



**UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO**

---

---

**FACULTAD DE INGENIERÍA**

**“SISTEMA DE INSCRIPCIÓN Y CONTROL DE CURSOS PARA  
LA UNIDAD DE SERVICIO-S DE CÓMPUTO ACADÉMICO  
(SICC)”**

**T E S I S**

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMPUTACIÓN

P R E S E N T A :

**JOSÉ ARMANDO FLORES DÍAZ**

DIRECTOR DE TESIS:  
ING. JULIO CÉSAR JUÁREZ SOSA

CODIRECTOR DE TESIS:  
ING. MARÍA DEL ROSARIO BARRAGÁN PAZ



Ciudad Universitaria

México, D.F. 2004

---

## DEDICATORIAS

A Dios por darme la oportunidad de vivir y regalarme una familia maravillosa.

A mis padres por todo su amor y apoyo incondicional. Gracias papá y mamá por darme una carrera, aunque hemos pasado momentos difíciles siempre han estado conmigo brindándome su cariño y comprensión, por lo tanto este trabajo es para ustedes.

A mis hermanos, que este trabajo sea el primero de muchos en la familia.

A Laura G. Segoviano López por haber compartido conmigo los mejores momentos de mi vida. No hay un día en el que no me acuerde de ti. Aunque ya no estas conmigo, siempre guardaré un bello recuerdo tuyo.

---

# ÍNDICE

DEDICATORIAS	i
AGRADECIMIENTOS	ii
ÍNDICE	iii
OBJETIVOS	vi
INTRODUCCIÓN	vii
UNICA	viii
<b>1. ANTECEDENTES</b>	<b>1</b>
1.1. La filosofía Orientada a Objetos	1
1.1.1. Abstracción	2
1.1.2. Herencia	2
1.1.3. Polimorfismo	3
1.1.4. Encapsulamiento	3
1.1.5. Mensajes	4
1.2. Evolución de la Tecnología Orientada a Objetos	5
1.3. Especificaciones del Sistema de Inscripción y Control de Cursos para la Unidad de Servicios de Cómputo Académico (SICC)	10
1.3.1. Antecedentes	10
1.3.2. Control de la información de los grupos	12
1.3.2.1. Información de las inscripciones	12
1.3.2.2. Inscripción a grupos	15
1.3.3. Reservación de un lugar en un grupo vía Internet	16
<b>2. METODOLOGÍA ORIENTADA A OBJETOS Y EL LENGUAJE DE MODELADO UNIFICADO</b>	<b>18</b>
2.1. Metodologías Orientadas a Objetos	18
2.1.1. El enfoque SA/SD	18
2.1.2. JSD Jackson System Development	19
2.1.3. El método de Booch	21
2.1.4. Metodología de Shlaer y Mellor	21
2.1.5. El método de Coad y Yourdon	22
2.1.6. El método de Jacobson	23
2.1.7. Método de Wirfs-Brock	25
2.1.8. OMT Object Modeling Technique (Técnica de Modelado de Objetos)	26
2.2. Metodología OMT	28
2.2.1. Diferencias entre OMT y la metodología funcional	31
2.3. Procesos de la metodología OMT	34
2.3.1. Análisis	34
2.3.1.1. Definición del problema	35
2.3.1.2. Modelado de Objetos	35
2.3.1.2.1. Notación del Modelo de Objetos	36
2.3.1.3. Elaboración del Modelo de Objetos	46

---

2.3.1.4. Modelado Dinámico	48
2.3.1.5. Modelado Funcional	53
2.3.1.5.1. Notación del Modelado Funcional	53
2.3.2. Diseño del Sistema	58
2.3.3. Diseño de Objetos	61
2.3.4. Implementación	62
2.4. Lenguaje de Modelado Unificado (UML)	63
2.4.1. La necesidad de UML	63
2.4.2. Origen de UML	64
2.5. Diagramas de UML	66
2.5.1. Referencia rápida de UML	79
2.6. Relación del Lenguaje de Modelado Unificado y la Metodología OMT	80
<b>3. IMPLEMENTACIÓN DEL SISTEMA DE CONTROL DE CURSOS E INSCRIPCIONES</b>	82
3.1. Herramientas CASE a utilizar	82
3.1.1. Herramientas CASE: Definición	82
3.1.2. Clasificación de las herramientas CASE	83
3.2. Herramientas CASE utilizadas en el SICC	87
3.2.1. Rational Rose	87
3.2.2. Erwin 4.1	88
3.3. Planeación y Análisis de Requerimientos del SICC	89
3.3.1. Descripción de la tecnología utilizada	91
3.4. Modelo Entidad – Relación	97
3.5. Conversión del Modelo Entidad – Relación a un esquema de Base de Datos para el SICC	99
3.5.1. Estructura de la Base de Datos	100
3.6. Diseño e Implementación del SICC mediante OMT y UML	107
3.6.1. Modelo de Objetos	108
3.6.2. Modelo Dinámico	110
3.6.3. Modelado Funcional	122
<b>4. GENERACIÓN DE CLASES Y OBJETOS CON VISUAL BASIC</b>	130
4.1. Creación de Clases y Objetos	130
4.1.1. Procedimientos Property	132
4.1.2. Colecciones	134
4.1.3. Implementación de Clases en el SICC	136
4.1.4. Implementación de Objetos en el SICC	148
4.2. Creación de Procedimientos y Funciones	151
4.2.1. Diferencias entre Procedimientos y Funciones	152
4.3. Interfaz del sistema de control de cursos e inscripciones	159
4.3.1. Definición de Interfaz	159
4.3.2. Interfaz del SICC	159
4.3.2.1. Interfaz Modulo 3 (Control e Inscripción a cursos)	160
4.3.2.1.1. Interfaz de Administración	160
4.3.2.1.2. Interfaz de Inscripción	170

---

---

4.3.2.2. Interfaz Modulo 4 (Apartado de cursos vía web)	176
4.4. Generación de Reportes y Constancias	184
<b>5. EVALUACIÓN Y RESULTADOS DEL SICC</b>	194
5.1. Pruebas del usuario final	194
5.1.1. Objetivo de la fase de pruebas	194
5.1.2. Metodología de Pruebas	194
5.2. Plan de pruebas para el SICC	197
5.2.1. Pruebas de caja negra	197
5.2.2. Pruebas para la aplicación web	199
5.2.3 Pruebas de caja blanca	200
5.3. Aplicación y Resultados del Plan de Pruebas del SICC	205
5.3.1. Pruebas de Búsqueda	213
5.3.2. Pruebas para la aplicación web	217
5.4. Mantenimiento del sistema	225
5.4.1. Causas del mantenimiento	225
5.4.2. Dificultades y costos del mantenimiento	226
5.4.3. Tipos de mantenimiento	226
5.5. Mantenimiento del SICC	228
5.5.1. Mantenimiento adaptativo	228
5.5.2. Mantenimiento correctivo	229
5.5.3. Mantenimiento preventivo	230
5.5.4. Mantenimiento perfectivo	230
<b>6. CONCLUSIONES</b>	231
6.1. Contribuciones de la tesis	231
6.2. Perspectivas a Futuro	234
<b>APÉNDICE A</b>	235
Manual de Usuario	235
<b>BIBLIOGRAFÍA</b>	254

---

# OBJETIVOS

## Objetivo General:

Desarrollar un sistema de cómputo para llevar a cabo las inscripciones a los cursos que imparte UNICA, así como la generación de constancias y reportes de forma automática.

## Objetivos específicos:

- Manejar toda la información referente a los cursos que imparte UNICA
- Generar las listas de grupos mediante la asignación del curso, fecha de inicio y terminación, horario, lugar y costo.
- Generar recibos y comprobantes de inscripción
- Controlar las inscripciones de los alumnos a los cursos.
- Proveer las herramientas necesarias para generar las constancias correspondientes.

---

# INTRODUCCIÓN

Una de las funciones de la Unidad de Servicios de Cómputo Académico en la Facultad de Ingeniería, es la de impartir cursos de cómputo dirigidos a la comunidad universitaria y público en general. La primer actividad es la de desarrollar una propuesta de calendario de cursos. Para cumplir con esta función se requiere de una planeación específica y sistemática que conlleva a la creación de grupos mediante la asignación del curso, una fecha, un horario, un lugar y un costo. Finalizada esta etapa se genera un listado de los grupos que se abrirán en el período establecido.

La inscripción a un grupo se lleva a cabo de la siguiente forma: cada individuo debe presentarse en las instalaciones de UNICA y preguntar por el curso que le interesa, si existe cupo puede realizar su inscripción proporcionando sus datos personales. Después realiza el pago correspondiente a dicho curso, presenta el comprobante y con ello asegura definitivamente su lugar en el curso.

Una vez finalizados y evaluados los asistentes a los cursos, se otorgan constancias a las personas que aprobaron un determinado curso.

Actualmente, este trabajo es realizado de forma semiautomática, por lo que surge la necesidad de desarrollar un sistema de cómputo que ayude a la planeación de los cursos, agilizando así, el proceso de inscripción.

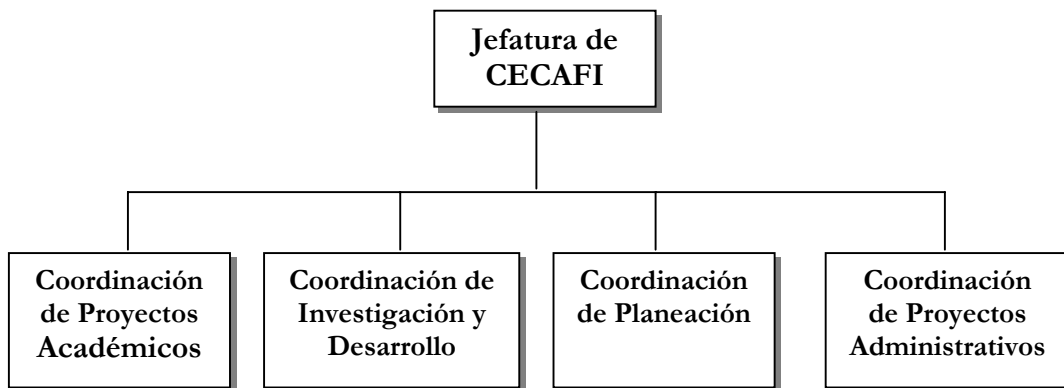
---

# UNICA

La Unidad de Servicios de Cómputo Académico de la Facultad de Ingeniería (UNICA) es resultado de la división del antiguo Centro de Cálculo (CECAFI), que se realizó en febrero de 1994.

El CECAFI se fundó en 1972 por acuerdo del entonces Director Juan Casillas García de León; entre los principales servicios que brindaba estaban: La elaboración de sistemas, programas de apoyo académico-administrativo, asesorías, cursos para alumnos, personal académico y apoyo a las asignaturas ofreciendo servicios en temas de cómputo (préstamo de equipo y asesorías), elaboración de material didáctico de computación, realización de sistemas y bases de datos de los alumnos de la Facultad, así también se investigaba acerca de las nuevas aplicaciones de la computación en las distintas áreas de estudio de la Facultad de Ingeniería.

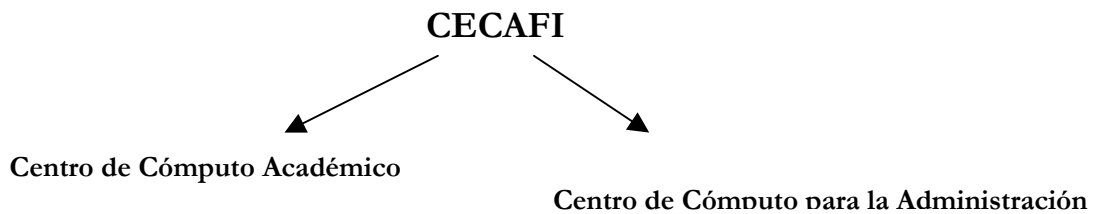
Estructura de CECAFI hasta 1992





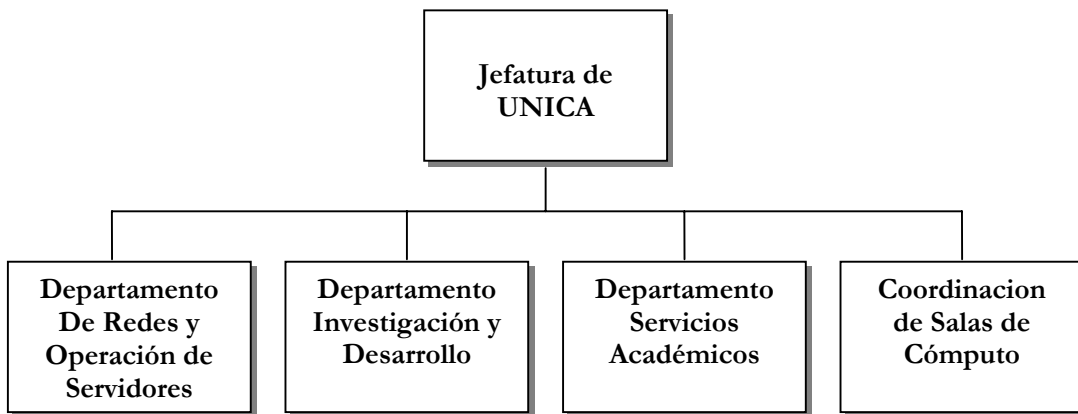
---

En 1993 el CECAFI se reestructuró dividiéndose en dos Centros de Cómputo:



Hasta entonces el Centro de Cómputo Académico tuvo las funciones de formar recursos humanos y proporcionar servicios en cómputo; mientras tanto, las funciones del Centro de Cómputo para la Administración fueron mantener la información académica, elaborando sistemas de información y administrando equipos de cómputo.

En 1994 el Centro de Cómputo Académico cambió su nombre a Unidad de Servicios de Cómputo Académico (UNICA). Actualmente esta conformada por los siguientes departamentos:



El personal que integra UNICA es: personal académico, administrativos de confianza, administrativos de base, becarios y servicio social. Cabe hacer notar que la mayor parte del personal la constituyen los becarios.

---

## Objetivos y metas de UNICA

UNICA tiene objetivos y metas definidos desde el año de 1996.

### Objetivos:

- Proporcionar recursos de cómputo de calidad a la comunidad de la Facultad de Ingeniería.
- Apoyar a Divisiones y Secretarías en cualquier actividad relativa a cómputo.
- Apoyar y colaborar con la Secretaría General en las actividades que involucran a toda la Facultad de Ingeniería.
- Formar recursos humanos de calidad no sólo en el área de cómputo sino también en su vida profesional.

### Meta:

Mantener el liderazgo y estar a la vanguardia en cómputo tanto dentro de la Facultad de Ingeniería como en el entorno universitario.

### Misión:

Ofrecer a la comunidad de la Facultad de Ingeniería un servicio de calidad suministrando recursos de cómputo comerciales y especializados que el avance de la educación demande.

---

## Servicios que ofrece.

La Facultad de Ingeniería (FI) cuenta con ocho Divisiones que ofrecen recursos de cómputo en temas especializados. UNICA no particulariza en ninguna rama de la ingeniería, sino que ofrece recursos desde un punto de vista general a toda la comunidad de FI. Actualmente se ofrecen los siguientes servicios:

**Servicios de cómputo.** Para brindar este servicio a la comunidad de la FI, UNICA cuenta con tres salas de cómputo. La primera se encuentra en el edificio Principal y las siguientes dos en el anexo de la Facultad.

Sala 1.- Ubicada en el edificio principal (edificio del CECAFI) de la FI. Cuenta con dos salas: La sala de estaciones de trabajo y la sala de computadoras personales.

Sala 2.- Ubicada en la División de Ciencias Básicas de la Facultad de Ingeniería abajo del auditorio Sotero Prieto. Esta sala cuenta con 4 salas A, B, C y D.

Sala 3.- Ubicada en la Torre de Ciencias Básicas de la Facultad de Ingeniería en la planta baja. Esta sala cuenta con cuatro salas E, F, G y H.

Cursos de computación.

Intersemestrales.- Cada periodo intersemestral la Facultad de Ingeniería ofrece diversos cursos de computación, los cuales son impartidos por las divisiones de la Facultad, éstas envían sus propuestas de cursos a UNICA donde se realiza la planeación, inscripción, propaganda, difusión y se generan las constancias de aprovechamiento.

Semestrales.- Funciona igual que los intersemestrales, solo que son dirigidos principalmente a los alumnos de la facultad

---

Sábados y domingos.- UNICA también coordina los cursos que se ofrecen en sábados y domingos durante el período semestral, es decir, durante el periodo de clases. Estos cursos están abiertos para el público en general.

Personal Académico.- También se apoya a la Dirección General de Apoyo a Personal Académico DGAPA impartiendo diversos cursos de cómputo durante el semestre, de lunes a viernes.

Especiales.- Así también se preparan cursos especiales para las Divisiones y Secretarías de la Facultad cuando éstas así lo soliciten.

Externos.- Se preparan cursos especiales para empresas o personas externas a la UNAM.

**Formación de recursos humanos.** Al final de semestre, UNICA convoca a los alumnos de la Facultad de Ingeniería al concurso de ingreso al plan de formación de becarios, en esta etapa se selecciona y prepara a los candidatos que cumplan con el perfil requerido y se les da capacitación en cómputo a fin de contar con los mejores candidatos para formar parte de su personal. Al final de su preparación, los aspirantes a becarios tienen que presentar un examen final y de acuerdo a su desempeño son seleccionados por el comité del programa “Plan de Formación de Becarios”. Siendo ya becarios, UNICA los asigna a un Departamentos de acuerdo con sus intereses y los de la Unidad, donde posteriormente desarrollarán proyectos especiales, sistemas, y docencia entre otros.

**Control del equipo de cómputo.** UNICA tiene la responsabilidad de concentrar los requerimientos para la compra de equipo de cómputo de las Divisiones y Secretarías de la FI; así mismo realiza investigación de características técnicas y contacta con proveedores.

---

**Coordinación del mantenimiento preventivo del equipo de cómputo.** La Facultad de Ingeniería tiene contrato con una empresa de mantenimiento preventivo de equipo de cómputo, UNICA en conjunto con Secretaría Administrativa realiza un plan de mantenimiento para el equipo de la Facultad y verifica que se cumpla.

**Censo de equipo de cómputo.** Cada año, el Consejo Asesor de Cómputo de la UNAM solicita una relación de equipo de cómputo a cada Facultad, UNICA es responsable de hacer el concentrado del inventario de las Divisiones y Secretarías y entregar el censo correspondiente a la Facultad de Ingeniería.

**Coordinación del inventario de la Facultad de Ingeniería.** UNICA realiza un plan de acción para concentrar la información del inventario de equipo de cómputo de la Facultad de Ingeniería, coordina a todas las Divisiones y Secretarías de la Facultad para realizar un inventario de sus áreas.

**Elaboración de material didáctico.** En la Unidad se elaboran notas y material didáctico sobre temas de computación de acuerdo a los cursos impartidos.

**Estudio y aplicación de software.** Investigación y desarrollo del software comercial y especializado que la actividad de la Facultad requiere y prestación de servicios de asesoría sobre estos tópicos; desarrollo de sistemas necesarios en UNICA y que la comunidad de la Facultad demanda.

---

## **Necesidad del Sistema de Inscripción y Control de Cursos(SICC)**

Como acabamos de ver, la administración y control de los servicios que ofrece la Unidad requiere de herramientas que apoyen y optimicen dichos procesos.

En este contexto, es donde surge la idea de desarrollar un sistema de cómputo que ayude a la planeación de los cursos, agilizando así, el proceso de inscripción.

Dicho sistema ayudará a tener un mejor manejo, mantenimiento y administración de los procesos de UNICA, lo que se traducirá en una mejora significativa y crecimiento global de los servicios que ofrece dicha dependencia.

### 1.1 La filosofía Orientada a Objetos

A grandes rasgos, el término “Orientado a Objetos” significa que el software es organizado como una colección de elementos del mundo real, pero conceptualizados de tal forma que se puedan implementar como elementos de software. Este enfoque permite modelar y diseñar prácticamente toda la aplicación en términos mucho más sencillos y más cercanos a la realidad.

Los objetos, concretos o virtuales, están a nuestro alrededor conformando nuestro mundo. Cuando hablamos de objetos nos referimos a instancias de una clase, es decir, elementos concretos provenientes de un molde o plantilla a la cual llamamos clase.

Usted y yo, por ejemplo, somos instancias de la clase Persona. Tenemos características en común: nombre, altura, peso, edad, etc. Estas características son atributos de la clase a la que pertenecemos y por lo tanto son inherentes a nosotros. Además de atributos, una clase posee acciones ó métodos que no son otra cosa que determinadas actividades que los objetos pertenecientes a dicha clase pueden realizar. Algunos métodos de la clase Persona podrían ser: comer, dormir, leer, escribir, hablar, trabajar y más. Una clase entonces funciona como molde o plantilla que nos permite crear objetos a partir de ella. Algo similar a un molde de gelatinas que produce muchas gelatinas.

La orientación a objetos se refiere a algo más que atributos y acciones. Incluye conceptos tales como: **abstracción, herencia, polimorfismo y encapsulamiento**.

### 1.1.1 Abstracción

A grandes rasgos la abstracción se refiere a quitar las propiedades y acciones de un objeto para dejar sólo aquellas que sean necesarias. Esto permite crear una clase con los atributos y acciones generales y suficientes para poder crear instancias de dicha clase que respondan a la mayoría de las expectativas.

Pensemos en una clase lavadora v.gr. Dicha clase contendría atributos tales como marca, modelo, numero de serie, capacidad; y métodos tales como agregar ropa ó secar ropa. Evidentemente, habrá objetos lavadora que requieran de algún otro método o atributo: velocidad del motor, cronómetro, v.gro. Sin embargo, estaríamos hablando de situaciones específicas y no de circunstancias generales. La finalidad de la abstracción es la de encontrar aquellas características y métodos que permitan la creación de objetos de forma general, no haciendo énfasis en las situaciones particulares.

### 1.1.2 Herencia

Como se mencionó anteriormente, un objeto es una instancia de una clase y como tal, tiene todas las características de la clase de la cual procede. A esto se le conoce como herencia.

Un objeto no sólo hereda de una clase, sino que una clase también puede heredar de otra. Las lavadoras, refrigeradores, hornos de microondas, lavaplatos, tostadores, etc, forman parte de una clase más genérica llamada Electrodoméstico, y de la cuál heredan características como interruptor y cable eléctrico, y las operaciones de encendido y apagado. Otra forma de entenderlo es que la lavadora, refrigerador, tostador, etc son subclases de la clase Electrodoméstico la cual suele ser llamada entonces superclase.



### 1.1.3 Polimorfismo

En ocasiones una operación tiene el mismo nombre en diferentes clases. Por ejemplo, se puede abrir una puerta, una ventana, un regalo o una cuenta bancaria. En todos los casos la operación es básicamente la misma. En la orientación a objetos cada clase sabe como hacer tal operación. Esto es el polimorfismo. Este concepto es importante tanto para los modeladores como para los desarrolladores del software puesto que además de facilitar la implementación de los sistemas, permite mantener un lenguaje directo con el cliente con su propia terminología. En ocasiones, las palabras y terminología del cliente nos conducen a palabras de acción tales como “abrir” que pueden tener más de un significado.

### 1.1.4 Encapsulamiento

El encapsulamiento se refiere a que los objetos ocultan lo que hacen; es decir; encapsulan la funcionalidad interna de sus operaciones. Tomemos como ejemplo una televisión. Por lo general, la gente que ve televisión no se preocupa de la complejidad electrónica que hay detrás de la pantalla ni de todas las operaciones que deben ocurrir para mostrar una imagen en la pantalla. La televisión realiza su función sin mostrarnos todos los procesos que debe hacer para ello, esto es, encapsula su funcionalidad.

En el desarrollo de software, el encapsulamiento permite reducir el potencial de errores que pudieran ocurrir. En un sistema que consta de objetos, éstos dependen unos de otros en diversas formas. Si uno de ellos falla, el ocultar sus operaciones de otros objetos significará que tal vez no será necesario modificar los demás objetos.

En el mundo real, también se puede ver la importancia del Encapsulamiento en los objetos. Por ejemplo, el monitor de una computadora en cierto sentido oculta su funcionalidad del CPU, de tal manera que si falla algo en el monitor se puede reparar o reemplazar pero no es necesario reparar o reemplazar el CPU al mismo tiempo.

### 1.1.5 Mensajes

En un sistema orientado a objetos éstos trabajan en conjunto. Esto se logra mediante el envío de mensajes. Un objeto envía un mensaje a otro para realizar una operación, y el objeto receptor ejecuta la operación. Una televisión y su control remoto pueden ser un ejemplo. Al oprimir el botón de encendido del control remoto, éste envía un mensaje al televisor para que se encienda. El televisor recibe el mensaje, lo identifica como una instrucción para encenderse y así lo hace. El control remoto también puede enviar otros mensajes como ajustar el volumen, cambiar de canal, activar subtítulos, etc..

## 1.2 Evolución de la Tecnología Orientada a Objetos

Durante muchos años el término orientado a objetos se usó para señalar un enfoque de desarrollo de software que usaba uno de los lenguajes orientados a objetos (Ada 95, C++, Eiffel, Smalltalk).<sup>1</sup> Hoy en día el paradigma orientado a objetos encierra una completa visión de la ingeniería de software.

En los primeros días de la computación, los lenguajes ensambladores facilitaban a los programadores la utilización de las instrucciones máquina para manipular los datos en un sistema de cómputo. El nivel de abstracción que se aplicaba al dominio de la solución era muy bajo.

En general, el diseño de software se enfocaba sobre la representación del detalle de los procedimientos necesarios para realizar el sistema requerido usando el lenguaje de programación elegido. Los conceptos de diseño, tales como refinamientos sucesivos de una función, modularidad y, posteriormente, programación estructurada, fueron introducidos entonces.

Durante los años 1970, se introdujeron conceptos tales como abstracción y ocultación de la información, y emergieron métodos de diseño conducido por los datos, pero los que desarrollaban software aún se encontraban sobre el proceso y su representación. Al mismo tiempo, los lenguajes de alto nivel modernos como Pascal introdujeron una variedad mucho más rica de tipos y estructuras de datos.

Aunque los lenguajes de alto nivel convencionales (lenguajes a partir de FORTRAN y ALGOL) evolucionaron durante los años 1960 y 1970, los investigadores estaban trabajando mucho sobre una nueva clase de lenguaje de simulación y construcción de prototipos, tales como SIMULA y Smalltalk. En estos lenguajes, la abstracción de datos tenía una gran importancia, y los problemas del mundo real se representaban mediante un conjunto de

---

<sup>1</sup> Primeros lenguajes considerados como orientados a objetos a finales de los años 80 y principios de los 90

objetos de datos, a los cuales se les añadía el correspondiente conjunto de operaciones. El uso de estos lenguajes era radicalmente diferente del uso de lenguajes más convencionales.

En 1967 la compañía Ericsson poseía un método de desarrollo muy eficaz. Modelaba el sistema entero como un conjunto de bloques interconectados actualmente, UML *Unified Modeling Language* (Lenguaje de Modelado Unificado) los denomina “subsistemas” y se implementan mediante “componentes”). Después, ensamblaba los bloques de más bajo nivel en subsistemas de más alto nivel, para hacer el sistema más manejable.

El primer producto del trabajo de las actividades de diseño era una *descripción de arquitectura* software. Se basaba en la comprensión de los requisitos más críticos, y describía brevemente cada bloque y su agrupamiento en subsistemas. Un conjunto de diagramas describían a los bloques y a sus interconexiones. Sobre las interconexiones se comunicaban señales, es decir, un tipo de mensaje. Todos los mensajes quedaban descritos, uno por uno, en una biblioteca de mensajes. La descripción de la arquitectura software y la biblioteca de mensajes eran los documentos fundamentales que guiaban el trabajo de desarrollo, pero también se utilizaban para presentar el sistema a los clientes.

En esencia el método que utilizaban era el que hoy conocemos como **Desarrollo basado en componentes**. Ivar Jacobson fue el creador de éste método de desarrollo. El dirigió su evolución hacia un proceso de desarrollo de software durante muchos años en el período anterior a *Objectory*.<sup>2</sup>

En 1976 se presentó un avance significativo, la publicación por parte del CCITT (*Consultative Committee on International Telegraphy and Telephony*), el organismo internacional para la estandarización en el área de las telecomunicaciones, del Lenguaje de Especificación y Descripción (*Specification and Description Language*, SDL) para el comportamiento funcional de los sistemas de telecomunicación. Este estándar, influenciado fuertemente por el método de Ericsson, especificaba un sistema como un conjunto de bloques interconectados que se comunicaban unos con otros únicamente a través de mensajes (llamados “señales” en el

---

<sup>2</sup> “Objectory” es una abreviatura de “Object Factory”, fábrica de objetos. Este proceso de desarrollo de software fue creado en 1987 por Ivar Jacobson y sus colaboradores en Estocolmo, Suiza.

estándar). Cada bloque poseía un conjunto de “procesos”, que era el término SDL para designar las clases activas. Un proceso poseía instancias de manera muy parecida a cómo lo hacen las clases en términos de orientación a objetos. Las instancias de los procesos interactuaban mediante mensajes. SDL proponía diagramas que eran especializaciones de lo que hoy UML llama diagramas de clases, diagramas de actividad, diagramas de colaboración y diagramas de secuencias.

SDL era un estándar de modelado de objetos especializado. Todavía es utilizado por más de 10 000 desarrolladores y cuenta con el soporte de varios fabricantes de herramientas. Fue desarrollado inicialmente hace poco más de veinte años, y estaba muy por delante de su tiempo. Sin embargo, se desarrolló en un momento en el cual el modelado de objetos no había madurado.

En 1987 Ivar Jacobson, quién en esos momentos trabajaba para Ericsson, renunció y fundó Objectory AB en Estocolmo.

El desarrollo del proceso Objectory se tradujo en una serie de versiones, desde Objectory 1.0 en 1998 a la primera versión interactiva, Objectory 3.8, en 1995.

Rational Software Corporation quién era una de las principales compañías dedicadas al trabajo con los métodos de desarrollo de software, compró Objectory AB a finales de 1995. Previamente a esto, Rational había creado algunas prácticas de desarrollo de software, la mayoría de ellas complementarias a las contenidas en Objectory.

En el momento de la fusión, Objectory 3.8 había demostrado que se puede crear y modelar un proceso de desarrollo software como si fuese un producto. Había diseñado la arquitectura original de un proceso de desarrollo software. Había identificado un conjunto de modelos que documentaban el resultado del proyecto. Estaba correctamente desarrollado en áreas como el modelado de casos de uso, análisis y diseño, aunque en otras áreas, gestión de requisitos aparte de los casos de uso, implementación y pruebas, no estaba tan bien desarrollado. Además, decía poco sobre gestión del proyecto, gestión de la configuración,

distribución, y sobre la preparación del entorno de desarrollo (obtención del proceso y las herramientas).

Por ello se le añadieron la experiencia y prácticas de Rational para formar el Proceso Objectory de Rational 4.1 (*Rational Objectory Process*, ROP)..

En estos momentos, UML estaba en fase de desarrollo y se incorporó como el lenguaje de modelado del Proceso Objectory de Rational.

Durante este período, Grady Booch había creado el método Booch, y James Rumbaugh era el desarrollador principal de OMT *Object Modelling Technique* (Técnica de Modelado de Objetos) creado en el Centro de Investigación y Desarrollo de General Electric. Cuando este último se incorporó a Rational en Octubre de 1994, los dos comenzaron el trabajo de unificar sus métodos, en coordinación con muchos de los clientes de Rational. Publicaron la versión 0.8 del Método Unificado en octubre de 1995, de forma simultánea con la incorporación de Ivar Jacobson a Rational.

Los tres, trabajando juntos, publicaron la versión 0.9 del Lenguaje Unificado de Modelado. El esfuerzo se amplió para incluir a otros metodologistas, así como a diversas empresas, que incluían a IBM, HP y Microsoft, cada una de las cuales contribuyó al estándar en evolución. En noviembre de 1997, después de pasar por el proceso de estandarización, el OMG (*Object Management Group*) publicó como estándar la versión 1.1 del Lenguaje Unificado de Modelado.

Rational compró o se fusionó a otras empresas fabricantes de herramientas. Cada una de ellas aportó a la mezcla su experiencia en áreas del proceso que ampliaron más aún el proceso Objectory de Rational.

Finalmente en junio de 1998, Rational publicó una nueva versión del producto, el Proceso Unificado de Rational mejor conocido como RUP.

En el siguiente cuadro podemos ver un resumen de las metodologías más importantes y el año de su aparición:

Año	Metodología
1968	Conceptos sobre la programación estructurada de DIJKSTRA
1974	Técnicas de programación estructurada de WARNIER y JACKSON
1975	Primeros conceptos sobre diseño estructurado de MYERS y YOURDON
1977	Primeros conceptos sobre análisis estructurado GANE y SARSON
1978	Análisis estructurado: DEMARCO y WEINBERG
1985	Análisis y Diseño estructurado para sistemas de tiempo real de WARD y MELLOR
1987	Análisis y Diseño estructurado para sistemas de tiempo real de HATLEY y PIRHBAY
1991	OMT
1993	Objectory/OOSE (Object Oriented Software Engineering) Jacobson
1994	BOOCH
1999	PROCESO UNIFICADO (JACOBSON)
1999	RATIONAL UNIFIED PROCESS (RUP)

Tabla 1.1 Metodologías de desarrollo de software más importantes y sus fechas de aparición.

Hoy en día, el desarrollo orientado a objetos se usa en aplicaciones de diseño de software que van desde animaciones de gráficos por computadoras a las telecomunicaciones.

Se utilizan técnicas orientadas a objetos para desarrollar compiladores, interfaces de usuario, bases de datos, sistemas CAD (Diseño Asistido por Computadora), simulaciones, sistemas de control y otras aplicaciones. Se han utilizado modelos orientados a objetos para documentar programas existentes que están mal estructurados y resultan difíciles de entender.

## **1.3 Especificaciones del Sistema de Inscripción y Control de Cursos para la Unidad de Servicios de Cómputo Académico (SICC).**

### **1.3.1 Antecedentes**

Para llevar un control de las diversas actividades que realiza, la Unidad de Servicios de Cómputo Académico de la Facultad de Ingeniería contaba con un sistema de cómputo desarrollado en Visual Basic y que interactuaba con una base de datos en Access.

A pesar de que durante algún tiempo el sistema funcionó correctamente, con el paso del tiempo empezó a presentar limitantes. Por ejemplo, la interfaz del sistema anterior era en cierta forma rudimentaria y poco amigable con el usuario. Tenía los controles estrictamente necesarios y por interactuar con una base de datos hecha en access, presentaba un nivel de seguridad mínimo. Además, presenta fallas de visualización en la interfaz donde se muestra la información de grupos puesto que no muestra características esenciales como son: el período, el tipo de grupo de que se trata (intersemestrales, internos, especiales, etc), el estado (abierto, cancelado o cerrado), el número de alumnos inscritos, el nivel (A, B, C ó D) y el costo del grupo.

También presenta algunas fallas de operatividad como las siguientes: no tiene una clasificación de los grupos de acuerdo a su tipo, por lo tanto no permite la creación de nuevos tipos de grupo o la modificación de ellos. Tampoco permite la creación de nuevos períodos de cursos, función primordial en la administración de cursos.

Ofrece pocas posibilidades en cuanto a la manipulación de la información dado que las operaciones que permite son muy básicas, por ejemplo, no se pueden realizar búsquedas bajo algún criterio, ya que en una aplicación como ésta, la función de búsqueda y el detalle de la misma son muy importantes.



También presenta algunas carencias en cuanto a la organización y distribución de la información en las distintas interfaces que componen el sistema, es decir, la información se encuentra concentrada en una región pequeña del área de trabajo desperdiciando la mayor parte de ella, y haciendo poco legible su manejo, ya que al presentarse una gran cantidad de datos, la interfaz se vuelve poco amigable.

Por otro lado, el proceso de inscripción de los alumnos a los grupos es un tanto engorroso. Esto se debe en mucho a que la interfaz que presenta para ello es simple y poco ortodoxa y limita muchas de las operaciones indispensables durante dicho proceso.

En lo que se refiere a los reportes, presenta algunos de ellos pero de una forma muy básica sin mucho detalle siendo éstos no de gran utilidad para la administración del proceso de las inscripciones.

Finalmente, la implementación del sistema se hizo sin la ayuda de alguna metodología de desarrollo de software que lo sustentara, por lo que la implementación presenta muchos detalles de codificación derivados del estilo de programación lineal.

Debido a lo anterior, resultó indispensable desarrollar un nuevo sistema de cómputo que cumpliera con los nuevos requerimientos encontrados hasta ese momento.

Se realizó una migración de la información que se tenía y toda ella se estandarizó bajo una nueva base de datos. El nuevo sistema se dividió en 4 módulos los cuáles abarcan todas las actividades básicas de control y administración que requiere UNICA.

A grandes rasgos los módulos 1 y 2 se refieren al control de la información relacionada con el personal que labora en UNICA; horario, datos personales, antecedentes laborales, experiencia en la impartición de cursos, etc.

El módulo 3 consiste en el control y administración de la información para la impartición de cursos. A su vez se subdivide en dos áreas. La primera se refiere al registro de cursos y la obtención de la información de las inscripciones, y el segundo es la inscripción a los

cursos. Entre las distintas actividades que debe realizar figuran: la creación de cursos, grupos, inscripción de alumnos, generación de reportes y constancias, entre otras.

El modulo 4 se refiere al apartado de cursos vía Internet, una vez que se han creado los mismos y se han publicado en el web.

El sistema que se presenta en este trabajo de tesis corresponde al modulo 3 y al modulo 4 del SICC que básicamente es el manejo de la información de los cursos.

El sistema presenta los siguientes requerimientos:

## **1.3.2 Control de la información de los grupos**

### **1.3.2.1 Información de las inscripciones**

Dado que se necesita llevar un control total acerca de los cursos que se imparten en UNICA, el sistema debe permitir la creación de dichos cursos en un período determinado. Los cursos deben ser clasificados como ya se dijo por período y por tipo.

Se requiere una bitácora o historial que contenga información correspondiente a los cursos que se imparten, en donde se incluyan datos tales como los grupos que se abrieron para determinado período, grupos cancelados, el número de alumnos asistentes a ellos, número de personas pertenecientes a la UNAM (alumno, académico y trabajador), particulares, otras instituciones y personas becadas, los instructores asignados, el lugar donde se impartieron, el costo, la duración, la fecha de inicio, la fecha de terminación y el horario.

De forma similar, se requiere de un historial de las personas que han asistido a los cursos en donde aparezcan sus datos personales tales como: nombre completo, teléfono, email, rfc y número de cuenta. Esto es básicamente para llevar un mejor control y poder realizar los descuentos pertinentes. Por otro lado, se necesita tener una estadística de que alumnos tomaron que cursos y cuantos los aprobaron.

La administración necesaria para los cursos también debe ser llevada con el sistema, por lo tanto, la aplicación se encargará de la creación de los grupos así como de la inscripción de los alumnos a los mismos. La inscripción se llevará a cabo bajo los estándares establecidos en UNICA y con la organización definida por dicha Institución.

Para la creación de los grupos, el sistema debe permitir operaciones tales como la asignación de distintos tipos de curso: semestrales (destinados a cualquier tipo de alumno. A su vez se subdividen en: intersemestrales (destinados a cualquier tipo de alumno y sólo son entre semana), fines de semana (sábados y domingos), internos (destinados únicamente al personal de UNICA), externos (son cursos que no tienen un costo fijo y están destinados a una persona o grupo de personas de una determinada empresa.), especiales (son cursos gratuitos impartidos al personal académico o administrativo de la FI.) o algún otro.

Por otro lado, debe permitir la creación de períodos y establecer a uno de ellos como el período actual de trabajo. Para cada período creado deben corresponder determinados tipos de curso y se debe contar con la posibilidad de crear nuevos tipos, de modificarlos e inclusive borrarlos.

Una vez elegido el tipo y el período al cuál corresponde un grupo en particular, se le debe asignar una sala, un instructor, un costo. También se debe determinar su duración, su fecha de inicio y de terminación, así como su horario. Todo esto debe ser cubierto por el sistema.

Por otro lado, se deben generar los siguientes reportes:

- ☞ Lista de grupos. Reporte con los datos más relevantes acerca de un grupo: nombre del curso, número de vacantes, fecha y hora de inicio.
  
- ☞ Lista de alumnos. En este reporte deben aparecer los nombres de todos los alumnos confirmados para un determinado grupo. Los datos de dicha lista son: nombre del curso, fecha de inicio y fecha de terminación. Además deben aparecer los nombres de los alumnos confirmados y un apartado para poder tomar asistencia y evaluaciones. También debe contener el nombre del instructor o instructores asignados.

- ☞ Lista de alumnos generales. En este reporte aparecerán los nombres de todos los alumnos registrados para un período en particular. Los datos que deben aparecer son los siguientes. Para los alumnos: nombre completo, rfc y si recibió constancia por el curso tomado. Los datos para el curso deben ser los siguientes: período, grupo y nombre del curso.
- ☞ Gafetes. En este reporte deben aparecer los datos de los alumnos inscritos a un curso, así como el nombre de dicho curso incluyendo su fecha de inicio y de terminación a manera de credenciales. Este reporte es muy importante porque contiene las credenciales con las cuales los alumnos se pueden identificar como asistentes a un determinado curso.
- ☞ Constancias. Se trata de un listado en el que aparecerán los nombres de aquellos alumnos que acreditaron un determinado curso así como el nombre del mismo y sus respectivas fechas de inicio y terminación.
- ☞ Diplomas de alumnos. En este reconocimiento debe aparecer el nombre del alumno, el curso que acreditó, la fecha en que tomó el curso así como la duración del mismo, además evidentemente, la Institución que lo otorga así como de las personas responsables y acreditadas para ello.
- ☞ Diplomas de Instructores. En este reporte debe aparecer el nombre del instructor así como del curso que impartió, la fecha en la que lo hizo y la duración del mismo. Debe contener el nombre de la Institución que lo otorga así como de las personas responsables de ello.
- ☞ Recibos. Consiste en un reporte tipo recibo que se entregará a los alumnos como comprobante de pago. Debe contener los siguientes datos del alumno: nombre, número de cuenta, teléfono, email y tipo de alumno. Además de los siguientes datos del curso: nombre, costo, fecha de inicio así como de terminación, horario, sala y grupo. Este recibo se generará cuando un alumno sea dado de alta y haya sido asignado a un determinado curso.

Asimismo, el sistema debe permitir también la modificación y eliminación de grupos. Para el caso de la eliminación, ésta será permitida siempre y cuando no existan alumnos inscritos al grupo.

También se permitirá la búsqueda de grupos de acuerdo a los siguientes criterios:

- ❖ Búsqueda por grupo
- ❖ Búsqueda por nombre del curso
- ❖ Búsqueda por fecha de inicio
- ❖ Búsqueda por estado (abierto o cerrado)

En cualquier momento, la aplicación debe permitir la actualización de la información mostrada para tener la certeza de que se están visualizando los datos actuales con que cuenta la base de datos.

Además, debe permitir la libre navegación entre registros para garantizar el acceso a cualquiera de ellos

### **1.3.2.2 Inscripción a grupos**

La administración deseada antes descrita para los grupos es muy similar a la requerida para los alumnos.

El sistema debe permitir el control total para la información referente a los alumnos. A través de él se podrá dar de alta un nuevo alumno con los siguientes datos: nombre completo, rfc, número de cuenta, teléfono, sexo, email. También se le asignará un tipo de alumno de acuerdo a las características antes descritas. Los tipos de alumno podrán ser los siguientes: estudiantes de la UNAM, académico de la UNAM, trabajador de la UNAM, otras instituciones educativas, becario o particular.

Finalmente, se asignará el alumno a un determinado curso si éste último tiene vacantes.

Por otro lado, se debe contar con la posibilidad de modificar alguno de éstos datos e incluso de borrar a un determinado alumno del sistema o reasignarle algún otro curso si es necesario.

También se permitirán las búsquedas de alumnos de acuerdo a los siguientes criterios:

- ❖ Búsqueda por nombre del alumno
- ❖ Búsqueda por RFC
- ❖ Búsqueda por tipo de grupo
- ❖ Búsqueda por curso

Al igual que en los grupos, se debe permitir la libre navegación entre registros y la posibilidad en todo momento de actualizar la información mostrada proveniente de la base de datos.

### **1.3.3 Reservación de un lugar en un grupo vía Internet**

Finalmente, el sistema debe mostrar los cursos a través de un sitio web y permitir el apartado de los mismos. Para ello se tendrán las siguientes restricciones: la reservación del lugar puede ser hasta de dos grupos por persona. Será válida durante tres días a partir del día que se realizó la reservación. Posteriormente, estará vacante para ser reservado por la misma u otra, el apartado de los cursos es válido sólo hasta tres días antes del inicio del curso (sin contar fines de semana y días festivos). Si en este período el alumno que realizó el apartado no se presenta a confirmar su asistencia y realizar el pago correspondiente, perderá su lugar dentro de dicho curso.

Una vez que la persona se haya inscrito debe imprimir los siguientes datos:

- ❖ Fecha en la que realizó su reservación y fecha de vencimiento.
- ❖ Nombre, apellidos y rfc
- ❖ Grupo y curso

La inscripción se realizará en persona con el responsable de las inscripciones en UNICA.

La inscripción a un grupo por Internet no es posible debido a que no existe una autoridad certificadora que autentifique las cuentas de banco de los clientes y tampoco un mecanismo de control para conocer el tipo de alumno.

## Metodología Orientada a Objetos y el Lenguaje de Modelado Unificado

### 2.1 Metodologías Orientadas a Objetos

La popularidad de las tecnologías de objetos ha generado docenas de métodos. Cada uno de ellos introduce un proceso para el análisis de un producto o sistema, un conjunto de modelos que evoluciona fuera del proceso, y una notación que posibilita al ingeniero del software crear cada modelo de una manera consistente.

En los párrafos que siguen se esbozan algunos de los métodos más populares orientados a objetos:

#### 2.1.1 El enfoque SA/SD

*SA/SD Structured Analysis/Structured Design.* La metodología del Análisis Estructurado/Diseño Estructurado incluye toda un conjunto de notaciones para especificar de manera formal el software. Durante el análisis, utiliza diagramas de flujo de datos, diagramas de transición de estados y diagramas de entidad-relación. En el diseño añade detalles a los modelos de análisis y convierte los diagramas de flujo de datos a descripciones llamadas cartas de estructuras del código del lenguaje de programación elegido.



Los diagramas de flujo de datos son el centro de atención de SA/SD. Modelan las transformaciones de los datos a medida que fluyen a través del sistema.

SA/SD divide recursivamente los procesos complejos en subdiagramas, hasta que logra muchos procesos pequeños que son fáciles de implementar. Cuando esto ocurre, la descomposición se detiene, y se escribe una especificación de proceso para cada uno de los procesos del más bajo nivel.

El diccionario de datos contiene los detalles que faltan en los diagramas de flujo de datos. Define los flujos y almacenes de datos, y el significado de los distintos nombres.

Los diagramas de transición de estados modelan el comportamiento dependiente del tiempo.

Los diagramas de entidad-relación ponen de manifiesto las relaciones entre almacenes de datos que de otro modo se verían solamente en las especificaciones de procesos.

Las herramientas anteriores se utilizan durante el proceso de análisis estructurado. El diseño estructurado sigue al análisis estructurado y aborda los detalles de bajo nivel.

### **2.1.2 JSD *Jackson System Development***

La metodología JSD o Desarrollo estructurado de Jackson fue desarrollada por Michael Jackson y es especialmente popular en Europa.

Sus modelos describen el mundo real en términos de entidades, acciones y secuencias de acciones. Las entidades suelen aparecer como sustantivos en los documentos de especificaciones de requerimientos, y las acciones aparecen como verbos.

JSD comienza ciertamente con unas consideraciones acerca del mundo real, y en ese sentido, se podría decir que esta orientado a objetos. Sin embargo, Jackson identifica pocas entidades (objetos) y muestra sólo un poco de su estructura.

La aproximación JSD es compleja y difícil de comprender en su totalidad. Una razón de dicha complejidad es el fuerte uso que hace del pseudocódigo; los modelos gráficos son más fáciles de entender.

JSD es una metodología útil para los siguientes tipos de aplicaciones:

- ❖ Software concurrente en el cual los procesos deben sincronizarse entre sí.
- ❖ Software de tiempo real. EL modelo JSD es extremadamente detallado, y se centra en el tiempo.
- ❖ Programación de computadoras paralelas. El paradigma de múltiples procesos de JSD puede servir de ayuda.

JSD no esta bien adaptado para otras aplicaciones:

- ❖ Análisis de alto nivel, JSD no facilita una alta comprensión del problema. No es eficiente a efectos de abstracción y simplificación. Maneja meticulosamente los detalles, pero no ayuda a los desarrolladores a captar la esencia del problema.
- ❖ Bases de datos. El modelado JSD está sesgado hacia las acciones, apartándose de las entidades y de los atributos. Por lo tanto, es una técnica poco adecuada para el diseño de bases de datos.
- ❖ Software convencional que corre con un sistema operativo. La abstracción de JSD formada por cientos o miles de procesos produce confusión y es innecesaria.

### 2.1.3 El método de Booch.

El método de Booch abarca un micro proceso de desarrollo y un macro proceso de desarrollo. El nivel micro define un conjunto de tareas de análisis que se reaplican en cada etapa en el macro proceso. Por esto se mantienen un enfoque evolutivo. El método Booch está soportado por una variedad de herramientas automatizadas.

Booch explica que el desarrollo orientado a objetos es fundamentalmente diferente de las aproximaciones funcionales tradicionales al diseño, como pueden ser las basadas en flujos de datos.

La metodología de Booch incluye toda una variedad de modelos que abarcan los aspectos de objetos, dinámico y funcional de un sistema de software.

### 2.1.4 Metodología de Shlaer y Mellor

Shlaer describe toda una metodología de análisis orientado a objetos que descompone el análisis en tres fases: Modelado estático de objetos, modelo dinámico de estados y sucesos, y modelado funcional.

Un fallo del tratamiento de Shlaer y Mellor es su excesiva preocupación por las tablas de bases de datos relacionales y por las claves de bases de datos.

Shlaer y Mellor dicen que su metodología es sólo una aproximación al análisis, y previenen que el diseño final podría ser distinto.

### 2.1.5 El método de Coad y Yourdon

El método de Coad y Yourdon se considera con frecuencia, como uno de los métodos más sencillos de aprender. La notación del modelado es relativamente simple y las reglas para desarrollar el modelo son evidentes.

A continuación se muestra una descripción resumida del proceso de Análisis que proponen Coad y Yourdon.

- ❖ Identificar objetos usando el criterio de “qué buscar”.
- ❖ Definir una estructura de generalización-especificación.
- ❖ Definir una estructura de todo-parte.
- ❖ Identificar temas (representaciones de componentes de subsistemas).
- ❖ Definir atributos.
- ❖ Definir servicios.

En lo referente al Diseño, el método de Coad y Yourdon se desarrolló estudiando como realizan su trabajo de diseño “diseñadores especializados del software orientado a objetos”. El enfoque de diseño se dirige no solamente a la aplicación, sino también a la infraestructura para la aplicación. Una breve descripción del proceso de diseño orientado a objetos de Coad y Yourdon sigue a continuación:

Componente del dominio del problema:

- ❖ Agrupar todas las clases específicas al dominio.
- ❖ Diseñar una jerarquía de clases apropiada para las clases de aplicación.
- ❖ Trabajar, cuando sea apropiado, para simplificar la herencia.
- ❖ Refinar el diseño para mejorar el rendimiento.
- ❖ Desarrollar una interfaz con el componente de gestión de datos.
- ❖ Refinar y añadir objetos a bajo nivel, en caso de ser necesario.
- ❖ Revisar el diseño y proponer adiciones al modelo de análisis.

Componente de interacción humana:

- ❖ Definir los actores humanos.
- ❖ Desarrollar escenarios para las tareas.
- ❖ Diseñar una jerarquía de órdenes de usuario.
- ❖ Refinar la secuencia de interacción del usuario.
- ❖ Diseñar clases relevantes y la jerarquía de clases.
- ❖ Integrar adecuadamente las clases de Interfaz Gráfica de Usuario.

Componente para la gestión de tareas:

- ❖ Identificar tipos de tareas.
- ❖ Establecer prioridades.
- ❖ Identificar la tarea que servirá de coordinadora para otras tareas.
- ❖ Diseñar objetos apropiados para cada tarea.

La componente para la gestión de datos:

- ❖ Diseñar las estructuras de datos y su distribución
- ❖ Diseñar servicios necesarios para manejar las estructuras de datos
- ❖ Identificar las herramientas que pueden ayudar en la implementación de la gestión de datos.
- ❖ Diseñar clases apropiadas y la jerarquía de clases.

## 2.1.6 El método de Jacobson

También llamado ISOO (Ingeniería del Software Orientada a Objetos), el método de Jacobson es una versión simplificada de Objectory, un método patentado, también desarrollado por Jacobson. Este método se diferencia de los otros por la importancia que da al caso de uso, una descripción o escenario que describe como el usuario interactúa con el producto o sistema.

Analiza el sistema en términos de entidades (un modelo de objetos) y casos prácticos (prototipos de escenarios que abarcan un comportamiento dinámico). Para la implementación, la funcionalidad se divide en servicios, que son grupos de requisitos funcionales relacionados. El diseño consiste en construir una arquitectura del sistema en términos de bloques modulares.

Un acercamiento al análisis de Jacobson se muestra a continuación:

- ❖ Identificar los usuarios del sistema y sus responsabilidades globales.
- ❖ Construir un modelo de requisitos.
- ❖ Definir los actores y sus responsabilidades.
- ❖ Identificar los casos de uso para cada actor
- ❖ Preparar una visión inicial de los objetos del sistema y sus relaciones.
- ❖ Revisar el modelo usando los casos de uso como escenarios para determinar su validez.
- ❖ Construir un modelo de análisis.
- ❖ Identificar objetos de interfaz usando información del tipo actor-interacción.
- ❖ Crear vistas estructurales de los objetos de interfaz.
- ❖ Representar el comportamiento del objeto.
- ❖ Aislar subsistemas y modelos para cada uno.
- ❖ Revisar el modelo usando casos de uso con escenarios para determinar su validez.

El modelo de diseño hace hincapié en el seguimiento al modelo de análisis ISOO. En seguida se muestra una breve presentación del diseño orientado a objetos de Jacobson:

- ❖ Considerar adaptaciones para hacer que el modelo de análisis ideal cumpla con el entorno del mundo real.
- ❖ Crear bloques como objetos de diseño primario. Un bloque es la abstracción de diseño que permite la representación de un objeto agregado.
- ❖ Definir un bloque para implementar el análisis de objetos relacionados.
- ❖ Identificar bloques de interfaz, bloques de entidades y bloques de control.

- ❖ Describir cómo los bloques se comunican durante la ejecución.
- ❖ Identificar los estímulos que se pasan entre bloques y su orden de comunicación.
- ❖ Crear un diagrama de interacción que muestre cómo se pasan los estímulos entre bloques.
- ❖ Organizar los bloques en subsistemas.
- ❖ Revisar el trabajo de diseño.

### 2.1.7 Método de Wirfs-Brock

El método de Wirfs-Brock no hace una distinción clara entre las tareas de análisis y diseño. En su lugar, se propone un proceso continuo que comienza con la valoración de una especificación del cliente y termina con el diseño. El análisis propuesto en éste método se describe a continuación:

- ❖ Evaluar la especificación del cliente.
- ❖ Usar un análisis gramatical para extraer las clases candidatas de la especificación.
- ❖ Agrupar las clases en un intento de determinar superclases.
- ❖ Definir responsabilidades para cada clase.
- ❖ Asignar responsabilidades a cada clase.
- ❖ Identificar relaciones entre clases.
- ❖ Definir colaboraciones entre clases basándose en sus responsabilidades.
- ❖ Construir representaciones jerárquicas de clases para mostrar relaciones de herencia.
- ❖ Construir un grafo de colaboraciones para el sistema.

Wirfs-Brock define una secuencia de tareas técnicas en el cual el análisis conduce ineludiblemente al diseño. Una breve descripción de las tareas relacionadas al diseño de Wirfs-Brock se indica a continuación:

- ❖ Construir protocolos para cada clase. Un protocolo es una descripción formal de los mensajes a los cuales la clase responderá.
- ❖ Refinar contratos entre objetos en protocolos refinados.
- ❖ Diseñar cada operación (responsabilidad)
- ❖ Diseñar cada protocolo (diseño de interfaz)
- ❖ Crear una especificación de diseño para cada clase.
- ❖ Describir, en detalle, cada contrato.
- ❖ Definir responsabilidades privadas.
- ❖ Especificar algoritmos para cada operación.
- ❖ Observar consideraciones y restricciones especiales.
- ❖ Crear una especificación de diseño para cada subsistema.
- ❖ Identificar todas las clases encapsuladas.
- ❖ Describir los contratos en detalle para los cuales el subsistema es un servidor.
- ❖ Observar consideraciones y restricciones especiales.

### 2.1.8 OMT *Object Modeling Technique* (Técnica de Modelado de Objetos)

La metodología OMT (Object Modeling Technique) fue creada por James Rumbaugh y sus colegas en 1991, mientras James dirigía un equipo de investigación de los laboratorios General Electric.

OMT es una técnica de modelado orientada a objetos que consta básicamente de las siguientes etapas: Análisis, Diseño del sistema, Diseño de objetos, Implementación y Pruebas.

La actividad de análisis crea tres modelos: el modelo de objeto (una representación de objetos, clases, jerarquías y relaciones), el modelo dinámico (una representación del comportamiento del sistema y los objetos) y el modelo funcional (una representación a alto nivel del flujo de información a través del sistema similar al Diagrama de Flujo de Datos).



OMT abarca una actividad de diseño que encara el diseño de manera que este se conduzca a dos niveles diferentes de abstracción. El diseño del sistema se centra en la distribución de los componentes necesarios para construir un producto o sistema completo. El diseño de objetos enfatiza la distribución detallada de un objeto individual.

Como pudimos observar a lo largo de este breve recorrido hacia algunas metodologías consideradas orientadas a objetos, muchas de ellas poseen rasgos en común evidentes y que las hacen en algunas situaciones, complementarias unas de otras. En consecuencia, deberían compararse más con las metodologías no orientadas a objetos que entre sí.

## 2.2 Metodología OMT

OMT *Object Modeling Technique* es una de las metodologías de Análisis y Diseño Orientadas a Objetos, más maduras y eficientes existentes hoy en día. La gran virtud que aporta esta metodología es su carácter de **abierto** (no propietaria), que le permite ser de dominio público y, en consecuencia, sobrevivir con enorme vitalidad. Esto facilita su evolución para acoplarse a las necesidades actuales y futuras de la ingeniería de software.

OMT está soportada por numerosas herramientas CASE<sup>1</sup>, de diferentes fabricantes, y ha sido adoptada por numerosas organizaciones, empresas y consultoras de América y Europa.

La Técnica de Modelado de Objetos se extiende desde el análisis hasta la implementación pasando por el diseño. En primer lugar, se construye un modelo de análisis que incluye los aspectos fundamentales de la aplicación sin tener en cuenta la futura implementación. Los objetos del modelo de análisis constituyen el marco de trabajo para el modelo de diseño, pero se implementan en términos de objetos del dominio de la computadora. Por último, el modelo de diseño se implementa en algún lenguaje de programación, base de datos o hardware.

OMT provee de una notación gráfica para expresar modelos orientados a objetos. Los objetos del dominio de la aplicación y del dominio de la computadora se pueden modelar, diseñar e implementar utilizando la misma notación orientada a objetos. Esta misma notación se utiliza desde el análisis hasta la implementación, pasando por el diseño, de una forma tal que la información añadida en una fase de desarrollo no necesita perderse, ni ser traducida, para la fase siguiente.

A grandes rasgos se pueden distinguir las siguientes fases como constitutivas de la Metodología Orientada a Objetos:

---

<sup>1</sup> CASE *Computer-Aided Software Engineering* (Ingeniería de Software Asistida por Computadora), es un conjunto de tecnologías, disciplinas y productos software, que soportan la actividad de desarrollar software en un proceso automatizado. Ejemplos: Rational Rose, Erwin 3.0, Allfusion Component Modeler, etc.

**Análisis:** Comenzando desde la descripción del problema el analista construye un modelo de la situación del mundo real que muestra sus propiedades importantes. Dicho analista debe trabajar constantemente con el cliente, puesto que muchas veces las definiciones que proporciona éste último no suelen ser completas ni correctas.

El modelo de análisis es una abstracción resumida y precisa de lo que debe hacer el sistema deseado y no de la forma en que se hará. Los objetos del modelo son conceptos del dominio de la aplicación, y no conceptos de implementación de la computadora tales como estructuras de datos. El modelo de análisis no contiene ninguna decisión de implementación.

**Diseño del sistema:** En esta etapa se toman decisiones acerca de la arquitectura global. El sistema es organizado en subsistemas basados tanto en el modelo de análisis como en la arquitectura propuesta. El diseñador de sistemas debe decidir qué características de rendimiento hay que optimizar. Selecciona una estrategia para atacar el problema y efectúa una reserva de recursos tentativa.

**Diseño de objetos:** En esta fase se construye un modelo de diseño basándose en el modelo de análisis que lleven incorporados detalles de implementación. El diseñador añade detalles al modelo de acuerdo con la estrategia establecida durante el diseño del sistema. En esta parte del ciclo de vida la atención se centra en las estructuras de datos y los algoritmos necesarios para implementar cada una de las clases. Las clases de objetos procedentes del análisis siguen siendo significativas pero se aumentan con estructuras de datos y algoritmos del dominio de la computadora seleccionados para optimizar medidas importantes de rendimiento.

Tanto los objetos del dominio de la aplicación como los del dominio de la computadora, se describen utilizando los mismos conceptos y la misma notación orientados a objetos, aunque se encuentren en planos conceptualmente diferentes.

