UNIVERSIDAD NACIONAL AUTÓNOMA
DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES
CUAUTITLÁN

DESARROLLO E IMPLEMENTACION DEL SISTEMA PARA LA ASIGNACIÓN DE LA DOCENCIA EN LA FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLÁN.

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

LICENCIADO EN INFORMÁTICA

PRESENTA:
IVÁN NÚÑEZ CONSUELOS

ASESOR: DR. ROBERTO MURCIO VILLANUEVA

CUAUTITLÁN IZCALLI, EDO. DE MEX. 2011

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLÁN
UNIDAD DE ADMINISTRACIÓN ESCOLAR
DEPARTAMENTO DE EXÁMENES PROFESIONALES

ASUNTO: VOTO APROBATORIO

DRA. SUEMI RODRÍGUEZ ROMO
DIRECTORA DE LA FES CUAUTITLÁN
PRESENTE

ATN: L.A. ARACELI HERRERA HERNÁNDEZ
Jefa del Departamento de Exámenes
Profesionales de la FES Cuautitlán

Con base en el Art. 28 del Reglamento de Exámenes Profesionales nos permitimos comunicar a usted que revisamos **LA TESIS**:

Desarrollo e implementación del Sistema de Asignaciones de la Facultad de Estudios Superiores Cuautitlán "SAFESC"

Que presenta el pasante: **Iván Nuñez Consuelos**
Con número de cuenta: **09824184-2** para obtener el Título de: **Licenciado en Informática**

Considerando que dicho trabajo reúne los requisitos necesarios para ser discutido en el **EXAMEN PROFESIONAL** correspondiente, otorgamos nuestro **VOTO APROBATORIO**.

ATENTAMENTE
"POR MI RAZA HABLARA EL ESPÍRITU"
Cuautitlán Izcallí, Méx. a 20 de Septiembre de 2011.

PROFESORES QUE INTEGRAN EL JURADO

	NOMBRE	**FIRMA**
PRESIDENTE	MCC. Valentín Roldán Vázquez	
VOCAL	DCC. Roberto Murcio Villanueva	
SECRETARIO	L.A. Jorge Altamira Ibarra	
1er SUPLENTE	Dr. Oscar Ibáñez Orozco	
2do SUPLENTE	I.MAC. Leonel G. López Salazar	

NOTA: los sinodales suplentes están obligados a presentarse el día y hora del Examen Profesional (art. 120).
HHA/pm

DEDICATORIAS Y AGRADECIMIENTOS

A Dios

Por darme la oportunidad de realizar este sueño.

A Cinthya y Johan

Solo por existir y por ser el motor que hace que mi mundo gire, gracias por confiar en mí y gracias por darme la fuerza para salir adelante cada día.

A mis padres y hermanos

Gracias por hacer de mi lo que soy, este título es de ustedes.

Al profesor Leonel López

Gracias por todo su apoyo, sus enseñanzas, por creer en mí y por ser un ejemplo a seguir.

A mi asesor

Dr. Roberto murcio gracias por tu apoyo, tu guía y tu confianza.

A mis amigos Alex, Alfonso, Gerardo, Josafat, Julio y Raúl

Gracias por hacer de esto una aventura, gracias amigos.

A la Universidad Nacional Autónoma de México

Por hacer de mi una mejor persona.

ÍNDICE

INTRODUCCIÓN

En la actualidad, en México y en el mundo, la tecnología avanza a pasos agigantados, brindando ventajas competitivas a aquellas organizaciones que se mantienen actualizadas tecnológicamente. El factor tecnológico es imprescindible en una organización, ya que de este puede depender el éxito o fracaso de la misma, por lo tanto es necesario renovar e innovar constantemente la tecnología con la que cuenta.

En la Facultad de Estudios Superiores Cuautitlán de la UNAM se contaba con un sistema para la administración de los grupos que se crean semestralmente, los docentes que son asignados a dichos grupos, la revisión por parte de la Unidad de Asuntos del Personal Académico y las aprobaciones del Consejo Técnico, pero a causa de que dicho sistema presentaba serias restricciones para su mantenimiento y funcionamiento tales como:

- No se cuenta con el código fuente del mismo.
- No existe documentación alguna ni diccionario de datos de la base de datos.
- Las tecnologías utilizadas para su construcción requieren de licencias para su uso, por lo que hay que pagar para poder utilizarlas.
- Tiene una arquitectura cliente servidor de dos capas y la aplicación corre del lado del cliente, por lo cual es necesario instalar la aplicación en cada equipo que requiera acceso al sistema y su tecnología en general es antigua

Se ha implementado un nuevo sistema en la facultad el cual fue llamado Sistema de Asignaciones de la Facultad de Estudios Superiores Cuautitlán (SAFESC) y que cuenta con una tecnología cliente servidor de n capas, lo que brinda una gran accesibilidad y disponibilidad al sistema por parte de los usuarios, ya que solo requieren una computadora con acceso a internet y el navegador Mozilla Firefox[1] para poder acceder al sistema; las tecnologías utilizadas para su construcción cuentan con una licencia publica general o GNU por sus siglas en ingles, por lo que no se tiene que pagar para utilizarlas, además de que son de las tecnologías más utilizadas actualmente, lo que brinda confiabilidad y robustez a la aplicación

[1] www.mozilla.com/es-ES/firefox/

debido a que se cuenta con un gran número de usuarios a nivel mundial y tienen amplia documentación.

Lo anterior permitió a la facultad renovar su tecnología y mantenerse a la vanguardia, además de mejorar de forma considerable la administración del proceso antes mencionado y brindar una herramienta fácil de utilizar que les permite aumentar su eficiencia y reducir sus tiempos de proceso.

CAPÍTULO I. ANTECEDENTES

1.1 Antecedentes de la Facultad de Estudios Superiores Cuautitlán

La Facultad de Estudios Superiores Cuautitlán (FESC) de la UNAM imparte actualmente 15 carreras: Administración, Bioquímica Diagnostica, Contaduría, Diseño y Comunicación Visual, Farmacia, Informática, Ingeniería Agrícola, Ingeniería en Alimentos, Ingeniería Mecánica Eléctrica, Ingeniería Química, Medicina Veterinaria y Zootecnia, Química, Químico Farmacéutico Biólogo, Química Industrial y Tecnología.

Además de las licenciaturas en la FESC se imparten los posgrados de: Posgrado en Artes Visuales, Posgrado en Ciencias de la Administración, Posgrado en Ciencia e Ingeniería de la Computación, Maestría y Doctorado en Ciencias de la Producción y Salud Animal, Maestría y Doctorado en Ciencias Químicas, Maestría en Docencia para la Educación Media Superior, Programa de Maestría y Doctorado en Ingeniería y Especialidad en Producción de Ovinos y Caprinos.

Hoy tiene un total de 71 cátedras de investigación, 23 proyectos en el Programa de Apoyo para Proyectos de Investigación e Innovación Tecnológica (PAPIIT) y 16 en el CONACYT.

La facultad cuenta con 49 de sus profesores en el Sistema Nacional de Investigadores (13 candidatos, 25 nivel I, 7 nivel II y 4 del nivel III).

Para el desarrollo de sus funciones la Facultad cuenta con 1,378 académicos, de los cuales 163 son ayudantes de profesor, 901 profesores de asignatura, 227 profesores de carrera, 84 técnicos académicos, 2 investigadores y 1 profesor emérito.

La población estudiantil es de 12,916 alumnos a nivel licenciatura y 205 a nivel posgrado en el periodo escolar 2011-1.

Todo lo anterior es parte de las actividades sustantivas de la facultad acordes con la misión y visión de la misma.

1.1.1 Misión y visión de la FESC

Misión

La Facultad de Estudios Superiores Cuautitlán, entidad académica descentralizada de la UNAM creada en 1974, realiza docencia a nivel de licenciatura y posgrado en las áreas de las Ciencias Químicas, Ingenierías, Administración, Agropecuarias y en Artes y Humanidades para instruir, educar y formar profesionales de alto nivel, de fácil inserción laboral, con un claro proyecto de vida y vocación de servicio a su comunidad y al país.

Realiza investigación que busca contribuir al avance del conocimiento científico y tecnológico, a la solución de retos y problemas de interés regional y nacional. Por medio de sus servicios de extensión y difusión constituye la mejor propuesta de desarrollo educativo y cultural en su zona de influencia.

Para realizar estas funciones, la FESC se ha organizado de forma departamental y matricial con un enfoque multi, ínter y transdisciplinario.

Cuenta con profesores e investigadores con formación académica y profesional relevante y pertinente, acorde a las áreas que cultiva y, con infraestructura que le permite desarrollar sus actividades sustantivas.

Los principios que guían sus actividades son la libertad de cátedra, la justicia social, la equidad, la creatividad y el liderazgo para el desarrollo educativo de la zona.

Visión

Es una Facultad reconocida por la UNAM y otras instituciones de educación superior nacionales e internacionales por la calidad del aprendizaje de sus alumnos que recibirán una educación pertinente con programas de estudio dinámicos, flexibles y actualizados que han sido acreditados, gracias a la formación y al compromiso de su planta académica, a la creación, aplicación y comunicación del conocimiento que genera y a su significativa vinculación con su zona de influencia, constituyéndose así como un polo de desarrollo y punto de encuentro de nuestra entidad con el entorno que la alberga, mediante procedimientos decididos y consensuados con la comunidad, que refleja su perfil multidisciplinario.

Es referente regional y generadora de conocimientos, tecnologías relevantes y pertinentes, y fuente de superación permanente de su comunidad (sus profesores, estudiantes y trabajadores administrativos).

La FESC es una entidad universitaria organizada bajo un modelo de gestión de la calidad, eficiente que se refleja en la certificación de sus laboratorios, procesos administrativos, en la calidad y pertinencia de su infraestructura de apoyo a la docencia e investigación y una eficiente transparente gestión de recursos humanos, financieros y materiales.

1.1.2 Estructura de la FESC

Dirección

La dirección de la Facultad tiene como visión instruir, educar y formar profesionales de alto nivel, de fácil inserción laboral, con un claro proyecto de vida y vocación de servicio a su comunidad y al país. Los principios que guían sus actividades son la libertad de cátedra, la justicia social, la equidad, la creatividad y el liderazgo para el desarrollo educativo de la zona.

Secretaría Administrativa

La Secretaría Administrativa es un apoyo fundamental para el desarrollo de las actividades de la Facultad de Estudios Superiores Cuautitlán de la Universidad Nacional Autónoma de México, ya que de ella emanan la mayoría de los tramites ante otras instancias internas y externas y los servicios a las instalaciones; su misión es optimizar los recursos humanos, financieros y materiales para apoyar el cumplimiento de los objetivos sustantivos de la FESC y de la UNAM.

Secretaria de Planeación

La Secretaria de Planeación es la instancia responsable de asesorar a la Dirección de la Facultad en materia de planeación, seguimiento y evaluación institucionales, para apoyar y fortalecer el desarrollo de su natural desempeño; así como servir de enlace entre la Facultad con las diferentes instancias de la UNAM, para fortalecer el trabajo en conjunto durante el proceso de planeación institucional.

Coordinación de Extensión Universitaria

La coordinación de Extensión Universitaria se encarga de coadyuvar a la formación integral de los estudiantes, propiciar el desarrollo armónico y la superación académica de los docentes e impulsar y proponer alternativas novedosas de acercamiento y extensión del conocimiento y la cultura; su misión es persuadir tanto a la comunidad interna como a la externa de participar en actividades extracurriculares llámese de formación, de educación continua o a distancia y en actividades deportivas que permitan engrandecer la superación personal o académica de los individuos en general y de la comunidad universitaria de la Facultad.

Consejo Técnico

El Consejo Técnico es la autoridad máxima de la Facultad y tiene numerosas funciones que competen a la planeación de la vida académica de la misma, a todo tipo de trámites y a la evaluación del personal académico y los alumnos.

Secretaría de Desarrollo Institucional y Estudios Profesionales

Esta Secretaria se encarga de apoyar a las distintas Coordinaciones de Carrera de esta Facultad en su compromiso por mejorar la enseñanza, otorgando condiciones y estableciendo espacios académicos en los que se desarrollen posibilidades reales para una efectiva elevación de la competitividad pedagógica.

Coordinación de Difusión Cultural

La coordinación de difusión Cultural se encarga de desarrollar las políticas y acciones para la puesta en práctica de un Programa Cultural, que repercuta en la formación integral de estudiantes, profesores y comunidad externa de la zona aledaña a la FESC, así como delinear propuestas que contribuyan a mejorar las condiciones en que dicha actividad se realiza, para el fomento y preservación de la cultura y el arte universales, así como la divulgación científica.

Secretaria de Posgrado e Investigación

En cuanto al posgrado su misión es preparar para la investigación, la docencia y la práctica profesional de alto nivel a hombres y mujeres que desarrollen capacidades de pensamiento complejo y crítico, para la creación y recreación del conocimiento; de esta manera, formar profesionales con claro sentido ético y compromiso con la realidad social, que contribuyan al desarrollo de la ciencia, la tecnología, las humanidades y las artes, y que coadyuven al reconocimiento y a la solución de los problemas nacionales.

Secretaría General

La Secretaría General tiene como misión coadyuvar, con la directora, en la vigilancia, el cumplimiento de la Legislación Universitaria, en los planes y programas de trabajo, y en las disposiciones generales, además de los acuerdos que normen la estructura de la Universidad y de la Facultad.

