
$cargdep="checked";
else
$cargdep="";
if($rs->PERMISOS[21]==1)
$admitot="checked";
else
$admitot="";
if($rs->PERMISOS[22]==1)
$mapatot="checked";
else
$mapatot="";
}
}
?>
</select>
</form>
<form method="post" action="index.php?sec=agrup" name="actualizar" id="3">
<input type="hidden" name="actualizar" value="true">
<input type="hidden" name="idgrupo" value="<?=$rs->ID?>">
</td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224); "><b>Permisos
Asignaturas</b></td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224);">Control total
<input type="checkbox"  name =easigtot <?=$asigtot?>> Ver<input type="checkbox"  name
=easigver <?=$asigver?>> </td>
</td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224); "><b>Permisos
Aulas</b></td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224);">Control total
<input type="checkbox"  name =eaulatot <?=$aulatot?>> Ver<input type="checkbox"  name
=eaulaver <?=$aulaver?>> </td>
</td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224); "><b>Permisos
Departamentos</b></td>
```

```
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224);">Control total
<input type="checkbox"  name =edepatot <?=$depatot?>> Ver<input type="checkbox"  name
=edepaver <?=$depaver?>> </td>
</td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224); "><b>Permisos
Escuelas</b></td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224);">Control total
<input type="checkbox"  name =eescutot <?=$escutot?>> Ver<input type="checkbox"  name
=eescuver <?=$escuver?>> </td>
</td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224); "><b>Permisos
Especialidades</b></td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224);">Control total
<input type="checkbox"  name =eespetot <?=$espetot?>> Ver<input type="checkbox"  name
=eespever <?=$espever?>> </td>
</td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224); "><b>Permisos
Pensum</b></td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224);">Control total
<input type="checkbox"  name =epenstot <?=$penstot?>> Total DPTO <input
type="checkbox"  name =epensdep <?=$pensdep?>> Ver<input type="checkbox"  name
=epensver <?=$pensver?>> </td>
</td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224); "><b>Permisos
Profesores</b></td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224);">Control total
<input type="checkbox"  name =eproftot <?=$proftot?>> Total DPTO <input type="checkbox"
name =eprofdep <?=$profdep?>> Ver<input type="checkbox"  name =eprofver
<?=$profver?>> </td>
</td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224); "><b>Permisos
```

```
Programacion</b></td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224);">Control total
<input type="checkbox"  name =eprogtot <?=$progtot?>> Total DPTO <input
type="checkbox"  name =eprogdep <?=$progdep?>> Ver<input type="checkbox"  name
=eprogver <?=$progver?>> </td>
</td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224); "><b>Permisos de
Cargas</b></td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224);">Total DPTO
<input type="checkbox"  name =ecargtot <?=$cargtot?>> Carga individual <input
type="checkbox"  name =ecargdep <?=$cargdep?>> </td>
</td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224);
"><b>Administracion</b></td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224);">Control
administrativo <input type="checkbox"  name =eadmitot <?=$admitot?>> </td>
</td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224); "><b>Mapa de
aulas</b></td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224);"> Ver mapa <input
type="checkbox"  name =emapatot <?=$mapatot?>> </td>
</td>
</tr>
<tr>
<td rowspan="1" colspan="4" style="text-align: center; background-color: rgb(203, 216,
224);"><button value="Enviar" name="Enviar" style="width: 80px; height: 22px;" id="enviar"
type="submit">Actualizar</button> <button type="reset" value="Limpiar"
name="Borrar" style="width: 55px; height: 22px;">Borrar</button></td>
</tr>
</tbody>
</table>
</form>
</td>
</tr>
</tbody>
</table>
<br>
```

```
</div>
```

Nombre del fichero: menu.php

Descripcion: Muestra las distintas opciones del menú dependiendo del acceso del usuario al sistema

Código Fuente:

```
<?php
if(basename($_SERVER['PHP_SELF'])=="menu.php"){
echo "Intento de hacking bloqueado.";
exit;}
echo '<div style="margin-left: 20px; width: 80%;" id="navmenu" left="" align="left"><p>';
if($_SESSION["permisos"][0] || $_SESSION["permisos"][2] || $_SESSION["permisos"][0]

|| $_SESSION["permisos"][6] || $_SESSION["permisos"][8] || $_SESSION["permisos"][10]
|| $_SESSION["permisos"][11] || $_SESSION["permisos"][13] || $_SESSION["permisos"][14]
|| $_SESSION["permisos"][16] || $_SESSION["permisos"][17])
echo '<center><menmas><b>INGRESAR</b></menmas></center>';
if($_SESSION["permisos"][0])
echo '<a class="az" href="index.php?sec=aasig">Asignaturas</a><br>';
if($_SESSION["permisos"][2])
echo '<a class="az" href="index.php?sec=aaula">Aulas</a><br>';
if($_SESSION["permisos"][4])
echo '<a class="az" href="index.php?sec=adpto">Departamentos</a><br>';
if($_SESSION["permisos"][6])
echo '<a class="az" href="index.php?sec=aescu">Escuelas</a><br>';
if($_SESSION["permisos"][8])
echo '<a class="az" href="index.php?sec=aespe">Especialidades</a><br>';
if($_SESSION["permisos"][10] || $_SESSION["permisos"][11])
echo '<a class="az" href="index.php?sec=apens">Pensum</a><br>';
if($_SESSION["permisos"][13] || $_SESSION["permisos"][14])
echo '<a class="az" href="index.php?sec=aprof">Profesores</a><br>';
if($_SESSION["permisos"][16] || $_SESSION["permisos"][17])
echo '<a class="az" href="index.php?sec=aprog">Programacion</a><p>';
echo '<center><menmas><b>CONSULTAR</b></menmas></center>';
if($_SESSION["permisos"][1] || $_SESSION["permisos"][0])
echo '<a class="az" href="index.php?sec=vasig">Asignaturas</a><br>';
if($_SESSION["permisos"][3] || $_SESSION["permisos"][2])
echo '<a class="az" href="index.php?sec=vaula">Aulas</a><br>';
if($_SESSION["permisos"][19] || $_SESSION["permisos"][20])
echo '<a class="az" href="index.php?sec=vcarg">Cargas</a><br>';
if($_SESSION["permisos"][5] || $_SESSION["permisos"][4])
echo '<a class="az" href="index.php?sec=vdepa">Departamentos</a><br>';
if($_SESSION["permisos"][7] || $_SESSION["permisos"][6])
echo '<a class="az" href="index.php?sec=vescu">Escuelas</a><br>';
if($_SESSION["permisos"][9] || $_SESSION["permisos"][8])
echo '<a class="az" href="index.php?sec=vespe">Especialidades</a><br>';
if($_SESSION["permisos"][22])
```