**Implementación:** las clases de objetos y las relaciones desarrolladas durante el diseño se traducen a un lenguaje de programación concreto, a una base de datos o a una implementación

en hardware. La programación debe ser una parte relativamente pequeña del ciclo de desarrollo y fundamentalmente mecánica puesto que las decisiones de peso deben hacerse en el diseño. El lenguaje final de implementación ciertamente influye sobre las decisiones de diseño pero no debe representar un elemento del cual dependa dicho diseño.

Es posible aplicar conceptos orientados a objetos a lo largo de todo el ciclo de vida de desarrollo del sistema, desde el análisis hasta la implementación pasando por el diseño. En OMT se pueden traspasar las mismas clases de una etapa a otra sin modificar la notación aunque ganarán detalles adicionales de implementación en las etapas posteriores.

La metodología OMT emplea tres clases de modelos para describir el sistema: el *Modelo de Objetos* que describe los objetos del sistema y sus relaciones; el *Modelo Dinámico* que describe las interacciones existentes entre objetos del sistema; y el *Modelo Funcional* que describe las transformaciones de datos del sistema. Todos los modelos son aplicables en la totalidad de las fases del desarrollo y van adquiriendo detalles de implementación a medida que progresa el desarrollo. Para describir completamente al sistema se requieren los tres modelos.

El modelo de objetos describe la estructura estática de los objetos del sistema y también sus relaciones. El modelo de objetos contiene diagramas de objetos. Un *diagrama de objetos* es un grafo cuyos nodos son clases de objetos y cuyos arcos son relaciones entre clases.

El *modelo dinámico* describe los aspectos de un sistema que cambian con el tiempo. Se utiliza para especificar e implementar los aspectos del control del sistema. Los modelos dinámicos contienen *diagramas de estados*. Un diagrama de estados es un grafo cuyos nodos son estados y cuyos arcos son transiciones entre estados causadas por sucesos.

El *modelo funcional* describe las transformaciones de valores de datos que ocurren dentro del sistema. Contiene diagramas de flujo de datos. Un diagrama de flujo de datos es un grafo cuyos nodos son procesos y cuyos arcos son flujos de datos.

Los tres modelos son partes ortogonales de la descripción del sistema completo y están enlazados entre sí. Si embargo, el más importante es el modelo de objetos porque es necesario para describir qué está cambiando o transformándose antes de describir cuándo o cómo cambia.

Un procedimiento típico de software contiene esos tres aspectos: utiliza estructuras de datos (modelo de objetos), secuencia las operaciones en el tiempo (modelo dinámico) y transforma valores (modelo funcional).

Los distintos modelos no son completamente independientes (un sistema es algo más que una colección de partes independientes) pero cada modelo puede ser examinado y comprendido por sí mismo en gran parte. Cada uno de los tres modelos va evolucionando durante el ciclo de desarrollo.

### **2.2.1 Diferencias entre OMT y la metodología funcional**

El desarrollo orientado a objetos trabaja de forma inversa con respecto a la anterior tecnología orientada a funciones. En esta metodología se hace especial hincapié en especificar y descomponer la funcionalidad del sistema. Este enfoque puede parecer la forma más directa de implementar un objetivo deseado pero el sistema resultante suele ser más frágil. Si cambian los requisitos, un sistema basado en la funcionalidad puede hacer necesaria una importante reestructuración. Por el contrario, el enfoque orientado a objetos se centra primordialmente en identificar objetos ajustándoles después los procedimientos. Aunque este enfoque puede parecer más indirecto, el software orientado a objetos soporta mejor las evoluciones de los requerimientos porque está basado en el entorno subyacente del dominio de la aplicación en si más que en los requisitos funcionales de un único problema.

La metodología OMT difiere de los enfoques tradicionales del desarrollo de software. Estas diferencias afectan al proceso de desarrollo de software, y en última instancia al producto software en sí.

**Énfasis en el Análisis.** OMT traslada gran parte del esfuerzo del desarrollo de software hacia la fase de análisis del ciclo de vida. En algunas ocasiones resulta desconcertante invertir tanto tiempo en el análisis y diseño, pero este esfuerzo resulta compensado por una implementación más rápida y sencilla.

**Interés en las estructuras de datos antes que en las funciones.** OMT centra su atención en las estructuras de datos, y no en las funciones que hay que implementar. Este cambio del foco de atención proporciona al desarrollo una base más estable, y utilizar un único concepto unificador de software a lo largo del proceso: el concepto de objeto. Todos los demás conceptos, como las asociaciones, funciones y sucesos se organizan en torno a los objetos, de tal forma que la información obtenida durante el análisis no se pierde cuando se realiza el diseño y la implementación.

Las estructuras de datos son muchos menos vulnerables que las operaciones que se aplican a los datos cuando ocurren cambios en los requisitos.

La estabilidad que proporciona la organización de un sistema en torno a objetos y no en torno de funciones es invaluable y permite un desarrollo más confiable y eficaz.

**Un proceso de desarrollo sin discontinuidades.** En la técnica de Modelado de Objetos, el modelo de objetos desarrollado durante el análisis se utiliza para el diseño y para la implementación, y el trabajo se ve orientado hacia un refinamiento del modelo en vez de una transformación de una representación a otra. El proceso carece de discontinuidades en las cuáles una notación utilizada en una fase sea sustituida por otra notación distinta en otra fase.

**Iterativo más que secuencial.** Aún cuando la descripción de OMT aparece como lineal, el proceso de desarrollo real es iterativo. La carencia de discontinuidades hace más sencillo repetir los pasos de desarrollo con grados de detalle cada vez más finos. Cada iteración

añade o clarifica características, en lugar de modificar un trabajo ya hecho, de tal forma que hay menos oportunidad de introducir incongruencias y errores.

## 2.3 Procesos de la metodología OMT

Una metodología de ingeniería del software es un proceso para producir software de manera organizada, empleando una colección de técnicas y convenciones de notación predefinidas.

Los pasos de producción del software suelen organizarse en un ciclo de vida que consta de varias fases de desarrollo.

La tecnología OMT consta de varias fases o procesos:

### 2.3.1 Análisis

El análisis, que es el primer paso de la metodología OMT, se refiere a la obtención de un modelo preciso, conciso, comprensible y correcto del mundo real.

El modelo de análisis tiene varios propósitos: clarificar los requisitos, proporcionar una base para el acuerdo entre el cliente y el desarrollador, y llegar a ser el marco de trabajo para el posterior diseño e implementación. Comienza con la definición de un problema generada por el cliente y, en algunas ocasiones, por los desarrolladores. La definición puede ser incompleta o informal; el análisis hace que sea más precisa y expone las ambigüedades e incongruencias y no debería tomarse como inmutable, sino que tiene que servir como base para refinar los requisitos reales.

El modelo de análisis es una representación precisa y concisa del problema, que permite responder a preguntas y construir una solución. Se aplica a los tres aspectos de los objetos: estructura estática (modelo de objetos), secuenciado de interacciones (modelo dinámico) y transformación de datos (modelo funcional).



### 2.3.1.1 Definición del problema

La definición del problema debe indicar lo que hay que hacer, y no cómo hay que hacerlo. Debe ser una exposición clara de las necesidades, y no una propuesta de solución.

El cliente debería indicar que características son obligatorias, así como las que sean opcionales, todo ello para no restarle flexibilidad a la implementación.

La mayoría de las definiciones de problema son ambiguas, están incompletas, y no son ni siquiera congruentes. Sin embargo, la definición del problema es solamente un punto inicial para comprenderlo, y no un documento inmutable. El propósito del análisis subsiguiente es comprender en su totalidad el problema y sus implicaciones.

El analista debe trabajar con el cliente para refinar los requisitos, de tal forma que estos representen la verdadera intención de aquel. Esto implica cuestionar los requisitos y buscar la información que falta.

### 2.3.1.2 Modelado de Objetos

EL primer paso para analizar los requisitos es construir un modelo de objetos. El modelo de objetos muestra la estructura estática de datos correspondiente al sistema del mundo real, y la organiza en segmentos manejables describiendo clases de objetos del mundo real, y sus relaciones entre sí.

El modelo de objetos es el más importante de los tres modelos. Se enfatiza la construcción de un sistema en torno a los objetos y no en torno a la funcionalidad, porque el modelo de objetos se corresponde con el mundo real de manera más fiel y, por tanto, es más flexible frente al cambio. Los modelos de objetos proporcionan una representación gráfica intuitiva del sistema y son valiosos para comunicarse con el cliente y para documentar la estructura del sistema.

### 2.3.1.2.1 Notación del Modelo de Objetos

- **Objetos**

Un objeto es un concepto, abstracción o cosa con límites bien definidos y con significado a efectos del problema que se tenga. Los objetos tienen dos propósitos: promover la comprensión del mundo real y proporcionar una base práctica para la implementación por computadora.

Todos los objetos poseen su propia identidad y se pueden distinguir entre sí. El término **identidad** significa que los objetos se distinguen por su existencia inherente y no por las propiedades descriptivas que puedan tener.

- **Clases**

Una clase de objetos describe un grupo de objetos con propiedades (atributos) similares, con relaciones comunes con otros y con una semántica común.

Es frecuente utilizar la abreviatura **clase** en lugar de decir **clase de objetos**. Los objetos de una clase tienen los mismos atributos y los mismos patrones de comportamiento.

Al agrupar los objetos en clases se abstrae el problema. La abstracción da al modelado su potencia y capacidad para generalizar, partiendo de unos pocos casos específicos, hasta llegar a una multitud de casos similares.

- **Diagramas de objetos**

Los diagramas de objetos proporcionan una notación gráfica formal para el modelado de objetos, clases y sus relaciones entre sí, son útiles, tanto para el modelado abstracto como para diseñar programas reales. Hay dos tipos de diagramas de objetos: diagramas de clases y diagramas de instancias.

Un diagrama de clases es un esquema, patrón o plantilla para describir muchas instancias de datos posibles. Los diagramas de clases describen clases de objetos.

Un diagrama de instancias describe la forma en que un cierto conjunto de objetos se relacionan entre sí. Son útiles para documentar casos prácticos(casos de uso) y para describir ejemplos. Un diagrama de clases dado se corresponde con un conjunto infinito de diagramas de instancia.

El símbolo OMT de una instancia de objeto es un cuadro con esquinas redondeadas. El nombre de la clase entre paréntesis aparece en la parte superior del cuadro de objetos y en negrita. Los nombres de los objetos se escriben con tipo de letra normal. El símbolo OMT correspondiente a una clase es un cuadro con el nombre de la clase en negrita.

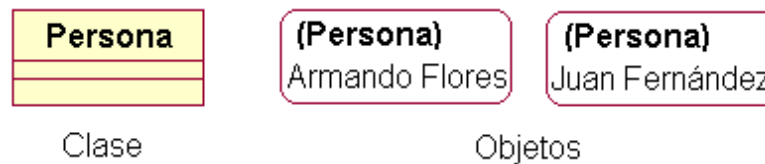


Figura 2.1. Notación OMT para una clase y para objetos.

- **Atributos**

Un atributo es un valor de un dato que está almacenado en los objetos de una clase. Cada atributo tiene un valor para cada instancia del objeto. Las instancias distintas de un cierto objeto pueden tener el mismo valor o valores distintos para un atributo dado. El nombre del atributo es único dentro de la clase pero no único en todas las clases.

El nombre de cada atributo puede ir seguido por detalles opcionales, tales como el tipo y el valor por omisión. El tipo va precedido por dos puntos, y el valor por omisión por un signo igual. Los cuadros de clase tienen una línea entre el nombre de la clase y los atributos, a diferencia de los cuadros de objetos que no tienen dicha línea, para distinguirlos aún más de los cuadros de clase.



Figura 2.2. Notación OMT para los atributos de una clase y de un objeto

- **Operaciones y Métodos**

Una operación es una función o transformación que se puede aplicar o que puede ser aplicada por los objetos de una clase.

Una misma operación puede aplicarse a muchas clases distintas. Tal operación será **polimórfica**; esto es, una misma operación adopta distintas formas en distintas clases. Un método es la implementación de una operación para una clase. Cada método puede estar implementado mediante un segmento de código diferente.

Una operación puede poseer argumentos. Tales argumentos parametrizan la operación pero no afectan a la elección del método.

Las operaciones se escriben en el tercio inferior del cuadro de clase. El nombre de cada operación puede ir seguido por detalles opcionales, tales como la lista de argumentos y el tipo de resultado. Una lista de argumentos se escribirá entre paréntesis a continuación del nombre; los argumentos irán separados por comas. El nombre y el tipo de cada argumento pueden indicarse también. El tipo y el resultado viene precedido por dos puntos y no debería omitirse porque es importante distinguir aquellas operaciones que proporcionan valores de las que no.

Una lista de argumentos vacía entre paréntesis muestra explícitamente que no hay argumentos.

- **Enlaces y Asociaciones**

Los enlaces y asociaciones son los medios para establecer relaciones entre objetos y clases.

Un enlace es una conexión física o conceptual entre instancias de objetos. Un enlace es una instancia de una asociación.

Una asociación describe un grupo de enlaces con estructura y semántica semejantes.

Todos los enlaces de cada asociación conectan objetos procedentes de las mismas clases.

Las asociaciones y los enlaces suelen aparecer como verbos en la definición del problema.

Las asociaciones describen un conjunto de enlaces potenciales del mismo modo que las clases describen un conjunto de objetos potenciales.

Cada asociación del diagrama de clases corresponde a un conjunto de enlaces en el diagrama de instancias, del mismo modo que cada clase corresponde con un conjunto de objetos. La notación OMT para las asociaciones es una línea entre clases. Se trazan los enlaces como líneas entre objetos. Los nombres de las asociaciones se ponen en cursiva.

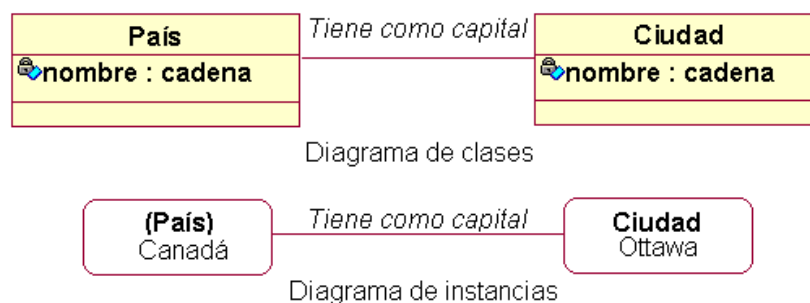


Figura 2.3. Asociaciones y enlaces en OMT

- **Multiplicidad**

La multiplicidad especifica el número de instancias de una clase que pueden estar relacionadas con una única instancia de una clase asociada. Los diagramas de objetos indican la multiplicidad mediante símbolos especiales al final de las líneas de asociación. En el caso más general se puede especificar la multiplicidad mediante un número o mediante un conjunto de intervalos, tal como puede ser “1” (uno exactamente), “1+” (uno ó más), “3-5” (entre tres y cinco), y “2,4 y 18” (dos, cuatro ó bien dieciocho). Existen terminadores de línea especiales que indican ciertos valores frecuentes de multiplicidad. Un círculo negro es el símbolo OMT que denota “muchos”, lo cual quiere decir cero o más. Un círculo blanco indica “opcional”, lo cual quiere decir cero o uno. Una línea sin símbolos de multiplicidad denota una asociación uno a uno. La multiplicidad se escribe junto al final de la línea de asociación.

- **Atributos de los enlaces**

Un atributo de enlace es una propiedad de los enlaces de una asociación. La notación OMT para un atributo de enlace es un cuadro ligado a la asociación mediante un lazo; pueden aparecer uno o más atributos de enlace dentro del cuadro.

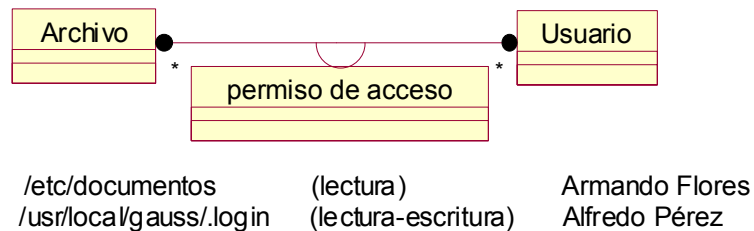


Figura 2.4. Atributos de enlace para una asociación

- **Nombre de rol**

Una asociación binaria posee dos roles, cada uno de los cuales puede poseer un nombre de rol. Un nombre de rol es un nombre que identifica de forma única un extremo de una asociación. Un rol es un extremo de una asociación.

Los roles suelen aparecer como sustantivos en las descripciones de problemas. El nombre de rol se escribe junto a la línea de la asociación, al lado de la clase que desempeñe.

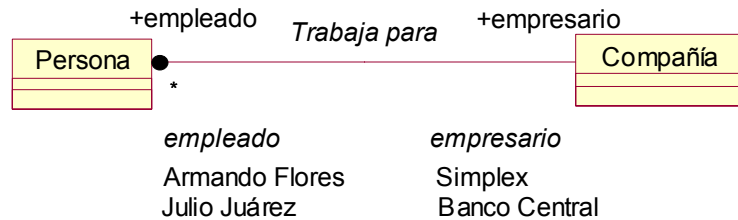


Figura 2.5. Nombres de rol para una asociación

Los nombres de rol son necesarios para las asociaciones entre dos objetos de la misma clase.

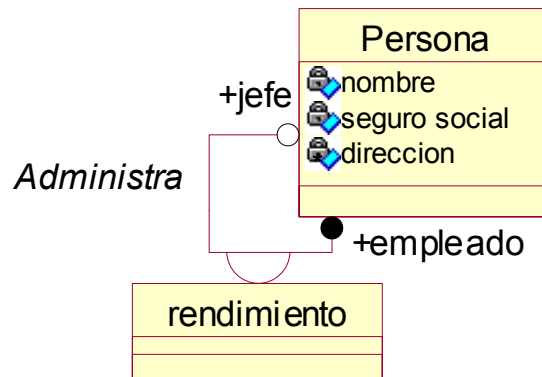


Figura 2.6. nombres de rol para una asociación entre dos objetos de una misma clase

- **Clasificación**

Normalmente, los objetos del lado “muchos” de una asociación no tienen un orden explícito y se pueden considerar como un conjunto. Sin embargo, en algunas ocasiones los objetos están ordenados explícitamente, por ejemplo, las ventanas abiertas en una pantalla de una computadora. Dichas ventanas están ordenadas explícitamente de tal forma, que la ventana más próxima al usuario es visible desde cualquier punto de la pantalla.

Un conjunto ordenado de objetos en el extremo “muchos” de la asociación se indica escribiendo “{ordenado}” al lado del símbolo de multiplicidad correspondiente al rol.

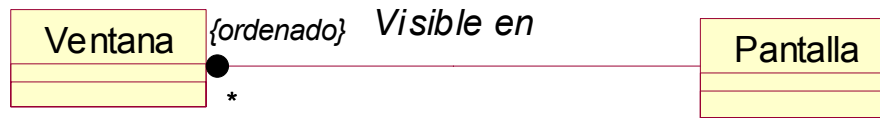


Figura 2.7. Conjuntos ordenados en una asociación

- **Cualificación**

Una asociación cualificada o calificada relaciona dos clases de objetos y un cualificador.

El cualificador es un atributo especial que reduce la multiplicidad efectiva de una asociación.

Por ejemplo, un directorio posee muchos archivos. Un archivo sólo puede pertenecer a un único directorio. En el contexto de un directorio el nombre del archivo especifica un único archivo. **Directorio** y **Archivo** son clases de objetos y **nombre de archivo** es el cualificador.

Las asociaciones uno a muchos y muchos a muchos pueden ser cualificadas. El cualificador distingue entre el conjunto de objetos que se encuentra en el extremo “muchos” de la asociación.

La cualificación mejora la precisión semántica e incrementa la visibilidad de las vías de navegación. Un cualificador se dibuja en la forma de un cuadro pequeño en el extremo de la línea de asociación que se encuentra más próximo a la clase a la cual califica.



Figura 2.8. Asociación cualificada



- **Agregación**

Una agregación es la relación “parte-todo” o “una-parte-de” en la cual los objetos que representan los componentes de algo se asocian a un objeto que representa el ensamblaje completo.

La propiedad más significativa de la agregación es la transitividad, esto es, si A es parte de B y B es parte de C, entonces A es parte de C. La agregación es, además, antisimétrica esto es, si A es parte de B, entonces B no es parte de A. Finalmente, algunas propiedades del ensamblaje se propagan también a sus componentes, posiblemente con modificaciones locales.

Se define la relación de agregación como aquella que relaciona una clase ya ensamblada con una clase componente.

Las agregaciones se dibujan igual que las asociaciones, salvo por un pequeño rombo que indica el extremo de ensamblaje de la relación.

El siguiente ejemplo muestra la relación de agregación entre Documento, Párrafo y Frase. Un documento consta de muchos párrafos, cada uno de los cuales contiene un gran número de frases.

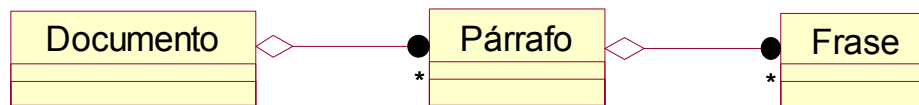


Figura 2.9. Relación de agregación entre clases

- **Generalización y Herencia**

La generalización es la relación entre una clase y una o más versiones refinadas de esa misma clase. La que se está refinando se denomina la **superclase** y cada versión refinada se denomina una **subclase**. Los atributos y operaciones comunes a un grupo de subclases se

asocian a la superclase y son compartidos por todas las subclases. Se dice que cada subclase hereda las características de su superclase.

Toda subclase hereda, no solamente todas las características de sus antecesoras sino que además añade sus propios atributos y operaciones específicos.

La notación para la generalización es un triángulo que conecta una superclase con sus subclases. La superclase se conecta mediante una línea a la parte superior del triángulo. Las subclases se conectan mediante líneas a una barra horizontal asociada a la base del triángulo.

La generalización es una construcción útil para el modelado conceptual como para la implementación. La generalización facilita el modelado estructurando las clases y capturando de forma concisa lo que es similar y lo que es distinto con respecto a las clases. La herencia de operaciones resulta útil durante la implementación como vehículo para la reutilización del código.

La siguiente figura ilustra una generalización de equipo. Cada pieza del equipo es una bomba, un intercambiador de calor ó un tanque. Existen varias clases de bombas: centrífugas, de diafragma y de pistón. De igual forma, se presentan tres clases de tanques: de techo flotante, a presión y esféricos.

Se muestran varias instancias de objetos en la parte inferior de la figura. Cada objeto hereda las características de una clase en cada nivel de la generalización.

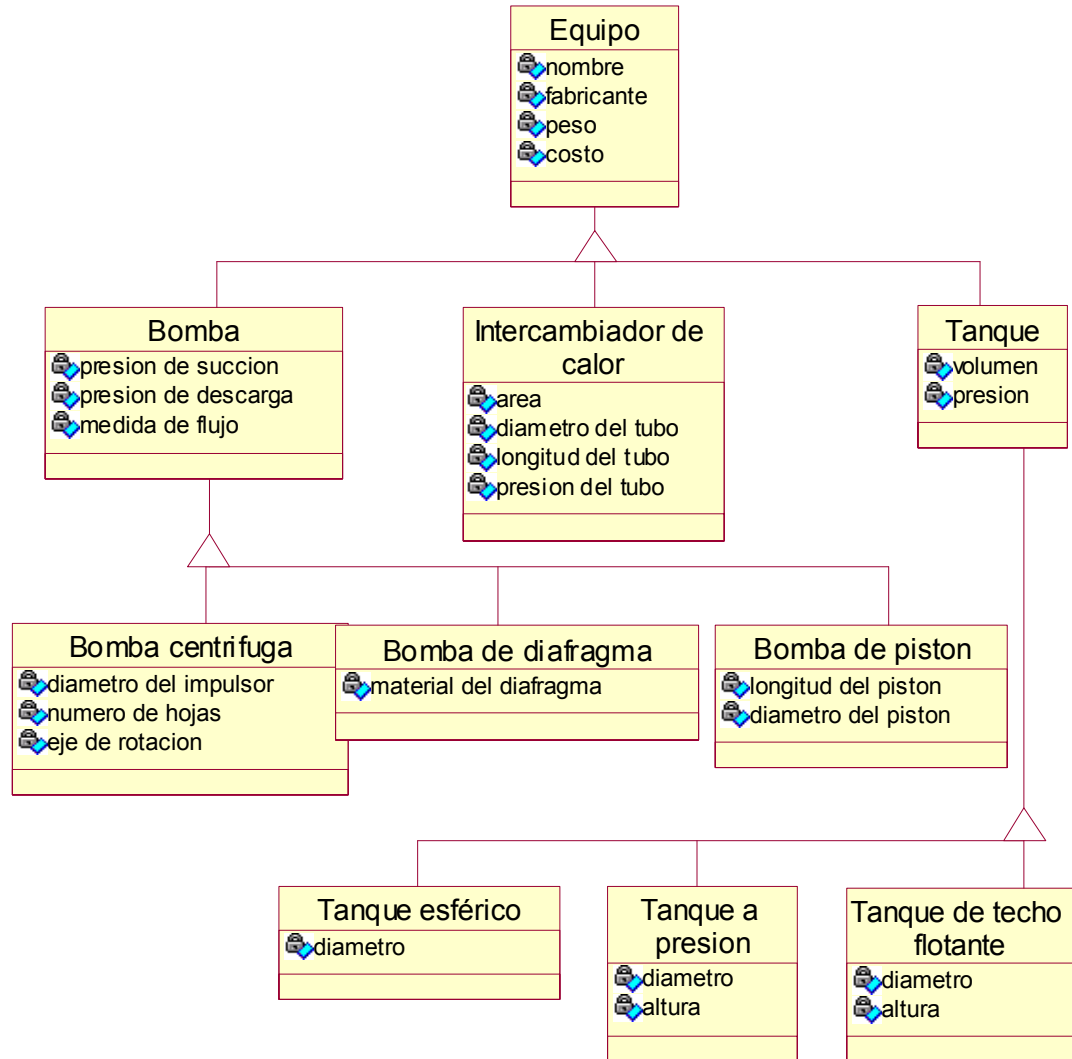


Figura 2.10. Herencia en OMT

- **Módulo**

Un módulo es una construcción lógica para agrupar clases, asociaciones y generalizaciones. Los módulos capturan una perspectiva o vista de una situación.

Los nombres de clases y asociaciones deberán ser únicos dentro del módulo. El nombre del módulo suele enumerarse en la parte superior de cada hoja. No existe otra notación especial para los módulos.

La información del modelo de objetos proviene de la definición del problema, del conocimiento de expertos acerca de la aplicación que se pretende, y del conocimiento general del mundo real. Si el diseñador no es un experto de la aplicación, puede obtener la información que requiera a partir de un experto de la aplicación, y habrá que confrontar repetidamente el modelo con esta información.

### 2.3.1.3 Elaboración del Modelo de Objetos

Para construir un modelo de objetos, se llevan a cabo los siguientes pasos:

- **Identificar los objetos y las clases**

El primer paso para construir un modelo de objetos es identificar las clases de objetos relevantes en la aplicación. Entre éstos se encuentran las entidades físicas, tales como casas, empleados y máquinas, así como conceptos como trayectorias, reservas de asientos y planes de pago.

- **Preparar un diccionario de datos**

Se debe preparar un diccionario de datos para todas las entidades de modelado, ya que las palabras aisladas tienen demasiadas interpretaciones. Hay que escribir un párrafo que describa con precisión cada clase de objetos. El diccionario de datos también describe las asociaciones, atributos y operaciones.

- **Identificar las asociaciones entre objetos**

Toda dependencia entre dos o más clases es una asociación y una referencia de una clase a otra también lo es.

Las asociaciones suelen corresponderse con verbos de estado o con locuciones verbales.

Entre ellas se incluye la ubicación física (junto a, parte de, o contenido en), las acciones dirigidas (conduce), las comunicaciones (habla con), la propiedad (tiene o parte de), o el cumplimiento de alguna condición (trabaja para, casado con o administra).

- **Identificar atributos de objetos y enlaces**

Los atributos son propiedades de objetos individuales, como el nombre, velocidad o color.

A diferencia de las clases y de las asociaciones, es menos probable que los atributos se describan por completo en la definición del problema. Es preciso recurrir a nuestro conocimiento acerca de la aplicación, y del mundo real, para encontrarlos.

Los atributos derivados deben ser omitidos o bien deben ser marcados claramente. Los atributos de enlace también deben ser identificados. Un atributo de enlace es una propiedad del enlace entre dos objetos, en lugar de ser una propiedad de un objeto individual.

- **Organizar y simplificar las clases de objetos empleando la herencia**

El paso siguiente es organizar las clases empleando la herencia para compartir una estructura común. La herencia se puede añadir en dos direcciones: Generalizando aspectos comunes de clases existente en una superclase (refinamiento ascendente) o bien refinando las clases existentes para dar subclases especializadas (refinamiento descendente).

- **Iterar y refinar el modelo**

Dado que es raro que un modelo de objetos sea correcto a la primera pasada, todo el proceso de desarrollo de software se convierte en una continua iteración. Si se encuentra un error, hay que volver a una etapa anterior, si es necesario para corregirlo.

- **Agrupar las clases en módulos**

El último paso del modelado es agrupar las clases en folios y módulos. Los diagramas se pueden dividir en folios de tamaño uniforme por comodidad para dibujarlos, imprimirlos y estudiarlos. Las clases que estén fuertemente acopladas deberían ir agrupadas, pero teniendo en cuenta que un folio tiene una cantidad prefijada de información, las rupturas serán, en ocasiones, arbitrarias. Un módulo es un conjunto de clases (uno o más folios) que captura algún subconjunto lógico del modelo completo.

### 2.3.1.4 Modelado Dinámico

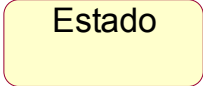

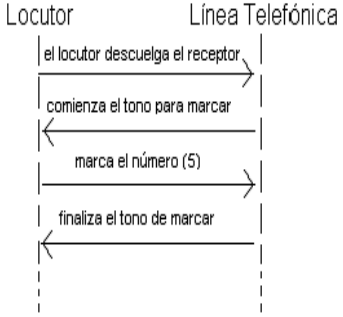
El modelo dinámico muestra la forma en que el comportamiento del sistema y de los objetos de que consta va variando con el tiempo.




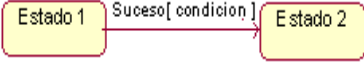
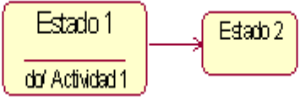
Básicamente se refiere a la parte del sistema que describe las secuencias de operaciones que se producen en respuesta a estímulos externos, sin tener en cuenta los que hagan las operaciones, aquello a lo que afectan o la manera en que serán implementadas.

Los conceptos más importantes del modelado dinámico son los sucesos, que representan estímulos externos, y los estados, que representan los valores de los objetos.

Ambos aparecen en lo que se denomina un diagrama de estados, un grafo cuyos nodos son los estados, y cuyos arcos dirigidos son transiciones enmarcadas por nombres de sucesos.

El siguiente cuadro muestra los elementos más importantes que componen un diagrama de estados:

Elemento	Descripción	Notación	Ejemplo
Estado	Se denomina como estado a los valores de los atributos y de los enlaces mantenidos por un objeto. Los estados tienen duración, ocupan un intervalo de tiempo y suelen asociarse con actividades continuas o con actividades que tardan un cierto tiempo en concluir.	Los estados se representan como cuadros redondeados que contienen un nombre opcional.	
Suceso	Un suceso o evento es una transmisión de información de dirección única entre un objeto y otro. Transcurre durante un período de tiempo.	La notación de un suceso es muy sencilla, sólo se coloca el nombre del suceso sobre la línea que une a los estados involucrados.	
Seguimiento de sucesos	Es un diagrama que muestra una secuencia de sucesos y los objetos que intercambian dichos sucesos.	Este diagrama muestra cada objeto como una línea vertical, y cada suceso como una flecha horizontal que va desde el objeto emisor al objeto receptor. El tiempo aumenta de arriba hacia abajo.	

Transición	Es un cambio de estado causado por un suceso	Se representan en forma de flechas desde el estado receptor hasta el estado de destino	
Estado Inicial	Indica el comienzo de una secuencia de estados	Se muestra mediante un círculo negro	
Estado Final	Indica el fin en una secuencia de estados	Se denota mediante un círculo blanco con el centro negro.	
Condición	Es una función booleana lógica que tiene a objetos como valores. Se puede utilizar como protección en las transiciones. Una transición con protección se dispara cuando se produce su suceso, pero sólo si la condición de protección es verdadera.	Las condiciones de protección de una transición se muestran como expresiones booleanas entre corchetes a continuación del nombre del suceso.	
Actividad	Es una operación cuya realización requiere un cierto tiempo. Pueden ser operaciones continuas (mostrar una imagen en pantalla), u operaciones secuenciales que terminan por sí mismas.	La notación "hacer:A" dentro del cuadro de un estado indica que la actividad A principia al entrar en ese estado y finaliza al salir de él.	



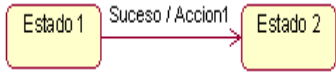
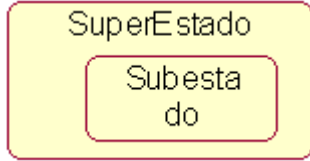
Acción	Es una operación instantánea que va asociada a un suceso	La notación de una acción que afecta a una transición es una barra (“/”) y el nombre (o descripción), después del nombre del suceso que la produce.	
Subestado	Los estados pueden tener subestados que heredan sus transiciones. Dichos estados son llamados entonces superestados.	Los superestados se dibujan como un gran cuadro redondeado que encierra a todos sus subestados.	

Tabla 2.1. Elementos más importantes de un diagrama de estados

Para construir un modelo dinámico se llevan a cabo los pasos siguientes:

- **Preparación de escenarios**

Un escenario es un diálogo típico entre el usuario y el sistema. Los escenarios muestran las principales interacciones, los formatos de visualización externa y los intercambios de información.

En primer lugar se preparan escenarios para casos “normales”, esto es, situaciones sin ninguna entrada extraña y sin errores. A continuación se consideran los casos “especiales”, tales como secuencias de entrada que se omiten, valores límite (máximo y mínimo), y valores repetidos. Luego se consideran casos de error por parte del usuario, incluyendo valores incorrectos y casos en que no hay respuesta.

Un escenario es además una secuencia de sucesos. Los sucesos se producen siempre que se intercambia información entre un objeto del sistema y un agente externo. Los valores de

información intercambiados son parámetros del suceso. Para cada suceso hay que identificar el actor (sistema, usuario o algún otro agente), y también hay que identificar los parámetros del suceso.

- **Identificación de sucesos**

A través de un análisis de los escenarios se pueden identificar todos los sucesos externos.

Estos incluyen las señales, entradas, decisiones, interrupciones, transacciones y acciones procedentes o destinadas al usuario o a dispositivos externos.

También se deben agrupar con un mismo nombre aquellos sucesos que tengan el mismo efecto sobre el flujo de control. Cada tipo de suceso se debe asignar a las clases de objetos que lo envían y lo reciben.

Los escenarios deben mostrarse como una **traza de sucesos**, esto es, una lista ordenada en la que las columnas corresponden a objetos. Si hay más de un objeto de la misma clase que participa en el escenario, se le asigna una columna distinta. Siguiendo una determinada columna, se pueden ver los sucesos que afectan directamente a un objeto concreto.

- **Construcción de un diagrama de estados**

Se debe construir un diagrama de estados para todas las clases de objetos que tengan un comportamiento dinámico no trivial.

El diagrama de estados de una clase se considera terminado cuando abarque todos los escenarios, y cuando maneje todos los sucesos que afecten a un objeto de dicha clase.

- **Correspondencia de sucesos entre objetos**

Una vez completados todos los diagramas de estados para las distintas clases relevantes del sistema, se debe comprobar la consistencia del sistema.

Los estados sin predecesores o sucesores resultan sospechosos así que hay que asegurarse de que representen puntos iniciales o finales de la secuencia de interacción.

Hay que asegurarse de que los sucesos correspondientes entre distintos diagramas de estados sean congruentes.

El conjunto de diagramas de estados para las clases de objetos que tienen un comportamiento importante dentro del sistema constituye el modelo dinámico de una aplicación.

### **2.3.1.5 Modelado Funcional**

El modelo funcional describe los cálculos existentes dentro del sistema sin detallar ni cómo ni cuándo se calculan. Muestra la forma en que se derivan los valores producidos en un cálculo a partir de los valores de entrada.

#### **2.3.1.5.1 Notación del Modelado Funcional**

- **Diagramas de flujo de datos**

El modelo funcional consta de múltiples diagramas de flujo de datos, que especifican el significado de las operaciones y de las restricciones. Un diagrama de flujo de datos (DFD) muestra las relaciones funcionales entre los valores calculados por un sistema, incluyendo los valores introducidos, los obtenidos, y los almacenes internos de datos. Un DFD contiene

**procesos** que transforman datos, **flujos de datos** que los trasladan, objetos **actores** que producen y consumen datos, y **almacenes de datos** que los almacenan de forma pasiva.

Un proceso transforma valores de datos. Los procesos se dibujan en forma de elipses que contienen una descripción de la transformación, normalmente su nombre. Cada proceso tiene un número fijo de entradas y salidas, cada una de las cuales lleva un tipo asociado. Dichas entradas y salidas se pueden rotular, para mostrar su uso en el cálculo.

Los procesos se implementan como métodos de operaciones que se aplican a clases de objetos.

Un flujo de datos conecta la salida de un objeto o proceso con la entrada de otro objeto o proceso. Se dibujan como flechas entre el productor y el consumidor de ese valor de datos. La flecha se rotula con una descripción de los datos. El mismo valor se puede enviar a distintos lugares; esto se denota mediante una bifurcación con varias flechas que emergen de ella.

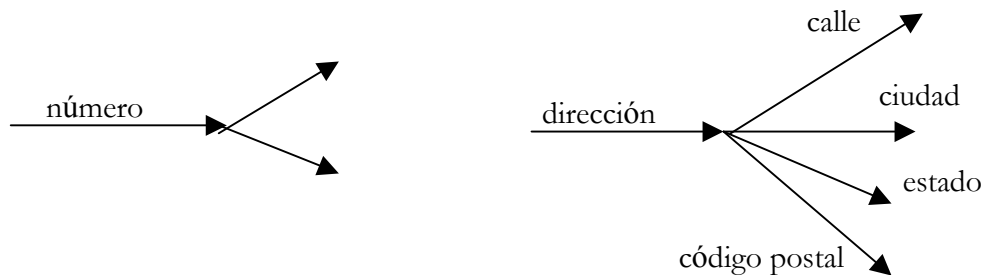


Figura 2.11. Representación del flujo de datos en el modelado funcional

Cada flujo de datos representa un valor en algún momento del cálculo. Los flujos de datos internos al diagrama representan valores intermedios dentro de un cálculo, y no tienen necesariamente significado en el mundo real.

Un actor es un objeto activo que controla el grafo de flujo de datos produciendo o consumiendo valores. Los actores están asociados a las entradas y salidas del grafo del flujo de

datos. Los actores son llamados **terminadores** porque se encuentran en la frontera del grafo y porque hacen que concluya el flujo de datos como fuentes o como receptores de datos.

Los actores se representan como rectángulos para mostrar que son objetos. Las flechas entre el actor y el diagrama son las entradas y salidas del diagrama.

Un almacén de datos es un objeto pasivo que almacena datos para su posterior utilización. Permiten acceder a los valores por un orden distinto al que fueron generados. Se dibujan en forma de un par de líneas paralelas que contienen el nombre del almacén. Las flechas de entrada indican información u operaciones que afectan los datos almacenados; las flechas de salida indican información que se ha extraído del almacén.

Un diagrama de flujo de control muestra todas las posibles vías de computación para los valores. No muestra cuáles son las vías que se ejecutan ni en que orden (modelo dinámico).

Un flujo de control es un valor booleano que afecta a si un proceso es o no evaluado. Se dibuja mediante una línea discontinua que va desde un proceso que produce un valor booleano hasta el que se está controlando.

- **Operaciones**

Al hablar de operaciones dentro del modelo funcional se debe hacer referencia a una signatura y a una transformación para especificar cualquier operación. La signatura define la interfaz de la operación: argumentos (número, orden y tipo), y valores de salida (número, orden y tipos) . La transformación define el efecto de una operación es decir, la relación de los valores de salida en función de los valores de entrada y los efectos colaterales en los objetos tomados como argumentos.

Básicamente el modelado funcional trata tres tipos de operaciones: consultas, acciones y actividades.

Las consultas son operaciones que carecen de efectos laterales para un objeto. Por ejemplo, un objeto se puede expresar tanto en coordenadas polares como en coordenadas cartesianas; esto no afecta al objeto.

Una acción es una transformación que presenta efectos laterales sobre el objeto destino y también sobre aquellos objetos que resulten alcanzables por dicha operación.

Una actividad es una operación hecha por o sobre un objeto que presenta una cierta duración de tiempo, en contraste con las consultas y acciones que se consideran instantáneas.

Para construir un modelo funcional se deben seguir los siguientes pasos:

- **Identificar valores de entrada**

Se enumeran los valores de entrada y de salida. Dichos valores son los parámetros de los sucesos que se intercambian entre el sistema y el exterior.

- **Construcción de diagramas de flujo de datos**

En esta etapa se construye un diagrama de flujo de datos que muestre la forma en que se calcula cada valor de salida a partir de las entradas.

Se deben distinguir los almacenes de datos de los flujos de datos y de los procesos, lo cual resulta sencillo puesto que los almacenes de datos reciben valores que no dan lugar a salidas inmediatas.

- **Descripción de funciones**

Una vez refinado el diagrama de flujo de datos, se debe escribir una descripción de cada función en donde se establezca lo que hace pero no la forma en que se debe implementar.

Dicha descripción puede hacerse en lenguaje natural, en pseudocódigo, en ecuaciones matemáticas o algún otro formato adecuado.

- **Identificación de restricciones**

Se deben identificar las restricciones entre objetos. Las restricciones son dependencias funcionales entre objetos diferentes a las dependencias entrada-salida. Pueden afectar a dos objetos a la vez, a dos instancias de un mismo objeto en momentos diferentes o entre instancias de diferentes objetos en momentos diferentes.

- **Especificación de criterios de optimización**

En esta etapa se especifican aquellos valores que deben ser maximizados, minimizados u optimizados de alguna forma.

- **Iteración del análisis**

Generalmente los modelos de análisis requieren más de una pasada para finalizar. La forma de abordar el análisis no es lineal debido a que las distintas partes del problema interactúan.

En una primera pasada, el modelo de análisis puede mostrar incongruencias y desequilibrios. Hay que iterar las distintas fases para obtener un diseño más limpio y coherente.

El modelo final debe ser revisado por el cliente y por expertos de la aplicación, los cuales deben asegurarse de que se está modelando correctamente el mundo real.

El modelo de análisis una vez refinado sirve como base para la arquitectura, diseño e implementación del sistema.

### 2.3.2 Diseño del Sistema

El diseño de sistemas es la etapa en la que se decide cual será la estrategia para resolver el problema y construir una solución. Dichas decisiones se refieren a la organización del sistema en subsistemas, la asignación de dichos subsistemas a componentes de hardware y software, decisiones conceptuales y de política que constituyen el entorno de trabajo para el diseño detallado.

La organización global del sistema es lo que se denomina la **arquitectura del sistema**, la cuál proporciona el contexto en el cual se toman las decisiones más detalladas en una fase posterior al diseño.

Para realizar el diseño de sistemas se deben tomar las decisiones siguientes:

- **Organizar el sistema en subsistemas.**

Se divide el sistema en un pequeño número de componentes. Cada uno de los componentes principales del sistema se llama subsistema. Estos subsistemas agrupan aspectos del sistema que comparten alguna propiedad común (funcionalidad similar, misma ubicación física).

Un subsistema es un paquete de clases, asociaciones, operaciones, sucesos y restricciones interrelacionados, y que tienen una interfaz bien definida con los demás subsistemas.

Un subsistema puede ser organizado por capas o por particiones. Cuando se presenta por capas, el sistema es un conjunto ordenado de mundos virtuales, cada uno de los cuales está construido en términos de los que tiene abajo, y proporciona la base de implementación para aquellos que estén por encima de él.



Las particiones dividen verticalmente un sistema en varios subsistemas independientes o débilmente acoplados, cada uno de los cuales proporciona una clase de servicio.

- **Identificación de la concurrencia**

Basándonos en el modelo dinámico podemos encontrar la concurrencia en un sistema.

Dos objetos son inherentemente concurrentes si pueden recibir sucesos al mismo tiempo sin interactuar.

Al examinar los diagramas de estado de objetos individuales y de intercambio de sucesos entre ellos, es frecuente encontrar que muchos objetos se pueden ejecutar en un único hilo de control. Un hilo de control es una vía a través de un conjunto de diagramas de estado en la cual solamente está activado un objeto en cada instante.

- **Asignación de subsistemas a procesadores y a tareas.**

Cada subsistema asociado debe ser asociado a una unidad de hardware ya se aun procesador o una unidad funcional especializada.

El diseñador del sistema debe estimar la potencia de procesamiento del equipo de hardware. Generalmente la estimación es imprecisa.

El diseñador del sistema debe decidir qué subsistemas serán implementados en hardware y cuáles en software.

- **Administración de almacenes de datos**

Dentro de los distintos tipos de almacenamiento de datos podemos encontrar los siguientes:

Los archivos, que son una forma de almacenamiento de datos barata, sencilla y permanente. Las implementaciones de los archivos son distintas según las diferentes plataformas, así que se debe tener cuidado en aislar las dependencias con sistemas de archivos.

Las bases de datos son otro tipo de almacenamiento. Existen varios tipos: jerárquicas, en red, relacionales, orientadas a objetos y lógicas.

Las bases de datos son potentes y hacen que las aplicaciones sean más fáciles de transportar a sistemas operativos y a distintas plataformas. Una desventaja es que algunas bases de datos tienen una interfaz algo compleja.

- **Manejo de recursos globales**

Se deben identificar los recursos globales y se tienen que establecer mecanismos de acceso a ellos. Entre los recursos globales se encuentran: Unidades físicas, tales como procesadores, unidades de cinta y satélites de comunicación: el espacio, tales como el espacio en disco: nombres lógicos, tales como la identificación de los objetos, nombres de archivos y nombres de clases, y el acceso a datos compartidos tales como bases de datos.

- **Manejo de las condiciones de contorno**

Las condiciones de contorno a que se hace referencia en esta parte son las siguientes:

**Iniciación.** El sistema debe traerse desde un estado inicial de reposo hasta una situación estacionaria duradera, es decir, se debe considerar el escenario en el cual el sistema empezará a funcionar desde un estado completo de inactividad y permanecerá funcionando por un período indefinido. Entre las cosas que hay que iniciar se incluyen los datos constantes, parámetros, variables globales, tareas y posiblemente la jerarquía de clases.

**Terminación.** La terminación suele ser más sencilla que la iniciación, porque hay muchos objetos internos que se pueden abandonar. La terminación es básicamente un proceso de liberación de recursos.

**Fallos.** Un fallo es la terminación no planeada de un sistema. Pueden surgir de errores del usuario, del agotamiento de recursos del sistema o de algún fallo externo.

### 2.3.3 Diseño de objetos

Durante el diseño de objetos, se ejecuta la estrategia elegida durante el diseño del sistema. El énfasis ahora recae en los conceptos propios de las computadoras en lugar de en conceptos propios de la aplicación. En particular, las operaciones encontradas durante el análisis deben expresarse en forma de algoritmos, descomponiendo las operaciones complejas en operaciones más sencillas. Las clases, atributos y asociaciones del análisis deben implementarse en forma de estructuras de datos específicas.

Una vez concluido el análisis se tienen los modelos de objetos, dinámico y funcional, sin embargo, el modelo de objetos es el más importante pues representa el entorno alrededor del cual se construye el diseño.

El diseñador debe transformar las acciones y actividades del modelo dinámico y los procesos del modelo funcional en operaciones asociadas a las clases del modelo de objetos. Al efectuar esta transformación, comienza el proceso consistente en hacer corresponder la estructura lógica del modelo de análisis en una organización física de un programa.

Las decisiones de diseño deben ser documentadas porque lo contrario producirá confusión. Esto resulta especialmente cierto cuando se está trabajando con otros desarrolladores. La documentación suele ser la mejor manera de transmitir el diseño a otros, y también de registrarlo para hacer referencia a él durante la fase de mantenimiento.

El documento de diseño incluye una descripción revisada y mucho más detallada del modelo de objetos, tanto en forma gráfica (diagramas del modelo de objetos) como en forma textual (descripción de clases).

### 2.3.4 Implementación

La mayoría de los lenguajes ejecutables es capaz de expresar los tres aspectos de la especificación del software: Estructura de datos, flujo dinámico de control y transformación funcional.

Las estructuras de datos se expresan en un subconjunto declarativo del lenguaje. El flujo de control se puede expresar bien mediante procedimientos (condicionales, bucles y llamadas) o bien sin procedimientos (reglas, restricciones, tablas y máquinas de estados).

Las transformaciones funcionales se expresan en términos de los operadores primitivos del lenguaje, así como en forma de llamadas a subprogramas.

Es relativamente sencillo implementar un diseño orientado a objetos con un lenguaje orientado a objetos, dado que las estructuras del lenguaje son similares a las estructuras de diseño. En general, un lenguaje orientado a objetos admite objetos, polimorfismo y herencia.

En el caso de OMT, los tres modelos antes descritos contribuyen al desarrollo de código. El modelo de objetos contiene la mayor parte de la estructura declarativa: la especificación de clases, atributos, jerarquía de herencia y asociaciones. El modelo dinámico especifica la estrategia de control de alto nivel para el sistema: controlado por procedimientos, controlado por sucesos o multitarea. El modelo funcional captura la funcionalidad de los objetos, que deberá incorporarse a los métodos.

## 2.4 Lenguaje de Modelado Unificado (UML)

UML *Unified Modeling Language* (Lenguaje Unificado de Modelado) es una herramienta que ayuda a capturar la idea de un sistema para comunicarla posteriormente a quien esté involucrado en su proceso de desarrollo; sirve como enlace entre quien tiene la idea y el desarrollador; esto se lleva a cabo mediante un conjunto de símbolos y diagramas. Cada diagrama tiene fines distintos dentro del proceso de desarrollo.

Antes de la llegada de UML, el desarrollo de sistemas era, con frecuencia, una propuesta al azar. Los analistas de sistemas intentaban evaluar los requerimientos de sus clientes, generar un análisis de requerimientos en algún tipo de notación que ellos mismos comprendieran (aunque el cliente no lo comprendiera), dar tal análisis a uno o varios programadores y esperar que el producto final cumpliera con lo que el cliente deseaba.

Dado que el desarrollo de sistemas es una actividad humana, inminentemente esta propensa a presentar errores en cualquier etapa del proceso, por ejemplo, el analista pudo haber malentendido al cliente y en consecuencia, generó un documento erróneo de análisis. Tal vez ese documento tampoco fue comprendido por los programadores quienes, por ende, generaron un programa difícil de utilizar y que no representaba la solución que el cliente esperaba.

### 2.4.1 La necesidad de UML

En los principios de la computación, los programadores no realizaban análisis muy profundos sobre el problema por resolver. Si acaso, hacían un bosquejo en una hoja de papel. Con frecuencia comenzaban a escribir el programa desde el principio.

Hoy en día, es necesario contar con un plan bien analizado. El cliente debe comprender que es lo que hará el equipo de desarrolladores; además debe ser capaz de señalar cambios si no se han captado claramente sus necesidades. A su vez, el desarrollo es una suma

de esfuerzos orientado a equipos, por lo que cada uno de sus miembros debe entender perfectamente qué lugar toma su trabajo en la solución final, así como saber cual es la solución general.

Un arquitecto no podría crear una compleja estructura como lo es un edificio de oficinas sin crear primero un anteproyecto detallado; asimismo un desarrollador de software no podría generar un complejo sistema de cómputo en tal edificio de oficinas sin crear antes un plan de diseño detallado. La idea central es que así como un arquitecto le muestra un anteproyecto a la persona que lo contrató, el desarrollador de software deberá mostrarle su plan de diseño al cliente. Tal plan de diseño debe ser el resultado de un cuidadoso análisis de las necesidades del cliente.

Otra característica del desarrollo de sistemas contemporáneo es reducir los tiempos de entrega. Cuando los plazos se encuentran muy cerca uno del otro es absolutamente necesario contar con un diseño robusto.

La necesidad de diseños robustos ha traído consigo la creación de una notación de diseño que los analistas, desarrolladores y clientes acepten como guía y que además sirva como estándar para la comunicación entre miembros de la comunidad de desarrollo de software. UML es dicha notación.

### **2.4.2 Origen de UML**

UML es la creación de Grady Booch, James Rumbaugh e Ivar Jacobson. Ellos trabajaban en empresas distintas durante la década de los años ochenta y principios de los noventa y cada uno diseñó su propia metodología para el análisis y diseño orientado a objetos. Sus metodologías predominaron sobre las de sus competidores. A mediados de los años noventa empezaron a intercambiar ideas entre sí y decidieron trabajar en equipo.

En 1994 Rumbaugh ingresó a Rational Software Corporation, donde ya trabajaba Booch. Jacobson ingresó a Rational un año después.

Los primeros trabajos de UML empezaron a circular y las reacciones de la industria del software no se dejó esperar trayendo consigo considerables modificaciones. Conforme diversos corporativos vieron que UML era útil para sus propósitos, se conformó un consorcio del UML. Entre los miembros se encuentran DEC, Hewlett-Packard, Intellicorp, Microsoft, Oracle, Texas Instruments y Rational. En 1997 el consorcio publicó la versión 1.0 de UML y la puso a consideración del OMG *Object Management Group* (Grupo de administración de objetos).

El consorcio aumentó y generó la versión 1.1, misma que se puso nuevamente a consideración del OMG. El grupo adoptó esta versión a finales de 1997. El OMG se encargó de la conservación del UML y produjo otras dos versiones en 1998. UML ha llegado a ser el estándar en la industria del software, y su evolución continúa.

## 2.5 Diagramas de UML

UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. Debido a que UML es un lenguaje, cuenta con reglas para combinar tales elementos.

La finalidad de los diagramas es presentar diversas perspectivas de un sistema, conocidas como **modelo**. El modelo UML de un sistema es similar a un modelo a escala de un edificio junto con la interpretación del arquitecto del edificio. Cabe señalar que un modelo UML describe lo que hará un sistema, pero no dice cómo implementar dicho sistema.

A continuación se describirán brevemente los diagramas más comunes de UML y los conceptos que representan.

- **Diagrama de clases**

Una clase es una categoría o grupo de cosas que tienen atributos y acciones similares.

El símbolo para representar a una clase en UML es un rectángulo que se divide en tres áreas. El área superior contiene el nombre de la clase, el cual es por convención, una palabra empezando con la primer letra en mayúscula. El área central contiene los atributos. Un atributo es una propiedad o característica de una clase y describe valores que la propiedad puede contener en los objetos de dicha clase. Por convención, el nombre del atributo va escrito con letras minúsculas. Además, se puede especificar un tipo para cada valor del atributo: cadena (string), número de punto flotante (float), entero (integer) y booleano (boolean) o algún otro. El tipo va separado por dos puntos (:) del nombre del atributo. Por último, el área inferior contiene las operaciones o acciones. Una operación es algo que una clase puede realizar. El nombre de la operación también va en minúsculas. Es posible indicar información adicional en las operaciones como puede ser los argumentos que recibe junto con su tipo, ó el tipo de dato devuelto.



A continuación se muestra un ejemplo sencillo para ilustrar lo anterior. Se trata de una clase “lavadora” la cual servirá como base para desarrollar los ejemplos que ilustrarán los diagramas restantes.

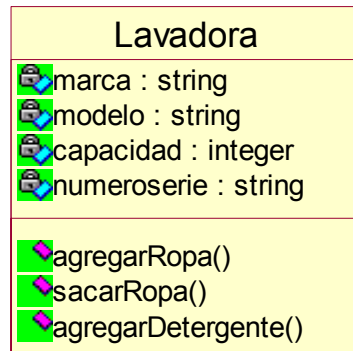


Figura 2.12. Notación UML para una clase

- **Asociaciones**

Una asociación es una conexión entre dos clases. Se representa por una línea con punta de flecha que conecta a ambas clases, con el nombre de la asociación arriba de dicha línea.

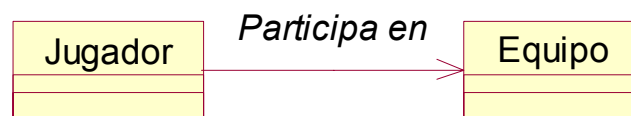


Figura 2.13. Asociaciones en UML

- **Multiplicidad**

La multiplicidad especifica la cantidad de objetos de una clase que se relacionan con un objeto de la clase asociada.

Hay varios tipos de multiplicidad: uno a uno, uno a muchos, uno a uno ó más, uno a ninguno o uno, uno a un intervalo (por ejemplo, uno a cinco hasta diez), uno a exactamente  $n$ , o uno a un conjunto de opciones (por ejemplo, uno a nueve o diez). UML denota la multiplicidad **más** con un asterisco(\*) y también a la multiplicidad **muchos** . El contexto O se

representa por dos puntos, como en “1..\*” (“uno ó más”). El contexto O también se denota por una coma, como en “5,15” (“5 o 15”).

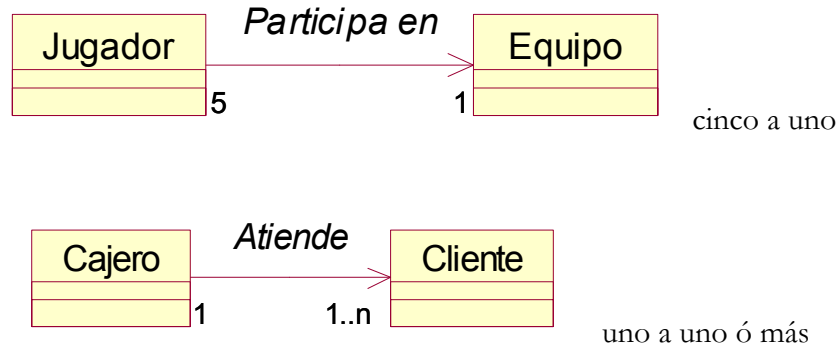


Figura 2.14. Distintos tipos de multiplicidad en UML

- **Herencia y Generalización**

Una clase (subclase) puede heredar los atributos y operaciones de otra (superclase). En el mundo de la orientación a objetos, lo anterior se conoce como herencia. UML lo denomina generalización y la representa con una línea que une a la clase principal o superclase, con la clase secundaria o subclase. En la parte de la línea que se conecta con la clase principal, se coloca un triángulo sin rellenar que apunta a la clase principal.

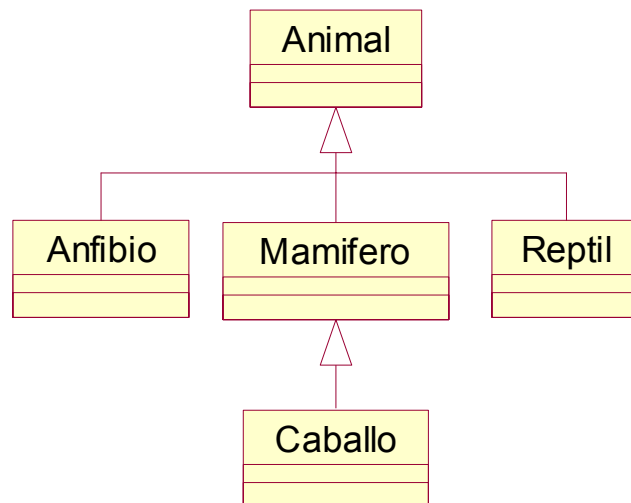


Figura 2.15. Notación de UML para la herencia y generalización

Los diagramas de clases facilitan las representaciones a partir de las cuales los desarrolladores pueden trabajar. Colaboran en lo referente a la etapa de análisis puesto que permiten al analista hablarle a los clientes en su propia terminología. En un diagrama de clases vamos a encontrar básicamente los elementos antes descritos: clases, asociaciones, multiplicidad, etc. El propósito fundamental del diagrama de clases es el de mostrar las entidades que actúan en el sistema y como se relacionan entre sí.

- **Agregaciones**

La agregación es un tipo especial de asociación que indica que una clase consta de otras clases. Se representa como una jerarquía dentro de la clase completa en la parte superior, y los componentes por debajo de ella. Una línea conecta el todo con un componente mediante un rombo sin relleno que se coloca en la línea más cercana al todo.

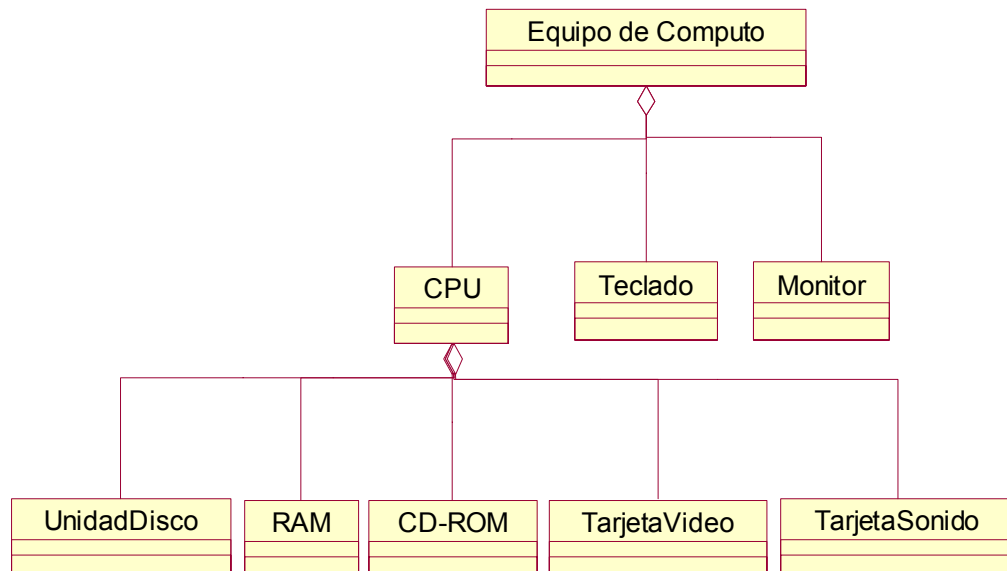


Figura 2.16. Notación UML para la agregación

- **Interfaces**

Una interfaz es un conjunto de operaciones que especifica cierto aspecto de la funcionalidad de una clase, y es un conjunto de operaciones que una clase presenta a otras.