En la figura 1.1 podemos observar gráficamente por medio de un organigrama las diversas entidades anteriormente descritas de las FESC.

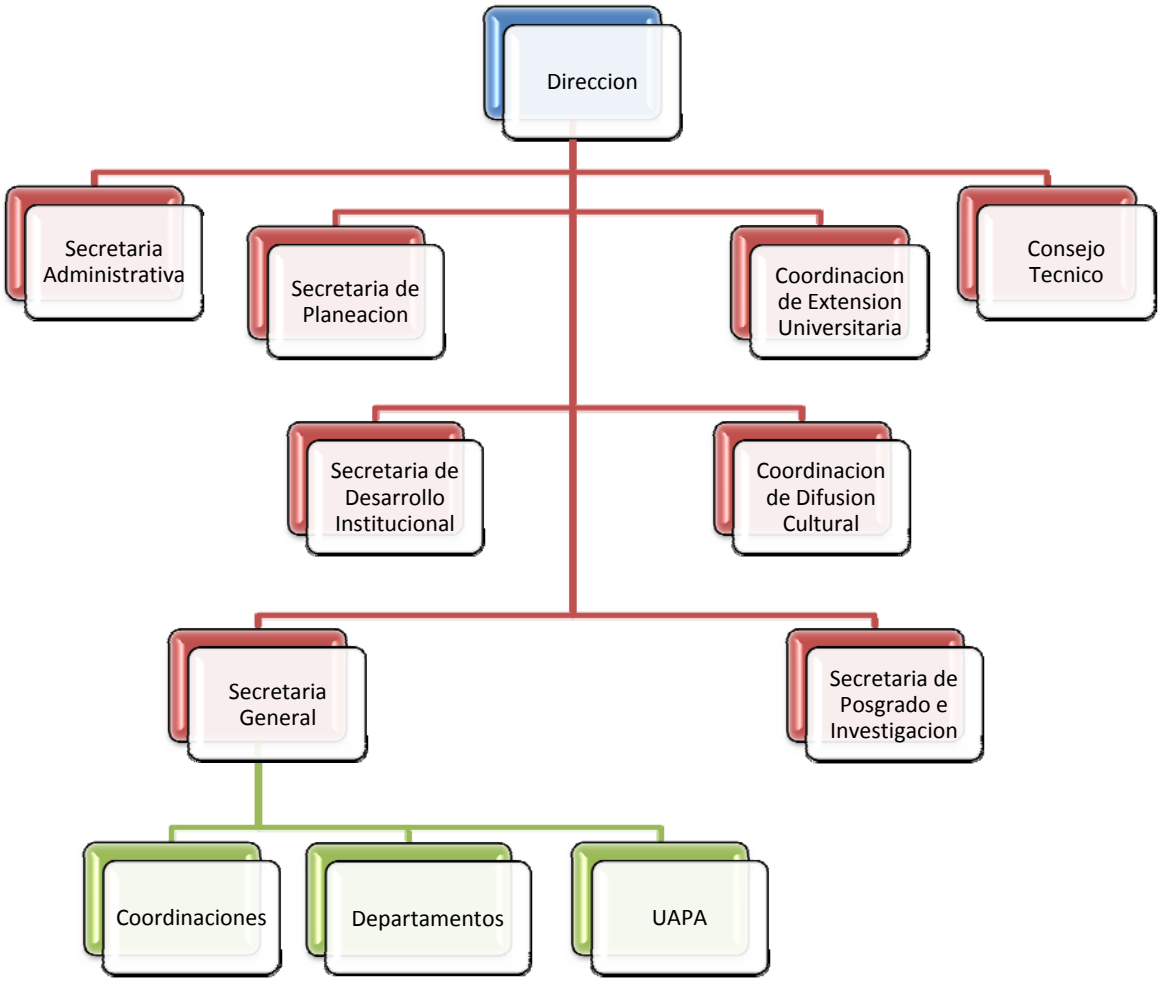


Figura 1.1 Organigrama de la FESC

1.2 Análisis de la problemática

La Facultad de Estudios Superiores Cuautitlán lleva a cabo año con año un proceso bien definido para la administración de su docencia, en el cual las diferentes áreas, previamente definidas interactúan entre sí para llevar a cabo dicho proceso, el cual se describe a continuación:

1) Las coordinaciones de carrera que dependen de la secretaría de Desarrollo Institucional crean los grupos correspondientes a cada materia según las necesidades y la demanda de la carrera

2) Una vez terminado el proceso de creación de grupos el Consejo Técnico tiene la tarea de evaluar las propuestas de grupos de cada coordinación para así aprobarlas o declinarlas

3) Los departamentos se encargan de asignar profesores a cada grupo creado por las coordinaciones, esta tarea es una de las más complicadas, debido a que cada profesor debe empatar perfectamente las horas asignadas a su nombramiento con las horas de los grupos que se le asignen, respetando en todo momento del proceso las reglas definidas en el Estatuto del Personal *Aca*démico[2] (EPA) para cada tipo de nombramiento

4) El departamento debe enviar a la Unidad de Asuntos del Personal Académico (UAPA) las asignaciones de grupos cargadas a cada profesor por nombramiento, para que la UAPA se encargue de evaluar que toda asignación cumpla con lo establecido con el EPA, como se mencionó anteriormente y de esta forma poder pasar a la siguiente fase del proceso

5) El Consejo técnico se da a la tarea de revisar por última vez las asignaciones de cada profesor, para aprobarlas o rechazarlas según su criterio

6) Finalmente una vez aprobadas las asignaciones por el Consejo Técnico el área de personal procede a elaborar los contratos de los profesores

Todo lo anterior está determinado por una serie de tiempos que se deben respetar para que el proceso tenga éxito; dicho proceso se describe en el siguiente diagrama para observar más a detalle el flujo que sigue.

[2] El estatuto del personal académico es reglamento que regirá las relaciones entre la universidad y su personal académico, en el se establecen los derechos y obligaciones del personal académico de la facultad.

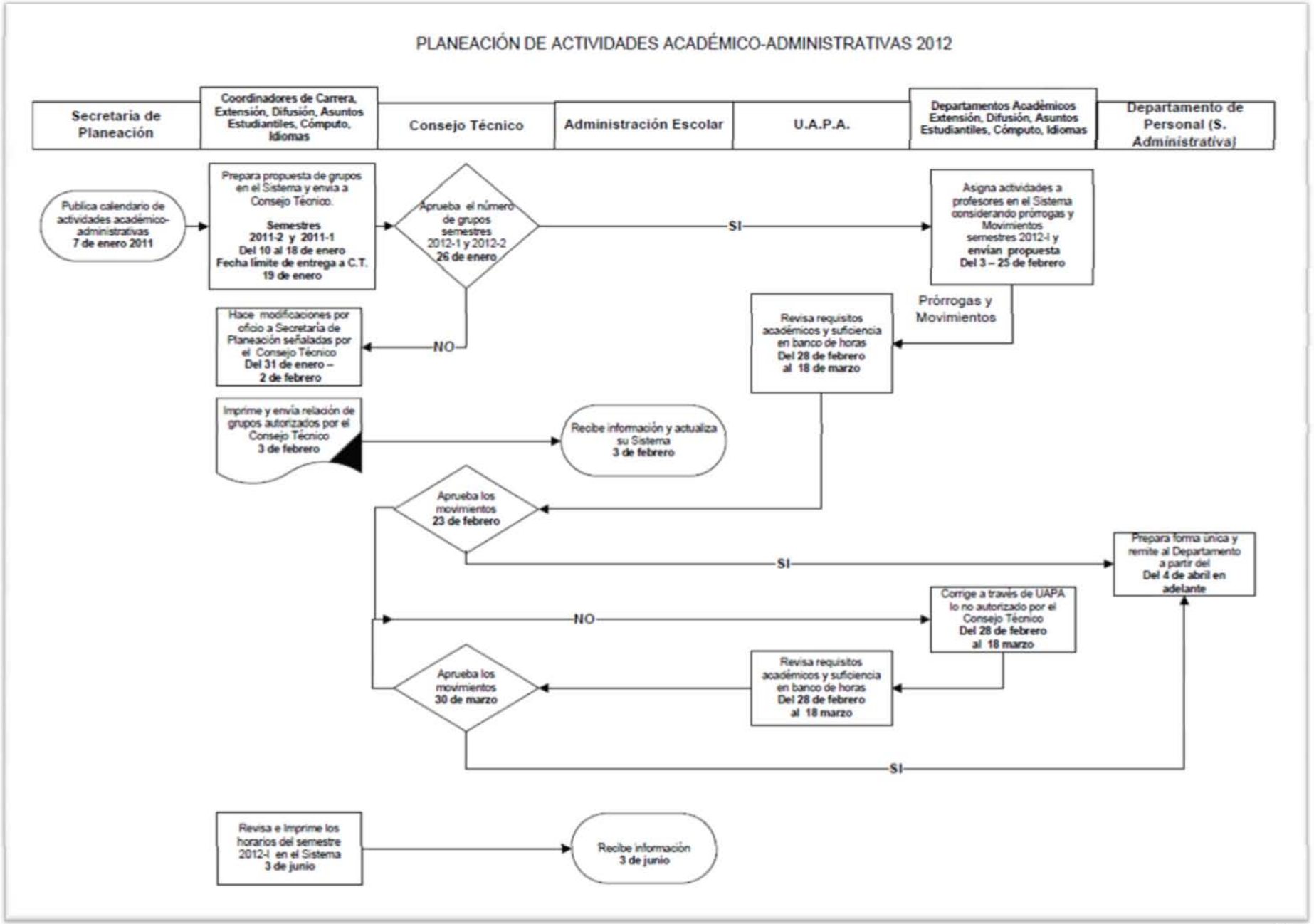


Figura 1.2 Diagrama de Actividades para la Asignación de la Docencia de la FESC

Actualmente la Facultad de Estudios Superiores Cuautitlán cuenta para el apoyo del proceso descrito con un sistema implantado en el año 2000, dicho sistema fue desarrollado con el lenguaje de programación Power Builder[3] y trabaja en una arquitectura cliente-servidor de dos capas en la cual la aplicación se ejecuta del lado del cliente, lo que significa que para que ser utilizada dicha aplicación es necesario instalarla en el equipo del usuario, lo que ocasiona una pérdida de tiempo para los usuarios y para los administradores ya que es necesario reinstalar la aplicación cada que un equipo falla o cada que se hace una actualización al sistema, además de que sus características operacionales se han visto rebasadas tecnológicamente por el paso del tiempo y este sistema no es capaz de evaluar la información que ingresan los usuarios, por lo tanto la probabilidad de error y de repetir el trabajo es muy alta; actualmente en la FESC se crean aproximadamente 5000 grupos por semestre y se cuenta con una plantilla de 1378 académicos para dar servicio a 13,121 alumnos de licenciatura y posgrado, por lo cual cada vez es más alta la carga de trabajo y la exigencia de los usuarios por un sistema que agilice, automatice y valide dicho proceso.

1.3 Propuesta de solución

Después de realizar un análisis en el cual se pudo observar que actualizar el sistema anterior era una tarea imposible debido a que se carece de documentación, su tecnología es antigua y no se tiene acceso al código fuente, se determino que la opción más viable para mejorar el proceso de asignación de la docencia de la FESC es la creación de un nuevo sistema que permita a la facultad tener un mejor control y una mayor administración del mismo; con esto se busco lograr la automatización de todo el proceso, evitando así el mayor numero de errores posibles por parte de los usuarios, logrando con ello una mayor eficiencia y optimización de recursos; la aplicación se desarrollo en el lenguaje de programación Java 1.6 con una base de datos relacional PostgreSQL 9.0 y en una arquitectura de n capas, lo cual permitirá su uso desde cualquier equipo con conexión a internet sin importar el sistema operativo del equipo, además de que las tecnologías utilizadas son software libre, por lo tanto no hay necesidad de pagar licencias por su uso.

[3] Lenguaje de programación desarrollado por PowerSoft en 1990

1.4 Objetivos

1.4.1 Objetivo general

- Crear un sistema automatizado implantado en un servidor web que permita a los usuarios acceder a el desde cualquier equipo con conexión a internet y que agilice el proceso de asignación de la docencia de la facultad, así como lograr tener el control de dicho proceso en todo momento.

1.4.2 Objetivos específicos

- Automatizar en 100% el proceso de asignación docencia de la facultad
- Facilitar el proceso de creación y asignación de la docencia a coordinaciones y departamentos
- Disminuir el margen de error humano en la creación de grupos y en la asignación de profesores
- Facilitar el trámite administrativo para el proceso de revisión de la asignación de la docencia
- Lograr una reducción de tiempo en la realización del proceso completo de asignación de la docencia

1.5 Metas

Implantar un nuevo sistema que permita mejorar la eficiencia a través de la gestión sistemática del proceso de asignación de la docencia, así como lograr una mejor administración y control del mismo, además de una reducción en los tiempos establecidos para dicho proceso, por lo cual es necesario ahondar acerca de los sistemas de información.

CAPITULO II. SISTEMAS DE INFORMACIÓN

Actualmente es vital para cualquier organización sistematizar sus procesos para lograr tener un mejor control de los mismos. Esto es lo que actualmente llamamos Gestión de procesos de negocios o BPM por sus siglas en ingles (Business Process Management [1] que nos permite mejorar la eficiencia a través de la administración sistemática de los procesos de negocio, que se deben modelar, automatizar, integrar, monitorizar y optimizar de forma continua para poder realizar la toma de decisiones de la organización basados en Business Intelligence que es el uso de la información que nos proporciona el BPM.