```
echo '<a class="az" href="index.php?sec=maula">Mapa de aulas</a><br>';
if($_SESSION["permisos"][12] || $_SESSION["permisos"][10] || $_SESSION["permisos"][11])
echo '<a class="az" href="index.php?sec=vpens">Pensum</a><br>';
if($_SESSION["permisos"][15] || $_SESSION["permisos"][13] || $_SESSION["permisos"][14])
echo '<a class="az" href="index.php?sec=vprofe">Profesores</a><br>';
if($_SESSION["permisos"][18]|| $_SESSION["permisos"][16] || $_SESSION["permisos"][17])
echo '<a class="az" href="index.php?sec=vprogr">Programacion</a><br><p>';
if($_SESSION["permisos"][21])
echo '<center><menmas><b>ADMINISTRAR</b></menmas></center>';
if($_SESSION["permisos"][21])
echo '<a class="az" href="index.php?sec=agrup">Grupos y Permisos</a><br>';
if($_SESSION["permisos"][21])
echo '<a class="az" href="index.php?sec=lbdad">Gesti&oacute;n BD</a><br>';
echo '</div>'; ?>
```

- **Modelo de gestión de contenido de la página limpiar base de datos.**

Esta está diseñada para eliminar todos los registros de la base de datos pertenecientes a la tabla de la programación, realizando antes una copia de seguridad, esto para poder cargar la nueva programación. A continuación se presenta la vista de la interfaz en la figura 5.15 y el código fuente.

Figura 5.16. Vista de la interfaz limpiar base de datos

Nombre del fichero: limpiar_bd.php

Descripcion: Archivo que permite realizar rutinas administrativas a la base de

datos del sistema, como pueden ser: backup, restauracion y reinicializacion.

Código Fuente:

```
<?
if(basename($_SERVER['PHP_SELF'])=="limpiar_bd.php")
{
echo "Intento de hacking bloqueado.";
exit;
}
if($_SESSION["permisos"][21])
{
include "dbcon.php";
include "adodb5/adodb.inc.php";
$db = NewADOConnection("$MANEJADOR_SQL");
$db->Connect("$LUGAR_SQL", "$USUARIO_SQL", "$CLAVE_SQL", "$BD_SQL");
$resultado = $db->Execute("SELECT * FROM sacpa.BACKUP ORDER BY ID DESC");
if ($resultado === false)
{
echo "FallÃ³ la consulta de los backups.<BR>";
$actual="Error al obtener el dato.";
}
else
{
$actual=$resultado->FetchNextObject();
}
if($_POST['ejecutar'])
{
//Realizamos el backup en las tablas destinadas para este fin.
if($_POST['progbackup'])
{
$resultado = $db->Execute("TRUNCATE TABLE sacpa.BAULAS");
if ($resultado === false)
echo "Fall&oacute; el vaciado de la tabla BAULAS.<BR>";
$resultado = $db->Execute("INSERT INTO sacpa.BAULAS SELECT * FROM
sacpa.AULAS");
if ($resultado === false)
echo "Fall&oacute; el backup de la tabla AULAS.<BR>";
else
echo "<div align=\"center\"><b>Backup de la tabla AULAS realizado con
&eacute;xito</b></div>";
$resultado = $db->Execute("TRUNCATE TABLE sacpa.BMAPA");
if ($resultado === false)
echo "Fall&oacute; el vaciado de la tabla BMAPA.<BR>";
$resultado = $db->Execute("INSERT INTO sacpa.BMAPA SELECT * FROM sacpa.MAPA");
if ($resultado === false)
echo "Fall&oacute; el backup de la tabla MAPA.<BR>";
else
echo "<div align=\"center\"><b>Backup de la tabla MAPA realizado con
&eacute;xito</b></div>";
$resultado = $db->Execute("TRUNCATE TABLE sacpa.BPROACH");
```