Por ejemplo, el teclado que utilizamos para comunicarnos con la computadora es una interfaz reutilizable proveniente de la máquina de escribir. La forma de representar una clase y su interfaz es con un pequeño círculo que se conecta mediante una línea a la clase.



Figura 2.17 Representación de UML para las interfaces

- **Diagrama de objetos**

Un objeto es una instancia de clase (una entidad que tiene valores específicos de los atributos y acciones). El símbolo que utiliza UML para representar un objeto es un rectángulo con el nombre subrayado. El nombre del objeto se encuentra a la izquierda de dos puntos (:) que lo separan del nombre de la clase de la que precede.

Siguiendo con el ejemplo de la lavadora, la forma en que se representaría un objeto de la clase Lavadora sería la siguiente:

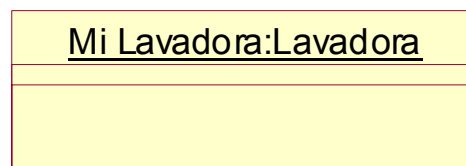


Figura 2.18. Representación de objetos en UML

- **Diagrama de casos de uso**

Un caso de uso es una descripción de las acciones de un sistema desde el punto de vista del usuario. Esta es una herramienta sumamente valiosa puesto que permite a los desarrolladores obtener una los requisitos del sistema desde el punto de vista del usuario.

El caso de uso esta formado básicamente por tres elementos: el actor, el caso de uso y las líneas de interconexión. El actor es el que inicia el caso de uso y puede ser una persona u otro sistema. Las líneas de interconexión son aquellas que permiten conectar al actor con el caso de uso.

Generalmente, los actores están fuera del sistema, mientras que los casos de uso están dentro de él. La notación de UML para los diagramas de casos de uso designa un rectángulo (con el nombre del sistema en algún lugar dentro de él) para representar el confín del sistema. El rectángulo envuelve a los casos de uso del sistema. Una elipse representa el caso de uso, mientras que una figura agregada representa al actor.

Para el ejemplo de la clase Lavadora, un caso de uso podría ser “lavar ropa”. La siguiente figura ilustra esto con la notación correspondiente de UML:

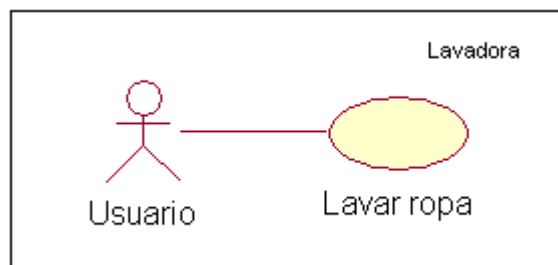


Figura 2.19. Casos de Uso

- **Diagrama de estados**

El diagrama de estados de UML presenta los estados en los que puede encontrarse un objeto junto con las transiciones entre los estados, y muestra los puntos inicial y final de una secuencia de cambios de estado.

Un rectángulo de vértices redondeados representa a un estado, junto con una línea continua y una punta de flecha, mismas que representan a una transición. Un círculo relleno simboliza un punto inicial y un círculo con un círculo relleno dentro representa un estado final.



Figura 2.20. Representación de estados en UML

Un diagrama que representaría los cambios de estado en una lavadora sería el siguiente:

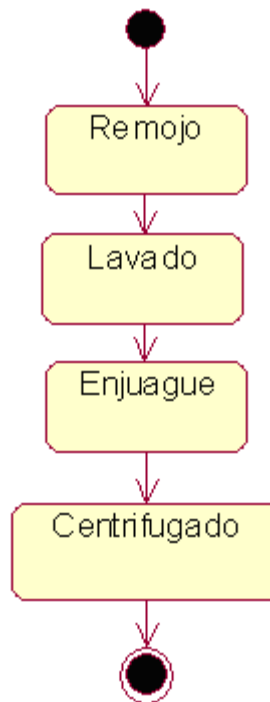


Figura 2.21. Diagrama de estados

- **Diagrama de secuencias**

El diagrama de secuencias de UML muestra la mecánica de la interacción con base en tiempos. Esta formado por objetos que se representan de la forma usual: rectángulos con nombre subrayado, mensajes representados por líneas continuas con punta de flecha y el tiempo se representa como una progresión vertical.

Los objetos se colocan en la parte superior del diagrama de izquierda a derecha. Debajo de cada objeto se dibuja una línea discontinua conocida como **línea de vida** de un objeto. Junto con la línea de vida de un objeto se encuentra un pequeño rectángulo conocido como **activación**, el cual representa la ejecución de una operación que realiza el objeto. La longitud de dicho rectángulo representa la duración de la activación.

Continuando con el ejemplo de la lavadora, entre los componentes de la lavadora se encuentran: una manguera de agua (para obtener agua), un tambor (donde se coloca la ropa) y un sistema de drenaje. Para el caso de uso “lavar ropa”, el diagrama de secuencias sería similar al siguiente:

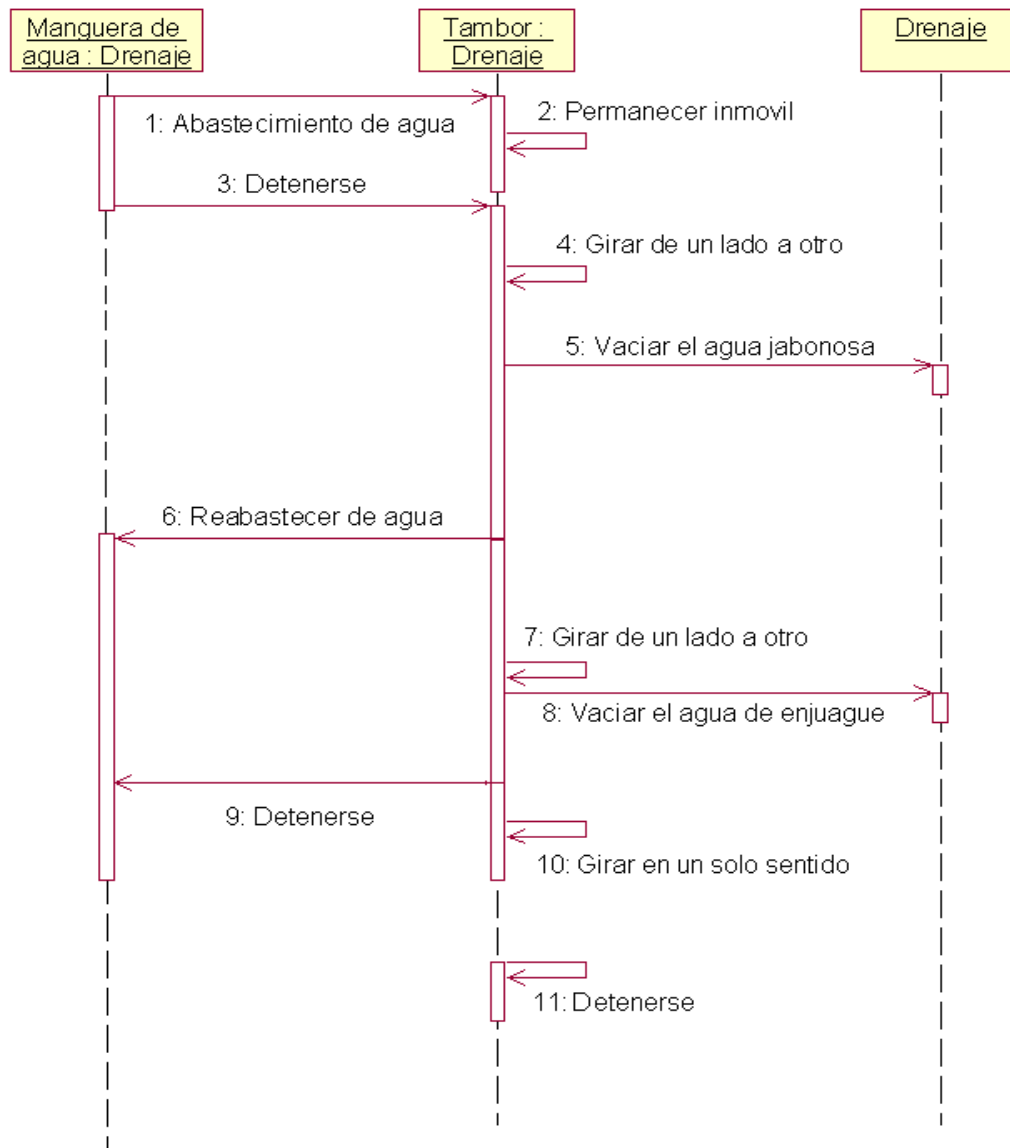


Figura 2.22. Diagrama de secuencias

- **Diagrama de actividades**

El diagrama de actividades de UML es muy parecido a los viejos diagramas de flujo. Muestra los pasos o actividades que ocurren durante una operación o proceso. Se puede considerar como una extensión de un diagrama de estados, recordemos que éste último



muestra los estados de un objeto y representa las actividades como flechas que conectan a los estados. El diagrama de actividades resalta, precisamente, a las actividades.

A cada actividad se le representa por un rectángulo con las esquinas redondeadas un poco más angosto y ovalado que la representación correspondiente a un estado. Una flecha representa la transición de una actividad a otra. Al igual que el diagrama de estados cuenta con un punto inicial y un punto final designados con los mismos símbolos.

A continuación se muestra un diagrama de actividades para el ejemplo de la lavadora:

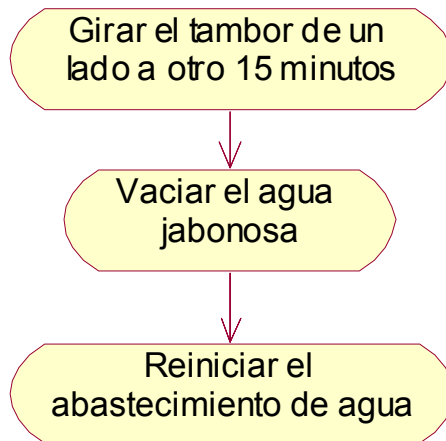


Figura 2.23. Diagrama de Actividades

- **Diagrama de colaboraciones**

Los diagramas de colaboraciones muestran la forma en que los objetos colaboran entre sí. Muestra los objetos junto con los mensajes que se envían entre ellos.

Para representar un mensaje, se dibuja una flecha cerca de la línea de asociación entre dos objetos, esta flecha apunta al objeto receptor. El tipo de mensaje se muestra en una etiqueta cerca de la flecha; por lo general, el mensaje indica que el objeto receptor ejecute

alguno de sus métodos u operaciones. El mensaje finaliza con un par de paréntesis dentro del cual pueden ir o no los argumentos con los que funcionará la operación.

Para ilustrar lo anterior, tenemos el siguiente ejemplo de una GUI (Interfaz Gráfica de Usuario). Un actor indica la secuencia de interacción al oprimir una tecla, con lo que los mensajes ocurren de manera secuencial. Tal secuencia se puede mostrar en un diagrama de colaboraciones de la siguiente manera:

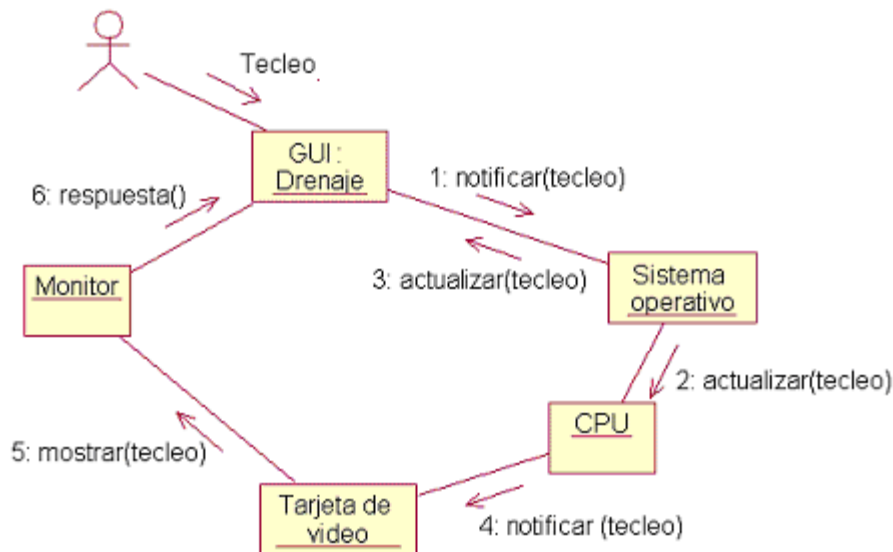


Figura 2.24. Diagrama de Colaboraciones

- **Diagrama de componentes**

Un componente de software es una parte física de un sistema que se encuentra en la computadora, por ejemplo, una tabla, un archivo, un documento, un archivo ejecutable, etc.

A grandes rasgos un componente es la implementación en código de una clase.

Los componentes son útiles puesto que permiten a los clientes ver la estructura del sistema finalizado; además proporcionan la estructura con la que los desarrolladores trabajarán

en adelante y un camino para que las personas encargadas de escribir las notas técnicas y la documentación entiendan de que escribirán.

Un concepto importante al tratar con los componentes es el de *interfaz*. Una interfaz es un conjunto de operaciones que especifica algo respecto al comportamiento de una clase. De manera general, se puede entender a la interfaz como un conjunto de operaciones que presenta una clase a otras.

Otro concepto importante en esta parte es el de **realización**.

Se llama realización a la relación entre una clase y su interfaz.

El símbolo que utiliza UML para denotar a un componente es un rectángulo que tiene otros dos sobrepuestos en su lado izquierdo. El nombre del componente se coloca dentro del símbolo.

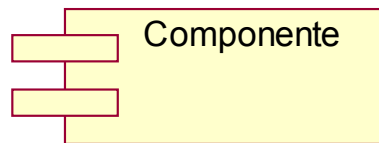


Figura 2.25. Notación UML para un componente

Existen dos formas para representar a un componente y sus interfaces: la primera muestra la interfaz como un rectángulo que contiene la información que se le relaciona, se conecta al componente por la línea discontinua y una punta de flecha representada por un triángulo sin rellenar que visualiza la realización.

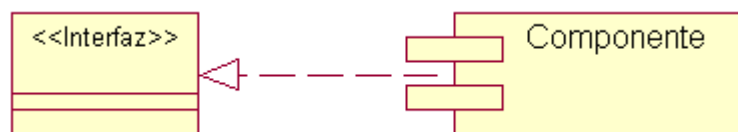


Figura 2.26. 1a. Representación de un componente y su interfaz

La otra forma representa a la interfaz como un pequeño círculo que se conecta al componente por una línea continua. En este contexto la línea representa la relación de realización.

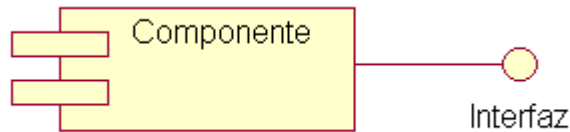


Figura 2.27. 2a. Representación de UML para un componente y su interfaz

- **Diagrama de distribuciones**

El diagrama de distribución de UML muestra la arquitectura física de un sistema de cómputo. Puede representar los equipos y dispositivos, mostrar sus interconexiones y el software que se encontrará en cada máquina. Cada computadora se representa por un cubo y las relaciones entre las computadoras están representadas por líneas que conectan a los cubos.

A grandes rasgos, el diagrama de distribución de UML ilustra la forma en que lucirá un sistema físicamente cuando se haya realizado.

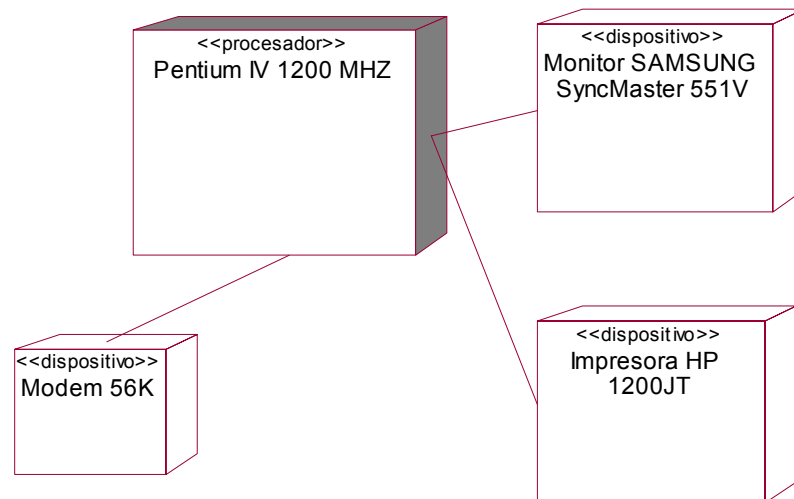


Figura 2.28. Diagrama de distribuciones

2.5.1 Referencia rápida de UML

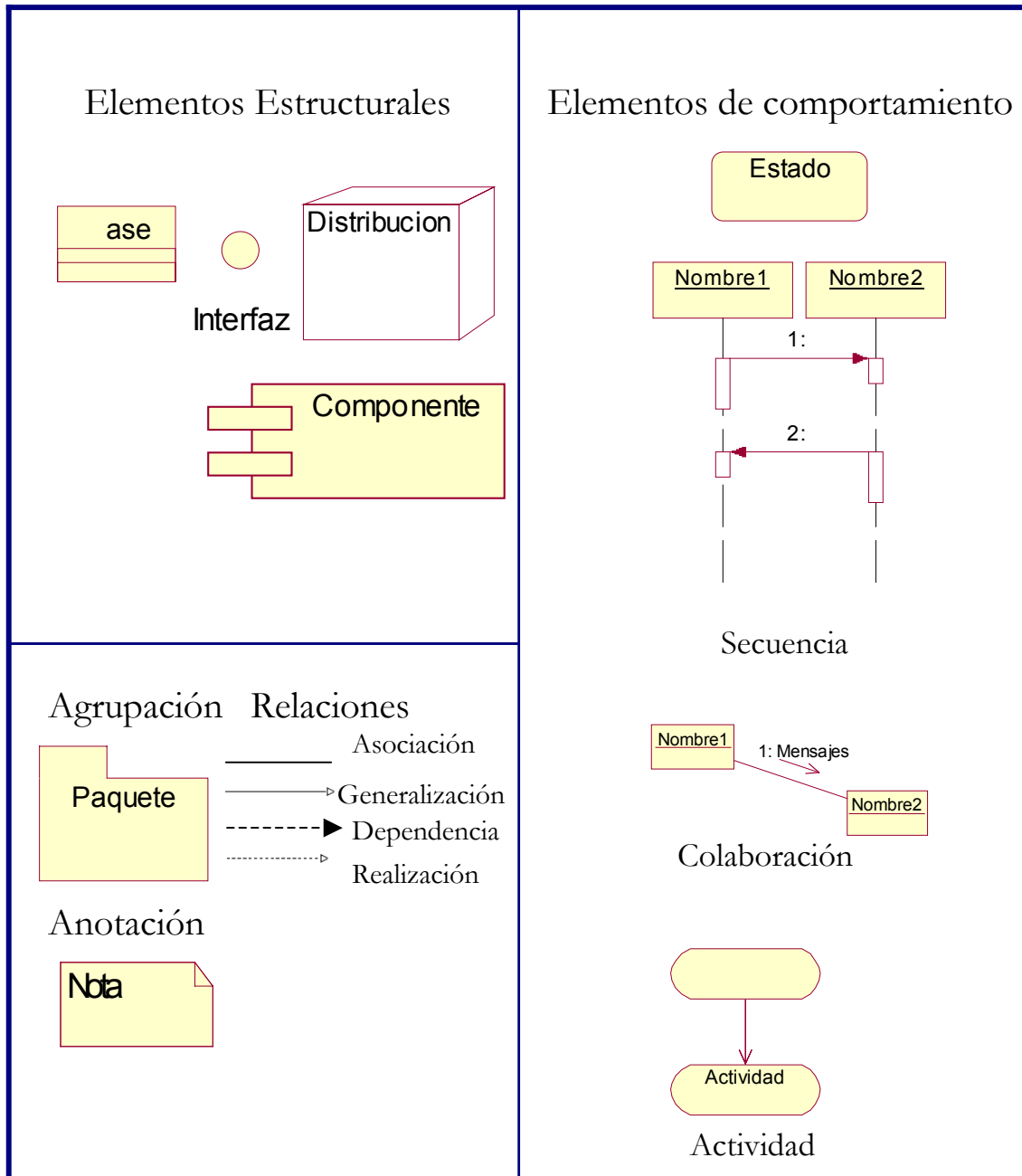


Tabla 2.2. Referencia breve de UML

## 2.6 Relación del Lenguaje de Modelado Unificado y la Metodología OMT

El lenguaje de Modelado Unificado UML ofrece un modo estándar de visualizar, especificar, construir, documentar y comunicar los elementos que constituyen una aplicación de software. Sin embargo, la utilización de UML debe estar inmersa en el marco de una metodología robusta que defina **qué**, **cuándo** y **cómo** dichos elementos intervienen en el proceso de desarrollo del software. Para el desarrollo e implementación del SICC se eligió utilizar OMT como metodología de desarrollo y a UML como lenguaje de modelado.

Este enfoque ofrece muchas ventajas:

Uno de las personas que participó en la creación de UML es James Rumbaugh quien es considerado el principal desarrollador de OMT. Rumbaugh aportó los conocimientos que adquirió durante su trabajo con OMT para el desarrollo de UML. Por lo tanto, en muchos aspectos se puede observar que tanto OMT como UML comparten criterios acerca de la forma de analizar y diseñar sistemas informáticos.

OMT se extiende desde el análisis hasta la implementación pasando por el diseño. Es precisamente en la fase de análisis y diseño de OMT donde se utilizó a UML para dar soporte a los diversos modelos requeridos en estas etapas.

Como se mencionó anteriormente, en la etapa de análisis se elaboran tres modelos que proporcionan una visión completa del problema y sirven como antesala del diseño. Se trata del modelo de objetos, el modelo dinámico y el modelo funcional. El modelo de objetos muestra la estructura estática de datos correspondiente al sistema del mundo real y lo organiza en clases, objetos y relaciones entre ellos. Para ello utiliza diagramas de clases el cual se corresponde directamente con el diagrama de clases de UML. Esta relación es posible debido a que utilizan términos definidos de forma similar: clases, objetos, atributos, asociaciones,

multiplicidad, roles, generalización, herencia, etc. De hecho, la notación es prácticamente la misma y lo más importante es que se conserva el mismo enfoque.

Para mostrar la parte dinámica del sistema, OMT cuenta con el modelo dinámico el cual describe los aspectos del sistema que cambian con el tiempo. Básicamente se utiliza para especificar e implementar los aspectos de control del sistema. Hace uso de los diagramas de estado para lograr su objetivo. UML proporciona dos diagramas que tienen el mismo cometido. El diagrama de estados y el diagrama de secuencias. El diagrama de estados de UML se corresponde directamente con el de OMT. Sus conceptos son prácticamente los mismos al igual que su notación. El diagrama de secuencias se corresponde directamente con el diagrama de seguimiento de sucesos de OMT planteado también en la etapa del modelado dinámico.

Utiliza los mismos elementos y comparten la misma idea en cuanto a la organización de ellos en el diagrama y su objetivo dentro de dicho diagrama.

El modelo funcional describe las transformaciones de valores de datos que ocurren dentro del sistema. Hace uso de diagramas de flujo de datos. UML proporciona una especie de diagrama de flujo de datos mejorado. Se trata del diagrama de actividades. Es muy parecido a los viejos diagramas de flujo. Muestra los pasos (en UML **actividades**), así como puntos de decisión y bifurcaciones; además presenta mejoras bastante significativas con respecto a su predecesor lo cual lo hace más útil durante el análisis.

La etapa de diseño de OMT también puede ser apoyada por diagramas de UML. Se trata de los diagramas de componentes y distribución los cuáles permiten dar una visión gráfica global del sistema acorde con los objetivos del diseño en OMT.

Como se puede observar, prácticamente toda la etapa de análisis y diseño propuesta por OMT es soportada por UML de una forma bastante sólida y estable. La correspondencia entre diagramas y conceptos es sumamente significativa y promueve la utilización de ambas herramientas durante el proceso de desarrollo de software de cualquier sistema informático.

## Implementación del sistema de control de cursos e inscripciones

### 3.1 Herramientas CASE a utilizar

#### 3.1.1 Herramientas CASE: Definición

Las herramientas de Ingeniería de Software Asistida por Computadora (CASE *Computer-Aided Software Engineering*), son un conjunto de tecnologías, disciplinas y productos software, que reemplazan al papel y al lápiz por la computadora para transformar la actividad de desarrollar software en un proceso automatizado. El uso de herramientas CASE permite mejoras radicales en la productividad y en la calidad en todos los aspectos del desarrollo de aplicaciones software.

A grandes rasgos, los objetivos del uso de herramientas CASE son los siguientes:

- ❖ Permitir la aplicación práctica de metodologías, lo que resulta muy difícil sin emplear herramientas.
- ❖ Facilitar la realización de prototipos y el desarrollo conjunto de aplicaciones.
- ❖ Simplificar el mantenimiento del software.
- ❖ Mejorar y estandarizar la documentación.



- ❖ Aumentar la portabilidad de las aplicaciones.
- ❖ Facilitar la reutilización de componentes de software.
- ❖ Permitir un desarrollo y un refinamiento -visual- de las aplicaciones, mediante la utilización de controles gráficos (piezas de código reutilizables).
- ❖ Mejorar la calidad del software desarrollado.
- ❖ Reducir tiempos y costos de desarrollo y mantenimiento del software.

### 3.1.2 Clasificación de las herramientas CASE

No existe una única clasificación de herramientas CASE y, en ocasiones, es difícil incluirlas en una clase determinada. Estas podrían clasificarse de acuerdo a:

- Las plataformas que soportan.
- Las fases del ciclo de vida del desarrollo de sistemas que cubren.
- La arquitectura de las aplicaciones que producen.
- Su funcionalidad.

Las herramientas CASE, en función de las fases del ciclo de vida, se pueden agrupar de la forma siguiente:

**Herramientas integradas, I-CASE** (*Integrated CASE*, CASE integrado): abarcan todas las fases del ciclo de vida del desarrollo de sistemas.

Se basan generalmente en una metodología y aportan técnicas para todas las fases del ciclo de vida. Sin embargo, no todas ellas son modernas en el sentido de aprovechar la potencia de las estaciones de trabajo o la utilización de lenguajes de alto nivel o técnicas de construcción de prototipos.

**Herramientas de alto nivel, U-CASE** (*Upper CASE* - CASE superior o *front-end*)

Orientadas a la automatización y soporte de las actividades desarrolladas durante las primeras fases del desarrollo: análisis y diseño.

**Herramientas de bajo nivel, L-CASE** (*Lower CASE* - CASE inferior o *back-end*)

Dirigidas a las últimas fases del ciclo de vida: desarrollo e implementación.

Otra posible clasificación, tomando como criterio principal la funcionalidad es la siguiente:

**Herramientas de planificación de sistemas de gestión**

Sirven para modelizar los requisitos de información estratégica de una organización. Proporcionan un "metamodelo" del cual se pueden obtener sistemas de información específicos. Su objetivo principal es ayudar a comprender mejor cómo se mueve la información entre las distintas unidades organizativas. Estas herramientas proporcionan una ayuda importante cuando se diseñan nuevas estrategias para los sistemas de información y cuando los métodos y sistemas actuales no satisfacen las necesidades de la organización.

**Herramientas de Análisis y Diseño**

Permiten al desarrollador crear un modelo del sistema que se va a construir. Entre ellas podemos encontrar:

- Herramientas de análisis y diseño (Modelado).
- Herramientas de creación de prototipos y de simulación.
- Herramientas para el diseño y desarrollo de interfaces.

**Herramientas de programación**

Se engloban aquí los compiladores, los editores y los depuradores de los lenguajes de programación convencionales. Ejemplos de estas herramientas son:

- Herramientas de codificación convencionales.
- Herramientas de codificación de cuarta generación.
- Herramientas de programación orientadas a objetos.

### **Herramientas de integración y prueba**

Sirven de ayuda a la adquisición, medición, simulación y prueba de los equipos lógicos desarrollados. Entre las más utilizadas están:

- Herramientas de análisis estático.
- Herramientas de generación de casos de prueba.

### **Herramientas de gestión de prototipos**

Los prototipos son utilizados ampliamente en el desarrollo de aplicaciones, para la evaluación de especificaciones de un sistema de información, o para un mejor entendimiento de cómo los requisitos de un sistema de información se ajustan a los objetivos perseguidos.

### **Herramientas de mantenimiento**

La categoría de herramientas de mantenimiento se puede subdividir en:

- Herramientas de Ingeniería Inversa.
- Herramientas de reestructuración y análisis de código.
- Herramientas de reingeniería.

### **Herramientas de gestión de proyectos**

La mayoría de las herramientas CASE de gestión de proyectos, se centran en un elemento específico de la gestión del proyecto, en lugar de proporcionar un soporte global para la actividad de gestión. Utilizando un conjunto seleccionado de las mismas se puede: realizar estimaciones de esfuerzo, coste y duración, hacer un seguimiento continuo del proyecto, estimar la productividad y la calidad, etc. Existen también herramientas que permiten al comprador del desarrollo de un sistema, hacer un seguimiento que va desde los requisitos del pliego de condiciones técnicas inicial, hasta el trabajo de desarrollo que convierte estos requisitos en un producto final. Se incluyen dentro de las herramientas de control de proyectos las siguientes:

- Herramientas de planificación de proyectos.
- Herramientas de seguimiento de requisitos.
- Herramientas de gestión y medida.

### **Herramientas de soporte**

Se engloban en esta categoría las herramientas que recogen las actividades aplicables en todo el proceso de desarrollo, como las que se relacionan a continuación:

- Herramientas de documentación.
- Herramientas para software de sistemas.
- Herramientas de control de calidad.
- Herramientas de bases de datos.

## 3.2 Herramientas CASE utilizadas en el SICC

### 3.2.1 Rational Rose

**Rational Rose** es la herramienta CASE desarrollada por los creadores de UML (Booch, Rumbaugh y Jacobson), que cubre todo el ciclo de vida de un proyecto: concepción y formalización del modelo, construcción de los componentes, transición a los usuarios y certificación de las distintas fases y entregables.

El navegador UML de **Rational Rose** permite establecer una trazabilidad real entre el modelo (análisis y diseño) y el código ejecutable. Facilita el desarrollo de un proceso cooperativo en el que todos los agentes tienen sus propias vistas de información (vista de Casos de Uso, vista Lógica, vista de componentes y vista de Despliegue), pero utilizan un lenguaje común para comprender y comunicar la estructura y la funcionalidad del sistema en construcción.

Se eligió el uso de esta herramienta para el SICC puesto que es la que da mejor soporte al modelado con UML y porque permite el desarrollo del sistema en todas las fases del ciclo de vida.

Por otro lado, rational rose proporciona un ambiente de trabajo bastante amigable y en general, toda su estructura es bastante robusta y sencilla.

Además de todo el soporte que proporciona para UML, también hace lo propio con OMT, lo que viene a consolidarla como la herramienta idónea para el desarrollo del sistema SICC.

### 3.2.2 Erwin 4.1

Erwin es una herramienta para modelar, que ayuda a diseñar bases de datos de alto desempeño para cliente/servidor y *web/Intranet*<sup>1</sup>, así como aplicaciones de *data warehousing*.<sup>2</sup>

Erwin no solo ayuda a diseñar modelos de datos lógicos, también construye automáticamente estructuras de datos físicos con la información del diagrama

Cuando el modelo de datos esta listo para usarse, simplemente se selecciona el servidor donde se quiere construir la base de datos y se eligen las opciones de generación de esquema que se quieran incorporar. Erwin automáticamente construye la base de datos física, incluyendo todas las tablas, índices, procedimientos almacenados, *triggers* de integridad referencial y otros componentes necesarios para manejar exitosamente los datos.

Se eligió esta herramienta para realizar el modelo entidad – relación de la base de datos del SICC porque es una de las más robustas del mercado, y además porque sus características la hacen propicia para el modelado de bases de datos como la que se requiere en el sistema.

Para la elección de ambas herramientas, Rational Rose y Erwin, se tomaron en cuenta además, factores tales como: facilidad de acceso a la herramienta, facilidad de uso, experiencia, disponibilidad, confiabilidad y habilidad en su manejo, entre otras.

---

<sup>1</sup> Una intranet es un sitio web interno y privado de una empresa al que los usuarios pueden acceder mediante un navegador y que permite múltiples funcionalidades encaminadas a mejorar la eficiencia de la organización en la que se implanta.

<sup>2</sup> Un data warehouse es una colección de datos en la cual se encuentra integrada la información de la Institución y que se usa como soporte para el proceso de toma de decisiones gerenciales.

### 3.3 Planeación y Análisis de Requerimientos del SICC

Como se ha mencionado en capítulos anteriores, UML no es una metodología, tan sólo es un lenguaje que nos permite modelar objetos. En el proyecto “Sistema de Inscripción y Control de Cursos” (SICC), utilizamos este lenguaje con el fin de diseñar nuestro proyecto mediante clases. También haremos uso de la metodología OMT, la cual, nos proporcionará los pasos a seguir para la creación del SICC. El ciclo de vida que se propone para el SICC de acuerdo con esta metodología, es el siguiente:

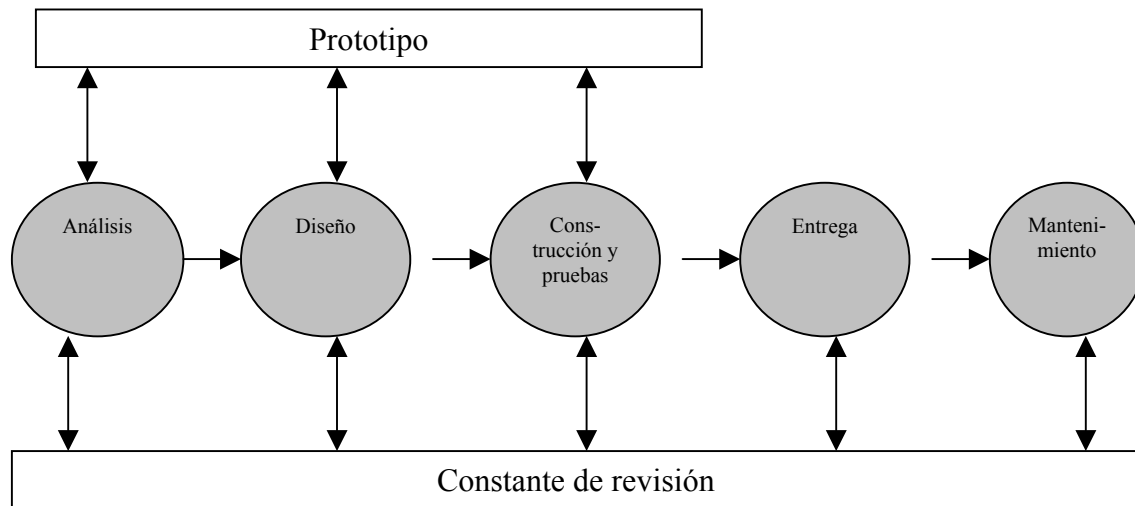


Figura 3.1. Ciclo de vida del SICC

La fase de análisis esta compuesta básicamente por el planteamiento del problema junto con la especificación de requerimientos. Ambos temas fueron tratados en el capítulo 1. Un punto importante en ésta etapa son los modelados de objetos, dinámico y funcional. Dichos modelados se harán a través de los diagramas que UML proporciona y que dan soporte a esa parte y a los cuáles nos referiremos más adelante.

La fase de Diseño, comienza con la elaboración de la arquitectura del sistema tomando como punto de partida el análisis de la especificación de requerimientos.

De acuerdo a los requerimientos proporcionados se determino el siguiente diseño:

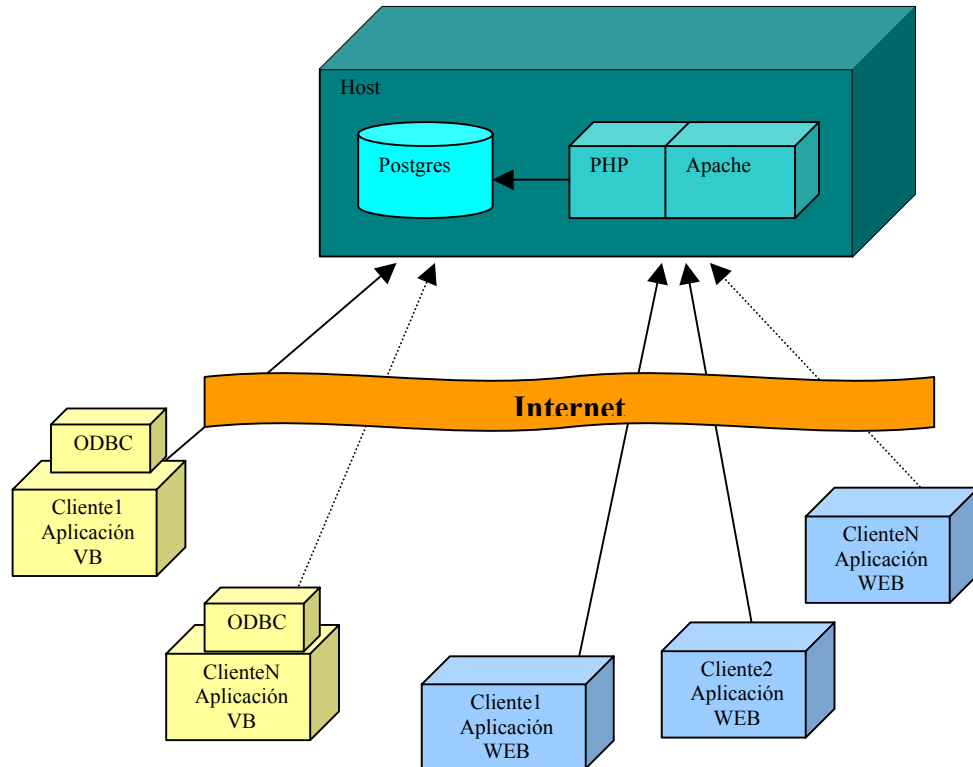


Figura 3.2 Arquitectura del SICC

Como se puede observar en la gráfica, existen básicamente dos tipos de aplicaciones que darán respuesta a los requisitos planteados en la especificación de requerimientos. La primera de ellas es un módulo hecho en visual basic con conexión a una base de datos en postgres, la cuál reside en un servidor. Dicha aplicación utiliza tecnología de Conectividad Abierta de Bases de Datos (ODBC *Open DataBase Connectivity*) para realizar el enlace con la base de datos.

La otra aplicación consiste en un módulo desarrollado en php que hará las funciones pedidas para el web. Interactúa con un servidor de apache el cuál reside en la misma máquina servidor.



En el capítulo 4 “Generación de Clases y Objetos con Visual Basic” se verá a detalle cada uno de éstos módulos y sus componentes así como las tecnologías que utilizan.

Las aplicaciones de visual basic serán las encargadas del control y administración de la información de los cursos (ver Cap.1 subtema: 1.3.1 apartados a y b).

### **3.3.1 Descripción de la tecnología utilizada**

- ODBC

Es la tecnología que permite la interacción o comunicación entre una determinada aplicación y distintos manejadores de bases de datos. Para entender mejor lo anterior pensemos en el siguiente caso. Tenemos una aplicación para acceder a las tablas de una BD (Base de datos) en Access. ¿Qué pasará si pretendemos utilizar la misma aplicación para trabajar con una BD de SQL Server u otra BD cualquiera?. Sencillamente no funcionará. Nuestra aplicación diseñada para trabajar con un gestor de BD determinado no sabrá dialogar con el otro. Esto ocurre porque los manejadores de bases de datos no funcionan igual.

La idea de ODBC es la de solucionar esta problemática. Podemos ver a ODBC como un elemento que por un lado siempre es igual y por el otro es capaz de dialogar con una BD concreta. Por lo tanto, cuando se requiere comunicarse con un motor de datos distinto, basta con cambiar dicho elemento y nuestra aplicación siempre funcionará sin importar lo que haya del otro lado, algo así como ir cambiando las boquillas de una manguera. A esas piezas intercambiables las llamaremos orígenes de datos de ODBC.

La siguiente gráfica ilustra mejor la idea expuesta en los párrafos anteriores:

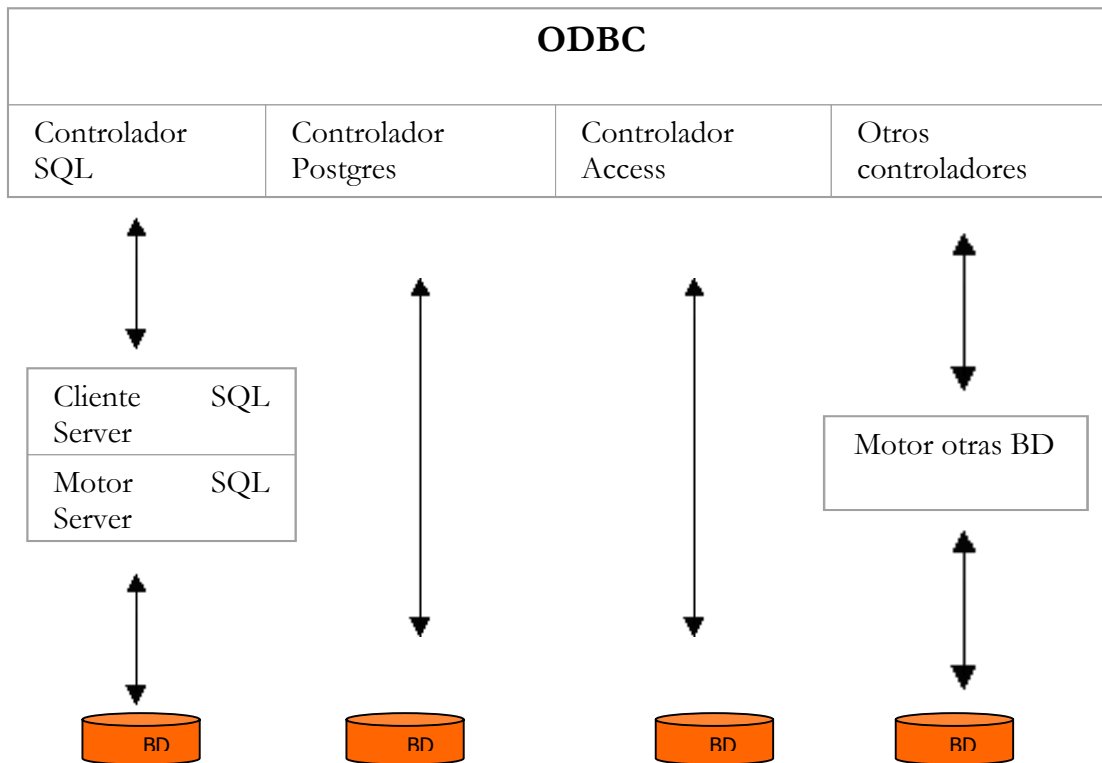


Figura 3.3. Tecnología ODBC

Por un lado el ODBC provee de unas características siempre homogéneas, y por el otro permite distintos controladores que aseguran la conectividad de la aplicación con diferentes bases de datos.

- Objetos ADO

ADO permite crear aplicaciones capaces de manipular bases de datos a través de un proveedor de vinculación e incrustación de objetos (OLE DB *Object Linking and Embedding for DataBase*). El objetivo de OLE DB es proporcionar una herramienta de nivel inferior que de acceso universal a los datos con independencia del origen de datos, ya sea un servidor de correo electrónico, una base de datos, una hoja de cálculo u otro tipo de almacenamiento de datos. Debido a la complejidad de los elementos de OLE DB, no se puede acceder a ellos directamente desde Visual Basic; para ello se utilizan los objetos ADO, que permiten acceder a prácticamente la totalidad de las funciones de OLE DB.

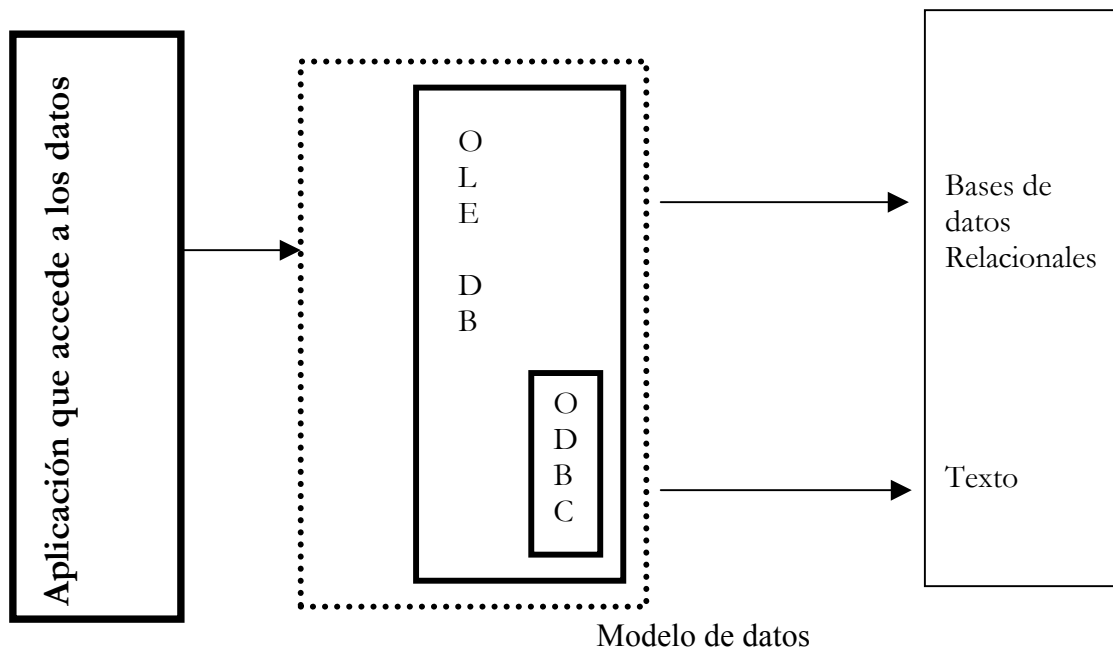


Figura 3.4 Objetos ADO

### Jerarquía de objetos ADO

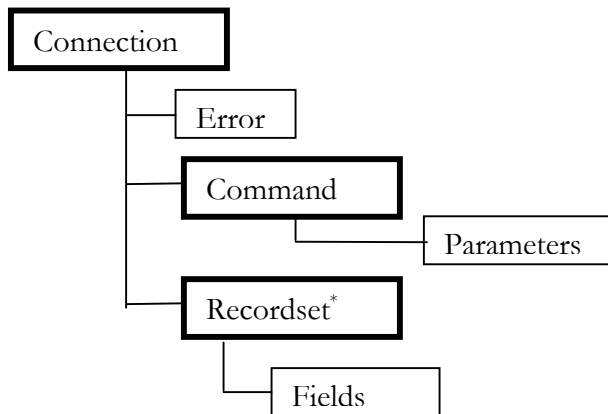


Figura 3.5. Jerarquía de Objetos ADO

Como puede verse en la figura 3.5, existen tres objetos principales dentro de ADO: El objeto Connection, el objeto Command y el objeto recordset.


El objeto **Connection** representa una sesión con el origen de los datos. A grandes rasgos, es el encargado de establecer la conexión con el servidor donde reside la base de datos. Dependiendo de la funcionalidad del proveedor de los datos se pueden utilizar determinadas propiedades, métodos y colecciones de este objeto.

Tiene asociada una colección de errores los cuáles pueden presentarse a la hora de establecer el enlace con la máquina servidor.

El objeto **Recordset** representa un conjunto de registros.

El recordset en ADO es el encargado de obtener los registros provenientes de alguna consulta SQL. Tiene asociados una colección de objetos llamados Fields que son propiamente los encargados de contener los registros resultantes de la consulta, es decir, para acceder a dichos registros tendremos que utilizar los objetos Fields del recordset en cuestión.

Un objeto **Command** es la definición de un comando específico que se piensa ejecutar contra un origen de datos. El objeto **Command** se utiliza para añadir o eliminar registros, ejecutar una operación de manejo masivo de datos o para manipular la estructura de una base de datos. Tiene una colección de parámetros asociada que permite realizar operaciones específicas sobre la base de datos.

El uso de objetos ADO para la implementación del SICC esta basado en la funcionalidad y el alcance que tiene el desarrollo del sistema con el acceso a base de datos. Es una de las tecnologías más avanzadas desarrolladas hasta el momento por Microsoft y proporciona un cúmulo  enso de posibilidades y ventajas con respecto a otras tecnologías, principalmente ofrece un modelo de objetos más simple que el utilizado por sus antecesores (DAO y RDO<sup>3</sup>), lo que se traduce en menos objetos y más propiedades, métodos y eventos.

---

<sup>3</sup> DAO (*Data Access Objects*). Utiliza el motor de bases de datos Microsoft Jet para proporcionar un conjunto de objetos de acceso a datos: objetos de base de datos, objetos de definición de tabla, objetos de definición de consulta y objetos de conjunto de registros, entre otros.

La conexión a la base de datos dentro del SICC se realizará mediante un solo objeto connection al cuál acudirán las distintas aplicaciones cuando así lo requieran.

Cada aplicación creará los recordset que necesite y los destruirá cuando lo considere necesario. Básicamente el uso de objetos recordset se limitará a consultas de sólo lectura en la base de datos. Mientrás que las operaciones de modificación, inserción, borrado o alguna otra similar se realizarán mediante objetos Command.

La aplicación web se realizará utilizando el lenguaje de programación php y un servidor apache. La razón de hacerlo en php es por la facilidad y robustez que presenta dicho lenguaje, además de que se trata de software libre que se adapta muy bien al trabajo con una base de datos en postgres el cuál también es de licencia libre.

La aplicación web se encargará de atender lo concerniente al apartado de cursos vía Internet (ver Cap.1 subtema: 1.3.2).

Finalmente, la base de datos se realizará con postgres ya que es un manejador de licencia libre que se adapta muy bien al esquema que se tiene planeado para dicha base, y permite la comunicación con visual basic y php de forma muy sencilla y cómoda.

De acuerdo a lo visto anteriormente, los requerimientos de software y hardware necesarios para llevar a cabo el sistema son los siguientes:

Computadora personal con procesador pentium III o mayor y tarjeta de red con el siguiente software:

❖ Sistema Operativo Windows 9x ó NT 4.0 / XP

---

RDO (*Remote Data Objects*). Es una interfaz orientada a objetos para acceso a datos que se comunica directamente con ODBC sin necesidad de atravesar el motor MS Jet. Es la interfaz elegida por muchos desarrolladores de bases de datos SQL Server, Oracle y otras bases de datos relacionales grandes.

- ❖ Protocolo TCP/IP
- ❖ Un visualizador WEB (Netscape Communicator 4.7 e InternetExplorer 5)
- ❖ Visual Basic 6.0
- ❖ ODBC Postgres 6.5
- ❖ Rational Rose
- ❖ Erwin

Una computadora con procesador pentium III o mayor que funcione como servidor y con el siguiente software:

- ❖ Sistema Operativo Linux (Red Hat 7.0) o superiores
- ❖ Protocolo TCP/IP
- ❖ Servidor WEB (Apache)
- ❖ Servidor PHP 4.0
- ❖ Servidor Postgres 7.0

### 3.4 Modelo Entidad-Relación

El modelo entidad –relación<sup>4</sup> definido en Erwin 4.1 consiste en un diagrama que muestra las entidades generadas a partir del análisis de los requisitos del sistema así como de la arquitectura planteada anteriormente.

La notación que aparece en el modelo es muy sencilla. Erwin distingue entre una relación fuerte (aquella que une a dos entidades pasando la llave primaria<sup>5</sup> de la entidad padre a la llave primaria del hijo.) y una débil (une a dos entidades pasando la llave primaria de la entidad padre a los atributos de la entidad hijo) utilizando para la primera, una línea continua que une a las entidades en cuestión, y una línea discontinua para representar al segundo tipo de relación. Las relaciones de uno a muchos son indicadas con un punto relleno del lado donde esta la cardinalidad muchos.

Algunas de las tablas que aparecen en el diagrama como las de instructor\_grupo ó la de grupo\_alumno se crearon para poder romper la relación de muchos a muchos existentes en el modelo.

No todas las tablas que aparecen en este modelo son utilizadas por las aplicaciones que presento en este trabajo de tesis. Dichas tablas son las siguientes: antecedente, curso\_antecedente, tipo\_personal, personal, horario\_instructor, experiencia y propuesta. Esto es porque el uso de éstas tablas corresponde a otros módulos. El resto de ellas son utilizadas

---

<sup>4</sup> Denominado por sus siglas como: E-R; Este modelo representa a la realidad a través de **entidades**, que son objetos que existen y que se distinguen de otros por sus características, por ejemplo: un alumno se distingue de otro por sus características particulares como lo es el nombre, o el número de cuenta asignado al entrar a una institución educativa, así mismo, un empleado, una materia, etc.

<sup>5</sup> Es un atributo definido como atributo principal. Es una forma única de identificar a una entidad. Por ejemplo, la CURP de un empleado se distingue de otro por que las CURP no pueden ser iguales.

en algún momento por mis aplicaciones lo cuál se detallará más adelante en los diagramas de diseño creados con UML.

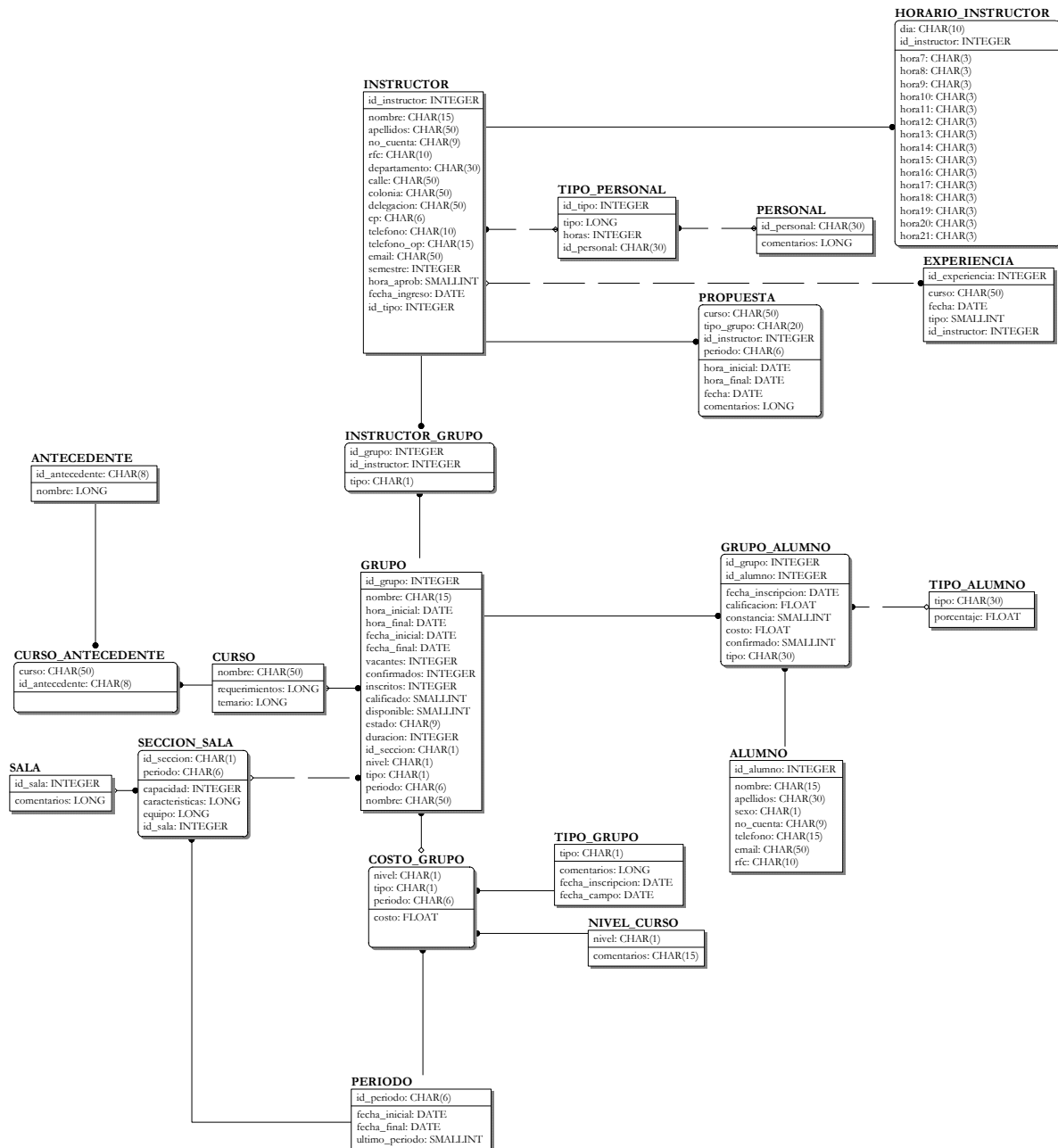


Figura 3.6. Modelo Entidad - Relación del SICC



## 3.5 Conversión del Modelo Entidad – Relación a un esquema de Base de Datos para el SICC

A partir del modelo entidad – relación mostrado en el apartado anterior, se implementó la base de datos para el sistema con las características que se mencionan a continuación.

El manejador de base de datos utilizado es PostgreSQL 7.3 y se encuentra sobre un servidor con sistema operativo Linux Red Hat 7.2. Para permitir la interacción entre la base de datos y visual basic se utiliza el driver psqLODBC para postgresQL 7.3.

Algunas características importantes de postgres son las siguientes:

Es un sistema gestor de bases de datos relacionales derivado del paquete Postgres escrito en Berkeley. Es considerado como uno de los gestores de bases de datos de código abierto más avanzado hoy en día, ofreciendo control de concurrencia multi-versión, soportando casi toda la sintaxis SQL (incluyendo subconsultas, transacciones, tipos y funciones definidas por el usuario), contando también con un amplio conjunto de enlaces con lenguajes de programación (incluyendo C, C++, Java, perl y python).

Postgres ofrece una potencia adicional sustancial al incorporar los siguientes cuatro conceptos adicionales básicos que permiten a los usuarios ampliar fácilmente las posibilidades que el sistema ofrece.

- Clases
- Herencia
- Tipos
- Funciones

Otras características que aportan potencia y flexibilidad adicional son las siguientes:

Restricciones (Constraints)<sup>6</sup>

Disparadores (triggers)<sup>7</sup>

Reglas (rules)

Integridad transaccional

Estas características colocan a Postgres en la categoría de las Bases de Datos identificadas como objeto-relacionales. Éstas son diferentes de las referidas como orientadas a objetos, que en general no son bien aprovechables para soportar lenguajes de Bases de Datos relacionales tradicionales. Postgres tiene algunas características que son propias del mundo de las bases de datos orientadas a objetos. De hecho, algunas Bases de Datos comerciales han incorporado recientemente características en las que Postgres fue pionera.

Finalmente, para interactuar con visual basic sólo se requiere instalar el driver correcto y conocer el lenguaje sql adecuadamente para establecer la comunicación con la base de datos. Por otro lado, dado que postgres se instala sobre un sistema linux, se cuenta con un nivel de seguridad bastante confiable.

### 3.5.1 Estructura de la Base de Datos

De acuerdo al análisis realizado se decidió crear distintos tipos de usuarios de la base de datos con ciertos privilegios sobre ella, mismos que aparecen en el siguiente cuadro:

---

<sup>6</sup> Líneas de código insertadas en el manejador que permiten colocar restricciones a valores en los campos de las tablas de la base de datos.

<sup>7</sup> Trigger: Fragmento de código almacenado en la base de datos asociado con una tabla específica y que realiza una determinada operación como puede ser: verificar la modificación de datos en una tabla, derivación de valores en columnas de manera automática, mantenimiento general de las tablas, etc.

Modulo	Usuario	Descripción	Tablas
Modulo3	coordsicc	Usuario que manipula toda la información del sistema (instructores, cursos, salas, alumnos, grupos, etc). Es el usuario final con mayores privilegios.	instructor horario_instructor experiencia propuesta instructor_grupo grupo curso curso_antecedente antecedente seccion_sala sala costo_grupo tipo_grupo periodo nivel_curso grupo_alumno alumno tipo_alumno personal tipo_personal
Modulo3	inscrisicc	Usuario que sólo accesa a la información de grupos y alumnos. Es un usuario final	instructor instructor_grupo grupo tipo_alumno grupo_alumno alumno costo_grupo periodo tipo_grupo
Modulo4	inscriweb	Usuario que sólo accesa a la información de grupos y alumnos. Es un usuario final.	instructor grupo costo_grupo tipo_grupo periodo grupo_alumno alumno

Tabla 3.1 Tipos de usuarios para la base de datos del SICC

La creación de las tablas se realizó conforme al procedimiento establecido por el manejador de la base de datos.