2.1 Sistema de Información

Existe una gran variedad de definiciones para lo que es un sistema de información y de la importancia que estos tienen dentro de una organización. Por ejemplo:

Para Mario G. Piattini[2] un sistema de información es:

> “Un conjunto formal de procesos, que operando sobre una colección de datos estructurada según las necesidades de la empresa, elaboran y distribuyen la información (o parte de ella) necesaria para las operaciones de dicha empresa y para las actividades de dirección y control correspondientes (decisiones), para desempeñar su actividad de acuerdo a su estrategia de negocio”

Para Kennet[3] un sistema de información es:

> “Componentes interrelacionados para reunir, procesar, almacenar y distribuir información para apoyar la toma de decisiones, la coordinación, al análisis y la visualización de una organización”

Por lo tanto, podemos decir que un sistema de información es un conjunto de tecnologías, procesos, usuarios y datos que interactúan entre sí para recopilar, organizar y procesar la información generada dentro de la organización con la finalidad de optimizar los procesos cotidianos que le permitan realizar sus operaciones básicas, así como tomar decisiones basadas en la información generada.

2.2 Ciclo de vida de software

El concepto "ciclo de vida del software" describe el desarrollo de sistemas de información, desde su fase de planteamiento del problema hasta su fase de liberación del sistema.

La ISO (International Organization for Standardization), en su norma 12207 define al ciclo de vida de un software como: un marco de referencia que contiene las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto de software, abarcando desde la definición hasta la finalización de su uso [4].

El objetivo del ciclo de vida de software es definir las diferentes fases que se requieren para validar el desarrollo del software para garantizar que cumpla los requisitos para la aplicación y verificación de los procedimientos de desarrollo.

De igual forma, el ciclo de vida de software permite que se detecten errores lo antes posible de la implementación. Esto permite a los desarrolladores concentrarse en la calidad del software.

Se puede considerarse que el ciclo de vida del software contempla cinco etapas claramente diferenciadas que describiremos a continuación.

- Inicio: Es el nacimiento de la idea. Se definen los objetivos del proyecto y los recursos a utilizar

- Planificación: se genera un planeamiento detallado que guíe la gestión del proyecto, el tiempo y su inversión económica

- Implementación: se acuerda el conjunto de actividades que componen la realización del producto

- Puesta en producción: es la presentación del producto al cliente final

- Control de producción: Es el control del producto. Se analiza cómo el proceso difiere o no de los requerimientos originales e iniciar las correcciones si es necesario

El orden y la presencia de cada uno de estos procedimientos en el ciclo de vida de un software dependen del tipo de modelo de ciclo de vida acordado entre el cliente y el equipo de desarrolladores.

2.3 Modelos de ciclo de vida

Existen muchos modelos sobre ingeniería de software que representan una descripción simplificada de un proceso de software. Estos modelos pueden incluir actividades como parte de los procesos y productos de software, así como el papel de las personas involucradas en la ingeniería del software.

El proceso de desarrollo de software no es único. No existe un proceso de software universal que sea efectivo para todos los contextos de proyectos de desarrollo. Debido a esta diversidad, es difícil automatizar todo un proceso de desarrollo de software.

Cada proyecto de software requiere de una forma particular de abordar el problema. Las propuestas comerciales y académicas actuales promueven procesos iterativos, donde en cada iteración puede utilizarse uno u otro modelo de proceso, considerando un conjunto de criterios (por ejemplo: grado de definición de requisitos, tamaño del proyecto, riesgos identificados, entre otros).

A pesar de la variedad de propuestas, existe un conjunto de actividades fundamentales que se encuentran presentes en todos los modelos:

1. Especificación de software: Se debe definir la funcionalidad y restricciones operacionales que debe cumplir el software. Esta fase es de suma importancia en el desarrollo de sistemas ya que es en la fase en la que se analiza el problema , para comprender cuál será el impacto del software sobre el negocio, que es lo que el cliente quiere y cómo interactuarán los usuarios finales con el software, para lo cual podemos tomar como base los cuatro elementos del proceso de ingeniería de requerimientos:

    a. Estudios de viabilidad. Se recopilan un conjunto de requerimientos de negocio preliminares, una descripción resumida del sistema y de cómo este pretende contribuir a los procesos del negocio

    b. Obtención y análisis de requerimientos. Se trabaja con los clientes y los usuarios finales del sistema para determinar el dominio de la aplicación, qué servicios debe de proporcionar el sistema, el rendimiento requerido del sistema, las restricciones hardware, software, etc. para lo cual podemos utilizar diversas técnicas para interactuar con los diferentes usuarios del sistema, como lo son las entrevistas, lluvia de ideas, cuestionarios, casos de uso y análisis jerárquicos

    c. Validación de requerimientos. Trata de demostrar que los requerimientos establecidos realmente definen el sistema que el

cliente desea; esto es de suma importancia debido a que los errores que pudieran existir en el documento de requerimientos pueden conducir a importantes costes al repetir el trabajo cuando dichos errores son descubiertos durante el desarrollo o después de que el sistema este en uso

d. Administración de requerimientos. Es el proceso de comprender y controlar los cambios en los requerimientos del sistema.

2. Diseño e Implementación: En esta fase de diseño se establece la forma en la que el sistema cumplirá con los requerimientos establecidos durante la fase de análisis. Aquí se definirá en detalle entidades y relaciones de las bases de datos, se pasara de casos de uso esenciales a su definición como casos expandidos reales, se seleccionara el lenguaje de programación más adecuado, el Sistema Gestor de Base de Datos a utilizar, configuraciones de hardware, redes, etc. Una vez concluida la fase de diseño se empiezan a codificar algoritmos y estructuras de datos, definidos en las etapas anteriores. Los especialistas en sistemas se refieren a esta etapa como diseño físico.

3. Validación: El software debe validarse, para asegurar que cumpla con lo que el cliente requiere.

4. Implantación y Evolución: Una vez que el software a superado las pruebas de fiabilidad, que es la operación libre de fallos de un programa de computadora en un entorno determinado y durante un periodo de tiempo especifico se procede a la instalación y capacitación de los usuarios del software, para poder así entregar el mismo al cliente. Una vez realizada la entrega, existen diversos factores por los que el software puede cambiar durante el paso del tiempo, por lo cual el software debe evolucionar, para adaptarse a las necesidades del cliente.

A continuación se muestran algunos modelos utilizados para el desarrollo de software:

Ciclo de vida lineal

Es el más sencillo de los modelos. Consiste en descomponer la actividad del proyecto de software en etapas separadas que son realizadas de manera lineal.

Ciclo de vida en cascada puro

Este modelo permite iteraciones, pero estas iteraciones son al término de todas las fases, permitiendo corregir errores que se hayan encontrado

durante las pruebas; sus etapas son: análisis de los requisitos del software, diseño, generación de código, pruebas y mantenimiento.

Ciclo de vida iterativo

Este modelo busca reducir el riesgo que surge entre las necesidades del usuario y el producto final por malos entendidos durante la etapa de solicitud de requerimientos. Es la interacción de varios ciclos de vida en cascada. Al final de cada interacción se le entrega al cliente una versión mejorada o con mayores funcionalidades del producto.

Ciclo de vida por prototipos

El objetivo de este modelo es lograr un producto intermedio, para conocer mediante el prototipo utilizado, cómo responderán las funcionalidades previstas para el producto final.

El ciclo de vida evolutivo

El desarrollo evolutivo se basa en la idea de desarrollar una implementación inicial, exponiéndola a los comentarios del usuario y refinándola a través de las diferentes versiones hasta que se desarrolla un sistema adecuado.

Ciclo de vida incremental

El modelo incremental combina elementos del modelo lineal secuencial con la filosofía interactiva de construcción de prototipos.

Se realiza construyendo por módulos que cumplen las diferentes funciones del sistema. Esto permite ir aumentando gradualmente las capacidades del software. Este modelo facilita la tarea de desarrollo permitiendo a cada miembro desarrollar un módulo particular que puede ser añadido a otros módulos de otros programadores.

2.3.1 Ciclo de vida en espiral

El modelo en espiral fue originalmente propuesto por Boehm en 1988. [5] Más que representar el proceso del software como una secuencia de actividades con retrospectiva de una actividad a otra, se representa como una espiral. Cada ciclo en la espiral representa una fase del proceso. Así, el ciclo mas interno podría referirse a la viabilidad del sistema, el siguiente ciclo a la definición de requerimientos, el siguiente ciclo al diseño del sistema, y así sucesivamente.

- Definición de requisitos. Para esta fase del proyecto se definen los objetivos específicos. Se identifican las restricciones del proceso y el producto, y se traza un plan detallado de gestión. Se identifican los riesgos del proyecto. Dependiendo de estos riesgos, se planean estrategias alternativas

- Evaluación y reducción de riesgos. Se lleva a cabo un análisis detallado para cada uno de los riesgos del proyecto identificados. Se definen los pasos para reducir dichos riesgos. Por ejemplo, si existe el riesgo de tener requerimientos inapropiados, se puede desarrollar un prototipo del sistema

- Desarrollo y validación. Después de la evaluación de riesgos, se elige un modelo para el desarrollo del sistema. Por ejemplo, el modelo en cascada puede ser el más apropiado para el desarrollo si el mayor riesgo identificado es la integración de los subsistemas

- Planificación. El proyecto se revisa y se toma la decisión de si se debe continuar con un ciclo posterior de la espiral. Si se decide continuar, se desarrollan los planes para la siguiente fase del proyecto

La ventaja más notoria de este modelo es que se puede comenzar el proyecto con alto grado de incertidumbre y existe bajo riesgo de retraso en caso detección de errores.

Algunas de las desventajas son el costo temporal, que suma cada vuelta del espiral, la dificultad para evaluar riesgos y la necesidad de la presencia continua con el cliente.

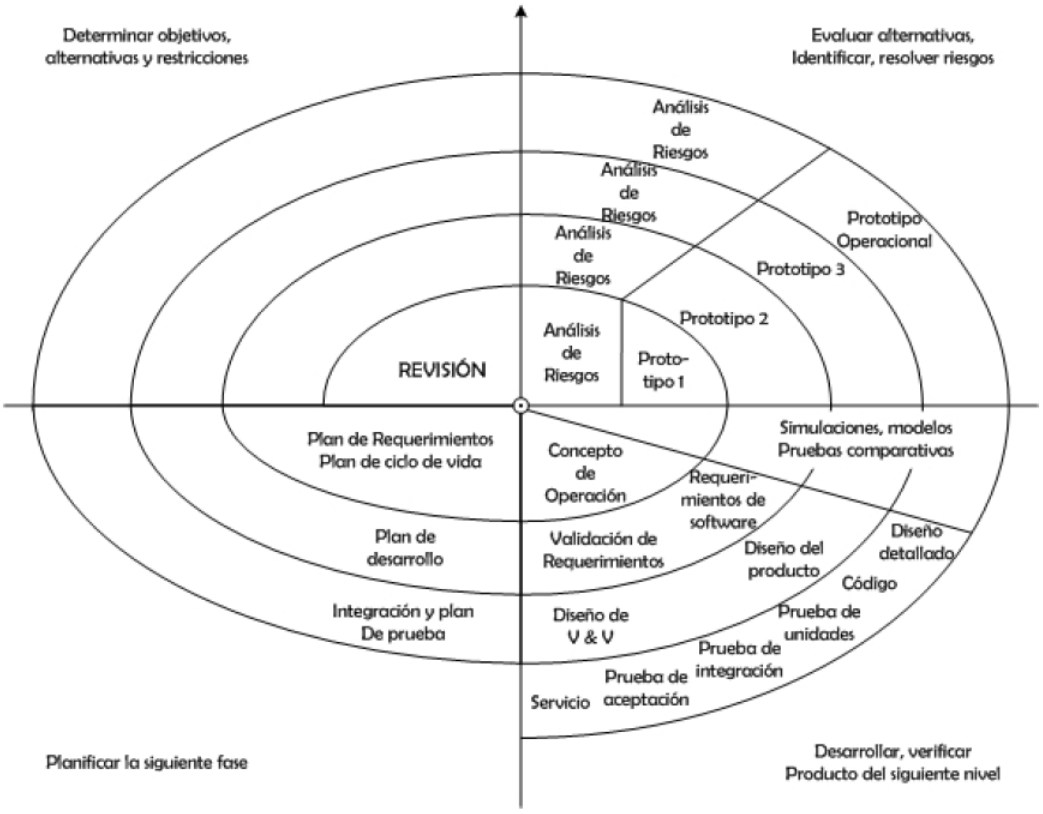


Figura 2.1 Diagrama del Modelo de ciclo de vida en espiral según Boehm

Una de las partes más importantes dentro de un sistema de información es el desarrollo y la implementación de las bases de datos, por lo cual en el siguiente capítulo vamos a profundizar en el tema de las bases de datos.

Capítulo III. BASES DE DATOS

3.1 Definición de Bases de datos y Sistema de Gestión de la Base de Datos (SGBD)

Definición de Base de datos.

"Colección de datos interrelacionados almacenados en conjunto sin redundancias perjudiciales o innecesarias; su finalidad es servir a una aplicación o más, de la mejor manera posible, los datos se almacenan de modo que resulten independientes de los programas que los usan; se emplean métodos bien determinados para incluir nuevos datos y para modificar y extraer los datos almacenados." (Martin, 1975) [6].