```
if ($resultado === false)
echo "Fall&oacute; el vaciado de la tabla sacpa.BPROACH.<BR>";
$resultado = $db->Execute("INSERT INTO sacpa.BPROACH SELECT * FROM
sacpa.PROACH");
if ($resultado === false)
echo "Fall&oacute; el backup de la tabla sacpa.BPROACH.<BR>";
else
echo "<div align=\"center\"><b>Backup de la tabla PROACH realizado con
&eacute;xito</b></div>";
$resultado = $db->Execute("TRUNCATE TABLE sacpa.BPROGH");
if ($resultado === false)
echo "Fall&oacute; el vaciado de la tabla BPROGH.<BR>";
$resultado = $db->Execute("INSERT INTO sacpa.BPROGH SELECT * FROM
sacpa.PROGH");
if ($resultado === false)
echo "Fall&oacute; el backup de la tabla PROGH.<BR>";
else
echo "<div align=\"center\"><b>Backup de la tabla PROGH realizado con
&eacute;xito</b></div>";
$backup= date("d-m-Y_H:i:s");
$backup=$backup."[".$_SESSION['login']."]";
$resultado = $db->Execute("INSERT INTO sacpa.BACKUP (ID,DATOS) VALUES
('','$backup') ");
if ($resultado === false)
echo "Fall&oacute; el almacenamiento de la tabla BACKUP.<BR>";
else
echo "<div align=\"center\"><b><FONT COLOR=\"GREEN\"> Backup finalizado con
&eacute;xito<br>log almacenado bajo el codigo:".$backup."</b></FONT></div>";
}
if($_POST['progreiniciar'])
{
$db = NewADOConnection("$MANEJADOR_SQL");
$db->Connect("$LUGAR_SQL", "$USUARIO_SQL", "$CLAVE_SQL", "$BD_SQL");
$resultado = $db->Execute("UPDATE sacpa.AULAS SET
LUNES='000000000000000000000000000000000000000000000000000000000000',
MARTES='000000000000000000000000000000000000000000000000000000000000',
MARTES='000000000000000000000000000000000000000000000000000000000000',
MIERCOLES='0000000000000000000000000000000000000000000000000000000000
00',
JUEVES='000000000000000000000000000000000000000000000000000000000000',
VIERNES='000000000000000000000000000000000000000000000000000000000000',
SABADO='000000000000000000000000000000000000000000000000000000000000'
WHERE 1=1");
if ($resultado === false)
echo "Fall&oacute; la reinicializaci&oacute;n de la tabla AULAS.<BR>";
else
echo "<div align=\"center\"><b>Reinicializaci&oacute;n de la tabla AULAS realizado con
&eacute;xito</b></div>";
$resultado = $db->Execute("TRUNCATE TABLE sacpa.MAPA");
if ($resultado === false)
echo "Fall&oacute; la reinicializaci&oacute;n de la tabla MAPA.<BR>";
```

```
else
echo "<div align=\"center\"><b>Reinicializaci&oacute;n de la tabla MAPA realizado con
&eacute;xito</b></div>";
$resultado = $db->Execute("TRUNCATE TABLE sacpa.PROACH");
if ($resultado === false)
echo "Fall&oacute; la reinicializaci&oacute;n de la tabla PROACH.<BR>";
else
echo "<div align=\"center\"><b>Reinicializaci&oacute;n de la tabla PROACH realizado con
&eacute;xito</b></div>";
$resultado = $db->Execute("TRUNCATE TABLE sacpa.PROGH");
if ($resultado === false)
echo "Fall&oacute; la reinicializaci&oacute;n de la tabla PROGH.<BR>";
else
echo "<div align=\"center\"><b>Reinicializaci&oacute;n de la tabla PROGH realizado con
&eacute;xito</b></div>";

}
//Realizamos la restauracion del backup en las tablas originales de sacpa.
if($_POST['progrestaurar'])
{
$resultado = $db->Execute("TRUNCATE TABLE sacpa.AULAS");
if ($resultado === false)
echo "Fall&oacute; el vaciado de la tabla BAULAS.<BR>";
$resultado = $db->Execute("INSERT INTO sacpa.AULAS SELECT * FROM
sacpa.BAULAS");
if ($resultado === false)
echo "Fall&oacute; la restauraci&oacute;n de la tabla AULAS.<BR>";
else
echo "<div align=\"center\"><b>Restauraci&oacute;n de la tabla AULAS realizado con
&eacute;xito</b></div>";
$resultado = $db->Execute("TRUNCATE TABLE sacpa.MAPA");
if ($resultado === false)
echo "Fall&oacute; el vaciado de la tabla MAPA.<BR>";
$resultado = $db->Execute("INSERT INTO sacpa.MAPA SELECT * FROM sacpa.BMAPA");
if ($resultado === false)
echo "Fall&oacute; la restauraci&oacute;n de la tabla MAPA.<BR>";
else
echo "<div align=\"center\"><b>Restauraci&oacute;n de la tabla MAPA realizado con
&eacute;xito</b></div>";
$resultado = $db->Execute("TRUNCATE TABLE sacpa.PROACH");
if ($resultado === false)
echo "Fall&oacute; el vaciado de la tabla PROACH.<BR>";
$resultado = $db->Execute("INSERT INTO sacpa.PROACH SELECT * FROM
sacpa.BPROACH");
if ($resultado === false)
echo "Fall&oacute; la restauraci&oacute; de la tabla PROACH.<BR>";
else
echo "<div align=\"center\"><b>Restauraci&oacute;n de la tabla PROACH realizado con
&eacute;xito</b></div>";
$resultado = $db->Execute("TRUNCATE TABLE sacpa.PROGH");
if ($resultado === false)
```

```
echo "Fall&oacute; el vaciado de la tabla PROGH.<BR>";
$resultado = $db->Execute("INSERT INTO sacpa.PROGH SELECT * FROM
sacpa.BPROGH");
if ($resultado === false)
echo "Fall&oacute; la restauraci&oacute;n de la tabla PROGH.<BR>";
else
echo "<div align=\"center\"><b>Backup de la tabla PROGH realizado con
&eacute;xito</b></div>";
}

}
}
?>
<div align="center" id="ayuda" style="display: none"><table border="0" cellspacing="2"
cellpadding="8" bdcolor="black" width="420px">
<tr bgcolor="#FF6600" align="center"><td><b><font color="white"><img
src="imagenes/s_notice.png" border="0"></img>  SACPA: Sistema de
ayuda</font></b></td></tr>
<tr bgcolor="yellow" align="left"><td>  Operaciones con la base de datos:<br>
<b>Backup:</b> Crea una imagen de la programaci&oacute;n actual del sistema en una
tabla paralela.<br>
<b>Inicializaci&oacute;n:</b> Vacia el contenido de la programaci&oacute;n actual, no se ve
afectado el backup ni las tablas no asociadas al semestre en curso (aulas, pensum, etc.)<br>
<b>Restauraci&oacute;n:</b> Recupera la imagen de la programaci&oacute;n almacenada
previamente en la tabla paralela dispuesta para este fin.<br>
</td></tr>
</table></div>
<BR>
<div align=center><form method="post" action="index.php?sec=lbdad" name="entrar"
id="1">
<input type="hidden" name="ejecutar" value="true">
<table style="text-align: left; width: 420px;" border="0" cellpadding="2" cellspacing="2">
<tbody>
<tr style="color: white; font-weight: bold;" align="center">
<td style="background-color: rgb(81, 105, 121);" colspan="3" rowspan="1"><img
src="imagenes/limbda.png" border="0"></img></td>
</tr>
<tr style="color: white; font-weight: bold;" align="center">
<td style="background-color: rgb(81, 105, 121);" colspan="3" rowspan="1">OPERACIONES
CON LA BASE DE DATOS</td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224);
"><b>Backup</b></td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224);">Crea un backup
de la programaci&oacute;n actual. <input type="checkbox"  name =progbackup /></td>
</td>
</tr>
<tr>
```