A continuación se muestra un ejemplo con la tabla de grupo. Se pueden observar las instrucciones necesarias para la asignación de campos, así como del tipo de los mismos y los valores que adquieren por default algunos de ellos. También se observan al final algunas restricciones de validación que se implementaron para algunos campos.

```
CREATE TABLE GRUPO (  
  ID_GRUPO INTEGER DEFAULT NEXTVAL('"GRUPO_ID_GRUPO_SEQ"'::TEXT) NOT  
NULL,  
  NOMBRE CHARACTER(15) NOT NULL,  
  HORA_INICIAL TIME WITHOUT TIME ZONE NOT NULL,  
  HORA_FINAL TIME WITHOUT TIME ZONE NOT NULL,  
  FECHA_INICIAL DATE NOT NULL,  
  FECHA_FINAL DATE NOT NULL,  
  VACANTES INTEGER NOT NULL,  
  CONFIRMADOS INTEGER DEFAULT 0 NOT NULL,  
  INSCRITOS INTEGER DEFAULT 0 NOT NULL,  
  ID_SECCION CHARACTER(2) NOT NULL,  
  CURSO CHARACTER(50) NOT NULL,  
  CALIFICADO BOOLEAN DEFAULT FALSE NOT NULL,  
  NIVEL CHARACTER(1) NOT NULL,  
  TIPO CHARACTER(1) NOT NULL,  
  PERIODO CHARACTER(6) NOT NULL,  
  DISPONIBLE BOOLEAN DEFAULT TRUE NOT NULL,  
  ESTADO CHARACTER(9) DEFAULT 'ABIERTO' NOT NULL,  
  DURACION INTEGER NOT NULL,  
  CONSTRAINT GRUPO_CONFIRMADOS CHECK ((CONFIRMADOS >= 0)),  
  CONSTRAINT GRUPO_ESTADO CHECK (((ESTADO = 'ABIERTO'::BPCHAR) OR  
(ESTADO = 'CERRADO'::BPCHAR)) OR (ESTADO = 'CANCELADO'::BPCHAR)),  
  CONSTRAINT GRUPO_INSCRITOS CHECK ((INSCRITOS >= 0)),  
  CONSTRAINT GRUPO_VACANTES CHECK ((VACANTES >= 0))  
);
```

Figura 3.7. Creación de la tabla Grupo en Postgres

El mismo procedimiento se siguió para todas las tablas. Algunas de ellas presentan características diferentes en cuanto al tipo de datos que contienen y las reglas de validación que requieren, pero todas siguen el mismo procedimiento para su creación.

Otra parte fundamental de la estructura de la base son los triggers y las funciones que se implementaron para realizar diferentes operaciones sobre la misma.

La misma funcionalidad de la base de datos, exigió la creación de varios triggers que dieran respuesta a situaciones específicas para mantener el buen funcionamiento de la misma.

A continuación se detalla un Trigger que se programó para actualizar el número de vacantes, número de alumnos inscritos y el número de alumnos confirmados en un grupo, una vez que un alumno es eliminado de dicho grupo.

```

DROP FUNCTION alumno_grupo_borrado ();
DROP TRIGGER decvacinsconf_grupo_alumno_borrado;

CREATE FUNCTION alumno_grupo_borrado () RETURNS "trigger"
AS 'DECLARE vac INTEGER;
      ins INTEGER;
      conf INTEGER;

      BEGIN
        /*DECREMENTA CONFIRMADOS*/
        IF old.confirmado=true THEN
          SELECT INTO conf confirmados FROM grupo WHERE
id_grupo=old.id_grupo;
          IF conf > 0 THEN
            conf=conf-1;
            UPDATE grupo SET confirmados = conf WHERE
id_grupo=old.id_grupo;
          END IF;
        END IF;

        /*DECREMENTA INSCRITOS*/
        SELECT INTO ins inscritos FROM grupo WHERE id_grupo=old.id_grupo;
        IF ins > 0 THEN
          ins=ins-1;
          UPDATE grupo SET inscritos=ins WHERE id_grupo=old.id_grupo;

          /*INCREMENTA VACANTES*/
          SELECT INTO vac vacantes FROM grupo WHERE id_grupo=old.id_grupo;
          vac=vac+1;
          UPDATE grupo SET vacantes=vac WHERE id_grupo=old.id_grupo;
        END IF;

        RETURN old;
      END; '
LANGUAGE plpgsql;

CREATE TRIGGER decvacinsconf_grupo_alumno_borrado
BEFORE DELETE ON grupo_alumno
FOR EACH ROW
EXECUTE PROCEDURE alumno_grupo_borrado ();

```

Figura 3.8. Ejemplo de un trigger implementado en la base de datos del SICC

El trigger que aparece arriba básicamente consta de dos partes. La primera de ellas que contiene a la función que realizará las operaciones necesarias para actualizar los campos, y la parte propiamente del Trigger que en sí, es la que dispara el evento y llama a la función para que se ejecute.

Observemos primero la sección de código correspondiente al Trigger y llamada de la función:

```
CREATE TRIGGER decvacinsconf_grupo_alumno_borrado
  BEFORE DELETE ON grupo_alumno
  FOR EACH ROW
  EXECUTE PROCEDURE alumno_grupo_borrado ();
```

La primer línea crea el trigger con las palabras reservadas `CREATE TRIGGER` y a continuación se coloca el nombre de dicho Trigger, en este caso, **decvacinsconf\_grupo\_alumno\_borrado**. A continuación se indica el momento en que se ejecutará el trigger. Para nuestro ejemplo, será antes de que el alumno sea borrado de la tabla **grupo\_alumno** donde ese relaciona a los alumnos con los grupos. Finalmente se invoca a la función que acompaña a nuestro trigger y que se llama **alumno\_grupo\_borrado** haciendo uso de las palabras reservadas `EXECUTE PROCEDURE`.

La siguiente sección de código es la de la función asociada al trigger y que a continuación se explica:

```
CREATE FUNCTION alumno_grupo_borrado () RETURNS "trigger"
  AS 'DECLARE vac INTEGER;
      ins INTEGER;
      conf INTEGER;

  BEGIN
    /*DECREMENTA CONFIRMADOS*/
    IF old.confirmado=true THEN
      SELECT INTO conf confirmados FROM grupo WHERE
id_grupo=old.id_grupo;
      IF conf > 0 THEN
        conf=conf-1;
        UPDATE grupo SET confirmados = conf WHERE
```

```
id_grupo=old.id_grupo;
    END IF;
END IF;

/*DECREMENTA INSCRITOS*/
SELECT INTO ins inscritos FROM grupo WHERE id_grupo=old.id_grupo;
IF ins > 0 THEN
    ins=ins-1;
    UPDATE grupo SET inscritos=ins WHERE id_grupo=old.id_grupo;

    /*INCREMENTA VACANTES*/
    SELECT INTO vac vacantes FROM grupo WHERE id_grupo=old.id_grupo;
    vac=vac+1;
    UPDATE grupo SET vacantes=vac WHERE id_grupo=old.id_grupo;
END IF;

RETURN old;
END; '
LANGUAGE plpgsql;
```

En la parte superior se coloca el nombre de la función con las palabras reservadas `CREATE FUNCTION`. También se puede observar la declaración de tres variables: **vac**, **ins** y **conf**, todas ellas del tipo entero.

A continuación aparece la parte correspondiente al decremento del campo confirmados. Se observa que se hace una consulta del campo **confirmados** de la tabla **grupo**. A continuación se valida si dicho campo es mayor a cero en cuyo caso lo decrementa en uno. Después hace la actualización correspondiente en la tabla **grupo**. Cabe señalar que para realizar las operaciones antes descritas se hace uso de una de las variables previamente declaradas (**conf**), la cuál sirve para almacenar temporalmente el número que devuelve la consulta realizada del campo **confirmados**.

Algo similar ocurre con la parte de *DECREMENTA INSCRITOS* sólo que ahora se modifica el campo **inscritos** de la tabla **grupo**.

En la parte de *INCREMENTA VACANTES* se realizan prácticamente los mismos pasos, sólo que ahora se modifica el campo vacantes de la tabla grupo y la operación es de incremento en vez de decremento como en los dos casos anteriores.

Finalmente se indica que el lenguaje que se está utilizando en la función es pgsql, el lenguaje por default de postgres, con la línea `LANGUAGE plpgsql;`.

Cabe señalar que al principio del trigger aparecen dos líneas particularmente diferentes al resto aquí descritas:

```
DROP FUNCTION alumno_grupo_borrado ();  
DROP TRIGGER decvacinsconf_grupo_alumno_borrado;
```

Estas dos líneas sólo sirven para limpiar el trigger y su función asociada cada vez que se mande llamar la ejecución del trigger. Esto es porque en postgres los triggers no se sobrescriben cada vez que se ejecutan.



## 3.6 Diseño e implementación del SICC mediante OMT y UML.

Como se mencionó en el capítulo 2, OMT emplea tres modelos para describir el sistema: el modelo de objetos que describe la estructura estática de los objetos del sistema y también sus relaciones, el modelo dinámico que describe los aspectos del sistema que van cambiando con el tiempo y el modelo funcional que describe las transformaciones de valores de datos que ocurren dentro del sistema.

A continuación se describen los tres modelos que se desarrollaron para el SICC. El primero de ellos es el modelo de objetos que como se dijo en capítulos anteriores, es el más importante porque es necesario para describir **qué** está cambiando o transformándose antes de describir **cuándo** o **cómo**.

### 3.6.1 Modelo de objetos

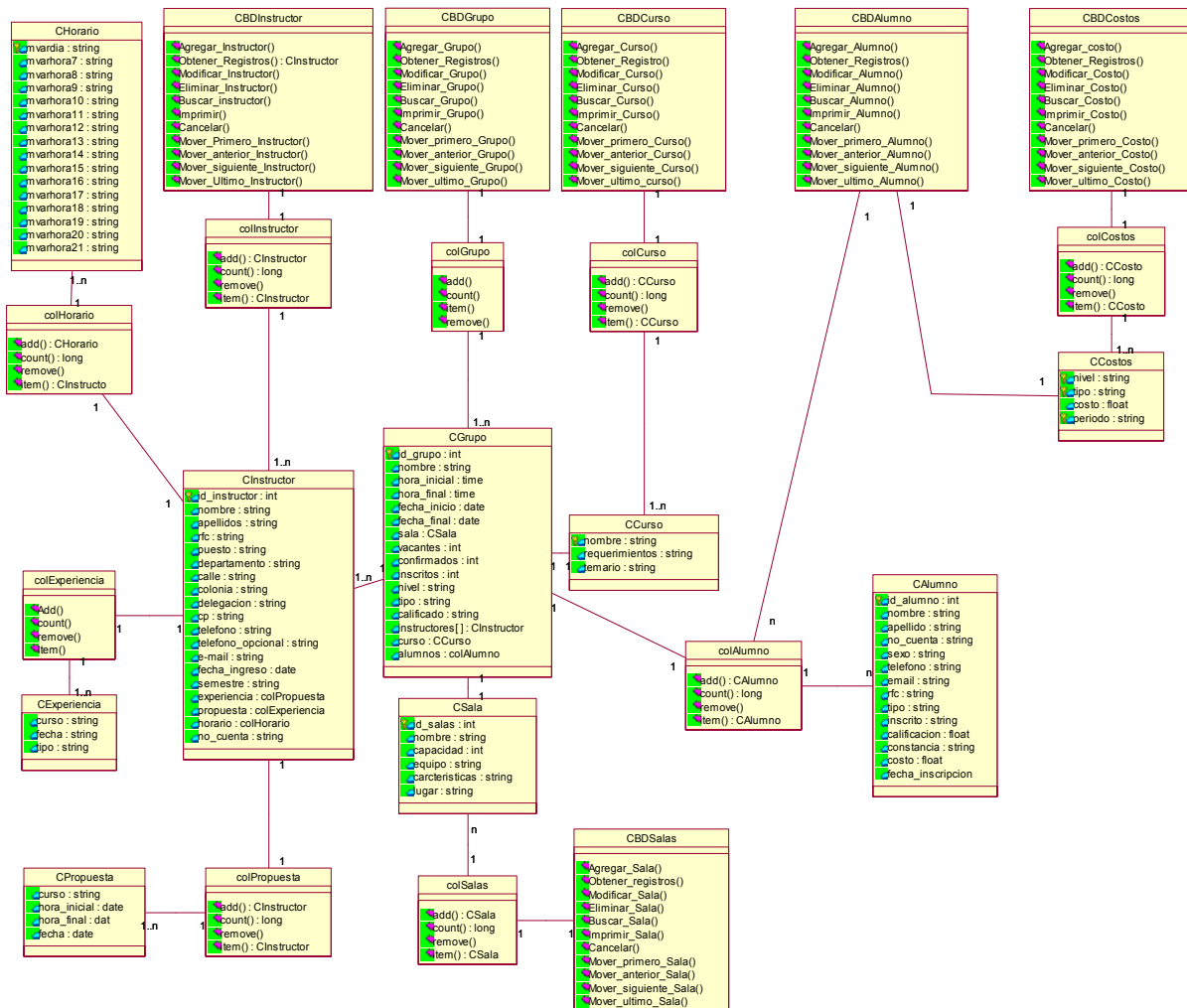


Figura 3.9. Diagrama de Clases del SICC

En el diagrama se pueden observar las clases diseñadas para el SICC así como sus relaciones y la cardinalidad de las mismas.

Las clases que tienen su nombre precedido por las siglas CBD (Clase de Base de Datos) como por ejemplo la CBDInstructor, CBDCurso, etc., son particularmente especiales. Dichas clases contienen todas las operaciones necesarias que se deben realizar sobre la base de datos (consultas, inserción, actualización, borrado, etc.). Esto fue una decisión de diseño que se implementó para tener un mejor control sobre la base de datos y utilizar al máximo todas las ventajas que ofrecen los objetos ADO para manejo de base de datos en Visual Basic.

Con dichas operaciones por separado, las clases restantes como Calumno, Cgrupo, etc., sólo contienen los atributos que caracterizan a los objetos de dicha clase y cada vez que se requiere hacer alguna operación sobre la Base de Datos sólo se invoca dicho método declarando un objeto de la clase correspondiente. Por ejemplo, si se requiere agregar un nuevo grupo en la base de datos, sólo se crea un objeto de la clase CBDGrupo y se invoca al método Agregar\_Grupo de dicho objeto. Evidentemente, el método Agregar\_Grupo utiliza para su operación un objeto de la clase Cgrupo que es en sí el registro que se agrega en la base de datos. Este tipo de procedimientos y funciones se verán más a detalle en el tema Creación de Procedimientos y Funciones del siguiente capítulo.

Existe otro tipo especial de clases que aparecen en éste diagrama. Son las clases cuyo nombre viene precedido por las siglas col (colección) como lo son: colInstructor, colGrupo, colCostos, etc. Se trata de un tipo de clase implementada como contenedor cuya finalidad es la de almacenar sólo objetos de la clase a partir de la cuál esta definida. En este caso, se declararon colecciones para las clases que de alguna forma requieren presentar un conjunto determinado de elementos u objetos asociados de una clase determinada. Por ejemplo, pensemos en un grupo que tiene un número determinado de alumnos inscritos. Pensando en objetos, de alguna forma el objeto grupo deberá tener asociado un conjunto de objetos alumno, para lo cuál se asocia al objeto grupo un objeto colección (colAlumno) que almacena a los alumnos en cuestión. Para este mismo ejemplo, se puede observar la cardinalidad de las

clases involucradas. Los objetos de la clase Cgrupo tienen una colección de alumnos (colAlumno) asociada. Mientras que cada objeto de la clase colAlumno tiene asociados uno o muchos alumnos (objetos de la clase CAlumno).

Lo mismo sucede para las colecciones colInstructor, colCurso, colCostos, colSalas, colPropuesta, colExperiencia, colGrupo y colHorario.

### **3.6.2 Modelo Dinámico**

El modelo dinámico elaborado para el SICC consta de un diagrama de estados en el que se muestra el comportamiento que tiene el sistema a través del tiempo. Sólo se muestran aquellos estados que de alguna forma tienen que ver con los módulos de que trata ésta tesis, por lo tanto, aparecen otros que sólo se mencionan pero que no se detallan. La siguiente figura muestra el diagrama de forma general y sólo se muestran los estados con sus respectivas transiciones. En las figuras subsecuentes se muestra el detalle de algunos de esos estados (aquellos referentes a éste trabajo).

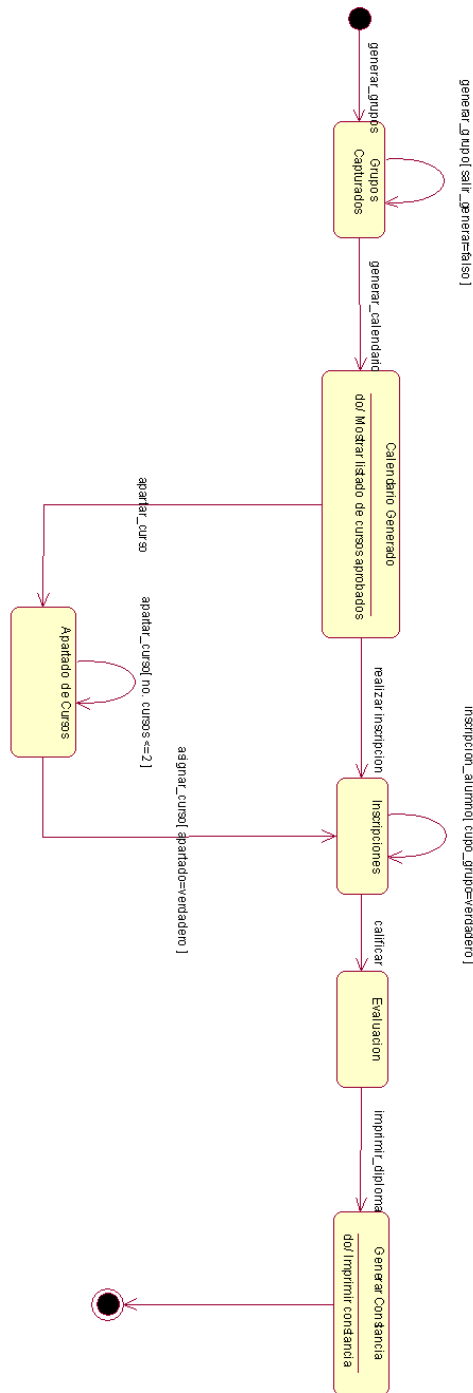


Figura 3.10. Diagrama de Estados General del SICC

En la figura 3.10 se observa el comportamiento dinámico del SICC a través de su diagrama de estados. En dicho diagrama aparece descrito el proceso de inscripción a cursos desde que se crea un grupo hasta la evaluación del mismo.

Se parte de un estado inicial en el cuál no se tiene ningún grupo creado. Por lo tanto, se procede al estado **Grupos Capturados** en donde se realizan todas las operaciones necesarias para crear un grupo. Dado que el proceso de crear grupos es algo extenso, posee su propio diagrama de estados el cuál se muestra en la fig. 3.11.

Cabe señalar que del estado inicial al de **Grupos Creados** se llega a través de la transición denominada `generar_grupos` y corresponde al evento en el cuál el usuario decide crear un grupo. Como es de suponer, el usuario puede quedarse en este estado de creación de grupos hasta que lo considere pertinente, para lo cuál se tiene una transición hacia sí mismo denominada `generar_grupo` con una condición de guarda llamada `salir_generar`. Si dicha condición es falsa indica que el usuario aún no desea salir y por lo tanto se quedará en este estado creando grupos hasta que decida proseguir al siguiente estado, momento en el cuál `salir_generar` tomará el valor de verdadero y permitirá salir de éste estado.

El siguiente estado es el de **Calendario Generado** al cuál se accede a través de la transición `generar_calendario` que ocurre cuando el usuario desea ver el listado de todos los cursos aprobados para un determinado período. Dicho estado tiene una acción llamada “Mostrar listado de cursos aprobados”. **Calendario Generado** no se detalla porque no posee un funcionamiento significativo que figure dentro del modelado dinámico del sistema. Para salir de él se tienen dos caminos, el primero de ellos conduce al estado de **Inscripciones** y se llega a él a través de la transición denominada `realizar_inscripción`; el otro conduce al estado **Apartado de cursos** al cuál se accede a través de la transición `apartar_curso`. Ambos estados figuran dentro del modelado dinámico como partes fundamentales. Por ello se detallan en las figuras 3.12 y 3.13 respectivamente. Por lo pronto se puede mencionar que para continuar en el estado de **Inscripciones** la condición de guarda `cupos_grupo` debe permanecer en verdadero para indicar que el grupo al cual se está inscribiendo alumnos aún tiene cupo para ello. Por

otro lado, para permanecer en el estado de **Apartado de Cursos** se requiere que el número de cursos apartados no sea mayor de 2 y esto corresponde con los requerimientos establecidos al principio del diseño del sistema.

También se puede observar que del estado **Apartado de Cursos** se llega al de **Inscripciones** a través de la transición `asignar_curso` que lleva la condición de guarda “`apartado`” con el valor de verdadero.

El siguiente estado es el de **Evaluación** al cuál se llega con la transición `calificar`. Las operaciones que realiza dicho estado no corresponden con los módulos que se abordan en este trabajo de tesis por lo que sólo se menciona pero no se detalla. A grandes rasgos, sólo involucra las operaciones necesarias para la evaluación de los alumnos inscritos a un determinado curso. De éste estado se pasa al de **Generar Constancia** a través del evento `imprimir_diploma`. Dicho estado no requiere mayor explicación. Sólo se trata de las operaciones necesarias para imprimir diplomas para los alumnos que hayan aprobado un curso determinado. Sin embargo, si se muestra con mayor detalle en el modelado funcional que aparecerá más adelante.

Finalmente se observa, que después de **Generar Constancia** se llega al estado final del diagrama, con lo cual termina este primer acercamiento al modelado dinámico del sistema. Lo siguiente es observar a detalle que ocurre con algunos estados comenzando por el de **Grupos Capturados**. La figura 3.11 ilustra lo anterior.

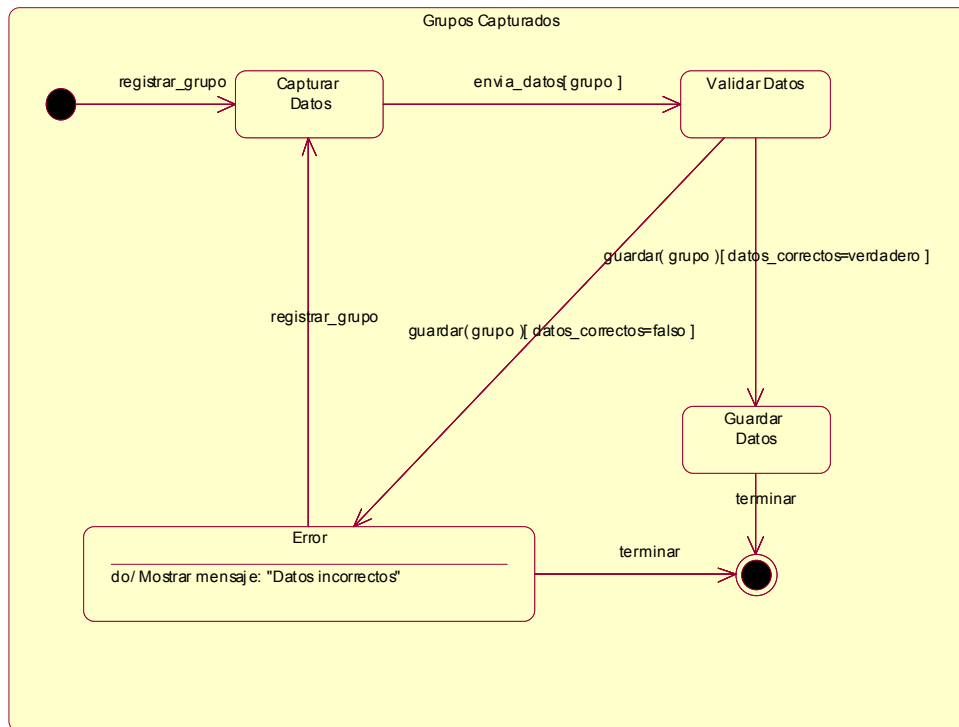


Figura 3.11. Subestados para el estado Grupos Capturados

Básicamente, el diagrama muestra las operaciones necesarias durante el proceso del registro de grupos. Comienza en un estado inicial a partir del cuál podemos llegar al estado **Capturar Datos** a través de la transición registrar\_grupo. En éste estado se capturan los datos referentes al grupo tales como: nombre, tipo de grupo, sala, instructores asignados, duración, fecha de inicio y terminación, horario y costo. Después de la captura de datos, se procede a la validación de los mismos en el estado **Validar Datos**. Aquí se realiza una verificación de la integridad de los datos es decir, se checa que hayan sido llenados correctamente y que no exista duplicidad de los mismos. Si la información fue correcta se procede a guardarla en la base de datos en el estado **Guardar Datos** y después termina el proceso. Si hubo algún error, se genera un mensaje y se regresa al estado de **Capturar Datos** o se puede terminar ahí el proceso.

El estado Inscripciones se muestra en la siguiente figura.



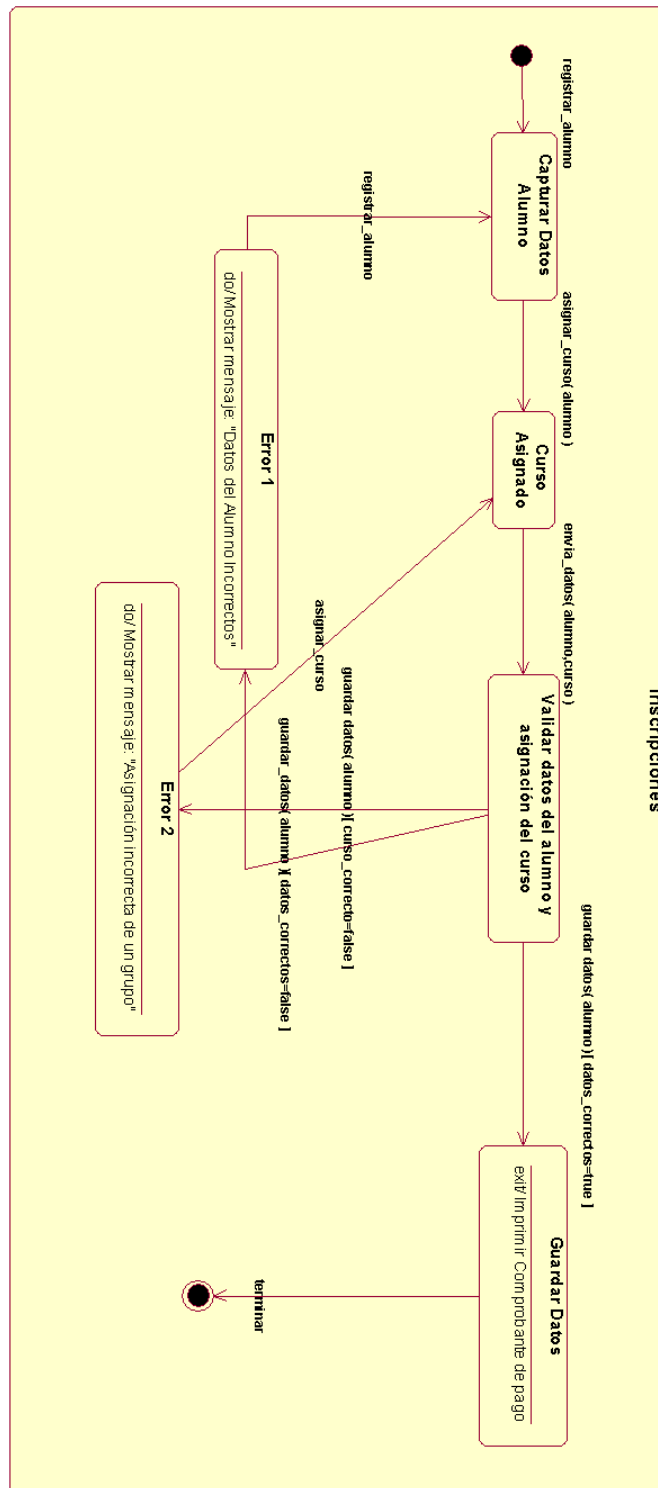


Figura 3.12. Subestados para el estado Inscripciones

Se trata de las operaciones involucradas en el proceso de inscripción de un alumno.

Del estado inicial pasamos a **Capturar Datos** . Aquí se obtienen los datos personales del alumno como son: nombre, rfc, no. de cuenta, teléfono, sexo y correo electrónico. A continuación se llega al estado **Curso Asignado** a través de la transición `asignar_curso` la cual lleva como argumento al alumno previamente registrado. En esta parte se le asigna un curso al alumno y se envía su información junto con el curso asignado al estado **Validar datos del alumno y asignación del curso**. En esta parte se validan los datos personales del alumno así como que se le haya asignado un curso correctamente. Si alguna de estas dos operaciones fue incorrecta, se procede al estado de error correspondiente. Si todo fue correcto se procede a guardar la información en la base de datos en el estado **Guardar Datos**, que a su vez realiza una acción al salir de él: “Imprimir comprobante de pago”, lo que indica el fin del proceso de inscripción.

Aquí cabría señalar que al estado de **Inscripción** se puede llegar de dos formas diferentes, como se mencionó anteriormente durante el análisis del diagrama de estados general del sistema. La primera de ellas es directamente a partir del estado **Calendario Generado** a través del evento `realizar_inscripción`, o también se puede acceder a él a través del estado **Apartado de cursos** a través del evento `asignar_curso` con la condición de guarda “`apartado`” con el valor de verdadero.



En la figura 3.13 se observan los subestados correspondientes al estado **Apartado de Cursos** de la aplicación web del sistema. Del estado inicial se tienen tres diferentes destinos. El primero de ellos es **Capturar Usuario** que ocurre cuando se ejecuta el evento `ingresar_usuario` que lleva como parámetros el nombre y el rfc del usuario. Esta acción se efectúa cuando un usuario ya se ha registrado anteriormente, es decir, ya ha realizado un apartado de cursos previamente. Después se valida al usuario y si los datos son correctos se procede al estado **Usuario Registrado** que tiene como acción de salida “Imprimir Recibo de Apartado”, que en realidad es una vista preliminar del recibo. Después se procede a su impresión y con ello termina esta operación.

Si el usuario no está registrado se manda el respectivo mensaje de error.

Otro camino que se puede tomar a partir del estado inicial es el que lleva directamente al estado de **Error3**. Para poder explicar dicho estado se definieron primero las variables que aparecerán a lo largo del diagrama y que se muestran en el siguiente cuadro:

Variable	Nombre	Descripción
fc	Fecha campo	Esta variable indica la fecha a partir de la cual los grupos pueden ser visualizados ya sea vía internet o alguna otra para su difusión. Sin embargo, no indica que la inscripción a dichos grupos se encuentre habilitada.
fa	Fecha actual	Indica la fecha actual en la que se encuentre el usuario.
ffinal	Fecha final	Indica la fecha de terminación de algún curso.
finscrip	Fecha de inscripción	Indica la fecha a partir de la cual se pueden iniciar las inscripciones a un determinado curso.
finicio	Fecha de inicio	Indica la fecha de inicio de algún curso
No. cursos	Número de cursos apartados	Indica el número de cursos apartados por el usuario.

Tabla 3.2. Variables involucradas en el diagrama de Apartado de Cursos

Para llegar al estado de **Error3** se deben cumplir cualquiera de las siguientes condiciones:

Que la fecha campo (fc) sea mayor que la fecha actual (fa), lo cuál indicaría que todavía no se pueden visualizar los cursos.

Que la fecha final de algún curso sea menor que la fecha actual lo que indicaría que los cursos ya terminaron.

Si ocurre alguna de las restricciones anteriores se llega a **Error3** el cual tiene una acción que es la de mostrar el mensaje: “No hay grupos disponibles”.

El último camino que se puede tomar a partir del estado inicial es el que lleva a **Tipos de Grupo** con la transición `visualizar_tipos_grupo` la cual a su vez valida que la fecha de campo sea mayor a la fecha actual. En éste estado se muestran los tipos de Grupo existentes para el período de cursos actual. También tiene un evento llamado `borrar_alumnos_sin_confirmar`. Dicho evento ocurre tres días hábiles antes de que comiencen los cursos y su función es la de borrar a los alumnos que no hayan confirmado su asistencia a un curso, es decir, que sólo realizaron el proceso de apartado pero no el de confirmación el cuál se realiza una vez que se realizó el pago correspondiente.

Una vez seleccionado el tipo de grupo se procede al estado **Mostrar Calendario de Cursos** donde se visualizan los cursos publicados para ese tipo de grupo. Se puede observar que a partir de éste estado se tienen 7 posibles caminos por donde continuar de acuerdo al cumplimiento de ciertas condiciones. Estas posibilidades se muestran en el siguiente cuadro:

Estado Presente	Condiciones	Descripción	Estado Siguiete
Mostrar Calendario de Cursos	no_cursos>2	Ocurre cuando el número de cursos apartados es mayor a 2 (el número máximo de cursos apartados es 2).	Error 2
	fa < finscrip .	Ocurre cuando la fecha actual es menor a la fecha de inscripciones y por ende, se pueden estar visualizando los cursos cuantas veces se desee.	Mostrar Calendario de Cursos
	fa<(finicio-3) y fa >= finscrip	Se puede considerar a éste como el estado normal. Ocurre cuando la fecha actual es menor a la fecha de inicio menos tres días y cuando es mayor a la fecha de inscripción. Lo que sucede entonces, es que se pueden empezar a registrar los alumnos para apartar un curso.	Captura Información
	fa > finscrip y fa>(finicio-3)	Cuando la fecha actual es mayor a la fecha de inscripción y mayor a la fecha de inicio de un curso menos tres días, las inscripciones sólo se pueden realizar en el Edif. Principal (Administración de Unica).	Inscripciones sólo en el Edif. Principal
	fa > finscrip y fa<(finicio-3) y curso=cerrado/cancelado	Este caso ocurre cuando las condiciones para apartar un curso son normales (fa > finscrip y fa<(finicio-3)), pero el curso esta cancelado o cerrado.	Calendario Cursos Cancelados

	$fa > finscrip$ y $fa > finicio$	Cuando la fecha actual es mayor a la fecha de inscripción y es mayor también a la fecha de inicio del curso, se pasa al estado de Cursos comenzados. Entonces, la inscripción sólo puede hacerla la Administración de Unica.	Calendario Cursos Comenzados
	$ffinal < fa$	Si la fecha de terminación del curso es menor a la fecha actual, indica que el curso ya terminó, por lo tanto se muestra el calendario de cursos pero sin aquellos que ya hayan finalizado.	Calendario sin cursos terminados

Tabla 3.3. Posibles caminos para el estado Mostrar Calendario de Curso.

Una de las posibilidades mostradas en el cuadro anterior corresponde al estado **Captura Información**. Dicho estado permite continuar con las actividades del proceso de apartado. Lo primero es que en **Captura Información**, se obtienen los datos personales del alumno, los cuáles después son enviados y analizados en **Validar Datos**. Si dicha información fue correcta se procede a almacenarla en la base de datos e Imprimir el recibo de apartado correspondiente. De no ser así se genera el mensaje de error correspondiente. Cabe señalar, que dicho recibo de apartado debe ser confirmado durante el proceso de inscripción que se mencionó anteriormente en la fig. 3.12 y se logra llegando directamente al estado **Curso Asignado**.

### 3.6.3 Modelado Funcional

El modelado Funcional del SICC esta compuesto por una serie de diagramas de actividades de UML (equivalentes a los diagramas de flujo), que muestran como son las transformaciones de valores que ocurren dentro del sistema. Para su mejor entendimiento se han dividido en subdiagramas que muestran a detalle los procesos involucrados en alguna actividad en particular. El primer diagrama que se muestra, contiene las actividades correspondientes a la creación e inscripción de un alumno a un determinado curso. A grandes rasgos podemos pensar en éstas, como las principales que competen en la realización del presente trabajo. Más adelante se detallan algunas de ellas en los diagramas posteriores.

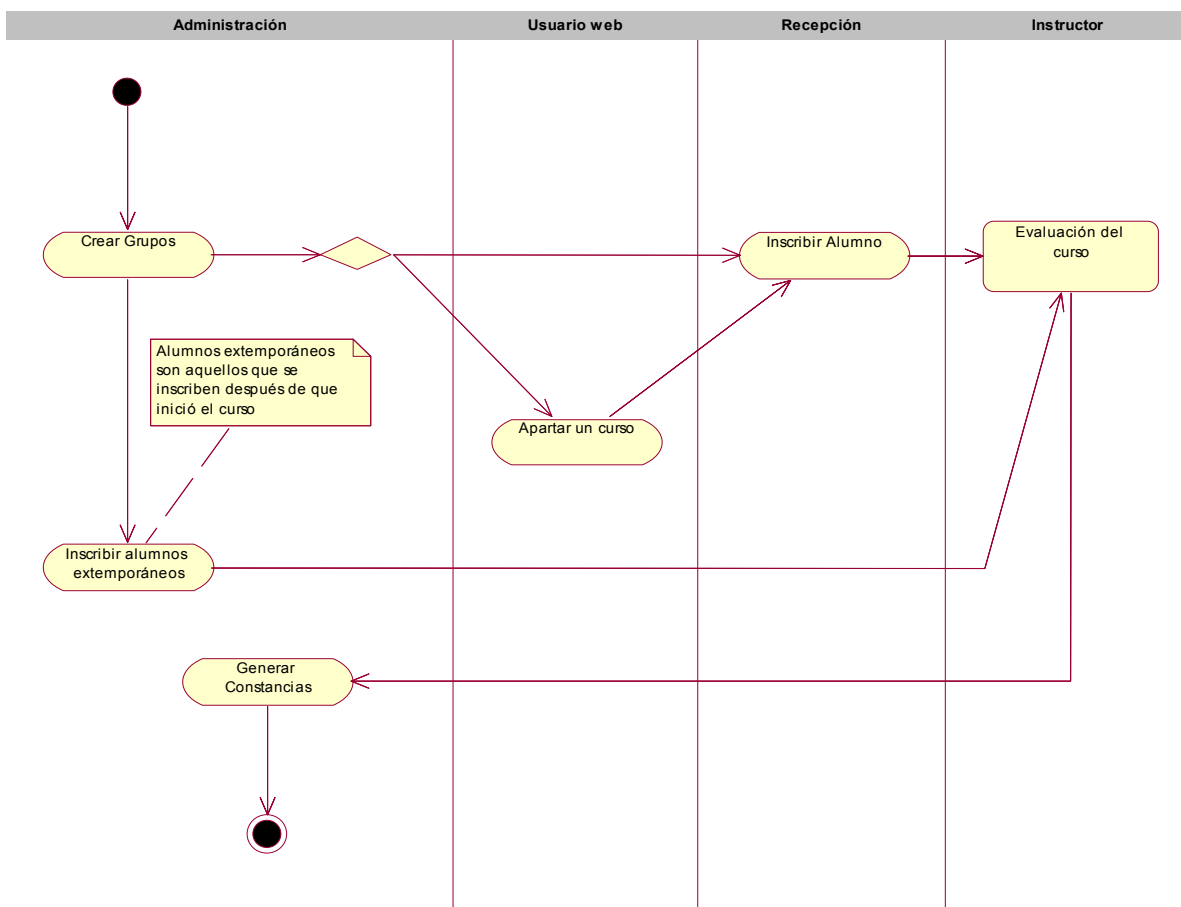


Figura 3.14. diagrama de Actividades General del SICC.



En la figura 3.14 observamos las actividades relacionadas con la inscripción de un alumno a un grupo empezando por la creación de éste, pasando por el apartado (vía web), la inscripción, evaluación del alumno y finalmente la expedición de su constancia o diploma.

Como se puede apreciar, las actividades están clasificadas de acuerdo al actor que las lleva a cabo. En éste caso, aparecen los siguientes actores que son los encargados de realizar dichas operaciones:

Actor	Descripción
Administración	Se refiere a la persona encargada de llevar la Administración en UNICA.
Usuario web	Es cualquier persona que pueda tener acceso a una pc con Internet.
Recepción	Se refiere a la persona encargada de inscribir a los alumnos en las instalaciones del Edif. Principal de la Facultad de Ingeniería.
Instructor	Es la persona encargada de impartir algún curso determinado.

Tabla 3.4. Actores involucrados en el Modelado Funcional del SICC.

En primer instancia, observamos que la creación de grupos corresponde única y exclusivamente a la administración de UNICA. Los procesos relacionados con dicha actividad se detallan en la figura 3.15.

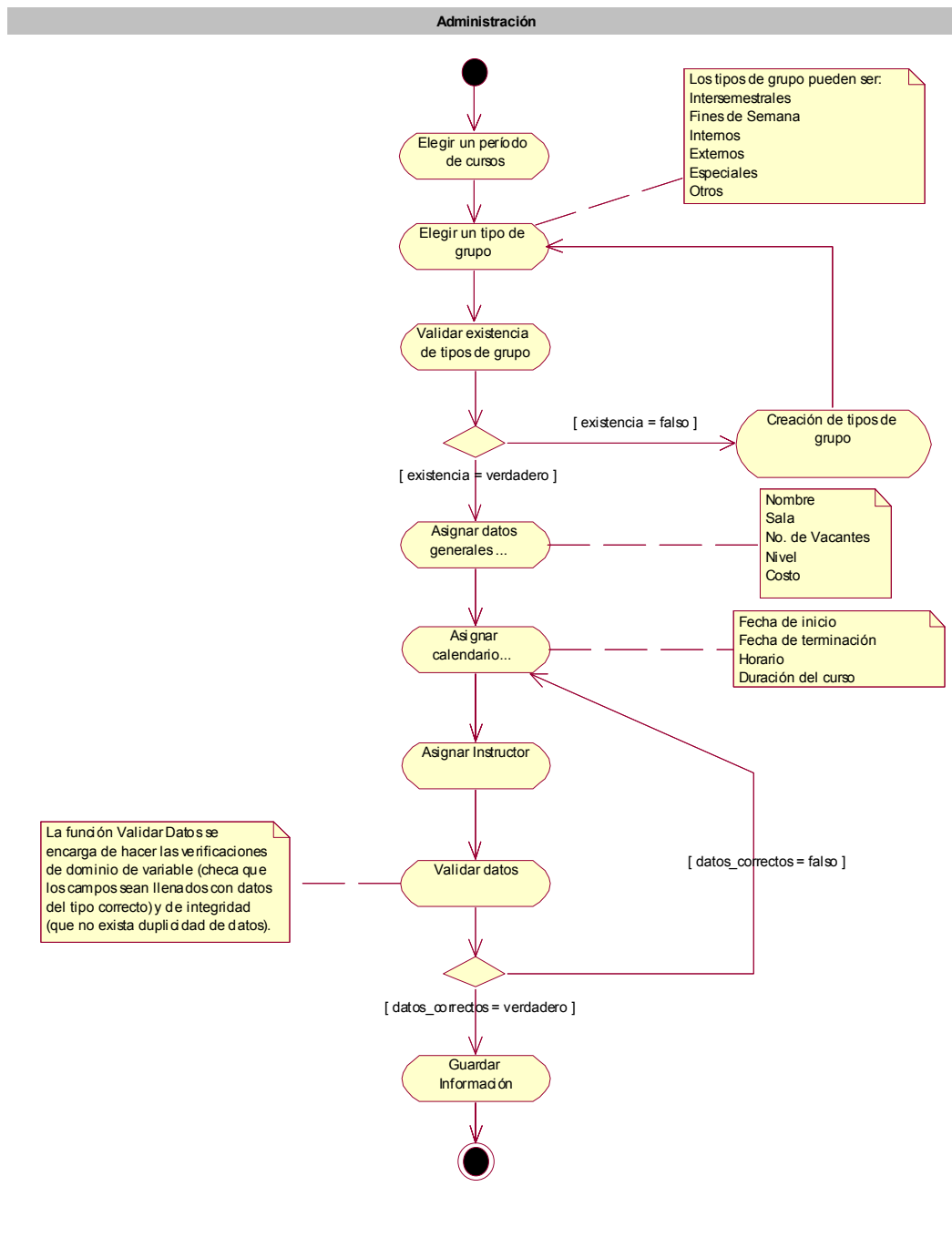


Figura 3.15. Diagrama de Actividades para la creación de grupos.

El primer paso consiste en elegir un período de cursos. Posteriormente se elige un tipo de grupo. Si no existen tipos de grupo, entonces se crean y se continúa con el flujo del proceso. Con el período y el tipo de grupo elegidos, se procede a la asignación de los datos

generales del grupo y su respectivo calendario (fechas de inicio y terminación, horario y duración del curso). Posteriormente se asigna un instructor, se valida que la información anterior sea correcta y se procede a guardarla en la base de datos.

Una vez creado el grupo se puede realizar un apartado vía web del mismo. Esto lo realiza el Usuario web. El detalle de dicha actividad se observa en la figura 3.16.

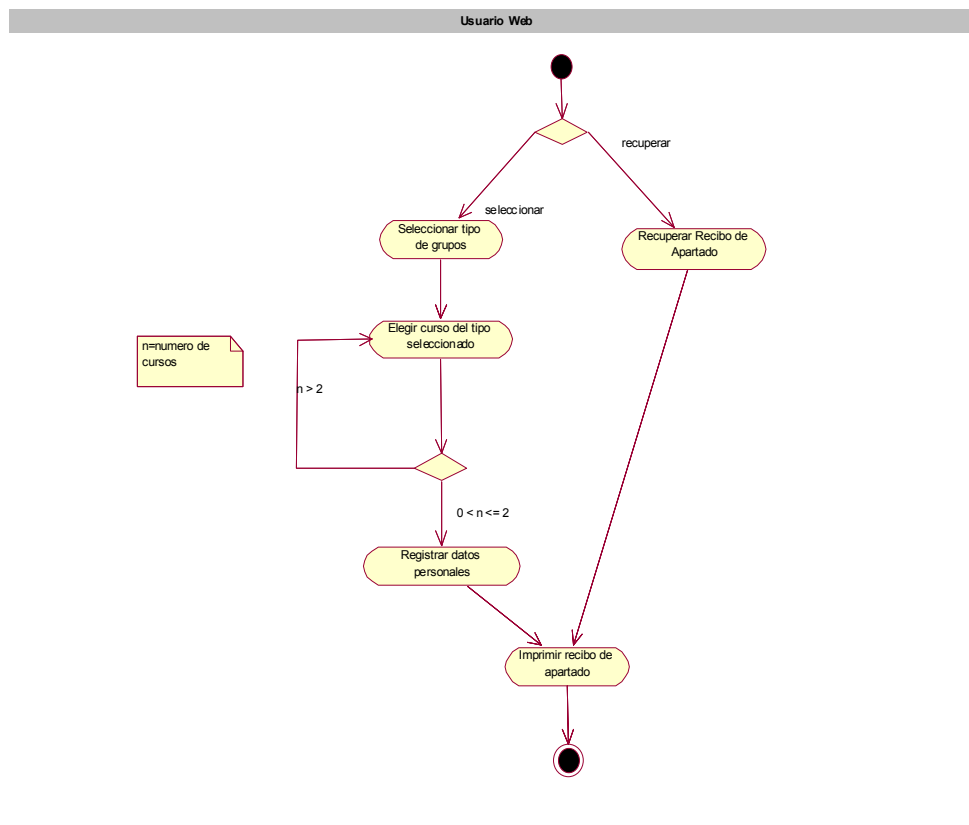


Figura 3.16. Diagrama de actividades para el apartado de un curso.

En el diagrama se observan dos caminos. El más sencillo de ellos ocurre cuando ya se ha apartado un curso y sólo se desea recuperar el recibo de apartado. El otro camino nos lleva del estado inicial a la actividad de **Seleccionar tipo de grupos**, de aquí proseguimos a elegir el curso que queremos apartar. Se pueden elegir hasta dos grupos según las especificaciones del sistema. Después se registran los datos personales y se procede a imprimir el recibo de apartado.

A la actividad de **Inscribir Alumno** se puede acceder ya sea a través del apartado o directamente. Esto es, porque un alumno puede apartar un curso vía web y después presentarse a la confirmación del mismo (hasta tres días antes del inicio de dicho curso), o puede llegar directamente a inscribirse al curso. En cualquiera de las dos situaciones anteriores, la inscripción es realizada por la persona encargada de la Recepción. Dicho proceso se detalla en la figura 3.17.

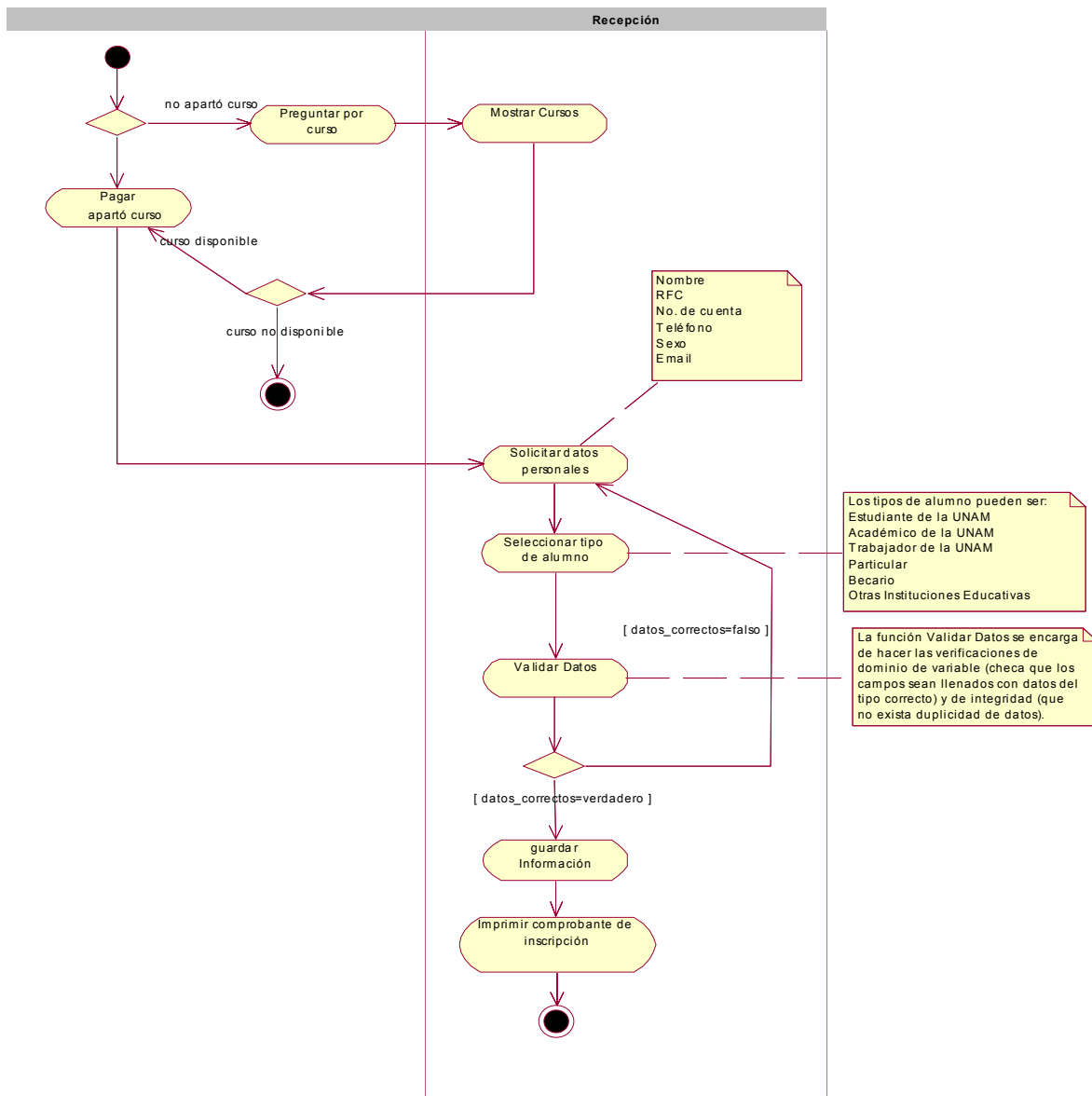


Figura 3.17. diagrama de actividades para la inscripción de alumnos

El proceso de inscripción comienza verificando si el alumno realizó un apartado previo o no. Si fue así, la actividad inmediata es la de pagar en el lugar correspondiente el costo del curso. A continuación se le piden sus datos personales, se verifica de que tipo de alumno se trata y se procede a validar dicha información. Si no hubo ningún inconveniente se guarda la información en la base de datos y finalmente se imprime el comprobante de inscripción correspondiente.

En el caso de que no haya apartado un curso, lo primero que realiza es preguntar en la recepción de UNICA por el curso que le interesa. Si esta disponible, se procede a la actividad de pagar su costo y se continúa con el flujo antes descrito. Si no esta disponible, termina el proceso de inscripción pudiendo claro, elegir algún otro curso.

Una actividad especial que realiza la administración es la de **Inscribir alumnos extemporáneos**. Esta situación ocurre cuando alguna persona desea inscribirse a un curso determinado una vez iniciado el curso. En este caso, sólo la persona encargada de la administración de UNICA puede decidir si realiza la inscripción o no. Evidentemente, cuando se trata de alumnos extemporáneos no se puede hablar de apartado por las razones que se exhibieron anteriormente.

Posteriormente se tiene la etapa de Evaluación. Los procesos relacionados con ésta actividad no incumben al trabajo que aquí se expone por lo que sólo se mencionarán.

Una vez evaluado el curso se procede a la expedición de constancias. Esto se aprecia mejor en el diagrama de la figura 3.18.

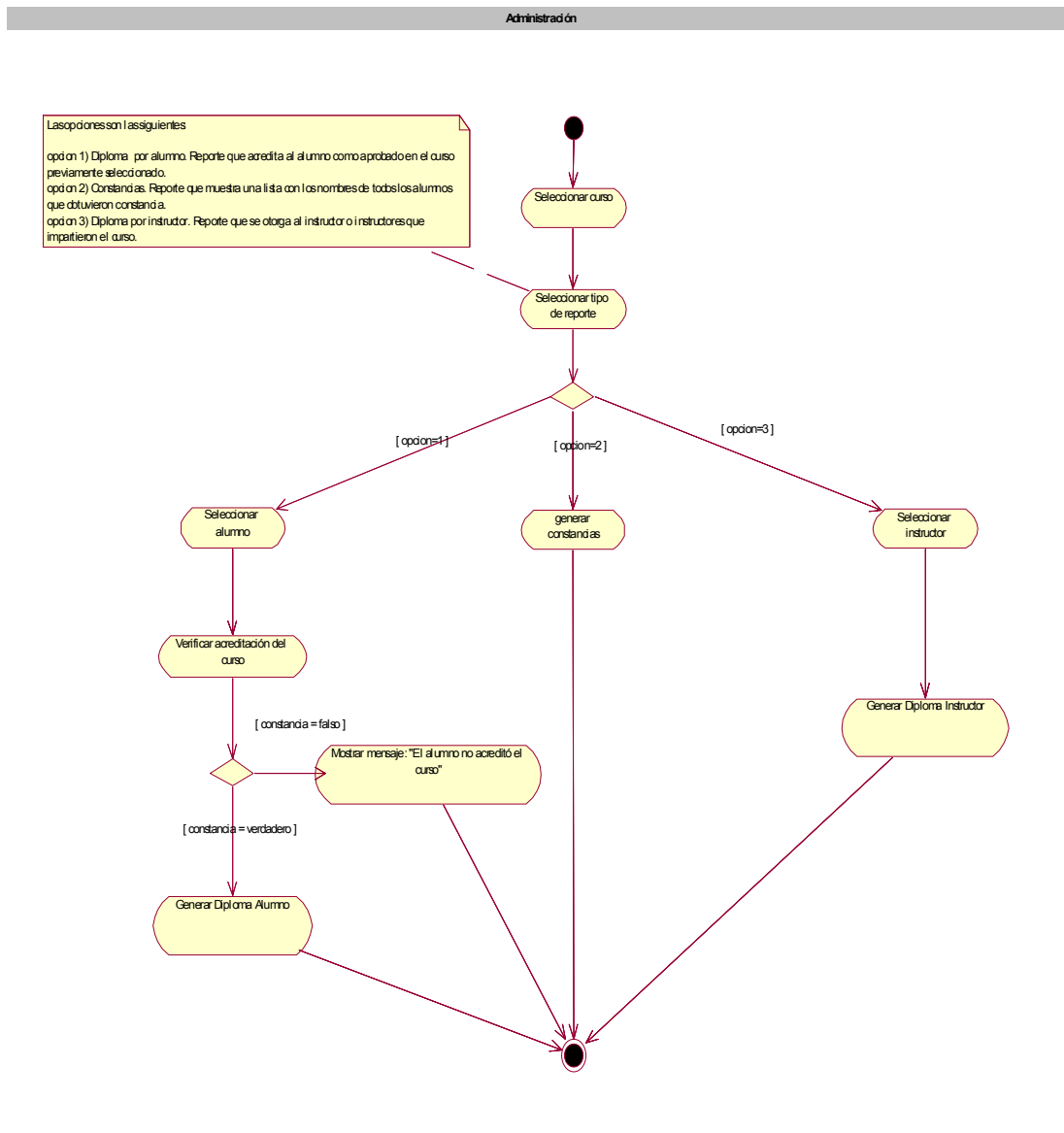


Figura 3.18. Diagrama de actividades para la generación de constancias

Este diagrama comienza con la actividad de **Seleccionar Curso** que básicamente se refiere a elegir el curso del cuál se pretende obtener las constancias. Posteriormente se elige el tipo de constancia que se desea. Si se elige la opción de “Diploma por Alumno” se debe seleccionar a continuación el alumno en cuestión y posteriormente se debe verificar si dicho alumno tiene derecho a constancia o no. Finalmente se manda a imprimir su Diploma. Lo mismo ocurre en los diplomas para Instructores. Otra opción que se tiene disponible es la de “Constancias” a la cuál recurrimos cuando se quiere una lista de todos los alumnos que obtuvieron constancia en un curso determinado.

## Generación de Clases y Objetos con Visual Basic

### 4.1 Creación de Clases y Objetos

La creación de clases para el proyecto SICC se realizó a partir del diagrama de clases elaborado en UML y al cual nos referimos en el capítulo anterior.

La elección de Visual Basic como lenguaje de programación fue debido a que así se indicó al principio del proyecto por parte del jefe del departamento de desarrollo de UNICA. Visual Basic no es un lenguaje orientado a objetos propiamente hablando. Tiene su propia manera de implementar la programación orientada a objetos. Muchos programadores argumentan que no es un lenguaje orientado a objetos, sino a eventos, debido a que no soporta al 100% los tres pilares de este concepto: encapsulación, herencia y polimorfismo.

En lo referente a la encapsulación, Visual Basic ofrece alternativas que si no soportan al 100% este concepto, permiten trabajar adecuadamente siguiendo la filosofía de ocultamiento y protección de la información. Esto se logra mediante la incorporación de los procedimientos Property, los cuáles permiten acceso a datos mediante una sintaxis simple y sin exponer datos internos a cambios no autorizados.

Visual Basic no proporciona características de lenguaje para compartir datos entre instancias de un objeto, esto se puede simular con variable públicas dentro de módulos estándar. Las variables públicas en módulos estándar dentro de un proyecto llegan a ser invisibles a los usuarios fuera del componente, por lo que hay alguna protección.



En la siguiente figura, son públicos al programa es decir, el objeto expone los métodos Guardar, Iniciar, Borrar y Modificar, así como las propiedades nombre y fecha\_inicio. Y son privados al objeto los procedimientos Guardar\_grupo, Borrar\_grupo y la variable nombre\_grupo. Y cada uno de estos métodos, propiedades, procedimientos y variables están encapsuladas en un objeto.

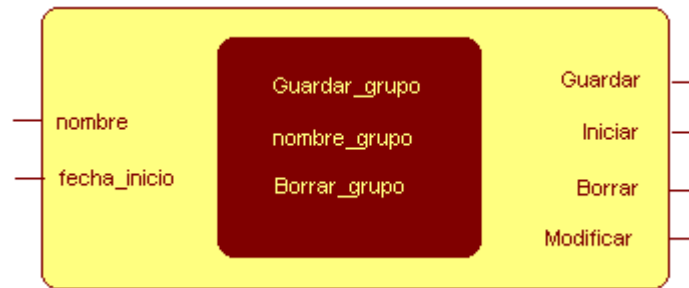


Figura 4.1. Encapsulamiento de Procedimientos y Variables en Visual Basic

Dado que visual basic es muy débil en las alternativas que proporciona para el polimorfismo y la herencia, en el sistema SICC se decidió suplir mediante código aquellas situaciones que requirieran el uso indispensable de dichos conceptos.

La implementación de las clases se realizó mediante los **módulos de clase** que proporciona el lenguaje para ello. Se trata de módulos muy parecidos a los formularios con algunas características particulares como son: cada módulo define un tipo de objeto, los objetos se crean en tiempo de ejecución, no presentan una interfaz gráfica al desarrollador tal y como lo hacen los formularios por lo tanto, sólo contienen código que define las características del objeto, de acuerdo a la configuración de las propiedades de la clase se pueden crear varias instancias u objetos de una misma clase, proporciona métodos y propiedades a exponer o no, entre otras.

### 4.1.1 Procedimientos Property

La definición de la clase en los módulos comienza por la especificación de las propiedades de dicha clase y posteriormente la de sus métodos. En el caso de las propiedades, se trata simplemente de variables declaradas con un determinado alcance y un tipo. Si las variables son declaradas como públicas son accesibles desde cualquier parte del proyecto. Aún más, si tienen éste perfil, no se tiene control acerca de si son propiedades de sólo lectura, escritura o ambas. Por ello, VB proporciona los métodos Property que además de solventar este problema, permiten tener una validación previa o ejecutar un procedimiento antes de actualizar la propiedad, generar un error o un evento como parte del código de validación o cambio de valor de la propiedad y limitar la propiedad a un grupo de posibles valores.

Se tienen tres tipos diferentes de procedimientos Property: Get, Let y Set. Si se declara una propiedad como Get únicamente, ésta es de sólo lectura. Si además se declara como Let la propiedad se convierte de lectura y escritura. Una propiedad declarada como Set indica que es de escritura, y se utiliza para establecer o cambiar la referencia de una variable de tipo objeto.