"Colección o deposito de datos integrados, almacenados en soporte secundario (no volátil) y con redundancia controlada. Los datos que han de ser compartidos con diferentes usuarios y aplicaciones, deben mantenerse independientes de ellos, y su definición (estructura de la base de datos) única y almacenada junto con los datos, se ha de apoyar en un modelo de datos, el cual ha de permitir captar las interrelaciones y restricciones existentes en el mundo real. Los procedimientos de actualización y recuperación, comunes y bien determinados, facilitaran la seguridad del conjunto de los datos. " (Piattini, 2007) [7].

El concepto de base de datos ha ido cambiando a lo largo del tiempo, pero podríamos decir que es un conjunto de entidades relacionadas entre sí, almacenadas en un medio informático no volátil y con una mínima redundancia; su estructura se basa en un modelo de datos, el cual les permite representar y almacenar junto con los datos las interrelaciones que existen en el mundo real; dichos datos han de servir a múltiples usuarios y a diferentes aplicaciones de las cuales son totalmente independientes.

Definición de Sistema de Gestión de la Base de Datos (SGBD).

"Se puede definir el Sistema de Gestión de la Base de Datos como un conjunto coordinado de programas, procedimientos, lenguajes, etc. que suministra a los distintos tipos de usuarios los medios necesarios para describir y manipular los datos almacenados en la base, garantizando su seguridad." (Piattini 2007)[7].

"Un Sistema Gestor de Bases de Datos consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a dichos datos, el objetivo principal de un SGBD es proporcionar una forma de almacenar y recuperar la información de una base de datos de manera que sea tanto practica como eficiente." (Shudarshan 2002)[8].

De lo anterior podemos entonces definir un SGBD como un conjunto de programas y procedimientos que le permiten a los usuarios almacenar, recuperar y actualizar los datos almacenados en una base de datos de forma eficaz y segura, por lo cual podemos decir que un SGBD funciona como una interfaz entre la base de datos, la aplicación y los usuarios.

3.2 Importancia de las Bases de Datos en el desarrollo de aplicaciones

Hoy en día, la información y el manejo de la misma es de vital importancia para cualquier organización, ya que de ella depende la toma de decisiones, punto clave para el éxito o fracaso de la organización; dicha información tiene un volumen cada vez mayor y es más compleja, lo cual ha rebasado las capacidades de una persona de gestionarla, por lo tanto es necesario apoyarnos de una herramienta que nos permita manipularla de forma eficiente y segura.

Para desarrollar este tipo de procesos dentro de la organización, es necesario el empleo de una base de datos, que permita la administración de grandes volúmenes de información de manera oportuna, además de que nos brindan las siguientes ventajas:

- Los datos son totalmente independientes de las aplicaciones, por lo tanto podemos obtener información desde diversas aplicaciones sin necesidad de modificar la estructura de la base de datos

- El acceso a la información por parte de los usuarios es trasparente, sin necesidad de que el usuario conozca aspectos técnicos del funcionamiento de la base de datos

- El acceso a la información es en tiempo real y en cuestión de segundos a pesar de trabajar con grandes cantidades de información

- Al no existir redundancia en los datos que se guardan, los datos solo se recogen y verifican una vez, por lo que el rendimiento de todo el proceso de validación y almacenamiento de los datos del sistema aumentan

- Garantiza que la información almacenada es confiable y que se encuentra segura debido a que tiene herramientas para el resguardo de la misma

- El espacio de almacenamiento de la información se reduce notablemente

- La documentación que se obtiene se encuentra más normalizada, debido a que, junto con los datos se integra su sentido semántico, lo cual veremos más adelante

Por lo anterior es que las bases de datos son utilizadas en todo tipo de organizaciones, desde pequeñas hasta trasnacionales, ya que les facilitan el acceso a la información y les garantizan que esta sea integral, confidencial y estará disponible en todo momento.

3.3 Modelos de Bases de Datos

Un modelo de bases de datos es un conjunto de ideas lógicas utilizadas para representar la estructura de datos y las relaciones entre ellos dentro de la base de datos.

Un modelo de datos es básicamente una "descripción" de algo conocido como *contenedor de datos* (algo en donde se guarda la información), así como de los métodos para almacenar y recuperar información de esos contenedores. Los modelos de datos no son cosas físicas: son abstracciones que permiten la implementación de un sistema eficiente de *base de datos*; por lo general se refieren a algoritmos, y conceptos matemáticos [8][9].

Además de la clasificación por la función de las bases de datos, éstas también se pueden clasificar de acuerdo a su modelo de administración de datos.

3.3.1 Modelo Jerárquico

Una base de datos jerárquica es un tipo de sistema gestor de base de datos que, como su nombre lo indica, almacena la información en una estructura jerárquica que enlaza los registros en forma de estructura de árbol (similar a un árbol visto al revés), en donde un nodo padre de información (raíz) puede tener varios nodos hijos, pero un nodo hijo solo puede tener un nodo padre, representando una relación de uno a muchos (1:N).

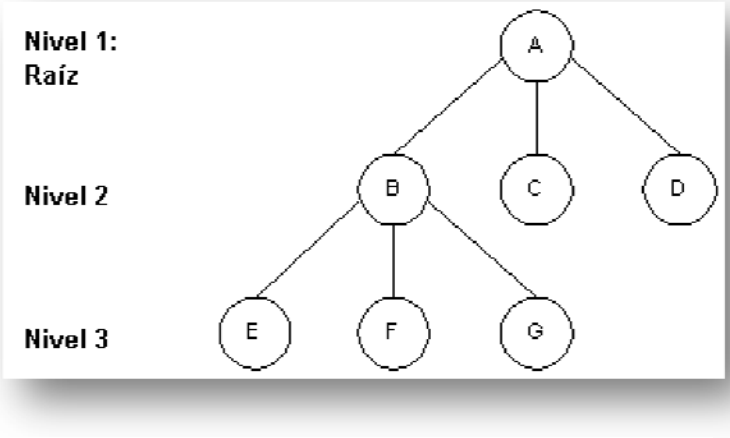


Figura 3.6 Diagrama del modelo jerárquico

El sistema de base de datos jerárquico más popular durante los años 70 y principios de los 80 fue el Sistema de administración de información (IMS, por sus siglas en ingles) resultado de los esfuerzos en conjunto de Rockwell-IBM.

3.3.2 Modelo de Red

Este modelo es resultado de la estandarización del Conference on Data System Languages (CODASYL) y fue creado para representar relaciones de datos complejas más eficientes de lo que el modelo jerárquico podía, para mejorar el desempeño de las bases de datos y para imponer un estándar de bases de datos, su característica distintiva es que el esquema es visto como un grafo en el que los tipos de objetos son los nodos y las relaciones son los arcos

Una estructura de datos en red, es muy similar a una estructura jerárquica, de hecho no es más que un súper conjunto de ésta.

En la terminología de las bases de datos en red, una relación se llama conjunto. Cada conjunto se compone de por lo menos dos tipos de registro: un registro propietario que equivale al padre en el modelo jerárquico y un registro miembro que equivale al hijo. Al igual que en la estructura jerárquica, cada nodo puede tener varios hijos pero, a diferencia de ésta, también puede tener varios padres.

Algunos de los sistemas manejadores de base de datos que soportan el modelo de red son: DBMS-10, Digital DBMS-20, Digital Equipment Corporation VAX DBMS, Honeywell IDS (Integrated Data Store), IDMS (Integrated Database Management System), RDM Embedded, RDM Server, TurboIMAGE, Univac DMS-1100.

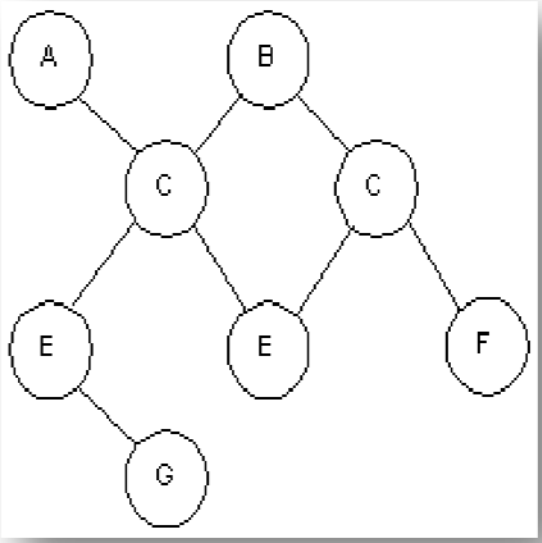


Figura 3.7 Diagrama del modelo de red

3.3.3 Modelo Relacional

El modelo relacional es un modelo de datos basado en la lógica de predicados y en la teoría de conjuntos. Es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente [10].

Su idea fundamental es el uso de relaciones. Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados *tuplas*, aunque la mayoría de las veces se conceptualiza de una manera más fácil de imaginar, esto es, pensando en cada relación como si fuese una tabla que está compuesta por *registros* (cada fila de la tabla sería un registro o *tupla*), y *columnas* (también llamadas *campos*).

Una base de datos relacional es un conjunto de una o más tablas estructuradas en registros (líneas) y campos (columnas), que se vinculan entre sí por un campo en común, en ambos casos posee las mismas características como por ejemplo el nombre de campo, tipo y longitud; a este campo generalmente se le denomina ID, identificador o llave. A esta manera de construir bases de datos se le denomina modelo relacional.

El modelo de datos relacional se ejecuta mediante un sistema de administración de base de datos relacional (Relational Data Base Management System RDBMS) muy complejo; la ventaja más importante del RDBMS es que permite que el usuario o le diseñador, operen en un ambiente lógico humano. El RDBMS maneja todos los detalles físicos complejos, de este modo, la base de datos relacional es percibida por el usuario como un conjunto de tablas en las que se guardan los datos; Cada tabla es una matriz compuesta por una serie de intersecciones de filas y columnas.

Las bases de datos relacionales pasan por un proceso al que se le conoce como normalización de una base de datos, el cual es entendido como el proceso necesario para que una base de datos sea utilizada de manera óptima [11].

Con la técnica de la normalización se trata de evitar la dependencia entre inserciones, actualizaciones y borrado de elementos de las tablas de la base de datos. También se reducen las operaciones de reorganización cuando hay que incorporar nuevos datos. La normalización tiene tres etapas que transforman las relaciones no normales en normalizadas y que se denominan *primera, segunda y tercera formas normales* [12].

Primera forma normal

El primer paso de la normalización es poner los datos en la primera forma normal. Esto se hace situando los datos en tablas separadas, de manera que los datos de cada tabla sean de un tipo similar, y dando a cada tabla una clave primaria y un identificador o etiqueta única. Esto elimina los grupos repetidos de datos.

Supongamos que inicialmente, tenemos toda la información a incluir en una base de datos en una sola tabla de nombre EMPLEADO cuyos campos pueden ser Nombre, Edad, Alojamiento, Responsable, Dirección, Oficio1, Ofici2 y Oficio3. (Figura 3.1)

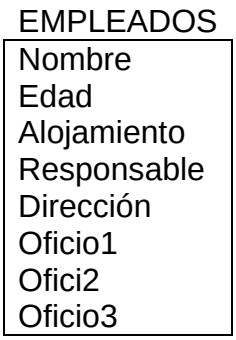


Figura 3.1

Los usuarios de la tabla tendrán problemas al almacenar los datos de los empleados de la compañía, puesto que al definir el oficio de los empleados solo dispone de tres posibilidades.

En lugar de fijar el numero de oficios de cada trabajador a tres como en la primera tabla, cada oficio del trabajador se localizara en su propia tabla separada (tabla OFICIO), con una fila por nombre, oficio y descripción del oficio. De esta forma se elimina un número variable de “oficios” en la tabla EMPLEADO.

Lo siguiente que se debe hacer es definir una clave primaria para cada tabla. Puesto que cada trabajador puede tener varias filas en la nueva tabla OFICIO, el nombre y el oficio serán la clave primaria completa de la tabla OFICIO. (Figura 3.2)

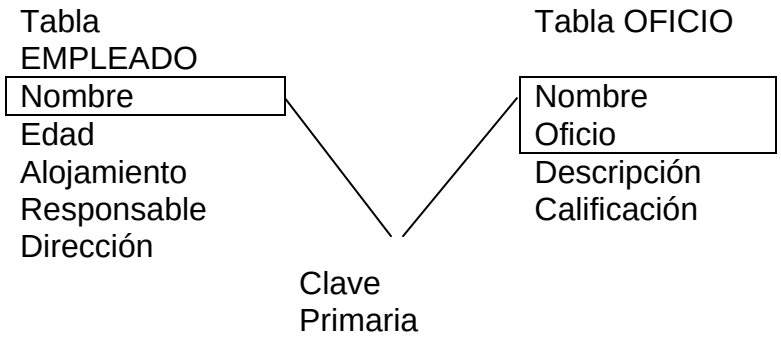


Figura 3.2

Segunda forma normal

La segunda forma normal, se centra en aislar los datos que solo dependen de una parte de la clave. Para obtener la segunda forma normal, se deben sacar Oficio y Descripción a una tercera tabla. La clave primaria te la tercera tabla es Oficio y su larga descripción aparece solo una vez.