```
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224);
"><b>Inicializaci&oacute;n</b></td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224);">Borra la
programaci&oacute;n actual. <input type="checkbox"  name =progreiniciar /></td>
</td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224);
"><b>Restauraci&oacute;n</b></td>
</tr>
<tr>
<td colspan=3 style="width: 411px; background-color: rgb(203, 216, 224);">Restaura el
backup en sistema. <input type="checkbox"  name =progrestaurar /><br>
Backup actual: <?=$actual->DATOS?>
</td>
</td>
</tr>
<tr>
<td rowspan="1" colspan="4" style="text-align: center; background-color: rgb(203, 216,
224);"><button value="Enviar" name="Enviar" style="width: 85px; height: 22px;" id="enviar"
type="submit">Ejecutar</button> <button type="reset" value="Limpiar" name="Borrar"
style="width: 55px; height: 22px;">Borrar</button></td>
</tr>
</tbody>
</table>
</form>
<br></div>
```

5.3 Pruebas

Para cada caso de uso se deben establecer pruebas de aceptación que validen la correcta implementación del caso de uso. El objetivo de la fase de pruebas de un programa es de detectar todo posible mal funcionamiento antes de que entre en producción.

5.3.1 Pruebas de Unidad

Las pruebas de unidad evalúan los componentes implementados como unidades individuales. Para efectuar las pruebas de unidad se utiliza la técnica de pruebas de caja negra.

Las pruebas de caja negra se centran en lo que se espera de un módulo, es decir, intentan encontrar casos en que el módulo no se atiene a su especificación. Por ello se denominan pruebas funcionales, y el probador se limita a suministrarle datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar haciendo el módulo por dentro. Estas pruebas están especialmente indicadas en aquellos módulos que van a ser interfaz con el usuario (en sentido general: teclado, pantalla, ficheros, canales de comunicaciones, entre otros).

5.3.1.1 Partición Equivalente

La partición equivalente es un método de prueba de caja negra que divide el dominio de entrada de un programa en clases de datos a partir de las cuales pueden derivarse casos de prueba. Un caso de prueba ideal de manejo simple descubre una clase de errores, por ejemplo el procesamiento incorrecto de todos los datos de caracteres, es necesaria la ejecución de muchos casos antes de que se observe el error general. La partición equivalente se esfuerza por definir un caso de prueba que descubra ciertas clases de errores, reduciendo así el número total de casos de prueba que deben desarrollarse.

El diseño de casos de prueba para una partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada. Si es posible enlazar un conjunto de objetos mediante relaciones simétricas, transitivas y reflexivas, entonces existe una clase de equivalencia. Una clase de equivalencia representa un conjunto de estados válidos y no válidos para las condiciones de

entrada. Por lo general, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición booleana. Las clases de equivalencia se definen de acuerdo con las siguientes directrices:

- Si una condición de entrada específica un rango, se definen una clase de equivalencia válida y dos no válidas.
- Si una condición de entrada requiere un valor específico, se define una clase de equivalencia válida y dos no válidas.
- Si una condición de entrada específica un miembro de un conjunto, se define una clase de equivalencia válida y dos no válidas.
- Si una condición de entrada es booleana, se define una clase de equivalencia válida y otra no válida.

Al aplicar estas directrices para la derivación de clases de equivalencia, se desarrollarán y ejecutarán los casos de prueba para cada objeto de los datos del dominio de entrada. Los casos de prueba se seleccionan de modo que el mayor número de atributos de clase de equivalencia se ejercita una vez.

5.3.1.2 Identificación de Clases de Equivalencia

1) Sólo números.
2) Caracteres y números.
3) Ningún carácter.

5.3.1.3 Grupo de Tipos de Entrada de Datos

1) Numéricos: recibe de entrada sólo números sin signo.
2) Alfanuméricos: recibe caracteres alfabeticos en mayúscula y minúscula, y caracteres numéricos. Al procesar la entrada, la cadena de caracteres se convierte a mayúscula.

3) No Vacío: se refiere a entradas de datos que deben contener al menos un carácter.

5.3.1.4 Aplicación de Casos de Prueba

La tabla 5.1 que se muestra a continuación describe la evaluación de casos de prueba que se realizaron al sistema.

Grupo	Caso de Prueba	Valida	No valida	Clase de equivalencia
1	123	x		1
1	abcABC		x	2
1	123ABC		x	2
1	abc123		x	2
1	" "		x	3
2	123	x		1
2	123ABC	x		2
2	abcABC	x		2
2	abc123	x		2
2	" "		x	3
3	123	x		1
3	abcABC	x		2
3	123ABC	x		2
3	abc123	x		2
3	" "		x	3

Tabla 5.1. Tabla de evaluación de casos de prueba

- **Caso de prueba: Ingresar asignaturas**

El problema que surgió durante esta prueba es que se pasaban por alto los caracteres: “+”, “-”, “.”, “,”. En la figura 5.17 se muestra el este error.

Figura 5.17. Captura de la interfaz para cargar asignaturas

En este campo solo se debe permitir la entrada de números sin signo. Sin embargo, como se puede observar en la siguiente imagen, los datos se insertaron en la base de datos, (ver figura 5.18).

Figura 5.18. Captura de la interfaz ver asignaturas

El problema se debia a la incorrecta aplicación de una función nativa de PHP que determinaba si una variable era numérica o no:

```
	//chequeamos que el campo cédula solo contenga números
	if(!is_numeric($codigo))
	{
		$this->error=$this->error."-El c&oacute;digo solo puede contener
n&uacute;meros.<br>";
		$ok=$false;
	}
```

Para solucionar este inconveniente, se sustituyó esta comprobación por una que permite, mediante el uso de expresiones regulares, una correcta validación del campo código.