El siguiente cuadro muestra la sintaxis de los diferentes tipos de procedimientos Property, así como un ejemplo para cada uno de ellos.

Proced. Property	Sintaxis	Ejemplo
<b>Get</b>	Property Get NombrePropiedad([Argumentos]) As tipodato 'Código End Property	Private psNombre as String <b>Public Property Get</b> Nombre() as String Nombre = psNombre <b>End Property</b>
<b>Let</b>	Property Let NombrePropiedad([Argumentos], NombreValor As tipodato) 'Código End Property	Private psNombre as String <b>Public Property Let</b> Nombre(Valor as String) psNombre = Valor <b>End Property</b>
<b>Set</b>	Property Set NombrePropiedad([Argumentos], Referencia) 'Código End Property	Private poEmpleado as clsEmpelado <b>Public Property Set</b> RefEmpleado(ObjEmpleado as clsEmpelado) Set poEmpleado = ObjEmpleado <b>End Property</b>

Tabla 4.1. Procedimientos Property en Visual Basic

Para el caso de una variable protegida con el método Get, se observa en el cuadro que después de la declaración de la misma viene la invocación del método Property y la asignación de un nombre para la variable. Esto es, porque la variable que se quiere proteger será referida mediante el nuevo nombre que le asigne el método Property. Algo muy importante es que la variable al igual que el método Property deben tener el mismo tipo. Después se hace la asignación correspondiente entre la variable que proporcionará el método Property y la que se desea proteger.

Algo similar ocurre para los procedimientos Let y Set.

## 4.1.2 Colecciones

Una colección es lo que algunos libros de computación denominan **diccionario**, esto es, una tabla de elementos en la que se pueden buscar registros usando una clave.

Visual Basic presenta un tipo de dato llamado **Collection**. Este presenta las siguientes características:

- Almacena elementos relacionados.
- Proporciona los métodos **Add**, **Remove**, **Item** y la propiedad **Count** para manipular sus elementos y realizar operaciones sobre ellos.
- Se puede utilizar **For Each ... In** para acceder a los datos.

Al momento de añadir un elemento a un objeto colección con el método **Add**, se le puede incorporar un nombre clave único que lo relacione. De esta manera, existen dos formas de acceder elementos en una colección con el método **item**: por índice o por el nombre clave.

El método **Remove** sirve para eliminar elementos de una colección y **Count** nos da el número total de elementos presentes en una colección.

Generalmente se utiliza una clase colección que administre al objeto colección.

Como el objeto colección puede aceptar cualquier tipo de elemento: objetos, string, enteros, etc.; los cuales son transformados al tipo **variant** que es el único tipo de dato que aceptan las colecciones, la información a insertar por el usuario puede ser variada sino se controla. Esto es, podríamos tener objetos de varios tipos y por ende la colección no sería uniforme y constante. Se requiere que si se insertan objetos de una clase determinada sólo ese tipo de objetos aparezcan dentro de la colección.

Existen tres formas de implementar una colección:

Declarar una variable publica **As New Collection** en un **módulo estandard**.

Ejemplo:

```
Public colEmp As New Collection
```

Desventaja: el objeto colección es público y no se tiene control en la validación al utilizar el método **Add** de la colección. No se valida lo que el usuario inserta en la colección y si los datos de la colección son homogéneos.

Declarar una variable **As New Collection** y hacerla **Privada** en un módulo estándar, definir para el objeto un conjunto de métodos para agregar y eliminar objetos de la colección.

```
Private colEmp As New Collection
```

```
Private Function AñadirEmpleado(Emp As CEmpleado) As Boolean
```

```
End Function
```

```
Private Function BorrarEmpleado(NombreEmp As String, Optional Indice As Long)
```

```
End Function
```

Desventaja: se le protege del código externo pero no del interno, ya que puede ser muy compleja en el módulo estándar que la contiene y como consecuencia podría producir errores (errores de programación).

Implementar una clase colección propia, mediante un módulo de clase. Es igual a la anterior pero dentro de un módulo de clase, por lo que el código esta protegido.

Ventajas: estilo de programación modular muy claro.

La siguiente figura nos permite conceptualizar mejor los conceptos de colección y clase.

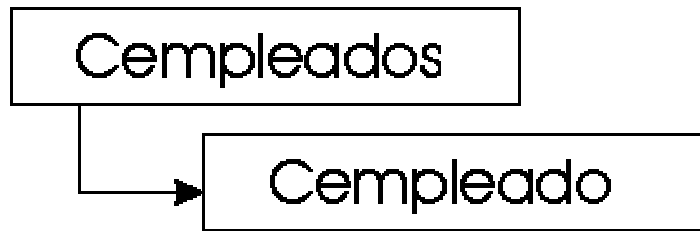


Figura 4.2. Colecciones y Clases en Visual Basic

Como se puede observar, tenemos una clase llamada **CEmpleado** la cual es singular y contiene todo el código relacionado a un solo empleado y por otro lado tenemos, la clase colección llamada **Cempleados** que es plural y esta basada en el objeto **CEmpleado**, esto es, mantiene la administración de todos los objetos creados con este tipo de clase.

### 4.1.3 Implementación de Clases en el SICC

Las clases previamente diseñadas en el diagrama de clases se implementaron en Visual Basic siguiendo la metodología antes descrita de módulos de clase y procedimientos Property.

La figura 4.3 muestra parte del código de la clase CGrupo. En ella se pueden apreciar algunas de las variables (atributos) que componen a la clase y la forma en que se protegieron mediante procedimientos Property.

```
Option Explicit
'Definicion de la clase Grupo
Private mintid_grupo As Integer
Private mstringnombre As String
Private mdatehora_inicial As Date
Private mdatehora_final As Date
Private mdatefecha_inicial As Date
Private mdatefecha_final As Date
Private mintvacantes As Integer
Private mintconfirmados As Integer
Private mintinscritos As Integer
Private mstringid_seccion As String
Private mstringcurso As String
Private mstringnivel As String
Private mstringtipo As String
Private mstringperiodo As String
Private mstringestado As String
Private mintduracion As Integer
Private mstringsala As String
Private mintcosto As Integer
Private mstringinstructorA As String
Private mstringinstructorB As String
Private malumnos As colAlumno

Public Property Get alumno() As colAlumno
    Set alumno = malumnos
End Property
Public Property Let alumno(ByVal vData As colAlumno)
    Set malumnos = vData
End Property
```

```
Public Property Let id(ByVal id_grupo As Integer)
    mintid_grupo = id_grupo
End Property
Public Property Get id() As Integer
    id = mintid_grupo
End Property

Public Property Let nombre(ByVal nombre_grupo As String)
    mstringnombre = nombre_grupo
End Property
Public Property Get nombre() As String
    nombre = mstringnombre
End Property

Public Property Let Hora_Inicial(ByVal hora_inicial_grupo As Date)
    mdatehora_inicial = hora_inicial_grupo
End Property
Public Property Get Hora_Inicial() As Date
    Hora_Inicial = mdatehora_inicial
End Property

Public Property Let Hora_Final(ByVal hora_final_grupo As Date)
    mdatehora_final = hora_final_grupo
End Property
Public Property Get Hora_Final() As Date
    Hora_Final = mdatehora_final
End Property
```

Figura 4.3. Parte del código de la clase Cgrupo implementada en el SICC



Al principio se observa el código correspondiente a la declaración de las variables que al aparecer en un módulo de clase automáticamente se convierten en atributos de la clase. En la parte superior se observa la instrucción **Option Explicit** la cuál se coloca para indicar que todas las variables que se usan en el módulo de clase han sido declaradas explícitamente.

Enseguida aparece la declaración de los procedimientos Property. Como ejemplo detallaremos dos considerando que el resto sigue el mismo formato y por ésta razón no aparecen todos en la figura 4.3.

El primero es el que se aplica a la variable alumnos de la clase grupo. Como se mencionó anteriormente, un grupo en particular tiene asociados un conjunto de alumnos. La forma de implementar esto en Visual Basic es a través de las clases tipo colección. En éste caso, los objetos de la clase grupo poseen un atributo del tipo colección que es el de **alumno**. El código que produce lo anterior es el siguiente:

```
Public Property Get alumno() As colAlumno  
    Set alumno = malumnos  
End Property  
Public Property Let alumno(ByVal vData As colAlumno)  
    Set malumnos = vData  
End Property
```

Se observa que la variable **malumnos** se convierte en el atributo protegido de lectura y escritura **alumno** gracias a la incorporación de los métodos Get y Let, y que además dicho atributo es del tipo colección (colAlumno). El código que crea la colección colAlumno se muestra en la figura 4.4.

Siguiendo con la descripción de la clase Cgrupo, veremos ahora otro atributo de la misma protegido por los procedimientos Get y Let. Se trata del atributo nombre y cuyo código se muestra en las siguientes líneas.

```
Public Property Let nombre(ByVal nombre_grupo As String)
```

```
mstringnombre = nombre_grupo  
End Property  
Public Property Get nombre() As String  
nombre = mstringnombre  
End Property
```

En primer instancia se observa la asignación del procedimiento Property Let y el nombre elegido para la variable protegida en este caso, **nombre**. (así es como nos referiremos de ahora en adelante a la propiedad “nombre” del objeto grupo) Se recibe un parámetro que nos sirve para almacenar temporalmente el valor que se pretende asignar a la variable (**nombre\_grupo**). Recordemos que el Property Let nos sirve para declarar una variable como de tipo escritura. Dicho parámetro es asignado a **mstringnombre** que según lo visto anteriormente es la variable que se quiere proteger. Por lo tanto, cuando se haga una escritura a la variable en cuestión, el valor capturado por el usuario se depositará en **mstringnombre** de forma automática, y como podemos ver en el código del Property Get, cuando se haga una lectura de dicha variable se hará a través de **nombre** que a su vez recibirá el valor pasado a **mstringnombre**.

Siguiendo éste mismo formato, se protegen todas las variables que aparecen en la clase CGrupo. Como se mencionó anteriormente, no aparece el código para el resto de las variables, sin embargo, se entiende que se trata del mismo procedimiento.

```
'variable local para contener colección  
Private mCol As Collection  
  
Public Function Add(Duracion As Integer, Estado As String, Periodo As String,  
Tipo As String, Nivel As String, Curso As String, Id_seccion As String, Inscritos As  
Integer, Confirmados As Integer, Vacantes As Integer, Fecha_Final As Date,  
Fecha_Inicial As Date, nombre As String, id As Integer, Hora_Final As Date,  
Hora_Inicial As Date, Costo As Integer, InstructorA As String, InstructorB As String,
```

```
Sala As String, Optional ByVal alumno As colAlumno, Optional sKey As String) As
CGrupo

'crear un nuevo objeto
Dim objNewMember As CGrupo
Set objNewMember = New CGrupo

'establecer las propiedades que se transfieren al método
objNewMember.Sala = Sala
objNewMember.InstructorB = InstructorB
objNewMember.InstructorA = InstructorA
objNewMember.Costo = Costo
objNewMember.Duracion = Duracion
objNewMember.Estado = Estado
objNewMember.Periodo = Periodo
objNewMember.Tipo = Tipo
objNewMember.Nivel = Nivel
objNewMember.Curso = Curso
objNewMember.Id_seccion = Id_seccion
objNewMember.Inscritos = Inscritos
objNewMember.Confirmados = Confirmados
objNewMember.Vacantes = Vacantes
objNewMember.Fecha_Final = Fecha_Final
objNewMember.Fecha_Inicial = Fecha_Inicial
objNewMember.Hora_Final = Hora_Final
objNewMember.Hora_Inicial = Hora_Inicial
objNewMember.nombre = nombre
objNewMember.id = id

'devolver el objeto creado
Set Add = objNewMember
```

```
        Set objNewMember = Nothing
    End Function

    Public Property Get Item(vntIndexKey As Variant) As CGrupo
        'se usa al hacer referencia a un elemento de la colección
        'vntIndexKey contiene el índice o la clave de la colección,
        'por lo que se declara como un Variant
        'Syntax: Set foo = x.Item(xyz) or Set foo = x.Item(5)
        Set Item = mCol(vntIndexKey)
    End Property

    Public Property Get Count() As Long
        'se usa al obtener el número de elementos de la
        'colección. Sintaxis: Debug.Print x.Count
        Count = mCol.Count
    End Property

    Public Sub Remove(vntIndexKey As Variant)
        'se usa al quitar un elemento de la colección
        'vntIndexKey contiene el índice o la clave, por lo que se
        'declara como un Variant
        'Sintaxis: x.Remove(xyz)
        mCol.Remove vntIndexKey
    End Sub
```

Figura 4.4. Código de la colección colGrupo implementada en el SICC. Se observa en la parte superior las instrucciones correspondientes al método Add. Se puede apreciar que los elementos que contendrá dicha colección corresponden a objetos de la clase CGrupo. El código restante es el referente a los métodos Item, Count y Remove de las colecciones y es agregado de manera automática al crear la colección mediante el wizard (ayudante en línea) de Visual Basic en un módulo de clase, indicando que se trata de una clase tipo colección.

Otra clase implementada y que es sumamente importante dentro del sistema es la de CBDGrupo. Dicha clase contiene todos los procedimientos necesarios para realizar operaciones sobre la base de datos en lo referente a los grupos. La siguiente figura muestra parte de su código en el que aparecen los elementos más importantes que contiene entre ellos el segmento correspondiente a la inserción de un nuevo grupo en la base de datos.

```
Option Explicit
'variables para la conexión a la base de datos
Public rsBDSICC As ADODB.Recordset
Private strSQL As String
Private strSQLtiposgrupo As String
    'Agregar clase y colección para los grupos
Public mgrupo As CGrupo
Public mgrupos As colGrupo
Public malumno As CAlumno
Public malumnos As colAlumno
Dim tipogrupos As String

    Private Sub Class_Initialize()
'aquí se crean los objetos para obtener los parámetros
Set mgrupo = New CGrupo
Set mgrupos = New colGrupo
    'para la conexión
Set rsBDSICC = New ADODB.Recordset
rsBDSICC.CursorType = adOpenKeyset
rsBDSICC.LockType = adLockOptimistic
End Sub

Public Function Agregar(valor() As Variant, ByRef Grupo As CGrupo)
aviso_error_instructor = False
rsBDSICC.Source = "select tipo from tipo_grupo where comentarios="
```

```
+ frmperido.cboTipo.List(frmperido.cboTipo.ListIndex) + " "  
cn.Open  
rsBDSICC.ActiveConnection = cn  
rsBDSICC.Open  
tipogrupo = rsBDSICC.Fields("tipo")  
rsBDSICC.Close  
cn.Close  
rsBDSICC.Source = "select * from grupo"  
cn.Open  
Set rsBDSICC.ActiveConnection = cn  
rsBDSICC.Open  
rsBDSICC.AddNew  
rsBDSICC.Fields("nombre") = Trim(valor(13))  
rsBDSICC.Fields("duracion") = Trim(valor(1))  
rsBDSICC.Fields("fecha_final") = Trim(valor(2))  
rsBDSICC.Fields("hora_inicial") = Trim(valor(3))  
rsBDSICC.Fields("hora_final") = Trim(valor(4))  
rsBDSICC.Fields("fecha_inicial") = Trim(valor(5))  
rsBDSICC.Fields("curso") = Trim(valor(6))  
rsBDSICC.Fields("id_seccion") = Trim(valor(8))  
rsBDSICC.Fields("estado") = Trim(valor(9))  
rsBDSICC.Fields("nivel") = Trim(valor(10))  
rsBDSICC.Fields("tipo") = tipogrupo  
rsBDSICC.Fields("periodo") = frmSICC.lblSemestre.Caption  
On Error GoTo er  
rsBDSICC.Update  
'valida que no elijan nombres de grupos ya existentes  
er:  
If Err.Number = -2147467259 Then  
    MsgBox "Debe elegir un nombre de grupo diferente a los existentes"
```

```
        aviso_error_instructor = True
        rsBDSICC.CancelUpdate
        rsBDSICC.Close
        cn.Close
        frmGrupos.mskgrupo(0).SetFocus
        Exit Function
    End If
    rsBDSICC.Close
    cn.Close
    End Function
    ....
    ....
```

Figura 4.5. Parte del código de la clase CBDGrupo implementada en el SICC.

En la parte superior observamos la declaración de las variables que se utilizan en la conexión a la base de datos. Básicamente se trata de objetos recordset y de algunas variables tipo cadena que se utilizan para almacenar los *queries*<sup>1</sup> correspondientes a cada operación en particular. En nuestro caso el objeto recordset rsBDSICC se utilizará más adelante en la operación de inserción de un nuevo grupo.

Después se observa la declaración de los objetos y de las colecciones que utiliza en algunos casos la aplicación.

Posteriormente aparece el código referente al evento Class\_Initialize que sólo contiene la creación del objeto grupo y de la colección mgrupos que se utilizan en muchas operaciones sobre la base de datos.

---

<sup>1</sup> Query. Fragmento de código que realiza una operación de consulta, insertado, borrado, actualización o alguna otra sobre una base de datos.

Enseguida viene la creación del objeto recordset `rsBDSICC` y la asignación de sus propiedades referentes al tipo de conexión que se hará con la base de datos. La primera de ellas asigna el tipo de cursor que utilizará el recordset y con la opción elegida (**`adOpenKeyset`**), los cambios que otros usuarios hacen permanecen visibles. La instrucción es la siguiente:

```
rsBDSICC.CursorType = adOpenKeyset
```

A continuación viene la instrucción que indica el tipo de bloqueo que se pone en los registros durante el proceso de edición.

```
rsBDSICC.LockType = adLockOptimistic
```

Con la opción **`adLockOptimistic`** (bloqueo optimista) sólo se bloquean los registros cuando se realiza alguna actualización sobre la base de datos.

Con estas propiedades asignadas, el objeto recordset está listo para ser utilizado.

El siguiente fragmento de código corresponde a la función “Agregar” que se encarga de insertar un nuevo grupo en la base de datos.

En la parte superior del código de dicha función se puede observar que recibe un arreglo llamado “valor” como parámetro. Dicho arreglo contiene todos los datos para el nuevo grupo (nombre, fecha inicial, fecha final, duración, etc) que previamente se capturaron mediante la interfaz creada para ello. Más adelante se asigna al objeto recordset un query para obtener el tipo de grupo que se está visualizando actualmente. Esto es porque la operación que se quiere realizar es la de insertar un nuevo grupo en la base de datos y por lo tanto, es necesario asignarle un tipo de grupo. Para ello se hace una consulta con el fin de saber en ese momento cuáles son los tipos de grupo que se están visualizando y poder asignar dicha información correctamente. Sólo se pueden asignar grupos del tipo que se esté visualizando en ese momento. Si se requiere asignar otro tipo de grupo es necesario visualizar grupos del tipo



deseado antes de poder crear uno nuevo. Esto se verá más a detalle en el subtema 4.3 Interfaz del sistema de control de cursos e inscripciones.

Una vez obtenido el tipo, se abre el recordset y se invoca a su método **ADDNew** para indicar que se insertará un nuevo registro en la base de datos. Posteriormente se asignan los nuevos valores que llevará dicho registro con las líneas:

```
rsBDSICC.Fields("nombre") = Trim(valor(13))
rsBDSICC.Fields("duracion") = Trim(valor(1))
rsBDSICC.Fields("fecha_final") = Trim(valor(2))
...
```

Finalmente con la instrucción **rsBDSICC.Update** se ejecuta la operación y se actualiza la base de datos. El código siguiente sirve para capturar un posible error de duplicado en los nombres de los grupos en cuyo caso la operación de insertado es cancelada.

Funciones como ésta son las que tiene la clase CBDGrupo y todas ellas referentes a operaciones sobre la base de datos. Lo mismo ocurre con las clases que tienen las siglas CBD en sus respectivos nombres y que como se mencionó anteriormente, se eligió implementarlas de ésta forma para obtener un mayor provecho de los objetos ADO y llevar un código más limpio en el desarrollo de las clases del sistema.

### 4.1.4 Implementación de Objetos en el SICC

La declaración de objetos o instanciación en Visual Basic se puede realizar mediante la instrucción **CreateObject** y una variable del tipo **Object** y mediante la instrucción **New** siendo ésta última mucho más sencilla de emplear.

Las instancias de clase son llamadas usualmente **Objetos**. Para declarar una instancia de clase se debe utilizar una variable del tipo de la clase que deseamos instanciar y utilizar la clausula **New** para generar la instancia u objeto.

**Dim Empleado as clsEmpleado**

**Set Empleado = New clsEmpleado**

La razón de usar **New** es porque con las instancias de clase, la variable no es lo mismo que la instancia. La variable es el único medio de comunicación con la instancia. Las variables de clase son también llamadas *variables de objeto* y las instancias son llamados *Objetos*.

Existen dos formas de instanciar las variables de objeto mediante la clausula *New*:

Declarar la variable de objeto y posteriormente utilizar *Set... New* para instanciar la clase.

Un ejemplo de este caso es como se manejo arriba.

*Dim VariableObjeto As NombreClase*

*Set VariableObjeto = New NombreClase*

Esta es la forma más recomendable si se desea tener el control sobre la instancia de la variable: cuando se debe crear, usar, terminar, volver a crear... etc.

Instanciar la variable de objeto en la misma declaración de ésta como *As New*.

*Dim VariableObjeto As New NombreClase*

De ésta forma a la hora que entre al modulo o procedimiento que la contiene se generara automáticamente la instanciación de la clase.

Se tiene menos control sobre la instanciación de la clase; además de no poder determinar el momento en que deseamos que nuestra variable de objeto sea instanciada.

Para los objetos del SICC se utilizó la primera opción ya que permite un mejor control sobre las instancias tal y como se mencionó anteriormente.

Las siguientes líneas muestran un ejemplo de la creación de un objeto de la clase CGrupo mediante Visual Basic:

```
Dim grupo As CGrupo
```

```
Set grupo = New CGrupo
```

De esta forma se tiene la instancia de la clase CGrupo llamada grupo que posee todas las propiedades y métodos de la clase de la cuál procede. Para acceder a ellas, se escribe el nombre del objeto seguido de un punto e inmediatamente aparece un menú desplegable con todas las características y métodos que posee dicho objeto. La siguiente figura ilustra mejor lo anterior:

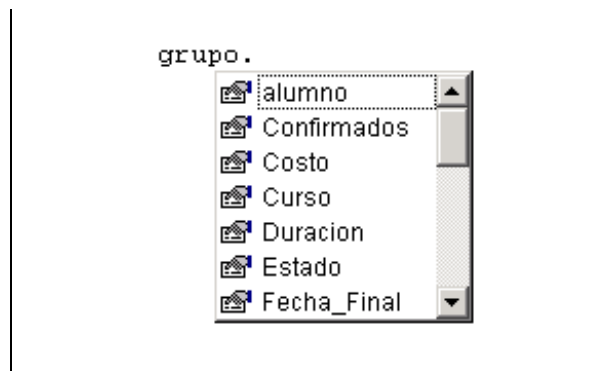


Figura 4.6. Objeto grupo creado a partir de la clase CGrupo en el SICC

De esta forma se trabaja con todos los objetos en el SICC y una vez que se han completado las operaciones con ellos se procede a eliminarlos de la memoria de la computadora y de ésta forma liberar recursos del sistema mediante la siguiente instrucción.

**Set VariableObjeto = Nothing**

## 4.2 Creación de Procedimientos y Funciones

Los procedimientos y funciones son muy interesantes y útiles en la programación. Sirven para realizar una tarea concreta y que probablemente se vaya a ejecutar varias veces a lo largo de la vida de un sistema. Esta tarea se especifica en un bloque de código de manera independiente y cuando se desean realizar las acciones del procedimiento se le llama. Una vez realizadas las acciones pertinentes se devuelve el flujo del programa al lugar desde donde se invocó ese procedimiento o función.

Lo primero que se debe hacer al crear un procedimiento es pensar las cosas que se desean hacer dentro de la función, la información que se necesita (y que se tiene que recibir como parámetros) y la información que se devolverá. Con éstas ideas claras se pueden construir los procedimientos y funciones sin mucha dificultad, siguiendo estas estructuras.

Para un procedimiento:

**Sub** nombre (parametro1, parametro2,...)

Codigo del procedimiento

End Sub

Para una función

**Function** nombre (parametro1, parametro2,...)

Codigo de la función

End Function

### 4.2.1 Diferencias entre Procedimientos y Funciones

Un procedimiento es una subrutina que se llama y realiza acciones, pero que no devuelve ningún valor y por lo tanto, no es posible utilizarla dentro de una expresión. Una función es un trozo de código que opera para devolver un valor.

En Visual Basic las funciones se utilizan como partes de expresiones y los procedimientos como si fuera una sentencia independiente. La llamada a una función, si se utiliza como parte de una expresión se debe llamar utilizando paréntesis.

**miResultado = suma(1,2)**

Si no se utiliza como parte de una expresión, no tienen por que utilizarse los paréntesis, pero el resultado de la función (lo que devuelve) se perderá.

**Call** Es una llamada a una subrutina, utilizada para transferir el flujo de la aplicación hacia una subrutina. Es necesario utilizar paréntesis cuando se utiliza. Además, si se utiliza con una función se perderá el resultado que devuelva.

**call suma(1,2)**

Salida de una subrutina. Podemos salirnos de un procedimiento o función en cualquier momento, independientemente de que la función haya terminado o no. La sentencia para escaparse de una función es **EXIT**, que se puede utilizar en cualquier lugar del procedimiento o función. La palabra **exit** debe ir acompañada del tipo de subrutina de la que se desea salir, así pues se deberá utilizar o bien **exit function** o **exit sub**.

Dentro del SICC se utilizaron ambos tipos de subrutinas: procedimientos y funciones. A continuación describiremos dos procedimientos, el primero de ellos se utiliza para visualizar los grupos de un determinado período y tipo de grupo. El segundo nos sirve para limpiar la

colección de grupos y los objetos de la clase CGrupo que se utilizan en muchas operaciones dentro del sistema. Por último, se detalla una función utilizada para realizar la validación de los usuarios que entran al sistema.

```
Public Sub Cargar_Grupos(seleccion As Boolean)
    Dim j As Integer
    j = 1

    'Obtener Registros de la base de datos.
    grupos.Obtener_Registros (seleccion)
    If grupos.mgrupos.Count = 0 Then
        hay_registros_grupo = False
    Else
        hay_registros_grupo = True
    End If

    If hay_registros_grupo Then
        For Each Grupo In grupos.mgrupos
            mfgGrupos.AddItem CStr(j) & vbTab & Grupo.nombre & vbTab &
            Grupo.Cursor & vbTab & Grupo.Estado & vbTab & Grupo.Vacantes & vbTab &
            Grupo.Fecha_Inicial & vbTab & Format(Grupo.Hora_Inicial, "hh:mm AMPM"),
            j
            j = j + 1
        Next
    End If
End Sub
```

Figura 4.7. Código correspondiente al Procedimiento Cargar\_Grupos del SICC

El procedimiento Cargar\_Grupos() se utiliza como se mencionó anteriormente para visualizar los grupos de un determinado período y tipo de grupo. La elección del período y tipo

de grupos se realiza en operaciones previas a la llamada de esta función, por lo que dichos parámetros no se seleccionan aquí.

Vemos que el procedimiento recibe como parámetro una variable del tipo booleano llamada **selección**. Posteriormente se declara una variable tipo entero llamada *j* y se inicializa en 1.

Enseguida viene la llamada a un procedimiento que tiene el objeto *grupos* el cuál es una instancia de la clase *CBDGrupo* a la cuál nos hemos referido anteriormente. Dicho procedimiento se llama *Obtener\_Registros ()* y recibe el mismo parámetro con el que se invocó a *Cargar\_Grupos()*. El código correspondiente a éste procedimiento es muy extenso y no lo mencionaremos en esta ocasión, sólo diremos que se encarga de hacer las operaciones necesarias sobre la base de datos para poder traer los grupos correspondientes al período y tipo de grupo seleccionados previamente. Dichos grupos se cargan en una colección (*mgrupos*) también declarada en *CBDGrupo* y por lo tanto, también se accede a ella a través del objeto *grupos*.

En seguida se hace una validación para saber si se obtuvieron registros o no en la consulta. Si existen grupos en la base de datos se hace verdadera a la variable de tipo booleano **hay\_registros\_grupo** en caso contrario toma el valor de falso.

Si la variable *hay\_registros\_grupo* es verdadera se procede a realizar el cargado de los datos en la interfaz correspondiente a través de un componente de visual basic llamado *flexgrid*<sup>2</sup>.

Para realizar el llenado del *flexgrid* en nuestro caso llamado *mfgGrupos* (las siglas *mfg* provienen de Microsoft FlexGrid y es el estándar seguido para los componentes de éste tipo utilizados en el SICCC), se hace uso de la colección que se mencionó anteriormente *mgrupos* y que contiene todos los grupos existentes para el período y tipo de grupo elegido. También se

---

<sup>2</sup> Microsoft Hierarchical FlexGrid Control 6.0 (OLEDB) componente de Visual Basic que muestra una tabla a manera de cuadrícula donde se pueden desplegar datos dispuestos de alguna forma en particular.



observa que se utiliza la instrucción **For Each...In** para acceder a cada elemento de la colección. Posteriormente se invoca al método **AddItem** del flexgrid para agregar nuevos elementos y se van desplegando uno a uno a través del objeto Grupo (proveniente de la colección mgrupos) mediante la invocación de sus propiedades nombre (Grupo.nombre), curso (Grupo.Curso), estado (Grupo.Estado), vacantes (Grupo.Vacantes), fecha\_inicial (Grupo.Fecha\_Inicial) y hora\_inicial (Grupo.Hora\_Inicial), éste último campo con el formato de horas y minutos separados con dos puntos y la posibilidad de desplegar si es horario AM ó PM.

La siguiente figura muestra el resultado de éste procedimiento y como se distribuyen los datos dentro del flexgrid.

	Grupo	Curso	Estado	Vacantes	Fecha Inicial	Hora Inicial
1	01-F	INTRODUCCION A LA COMPUTACION	ABIERTO	32	17/05/2004	09:00 a.m.
2	02-F	AUTOCAD BASICO	CERRADO	30	20/03/2004	09:00 a.m.
3	03-F	LINUX BASICO	CERRADO	29	20/03/2004	09:00 a.m.
4	04-F	JAVA SCRIPT	CANCELADO	10	25/03/2004	09:00 a.m.
5	05-F	PHP CON BASE DE DATOS	CERRADO	35	20/03/2004	09:00 a.m.
6	06-F	JAVA SERVER PAGES	ABIERTO	21	17/04/2004	09:00 a.m.
7	07-F	INTRODUCCION A OFFICE 2000	ABIERTO	29	17/04/2004	09:00 a.m.
8	08-F	HTML CON FLASH	ABIERTO	28	17/04/2004	09:00 a.m.
9	09-F	LENGUAJE C BASICO	ABIERTO	25	17/04/2004	09:00 a.m.
10	10-F	VISUAL BASIC CON BASES DE DATOS	ABIERTO	31	17/04/2004	09:00 a.m.
11	11-F	DISEÑO DE PAGINAS WEB CON HTML	ABIERTO	8	17/04/2004	09:00 a.m.
12	12-F	VISUAL BASIC .NET	ABIERTO	11	17/04/2004	09:00 a.m.
13	13-F	COMPUTACION PARA NIÑOS	CERRADO	0	20/03/2004	09:00 a.m.

Figura 4.8. Cargado de Grupos en el SICC como resultado de la ejecución del procedimiento **Cargar\_Grupos**.

El siguiente procedimiento permite limpiar una colección así como los objetos grupo que se utilizan dentro de ella.

```
Public Sub limpiar_coleccion_grupo()  
    Set grupos = Nothing  
    Set grupos = New CbdGrupo  
    Set mgrupos = Nothing  
    Set mgrupos = New colGrupo  
    mfgGrupos.Rows = 2  
    mfgGrupos.TextMatrix(1, 0) = vbNullString  
    mfgGrupos.TextMatrix(1, 1) = vbNullString  
    mfgGrupos.TextMatrix(1, 2) = vbNullString  
    mfgGrupos.TextMatrix(1, 3) = vbNullString  
    mfgGrupos.TextMatrix(1, 4) = vbNullString  
    mfgGrupos.TextMatrix(1, 5) = vbNullString  
    mfgGrupos.TextMatrix(1, 6) = vbNullString  
End Sub
```

Figura 4.9. Procedimiento limpiar\_colección\_grupo

En la parte superior se observa que el objeto **grupos** de la clase CBDGrupo se limpia, destruyéndose primero e inmediatamente después se crea.

Lo mismo ocurre para la colección mgrupos la cuál contiene los grupos para un determinado período y para un tipo de grupo específico.

Después, se limpia el flexgrid que despliega a los grupos. Esto se logra con la reasignación de dicho componente a dos renglones con la siguiente instrucción:

```
mfgGrupos.Rows = 2
```

y posteriormente se limpian asignando cada celda al valor de `vbNullString`<sup>3</sup> con las siguientes instrucciones:

```
mfgGrupos.TextMatrix(1, 0) = vbNullString
mfgGrupos.TextMatrix(1, 1) = vbNullString
mfgGrupos.TextMatrix(1, 2) = vbNullString
....
```

Finalmente veremos la función `Valida` que se utiliza para hacer la autenticación del usuario que pretende ingresar al sistema. La función verifica que el usuario sea “coordsicc” y deja la autenticación de contraseñas del lado del servidor.

```
Function Valida(usuario As String) As Boolean
    usuario = LCase(Trim(usuario))
    If usuario = "coordsicc" Then
        Valida = True
    Else
        Valida = False
    End If
End Function
```

Figura 4.10. Código correspondiente a la función **Valida** del SICCC

La función recibe la variable `usuario` como parámetro. La primera operación que se hace es convertir la cadena de usuario a minúsculas y se le quitan los espacios en blanco con la siguiente instrucción:

```
usuario = LCase(Trim(usuario))
```

Después se verifica que el usuario corresponde al “coordsicc” en cuyo caso la función toma el valor de verdadero. El paso siguiente es la verificación de la contraseña y los posibles casos de error que se pueden presentar y de lo cuál se encarga otra función.

---

<sup>3</sup> `VbNullString`. Constante de programación de Visual Basic que sirve para indicar que una variable tipo cadena tiene el valor de 0.

A grandes rasgos todas las funciones en el SICC siguen el mismo formato lo mismo que los procedimientos. El objetivo de detallar aquí algunos es sólo para mostrar la forma en que se implementaron pero no para mostrar todo el código actividad que resultará impráctica y tediosa.

## 4.3 Interfaz del sistema de control de cursos e inscripciones

### 4.3.1 Definición de Interfaz

Una interfaz de software es la parte de una aplicación que el usuario ve y con la cual interactúa. Está relacionada con la subyacente estructura, la arquitectura, y el código que hace el trabajo del software, pero no se confunde con ellos. La interfaz incluye las pantallas, ventanas, controles, menús, teclado, ratón, la ayuda en línea y la documentación. Cualquier cosa que el usuario ve y con lo cual interactúa es parte de la interfaz.

A grandes rasgos podemos decir que una interfaz es el elemento que hace posible la **comunicación** entre dos entes (aparatos, personas, entidades u objetos). Por ejemplo, Una persona que hace traducción simultánea es una interfaz que permite a otras dos personas comunicarse, cuando hablan idiomas diferentes. Un modem es una interfaz que permite comunicar líneas telefónicas con dispositivos de acceso a computadores. Una página web es una interfaz que permite a una entidad o persona comunicarse con gente que está navegando en la Internet.

### 4.3.2 Interfaz del SICC

La interfaz que se implementó para el sistema contiene menús desplegables y barras de herramientas. Esta desarrollada con Visual Basic 6.0 y posee una serie de componentes especiales que se describirán más adelante.

La interfaz de la aplicación web se desarrolló con el lenguaje de programación php y también se detalla en éste capítulo..

### 4.3.2.1 Interfaz Módulo 3 (Control e Inscripción a cursos).

Para el control y administración de toda la información referente a los cursos, se implementaron dos interfaces. La primera de ellas realiza todas las funciones correspondientes a la administración de grupos, tal como la creación, modificación o borrado de ellos. La otra es básicamente de consulta y no permite las operaciones antes mencionadas para la primer interfaz y es muy similar a ella salvo en las limitantes ya mencionadas.

#### 4.3.2.1.1 Interfaz de Administración

La primer pantalla que aparece en ésta interfaz es la de validación y en ella se pide que el usuario se autentifique de acuerdo a los privilegios que posee a través de la contraseñacorrespondiente.

La siguiente figura ilustra lo anterior:

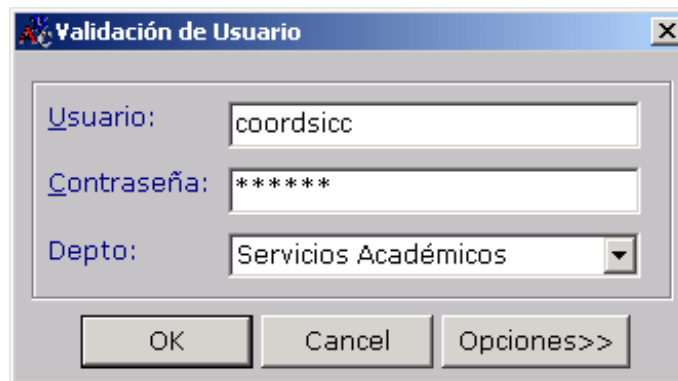


Figura 4.11. Ventana de Validación de usuario (Interfaz de Administración)

Con el botón de Opciones se pueden cambiar los parámetros de entrada es decir, la base de datos, el nombre de la base y el puerto de comunicación:



Figura 4.12. Ventana de validación de usuario con las opciones de servidor, nombre de la base de datos y puerto de comunicación habilitadas.

Si el usuario es válido aparece la pantalla de bienvenida mientras se carga la aplicación:



Figura 4.13. Ventana de bienvenida (Interfaz de Administración).

Dicha pantalla aparece unos cuantos segundos y después se muestra la interfaz principal del Sistema de Inscripciones y Control de Cursos. En ésta, se encuentran las operaciones básicas para la visualización de grupos de un determinado período y tipo de grupo. Por lo tanto, antes de acceder a dicha interfaz se debe elegir un período y un tipo de grupo en la ventana siguiente:

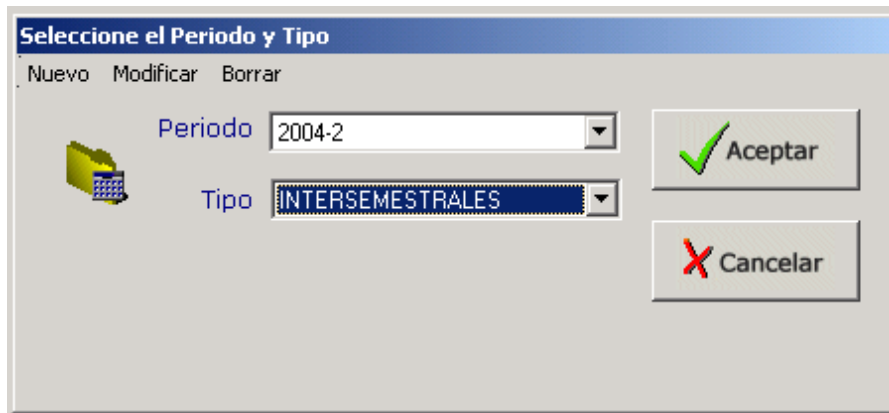


Figura 4.14. Ventana de Selección de Período y tipo de grupos.

Como se puede observar en el menú de la parte superior, ésta ventana nos permite realizar las operaciones básicas (Ingresar un nuevo tipo de grupo, modificar alguno existente o definitivamente borrarlo del sistema). Una vez elegido el período y el tipo de grupo se visualiza la interfaz principal del SICC:

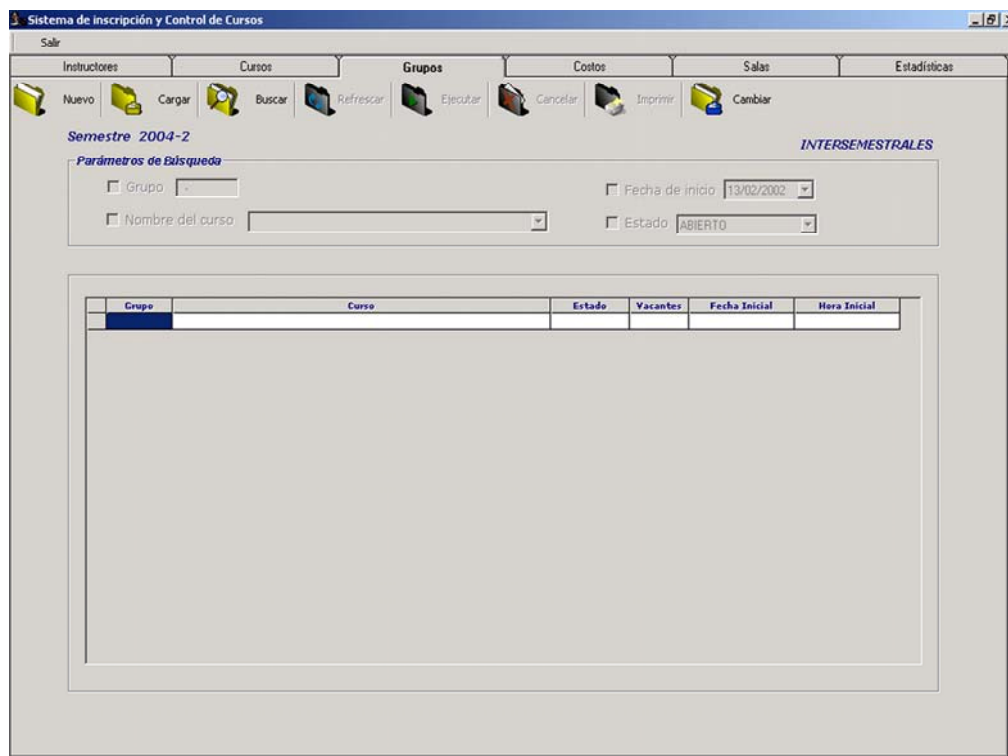


Figura 4.15. Pantalla Principal del SICC (Administración)



En la parte superior aparece un toolbar<sup>4</sup> que contiene las pantallas de “Instructores”, “Cursos”, “Grupos”, “Costos”, “Salas” y “Estadísticas”.

La parte de Grupos es la correspondiente al presente trabajo de tesis. Las otras forman parte de otros módulos del mismo sistema.

Dentro de la pestaña de grupos, se observa un nuevo toolbar con los botones de “Nuevo”, “Cargar”, “Buscar”, “Refrescar”, “Ejecutar”, “Cancelar”, “Imprimir” y “Cambiar”. El siguiente cuadro resume las operaciones que realizan cada uno de éstos botones.

Botón	Función
Nuevo	Permite la creación de un nuevo grupo habilitando otra pantalla en la cuál se realiza la captura de datos. El nuevo grupo por crear será para el período y tipo de grupo que se eligió previamente.
Cargar	Despliega todos los grupos existentes para el tipo de grupo y período elegidos.
Buscar	Permite realizar búsquedas de acuerdo a los siguientes criterios: <ul style="list-style-type: none"> <li>• Por nombre del grupo</li> <li>• Por curso</li> <li>• Por la fecha de inicio</li> <li>• Por su estado (abierto, cerrado o cancelado)</li> </ul>
Refrescar	Realiza nuevamente un cargado de los grupos de acuerdo a la última operación realizada (cargar o buscar), actualizando de esta forma los datos en pantalla.
Ejecutar	Lleva a cabo la operación de buscar. Es decir, cuando se oprime el botón de buscar se indica bajo que criterio se realizará dicha operación y se eligen los valores a buscar, pero no es hasta que se oprima el botón de ejecutar que se realizará dicha búsqueda.

<sup>4</sup> Toolbar es un control de Visual Basic 6 que aparece dentro del paquete Microsoft Windows Common controls 6.0. Es un control que presenta una barra de tareas. Es configurable, pudiendo poner los botones que se deseen con el icono apropiado.

Cancelar	Por medio de éste botón se cancela alguna operación previamente elegida.
Imprimir	Permite imprimir un reporte con los datos mostrados en pantalla acerca de los grupos (nombre, curso, estado, vacantes, fecha inicial y hora inicial)
Cambiar	Muestra una ventana en donde se puede elegir otro tipo de grupo y algún otro período válido para el sistema (figura 4.14).

Tabla 4.2. Botones de la Interfaz Principal de Grupos del SICC

La siguiente figura muestra como se despliegan los grupos una vez oprimido el botón de Cargar.

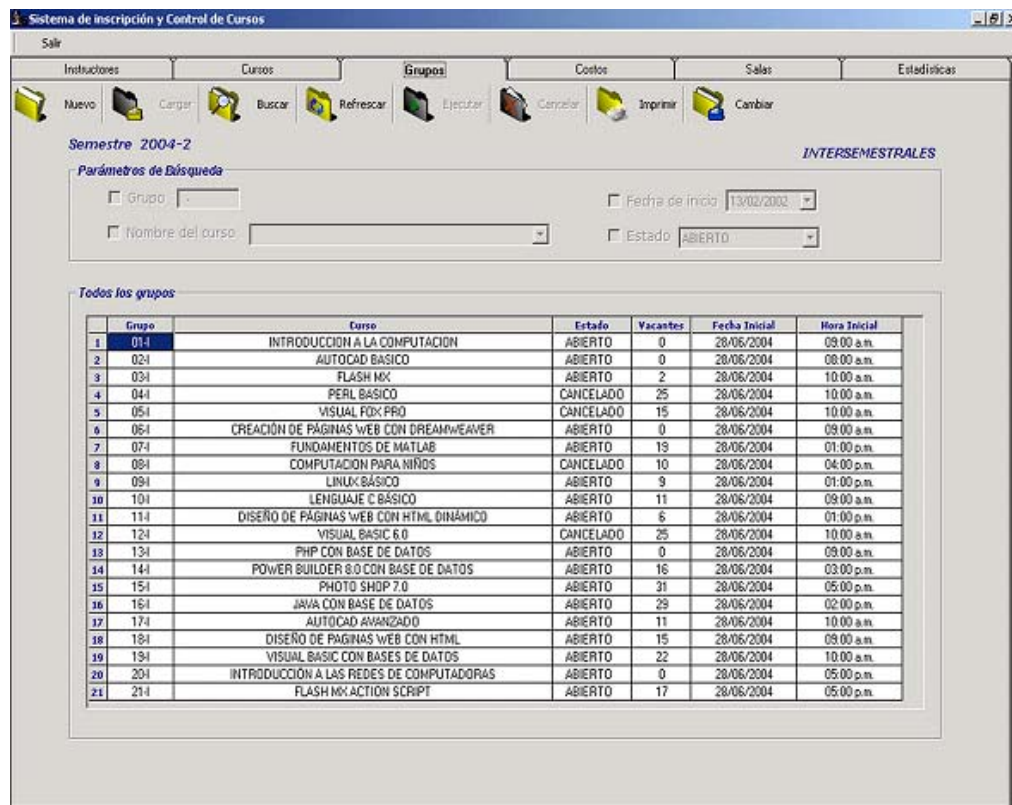


Figura 4.16. Visualización de Grupos

El siguiente ejemplo muestra el funcionamiento del botón Buscar. Se puede apreciar en la figura como se habilitan las opciones para realizar búsquedas especializadas por grupo, curso, fecha de inicio, estado o todas a la vez cuando se oprime dicho botón.

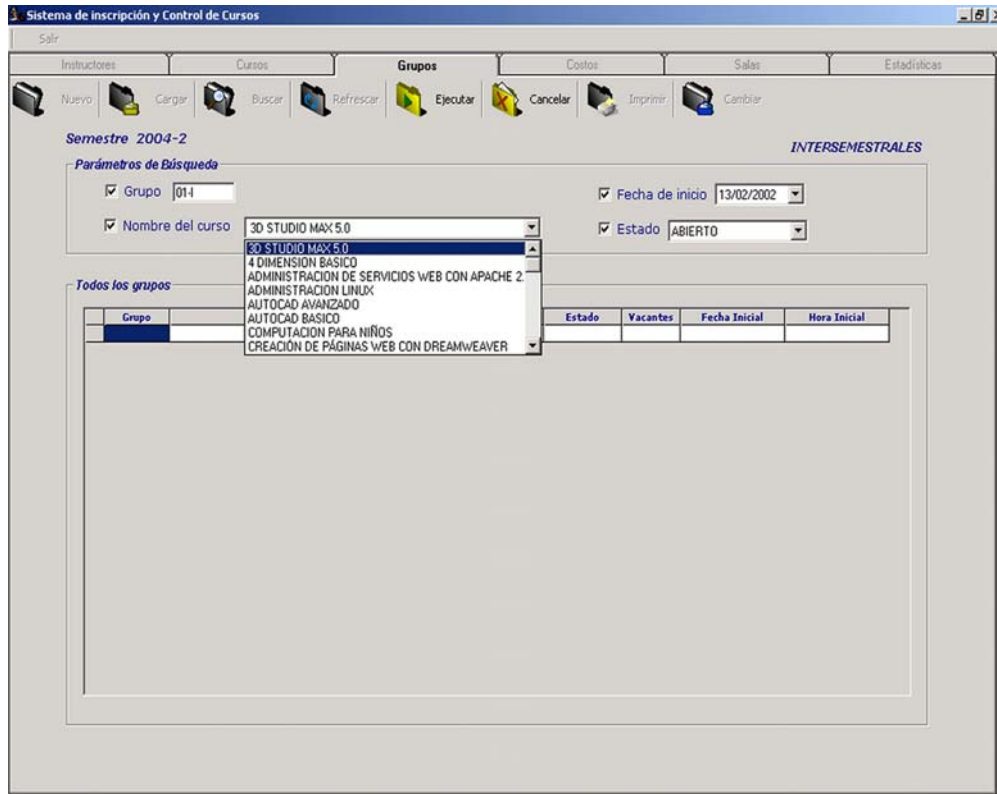


Figura 4.17. Búsqueda de grupos.

El botón de “Nuevo” permite la creación de un nuevo grupo habilitando otra interfaz especial para ello, la cuál se muestra en la siguiente figura:

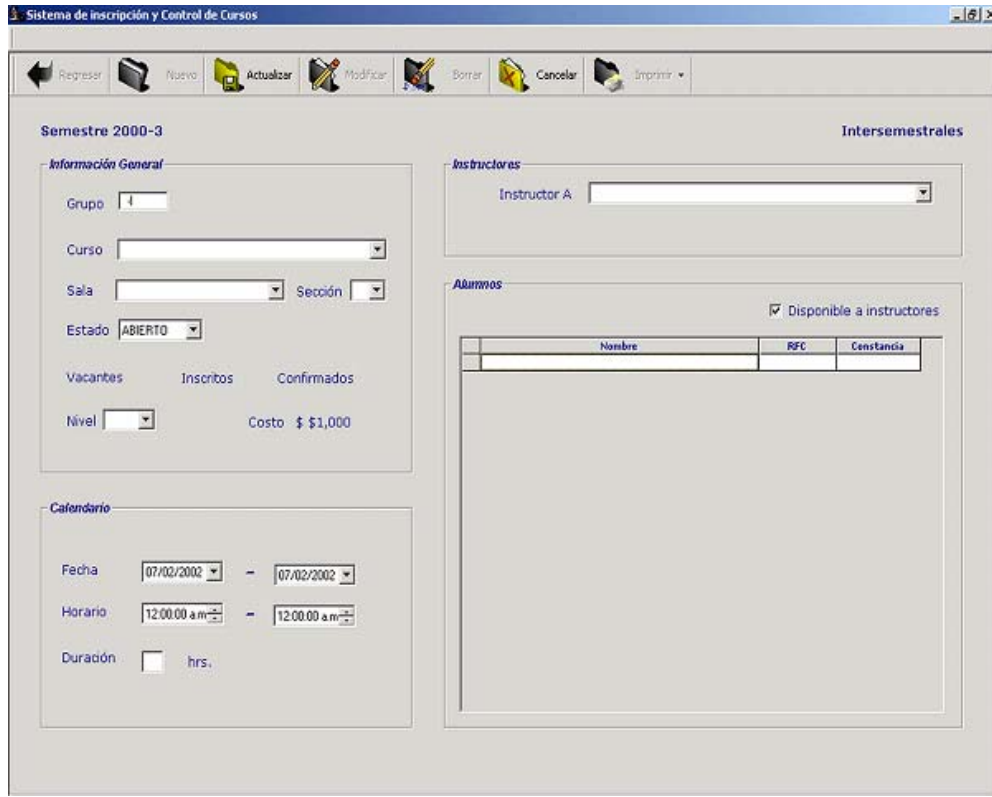


Figura 4.18. Pantalla de detalle de Grupos

A ésta pantalla se puede acceder de dos formas distintas. La primera de ellas es la mencionada antes a través del botón “Nuevo” de la interfaz principal la cuál la habilita para la captura de los datos del nuevo grupo. La otra es mediante el control flexgrid de la primera pantalla una vez desplegados los grupos y se dá doble clic sobre alguno de ellos. Esto llama a la interfaz pero con el detalle del grupo seleccionado en el flexgrid. El siguiente ejemplo ilustra dicho procedimiento:

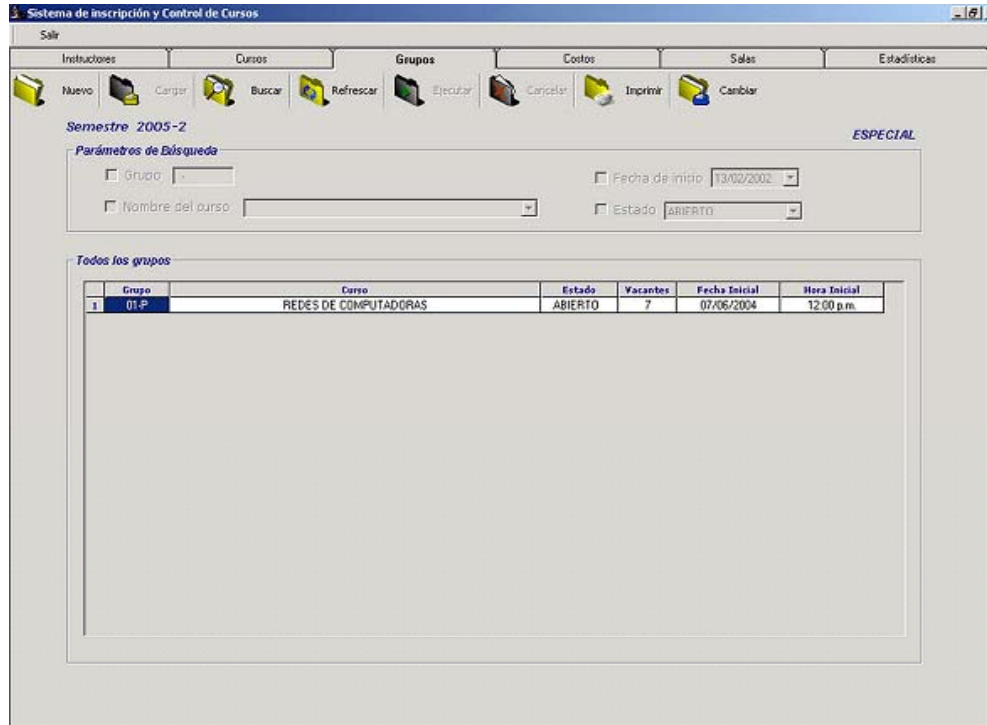


Figura 4.19. Cargado de grupos. Se observa que para el período y tipo de grupo seleccionado sólo aparece un grupo “Redes de Computadoras”. Dando doble clic sobre él se accede al detalle del mismo en la siguiente pantalla (figura 4.20).

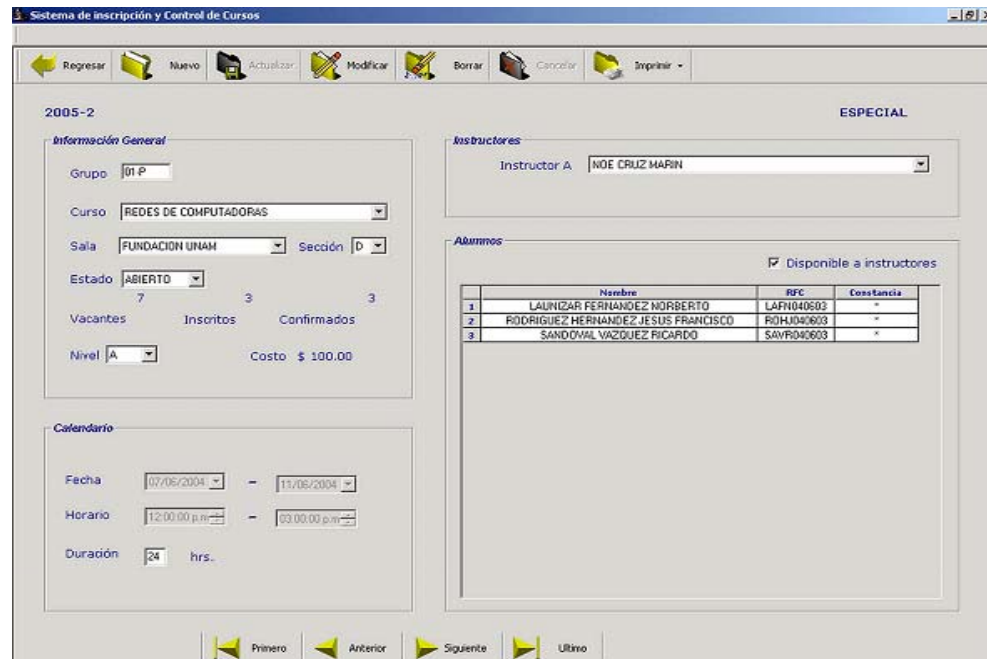


Figura 4.20. Pantalla de detalle para el grupo “Redes de Computadoras” de la figura anterior

Como se puede observar en la figura 4.19 existe para nuestro ejemplo, un grupo del tipo Especial y del Período 2005-2 con el curso Redes de Computadoras y con el nombre 01-P. En la siguiente figura 4.20 se observa la pantalla de detalle del grupo anterior a la cuál se llegó dando doble clic sobre el grupo en cuestión dentro del flexgrid en la vista anterior. En ésta pantalla de detalle, que es la misma para cuando se desea dar de alta un nuevo grupo, se observan todos los datos del grupo elegido incluyendo el instructor o instructores asociados, y los alumnos inscritos a él.

Siguiendo con la descripción de ésta pantalla de detalle, podemos apreciar que se tiene una barra de herramientas en la parte superior que contiene algunos botones cuya funcionalidad se describe en la siguiente tabla:

Botón	Función
Regresar	Este botón permite regresar a la interfaz principal (figura 4.15).
<b>Nuevo</b>	Este botón permite seguir creando grupos dentro de esta misma interfaz.
Modificar	Permite la modificación de algún grupo en cualquiera de los campos que se muestran (curso, sala, duración, instructor, etc).
Actualizar	Una vez que se han capturado las modificaciones correctamente, éste botón ejecuta la operación sobre la base de datos.
<b>Borrar</b>	Como su nombre lo indica, elimina por completo el grupo de la base de datos y por ende de la interfaz. Sólo se pueden eliminar aquellos grupos que no tengan alumnos inscritos.
Cancelar	Impide que se ejecute la acción previamente habilitada (Nuevo o Modificar).
Imprimir	Imprime distintos tipos de reporte según lo especificado. (ver subtema Constancias y Reportes en este mismo capítulo).

Tabla 4.3. Descripción de los botones de la Vista de detalle

Se pueden observar además de la barra de herramientas en esta misma pantalla, cuatro secciones sobresalientes. La primera de ellas es la de “Información General” la cuál contiene datos generales del grupo como lo son: nombre, curso, sala, sección de la sala, estado, número de vacantes, de alumnos inscritos y confirmados, el nivel y el costo.

La siguiente es la de “Calendario” que muestra las fechas inicial y final del curso, así como su horario y duración.

La de “Instructores” que muestra el o los instructores asignados para el grupo.

Finalmente aparece la de “Alumnos” que muestra los nombres de los alumnos asociados a dicho curso.

En la parte inferior se puede observar la barra de navegación entre registros, la cuál nos sirve para movernos entre los registros existentes. Tiene cuatro botones para acceder al primer elemento, al último, siguiente o anterior. Esta barra aparece al final de muchas pantallas más dentro del sistema sirviendo siempre al mismo objetivo.



Figura 4.21. Barra de navegación entre registros

#### 4.3.2.1.2 Interfaz de Inscripción

El paso siguiente es detallar la interfaz de inscripción que como se mencionó anteriormente, esta limitada a usarse como de sólo lectura excepto en el proceso referente a la inscripción de alumnos. Dicha interfaz está diseñada e implementada para ser utilizada por la persona encargada de la recepción en UNICA, por lo tanto, esta relacionada con la asignación de alumnos a los grupos ya existentes, es decir, se trata de la aplicación que inscribe a los alumnos a los grupos.

La primer pantalla es muy parecida a la aplicación de Administración ya que muestra datos muy parecidos, sin embargo, se puede observar que no tiene las pestañas de “Instructores”, “Cursos”, etc y que sólo contiene dos: la de “Grupos” y la de “Alumnos”.