Con la segunda forma normal, tanto "oficio" como su descripción permanecen en la base de datos, aunque en ese momento no haya ningún trabajador con esos oficios. Los "oficios" se pueden añadir incluso antes de que haya un trabajador con esos oficios. Por lo tanto normalizamos en segunda forma normal desdoblando la tabla OFICIO de la primera forma normal en otras dos tablas de nombre OFICIO-EMPLEADO (con los campos Oficio y Calificación) y OFICIO (con los campos Oficio y Descripción). Figura 3.3

Tabla EMPLEADO	Tabla OFICIO-EMPLEADO	Tabla OFICIO
Nombre	Nombre	Oficio
Edad	Oficio	Descripción
Alojamiento	Calificación	
Responsable		
Dirección		

Figura 3.3

Tercera forma normal

Es paso implica deshacerse de cualquier cosa de las tablas que no dependa únicamente de la clave primaria. La información de los trabajadores sobre el alojamiento depende de donde estén viviendo (si se trasladan habrá que actualizar la fila con el nuevo nombre del alojamiento en el que vive), pero el patrón de la posada y su dirección son independientes de si el trabajador vive ahí o no. Por eso la información del alojamiento se lleva a una tabla separada de nombre VIVIENDA y, por conveniencia, se usa una versión taquigráfica del nombre de la casa de alojamiento como clave primaria y el nombre total se guarda en el campo NombreCompleto.

Una vez normalizada la base de datos, presentamos el esquema de su diseño (Figura 3.4), en el que se observan las relaciones existentes.

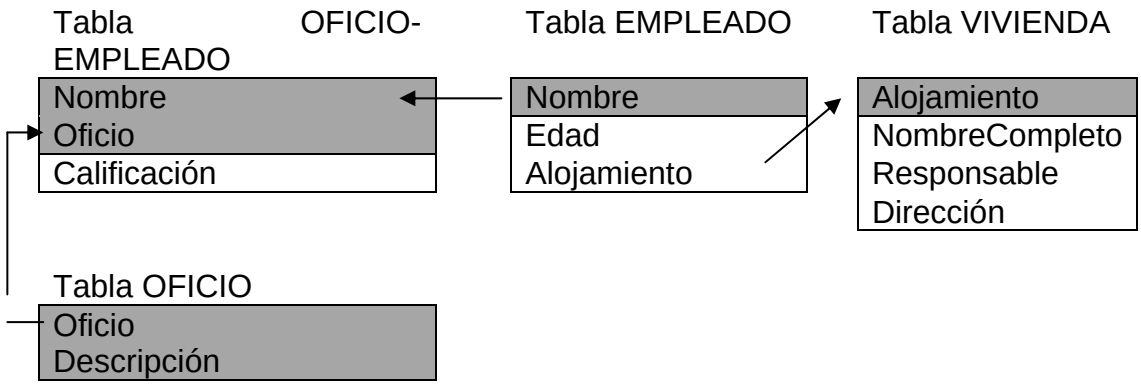


Figura 3.4

El proceso total de normalización se puede acabar de una forma menos tediosa usando directamente la tercera forma normal, que haciéndolo de forma a forma.

Entre las ventajas del modelo relacional están:

1. Garantiza herramientas para evitar la duplicidad de registros, a través de campos claves o llaves
2. Garantiza la integridad referencial: Así al eliminar un registro elimina todos los registros relacionados dependientes.
3. Favorece la normalización por ser más comprensible y aplicable.

Algunos de los gestores o manejadores relacionales actuales más populares encontramos: MySQL, PostgreSQL, Oracle, DB2, INFORMIX, Interbase, FireBird, Sybase y Microsoft SQL Server.

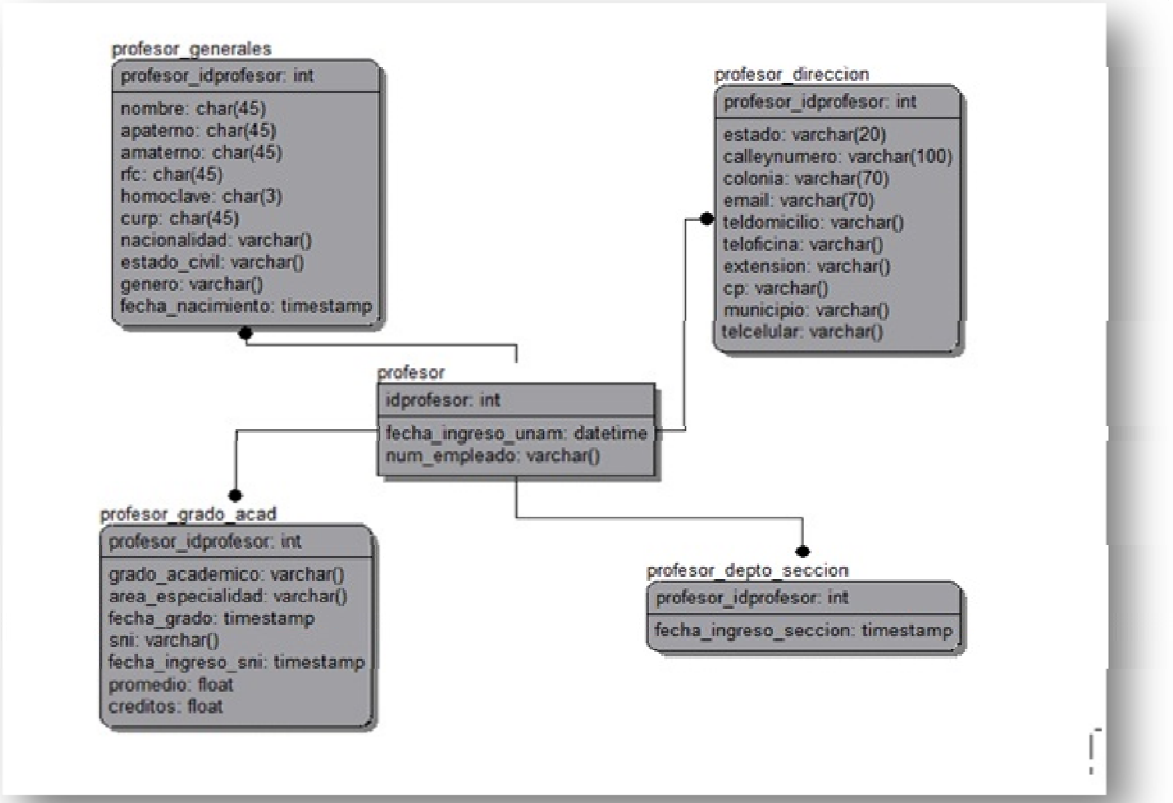


Figura 3.5 Diagrama del modelo relacional

3.3.4 Modelo Orientado a Objetos

Los cada vez más complejos problemas del mundo real demostraron que se requiere un modelo de datos diferente que represente con más fidelidad el mundo real. El primero de estos modelos fue el modelo de datos semántico[4] (SDM, por sus siglas en ingles), desarrollado por M. Hammer y d. McLeod y publicado en 1981 en su "Database Descrption with SDM: A Semántica Data Base Model".

El SDM modeló tanto datos como sus relaciones en una solo estructura conocida como objeto. Debido a que su estructura básica de modelo es un objeto, se dice

[4] La palabra semántica significa significado. Por consiguiente, en términos de base de datos orientada a objetos, se dice que el objeto tiene contenido semántico. Es por eso que Hammer y McLeod nombraron a su creación, modelo de base de datos semántico.

que el SDM es un modelo de datos orientado a objetos (OODM, por sus siglas en ingles). A su vez, el OODM se convierte en la base del modelo de base de datos orientado a objetos (OODBM, por sus siglas en ingles), el cual es manejado por un sistema de administración de base de datos orientado a objetos (OODBMS).

Los modelos de bases de datos orientados a objetos incorporan todos los conceptos importantes del paradigma de objetos:

- Encapsulación - Propiedad que permite ocultar la información al resto de los objetos, impidiendo así accesos incorrectos o conflictos
- Herencia - Propiedad a través de la cual los objetos heredan comportamiento dentro de una jerarquía de clases.
- Polimorfismo - Propiedad de una operación mediante la cual puede ser aplicada a distintos tipos de objetos.

En su estructura básica este modelo de base de datos está basado en por lo menos los siguientes componentes:

- Los objetos del modelo de datos son abstracciones de entidades o eventos del mundo real

- Los atributos describen las propiedades de un objeto, por ejemplo, un objeto PERSONA incluye los atributos incluye los atributos, nombre, número de seguro social y fecha de nacimiento

- Los objetos que comparten características similares se agrupan en clases. Una clase es un conjunto de objetos similares con estructura (atributos) y comportamientos (métodos) compartidos. En sentido general, una clase se parece al conjunto de entidades del modelo E-R; sin embargo, una clase es diferente de una entidad en que contiene una serie de procedimientos conocidos como métodos. Un método de clase representa una acción del mundo real tal como localizar el nombre de una PERSONA, cambiar el nombre de una PERSONA o imprimir el nombre de una PERSONA

- Las clases se organizan en una jerarquía de clases. La jerarquía de clase se parece a un árbol invertido donde cada clase tiene solo un padre. Por ejemplo la clase CLIENTE y la clase EMPLEADO comparten una clase PERSONA padre

- La herencia es la capacidad de un objeto, de la jerarquía de clase, de heredar los atributos los atributos y métodos de las clases que están sobre él

- El modelo de datos orientado a objetos representa un objeto como un cuadro vertical: todos los atributos y relaciones del objeto con otros objetos se incluyen dentro del cuadro del objeto

Algunos ejemplos de OODBMSs son Postgres, O2 creado por Ardent Software y el ObjectStore system creado por Object Design Inc.

Otro factor que es indispensable considerar en esta tesis son los sistemas distribuidos debido a que en la actualidad la mayoría de los sistemas utilizan esta tecnología, por lo cual en el siguiente capítulo se tratara el tema de sistemas distribuidos.

Capítulo IV. DESARROLLO DE SISTEMAS DISTRIBUIDOS

4.1 Definición de sistemas distribuidos

Se define como una colección de computadores autónomos conectados por una red, y con el software distribuido adecuado para que el sistema sea visto por los usuarios como una única entidad capaz de proporcionar facilidades de computación [Colouris 1994] [13].

Las aplicaciones de los sistemas distribuidos varían desde la provisión de capacidad de cómputo a grupos de usuarios, hasta sistemas bancarios, comunicaciones multimedia y abarcan prácticamente todas las aplicaciones comerciales y técnicas de los ordenadores. Los requisitos de dichas aplicaciones incluyen un alto nivel de fiabilidad, seguridad contra interferencias externas y privacidad de la información que el sistema mantiene. Se deben proveer accesos concurrentes a bases de datos por parte de muchos usuarios, garantizar tiempos de respuesta, proveer puntos de acceso al servicio que están distribuidos geográficamente, potencial para el crecimiento del sistema para acomodar la expansión del negocio y un marco para la integración de sistema usados por diferentes compañías y organizaciones de usuarios.

4.2 Características clave de los sistemas distribuidos

Son seis las características principales responsables de la utilidad de los sistemas distribuidos. Se trata de compartición de recursos, apertura (openness), concurrencia, escalabilidad, tolerancia a fallos y transparencia. En las siguientes líneas trataremos de abordar cada una de ellas.

1.- Compartición de Recursos

Los recursos en un sistema distribuido están físicamente encapsulados en una o varias de las computadoras y sólo pueden ser accedidos por otras computadoras mediante las comunicaciones (la red). Para que la compartición de recursos sea efectiva, ésta debe ser manejada por un programa que ofrezca un interfaz de comunicación permitiendo que el recurso sea accedido, manipulado y actualizado de una manera fiable y consistente. Surge el término genérico de gestor de recursos.

Un sistema distribuido puede verse de manera abstracta como un conjunto de gestores de recursos y un conjunto de programas que usan los recursos. Los usuarios de los recursos se comunican con los gestores de los recursos para acceder a los recursos compartidos del sistema. Esta perspectiva nos

lleva a dos modelos de sistemas distribuidos: el modelo cliente-servidor y el modelo basado en objetos.

2.- Apertura (opennesss)

Un sistema informático es abierto si el sistema puede ser extendido de diversas maneras. Un sistema puede ser abierto o cerrado con respecto a extensiones hardware (añadir periféricos, memoria o interfaces de comunicación, etc...) o con respecto a las extensiones software (añadir características al sistema operativo, protocolos de comunicación y servicios de compartición de recursos, etc...). La apertura de los sistemas distribuidos se determina primariamente por el grado hacia el que nuevos servicios de compartición de recursos se pueden añadir sin perjudicar ni duplicar a los ya existentes.

3.-Concurrencia

Cuando existen varios procesos en una única maquina decimos que se están ejecutando concurrentemente. Si el ordenador esta equipado con un único procesador central, la concurrencia tiene lugar entrelazando la ejecución de los distintos procesos. Si la computadora tiene N procesadores, entonces se pueden estar ejecutando estrictamente a la vez hasta N procesos.

4.-Escalabilidad

Los sistemas distribuidos operan de manera efectiva y eficiente a muchas escalas diferentes. La escala más pequeña consiste en dos estaciones de trabajo y un servidor de ficheros, mientras que un sistema distribuido construido alrededor de una red de área local simple podría contener varios cientos de estaciones de trabajo, varios servidores de ficheros, servidores de impresión y otros servidores de propósito especifico. A menudo se conectan varias redes de área local para formar internetworks, y éstas podrían contener muchos miles de ordenadores que forman un único sistema distribuido, permitiendo que los recursos sean compartidos entre todos ellos.