```
$patron = "^[[:digit:]]+$";
//chequeamos que el campo cédula solo contenga números
if(!eregi($patron, $codigo))
{
        $this->error=$this->error."-El c&oacute;digo solo puede contener
n&uacute;meros.<br>";
        $ok=$false;
}
```

Se procedió al borrado del registro (ver figura 5.19).

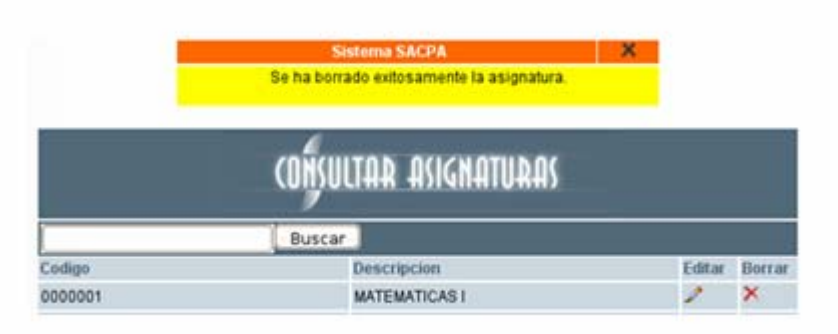


Figura 5.19. Captura de la interfaz consultar asignaturas

Se inserta nuevamente la asignatura con los mismos datos, luego de corregida la validación (ver figura 5.20).

Figura 5.20. Captura de la interfaz cargar asignaturas

Se comprueba la correcta validación de los datos (ver figura 5.21).

Figura 5.21. Captura de la interfaz cargar asignaturas

5.3.2. Pruebas de Integración

Conociendo el funcionamiento interno del producto, se aplican pruebas para asegurarse que "todas las piezas encajan", en otras palabras, están integradas correctamente. Se basa en un examen cercano al detalle procedimental. Estas pruebas se aplicaron a todos los componentes de software desarrollados, comprobando así la exitosa integración de la aplicación. Además, se probaron los enlaces de las páginas, que no hubiesen enlaces rotos o vacios, ni enlaces que direccionarán hacia alguna página indebida. A continuación se presentan ejemplos de algunos casos de prueba de integración que se aplicaron al sistema.

5.3.2.1 Diseño de Casos de Pruebas

Los casos de prueba deben realizarse sabiendo cómo funciona el programa, es decir, como es su desempeño internamente. A continuación se presenta un conjunto de tablas que describen los casos de prueba diseñados y la relación entre cada uno de ellos. Observe desde la tabla 5.2 hasta la 5.7 de manera consecutiva para visualizar el

ejemplo con las asignaturas y desde la tabla 5.7 hasta la 5.10 de manera consecutiva para visualizar el ejemplo con la porgramación.

Caso de prueba	Entrar al sistema
Entrada	Usuario: 12345678, Contraseña: 1234
Resultado	El usuario es autenticado y entra al sistema con sus respectivos permisos de acceso.
Condiciones	El usuario esta creado en la base de datos y no tiene una sesión abierta previamente.
Procedimiento	Escriba su usuario y contraseña en los cuadros de texto correspondiente. Presione el botón entrar. Se ingresa al sistema

Tabla 5.2. Caso de prueba Entrar al sistema

Luego de entrar al sistema procedemos a ingresar una asignatura.

Caso de prueba	Ingresar asignaturas
Entrada	Se desea ingresar la asignatura cuya información es la siguiente: Código: 1234567 Descripción: Nueva Materia
Resultado	Se muestra el formulario para ingresar la información. Los datos son guardados en el servidor.
Condiciones	El usuario está previamente autenticado y tiene permisos de ingresar datos. No existe un registro con ese código.
Procedimiento	Seleccione en el menú INGRESAR el enlace Asignaturas. Se muestra un formulario con los cuadros de texto de entrada. Introduzca los datos de la asignatura: código y descripción en los cuadros de texto que corresponden. Presione el botón cargar.

Tabla 5.3. Caso de prueba ingresar asignaturas

Ahora con los datos recién ingresados, se realizará una consulta:

Caso de prueba	Consultar asignaturas
Entrada	Se desea consultar una asignatura con la siguiente información: Código: 01234567 Descripción: Nueva Materia
Resultado	Se muestra una tabla con los resultados.
Condiciones	El usuario está previamente autenticado y tienen permisos de consultar datos. Existe el el código de la asignatura y la descripción.
Procedimiento	Seleccione en el menú CONSULTAR el enlace Asignaturas. Se muestra una lista con las asignaturas cargadas en el sistema, mediante el buscador puede visualizar una asignatura especifica ingresando el código o la descripción

Tabla 5.4. Caso de prueba consultar asignaturas

Luego de consultar una asignatura esta se puede modificar:

Caso de prueba	Modificar asignaturas
Entrada	Se desea modificar la asignatura con la siguiente información: Código: 1234567 Descripción: Nueva Materia
Resultado	Se muestra un formulario donde se permite modificar el campo descripción.
Condiciones	El usuario está previamente autenticado y tiene permisos de modificar datos. Existe el código de la asignatura y la descripción.
Procedimiento	Seleccione en el menú CONSULTAR el enlace Asignaturas. Se muestra una lista con las asignaturas cargadas en el sistema, mediante el

	buscador puede visualizar una asignatura especifica ingresando el código o la descripción, luego se presiona el botón modificar.

Tabla 5.5. Caso de prueba modificar asignaturas

Luego de consultar una asignatura esta también se puede eliminar:

Caso de prueba	Eliminar asignaturas
Entrada	Se desea eliminar la asignatura con la siguiente información: Código: 1234567 Descripción: Nueva Materia
Resultado	Se muestra un formulario donde se permite eliminar el registro de la asignatura seleccionada.
Condiciones	El usuario está previamente autenticado y tiene permisos de eliminar datos. Existe el código de la asignatura y la descripción.
Procedimiento	Seleccione en el menú CONSULTAR el enlace Asignaturas. Se muestra una lista con las asignaturas cargadas en el sistema, mediante el buscador puede visualizar una asignatura especifica ingresando el código o la descripción, luego se presiona el botón eliminar que se encuentra a la derecha del registro de la asignatura.