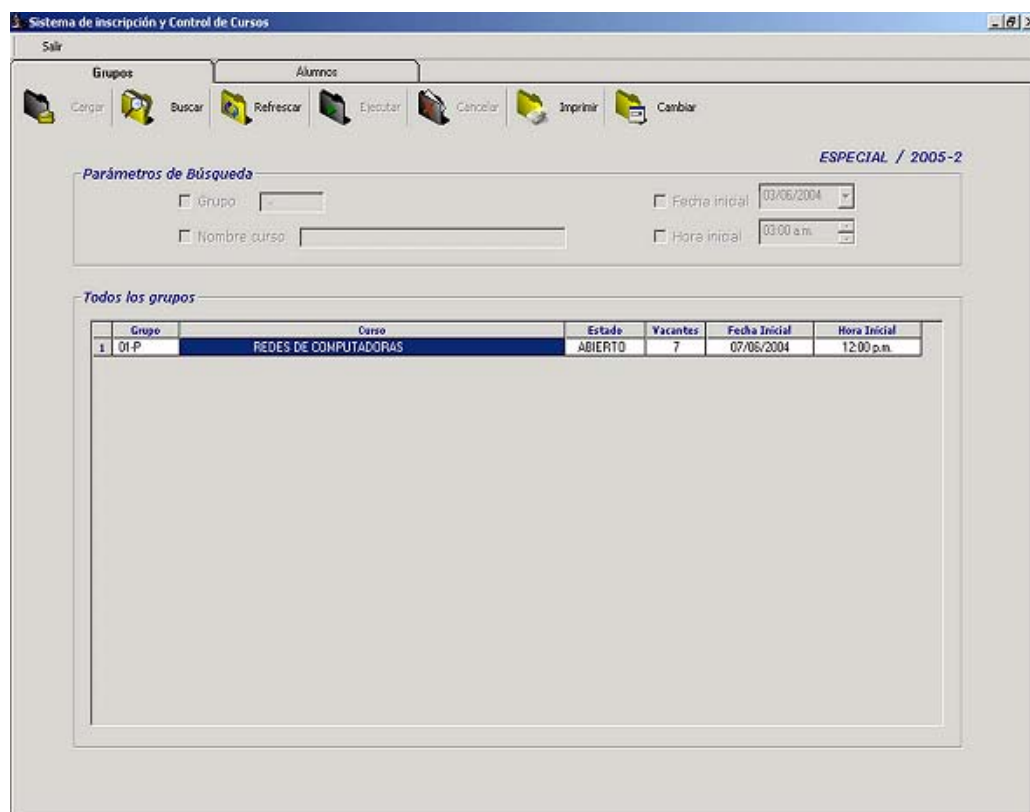


Figura 4.22. Interfaz de Inscripción del SICC



Tiene prácticamente los mismos controles a excepción del botón de “Nuevo” puesto que se trata de una aplicación de sólo lectura.

La pantalla de detalle también es muy parecida a la anterior de Administración pero sólo presenta dos botones el de “Regresar” que habilita nuevamente la pantalla anterior y el de “Imprimir” que genera ciertos reportes de acuerdo a los requerimientos.

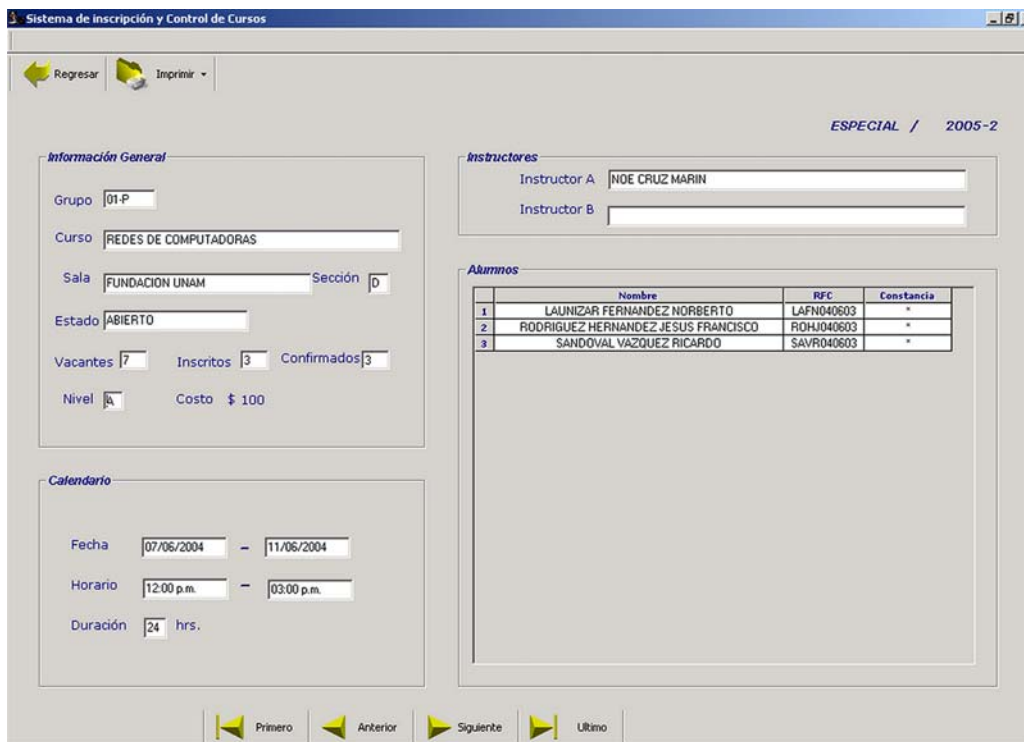


Figura 4.23. Interfaz de detalle para la vista de Inscripción.

La interfaz para la administración y control de la información de los alumnos tiene una apariencia similar a las anteriores sólo que ahora realiza operaciones sobre la información referente al alumnado.

Al igual que la de Grupos, presenta una pantalla principal en la que se muestran todos los alumnos registrados en un período en particular (aquí ya no importa el tipo de grupo seleccionado en la interfaz de grupos). Hace uso también de un flexgrid para mostrar a los alumnos. Si se desea entrar al detalle de alguno de ellos basta con hacer doble clic sobre el alumno elegido y automáticamente se habilita la ventana de detalle, la cuál también sirve para registrar un nuevo alumno, modificar alguno de sus datos e inclusive borrarlo del sistema, tal y como ocurría en la aplicación de Administración para grupos.

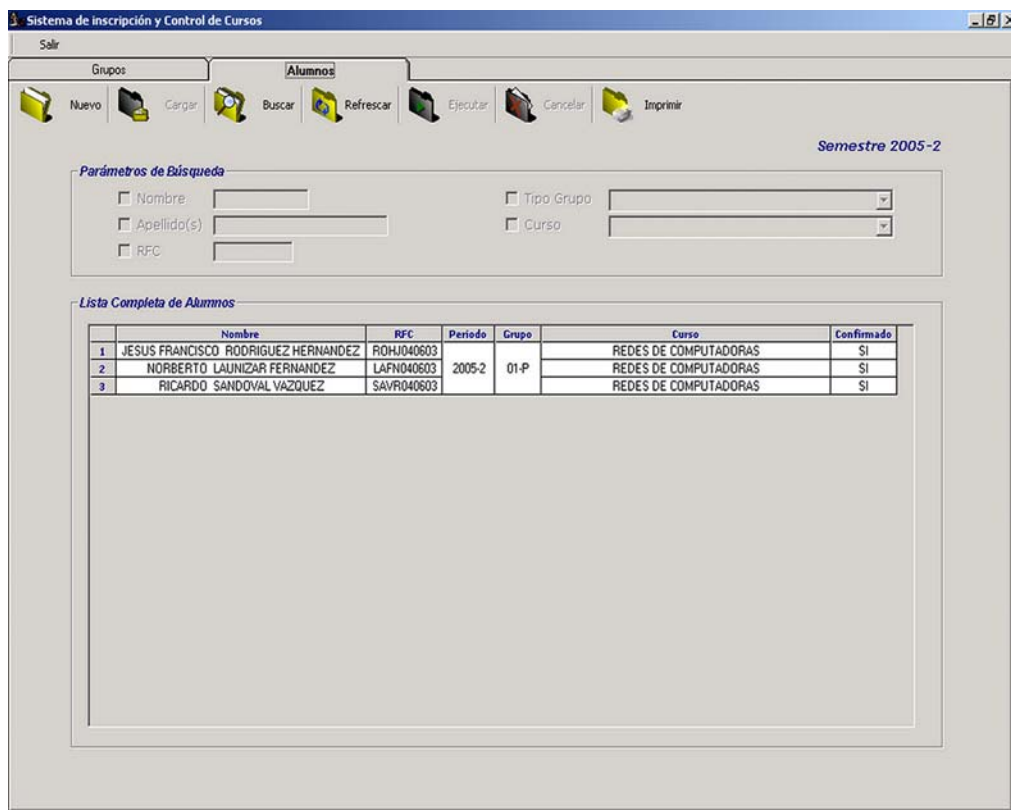


Figura 4.24. Pantalla principal de alumnos. Se observa en la parte superior un toolbar que contiene los mismos botones que la interfaz de grupos. Dichos botones realizan la misma función: cargar todos los alumnos, realizar búsquedas, refrescar los datos, etc. En el flexgrid se aprecia la siguiente información: nombre del alumno, RFC, período de cursos al que esta inscrito, el grupo y el curso del mismo, así como su estado (confirmado o no).

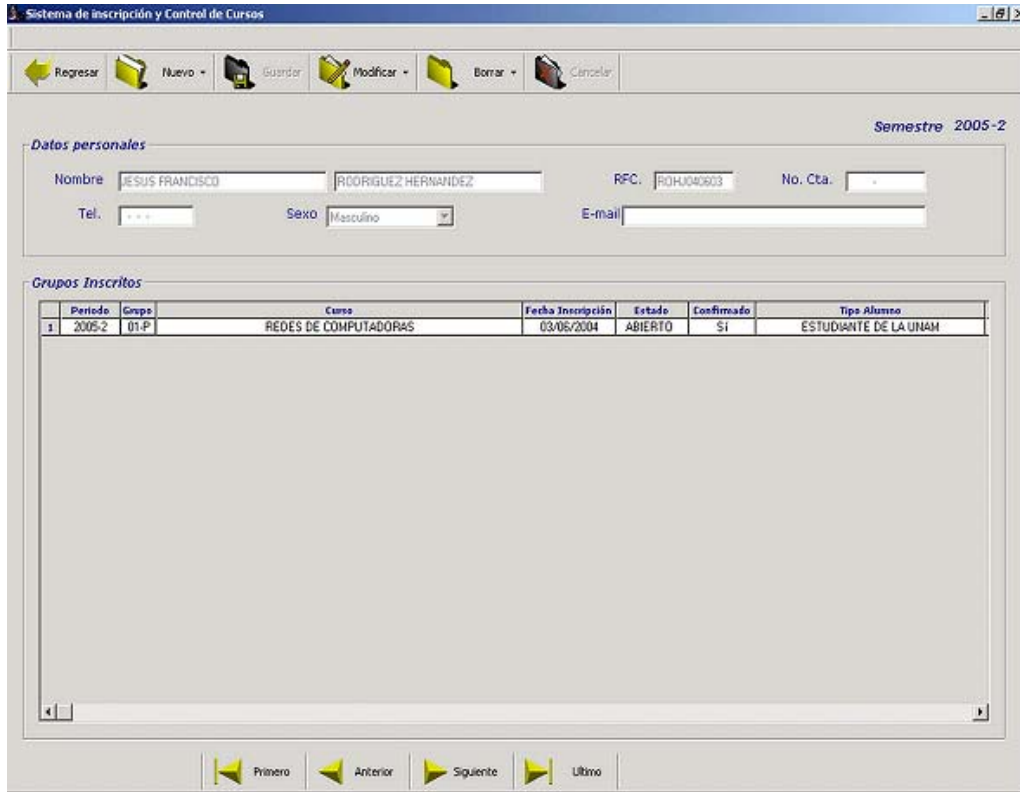


Figura 4.25. Pantalla de detalle del alumno. Una vez seleccionado un alumno en la pantalla anterior se puede entrar al detalle del mismo en ésta interfaz. Se observan sus datos personales, así como los cursos a los cuáles esta inscrito.

La pantalla principal de alumnos no difiere mayormente de su equivalente mostrada para grupos por lo que no se ahondará más en sus características. Sin embargo, la pantalla de detalle si presenta diferencias notorias con respecto a su simil de grupos por lo que vamos a profundizar un poco más en ella.

En la parte superior se observan seis botones cuya funcionalidad se muestra en el siguiente cuadro:

Botón	Función
Regresar	Permite volver a la interfaz principal.(figura 4.24)
*Nuevo	Permite realizar dos tipos de operaciones. La primera de ellas es la de capturar un nuevo alumno en la base de datos, para lo cuál habilita la sección de “Datos personales”. La otra operación que permite es la asignación de un curso para un alumno para lo que habilita la sección llamada “Grupos Inscritos”
<b>Guardar</b>	Ejecuta las operaciones de “Nuevo” o “Modificar” sobre la base de datos. Esto quiere decir, que cuando se oprime algún botón de los anteriores, dicha operación no se refleja en la base de datos hasta que se oprime el botón de “Guardar”.
*Modificar	Al igual que el botón “Nuevo”, permite realizar dos tipos de operaciones: modificar los datos de un alumno, o modificar los datos de algún curso asignado.
*Borrar	Lo mismo que para “Nuevo” y “Modificar” permite borrar a un alumno del sistema, o borrar algún curso que tenga asignado.
Cancelar	Este botón permite evitar que sea llevada a cabo una operación de captura de un nuevo alumno o asignación de curso, o alguna modificación si es que así se desea.

Tabla 4.4. Descripción de los botones de la interfaz de detalle de alumnos.

Los botones con asterisco presentan un submenú asociado y son aquellos que permiten hacer dos operaciones.

En el caso de las operaciones de Modificar Curso y Nuevo Curso se habilita una nueva interfaz para llevar a cabo dichos procesos, la cuál es mostrada en la siguiente figura:

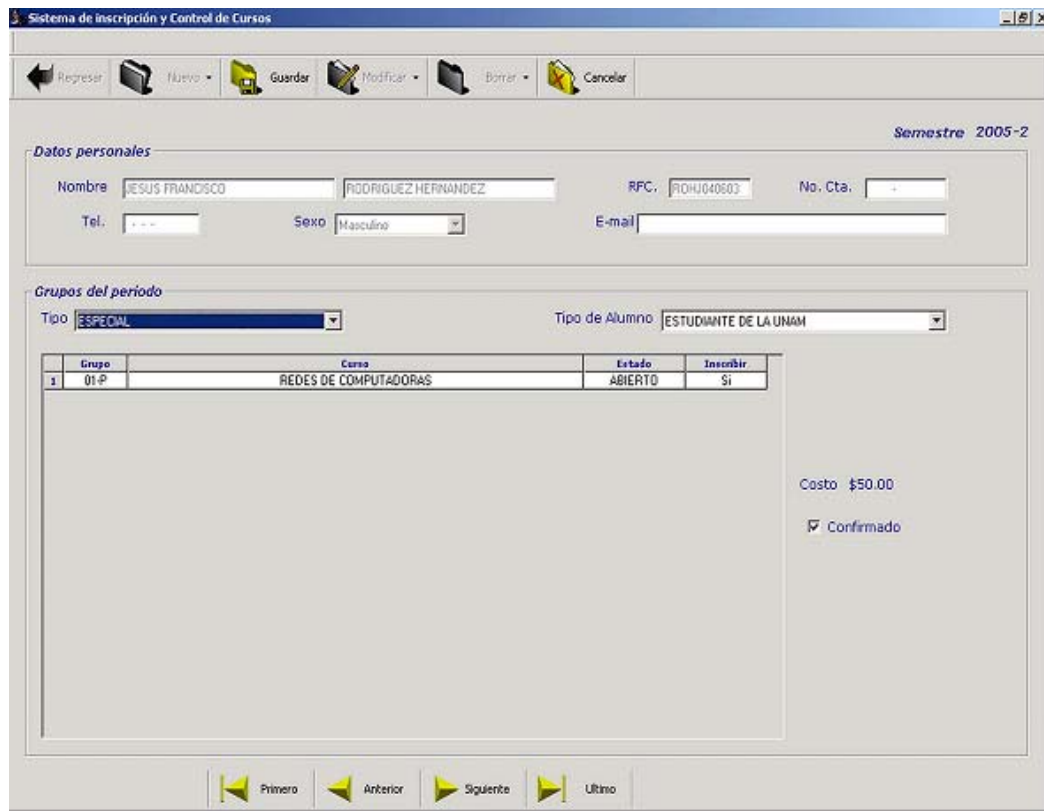


Figura 4.26. Vista de modificación o creación de cursos

Se puede observar que aparece un nuevo flexgrid con los datos del curso o cursos asignados al alumno. También aparecen dos catálogos nuevos. El primero de ellos contiene los tipos de grupos existentes para ese período. Esto es porque en una operación de Nuevo Curso, el alumno puede elegir un curso de cualquier tipo existente (intersemestral, semestral, fines de semana, etc.). El otro catálogo contiene a los tipos de alumno existentes (estudiante de la UNAM, académico de la UNAM, particular, etc.) y se utiliza para elegir el tipo correspondiente al alumno y conocer así el descuento a que tiene derecho.

### 4.3.2.2 Interfaz Módulo 4 (Apartado de cursos vía web)

La interfaz que permite el apartado vía web se realizó mediante php y algunas validaciones con javascript<sup>5</sup>. Cubre todo lo relacionado con el proceso de apartado de cursos que se describió en el capítulo 3.

La primer pantalla es la de Bienvenida y se muestra en la siguiente gráfica:



Figura 4.27. Interfaz para el apartado vía web

<sup>5</sup> JavaScript es un lenguaje de programación desarrollado por Netscape Corporation para permitir la ejecución de código dentro de las páginas en HTML. Se utiliza para conseguir interesantes efectos en las páginas web, comprobar la validez de la entrada de formularios, abrir y cerrar ventanas, cambiar dinámicamente el aspecto y los contenidos de una página, cálculos matemáticos sencillos y mucho más.

Esta primer ventana muestra los escudos respectivos de la Facultad de Ingeniería y de la Unidad de Servicios de Cómputo Académico. Dando clic en el botón “Iniciar” se accede a la siguiente ventana.




Figura 4.28. Selección de tipo de grupos para el apartado vía web

Se puede observar que aparece un listado con los tipos de grupo existentes para el período de cursos actual. De aquí se puede proceder de dos maneras. La primera de ellas es elegir un tipo de grupo y esperar a que nos muestre los cursos habilitados para él en la siguiente pantalla. Esto se logra marcando la opción correspondiente en la ventana de tipos y dando clic sobre el botón “Registrarse”. El otro camino nos lleva a obtener una reimpresión de nuestro comprobante si es que antes accedimos a esta pantalla y completamos el proceso de apartado de algún curso, para lo cuál damos clic en el botón “Verificar Comprobante”.

Veamos el primer caso. La pantalla que se muestra después de elegir el tipo de grupo es la siguiente:



SISTEMA DE INSCRIPCIONES Y CONTROL DE CURSOS (SICO) - Microsoft Internet Explorer


**Facultad de Ingeniería**  
 Unidad de Servicios de Cómputo Académico

## CURSOS

### SABADOS Y DOMINGOS

Extracurriculares 2004-1

ESTADO	GRUPO	CURSO	FECHA	HORARIO	DURACION (Horas)	SALA	ANTECEDENTES	%	Capo
OK	01-F	INTRODUCCION A LA COMPUTACION	29 de Junio al 9 de Julio	09:00-13:00	24	E	I	C	31
Cancelado	02-F	AUTOCAD BASICO	29 de Junio al 9 de Julio	09:00-13:00	24	E	III	C	30
OK	02-F	INTRODUCCION A OFFICE 2000	29 de Junio al 9 de Julio	09:00-12:00	24	E	III	D	5
OK	03-F	VISUAL BASIC 6.0	29 de Junio al 9 de Julio	09:00-13:00	24	E	III, V	D	6
Cancelado	03-F	UNIX BASICO	29 de Junio al 9 de Julio	09:00-13:00	24	H	II	C	29
Cancelado	05-F	PHP CON BASE DE DATOS	29 de Junio al 9 de Julio	09:00-13:00	24	F	III, VIII	C	25
OK	06-F	JAVA SERVER PAGES	29 de Junio al 9 de Julio	09:00-13:00	24	D	IX, V, XIII	C	21
OK	07-F	DOLPHI BASICO	29 de Junio al 9 de Julio	10:00-14:00	24	H	II, V	D	17
OK	07-F	INTRODUCCION A OFFICE 2000	29 de Junio al 9 de Julio	09:00-13:00	24	F	III	C	29
OK	08-F	FLASH MX	29 de Junio al 9 de Julio	09:00-12:00	24	E	III	D	25
OK	08-F	HTML CON FLASH	29 de Junio al 9 de Julio	09:00-13:00	24	E	III, IV	C	29
OK	09-F	LENGUAJE C BASICO	29 de Junio al 9 de Julio	09:00-13:00	24	G	V	C	25
OK	09-F	ADMINISTRACION LINUX	29 de Junio al 9 de Julio	09:00-13:00	24	H	VII	D	3
OK	10-F	VISUAL BASIC ORIENTADO A OBJETOS CON COMC	29 de Junio al 9 de Julio	09:00-13:00	24	G	VI, VIII, XII	D	29
OK	10-F	VISUAL BASIC CON BASES DE DATOS	29 de Junio al 9 de Julio	09:00-13:00	24	H	VI, VIII	C	31
OK	11-F	PHP CON BASE DE DATOS	29 de Junio al 9 de Julio	09:00-12:00	24	F	III, VIII	D	21
OK	11-F	DESIGN DE PROGRAMAS WEB CON HTML	29 de Junio al 9 de Julio	09:00-13:00	16	A	IV	B	9
OK	12-F	JAVA CON BASE DE DATOS	29 de Junio al 9 de Julio	09:00-13:00	24	C	IX	D	1
OK	12-F	VISUAL BASIC .NET	29 de Junio al 9 de Julio	09:00-13:00	24	G	VI, VIII, XII	C	12
OK	13-F	DESIGN DE PROGRAMAS WEB CON HTML	29 de Junio al 9 de Julio	09:00-13:00	16	A	IV	B	1
Cancelado	13-F	COMPUTACION PARA NEGOCIOS	20 de Junio al 9 de Julio	09:00-11:00	0	D	I	A	0
Cancelado	14-F	COMPUTACION PARA NEGOCIOS	29 de Junio al 9 de Julio	09:00-11:00	0	D	I	D	0

**Inscripciones**  
En la Unidad de Servicios de Cómputo Académico, Edificio Principal, Facultad de Ingeniería, UNAM 20 de Mayo del 2004

**Horario de inscripción**  
09:00 a 20:00 hrs. de Lunes a Viernes

A	B	C	D	E
1300	1800	2100	2500	500

Descuento del 50% a estudiantes y trabajadores de la UNAM, otras Instituciones Educativas 25%.

Recibe un 10 % de descuento adicional al inscribirte la primer semana.

**\*ANTECEDENTES**

- I NINGUNO
- II CONOCIMIENTOS BASICOS DE COMPUTACION
- III MANEJO DE AMBIENTE WINDOWS
- IV CONOCIMIENTOS BASICOS DE INTERNET
- IX FUNDAMENTOS DE JAVA
- V CONOCIMIENTOS BASICOS DE PROGRAMACION
- VI MANEJO DE VISUAL BASIC
- VII MANEJO DE LINUX
- VIII MANEJO DE BASES DE DATOS
- X ADMINISTRACION UNIX
- XI MANEJO DE SQL
- XII PROGRAMACION ORIENTADA A OBJETOS
- XIII CONOCIMIENTOS DE HTML
- XIV MANEJO DE AUTOCAD

[regresar](#)
[registrarse](#)

Figura 4.29. Calendario de cursos para un tipo de grupo.



En esta figura se aprecia el calendario de cursos para el tipo de grupo elegido y para el período actual. Los datos que se muestran son: estado del curso (si esta disponible aparece una casilla de elección para marcar e indicar que se quiere apartar), nombre del grupo, nombre del curso, fecha, horario, duración, sala, antecedentes, costo y cupo. Los antecedentes y el costo aparecen con una clave cuyo significado aparece en la parte inferior de la pantalla. También en la parte inferior aparece el lugar donde se realizan las inscripciones (UNICA Edif. Principal de la Fac. de Ingeniería) y la fecha a partir de la cual inician dichas inscripciones al igual que el horario. Finalmente aparecen los descuentos que existen para los distintos tipos de alumnos (particulares, académicos de la UNAM, y más.).

Una vez elegidos los cursos que se desean (dos como máximo), se procede al registro de los mismos en la siguiente pantalla así como de los datos personales del alumno.

Sistema de Inscripciones y Control de Cursos(SICC) - Microsoft Internet Explorer

 **Facultad de Ingeniería**

Unidad de Servicios de Cómputo Académico 

**Bienvenidos al Sistema de Inscripciones a Cursos**

**Registro**

Los campos marcados con asterisco (\*) son obligatorios

*NOMBRE:	*APELLIDOS:
José Armando	Flores
No. CTA (UNAM):	*R.F.C.:
095014872	FOOA270879
Teléfono:	E-mail:
55876578	jgoues@unibo.com
*Sexo:	Tipo de alumno
Femenino: <input type="checkbox"/> Masculino: <input checked="" type="checkbox"/>	BECARID

Su selección Fue: INTRODUCCION A LA COMPUTACION

Su selección Fue: INTRODUCCION A OFFICE 2000

**limpiar** **enviar** 

Figura 4.30. Captura de datos del alumno para el proceso de apartado

A continuación viene una pantalla para que el usuario verifique si sus datos están correctos, en caso afirmativo se procede al guardado de los mismos en la base de datos.



Figura 4.31. Verificación de datos en el proceso de apartado

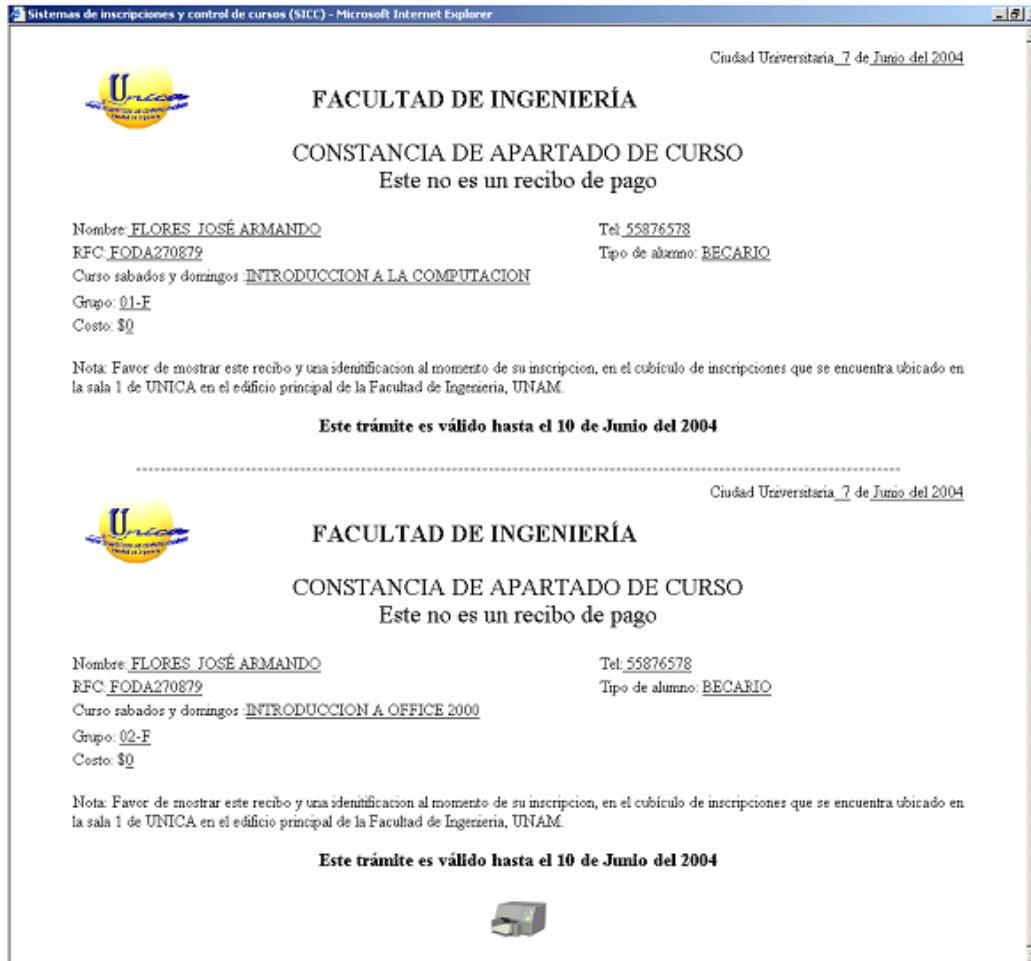


Figura 4.32. Comprobantes de apartado

Con la impresión de éste recibo termina el proceso de apartado en condiciones normales. En dicho recibo, como se puede apreciar, aparecen los datos personales del alumno (nombre, RFC y teléfono). También el tipo de alumno que le corresponde, en este caso BECARIO, el curso elegido, el grupo y el costo (para un becario es de \$0.00).

Finalmente aparece la leyenda que avisa al alumno que debe confirmar su asistencia al curso en las salas de UNICA ubicadas en el Edif. Principal de la Fac. de Ingeniería.

Ahora vamos a ver el caso en el que el usuario desea reimprimir su comprobante de apartado. Dicho proceso comienza accediendo nuevamente al sitio web del sistema de apartado (figura 4.27). Una vez aquí se oprime el botón “Iniciar” y la siguiente pantalla nos da la posibilidad de acceder a la reimpresión de los comprobantes de apartado (figura 4.28) a través del botón “Verificar Comprobante”.



Figura 4.33. Recuperación de comprobantes de apartado

Como se puede observar en esta pantalla se pide el nombre y RFC del usuario para validar su existencia en el sistema. Una vez ingresados dichos datos se continúa a través del botón “Enviar” y si los datos son correctos la siguiente pantalla mostrará el comprobante de apartado para el usuario en cuestión, en caso contrario se muestra el mensaje de error correspondiente.

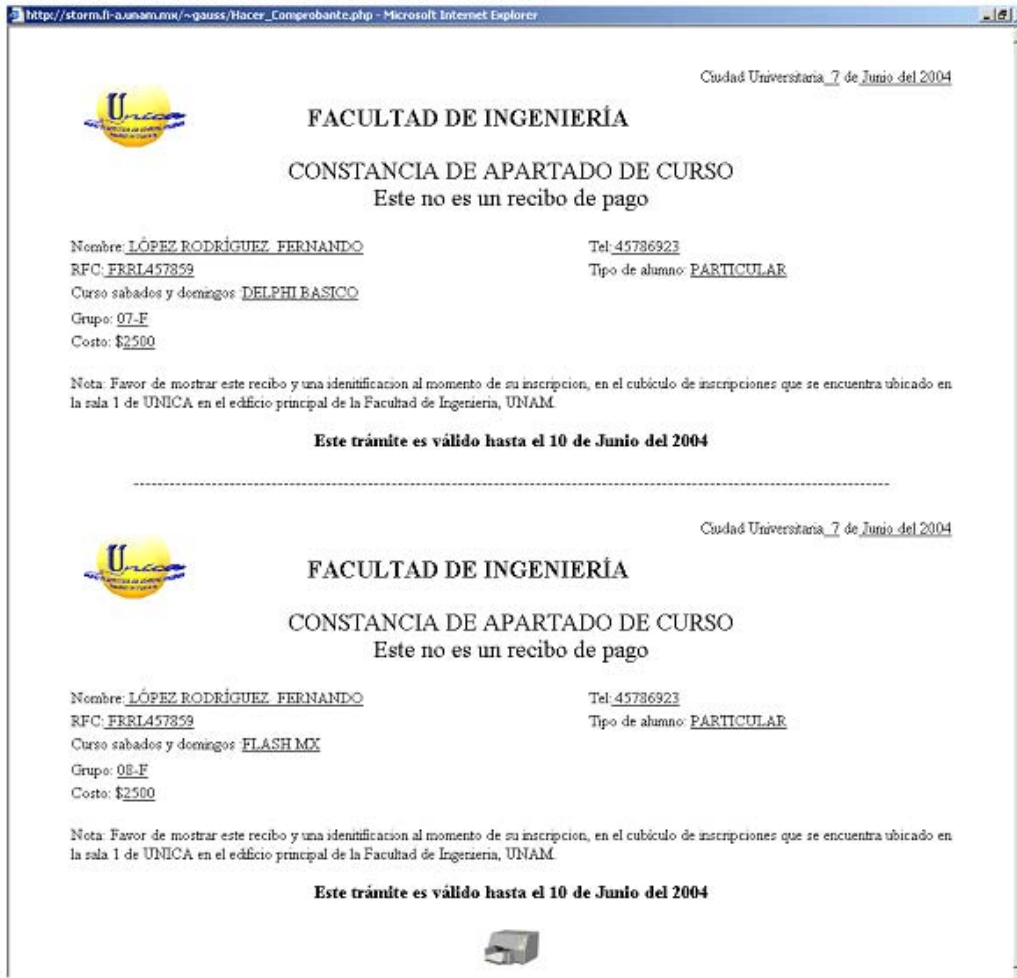


Figura 4.34. Comprobante de apartado recuperado

Los diferentes mensajes de error que ocurren de acuerdo a las condiciones temporales de los cursos, se tratarán en el tema de Evaluación y Resultados del SICC en el siguiente capítulo.

## 4.4 Generación de Reportes y Constancias

Para la creación de Reportes y Constancias en el SICC se utilizó el generador visual de informes Seagate Crystal Reports 8. Dicha herramienta cubre casi cualquier tipo necesidad de un programador: desde simples listas hasta complejas distribuciones de datos que incluyen gráficos, mapas y múltiples estadísticas, pasando por componentes de servidor web.

Aparte del tradicional programa independiente para el diseño de los informes, en la versión 8 se incluye un Diseñador ActiveX que permite su creación directamente en las diferentes herramientas de Visual Studio. Esto proporciona mayor comodidad, ya que no es necesario abandonar el entorno de desarrollo de Visual Basic en ningún momento.

El producto proporciona muchas características y utilidades, como son el diseñador de consultas, un editor de fórmulas, una completa API para generación de informes en tiempo de ejecución, un componente para servidores Web que permite la generación de informes para navegadores, además de multitud de asistentes y "Report Experts" para facilitar el trabajo.

Las características antes mencionadas convierten a Crystal Reports en un generador de informes adecuado para los requerimientos del SICC.

Para poder trabajar con él, se requiere instalar la versión Developer que incluye el diseñador ActiveX para Visual Studio.

Ahora bien, para la implementación de los reportes dentro del SICC se diseñó una función dentro del código del formulario que agrega Crystal Reports cuando se crea un nuevo reporte dentro de Visual Basic. La función crea una nueva conexión a la base de datos. Dicha conexión es utilizada por todos los reportes. Se crean también nuevos objetos recordset para cada reporte. Todo el código está contenido dentro de un *Select Case*<sup>6</sup>, el cuál nos permite

---

<sup>6</sup> Select Case. Estructura de decisión que proporciona Visual Basic y se utiliza cuando existe un gran número de instrucciones a evaluar.

acceder al reporte que se desee en el momento adecuado a través de un número de reporte. Así, todo el código de los reportes se encuentra en una sola forma y sólo se manda llamar a la función con el número de reporte deseado como parámetro.

La siguiente gráfica muestra un fragmento de dicha función:

```
Public Sub Reportes(No_Reporte As Integer, Optional parametro As Variant)
If cnreporte.State = adStateClosed Then cnreporte.Open
Select Case No_Reporte
Case 1 'Lista de Cursos
Set Report = New crptCursos
Dim rscursos As New ADODB.Recordset
Me.Caption = "Lista de Cursos"
If parametro Then
rscursos.Source = "select nombre, requerimientos from curso order by
nombre"
Else
rscursos.Source = strSQLcurso
End If
rscursos.ActiveConnection = cnreporte
rscursos.Open
Report.Database.SetDataSource rscursos
Report.PaperSize = Carta
Report.PaperSource = FormSelect
Report.PaperOrientation = crPortrait

Case 2 'Lista de grupos
Set Report = New crptgrupos
Dim rsgrupos As New ADODB.Recordset
Me.Caption = "Lista de Grupos"
strSQL = "select distinct grupo.nombre,grupo.curso,
```

```
grupo.estado,grupo.vacantes,grupo.fecha_inicial,grupo.hora_inicial,grupo.id_grupo,  
grupo.fecha_final,grupo.confirmados,grupo.inscritos, grupo.id_seccion,grupo.nivel,  
grupo.tipo, grupo.periodo,grupo.duracion,grupo.hora_final,costo_grupo.costo  
from  
grupo,tipo_grupo,periodo where grupo.periodo="" +  
frmperiodo.cboPeriodo.List(frmperiodo.cboPeriodo.ListIndex) + "" and  
grupo.periodo=periodo.id_periodo and tipo_grupo.comentarios="" +  
frmperiodo.cboTipo.List(frmperiodo.cboTipo.ListIndex) + "" and  
tipo_grupo.tipo=grupo.tipo and costo_grupo.nivel=grupo.nivel and  
costo_grupo.tipo=grupo.tipo and costo_grupo.periodo=grupo.periodo"  
If rsgrupos.State = adStateOpen Then rsgrupos.Close  
If parametros_grupo Then  
    rsgrupos.Source = strSQLparametros & " order by nombre"  
Else  
    rsgrupos.Source = strSQL & " order by nombre"  
End If  
rsgrupos.ActiveConnection = cnreporte  
rsgrupos.Open  
Report.Database.SetDataSource rsgrupos  
Report.PaperSize = Carta  
Report.PaperSource = FormSelect  
Report.PaperOrientation = crPortrait  
.....  
.....  
.....
```

Figura 4.35. Fragmento de código de la función Reportes implementada para la impresión de reportes y constancias.



Como se puede observar, a grandes rasgos para generar un reporte se crea el recordset correspondiente, se le pasa la conexión activa (única para todos los reportes), se le asigna el query correspondiente y se ejecuta la consulta sobre la base de datos.

Siguiendo con la especificación de requerimientos, se implementaron 8 reportes de la forma antes explicada, mismos que se describen en el siguiente cuadro:

Reporte	Descripción	Aplicación que lo genera
<b>Grupos</b>	Muestra un listado con los grupos creados para un determinado período y tipo de grupo. Los datos que aparecen son el nombre del grupo, el curso, el estado, el número de vacantes, la fecha de inicio y la hora de inicio. (figura 4.36)	SICC-Administración
<b>Lista de Alumnos por Grupo</b>	Lista de alumnos inscritos a un determinado grupo. Los datos que contiene son: nombre del curso, nombre del grupo, fecha de inicio y de terminación; nombre de los instructores; nombre del alumno, apartado de asistencias, evaluación y calificación final. (figura 4.37)	SICC-Administración
<b>Gafetes</b>	Muestra un listado con los gafetes que se otorgan a los alumnos una vez inscritos a un curso. Los datos que contiene el gafete son: nombre del alumno, nombre del grupo, curso, fecha de inicio y fecha de terminación. (figura 4.38)	SICC-Administración
<b>Constancias</b>	Muestra un listado con aquellos alumnos que alcanzaron constancia en un determinado curso.	SICC-Administración
<b>Diplomas Alumno</b>	Muestra el diploma que se entrega a los alumnos que acreditaron un curso (figura 4.39).	SICC-Administración
<b>Diplomas Instructor</b>	Muestra el diploma que se otorga a los instructores por haber impartido algún curso.	SICC-Administración
<b>Lista de Alumnos General</b>	Despliega una lista con los nombres de todos los alumnos inscritos en un determinado período de cursos: los datos que aparecen son los siguientes: período de cursos, nombre y RFC del alumno; grupo y curso al que está inscrito y si obtuvo constancia o no. (figura 4.40)	SICC-Inscripción
<b>Recibo de Inscripción</b>	Muestra un recibo donde aparecen los datos del alumno así como la información referente al curso elegido. (figura 4.41)	SICC-Inscripción

Tabla 4.5. Descripción de los reportes y constancias implementados en el SICC.

Lista de Grupos

Preview


**FACULTAD DE INGENIERÍA**  
**SECRETARÍA GENERAL**  
**UNIDAD DE SERVICIOS DE**  
**CÓMPUTO ACADÉMICO**

## GRUPOS

	Grupo	Curso	Estado	Vacantes	Fecha Inicial	Hora Inicial
1	01-F	INTRODUCCION A LA COMPUTACION	ABIERTO	30	28 / 06 / 2004	9:00:00 am
2	02-F	AUTOCAD BASICO	CERRADO	30	28 / 06 / 2004	9:00:00 am
3	03-F	LINUX BÁSICO	CERRADO	29	28 / 06 / 2004	9:00:00 am
4	04-F	JAVA SCRIPT	CANCELADO	10	28 / 06 / 2004	9:00:00 am
5	05-F	PHP CON BASE DE DATOS	CERRADO	35	28 / 06 / 2004	9:00:00 am
6	06-F	JAVA SERVER PAGES	ABIERTO	19	28 / 06 / 2004	9:00:00 am
7	07-F	INTRODUCCION A OFFICE 2000	ABIERTO	29	28 / 06 / 2004	9:00:00 am
8	08-F	HTML CON FLASH	ABIERTO	28	28 / 06 / 2004	9:00:00 am
9	09-F	LENGUAJE C BÁSICO	ABIERTO	25	28 / 06 / 2004	9:00:00 am
10	10-F	VISUAL BASIC CON BASES DE DATOS	ABIERTO	31	28 / 06 / 2004	9:00:00 am
11	11-F	DISEÑO DE PAGINAS WEB CON HTML	ABIERTO	8	28 / 06 / 2004	9:00:00 am
12	12-F	VISUAL BASIC .NET	ABIERTO	12	28 / 06 / 2004	9:00:00 am
13	13-F	COMPUTACION PARA NIÑOS	CERRADO	0	28 / 06 / 2004	9:00:00 am

1

10/06/2004

Figura 4.36. Reporte de Grupos



Figura 4.37. Lista de Alumnos por Grupo

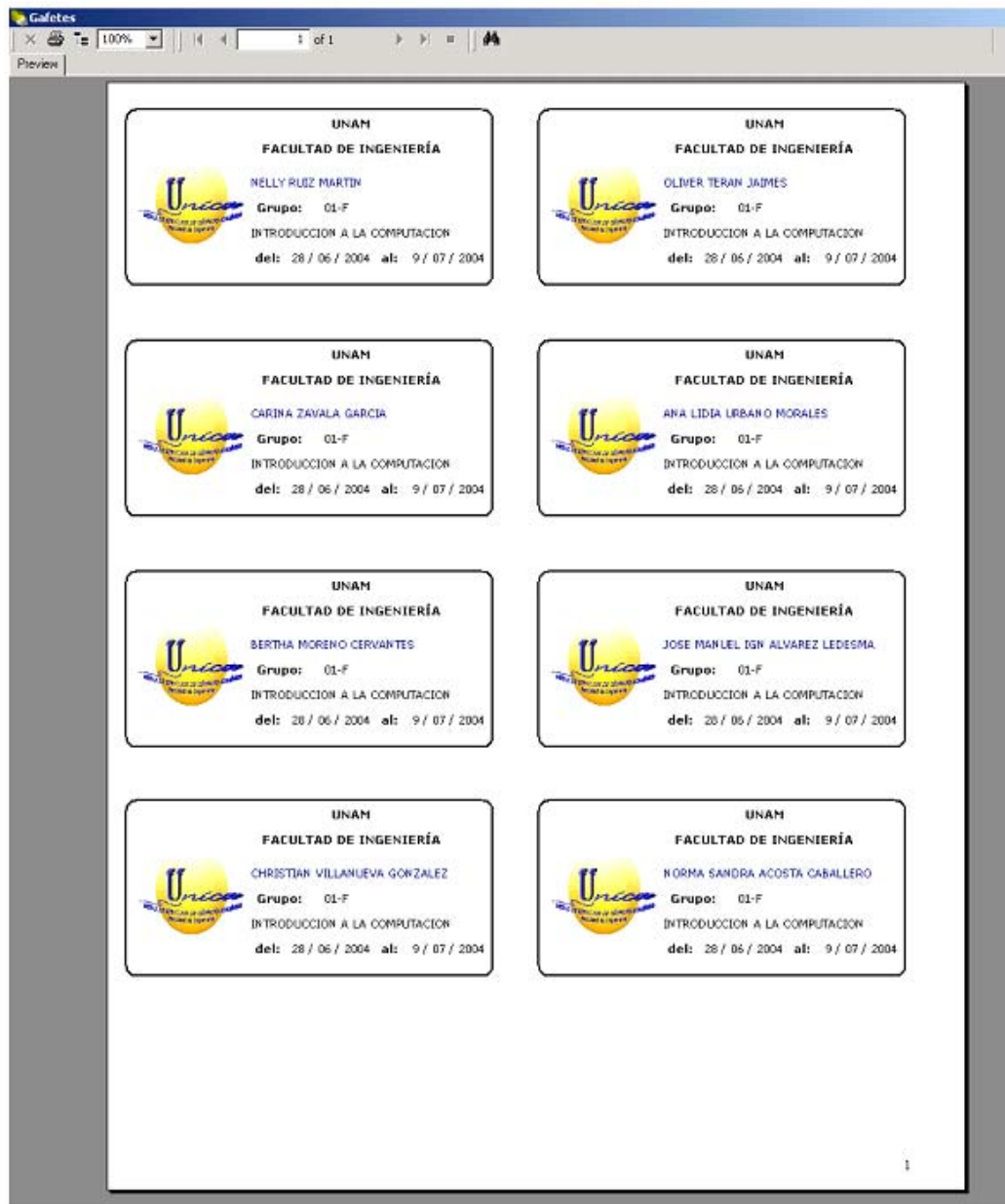


Figura 4.38. Gafetes



Figura 4.39. Diplomas de alumnos



Figura 4.40. Lista de Alumnos General

**UNICA**  
UNIDAD DE SERVICIOS DE COMPUTACIÓN ACADÉMICA

**RECIBO**

FACULTAD DE INGENIERÍA  
SECRETARÍA GENERAL  
UNIDAD DE SERVICIOS DE  
CÓMPUTO ACADÉMICO

Bueno Por: \$ 1,250.00  
Fecha: 10 de Junio del 2004

Recibimos de: MARIO ALBERTO VILCHIS RODRIGUEZ la cantidad de:  
\$ 1,250.00 Un mil doscientos cincuenta pesos por concepto de

Inscripción al curso: AUTOCAD BASICO Sala: G Grupo: 02-1

a efectuarse del 14 de Junio al 25 de Junio del 2004

Tipo de Alumno: ESTUDIANTE DE LA UNAM Horario: 10:00:00am a 1:00:00pm

Numero de cuenta: \_\_\_\_\_

telefono: \_\_\_\_\_ email: \_\_\_\_\_

**INTERSEMESTRALES**

Figura 4.41. Recibo de Inscripción

No todos los reportes son mostrados como en el caso de las Constancias y el Diploma para Instructores. Esto es porque en el primero, se trata sólo de una lista de los alumnos que obtuvieron constancia y no muestra mayor detalle. Y en el caso del diploma para instructores es idéntico al que se otorga a los alumnos sólo que difiere en la leyenda que aparece abajo del nombre, en este caso del instructor.

## Evaluación y Resultados del SICC

### 5.1 Pruebas del usuario final

#### 5.1.1 Objetivo de la fase de pruebas

El objetivo de la fase de pruebas de un sistema es el de detectar todo posible malfuncionamiento antes de que entre en producción. Un error detectado en ésta etapa puede ser costoso de reparar; pero siempre es peor que el error le aparezca al usuario final.

En este sentido, un plan de pruebas será de mayor calidad cuantos menos errores queden por descubrir tras haberlo pasado. Y, viceversa, si un programa aún tiene muchas fallas tras haber superado un plan de pruebas, diremos que dicho plan es de poca calidad.

#### 5.1.2 Metodología de pruebas

Existen dos enfoques principales de pruebas de software: pruebas de caja negra y pruebas de caja blanca.



### a) Pruebas de caja blanca

La prueba de caja blanca, denominada a veces prueba de caja de cristal es un método de diseño de casos de prueba que usa la estructura interna del programa para elegir los casos de prueba.

Mediante los métodos de prueba de caja blanca, el ingeniero del software puede obtener casos de prueba que:

- Garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo.
- Ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa.
- Ejecuten todos los bucles en sus límites y con sus límites operacionales.
- Ejerciten las estructuras internas de datos para asegurar su validez.<sup>1</sup>

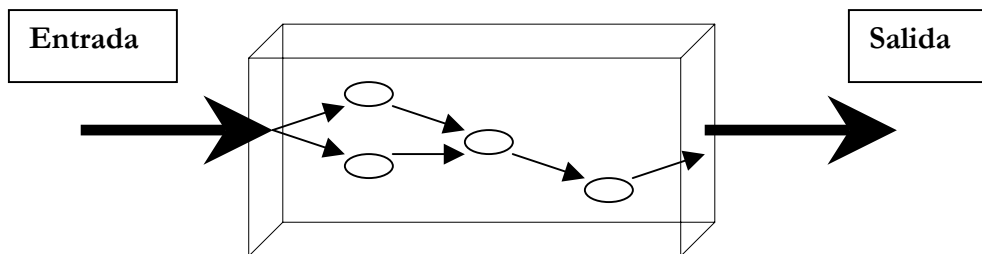


Figura 5.1. Pruebas de caja blanca

### b) Pruebas de caja negra

Las pruebas de caja negra se centran en los requisitos funcionales del software.

Permiten al ingeniero de software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa.

La prueba de caja negra intenta encontrar errores de los siguientes tipos:

---

<sup>1</sup> Pressman, Roger S. 1998. Ingeniería del Software. México, McGraw-Hill. P.305

- Funciones incorrectas o ausentes
- Errores de interfaz.
- Errores en estructuras de datos o en acceso a bases de datos externas.
- Errores de rendimiento
- Errores de inicialización y de terminación.<sup>2</sup>

A diferencia de la prueba de caja blanca, que se lleva a cabo previamente en el proceso de prueba, la prueba de caja negra tiende a aplicarse durante fases posteriores de la prueba, ya que la prueba de caja negra ignora intencionalmente la estructura interna del sistema y centra su atención en el campo de la información.

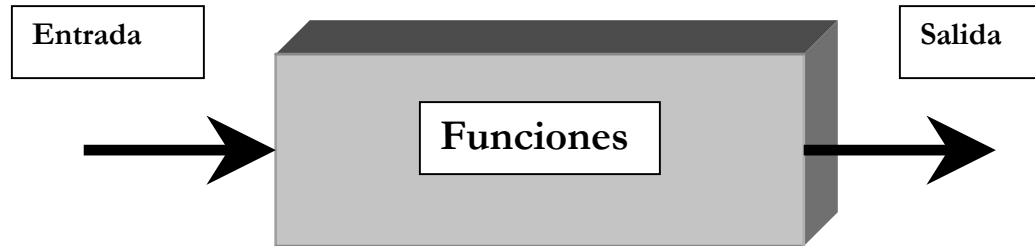


Figura 5.3. Pruebas de caja negra

---

<sup>2</sup> *ibid op.Cit* 315.

## 5.2 Plan de pruebas para el SICC

El diseño de pruebas para el SICC se basó principalmente en la técnica de pruebas de caja negra debido a las características del sistema mismo. Es decir, las pruebas estaban enfocadas principalmente en que el sistema hiciera todo lo que se esperaba de él y que se había planteado en la especificación de requerimientos. Sin embargo, también se realizaron pruebas de caja blanca en aquellos segmentos de código que de alguna manera tenían caminos o vertientes con una ocurrencia difícil o poco probable. Desafortunadamente, las pruebas de caja blanca no pudieron ser muy amplias debido a la falta de tiempo y recursos.

### 5.2.1 Pruebas de caja negra

Tomando como punto de partida la especificación de requerimientos se procedió a diseñar el siguiente plan de pruebas de caja negra para el sistema.

Una vez creado un período y asignado éste último como el período actual de cursos, se crearán grupos de prueba con las características que aparecen en la especificación de requerimientos (ver Capítulo 1 subtema 1.3.2 **Control de la información de los grupos**).

Para poder verificar la existencia de la bitácora requerida de grupos de acuerdo al período de cursos, se crearán algunos grupos para distintos períodos y se accederá a ellos para observar que los datos pueden ser recuperados correctamente.

Algo similar ocurrirá en el caso de los alumnos, es decir, una vez creados grupos en los distintos períodos de prueba se les asignarán alumnos para comprobar que dicho proceso de inscripción es exitoso y además, para poder crear el historial pedido de alumnos y comprobar que el acceso y recuperación de los datos es satisfactorio. Para ello, se darán de alta alumnos de prueba con los datos requeridos y se comprobará que se pueden inscribir a un grupo determinado siempre y cuando existan vacantes. Además, se realizarán algunas modificaciones

de los datos de los alumnos, y se darán de baja alumnos para verificar la correcta implementación de dichas operaciones.

Para probar que el sistema puede crear, modificar o borrar tipos de grupo según se requiera, se darán de alta algunos tipos de grupo y se realizarán modificaciones e incluso algunos de ellos serán eliminados.

Las pruebas relacionadas con los reportes consistirán básicamente en su generación e impresión para comprobar con ello, que cumplen con los requisitos especificados de manera puntual.

El paso siguiente dentro de éste plan de pruebas, consiste en la modificación de los datos de los grupos y la eliminación de algunos de ellos comprobando que la aplicación sólo podrá eliminar aquellos grupos en los cuáles no existan alumnos inscritos.

En lo referente a la búsqueda de grupos, se realizarán búsquedas de acuerdo a los criterios establecidos en la especificación de requerimientos. (ver Capítulo 1 subtema 1.3.2 **Control de la información de los grupos**).

### 5.2.2 Pruebas para la Aplicación web

Para comprobar que el sistema es capaz de desplegar los cursos vía Internet, se accederá a la página desarrollada con ese propósito y se realizará el apartado de cursos.

En una primer prueba se apartarán dos cursos y se verificará la generación del comprobante de inscripción. Después se tratará de apartar un tercer curso para comprobar que el sistema no permite dicha operación.

Por otro lado, el sistema debe dar de baja aquellos alumnos que no hayan confirmado su asistencia a los cursos que hayan apartado tres días antes de que comiencen dichos cursos (sin contar fines de semana y días festivos). Para verificar lo anterior, se realizará una prueba modificando las fechas de inicio e inscripción de los cursos desde la base de datos y se observará el comportamiento del sistema.

Además de la operación antes mencionada, existen otras que el sistema debe realizar de acuerdo a la fecha en la que se encuentre (ver Capítulo 3 subtema 3.5 **Diseño e implementación del SICC mediante OMT y UML figura 3.13**).

Para verificar esto, se realizarán los cambios de fecha debidos en la base de datos. Específicamente se realizarán las siguientes pruebas:

Tomando como punto de referencia la fecha actual se designará una fecha de inscripción a los cursos mayor y se observará la respuesta del sistema (ver Capítulo 3 subtema 3.5 **Diseño e implementación del SICC mediante OMT y UML tabla 3.3**).

Posteriormente se cambiará la fecha de inscripción a los cursos por una menor a la fecha actual y una fecha de inicio de cursos también menor a la actual y se observará la respuesta del sistema.

Finalmente se capturará una fecha final de cursos menor a la fecha actual y se observarán los resultados.

### 5.2.3 Pruebas de caja blanca

Se realizaron pruebas de caja blanca en algunos métodos y funciones implementadas para el sistema. No fue posible aplicar pruebas de caja blanca a toda la estructura interna del sistema debido a su gran extensión y por la limitante en cuanto al recurso tiempo con que siempre se contó durante el desarrollo del sistema.

Los métodos en los cuáles se aplicó pruebas de caja blanca fueron aquellos que por su misma estructura poseen caminos con poca probabilidad de ocurrencia. Por lo tanto, se intentó ejercitarlos en todos esos caminos para asegurarnos que funcionaban correctamente.

A continuación se describe la aplicación de la prueba de caja blanca para la operación de borrar grupos.

```
Case "Borrar"      'Borrar grupos
  If mfgAlumno.TextMatrix(1, 1) = "" Then
    respuesta = MsgBox("¿Desear realmente borrar el grupo actual y todos sus
    datos?", vbExclamation + vbYesNo + vbDefaultButton2)
  If respuesta = 6 Then
    grupos.Eliminar (grupos.mgrupos(registro_actual_grupo).id)
    grupos.mgrupos.Remove (registro_actual_grupo)
  If registro_actual_grupo = 1 And grupos.mgrupos.Count = 0 Then
    frmSICC.mfgGrupos.TextMatrix(1, 1) = vbNullString
    frmSICC.mfgGrupos.TextMatrix(1, 2) = vbNullString
    frmSICC.mfgGrupos.TextMatrix(1, 3) = vbNullString
    frmSICC.mfgGrupos.TextMatrix(1, 4) = vbNullString
    frmSICC.mfgGrupos.TextMatrix(1, 5) = vbNullString
    frmSICC.mfgGrupos.TextMatrix(1, 6) = vbNullString
```

```
Else
    frmSICC.mfgGrupos.RemoveItem (registro_actual_grupo)
End If
Call frmSICC.actualizar_indices_grupos
If registro_actual_grupo > 1 Then
    registro_actual_grupo = registro_actual_grupo - 1
Else
    registro_actual_grupo = 1
End If
If registro_actual_grupo = 1 And frmSICC.mfgGrupos.TextMatrix(1, 1)
= vbNullString Then
    hay_registros_grupo = False
    frmSICC.Show
    frmGrupos.Hide
Else
    Call grupo_registro(registro_actual_grupo)
End If
MsgBox "El registro ha sido borrado", vbInformation + vbOKOnly

Else
End If
Else
MsgBox "El grupo contiene alumnos " & vbCrLf & _
    "no puede ser borrado.", vbExclamation + vbOKOnly, "SICC"
End If
```

Figura 5.3. Código de la función Borrar Grupos

El código mostrado en la figura 5.3 realiza la operación de borrado de grupos en la base de datos. Para realizar la prueba de caja blanca, dicho código fue ejecutado en las distintas vertientes que tiene. En primer lugar se eliminó un grupo que no tenía alumnos inscritos. El proceso se muestra en las figuras 5.4, 5.5 y 5.6.

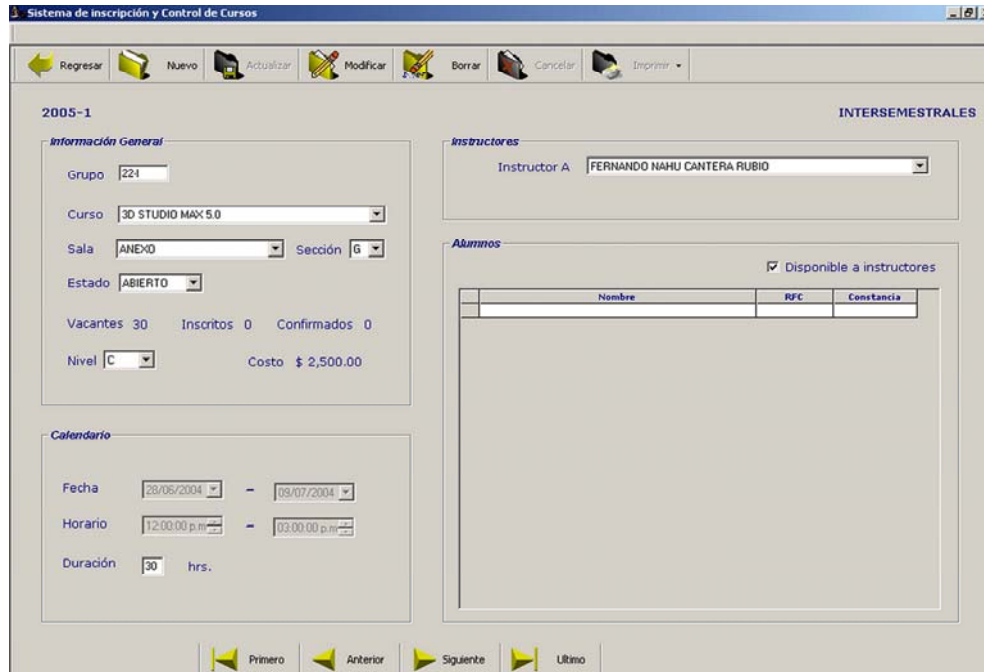


Figura 5.4. Operación de Borrado de grupos. En la imagen se observa el proceso de borrado de un grupo que no tiene alumnos inscritos. El primer paso es ubicarse sobre el registro indicado y a continuación oprimir el botón de Borrar ubicado en la barra de herramientas en la parte superior.

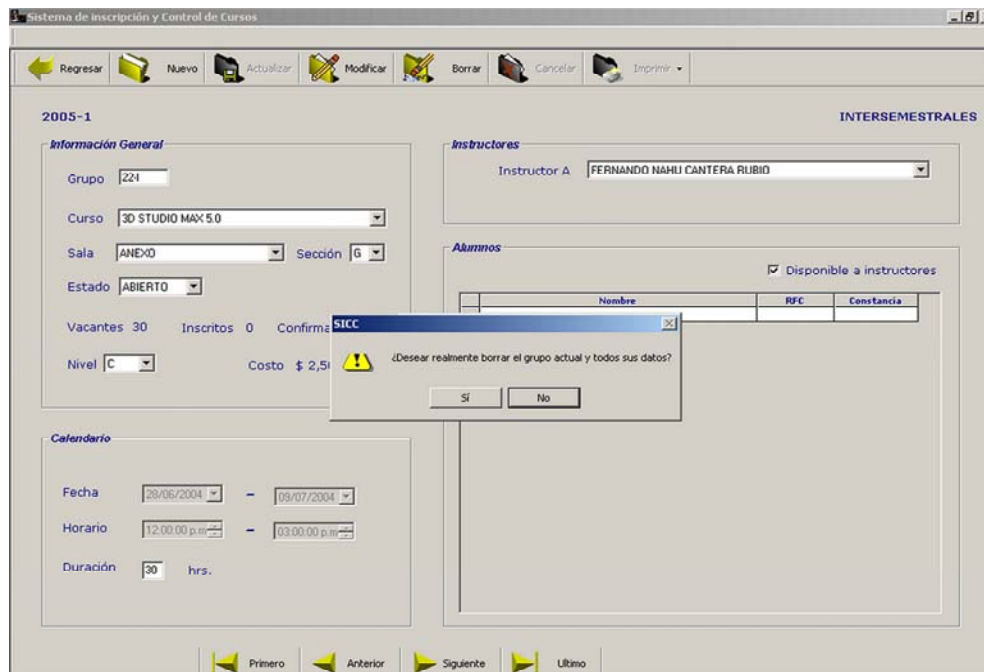


Figura 5.5. Mensaje de advertencia sobre el borrado de grupos. Una vez que se ubico el grupo a eliminar y que se oprimió el botón de Borrar, el sistema despliega un mensaje preguntando sobre la certeza acerca de la operación a realizar.