Tanto el software de sistema como el de aplicación no deberían cambiar cuando la escala del sistema se incrementa. La necesidad de escalabilidad no es solo un problema de prestaciones de red o de hardware, sino que esta íntimamente ligada con todos los aspectos del diseño de los sistemas distribuidos.

La demanda de escalabilidad en los sistemas distribuidos ha conducido a una filosofía de diseño en que cualquier recurso simple -hardware o software- puede extenderse para proporcionar servicio a tantos usuarios como se quiera

Cuando el tamaño y complejidad de las redes de ordenadores crece, es un objetivo primordial diseñar software de sistema distribuido que seguirá siendo eficiente y útil con esas nuevas configuraciones de la red. Resumiendo, el trabajo necesario para procesar una petición simple para acceder a un recurso compartido debería ser prácticamente independiente del tamaño de la red.

5.-Tolerancia a Fallos

Los sistemas informáticos a veces fallan. Cuando se producen fallos en el software o en el hardware, los programas podrían producir resultados incorrectos o podrían pararse antes de terminar la computación que estaban realizando. El diseño de sistemas tolerantes a fallos se basa en dos cuestiones, complementarias entre sí: Redundancia hardware (uso de componentes redundantes) y recuperación del software (diseño de programas que sean capaces de recuperarse de los fallos).

6.-Transparencia

La transparencia se define como la ocultación al usuario y al programador de aplicaciones de la separación de los componentes de un sistema distribuido, de manera que el sistema se percibe como un todo, en vez de una colección de componentes independientes. La transparencia ejerce una gran influencia en el diseño del software de sistema.

4.3 Modelo cliente-servidor

La arquitectura cliente-servidor consiste básicamente en un cliente que realiza peticiones a otro programa (el servidor) que le da respuesta. Aunque esta idea se puede aplicar a programas que se ejecutan sobre una sola computadora es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras.

En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa. Los tipos específicos de servidores incluyen los servidores web, los servidores de archivo, los servidores del correo, etc. Mientras que sus propósitos varían de unos servicios a otros, la arquitectura básica seguirá siendo la misma.

Una disposición muy común son los sistemas multicapa en los que el servidor se descompone en diferentes programas que pueden ser ejecutados por diferentes computadoras aumentando así el grado de distribución del sistema.

La arquitectura cliente-servidor sustituye a la arquitectura monolítica en la que no hay distribución, tanto a nivel físico como a nivel lógico.

La red cliente-servidor es aquella red de comunicaciones en la que todos los clientes están conectados a un servidor, en el que se centralizan los diversos recursos y aplicaciones con que se cuenta; y que los pone a disposición de los clientes cada vez que estos son solicitados. Esto significa que todas las gestiones que se realizan se concentran en el servidor, de manera que en él se disponen los requerimientos provenientes de los clientes que tienen prioridad, los archivos que son de uso público y los que son de uso restringido, los archivos que son de sólo lectura y los que, por el contrario, pueden ser modificados, etc [14][15].

4.3.1 Arquitectura de dos capas

La arquitectura cliente/servidor tradicional es una solución de dos capas. La arquitectura de dos capas consta de tres componentes distribuidos en dos capas: cliente (solicitud de servicios) y servidor (proveedor de servicios). Los tres componentes son:

- Interfaz de usuario
- Gestión del procesamiento
- Gestión de la base de datos

Hay dos tipos de arquitecturas cliente/servidor de dos capas: clientes pesados y clientes delgados [14][16].

Cliente pesado (Servidor delgado)

En este esquema de arquitectura el peso de la aplicación es ejecutada en el cliente, es decir, el nivel de presentación y el nivel de aplicación corren en un único proceso cliente, y el servidor es relegado a realizar las funciones que provee un administrador de base de datos. En general este tipo de arquitectura tiene mejor aplicación en sistemas de apoyo de decisiones (DSS: Decision Support System) y sistemas de información ejecutiva (EIS: Executive Information System

Servidor pesado (cliente delgado)

Este es el caso opuesto al anterior, el proceso cliente es restringido a la presentación de la interfaz de usuario, mientras que el peso de la aplicación corre por el lado del servidor de la aplicación. En general este tipo de arquitectura presenta una flexibilidad mayor para desarrollar una gran variedad de aplicaciones, incluyendo los sistemas de misión crítica a través de servidores de transacciones.

4.3.2 Arquitectura multi capas

En ingeniería de software, arquitectura multi capas (comúnmente llamada arquitectura de n capas) es una arquitectura cliente servidor en donde la presentación, la lógica de negocio y la gestión de los datos son procesos lógicamente separados. Por ejemplo una aplicación que utiliza middelware para satisfacer solicitudes de servicios de datos entre un usuario y la base de datos emplea la arquitectura de varios niveles. El uso más común de la arquitectura de n capas es la arquitectura de tres capas

Las aplicaciones de arquitectura de n capas proveen un modelo a los desarrolladores para crear aplicaciones flexibles y reutilizables.

Al separar una aplicación en capas, los desarrolladores sólo tienen que modificar o agregar una capa específica, en lugar de tener que reescribir toda la aplicación [14][17].

4.3.2.1 Arquitectura de tres capas

Arquitectura cliente servidor de tres capas es una arquitectura cliente servidor en la cual la interface de usuario, la lógica del negocio, el almacenamiento de los

datos y el acceso a los mismos son desarrollados como módulos independientes, comúnmente en plataformas separadas.

Aparte de las ventajas habituales de software modular con interfaces bien definidas, la arquitectura de tres capas tiene por objeto permitir remplazar o realizar actualizaciones en cualquiera de sus capas sin afectar toda la tecnología. Por ejemplo, un cambio de sistema operativo en la capa de presentación solo afectaría el código de la interfaz de usuario.

Usualmente, la interfaz de usuario corre en una computadora de escritorio o estación de trabajo y trabaja con una interfaz grafica de usuario, la capa de negocio puede consistir en uno o más módulos separados corriendo en un servidor de aplicaciones, y un RDBMS sobre un servidor de base de datos para almacenar los datos. La capa de en medio puede ser multicapa por si sola (en este caso toda la arquitectura es llamada arquitectura de n capas) [15][18].

Capítulo V. ANÁLISIS, DISEÑO E IMPLEMENTACION DEL SISTEMA DE ASIGNACION DE LA DOCENCIA DE LA FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN-MODULO DE CONSEJO TÉCNICO

5.1 Análisis y Requerimiento básicos del usuario

Para la fase de análisis se realizaron una serie de entrevistas con los distintos actores del sistema, mediante las cuales se logró recabar información acerca del proceso que desarrolla cada uno de ellos, además de los requerimientos para el nuevo sistema; dichos requerimientos fueron plasmados en minutas para mantener un control de los mismos; esta fase de análisis duro aproximadamente dos meses y una vez que se termino la fase se realizaron reuniones periódicas con los usuarios para aclaración de dudas y retroalimentación.

En esta tesis se analiza de forma particular el modulo de consejo técnico, el cual se describe a continuación:

Dentro de la facultad cada profesor que labora en ella es contratado bajo un nombramiento; dichos nombramientos tienen una serie de reglas que se establecen en el Estatuto del Personal Académico (EPA) [19], las cuales indican la cantidad de horas que el académico prestara sus servicios, le remuneración que recibirá, las horas máximas y mínimas de docencia permitidas según el nombramiento y la vigencia del nombramiento además de una serie de derechos y obligaciones.

El personal académico de la facultad está integrado por:

- Técnicos Académicos
- Ayudantes de profesor o de investigador
- Profesores
- Investigadores

Estos son los distintos nombramientos que existen en la universidad y de ellos derivan una serie de categorías[5] que permiten a la facultad tener una amplia gama de nombramientos para realizar la contratación de sus académicos. Estos nombramientos pueden ser interinos o definitivos, de los cuales los interinos son por un periodo de tiempo definido y los definitivos no tienen una fecha de término específica.

El consejo técnico es la entidad encargada de aprobar los nombramientos del personal académico de la facultad; cada nombramiento tiene una serie de reglas

[5] Ver apéndice A

establecidas en el Estatuto del Personal Académico, las cuales el consejo técnico debe de evaluar en cada nombramiento para asegurar que la normatividad de la institución sea respetada en todo momento; el consejo técnico realiza sesiones mensuales en las cuales se encargan de revisar dichos nombramientos.

Una vez que el consejo técnico aprueba los nombramientos de los académicos, estos nombramientos son enviados mediante un reporte al área de personal, ya que esta entidad se encarga de generar los contratos de los académicos cuyos nombramientos fueron aprobados. En la Figura 1.2 podemos observar la participación del consejo técnico en el proceso completo.

Requerimientos básicos del usuario CT:

- El sistema debería de contar con un modulo especifico para el consejo técnico

- El usuario de consejo técnico debería poder visualizar todos los nombramientos revisados por la Unidad de Asuntos del Personal Académico que hayan cumplido con la normatividad establecida en el Estatuto del Personal Académico

- Una vez que visualiza dichos movimientos el usuario de consejo técnico debe poder aprobar o rechazar los nombramientos de manera individual según lo establecido en la sesión de consejo técnico previa

- El usuario de consejo técnico podrá agregar datos extras como la reunión y la sesión en la que fue aprobado a cada nombramiento aprobado con fines de control de las aprobaciones del consejo técnico

- El usuario de consejo técnico podrá generar reportes con los nombramientos aprobados dependiendo si son prorrogas o movimientos, para ser enviados al Área de personal para que el proceso siga su curso

Con base en los requerimientos básicos del usuario se genero el modulo de consejo técnico acorde a sus especificaciones y necesidades.

Para la documentación de los requerimientos se utilizo el lenguaje UML[20] (Lenguaje Unificado de Construcción de Modelos) el cual es una notación utilizada para construir sistemas por medio de conceptos orientados a objetos.

5. 2 Características del sistema

Una vez establecidos los requisitos del usuario y las características principales del sistema se delimita el hardware y el software que se utilizara para la construcción del mismo.

5.2.1 Hardware

Para implantar el sistema se requirió de un servidor Dell PowerEdge R510 con 15gb de memoria RAM, 8 procesadores Intel(R) Xeon® a una velocidad de 2.67GHz y disco duro e 1tb.

5.2.2 Software

Linux: es un núcleo de sistema operativo libre tipo Unix. Es uno de los principales ejemplo de software libre. Linux esta licenciado bajo la GPL v2 (General Public License) y está desarrollado por colaboradores de todo el mundo. El núcleo Linux fue concebido por el entonces estudiante de ciencias de la computación finlandés, Linus Torvalds, en 1991, normalmente Linux se utiliza junto a un empaquetado de software, llamado distribución Linux [21].

GlassFish Server 3.0.1: es un servidor de aplicaciones de software libre desarrollado por Sun Microsystems, compañía adquirida por Oracle Corporation, que implementa las tecnologías definidas en la plataforma Java EE y permite ejecutar aplicaciones que siguen esta especificación. La versión comercial es denominada Oracle GlassFish Enterprise Server (antes Sun GlassFish Enterprise Server). Es gratuito y de código libre, se distribuye bajo un licenciamiento dual a través de la licencia CDDL y la GNU GPL [22][23].

PostgreSQL 9.0: es un potente sistema manejador de base de datos objeto-relacional de código libre; tiene más de 15 años de desarrollo activo y una arquitectura probada que se ha ganado una sólida reputación de fiabilidad e integridad de datos; corre sobre la mayoría de los sistemas operativos, incluido Linux, Unix UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), and Windows [24].

Java: es un lenguaje de programación y plataforma informática lanzado por Sun Microsystems en 1995. Java se ejecuta en más de 850 millones de computadoras personales en todo el mundo, y en miles de dispositivos en todo el mundo, incluyendo los dispositivos móviles y televisión [25].

Del lado del cliente solo es necesario el navegador web Mozilla Firefox y tener activado Javascript en el mismo, Mozilla Firefox puede ser descargado de forma gratuita e instalarse en cualquier equipo, sin importar el sistema operativo que tenga instalado el cliente, lo cual es una ventaja ya que el cliente puede trabajar sobre el sistema operativo de su preferencia y no tener ningún problema al trabajar con el sistema.

Cada uno de los componentes utilizados para el sistema, poseen una licencia que permite el libre uso sin ningún costo, así como un acceso gratuito a la documentación y actualizaciones de los mismos, lo cual fue una de las razones por los que se seleccionaron, además de que Java y PostgreSQL son de las aplicaciones más utilizadas actualmente para el desarrollo de sistemas, razón que añade confiabilidad y robustez de la plataforma desarrollada [26].

5.2.3 Struts

Struts es el framework más popular para desarrollar aplicaciones web basadas en java. Struts ha sido desarrollado como un proyecto de software libre iniciado por Apache Software Foundation y está basado en el Model View Controler (MVC). El modelo MVC se basa en la separación de responsabilidades ya que cada una de las partes se encarga de un aspecto específico y no interfiere con el de las otras y consiste en dividir una aplicación en tres componentes: Modelo, Vista y Controlador donde:

- El modelo representa a la lógica de negocio (manipulación de datos)
- La vista representa la presentación de los datos (diseño de páginas).
- El controlador representa el código de navegación de la aplicación (control de flujo).

Un framework es la extensión de un lenguaje, a través de una o más jerarquías de clases que implementan una funcionalidad y que pueden ser extendidas de manera opcional.

Struts provee la infraestructura básica para la implementación del patrón MVC, permitiendo así que los desarrolladores puedan concentrarse en la lógica de negocios, el framework está compuesto por aproximadamente 300 clases e interfaces que están organizadas en alrededor de 12 paquetes de nivel superior.