Tabla 5.6. Caso de prueba eliminar asignaturas

Ahora el caso de prueba con la programación

Luego de entrar al sistema procedemos a ingresar la programación, para esto debemos tener todos los demás datos de la programación academica cargados en el sistema: asignaturas, aulas, departamentos, escuelas, especialidades, pensum y profesores.

Caso de prueba	Ingresar programacion
Entrada	Se desea ingresar la programación con los siguientes datos: Especialidad: Nueva Carrera Asignatura: 1234567 - Nueva Materia Sección: 01 Cupos: 30 Profesor: Nuevo Profesor Dias: Lunes y Miercoles Horario: 8:00am – 10:00am Aula: Nueva aula
Resultado	El sistema muestra el formulario para cargar los datos de la programación y estos datos son guardados en el servidor.
Condiciones	El usuario está previamente autenticado y tiene permisos de ingresar datos. No existe un registro con ese código.
Procedimiento	Seleccione en el menú INGRESAR el enlace Porgramación. Se muestra un formulario con los cuadros de texto de entrada. Introduzca los datos de la programación: Especialidad, Asignatura, Sección, Cupos, Profesor, Dias, Horario y Aula, en los cuadros de texto que corresponden. Presione el botón cargar.

Tabla 5.7. Caso de prueba Ingresar programación

Ahora con los datos recién ingresados, se realizará una consulta:

Caso de prueba	Consultar programación
Entrada	Se selcciona la especialidad Nueva Materia.
Resultado	Se muestra una tabla con los resultados.
Condiciones	El usuario está previamente autenticado y tienen permisos para consultar. Existe la especialidad y esta cargada su programación.
Procedimiento	Seleccione en el menú CONSULTAR el enlace Porgramación. Se muestra el resultado de la consulta con la información de la porgamacion: Especialidad, Asignatura, Sección, Cupos, Profesor, Dias, Horario y Aulas.

Tabla 5.8. Caso de prueba consultar programación

Luego de consultar la prorgamacion se pueden eliminar registros de la misma:

Caso de prueba	Eliminar programación
Entrada	Se selecciona un registro de la programación perteneciente a la especialidad Nueva Materia.
Resultado	Se muestra un formulario donde se permite eliminar el registro seleccionado.
Condiciones	El usuario está previamente autenticado y tiene permisos para eliminar datos. Existe la especialidad y esta cargada su programación.
Procedimiento	Seleccione en el menú CONSULTAR el enlace Programación. Seleccione la especialidad de la prorgamacion donde desea eliminar registros. Para cada asignatura existe un registro con los siguientes datos: Codigo, Asignatura, Sección, Cupos, Profesor, Dias, Horario y Aula, luego se presiona el botón eliminar.

Tabla 5.9. Caso de prueba eliminar

Para la programación se pueden generar reportes en formato digital o impreso:

Caso de prueba	Generar reportes
Entrada	Se desea generar un reporte de la prorgramacion para la especialidad Nueva Carrera.
Resultado	Se genera el documento PDF y el usuario lo imprime o guarda en formato digital mediante el programa para visualizar PDF.
Condiciones	El usuario está previamente autenticado. Existe el la especialidad, y esta cargada su prorgramacion.
Procedimiento	Seleccione en el menú CONSULTAR el enlace Programación. Se selecciona la especialidad y luego se presiona el botón descargar el cual genera el documento en formato pdf.

Tabla 5.10. Caso de prueba generar reportes de la programación

5.4 Evaluación de la fase de construcción

La fase de construcción se ha realizado con éxito. Se requirió una iteración donde se explotaron los flujos de trabajo de implementación y pruebas. Durante la implementación se realizó la codificación efectiva de los modelos de gestión de contenido y los distintos componentes que conforman el software.

Cabe destacar que las herramientas que ofrece el lenguaje de modelado Web (WebML) fueron de mucha utilidad, ya que facilitan la codificación mediante estándares de programación web.

Todo software debe ser sometido a pruebas antes de su publicación o liberación. En esta fase, se le aplicaron pruebas al software, las cuales permitieron mejorar la calidad del software, ya que con la corrección de los errores presentados se garantiza la futura estabilidad operativa del mismo. En este caso fueron aplicadas las técnicas de pruebas de caja negra, y pruebas de integración. La ejecución de los procesos de implementación y pruebas han dado como resultado un producto estable y maduro como para ser entregado a la comunidad de usuarios para ser probado. Este proceso de prueba por parte de los usuarios debe realizarse en la próxima fase.

CAPITULO VI. FASE DE TRANSICIÓN

6.1 Introducción

Esta fase se centra en implementar la aplicación en su entorno de operación. El software se entrega a los usuarios para realizar pruebas beta, y la retroalimentación del usuario reporta tanto defectos como cambios necesarios. Además, el equipo de software crea la información de soporte necesaria (por ejemplo, manuales de usuario, guías de resolución de problemas, procedimientos de mantenimiento) para el lanzamiento. Al final de la fase de transición, el incremento de software se convierte en un lanzamiento de software utilizable. El principal objetivo que se ha planteado para esta fase es garantizar la satisfacción de todos los usuarios del sistema. También se proveerá el soporte básico necesario a través de un manual de usuario.