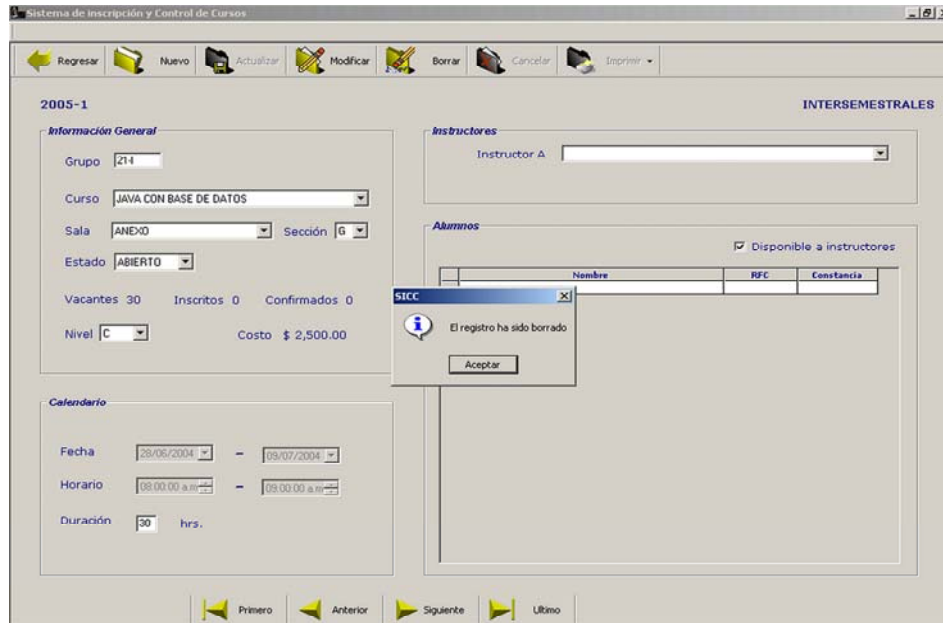


Figura 5.6. Grupo eliminado correctamente. Dado que el grupo que se eligió para ser eliminado no tenía alumnos inscritos, se pudo borrar sin mayor contratiempo del sistema.

A continuación se ejecutó la operación de eliminar un grupo con alumnos inscritos. Dicho proceso se observa en las figuras 5.7 y 5.8.

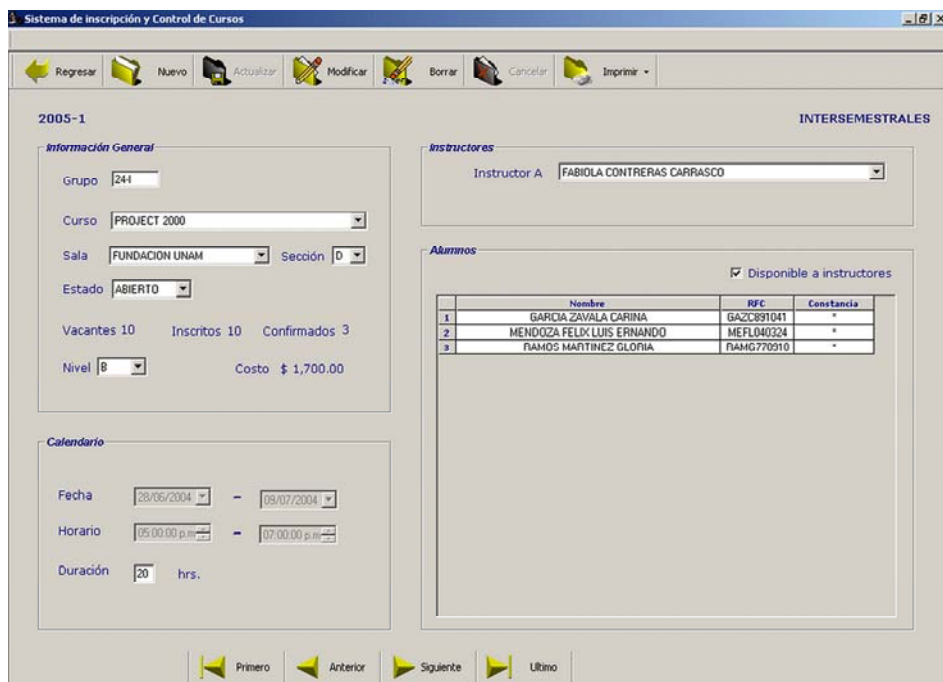


Figura 5.7. Borrado de grupos con alumnos inscritos. La imagen muestra la operación de borrado de un grupo que tiene alumnos inscritos.

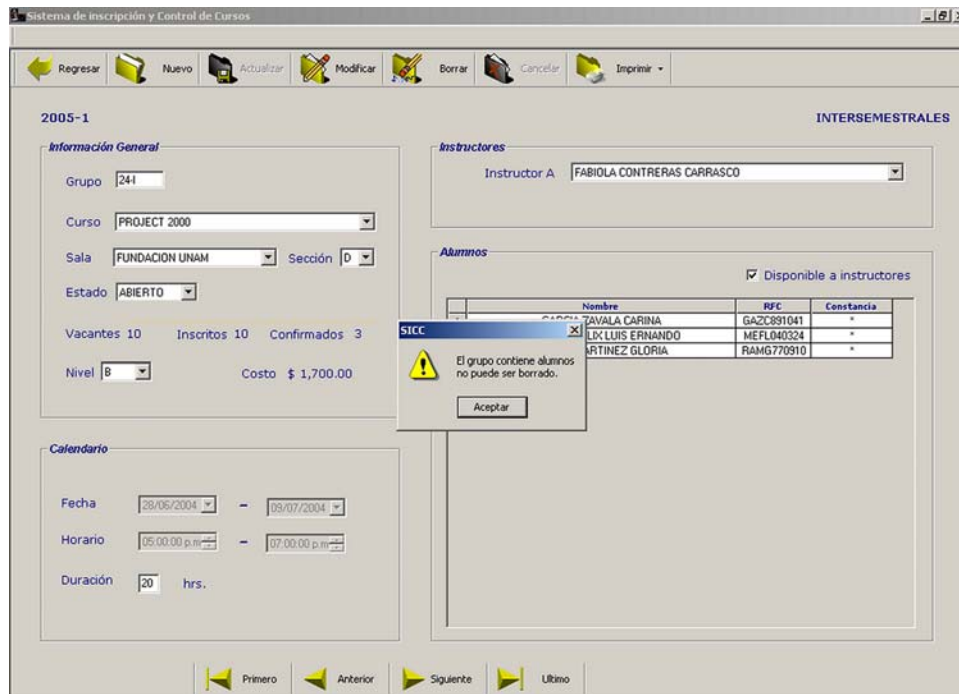


Figura 5.8. Mensaje de Error producido cuando se pretende borrar un grupo con alumnos inscritos.

Como se pudo observar, el sistema respondió correctamente puesto que en el primer caso, el grupo fue borrado sin ningún contratiempo dado que no tenía alumnos inscritos, mientras que el segundo caso, no se permitió borrar el grupo ya que tenía alumnos inscritos. Por lo tanto, podemos asegurar que dicha operación de Borrado de grupos esta cubierta completamente.

### 5.3 Aplicación y Resultados del Plan de Pruebas del SICC.

Para la primer parte del plan de pruebas referente a las aplicaciones de visual se crearon 10 grupos de prueba para el período 2005-1 y del tipo Intersemestrales. Dichos grupos fueron registrados con todos los datos requeridos (nombre del grupo, curso, sala, costo, etc). Así mismo se les asignó instructor.

La siguiente figura muestra los grupos creados:

Grupo	Curso	Estado	Vacantes	Fecha Inicial	Hora Inicial
1	01-I JAVA CON BASE DE DATOS	ABIERTO	30	28/06/2004	08:00 a.m.
2	02-I AUTOCAD BASICO	ABIERTO	30	28/06/2004	10:00 a.m.
3	03-I VISUAL BASIC 6.0	ABIERTO	30	28/06/2004	01:00 p.m.
4	04-I PROGRAMACION CON LENGUAJE C	ABIERTO	40	28/06/2004	09:00 a.m.
5	05-I LINUX BÁSICO	ABIERTO	37	28/06/2004	01:00 p.m.
6	06-I ADMINISTRACION LINUX	ABIERTO	37	28/06/2004	09:00 a.m.
7	07-I HTML CON JAVA SCRIPT	ABIERTO	37	28/06/2004	03:00 p.m.
8	08-I PHP CON BASE DE DATOS	ABIERTO	10	28/06/2004	11:00 a.m.
9	09-I FUNDAMENTOS DE JAVA	ABIERTO	37	28/06/2004	10:00 a.m.
10	24-I PROJECT 2000	ABIERTO	10	28/06/2004	05:00 p.m.

Figura 5.9. Creación de grupos para la fase de pruebas del SICC

Ahora bien, una vez creados se les asignaron alumnos. Para ello, se utilizó la interfaz de inscripción en su pantalla correspondiente. Para ilustrar lo anterior, elegimos el primer grupo llamado “01-I” con el curso “Java con Base de Datos” para asignar alumnos. El proceso se muestra en la figura 5.10:

Sistema de inscripción y Control de Cursos

Regresar Nuevo Guardar Modificar Borrar Cancelar

Semestre 2005-1

**Datos personales**

Nombre Lucia Ventura Gracia Franco RFC: GAFL831213 No. Cta. -  
 Tel. ... Sexo Femenino E-mail

**Grupos del periodo**

Tipo INTERSEMESTRALES Tipo de Alumno ESTUDIANTE DE LA UNAM

Grupo	Curso	Estado	Inscribir
1	061 ADMINISTRACION LINUX	ABIERTO	
2	021 AUTOCAD BASICO	ABIERTO	
3	091 FUNDAMENTOS DE JAVA	ABIERTO	
4	071 HTML CON JAVA SCRIPT	ABIERTO	
5	011 JAVA CON BASE DE DATOS	ABIERTO	Si
6	051 LINUX BASICO	ABIERTO	
7	081 PHP CON BASE DE DATOS	ABIERTO	
8	041 PROGRAMACION CON LENGUAJE C	ABIERTO	
9	241 PROJECT 2000	ABIERTO	
10	031 VISUAL BASIC 6.0	ABIERTO	

Costo \$1,250.00  
 Confirmado

Figura 5.10. Asignación de alumnos a un grupo. Para los grupos creados en el paso anterior, se asignaron alumnos para comprobar que dicho proceso es cubierto correctamente por el sistema. Para lograr lo anterior se utiliza la interfaz de Inscripción y se llenan los datos personales del alumno. Una vez hecho esto, se elige el curso deseado y al dar clic sobre el renglón correspondiente a dicho curso, automáticamente se asigna al alumno.

Sistema de inscripción y Control de Cursos

Regresar Imprimir

INTERSEMESTRALES / 2005-1

**Información General**

Grupo 011  
 Curso JAVA CON BASE DE DATOS  
 Sala ANEXO Sección G  
 Estado ABIERTO  
 Vacantes 30 Inscritos 3 Confirmados 3  
 Nivel C Costo \$ 2500

**Instructores**

Instructor A FERNANDO NAHU CANTERA RUBIO  
 Instructor B

**Alumnos**

	Nombre	RFC	Constancia
1	AGUILAR ARMENTA MARTHA GABRIELA	AUAM040416	*
2	FLORES FALCON MOISES ALEJANDRO	FOFM000417	*
3	GRACIA FRANCO LUCIA VENTURA	GAFL831213	*
4	MENDOZA FELIX LUIS ERNANDO	MEFL040324	*

Primero Anterior Siguiente Ultimo

Figura 5.11. Grupo con alumnos inscritos. El proceso de asignación de alumnos a un grupo visto en la figura anterior, se repitió 4 veces con alumnos diferentes para el grupo de prueba llamado “01-I” correspondiente al curso de Java con Base de datos. Dicho grupo con sus alumnos es mostrado en ésta figura.

Como se puede apreciar en ésta última figura, al grupo creado se le asignaron cuatro alumnos siguiendo el procedimiento descrito en la figura 5.10. Por lo tanto, las operaciones de crear grupo e inscripción de alumnos funcionan correctamente.

Lo siguiente es probar que las operaciones de modificación y de borrado funcionan correctamente tanto para los grupos como para los alumnos.

La primer prueba consistirá en modificar los datos a un grupo y observar como se comporta el sistema. Para ello, utilizaremos nuestro grupo de Java con Base de Datos creado anteriormente y el cuál aparece con todos sus datos en la figura 5.11. Se realizarán los siguientes cambios: reasignación de sala y sección de la sala, cambio de nivel y por lo tanto de costo y finalmente reducción de la duración en horas del curso.

The screenshot shows the 'Sistema de inscripción y Control de Cursos' interface. The 'Información General' section is updated with the following details:

- Grupo: 014
- Curso: JAVA CON BASE DE DATOS
- Sala: EDIFICIO PRINCIPAL FI
- Sección: I
- Estado: ABIERTO
- Vacantes: 30
- Inscritos: 3
- Confirmados: 3
- Nivel: D
- Costo: \$ 3,000.00

The 'Instructores' section shows Instructor A as FERNANDO NAHU CANTERA RUBIO. The 'Alumnos' section includes a table of students:

Nombre	RFC	Constancia
AGUILAR ARMENTA MARTHA GABRIELA	ALJAM0416	*
MOISES ALEJANDRO	FOFM00417	*
ED LUCIA VENTURA	GAFLE31213	*
LK LUIS ERNANDO	MEFL040324	*

A modal dialog box displays the message: 'Grupo modificado satisfactoriamente' with an 'Aceptar' button.

Figura 5.12. Modificado de un grupo. Se observa en la imagen que el grupo cuya sala era el “Anexo” (fig.5.11) ahora ha sido modificado con la sala del “Edificio Principal FI”. Lo mismo ocurrió con la sección, nivel, costo y la duración del curso.

Lo siguiente es modificar los datos de un alumno. Para ello utilizaremos el registro de uno de los alumnos del curso de Java.

Sistema de inscripción y Control de Cursos

Regresar Nuevo Guardar Modificar Borrar Cancelar

Semestre 2005-1

**Datos personales**

Nombre: LUIS ERNANDO MENDOZA FELIX RFC: MEFL040324 No. Cta.: -  
Tel.: - - - Sexo: Masculino E-mail: - - -

**Grupos Inscritos**

Periodo	Grupo	Curso	Fecha Inscripción	Estado	Confirmado	Tipo Alumno	
1	2005-1	01-1	JAVA CON BASE DE DATOS	24/03/2004	ABIERTO	No	ESTUDIANTE DE LA UNAM

Primero Anterior Siguiente Ultimo

Figura 5.13. Modificado de un alumno. Como se puede observar en la figura 5.13 el alumno no tiene email, teléfono y tampoco fue registrado con su número de cuenta. Por lo tanto, se modificará agregando esos datos.

Sistema de inscripción y Control de Cursos

Regresar Nuevo Guardar Modificar Borrar Cancelar

Semestre 2005-1

**Datos personales**

Nombre: LUIS ERNANDO MENDOZA FELIX RFC: MEFL040324 No. Cta.: 09514257-2  
Tel.: 45-78-78-98 Sexo: Masculino E-mail: lernando@redito.com

**Grupos Inscritos**

Periodo	Grupo	Curso	Fecha Inscripción	Estado	Confirmado	Tipo Alumno	
1	2005-1	01-1	JAVA CON BASE DE DATOS	24/03/2004	ABIERTO	No	ESTUDIANTE DE LA UNAM

SICC  
El alumno fue modificado correctamente  
Aceptar

Primero Anterior Siguiente Ultimo

Figura 5.14. Alumno modificado correctamente. Se aprecia que los datos de e-mail y número de cuenta aparecen en el alumno después de realizar la modificación.

Para probar la operación de borrado de alumnos, utilizaremos el ejemplo anterior de modificado, sólo que ahora la operación consistirá en eliminar al alumno del sistema. Esto se puede apreciar en la siguiente figura:

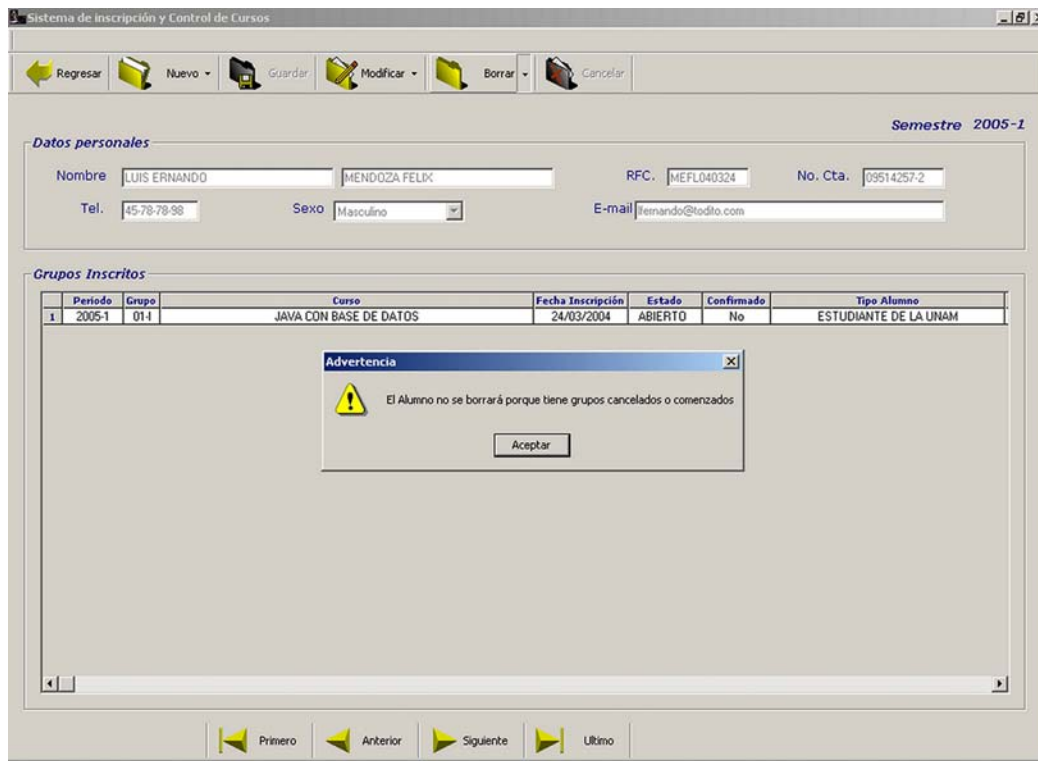


Figura 5.15. Borrado de un alumno. Se observa al alumno que anteriormente se modificó, pero ahora con la operación de borrar.

En este caso, como lo indica el mensaje, el alumno no puede ser borrado del sistema porque esta inscrito en un curso que ya comenzó. Este es el comportamiento esperado del sistema para este caso. Por lo tanto, para apreciar la operación de borrado de alumnos, asignaremos un alumno a un curso cuya fecha de inicio sea mucho mayor a la actual para indicar que dicho curso no ha comenzado aún por ejemplo el curso de Project 2000.

Sistema de inscripción y Control de Cursos

Regresar Nuevo Guardar Modificar Borrar Cancelar

Semestre 2005-1

**Datos personales**

Nombre: JOSE ARMANDO FLORES RFC: FIDA270979 No. Cta: 09501487-2  
 Tel: 55-87-65-78 Sexo: Masculino E-mail: jose@todio.com

**Grupos Inscritos**

Período	Grupo	Curso	Fecha Inscripción	Estado	Confirmado	Tipo Alumno
1	241	PROJECT 2000	07/06/2004	ABIERTO	No	BECARIO

Primero Anterior Siguiente Ultimo

Figura 5.16. Asignación de un alumno a un curso que no ha comenzado. Para observar el comportamiento del sistema en la operación de borrado de alumnos, se asignó el alumno que aparece en la figura al curso de Project 2000 cuya fecha de inicio es mayor a la fecha actual del sistema (para indicar que dicho curso no ha comenzado aún).

Sistema de inscripción y Control de Cursos

Regresar Nuevo Guardar Modificar Borrar Cancelar

Semestre 2005-1

**Datos personales**

Nombre: [ ] RFC: [ ] No. Cta: [ ]  
 Tel: [ ] Sexo: Masculino E-mail: [ ]

**Grupos Inscritos**

Período	Grupo	Curso	Fecha Inscripción	Estado	Confirmado	Tipo Alumno
---------	-------	-------	-------------------	--------	------------	-------------

SICC  
 El Alumno fue borrado del sistema  
 Aceptar

Primero Anterior Siguiente Ultimo

Figura 5.17. Alumno eliminado. Se observa que el alumno fue borrado del sistema puesto que el curso asignado a dicho alumno no ha comenzado, por lo tanto se pueden eliminar alumnos sin ningún inconveniente.



Como se puede observar el proceso de borrado responde correctamente a las especificaciones del proyecto.

La siguiente prueba consiste en la creación, modificado y borrado de un tipo de grupo. En específico se creará un nuevo tipo de grupo llamado “Extemporáneo” cuya clave será la letra X. Una vez creado se modificará y por último se eliminará para observar la respuesta del sistema.

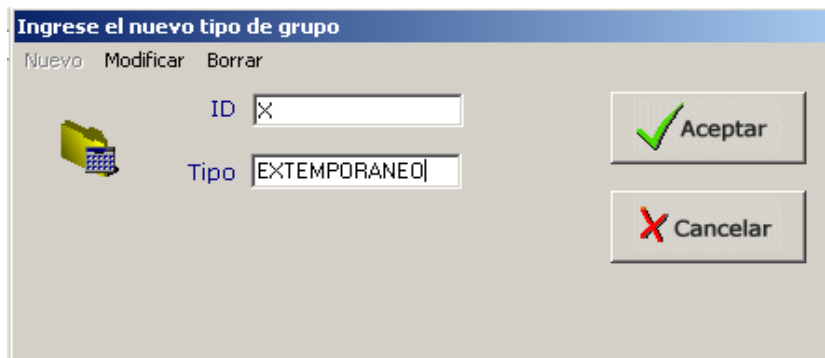


Figura 5.18. Creación de un nuevo tipo de grupo.

Como se aprecia en la figura 5.18, Para crear un nuevo tipo de grupo se oprime el botón de Nuevo y a continuación se colocan los datos necesarios. Si no existe ya un tipo de grupo con esos datos al oprimir el botón de Aceptar se creará sin ningún inconveniente.

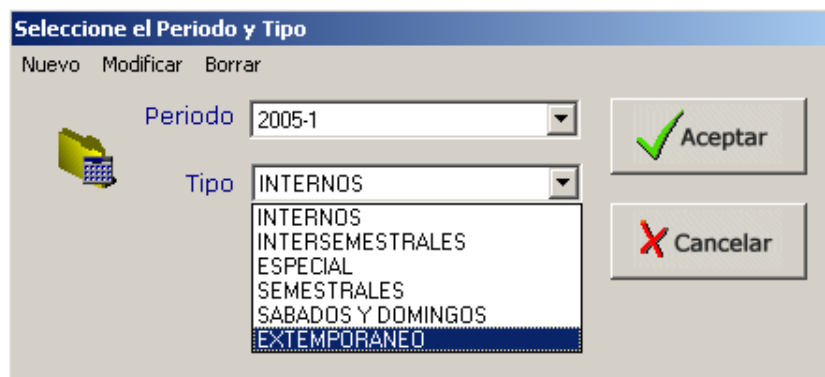


Figura 5.19. Nuevo tipo de grupo agregado a la colección.

En esta figura se observa que el tipo de grupo fue creado correctamente y ahora se pueden crear grupos con ese tipo de grupo.

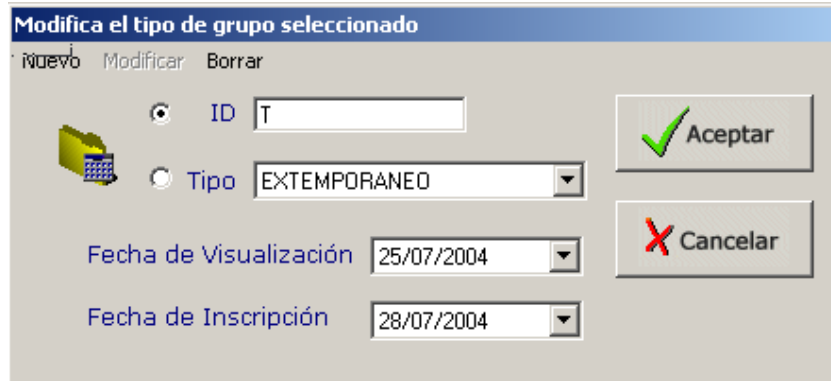


Figura 5.20. Modificado de un tipo de grupo.

Al oprimir el botón de modificar se pueden actualizar los datos del tipo de grupo ya sea por el tipo, su clave (ID), su fecha de visualización (fecha a partir de la cual se publicarán los cursos que sean de ese tipo) o por su fecha de inscripción (fecha a partir de la cual se podrán realizar inscripciones a los grupos de ese tipo). Para esta prueba se cambió el ID del tipo que era “X” por “T”.

Por último se realizó el borrado del tipo de grupo. Este proceso se aprecia en las figuras X y W.

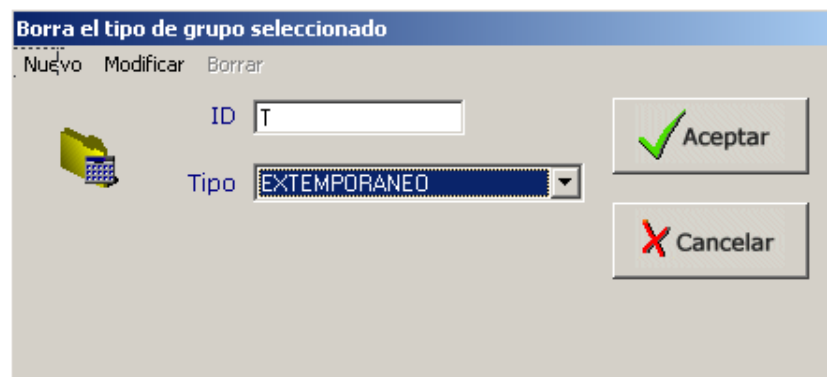


Figura 5.21. Borrado de un tipo de grupo. Para realizar la eliminación de un tipo de grupo, primero se debe posicionar sobre el tipo de grupo a eliminar y posteriormente oprimir el botón de aceptar.



Figura 5.22. Tipo de grupo eliminado satisfactoriamente.

Las pruebas de los reportes, tal como se mencionó en el plan de pruebas, consisten en la generación e impresión de cada uno de ellos para observar que cumplen con los requisitos establecidos. Sin embargo, esta parte no se detalla aquí puesto que fué atendida en el cap.4 Subtema 4.4 **Generación de Reportes y Constancias** en donde fueron generados y explicados cada uno de ellos por lo tanto, la prueba estuvo implícita en dicha descripción.

### 5.3.1 Pruebas de Búsqueda

Para probar la búsqueda de grupos se realizarán las siguientes búsquedas: por nombre del grupo, nombre del curso y por fecha inicial.

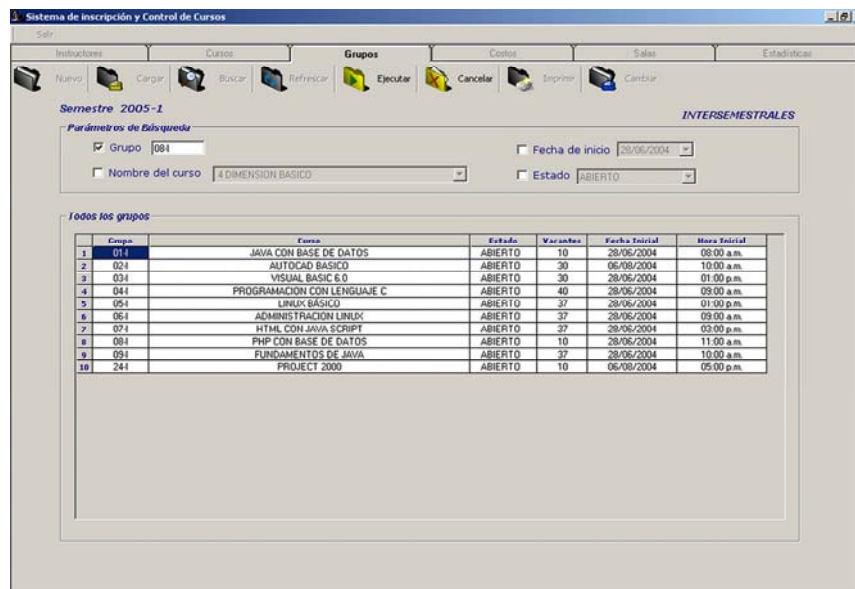


Figura 5.23. Búsqueda por nombre de grupo. En la imagen se aprecia una búsqueda de grupo mandando el nombre como parámetro, en este caso “08-I”. Como no se permiten nombre de grupos repetidos para un mismo período, el resultado debe arrojar sólo un grupo para dicha búsqueda.

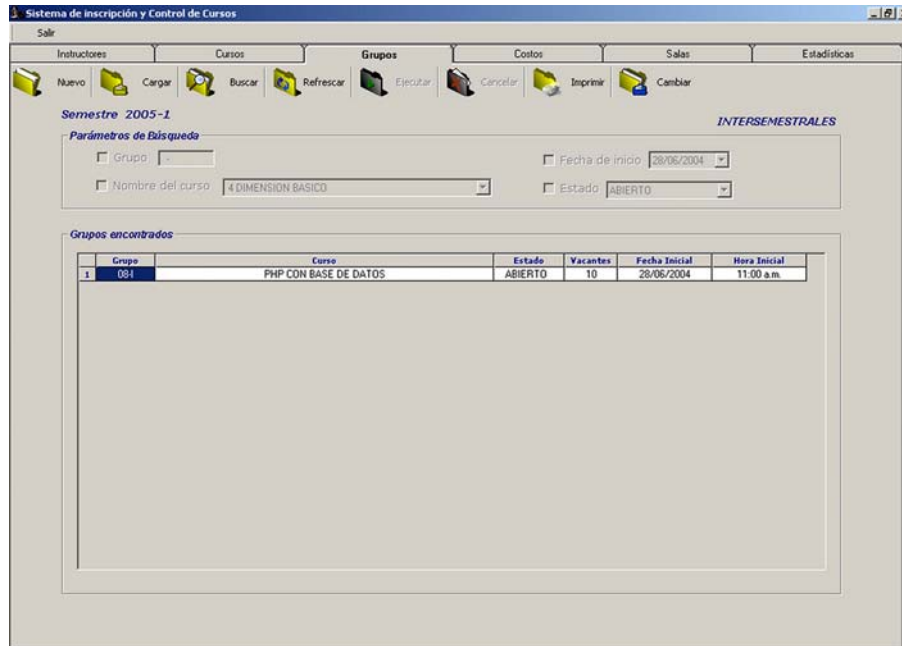


Figura 5.24. Resultado de una búsqueda por nombre de grupo. Se aprecia en la imagen el resultado de buscar un grupo por su nombre, en este caso “08-I”. Como se puede observar el sistema devuelve todos los datos referentes a ese grupo solamente.

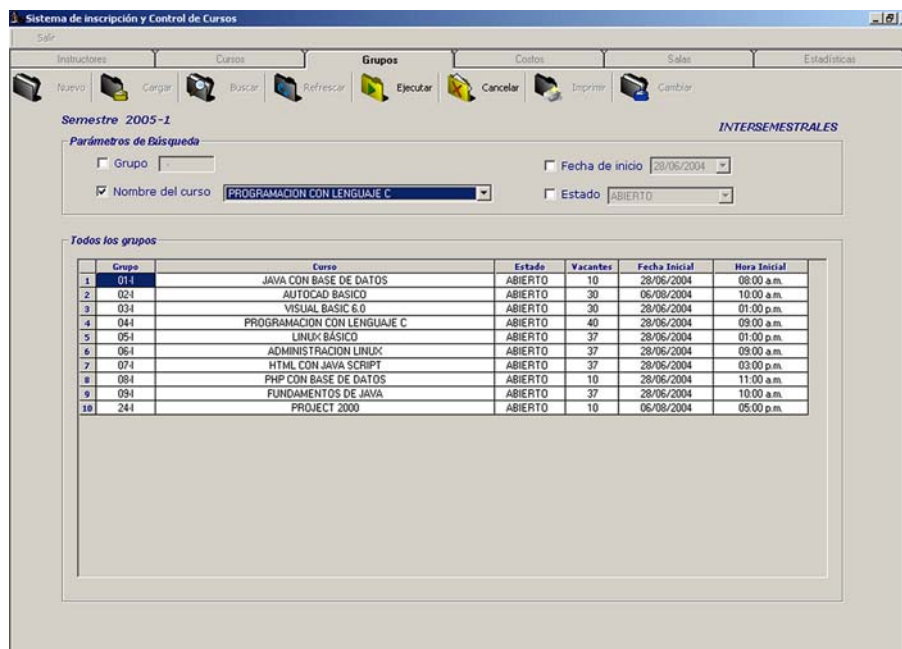


Figura 5.25. Búsqueda por nombre de curso. En este caso, la búsqueda se realiza a través del nombre de algún curso (para nuestro ejemplo “Programación con lenguaje C”). Dado que puede existir más de un grupo con el mismo curso, el resultado puede ser de uno o más registros.

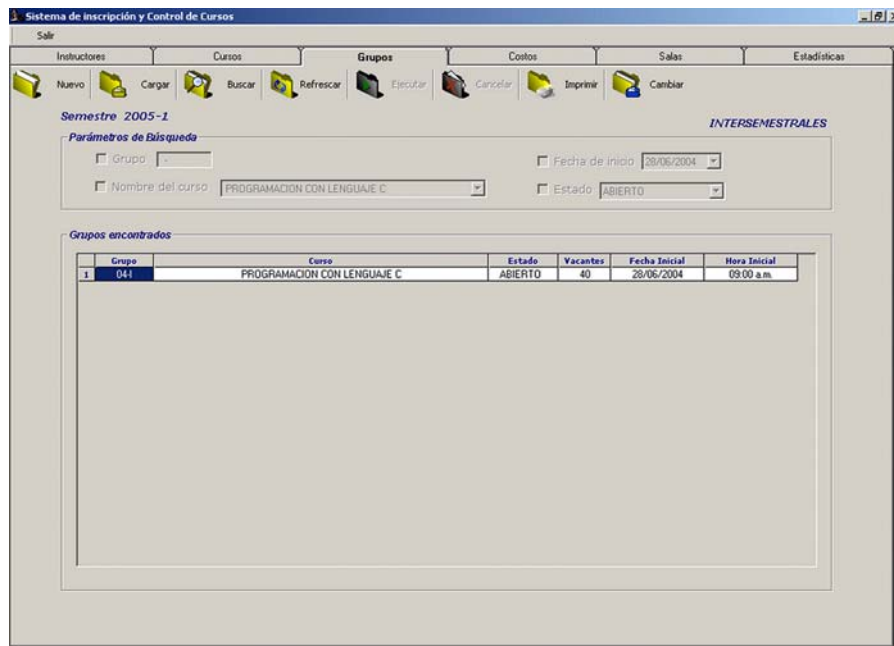


Figura 5.26. Resultado de una búsqueda por nombre de curso. Como se observa en la imagen, sólo existe un grupo con el curso de “Programación con lenguaje C”, por lo tanto dicho registro es visualizado.

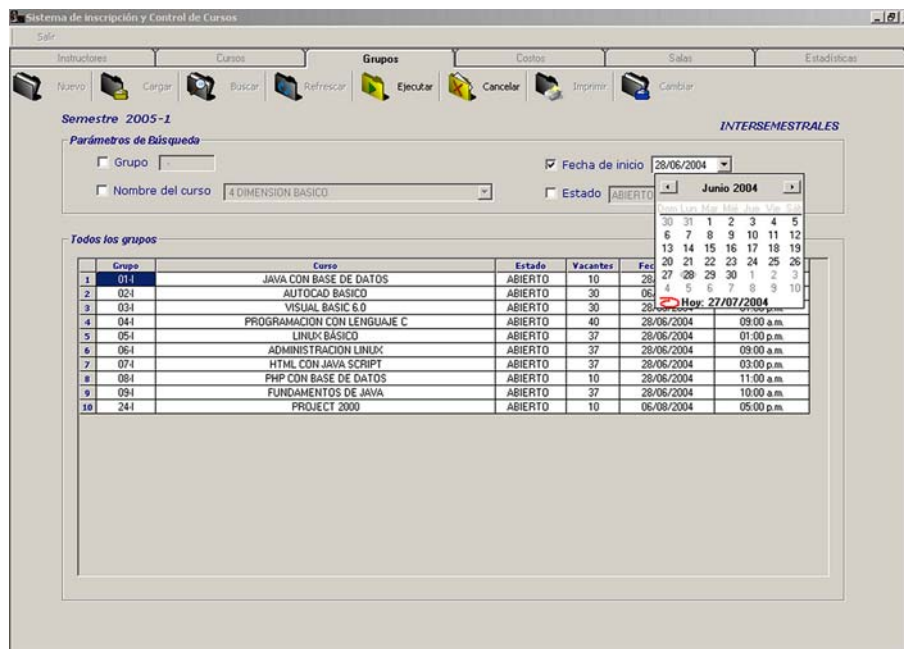


Figura 5.27. Búsqueda por fecha inicial de un curso. Para realizar ésta prueba, se decidió buscar aquellos grupos que iniciarán el 28 de Julio del 2004. Para nuestro caso, existe más de un grupo que cumple con dicho patrón.

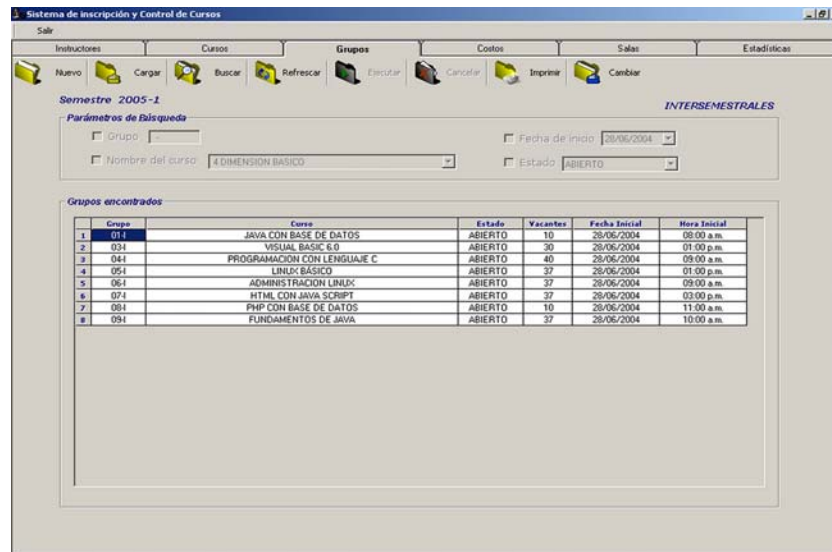


Figura 5.28. Resultado de una búsqueda por fecha inicial de curso. La imagen muestra todos los grupos encontrados con la fecha de inicio correspondiente al 28 de Julio del 2004. En esta ocasión se trató de más de un registro.

Ahora para probar que el sistema se comporta de forma adecuada, se realizará una búsqueda por nombre del grupo y por fecha inicial pero con datos erróneos, es decir, la fecha de inicio del grupo no corresponderá con dicho grupo. Las figuras 5.29 y 5.30 ilustran lo anterior:

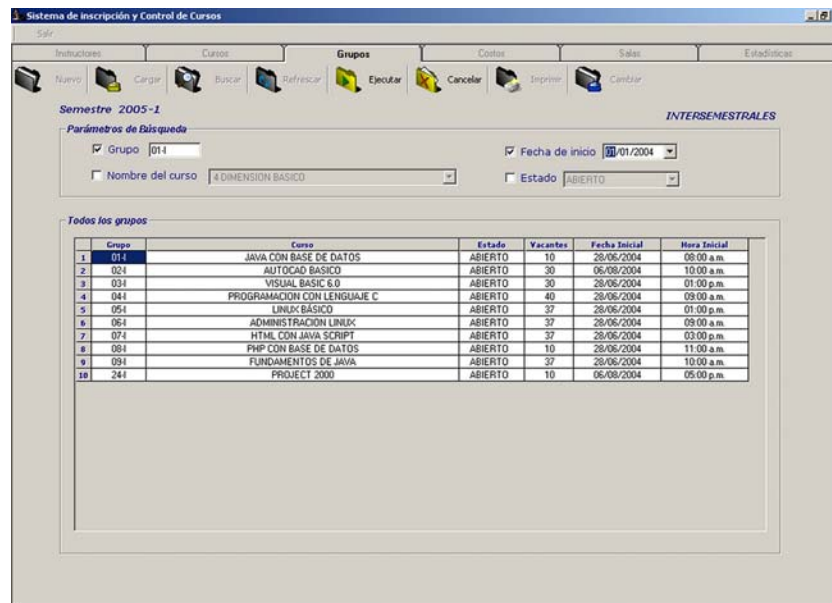


Figura 5.29. Búsqueda con datos erróneos. Para realizar esta prueba se mandaron datos equivocados para el nombre del grupo y la fecha de inicio. El grupo "01-I" inicia el 28 de Julio del 2004 y por lo tanto al combinar la búsqueda de dicho grupo con la fecha de inicio correspondiente al 1º. de enero del 2004, el resultado no debe mostrar ningún grupo con esas características.

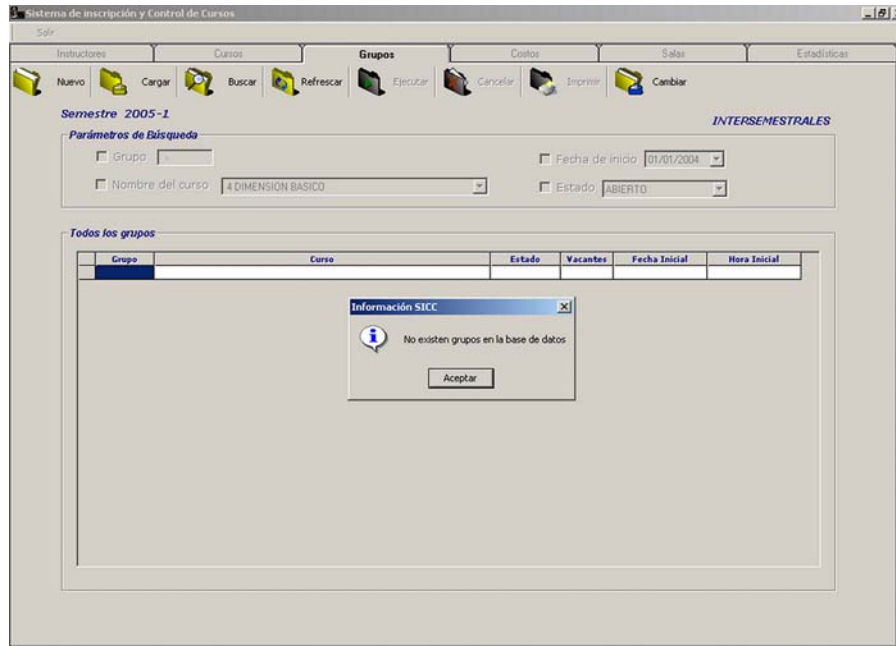


Figura 5.30. Resultado de una búsqueda con datos erróneos. Dado que los valores para el nombre del grupo y la fecha de inicio no son correctos, el sistema no muestra ningún grupo como resultado de la búsqueda.

### 5.3.2 Pruebas para la Aplicación web

Para comprobar que el sistema es capaz de desplegar los cursos vía Internet, se accederá a la página desarrollada con ese propósito y se realizará el apartado de cursos.

En una primer prueba se apartarán dos cursos y se verificará la generación del comprobante de inscripción. Después se tratará de apartar un tercer curso para comprobar que el sistema no permite dicha operación.

El sistema debe realizar diferentes operaciones de acuerdo a la fecha en la que se encuentre (ver Capítulo 3 subtema 3.5 **Diseño e implementación del SICC mediante OMT y UML figura 3.13**). Para verificar esto, se realizarán los cambios de fecha debidos en la base de datos. Específicamente se realizarán las siguientes pruebas:


Tomando como punto de referencia la fecha actual, se designará una fecha de inscripción a los cursos menor y la fecha actual será mayor a la fecha de inicio de cursos menos tres días. Como se observó en el cap.3 estas condiciones indican que las inscripciones sólo se pueden realizar directamente en la administración de UNICA Edif. Principal (ver Capítulo 3 subtema 3.5 **Diseño e implementación del SICC mediante OMT y UML tabla 3.3**).

Posteriormente se cambiará la fecha de inscripción a los cursos por una menor a la fecha actual y una fecha de inicio de cursos también menor a la actual y se observará la respuesta del sistema.

Finalmente se capturará una fecha final de cursos menor a la fecha actual y se observarán los resultados.



SISTEMA DE INSCRIPCIONES Y CONTROL DE CURSOS (SICC) - Microsoft Internet Explorer


**Facultad de Ingeniería**  
 Unidad de Servicios de Cómputo Académico 

CURSOS
INTERSEMESTRALES  
2005-1  
Extracurriculares

ESTADO	GRUPO	CURSO	FECHA	HORARIO	DURACION (Horas)	SALA	ANTECEDENTES	\$	Cupo
<input checked="" type="checkbox"/>	06-1	ADMINISTRACION LINUX	2 de Agosto al 16 de Agosto	09:00 12:00	30	H	VII	C	37
<input checked="" type="checkbox"/>	07-1	HTML CON JAVA SCRIPT	2 de Agosto al 16 de Agosto	15:00 18:00	30	F	III ,IV ,V	C	37
<input type="checkbox"/>	08-1	PHP CON BASE DE DATOS	2 de Agosto al 16 de Agosto	11:00 14:00	30	D	III ,VIII	C	10
<input type="checkbox"/>	09-1	FUNDAMENTOS DE JAVA	2 de Agosto al 16 de Agosto	10:00 13:00	30	F	V	C	37

**Inscripciones**

En la Unidad de Servicios de Cómputo Académico, Edificio Principal, Facultad de Ingeniería, UNAM 20 de Mayo del 2004

**Horario de inscripción**

09:00 a 20:00 hrs. de Lunes a Viernes

A	B	C	D	E
1300	1700	2500	3000	500

Descuento del 50% a estudiantes y trabajadores de la UNAM, otras Instituciones Educativas 25%

Recibe un 10 % de descuento adicional al inscribirte la primer semana.

**\*ANTECEDENTES**

- I NINGUNO
- II CONOCIMIENTOS BASICOS DE COMPUTACION
- III MANEJO DE AMBIENTE WINDOWS
- IV CONOCIMIENTOS BASICOS DE INTERNET
- IX FUNDAMENTOS DE JAVA
- V CONOCIMIENTOS BASICOS DE PROGRAMACION
- VI MANEJO DE VISUAL BASIC
- VII MANEJO DE LINUX
- VIII MANEJO DE BASES DE DATOS
- X ADMINISTRACION UNIX
- XI MANEJO DE SQL
- XII PROGRAMACION ORIENTADA A OBJETOS
- XIII CONOCIMIENTOS DE HTML
- XIV MANEJO DE AUTOCAD

regresar
registrarse

Figura 5.31. Prueba de apartado de cursos. En la figura se observa el apartado de dos cursos, “Administración Linux” y “HTML con Java Script”. Dado que se pueden elegir hasta dos cursos, el sistema debe responder generando los comprobantes correspondientes.

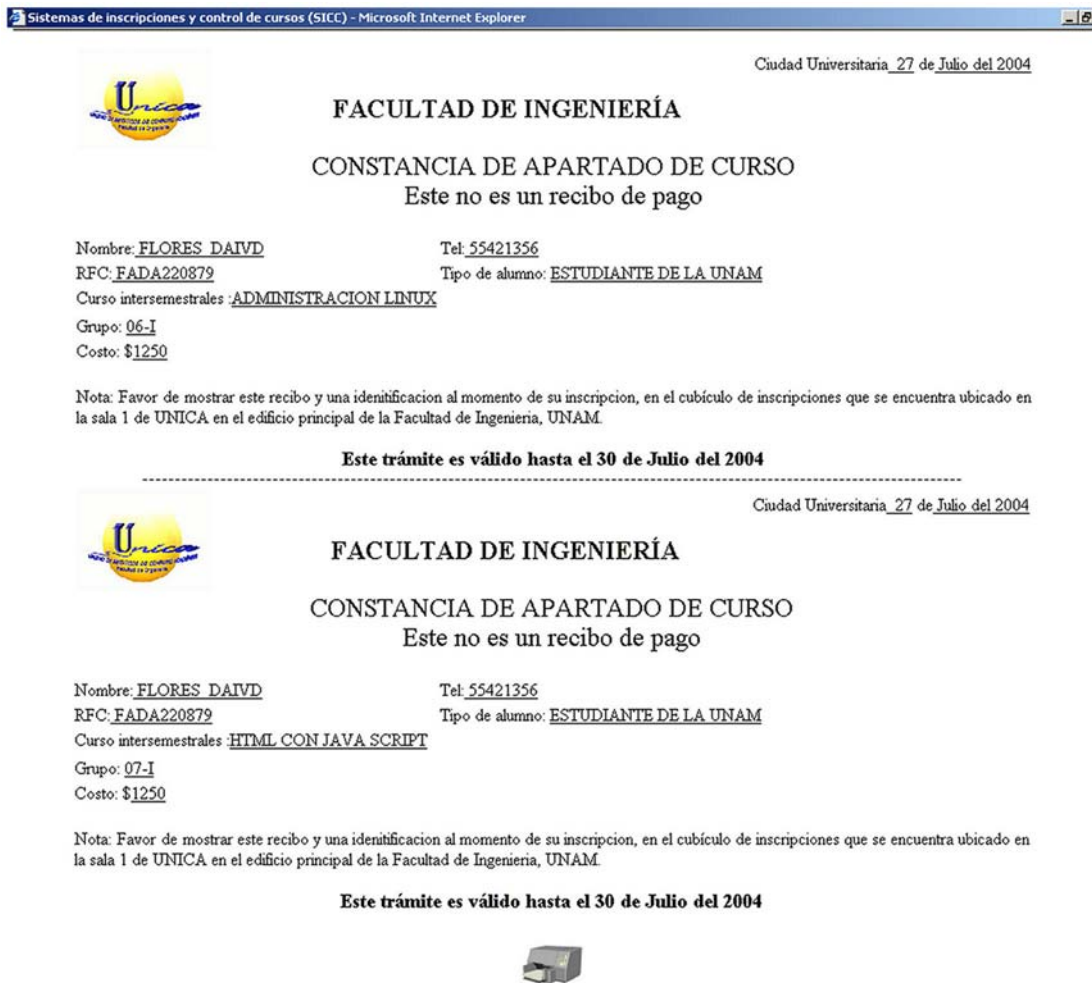


Figura 5.32. Constancias de apartado. Dado que se apartaron dos cursos, el sistema genera los comprobantes correspondientes: el primero de ellos para el curso de “Administración Linux” y el otro para “HTML con Java Script”.

Como se pudo apreciar en las figuras 5.31 y 5.32 el sistema respondió correctamente cuando se apartan dos cursos vía web. Para conocer más acerca del procedimiento de apartado consultar el Capítulo 4 **Interfaz de apartado** .

Ahora veamos que sucede cuando se pretenden apartar más de dos cursos en la aplicación.

The screenshot shows the 'SISTEMA DE INSCRIPCIONES Y CONTROL DE CURSOS (SICC)' web application. The header includes the 'Facultad de Ingeniería' logo and 'Unica' logo. The main title is 'CURSOS INTERSEMESTRALES Extracurriculares 2005-1'. Below this is a table with columns: ESTADO, GRUPO, CURSO, FECHA, HORARIO, DURACION (horas), SALA, ANTECEDENTES, \$, and Cupo. The table lists four courses: ADMINISTRACION LINUX (06-1), HTML CON JAVA SCRIPT (07-1), PHP CON BASE DE DATOS (08-1), and FUNDAMENTOS DE JAVA (09-1). A modal error message is displayed over the table, stating 'No se puede seleccionar mas de dos' (No more than two can be selected).

ESTADO	GRUPO	CURSO	FECHA	HORARIO	DURACION (horas)	SALA	ANTECEDENTES	\$	Cupo
<input type="checkbox"/>	06-1	ADMINISTRACION LINUX				H	VII	C	36
<input checked="" type="checkbox"/>	07-1	HTML CON JAVA SCRIPT	Agosto al 16 de Agosto	15:00 18:00	30	F	III ,IV ,V	C	36
<input checked="" type="checkbox"/>	08-1	PHP CON BASE DE DATOS	2 de Agosto al 16 de Agosto	11:00 14:00	30	D	III ,VIII	C	10
<input checked="" type="checkbox"/>	09-1	FUNDAMENTOS DE JAVA	2 de Agosto al 16 de Agosto	10:00 13:00	30	F	V	C	37

Microsoft Internet Explorer  
No se puede seleccionar mas de dos  
Aceptar

Inscripciones \*ANTECEDENTES

Figura 5.33. Mensaje de error de la aplicación web. Como se aprecia en la imagen, el usuario pretende apartar tres cursos a lo cuál el sistema responde enviando un mensaje de error, puesto que la operación de apartado sólo permite dos cursos según la especificación de requerimientos.



La siguiente prueba consiste en variar las fechas de la base de datos para verificar que se cumplan aquellas condiciones establecidas en el diseño del sistema (consultar Capítulo 3 subtema 3.5 **Diseño e implementación del SICC mediante OMT y UML tabla 3.3**).

En primer instancia, simularemos las condiciones en que la fecha actual es mayor a la fecha de inscripción y mayor a la fecha de inicio de un curso menos tres días, por lo tanto las inscripciones sólo se pueden realizar en el Edif. Principal (Administración de UNICA).

Considerando que la fecha actual es el 27 julio del 2004 entonces una fecha de inscripción a cursos apropiada para ésta primer prueba es el 20 de junio del 2004 y una fecha de inicio para los cursos apropiada es el 28 de julio del 2004. La respuesta del sistema se aprecia en la figura 5.34.



SISTEMA DE INSCRIPCIONES Y CONTROL DE CURSOS (SICC) - Microsoft Internet Explorer


**Facultad de Ingeniería**  
 Unidad de Servicios de Cómputo Académico 

CURSOS
INTERSEMESTRALES  
2005-1

Extracurriculares

ESTADO	GRUPO	CURSO	FECHA	HORARIO	DURACION (Horas)	SALA	ANTECEDENTES	\$	Cupo
Inscripciones	06-1	ADMINISTRACION LINUX	28 de Julio al 16 de Agosto	09:00 12:00	30	H	VII		C 36
Inscripciones	07-1	HTML CON JAVA SCRIPT	28 de Julio al 16 de Agosto	15:00 18:00	30	F	III ,IV ,V		C 36
Inscripciones	08-1	PHP CON BASE DE DATOS	28 de Julio al 16 de Agosto	11:00 14:00	30	D	III ,VIII		C 10
Inscripciones	09-1	FUNDAMENTOS DE JAVA	28 de Julio al 16 de Agosto	10:00 13:00	30	F	V		C 37

**Inscripciones**

En la Unidad de Servicios de Cómputo Académico, Edificio Principal, Facultad de Ingeniería, UNAM 20 de Junio del 2004

**Horario de Inscripción**

09:00 a 20:00 hrs. de Lunes a Viernes

A	B	C	D	E
1300	1700	2500	3000	500

Descuento del 50% a estudiantes y trabajadores de la UNAM, otras Instituciones Educativas 25%

Recibe un 10 % de dedscuento adicional al inscribirte la primer semana.

**\*ANTECEDENTES**

- I NINGUNO
- II CONOCIMIENTOS BASICOS DE COMPUTACION
- III MANEJO DE AMBIENTE WINDOWS
- IV CONOCIMIENTOS BASICOS DE INTERNET
- IX FUNDAMENTOS DE JAVA
- V CONOCIMIENTOS BASICOS DE PROGRAMACION
- VI MANEJO DE VISUAL BASIC
- VII MANEJO DE LINUX
- VIII MANEJO DE BASES DE DATOS
- X ADMINISTRACION UNIX
- XI MANEJO DE SQL
- XII PROGRAMACION ORIENTADA A OBJETOS
- XIII CONICIMIENTOS DE HTML
- XIV MANEJO DE AUTOCAD

Gracias por su visita

Figura 5.34. Se observa que el sistema ya no permite hacer el apartado y esto es porque según los requerimientos, dicho proceso sólo se permite hasta tres días antes de que comience el curso, una vez sobrepasado este límite, las inscripciones sólo se pueden hacer en las oficinas de UNICA en el Edif. Principal.

Para la siguiente prueba, se simularán las condiciones correspondientes a una fecha actual mayor a la fecha de inscripción y mayor también a la fecha de inicio del curso. Evidentemente, el sistema debe mostrar algún mensaje en el que se indique que los cursos ya comenzaron.

Considerando que la fecha actual es el 27 julio del 2004 entonces una fecha de inscripción a cursos apropiada para la prueba es el 20 de junio del 2004 y una fecha de inicio para los cursos apropiada es el 26 de julio del 2004. La siguiente figura muestra los resultados de la prueba.

SISTEMA DE INSCRIPCIONES Y CONTROL DE CURSOS (SICC) - Microsoft Internet Explorer

**Facultad de Ingeniería**  
 Unidad de Servicios de Cómputo Académico

**CURSOS INTERSEMESTRALES 2005-1**  
 Extracurriculares

ESTADO	GRUPO	CURSO	FECHA	HORARIO	DURACION (Horas)	SALA	ANTECEDENTES	\$	Cupo
Ya comenzó	06-1	ADMINISTRACION LINUX	26 de Julio al 16 de Agosto	09:00-12:00	30	H	VII	C	36
Ya comenzó	07-1	HTML CON JAVA SCRIPT	26 de Julio al 16 de Agosto	15:00-18:00	30	F	III ,IV ,V	C	36
Ya comenzó	08-1	PHP CON BASE DE DATOS	26 de Julio al 16 de Agosto	11:00-14:00	30	D	III ,VIII	C	10
Ya comenzó	09-1	FUNDAMENTOS DE JAVA	26 de Julio al 16 de Agosto	10:00-13:00	30	F	V	C	37

**Inscripciones**  
 En la Unidad de Servicios de Cómputo Académico, Edificio Principal, Facultad de Ingeniería, UNAM 20 de Junio del 2004

**Horario de Inscripción**  
 09:00 a 20:00 hrs. de Lunes a Viernes

A	B	C	D	E
1300	1700	2500	3000	500

Descuento del 50% a estudiantes y trabajadores de la UNAM, otras Instituciones Educativas 25%

Recibe un 10 % de deducción adicional al inscribirte la primer semana.

**\*ANTECEDENTES**  
 I NINGUNO  
 II CONOCIMIENTOS BASICOS DE COMPUTACION  
 III MANEJO DE AMBIENTE WINDOWS  
 IV CONOCIMIENTOS BASICOS DE INTERNET  
 IX FUNDAMENTOS DE JAVA  
 V CONOCIMIENTOS BASICOS DE PROGRAMACION  
 VI MANEJO DE VISUAL BASIC  
 VII MANEJO DE LINUX  
 VIII MANEJO DE BASES DE DATOS  
 X ADMINISTRACION UNIX  
 XI MANEJO DE SQL  
 XII PROGRAMACION ORIENTADA A OBJETOS  
 XIII CONOCIMIENTOS DE HTML  
 XIV MANEJO DE AUTOCAD

Gracias por su visita

[regresar](#)

Figura 5.35. Ahora se observa que el sistema envía el mensaje avisando que los cursos ya comenzaron.

Finalmente, simularemos el escenario en el que la fecha de terminación del curso es menor a la fecha actual lo cuál indica que el curso ya terminó, por lo tanto se debe mostrar el calendario de cursos pero sin aquellos que ya hayan finalizado.

Para realizar esta prueba, tomaremos un curso de los creados y le cambiaremos la fecha de inicio y de terminación. Ambas tendrán que ser menores a la fecha actual. Por lo tanto, los datos para realizar la prueba quedan de la siguiente manera:

Curso afectado: “Fundamentos de Java”

Fecha inicial anterior: 26 de julio del 2004

Fecha inicial nueva: 5 de julio del 2004

Fecha final anterior: 16 de agosto del 2004

Fecha final nueva: 19 de julio del 2004

SISTEMA DE INSCRIPCIONES Y CONTROL DE CURSOS (SICC) - Microsoft Internet Explorer


**Facultad de Ingeniería**  
 Unidad de Servicios de Cómputo Académico



## CURSOS INTERSEMESTRALES

Extracurriculares 2005-1

ESTADO	GRUPO	CURSO	FECHA	HORARIO	DURACION (Horas)	SALA	ANTECEDENTES	\$	Cupo
Ya comenzó	06-1	ADMINISTRACION LINUX	26 de Julio al 16 de Agosto	09:00 12:00	30	H	VII	C	36
Ya comenzó	07-1	HTML CON JAVA SCRIPT	26 de Julio al 16 de Agosto	15:00 18:00	30	F	III ,IV ,V	C	36
Ya comenzó	08-1	PHP CON BASE DE DATOS	26 de Julio al 16 de Agosto	11:00 14:00	30	D	III ,VIII	C	10

**Inscripciones**

En la Unidad de Servicios de Cómputo Académico, Edificio Principal, Facultad de Ingeniería, UNAM 20 de Junio del 2004

**Horario de Inscripción**

09:00 a 20:00 hrs. de Lunes a Viernes

A	B	C	D	E
1300	1700	2500	3000	500

Descuento del 50% a estudiantes y trabajadores de la UNAM, otras Instituciones Educativas 25%

Recibe un 10 % de descuento adicional al inscribirte la primer semana.