Básicamente, funciona de siguiente manera: el navegador genera una solicitud que es atendida por el Controlador (un Servlet especializado), el Controlador también se encarga de analizar la solicitud, seguir la configuración que se le ha programado en su XML y llamar al Action correspondiente pasándole los parámetros enviados; el Action instanciará y/o utilizará los objetos de negocio para concretar la tarea y según el resultado que retorne el Action, el Controlador

derivará la generación de interfaz a una o más Java Server Pages JSPs, las cuales podrán consultar los objetos del Modelo para mostrar información de los mismos[27][28][29].

5.2.3.1 Struts: Controlador

La clase ActionServlet es el corazón del framework ya que es el controlador y se configura como servlet en el archivo web.xml.

```
<!-- Standard Action Servlet Configuration (with debugging) -->
<servlet>
    <servlet-name>action</servlet-name>
    <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
    <init-param>
      <param-name>config</param-name>
      <param-value>/WEB-INF/struts-config.xml</param-value>
    </init-param>
```

Figura 5.1 ActionServlet

Cada vez que un usuario hace una petición es manejada por Struts Action Servlet; cuando el ActionServlet recibe la petición intercepta la URL y basado en el archivo de configuración struts-config.xml, dirige la petición al Action Class correspondiente.

Todas las peticiones atendidas por el framework siguen un patrón en su nomenclatura, por defecto *.do, pero éste se puede cambiar en el archivo web.xml

```
<!-- Standard Action Servlet Mapping -->
<servlet-mapping>
    <servlet-name>action</servlet-name>
    <url-pattern>*.do</url-pattern>
</servlet-mapping>
```

Figura 5.2 Archivo web.xml

Una aplicación Struts requiere de archivos de configuración en los cuales se registran y configuran los distintos objetos Struts que van a ser utilizados por la aplicación; dichos archivos se describen a continuación:

Struts-config.xml

Es el archivo de configuración de struts, aquí se mapean las acciones (peticiones).

La sección <form-beans> contiene las definiciones de beans y se usa un elemento <form-bean> por cada bean de formulario y tiene los siguientes atributos importantes:

- Name: Identificador único para referenciarlo en los mapeos de acciones.
- Type: ruta completa de la clase Java del bean de formulario.

La sección <action-mappings> contiene las definiciones de acciones, se usa un elemento <action> por cada acción, se definen los siguientes atributos:

- Path: nombre de la clase action en relación al contexto de la aplicación.
- Type: ruta completa de la clase Java de la clase Action
- Name: El nombre del <form-bean> para usar con esta action, si corresponde.
- Forward: ruta del archivo JSP a llamar

Dentro de cada <action> se pueden definir más de un forward, teniendo las siguientes propiedades:

- Name: nombre del forward que será utilizado al llamarlo desde la clase Action.
- Path: ruta de la página JSP

Struts Action Class

Una clase Action de una aplicación Struts extiende a la clase org.apache.struts.action.Action. Una clase Action actúa como un envoltorio para la lógica de negocio y provee una interfaz al modelo y es el mediador entre la vista y el Modelo, ya que transmite los datos desde la Vista hacia el proceso específico del modelo y retorna los resultados en sentido contrario.

Struts ActionForm

Una de las tareas que consume mucho trabajo es la interacción con formularios, ya sea para editar u obtener nueva información; las comprobaciones, la gestión de errores, el volver a presentar el mismo form al usuario con los valores que puso y los mensajes de error están soportados por Struts con los Action Forms y JSP.

Un ActionForm es un JavaBean que se extiende org.apache.struts.action.ActionForm que Implementa los métodos get y set para cada input de un formulario de una página, y los métodos validate y reset. Se llena

automáticamente en el lado del servidor con los datos introducidos desde un formulario en el lado del cliente

5.2.3.2 Struts: Modelo

Comprende toda la lógica del negocio, son las clases Java que el desarrollador tiene que implementar. No existe un formato definido para la creación de estos componentes, el framework proporciona interfaces a bases de datos, etc. Se elige el modelo de acuerdo con los requerimientos del cliente.

5.2.3.3 Struts: Vista

Los componentes de la Vista son JSP y se encargan de la presentación de la información al usuario y del ingreso de sus datos.

Muestran los datos proporcionados por los componentes del modelo.

Struts provee soporte para construir fácilmente la interfaz de usuario a través de la utilización de HTML Tag Library.

Pudiendo obtener aplicaciones multi-idioma, interacción con formularios y otras utilidades.

Para usar los Struts HTML Tags es necesario incluir la siguiente línea en el JSP:

- <%@ taglib uri="http://jakarta.apache.org/struts/tags-html" prefix="html" %>

Tag Library

- <html:html> Genera el tag <html>.
- <html:form> Genera <form>
- <html:button> Genera el tag <input type="button">.
- <html:submit> Genera <input type="submit"> para enviar los datos ingresados.
- <html:reset> Genera <input type="reset">
- <html:checkbox> Genera el tag <input type="checkbox">
- <html:hidden> Genera el tag <input type="hidden">.
- <html:img> Genera el tag <img>
- <html:link> Genera un link html
- <html:password /> Genera el tag <input type=“password”>.
- <html:radio> Genera un radio button (<input type="radio">).

- <html:select multiple="true" property="selectBox"> Genera la lista en el formulario, selectBox debe ser un array. Usar <html:options> para especificar las entradas.
- <html:text> Genera el tag <input type="text">
- <html:textarea> Genera el tag <textarea>.
- <html:file /> Genera el tag <input type="file"> para subir archivos.
- <html:base> Genera el tag <base>. Este tag debe ser usado dentro del tag <head>
- <html:errors> Genera el codigo html para mostrar los errors que pueden ocurrir en la página.
- <html:message key="thekey"/> Busca el valor correspondiente al key dado en el archivos de recursos y lo muestra.

5.2.4 Niveles de Seguridad

El sistema SAFESC maneja diversos niveles de seguridad para proteger la integridad del sistema y de los datos almacenados en su base de datos, entre los que se encuentran:

1) Sistema operativo: se refiere a software instalado en el sistema operativo (antivirus, firewall, contraseñas de inicio de sesión de usuarios, etc.) y que le permite garantizar la seguridad del equipo de forma remota o local.

2) De usuario: se refiere a las medidas que el sistema emplea para garantizar la integridad de los datos y el acceso a usuarios no autorizados.

a) Para lograr acceder al SAFESC es necesario realizar un inicio de sesión, para lo cual el usuario introduce su nombre de usuario y su contraseña.

b) Existen siete tipos de usuarios en SAFESC y cada uno de ellos tiene limitadas las acciones que puede realizar según el tipo de usuario que sea y los tiempos establecidos por el administrador del sistema para realizar modificaciones, además de que solo tienen acceso al modulo para el cual están dados de alta. Los módulos del sistema son:

- **Administrador**
- **Coordinador**
- **Jefe de departamento**
- **Jefe de sección**
- **UAPA**

- **Consejo Técnico**
- **Personal**

c) La clave del usuario y la contraseña se almacenan encriptadas mediante el algoritmo de encriptación SHA1, lo cual permite mantener un alto nivel de seguridad en las cuentas de usuario, ya que aunque la información almacenada en la base pudiera ser accedida por personal no autorizado no se revelaría el usuario y la contraseña reales de los usuarios, evitando así el acceso al sistema.

5.3 Diseño de la base de datos

Con el fin de almacenar la información generada por el sistema de una forma segura y protegiendo la integridad de los datos se implemento una base de datos relacional y se utilizo el sistema manejador de base de datos Postgres para implementarla. A continuación se muestra su diagrama y su diccionario de datos.

5.3.1 Diagrama de la base de datos

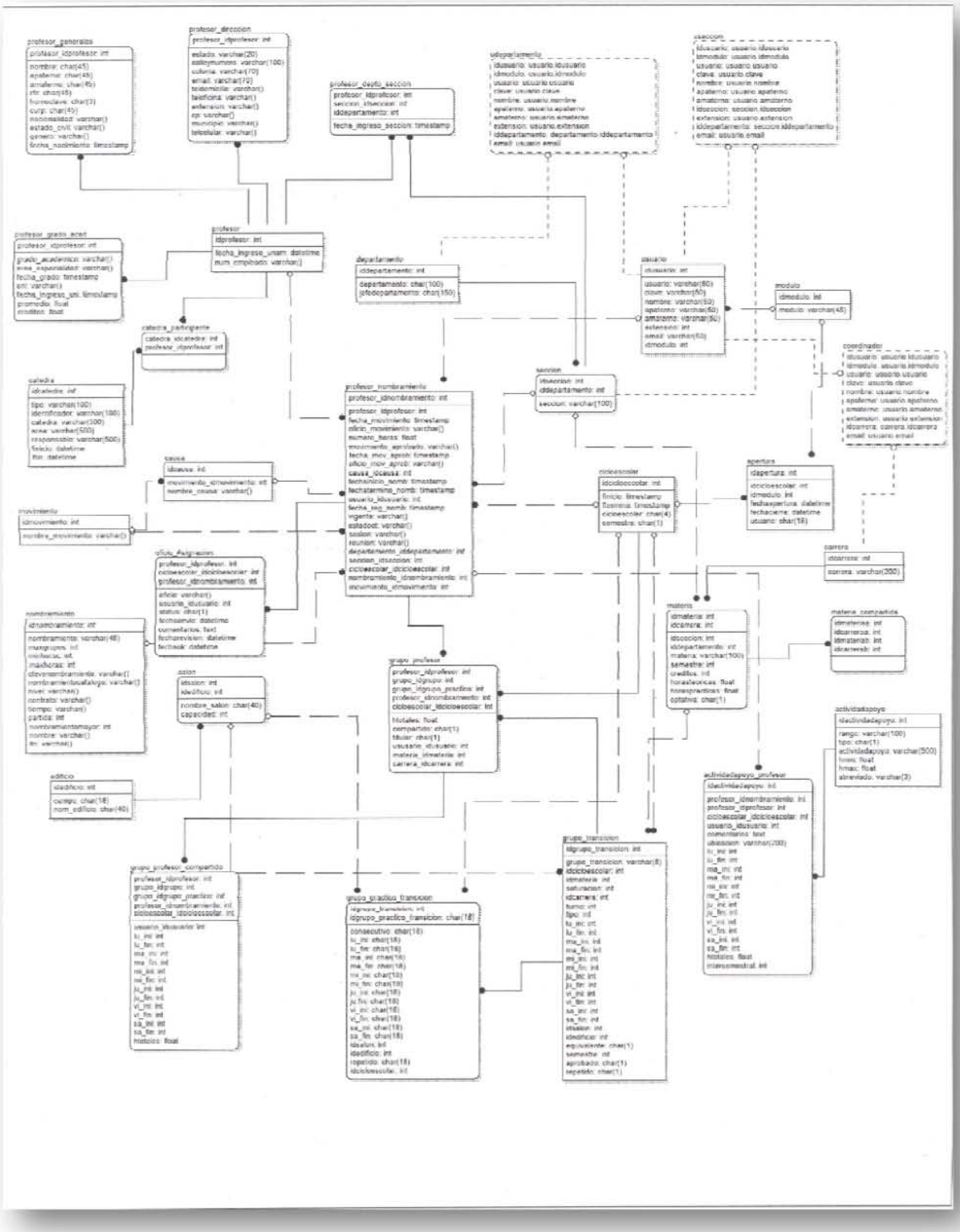


Figura 5.3 Diagrama de la base de datos

5.3.2 Diccionario de datos

Tabla	Nombre del campo	Tipo	Definición
actividadapoyo	Idactividadapoyo	int	Id de la actividad de apoyo.
	Rango	varchar(100)	PACIVE, PAPITT, Apoyo Académico, etc.
	tipo	char(1)	A, B, C.
	actividadapoyo	varchar(500)	Nombre de la actividad de apoyo.
	hmin	float	Mínimo de horas que se pueden asignar a esa actividad.
	hmax	float	Máximo de horas que se pueden asignar a esa actividad.
	abreviado	varchar(3)	Abreviatura del rango.
actividadapoyo_profesor	idactividadapoyo	int	Id de la actividad de apoyo.
	profesor_idnombramiento	int	Id del nombramiento del profesor.
	profesor_idprofesor	int	Id del profesor.
	cicloescolar_idcicloescolar	int	Id del ciclo escolar.
	usuario_idusuario	int	Id del usuario.
	comentarios	text	Descripción de la actividad que realiza.
	ubicacion	varchar(200)	Ubicación donde se realiza la actividad de apoyo.
	lu_ini	int	Horario
	lu_fin	int	Horario
	ma_ini	int	Horario
	ma_fin	int	Horario
	mi_ini	int	Horario
	mi_fin	int	Horario
	ju_ini	int	Horario
	ju_fin	int	Horario
	vi_ini	int	Horario
	vi_fin	int	Horario
	sa_ini	int	Horario
	sa_fin	int	Horario
	htotales	float	Total de horas asignadas a la actividad de apoyo
	Intersemestral	int	Indica si la actividad de apoyo corresponde al semestre o al intersemestre (1,0)
apertura	idapertura	int	Id único de la apertura.
	idcicloescolar	int	Id único del ciclo escolar.
	idmodulo	int	Modulo al que se dará apertura de manera general (secciones, coordinadores, depto., etc.).
	fechaapertura	datetime	Fecha de apertura para el modulo.
	fechacierre	datetime	Fecha de cierre para el modulo.
	usuario	char(18)	Usuario del modulo que tendrá