6.2 Lanzamiento de la versión beta

Para la preparación del lanzamiento de la versión beta de la aplicación, se ha planeado instalar el software en un área de trabajo, seleccionando un grupo de usuarios para que utilicen el software e instalar la estructura de hardware necesaria para que éste funcione. Entre los usuarios se requiere que estén presentes al menos, el jefe del Centro de Computación Académica y algunos Jefes de Departamento. A cada uno de ellos se le suministrará la documentación que indica lo siguiente:

- Como se realiza el acceso al software.
- Requisitos mínimos para el uso del sistema.
- Los niveles de permiso por usuario, es decir, las restricciones o limitaciones de acceso de cada usuario.
- Como utilizar el software.

6.3 Evaluación de la fase de transición

Durante la fase de transición se comprobó que el sistema implantado ha cumplido con los requerimientos y necesidades del Centro de Computación Académica, los Departamentos Académicos, las Direcciones de Escuela, la Coordinación académica y los Estudiantes de la Universidad de Oriente, Núcleo de Anzoátegui, así como también, ha satisfecho los objetivos planteados al inicio del proyecto.

CONCLUSIONES

1. Se llevó a cabo un análisis de la situación actual relacionada con el proceso de la programación académica, mediante las entrevistas a los profesores jefes de departamentos y visitas al centro de computación académica.

2. Se realizó un estudio detallado de todos los procedimientos relacionados con la programación académica, con lo que fue posible definir de forma concreta los requisitos necesarios para la elaboración del sistema.

3. Se concretó la arquitectura y organización del sistema mediante el uso del Proceso Unificado del Desarrollo de Software además de la herramienta WebML, que hizo posible el diseño de cada una de las páginas que conforman la aplicación, facilitando de esta manera el entendimiento y comprensión de las mismas.

4. Se realizó un estudio efectivo de la base de datos con la finalidad de que el sistema desarrollado funcionara sin problemas.

5. La integración de cada uno de los módulos programados mediante el lenguaje PHP se realizó sin ningún inconveniente lo que permitió obtener una aplicación robusta, confiable y segura.

6. Mediante la realización de pruebas de unidad e integración se encontraron y corrigieron fallos menores del sistema, que garantizan la obtención de un producto de buena calidad.

7. Se elaboró el manual de usuario que brinda soporte al acceso y uso del sistema.

8. Se instaló el sistema para las pruebas en el centro de computación académica obteniendo resultados satisfactorios.

9. Los objetivos planteados fueron cumplidos exitosamente durante el desarrollo de este proyecto.

RECOMENDACIONES

1. Luego de las encuestas realizadas, las cuales fueron fundamentales para obtener los requisitos de este sistema, se recomienda desarrollar un software que ayude en la planificación de la programación académica, ya que esta es una necesidad de todos los departamentos académicos de la Universidad de Oriente, Núcleo de Anzoátegui.

2. Es recomendable cambiar el servidor por uno más actualizao, puesto que este sistema será utilizado por muchos usuarios de la Universidad de manera simultánea en la víspera de las inscripciones de alumnos regulares.

3. Realizar un mantenimiento periódico a las tablas de la base de datos del sistema, así como también un backup (copia de seguridad) de las mismas.

4. Mejorar la base de datos existente a fin de que las consultas realizadas a la página se hagan en menor tiempo y de manera optimizada.

BIBLIOGRAFÍA

[1] Martha Elena García León y Víctor Julio Mújica Vargas, (2005). **"Desarrollo de un Software para la Automatización de las actividades Administrativas del Departamento de Computación y Sistemas de la Universidad de Oriente – Núcleo de Anzoátegui".**

[2] Dr. Sánchez David, (2007). **"Aplicación Web. Sistema Web del Estudiante – Universidad del Zulia".**

[3] Colmenares Giancarlo, (2004). **"Sistema de Inscripción vía Web".**

[4] Delegación de Teleinformática & Delegación de Información y Comunicación. **"Universidad de Oriente, Núcleo de Anzoátegui"**. http://www.anz.udo.edu.ve

[5] Senn, J. (1992). **"Análisis y Diseño de Sistemas de Información"**. Segunda Edición, Editorial Mc. Graw-Hill, México.

[6] Pressman, R. (2005). **"Ingeniería del Software. Un Enfoque Práctico".** Sexta Edición. Editorial McGraw-Hill. México.

[7] Miguel Angel Alvarez, **"La programación orientada a objetos". http://www.desarrolloweb.com/articulos/499.php**

[8] Jacobson, I., Booch, G. Y Rumbaugh, J. (1999) **"El Proceso Unificado de Desarrollo de Software"** Ediciones Addison-Wesley.

[9] Arlow J., Neustad I. (2006) **"Programación UML 2".** Primera Edición. Ediciones Anaya Multimedia. Madrid, España.

[10] Wikipedia. **"Intranet".** http://es.wikipedia.org/wiki/Intranet

[11] Wikipedia. **"Internet".** http://es.wikipedia.org/wiki/Acceso_a_internet