**\*ANTECEDENTES**

- I NINGUNO
- II CONOCIMIENTOS BASICOS DE COMPUTACION
- III MANEJO DE AMBIENTE WINDOWS
- IV CONOCIMIENTOS BASICOS DE INTERNET
- IX FUNDAMENTOS DE JAVA
- V CONOCIMIENTOS BASICOS DE PROGRAMACION
- VI MANEJO DE VISUAL BASIC
- VII MANEJO DE LINUX
- VIII MANEJO DE BASES DE DATOS
- X ADMINISTRACION UNIX
- XI MANEJO DE SQL
- XII PROGRAMACION ORIENTADA A OBJETOS
- XIII CONOCIMIENTOS DE HTML
- XIV MANEJO DE AUTOCAD

Gracias por su visita



Figura 5.36. Como se puede observar, se muestra el calendario pero sin el curso que ya terminó

## 5.4 Mantenimiento del sistema

El mantenimiento del software es aquella actividad relacionada con la modificación de un producto de software, después de su entrega al cliente o usuario para corregir defectos, para mejorar el rendimiento u otras propiedades deseables, o para adaptarlo a un cambio de ambiente.

### 5.4.1 Causas del mantenimiento

Algunas de las principales razones por las cuáles es indispensable el mantenimiento de software son las siguientes:

- ❖ **Cambio en los requerimientos.** Los requerimientos del sistema a menudo cambian cuando el sistema esta siendo desarrollado debido a que el ambiente cambia. Por lo tanto, el sistema entregado no cumplirá con sus requerimientos.
- ❖ **Cambios pedidos por los clientes.** El mantenimiento en algunas ocasiones, se debe a cambios pedidos por los clientes o por los requerimientos del mercado. Los cambios normalmente se atienden en orden de pedido y se implementan en una nueva versión del sistema.
- ❖ **Ambiente externo.** Si un programa es dependiente de su ambiente externo, tendrá que soportar modificaciones que reflejen los cambios en el medio ambiente.

Los esfuerzos dedicados a mejorar la mantenibilidad pueden ser benéficos en relaciones costo/beneficio a largo plazo.

#### 5.4.2 Dificultades y costos del mantenimiento

- ❖ Restricciones de tamaño y espacio de almacenamiento. Por lo general, mantener un sistema implica realizar nuevas versiones de dicha aplicación. Por lo tanto se requiere de espacio suficiente para almacenar las nuevas versiones del producto.
- ❖
- ❖ Insatisfacción del cliente cuando no se puede atender en un tiempo aceptable una petición de reparación que parece razonable.
- ❖
- ❖ Los errores ocultos que se introducen al cambiar el software. durante el mantenimiento reducen la calidad global del producto. Como el mantenimiento algunas veces implica generación de nuevo código, nuevos errores se pueden presentar lo que conlleva que la calidad del producto se vea afectada.
- ❖
- ❖ Perjuicio en otros proyectos de desarrollo cuando la plantilla tiene que dejarlos, total o parcialmente, para atender peticiones de mantenimiento. Es muy frecuente que una vez liberado un proyecto, los desarrolladores tomen nuevas aplicaciones. Sin embargo, cuando ocurre el mantenimiento de algún sistema, los desarrolladores preferentemente, deben regresar para trabajar en dicha actividad, lo que implica el retraso de los nuevos proyectos de la organización.

#### 5.4.3 Tipos de mantenimiento

- ❖ **Mantenimiento perfectivo:** conjunto de actividades para mejorar o añadir nuevas funcionalidades requeridas por el usuario.
- ❖
- ❖ **Mantenimiento adaptativo:** es el conjunto de actividades para adaptar el sistema a los cambios (hardware o software) en su entorno tecnológico
- ❖



- ❖ **Mantenimiento correctivo:** es el conjunto de actividades dedicadas a corregir defectos en el hardware o en el software detectados por los usuarios durante el uso del sistema.
- ❖
- ❖ **Mantenimiento preventivo:** conjunto de actividades para facilitar el mantenimiento futuro del sistema.

## 5.5 Mantenimiento del SICC

Desde el momento de su liberación, el SICC ha recibido distintos tipos de mantenimiento para convertirlo en un sistema por completo estable. A continuación se hace una descripción del mantenimiento del SICC y de los logros obtenidos con ello.

### 5.5.1 Mantenimiento adaptativo

Dentro del mantenimiento adaptativo se debe mencionar, la actualización que se realizó del manejador de base de datos postgres de la versión 7.3.3 a la versión 7.4.1 . Esto con el fin de evitar posibles vulnerabilidades de seguridad, dado que postgres es software de licencia libre y esta propenso a tener vulnerabilidades que podrían propiciar un incidente de seguridad.

Ahora bien, dentro de esta migración a la nueva versión, se tuvieron que actualizar algunos archivos utilizados para la administración de la base de datos (creación de tablas, permisos, etc). Esto es, porque de la versión anterior a la actual algunos comandos cambiaron, por lo tanto los scripts utilizados debieron actualizarse con la nueva sintaxis de postgres.

Algo similar ocurrió con algunos triggers que también debieron ser actualizados para adaptarse a la nueva versión del manejador.

También dentro del mantenimiento adaptativo pero en cuestiones de hardware, se tuvo que implementar un nuevo servidor en las salas de UNICA ubicadas en el Anexo de la Fac. de Ingeniería puesto que la comunicación resultaba algo lenta entre las oficinas del Edif. Principal y las del Anexo. Esto se debe, a que el cableado de la red que une ambas dependencias, no permitía una conexión rápida y por ello, muchos de los procesos se veían alentados.

También dentro de este tipo de mantenimiento, se tienen contemplados posibles cambios que en un futuro pudieran presentarse y que requirieran integrarse al sistema como podrían ser, la adición de nuevas tablas o la modificación de ellas dentro de la base de datos,

nuevos tipos de datos, nuevos usuarios del sistema, etc. Sin embargo, si ocurriera así, el cambio se deberá analizar para tomar una decisión de acuerdo a su naturaleza y decidir si se realiza la modificación o tal vez se programe para un nuevo módulo del sistema. Recordemos que el programa fue diseñado bajo la filosofía orientada a objetos, por lo que tiene el respaldo de dicha tecnología y por lo tanto es más estable frente a cambios en la especificación de requerimientos.

### **5.5.2 Mantenimiento correctivo**

En este tipo de mantenimiento se tuvieron algunas modificaciones de funcionalidad y algunas más de presentación (diseño de las interfaces).

Los cambios que debieron hacerse en cuanto a la funcionalidad del sistema básicamente correspondían a fallas de programación que de alguna manera sobrepasaron las pruebas y se detectaron cuando se liberó el producto. Sin embargo, dichas fallas de programación fueron mínimas y no requirieron mayor tiempo de programación. Las más significativas fueron la reformulación de algunas consultas a la base de datos a través de la depuración de los filtros programados en ellas. Por otro lado, la presencia de dichas fallas sirvieron para verificar la modularidad del sistema puesto que los errores no se propagaron hacia niveles que no les correspondían.

En cuanto a las interfaces, los cambios fueron de presentación involucrando modificaciones tales como: nuevas posiciones para algunos controles (botones, combos, etiquetas, etc), cambio de color de algunos de ellos, de tipo de fuente, etc.

La interfaz de la web es la que recibió tal vez más cambios pero ninguno de ellos de funcionalidad, todos fueron de visualización para obtener de esa forma, una aplicación mucho más amigable para el usuario.

Como se puede intuir, este mantenimiento tampoco requirió de mucha inversión de tiempo de programación por lo que pudo llevarse a cabo satisfactoriamente.

### **5.5.3 Mantenimiento preventivo**

Dentro del mantenimiento preventivo se tiene contemplada la realización de un script que realice de forma automática los respaldos de la base de datos. Esto es porque actualmente, la base de datos se borra al término de cada año (tiempo correspondiente a dos períodos de cursos) y los respaldos se realizan de forma manual mediante el manejador. Por lo tanto, se requiere de un script que respalde y limpie automáticamente la base de datos cada año. El desarrollo de dicho script aún está en proceso, sin embargo, su implementación sería una medida que facilitaría enormemente el mantenimiento del sistema así como de la administración de la base de datos.

### **5.5.4 Mantenimiento perfectivo**

Una nueva funcionalidad que se agregó para el SICC fue el otorgamiento de permisos para algunas tablas de su base de datos para que otras aplicaciones desarrolladas en UNICA pudieran consultarlas y utilizarlas en determinados procesos.

Por lo tanto, la base de datos del SICC se ha convertido de alguna manera, en la principal base de datos de UNICA puesto que otras aplicaciones además del SICC, la utilizan para llevar a cabo sus funciones y en un futuro se tienen contemplados nuevos productos que se sincronicen con el SICC y contribuyan de esta manera a la unificación de procesos dentro de la Institución (ver cap.6 Conclusiones, subtema Perspectivas a Futuro).

## 6.1 Contribuciones de la tesis

En este trabajo de tesis se presentó el desarrollo de un sistema de software basado en la metodología OMT y el lenguaje de modelado UML. Se analizó específicamente el problema de la administración y control de la información concerniente a los cursos de cómputo que se imparten en UNICA, así como la inscripción de los alumnos a ellos. Dicho análisis fue encauzado por los preceptos que la metodología propone, además de registrar también lo concerniente al diseño e implementación del sistema.

En la parte de diseño se modeló con UML obteniendo así una perspectiva completa de todas las facetas que componen al producto y logrando establecer un escenario apropiado para la implementación de la aplicación.

En lo referente a la fase de implementación, Visual Basic resultó ser un entorno de desarrollo bastante completo a pesar de sus limitantes para soportar el 100% de la filosofía orientada a objetos, permitiendo así, lograr los objetivos planteados al inicio del desarrollo, con resultados favorables en todo momento. No se presentaron inconvenientes que en algún momento hicieran dudar acerca de la elección de Visual Basic como lenguaje de desarrollo. Por el contrario, fue una agradable sorpresa encontrar que a pesar de no ser un lenguaje orientado a objetos, permite trabajar con la filosofía y lograr productos estables, robustos y completamente funcionales.

A continuación se listan las conclusiones obtenidas a partir del desarrollo del Sistema de Inscripción y Control de Cursos (SICC) para la Unidad de Cómputo Académico (UNICA):

- ❖ Se desarrolló un sistema de cómputo capaz de controlar y administrar los procesos relacionados con las inscripciones de los cursos que se imparten en UNICA, así como la generación de constancias y reportes.
- ❖ Mediante el sistema se genera toda la información relacionada con los cursos en UNICA incluyendo las listas de grupos, recibos de apartado, comprobantes de inscripción y constancias.
- ❖ A través del sistema ahora, se puede manejar toda la información referente al personal que labora en UNICA desde el Jefe de la Unidad, pasando por los jefes de los distintos departamentos, los instructores, los becarios y finalmente los alumnos que realizan el servicio social.
- ❖ En general, todo el proceso de inscripción de los alumnos, desde el registro de los mismos directamente en las oficinas o a través del web mediante la aplicación de apartado de cursos, hasta la generación de reportes y constancias tanto para alumnos como para instructores, es ahora controlado por el sistema. Lo que se tradujo en una optimización en tiempo y forma de los métodos de inscripción que posee la Unidad.
- ❖ Se desarrollaron las herramientas para generar las constancias de los alumnos y las de los instructores así como para la creación de las listas de control para la entrega de dichas constancias. Ahora mediante el sistema, con sólo oprimir un botón, inmediatamente son creadas con lo que los tiempos de entrega de constancias se redujeron de forma considerable.
- ❖ También gracias al sistema, se agilizó el proceso de planeación de los cursos puesto que ahora se cuenta con una herramienta de cómputo estable involucrada en dicho proceso, lo que conlleva una reducción de tiempo que se traduce en mejores resultados durante los períodos de cursos.

- ❖ También se implementó de manera formal el uso de una metodología de software para el desarrollo de sistemas dentro de la Unidad. Además, de que se ha profundizado en la investigación de los nuevos procesos involucrados en la Ingeniería de software, así como del desarrollo de nuevas metodologías y herramientas dentro de éste campo.
- ❖ Por otro lado, todo el trabajo y esfuerzo invertido durante el desarrollo del SICC, generó un grupo de expertos en el tema, lo que a su vez se tradujo en nuevos cursos, apoyos a otros sistemas y asesorías especializadas, por lo que al día de hoy, podemos hablar de la formación de recursos humanos de calidad con conocimientos específicos en el tema como consecuencia de su participación en el sistema.
- ❖ Finalmente podemos decir, que el sistema se ha convertido en la parte medular de otros sistemas que existen en UNICA como el SECC (Sistema de Encuestas y Cursos de Cómputo) y su parte de Evaluación de Instructores y el CORA (Sistema de Control y Reporte de Asesorías). Dichos sistemas utilizan la base de datos diseñada en el SICC para llevar a cabo sus procesos mediante consultas a ella.

## 6.2 Perspectivas a Futuro

Dados los alcances que tuvo este sistema, que originalmente estaba destinado a optimizar las inscripciones a los cursos y la creación de ellos, se ha transformado en el sistema de información de UNICA, ya que cuenta con los datos del personal que labora en la Unidad, así como la información referente al historial de cursos impartidos en donde se puede obtener registros acerca del número de cursos impartidos en determinado año, así como las personas que impartieron los cursos, los alumnos asistentes a ellos, sus calificaciones y más.

Por todo esto, la principal perspectiva a futuro es que el sistema crezca adaptándose a los cambios de ambiente, así como a los ajustes en cuanto a la organización que se adopte en la Unidad. Muy probablemente los sistemas que se desarrollen en la unidad y que utilicen información proveniente del SICC, sean integrados con él, logrando así la unificación de procesos de cómputo y administrando de esta manera todo desde una misma unidad central.

Por otro lado, la tendencia de los servicios integrales en Internet vislumbra otra perspectiva a futuro para el SICC: el crecimiento del sistema a través de la web, mediante la incorporación de empresas certificadoras que permitan la conclusión del proceso de inscripción vía web. Nos referimos principalmente a los pagos de los cursos en Internet. Como se mencionó a lo largo del presente trabajo de tesis, el sistema por el momento sólo permite el apartado de los cursos vía web a través del sitio diseñado con ese fin. A pesar de ser un sistema de apartado bastante confiable y seguro, sólo cubre una parte del proceso de inscripción. Por lo tanto, en un futuro se planea incorporar las mejoras necesarias para que se puedan apartar y pagar los cursos en Internet, de tal forma que el alumno sólo se presente al curso el día y el horario señalados, con su comprobante de pago y sólo se disponga a tomar el curso elegido sin mayor trámite.





# Manual de Usuario

## SICC

### *Sistema de Inscripciones y Control de Cursos*

### *Grupos*

## 1. Iniciando sesión

Al ejecutar el sistema se presenta una ventana como la que se muestra en la figura 1.1. Esta ventana tiene por objetivo autenticar al usuario.

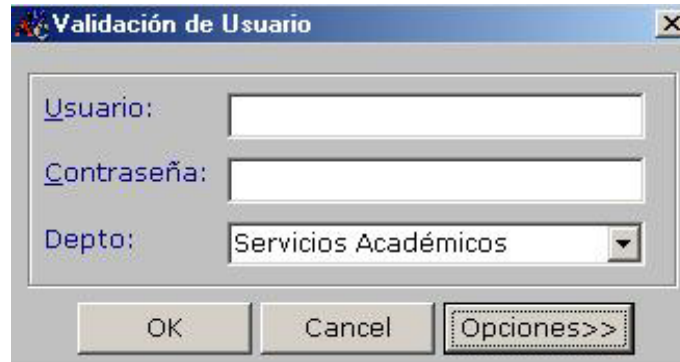


Fig 1.1

En ella se pedirá al usuario que proporcione su login y contraseña personal de acuerdo al departamento al cual pertenece. El departamento puede ser:

- Servicios Académicos
- Investigación y Desarrollo
- Redes y Operación de Servidores
- UNICA

El departamento se seleccionará en el combo llamado Depto:

También se cuenta con la opción de seleccionar el servidor, el nombre de la Base de Datos y el puerto por el cual se va a realizar la conexión. El servidor debe ser el que administra a la Base de Datos. En la sección de “Base de Datos” se coloca el nombre de la base en la cual podemos hacer las consultas y modificaciones a la información que manipula el sistema. El puerto es el enlace lógico a través del cual se va a conectar el sistema. Lo anterior se despliega con el botón “Opciones>>” y se observa en la figura 1.2:



Fig 1.2

Inmediatamente después de teclear el nombre de usuario y la contraseña, en caso de haber error por haber tecleado un login incorrecto o password incorrecto entonces aparecerá el siguiente mensaje:

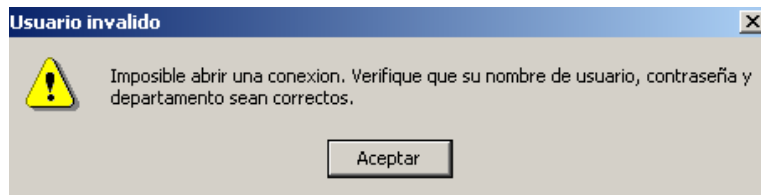


Fig 1.3

Se le da un clic en aceptar y se borra lo escrito en las cajas de texto de login y password, por lo que se deben de volver a introducir correctamente. Una vez hecho esto aparecerá la siguiente pantalla de Bienvenida al Sistema:



Fig 1.4

En donde se nos indica que el sistema esta siendo cargado, una vez concluido este proceso aparecerá la siguiente pantalla:

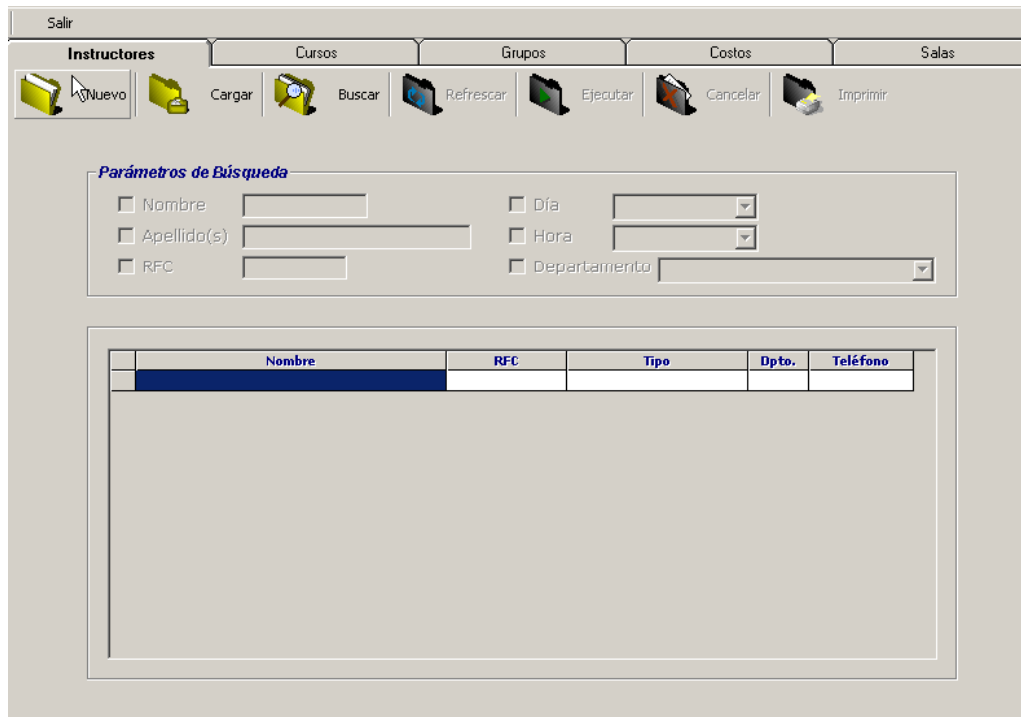


Fig 1.5

En esta pantalla tenemos las opciones de:

- ❖ Salir del sistema
- ❖ Administrar a los instructores, es decir darlos de alta o bien ver sus datos.
- ❖ Administrar los cursos que se van a impartir.
- ❖ Administrar los grupos que existen
- ❖ Administrar los costos de los cursos
- ❖ Administrar las salas.

Vamos a darle clic en la pestaña de grupos:

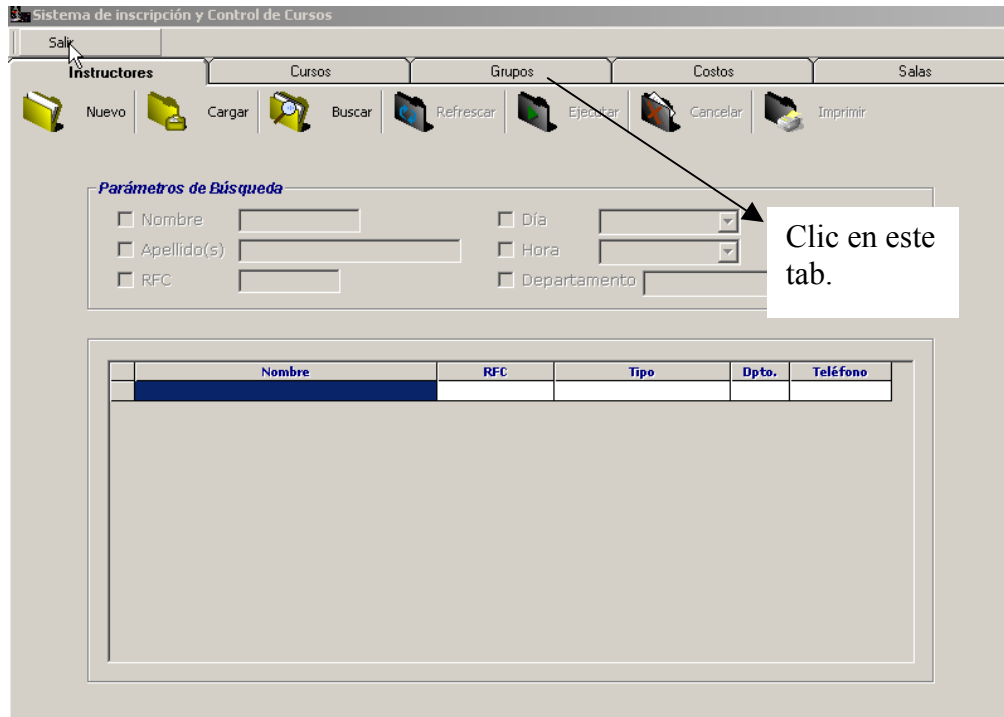


Fig 1.6

Una vez que se ha dado clic en esta parte aparecerá la siguiente ventana:

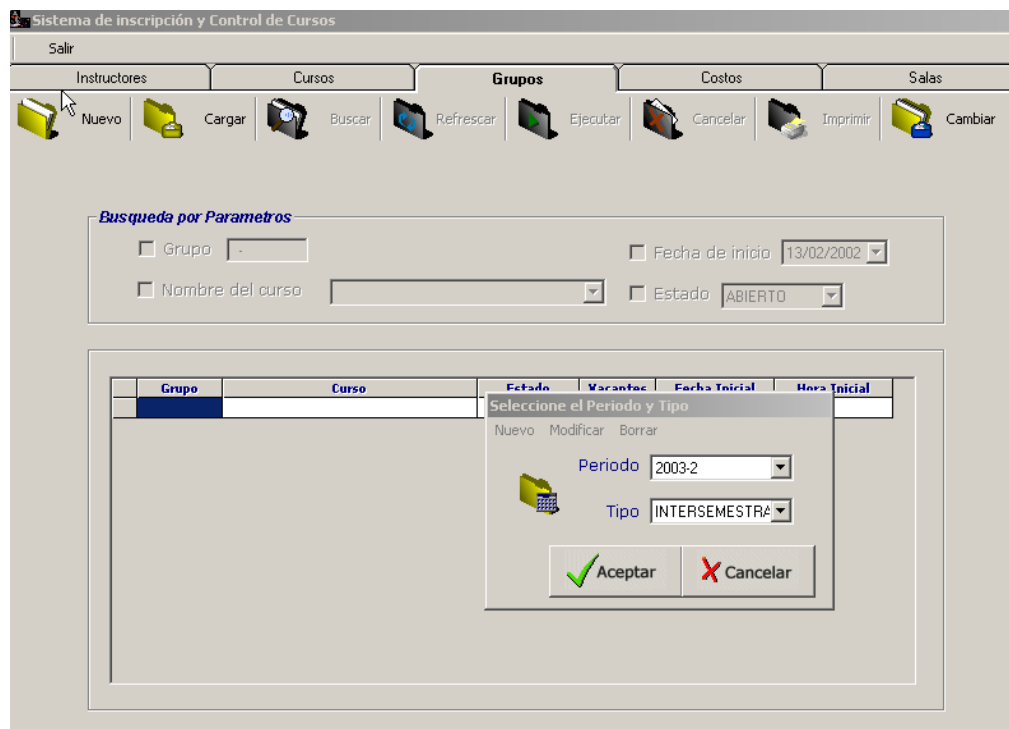


Fig 1.7

Esta pantalla nos muestra la administración que se le puede dar a los grupos, pero antes se debe de seleccionar el tipo de grupo en la pantalla pequeña que se ve en la figura 1.7 y que es la que a continuación se muestra:



Fig 1.8

En el combo llamado Periodo, se debe seleccionar el tipo de periodo al que corresponde el grupo (semestre). Mientras que en el combo llamado Tipo, se debe seleccionar el tipo de grupo que se desea administrar. Los tipos de grupo pueden ser:

- ❖ Intersemestrales
- ❖ Semestrales
- ❖ Fines de Semana
- ❖ Externos
- ❖ Internos

Una vez que tengamos el periodo y el tipo de grupo, damos clic en “Aceptar” lo cual despliega la siguiente pantalla:

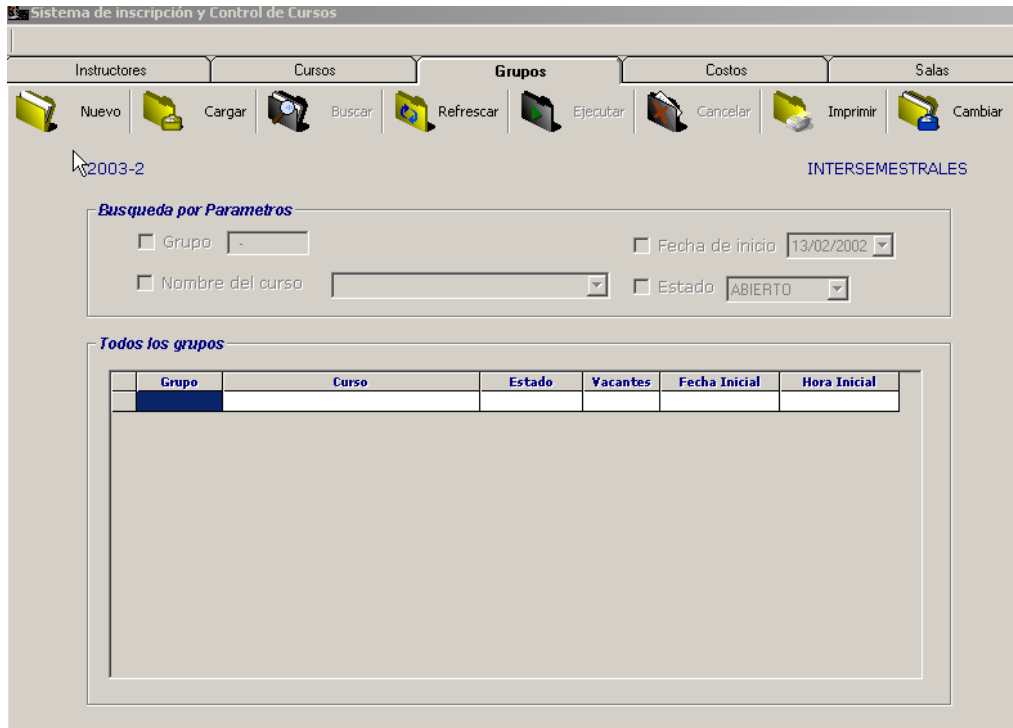


Fig. 1.9

Una vez en esta pantalla tenemos las siguientes opciones para administrar los grupos:

- ❖ Nuevo: Dar de alta un nuevo Grupo
- ❖ Cargar: Carga en pantalla todos los grupos existentes en la Base de Datos, de acuerdo al Periodo y Tipo seleccionados previamente (fig 1.8)
- ❖ Buscar : Busca dentro de la Base de Datos algún grupo de acuerdo a diferentes parámetros (esta opción se habilita al momento de cargar los grupos)
- ❖ Refrescar: Actualiza la consulta en caso de que se hayan dado modificaciones de algún registro.
- ❖ Ejecutar: Esta opción se utiliza para cuando se realiza la búsqueda de algún grupo.
- ❖ Cancelar: Cancela las acciones realizadas.
- ❖ Imprimir: Permite imprimir los grupos que se están visualizando.
- ❖ Cambiar: Despliega nuevamente la ventana de la fig 1.8, lo cual nos permite cambiar el tipo y el periodo de los grupos que se están administrando.

A continuación se verán más detalladamente las opciones:

## 2. NUEVO

Al darle clic al botón de nuevo aparece la siguiente ventana:

The screenshot shows a web application window titled "Sistema de inscripción y Control de Cursos". The window has a toolbar with buttons: Regresar, Nuevo, Actualizar, Modificar, Borrar, Cancelar, and Imprimir. Below the toolbar, the page is for "Semestre 2000-3" and "INTERSEMESTRALES".

The form is divided into several sections:

- Información General:** Includes fields for Grupo (dropdown), Curso (dropdown: BASES DE DATOS ORACLE), Sala (dropdown: FUNDACION UNAM FI), Sección (dropdown: E), Estado (dropdown: ABIERTO), Vacantes, Inscritos, and Confirmados. It also has Nivel (dropdown: A) and Costo (\$ 315,00).
- Calendario:** Includes Fecha (dropdown: 11/07/2003), Horario (dropdown: 08:00 - 09:00), and Duración (input field).
- Instructores:** Includes Instructor A (dropdown: ALVARO FERNANDEZ GOMEZ) and Instructor B (checkbox and dropdown).
- Alumnos:** A table with columns Nombre, RFC, and C.

Fig 2.1

Esta pantalla nos permite capturar la información general del grupo, las fechas en las que se llevara a cabo, los instructores que van a estar a cargo de él y los alumnos que se encuentran inscritos, no es posible darlos de alta desde aquí, solo sirve para visualizar quienes se encuentran inscritos.

Una vez llenos los campos, se da clic en el botón de Actualizar

Luego de darle clic al botón de nuevo, se presenta una nueva pantalla, en la cual podremos dar de alta un nuevo grupo en la base de datos:



Fig 2.2

Como podemos ver en la fig. 2.2 se piden los siguientes datos para poder dar de alta un grupo:

#### Información general del Grupo:

- ❖ Grupo: Es el nombre del grupo que se da de alta
- ❖ Curso: Nombre del curso que se da de alta
- ❖ Salas: Sala en la cual se va a impartir el curso
- ❖ Sección: Da el nombre de la sala donde se impartirá el curso, recordando que UNICA posee varias salas en diferentes edificios de la Fac. de Ingeniería.
- ❖ Estado : Indica si el grupo esta Abierto o Cancelado
- ❖ Vacantes: Indica el número de lugares que quedan libres una vez que los alumnos puedan inscribirse al grupo.
- ❖ Inscritos: Indica el número de alumnos que se han inscrito al grupo.
- ❖ Confirmados: Indica el número de alumnos que están inscritos y además que ya están confirmados para tomar el curso.

#### Calendario:

- ❖ Fecha: Indica la fecha de inicio y fin del grupo
- ❖ Horario: Indica la hora de inicio y fin del grupo
- ❖ Duración: Indica cuanto va a durar el curso en horas

#### Instructores:

- ❖ Instructor A: Se asigna un instructor titular al grupo, al darle clic al menú de instructores que se encuentra al lado de la etiqueta “instructor A”, se nos mostrará una lista con los nombres de los instructores dados de alta en la Base de Datos.
- ❖ Instructor B: Se puede asignar un instructor de apoyo, al igual que en caso del instructor A se nos muestra una lista.

#### Alumnos:

Una vez que el curso esta dado de alta, esta parte de la ventana mostrara a los alumnos que se encuentran inscritos en el grupo.

De esta manera damos de alta un grupo, como se puede observar en la figura 2.2, en la parte superior derecha, debajo de la barra de navegación se indica el tipo de periodo en el cual se impartirá el curso (semestral, intersemestral, etc.).

Los datos de número de grupo y duración son campos obligatorios, así que deben ser llenados o el sistema no dejara continuar.

Una vez realizado lo anterior, se puede dar de alta el grupo dando clic al botón Actualizar, y se finaliza el proceso de creación de un nuevo grupo.

### 3. CARGAR

Regresando a la pantalla principal del sistema en la parte de Grupos, se pueden cargar o visualizar los grupos que se encuentran dados de alta en la Base de Datos.

Para realizar esto se le da clic en el botón de la barra de herramientas que dice Cargar, una vez realizado esto, los grupos aparecerán en la parte inferior de la pantalla como lo muestra la fig. 3.1

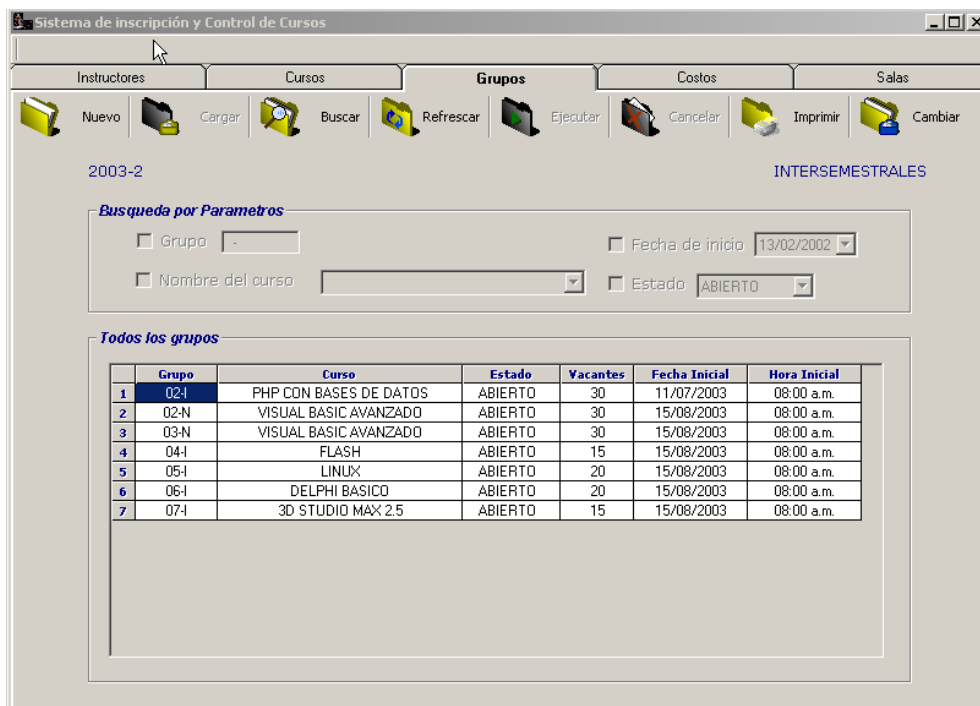


Fig. 3.1

Automáticamente el botón de Cargar se desactiva y los grupos son mostrados con los siguientes datos:

- ❖ Grupo : Indica el nombre del grupo
- ❖ Curso: Nombre del curso
- ❖ Estado : Si esta Abierto o Cancelado
- ❖ Duración: Horas que dura el curso.
- ❖ Fecha inicial : Cuando empieza
- ❖ Hora inicial: Hora de inicio del curso.

## 4. BUSCAR

Si se necesita encontrar algún grupo en particular o una serie de grupos se cuenta con la opción de Buscar, esta se puede realizar por diferentes parámetros:

- ❖ Grupo : Introduciendo el nombre
- ❖ Nombre del Curso
- ❖ Fecha de Inicio
- ❖ Estado

Como se muestra en la fig. 4.1 al dar clic en el botón Buscar se activa el campo de búsqueda por parámetros y se puede seleccionar el campo por el cual se desee buscar:

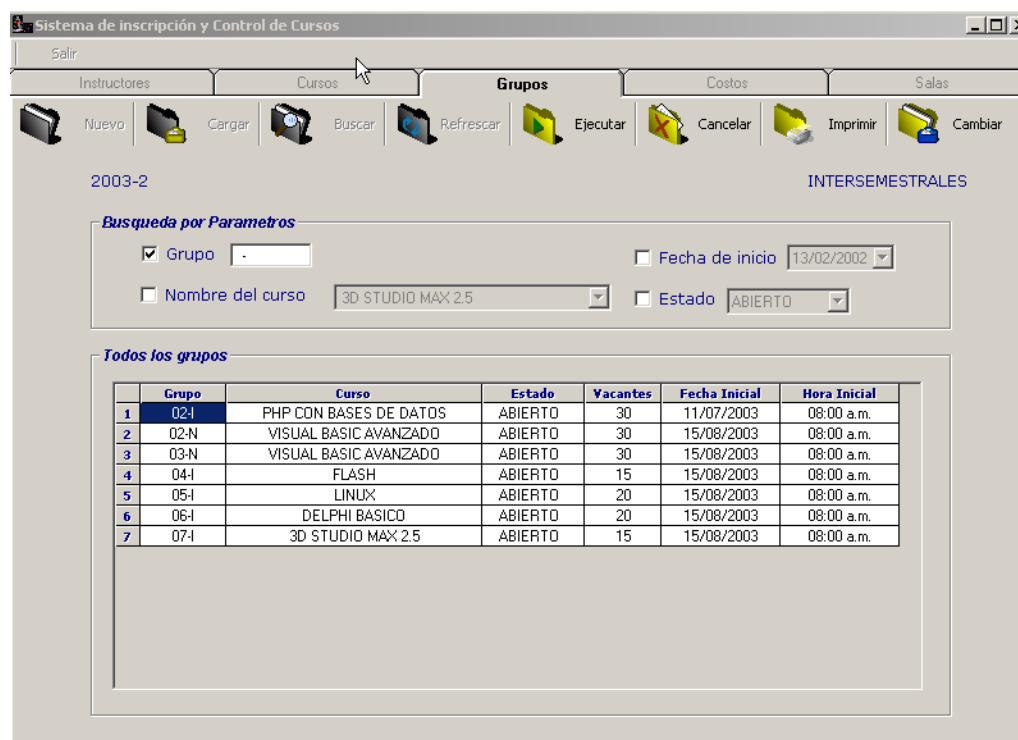


Fig. 4.1

En la figura podemos ver que esta seleccionada la casilla de búsqueda por Grupo, en esta caso se habilita la caja de texto que se encuentra frente a la opción y permite introducir el numero de grupo que se busca, de igual manera si se selecciona la opción de nombre de curso

se habilita el menú donde se puede elegir el nombre del curso. Se pueden seleccionar varios parámetros de búsqueda.

Una vez completada(s) la(s) opción(es) de búsqueda se da clic en el botón de ejecutar.

En caso de que la búsqueda no tenga ningún resultado que mostrar se despliega el mensaje de la fig. 4.2

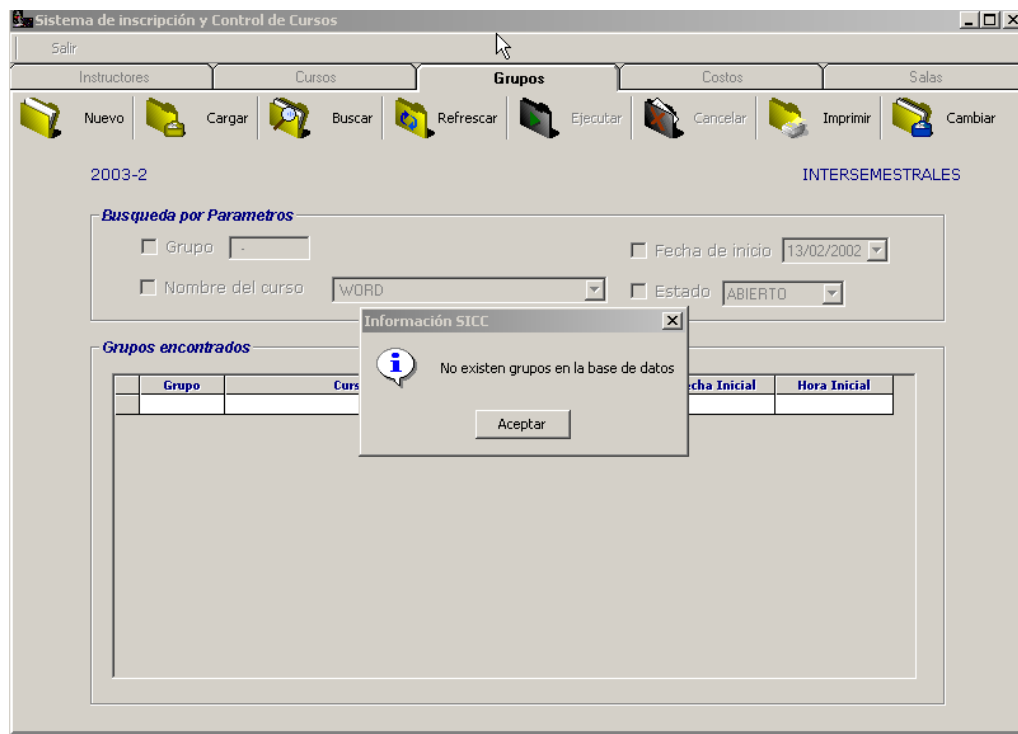


Fig. 4.2

En caso de que si tenga resultados que mostrar se carga el grupo en pantalla como lo muestra la fig. 4.3

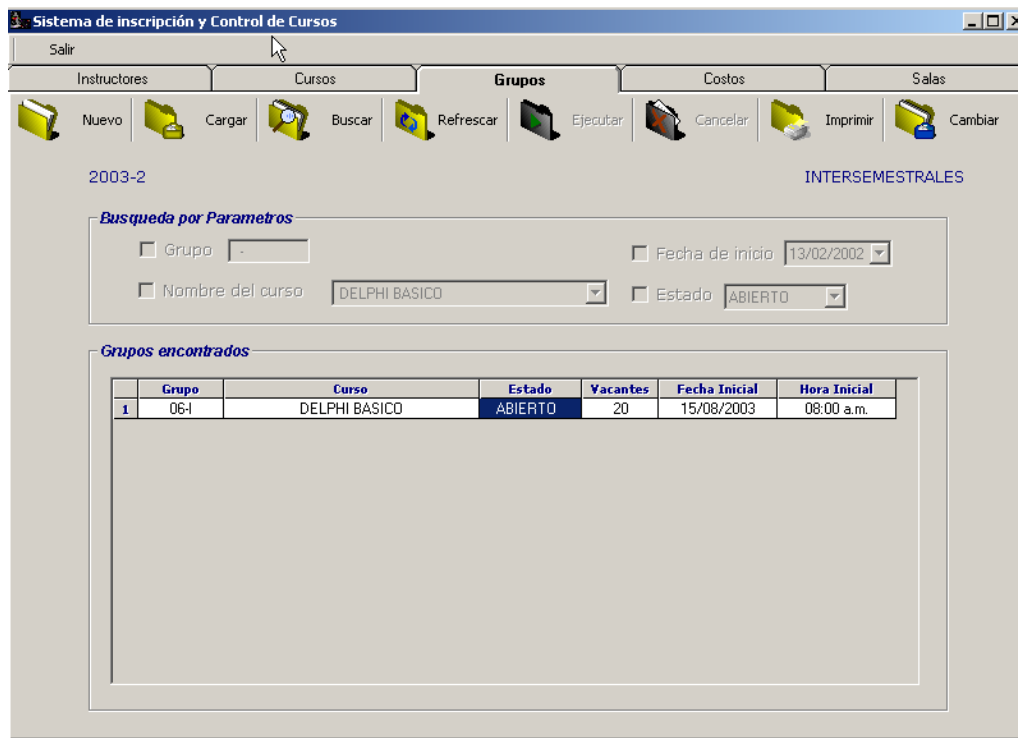


Fig.4.3

Si se desea ver más información de algún curso en particular se le da doble clic sobre el curso como lo indica la figura 4.4:

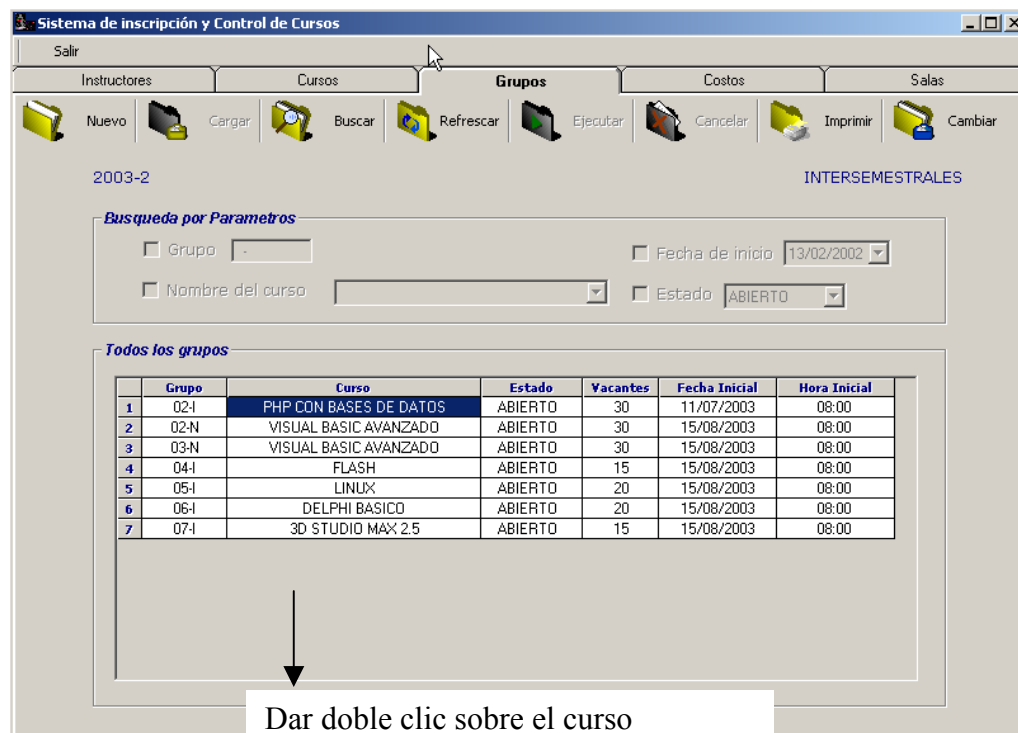


Fig 4.4

Una vez realizado esto, se presenta nuevamente la ventana de la figura 2.1 que nos permitía dar de alta un nuevo grupo, pero esta vez con la información relativa al curso que se eligió como lo muestra la fig 4.5:

2003-2

**Información General**

Grupo: 071

Curso: 3D STUDIO MAX 2.5

Sala: BIBLIOTECA ANEXO Sección: A

Estado: ABIERTO

Vacantes: 15 Inscritos: 0 Confirmados: 0

Nivel: A Costo: \$ 315,00

**Calendario**

Fecha: 15/08/2003 - 30/06/2003

Horario: 08:00 - 09:00

Duración: 12 hrs.

**Instructores**

Instructor A: FERNANDO FLORES SALGADO

Instructor B: JUAN ARMANDO

**Alumnos**

Nombre	RFC	C
--------	-----	---

Primero Anterior Siguiente Ultimo

Fig. 4.5

Aquí podemos observar que se cargo toda la información relativa al curso, para desplazarnos entre los diferentes cursos que existen en la base, en caso de que deseemos ver los datos de otros grupos, utilizaremos la barra de navegación que se encuentra en la parte inferior de la pantalla como lo muestra la fig 4.6

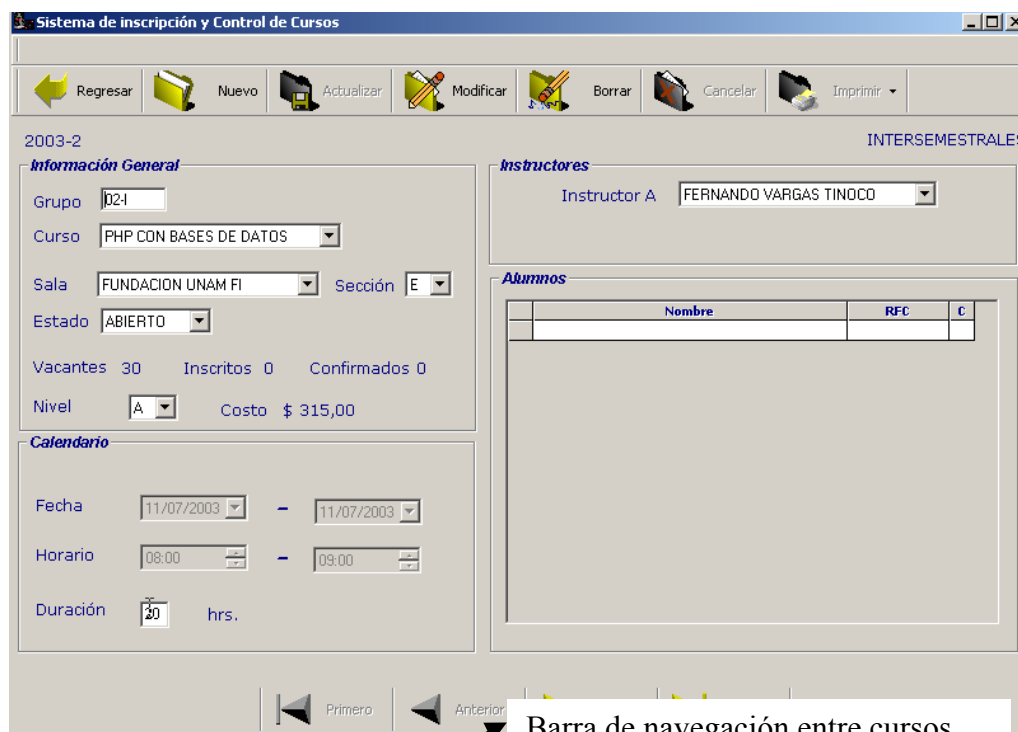


Fig. 4.6

Como lo muestra la figura anterior, se recorrió hasta el primer registro, por lo cual los botones respectivos de Primero y Anterior se deshabilitan, habilitándose de nuevo si se continua la navegación hacia adelante entre los demás grupos de la Base de Datos.

La opción de la Barra de Navegación llamada Regresar, nos permite regresar a la pantalla anterior en la cual se visualizaban los grupos, y se realizaban búsquedas.

En esta nueva pantalla, se nos presentan en la Barra de Navegación dos opciones nuevas:

- ❖ Modificar
- ❖ Borrar

La primera opción nos permite como su nombre lo indica modificar algún dato acerca de ese curso como por ejemplo el horario o bien el estado o cualquiera de los datos del curso.



La segunda opción nos permite el borrado de un grupo siempre y cuando no existan alumnos inscritos a él. Dicha opción visualiza una pantalla de confirmación como la que muestra la fig. 4.7

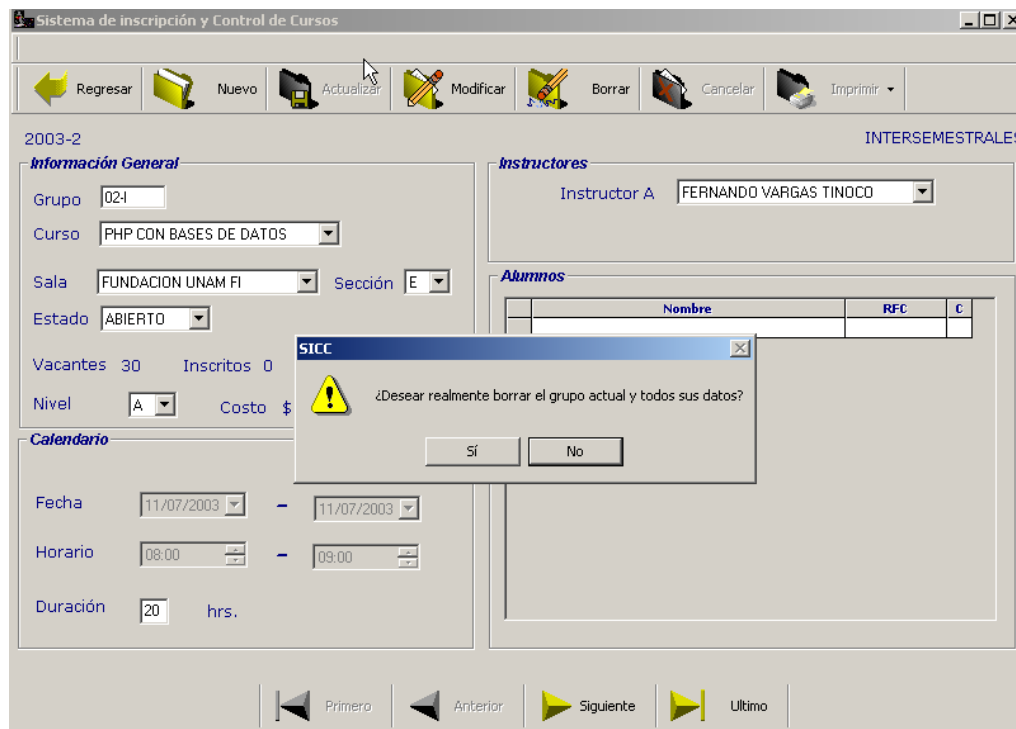


Fig 4.7

Si deseamos continuar se le da clic en “Si”, de lo contrario si no queremos borrar el grupo se le da clic en “No”

## 5. IMPRIMIR

Este botón de la barra de herramientas nos permite imprimir un reporte de los datos que se estén visualizando en ese momento, por ejemplo si estamos consultando todos los grupos dados de alta en la base de datos, y oprimimos el botón de imprimir se nos generará una pantalla como la que muestra la fig. 5.1

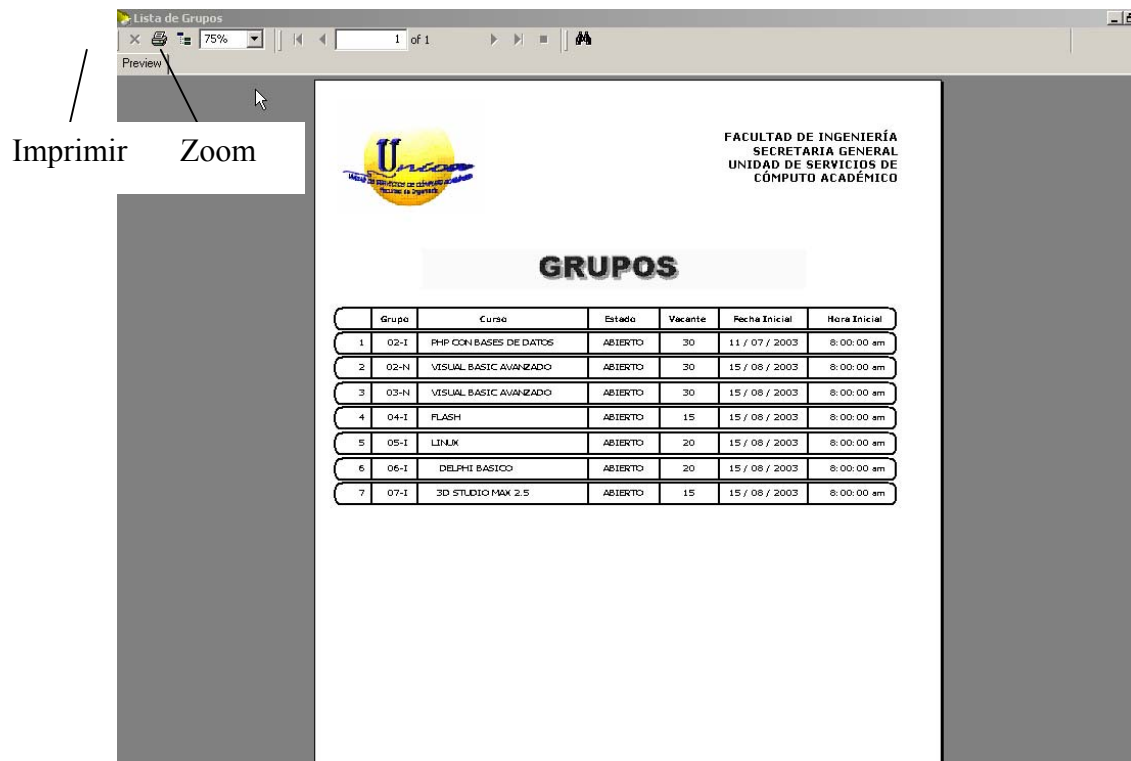


Fig. 5.1

Este reporte se puede visualizar en mayor o menor tamaño (zoom), mandarlo imprimir o bien cerrar la ventana y regresar al sistema.

## 6. Salir del Sistema

Para salir del sistema por completo tenemos una opción general de Salida, la cual se encuentra en la parte superior izquierda de la interfaz del Sistema, como lo muestra la fig 6.1

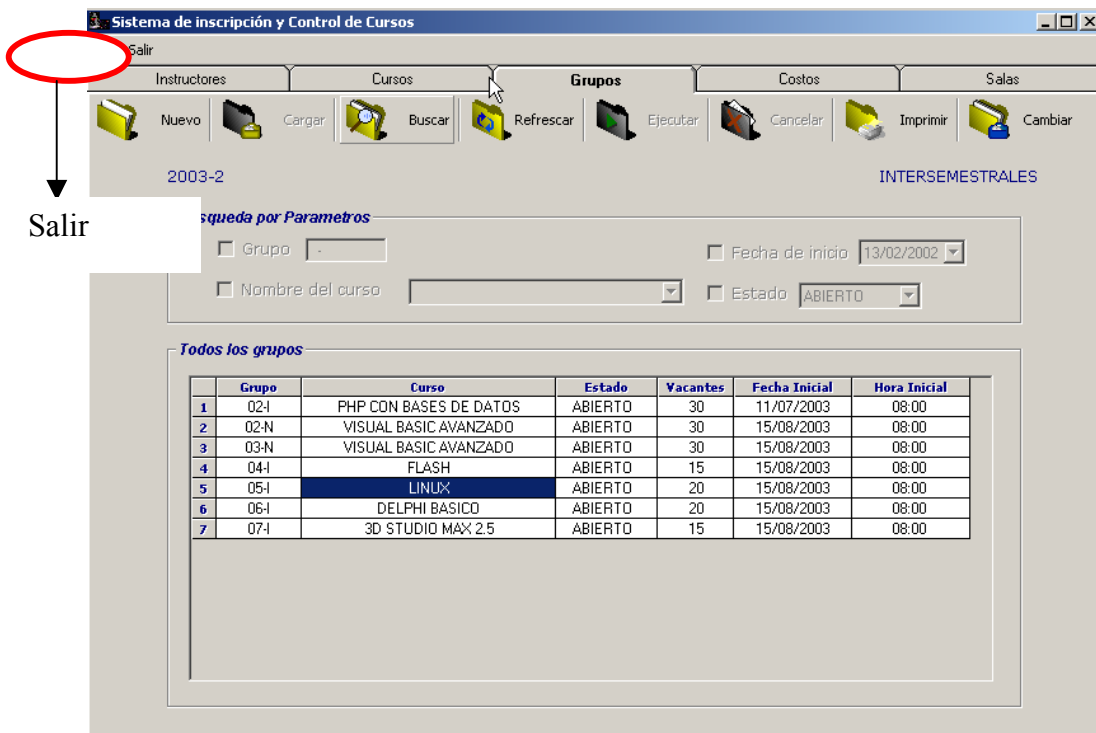


Fig 6.1

En el momento que se desee y navegando por cualquiera de las pantallas se tiene esta opción, la cual permite la salida por completo del Sistema.

---

## Bibliografía

JACOBSON, Ivar [et all]. 2000. El Proceso Unificado de Desarrollo de Software. Madrid. Addison Wesley. 464 p.

PRESSMAN, Roger S. 1997. Ingeniería de Software: un enfoque práctico. 4ª.edición. México, Mcgraw-Hill. 570 p.

RUMBAUGH, James [et all]. 1996. Modelado y Diseño Orientados a Objetos. España. Prentice – Hall. 635 p.

SCHMULLER, Joseph. 2000. Aprendiendo UML en 24 horas. México. Pearson Educacion. 448 p.

DEPARTAMENTO DE SISTEMAS, INFORMACIÓN Y COMPUTACIÓN.  
Ejemplo de desarrollo software utilizando la metodología RUP. 2004. [en línea].  
<<http://www.dsic.upv.es/asignaturas/facultad/lisi/ejemplorup/Pruebas.html#top>>

UNIVERSIDAD POLITECNICA VALENCIA. 2003. Curso: “Desarrollo de Software Orientado a Objetos usando UML”. [en línea]. [consulta: julio 2004]  
<<http://www.dsic.upv.es/~uml/index.html#mat>>

LA WEB DEL PROGRAMADOR. 2004. Curso de Visual Basic Avanzado. [en línea]. < <http://www.lawebdelprogramador.com/cursos/vbavanzado/default.php>>

DIRECCION GENERAL DE SERVICIOS DE COMPUTO ACADEMICO.  
DIRECCIÓN DE SISTEMAS DPTO. DE INNOVACIÓN. 2004. Herramientas CASE para el Análisis y Diseño Estructurado. [en línea]. <<http://sistemas.dgsca.unam.mx/publica/pdf/casestru.pdf>>

---

FACULTAD DE INFORMATICA UPV/EHU INGENIERÍA EN  
INFORMATICA DEPARTAMENTO DE LENGUAJES Y SISTEMAS INFORMATICOS.  
2003. Curso 2003/04 Pruebas de Software. [en línea]. <[http://siul02.si.ehu.es/~alfredo/  
iso/Tema3.pdf](http://siul02.si.ehu.es/~alfredo/iso/Tema3.pdf)>