[12] José Guillermo Valle. (2005). **"Aplicación Cliente-Servidor".** http://www.monografias.com/trabajos24/arquitectura-cliente-servidor/arquitectura-cliente-servidor.shtml

[13] Wikipedia. **"Servidor Web Internet Information Server"**. http://es.wikipedia.org/wiki/IIS

[14] Wikipedia. **"Lenguaje de Etiquetas por Hipertexto"**. http://es.wikipedia.org/wiki/C%C3%B3digo_HTML

[15] Wikipedia. **"Técnica de desarrollo web AJAX"**. http://es.wikipedia.org/wiki/AJAX

[16] Wikipedia. **"Aplicaciones Web"**. http://es.wikipedia.org/wiki/Aplicaci%C3%B3n_web

[17] Página oficial WebML **"WebML".** http://www.webml.org/webml/page1.do

[18] Álvarez M., Álvarez R., Cuenca C. (2004) **"Programación en PHP – Manual Completo"**. http://www.desarrolloweb.com/manuales/12.

[19] Elmasri R., Navathe S. (2000). **"Sistemas de Bases de Datos. Conceptos Fundamentales".** Editorial Pearson Educación. Segunda Edición. México.

[20] Wikipedia. **"Software Libre".**
http://es.wikipedia.org/wiki/C%C3%B3digo_libre

[21] Luis C. Garelli y Luis E. Millán, (2007). **"Tesis de Luis C. Garelli y Luis E. Millán".** Tesis.

[22] Luis Miguel Alvins Ríos. (2007). **"Tesis de Luis Miguel Alvins Ríos".** Tesis.

[23] Rosnuel José López González. (2008). **"Tesis de Rosnuel José López González".** Tesis.

[24] Marval H., Juan Luís y García S., Mariedys Sabrina. (2008). **"Tesis de Marval H., Juan Luís y García S., Mariedys Sabrina".** Tesis.

[25] Microsoft. (2001) **"Microsoft Diccionario de Informática e Internet"**. Editorial McGraw-Hill. Primera Edición. Madrid.

[26] Sabino, C. (1994). **"Como hacer una tesis".** Tercera Edición. Editorial Panapo. Caracas.

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

TÍTULO	DESARROLLO DE UN SISTEMA AUTOMATIZADO BAJO ENTORNO WEB PARA EL CONTROL DE LA PROGRAMACIÓN ACADÉMICA EN LA UNIVERSIDAD DE ORIENTE NÚCLEO DE ANZOÁTEGUI
SUBTÍTULO	

AUTOR (ES):

APELLIDOS Y NOMBRES	CÓDIGO CULAC / E MAIL
AFONSO R., MARIELA A.	CVLAC: 15.679.714 E MAIL: mariela0122@msn.com
SEGNINI R., JESÚS E.	CVLAC: 15.897.285 E MAIL: segnini75@hotmail.com
	CVLAC: E MAIL:
	CVLAC: E MAIL:

PALÁBRAS O FRASES CLAVES:

DESARROLLO SOFTWARE PROGRAMACIÓN ACADÉMICA WEBML UDO ANZOÁTEGUI

_____

_____

_____

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

ÀREA	SUBÀREA
INGENIERÍA Y CIENCIAS APLICADAS	INGENIERÍA EN COMPUTACIÓN

RESUMEN (ABSTRACT):

El siguiente proyecto de investigación se basa en el desarrollo de un sistema automatizado para el control de la Programación Académica (SACPA) para la Universidad de Oriente, Núcleo de Anzoátegui. El software se encarga de proporcionar una interfaz agradable y de fácil manejo en entorno web a los diferentes departamentos académicos de la Institución y a los directores de escuela, para ingresar y administrar la Programación Académica que elaboran durante cada periodo académico. Ademas de permitir consultas por parte de los estudiantes y los profesores. Esta información relacionada con la programación académica es guardada directamente en la base de datos ubicada en el Centro de Computación Académica que es el administrador del sistema, lo que mejora la comunicación entre este centro y los departamentos. Este sistema garantiza información confiable ya que uno de sus principales objetivos es validar los datos ingresados, ademas permite a los usuarios consultar un mapa de aulas con el fin de mejorar la planificación de su programación. El software se elaboró utilizando el lenguaje de programación PHP versión 5.2.0, la técnica de programación AJAX, el servidor Apache versión 5.0 y el motor de base de datos MySQL versión 5.0. Este proyecto se construyó siguiendo el Proceso Unificado de Desarrollo de Software, la herramienta WebML y el Lenguaje de Modelado UML.

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

CONTRIBUIDORES:

APELLIDOS Y NOMBRES	ROL / CÓDIGO CVLAC / E_MAIL				
Ordaz O., Lenys V.	ROL	CA	AS	TU X	JU
	CVLAC:	11.196.284			
	E_MAIL	lenyso@hotmail.com			
	E_MAIL				
Cortínez N., Claudio A.	ROL	CA	AS	TU	JU X
	CVLAC:	12.155.334			
	E_MAIL	cl_cortinez@cantv.net			
	E_MAIL				
Rodríguez M., Rhonald E.	ROL	CA	AS	TU	JU X
	CVLAC:	14.077.185			
	E_MAIL	rhoen@hotmail.com			
	E_MAI				

FECHA DE DISCUSIÓN Y APROBACIÓN:

2009	06	09
AÑO	MES	DÍA

LENGUAJE. SPA

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

ARCHIVO (S):

NOMBRE DE ARCHIVO	TIPO MIME
TESIS.prog.academica.doc	application/msword

CARACTERES EN LOS NOMBRES DE LOS ARCHIVOS: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z. a b c d e f g h i j k l m n o p q r s t u v w x y z. 0 1 2 3 4 5 6 7 8 9.

ALCANCE

ESPACIAL: Computación academica, UDO Anzoategui (OPCIONAL)

TEMPORAL: (OPCIONAL)

TÍTULO O GRADO ASOCIADO CON EL TRABAJO:

Ingeniero en Computación

NIVEL ASOCIADO CON EL TRABAJO:

Pregrado

ÁREA DE ESTUDIO:

Departamento de Computación y Sistemas

INSTITUCIÓN:

Universidad de Oriente, Núcleo de Anzoátegui

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

DERECHOS

Los trabajos de grado son de exclusiva propiedad de la Universidad y sólo podrán ser utilizados para otros fines con el conocimiento del Consejo de Núcleo respectivo, quién lo participará al Consejo Universitario

Afonso R., Mariela A.	Segnini R., Jesús E.
AUTOR	AUTOR

Ordaz, Lenys	Rodríguez, Rhonald	Cortínez,Claudio
TUTOR	JURADO	JURADO

POR LA SUBCOMISION DE TESIS