			acceso.
carrera	idcarrera	int	Identificador único de la carrera.
	carrera	varchar(200)	Nombre de la carrera
catedra	idcatedra	int	Identificador único de la catedra.
	tipo	varchar(100)	Tipo de la catedra (PACIVE, PAPITT, etc.).
	identificador	varchar(100)	Identificador de la catedra
	catedra	varchar(500)	Nombre de la catedra.
	area	varchar(500)	Área en la que se desarrolla la catedra.
	responsable	varchar(500)	Responsable de la catedra.
	finicio	datetime	Fecha de inicio de la catedra.
	Ffin	datetime	Fecha de termino de la catedra.
catedra_participante	idcatedra	int	Identificador único de la catedra.
	idprofesor	int	Identificador único del profesor.
causa	idcausa	int	Id único de la causa
	idmovimiento	int	Id del movimiento
	nombre_causa	varchar()	Nombre de la causa que tendrá el nombramiento de algún profesor.
cicloescolar	idcicloescolar	int	Id único del ciclo escolar
	finicio	timestamp	Fecha en que inicia el ciclo escolar.
	ftermino	timestamp	Fecha en que termina el ciclo escolar.
	cicloescolar	char(4)	Año del ciclo escolar (2011).
	semestre	char(1)	Semestre del ciclo escolar (1,2).
departamento	iddepartamento	int	Identificador único del departamento.
	departamento	char(100)	Nombre del departamento.
	jefedepartamento	char(150)	Responsable del departamento.
edificio	idedificio	int	Id único del edificio.
	campo	char(18)	Campo en el que se encuentra ubicado el edificio.
	nom_edificio	char(40)	Nombre del edificio.
grupo_practico_transicion	idgrupo_transicion	int	Id del grupo del cual es su parte práctica y que se encuentra en grupo_transicion.
	idgrupo_practico_transicion	char(18)	Identificador del grupo practico, es consecutivo según el número de grupos practico se abran para un grupo teórico.
	consecutivo	char(18)	Letra que se asigna al grupo practico, (A,B,C,....)
	lu_ini	char(18)	Horario
	lu_fin	char(18)	Horario
	ma_ini	char(18)	Horario
	ma_fin	char(18)	Horario
	mi_ini	char(18)	Horario
	mi_fin	char(18)	Horario

	ju_ini	char(18)	Horario
	Ju_fin	char(18)	Horario
	vi_ini	char(18)	Horario
	vi_fin	char(18)	Horario
	sa_ini	char(18)	Horario
	sa_fin	char(18)	Horario
	idsalon	int	Id del salon.
	idedificio	int	Id del edificio.
	repetido	char(18)	Se utiliza cuando se asignan dos aulas a un grupo.
	idcicloescolar	int	Id del ciclo escolar.
grupo_profesor	profesor_idprofesor	int	Id del profesor.
	idgrupo_transicion	int	Id del grupo.
	grupo_idgrupo_practico	int	El id del grupo practico en caso de que el grupo sea practico y 0 si es teórico
	profesor_idnombramiento	int	Id del nombramiento.
	idcicloescolar	int	Id del ciclo escolar.
	htotales	float	Horas asignadas al profesor en ese grupo.
	compartido	char(1)	Indica si el grupo ha sido partido para ser asignado en dos nombramientos del profesor o en nombramientos de distintos profesores.
	titular	char(1)	Indica quien es el titular del grupo.
	ususario_idusuario	int	Usuario que realizo el movimiento.
	materia_idmateria	int	Id de la materia.
	carrera_idcarrera	int	Id de la carrera.
grupo_profesor_compartido	profesor_idprofesor	int	Id del profesor
	idgrupo_transicion	int	Id del grupo.
	grupo_idgrupo_practico	int	Id del grupo práctico.
	profesor_idnombramiento	int	Id del nombramiento.
	idcicloescolar	int	Id del ciclo escolar.
	usuario_idusuario	int	Usuario que realiza el movimiento.
	lu_ini	int	Horario del grupo, este horario es nuevo debido a que el grupo ha sido partido y el horario es distinto, por lo tanto en la tabla grupo profesor los que tengan un grupo con un 1 en el campo compartido, deberán acceder al horario desde esta tabla.
	lu_fin	int	Horario
	ma_ini	int	Horario
	ma_fin	int	Horario
	mi_ini	int	Horario
	mi_fin	int	Horario

	ju_ini	int	Horario
	ju_fin	int	Horario
	vi_ini	int	Horario
	vi_fin	int	Horario
	sa_ini	int	Horario
	sa_fin	int	Horario
	htotales	float	Horas totales asignadas en este grupo.
grupo transicion	idgrupo_transicion	int	Id del grupo.
	grupo_transicion	varchar(5)	Nombre del grupo.
	idcicloescolar	int	Id del ciclo escolar.
	idmateria	int	Id de la materia.
	saturacion	int	Saturacion del grupo.
	idcarrera	int	Id de la carrera.
	turno	int	turno (matutino, vespertino)
	tipo	int	1 Teórico 2 Teórico-Práctico 3 Práctico
	lu_ini	int	Horario
	lu_fin	int	Horario
	ma_ini	int	Horario
	ma_fin	int	Horario
	mi_ini	int	Horario
	mi_fin	int	horario
	ju_ini	int	horario
	ju_fin	int	horario
	vi_ini	int	horario
	vi_fin	int	horario
	sa_ini	int	horario
	sa_fin	int	horario
	idsalon	int	salón
	idedificio	int	edificio
	equivalente	char(1)	
	semestre	int	semestre en que se imparte la materia
	aprobado	char(1)	Estado que guarda el grupo
	repetido	char(1)	
materia	idmateria	int	Identificador de la materia
	idcarrera	int	Identificador único de la carrera
	idseccion	int	Id de la sección a la que pertenece la carrera
	iddepartamento	int	Id del departamento al que pertenece la carrera
	materia	varchar(100)	Nombre de la materia
	semestre	int	Semestre en el que se imparte la materia

	creditos	int	Creditos de la materia
	horasteoricas	float	Horas teóricas de la materia según plan de estudios
	horaspracticas	float	Horas practicas de la materia según plan de estudios
	optativa	char(1)	Indica si la materia es optativa (1,0)
materia_compartida	idmateriaa	int	Identificador de la materia origen
	idmateriab	int	Identificador de la carrera origen
	idmateria	int	identificador de la materia compartida
	idcarrera	int	Identificador de la carrera a la que pertenece la materia compartida
modulo	idmodulo	int	Identificador del modulo
	modulo	varchar(45)	Nombre del modulo
movimiento	idmovimiento	int	Id del movimiento
	nombre_movimiento	varchar()	Nombre del tipo de movimiento (alta, baja, etc.)
nombramiento	idnombramiento	int	Id único del nombramiento
	nombramiento	varchar(45)	Nombre del nombramiento
	maxgrupos	int	Máximo de grupos que puede tener un profesor según el nombramiento.
	minhoras	int	Mínimo de horas de docencia según el nombramiento.
	maxhoras	int	Máximo de horas de docencia según el nombramiento
	clavenombramiento	varchar()	Clave tipo texto del nombramiento
	nombramientocatalogo	varchar()	Nombre del nombramiento
	nivel	varchar()	Nivel del nombramiento (A,B,C, etc.)
	contrato	varchar()	Tipo de contrato (Interino o definitivo)
	tiempo	varchar()	Cantidad de horas asignadas al nombramiento (horas, mt o tc), por horas es null en esta tabla por que el usuario las introduce cuando hace la relación profesor-nombramiento en la tabla de prof_nombramiento
	partida	int	Partida para el tipo de nombramiento.
	nombramientomayor	int	Nombramiento mayor
	nombre	varchar()	Nombre corto del nombramiento
	Ltn	varchar()	ltn
oficio_Asignacion	profesor_idprofesor	int	Id único del profesor
	cicloescolar_idcicloescolar	int	Id del ciclo escolar de la asignación que se enviara a UAPA
	profesor_idnombramiento	int	Id único del profesor
	oficio	varchar()	Numero de oficio
	usuario_idusuario	int	id del usuario que realizo el movimiento
	status	varchar(1)	Nos indica el status del nombramiento (aprobado,

			rechazado, en revisión, etc.).
	fechaenvio	datetime	Fecha de envío
	comentarios	text	Comentarios según el status
	fecharevision	datetime	Fecha de revisión por UAPA
	fechaok	datetime	Fecha revisión ok por UAPA
profesor	idprofesor	int	Identificador único del profesor
	fecha_ingreso_unam	datetime	Fecha de ingreso a la UNAM
	num_empleado	varchar()	Numero de trabajador asignado por la UNAM
profesor_depto_seccion	idprofesor	int	Identificador único del profesor
	idseccion	int	Identificador único de la seccion
	iddepartamento	int	Identificador único del departamento al que pertenece la seccion
	fecha_ingreso_seccion	timestamp	Fecha en que ingreso a la seccion
profesor_direccion	idprofesor	int	Identificador único del profesor
	estado	varchar(20)	Estado de la Republica donde vive
	calleynumero	varchar(100)	Calle y numero
	colonia	varchar(70)	Colonia
	email	varchar(70)	Correo electrónico
	teldomicilio	varchar()	Teléfono de domicilio
	teloficina	varchar()	Teléfono de oficina
	extension	varchar()	Extension
	Cp	varchar()	Código postal
	municipio	varchar()	Delegación o Municipio
	telcelular	varchar()	Teléfono celular
profesor_generales	idprofesor	int	Identificador único del profesor
	nombre	char(45)	Nombre del profesor
	apaterno	char(45)	Apellido paterno
	amaterno	char(45)	Apellido materno
	Rfc	char(45)	RFC del profesor
	homoclave	char(3)	Homoclave 3 dígitos
	curp	char(45)	CURP del profesor
	nacionalidad	varchar()	Nacionalidad del profesor
	estado_civil	varchar()	Estado civil
	genero	varchar()	Genero del profesor
	fecha_nacimiento	timestamp	Fecha de nacimiento
profesor_grado_acad	idprofesor	int	Identificador único del profesor
	grado_academico	varchar()	Grado académico al momento del registro
	area_especialidad	varchar()	Área de especialidad
	fecha_grado	timestamp	Fecha de obtención de grado
	sni	varchar()	Nivel del SNI en caso de tener
	fecha_ingreso_sni	timestamp	Fecha de ingreso al SNI

	promedio	float	Promedio en caso de no ser titulado
	creditos	float	Numero de créditos en caso de no ser titulado
profesor_nombramiento	profesor_idnombramiento	int	Id del nombramiento del profesor
	idprofesor	int	Identificador único del profesor
	fecha_movimiento	timestamp	Fecha en que se realizo el movimiento
	oficio_movimiento	varchar()	Oficio por el cual se crea el nombramiento
	numero_horas	float	Número de horas asignadas al nombramiento
	movimiento_aprobado	varchar()	Identificador de el estado del movimiento (aprobado o no aprobado)
	fecha_mov_aprob	timestamp	Fecha de aprobación
	oficio_mov_aprob	varchar()	Oficio del movimiento de aprobación
	idcausa	int	Identificador de la causa
	fechainicio_nomb	timestamp	Fecha de inicio para el nombramiento
	fechatermino_nomb	timestamp	Fecha de termino para el nombramiento
	idusuario	int	Usuario que realizo el movimiento
	fecha_reg_nomb	timestamp	Fecha de registro
	vigente	varchar()	Vigencia del nombramiento
	estadoct	varchar()	Estado de C.T.
	sesion	varchar()	Sesion en que aprobó C.T.
	reunion	varchar()	Reunion en que aprobó C.T.
	iddepartamento	int	Identificador de departamento
	idseccion	int	Id de la seccion
	idcicloescolar	int	Id del ciclo escolar.
	idnombramiento	int	Id del nombramiento.
	idmovimiento	int	Id del movimiento.
salon	idsalon	int	Id del salon.
	idedificio	int	Id del edificio.
	nombre_salon	char(40)	Nombre del salón.
	capacidad	int	Capacidad de alumnos que permite el salón.
seccion	idseccion	int	Identificador único de la sección
	iddepartamento	int	Identificador único del departamento al que pertenece la seccion
	seccion	varchar(100)	Nombre de la sección
usuario	idusuario	int	Identificado único del usuario
	usuario	varchar(50)	Nombre con que el usuario ingresara al sistema (SHA1)
	clave	varchar(50)	Contraseña de acceso al sistema (SHA1)
	nombra	varchar(50)	Nombre del usuario

	apaterno	varchar(50)	Apellido paterno del usuario
	amaterno	varchar(50)	Apellido materno del usuario
	extension	Int	Numero de extension dentro de la UNAM
	email	varchar(50)	Correo electrónico
	idmodulo	Int	Modulo al que pertenece el usuario según su contratación

5.4 Descripción del modulo de consejo Técnico

A continuación veremos una descripción mas a detalle y grafica de las función primordial del modulo de consejo técnico, que es la de aprobación o rechazo de nombramientos.

El usuario de consejo técnico deberá seleccionar el ciclo escolar sobre el que quiera trabajar como se muestra en la figura siguiente:

Figura 5.4 selsemestre.jsp

Técnicamente al seleccionar el ciclo escolar y presionar el botón aceptar hacemos uso del archivo selsemestre.jsp, del cual se muestra parte de su código en la figura 5.5

```
71 <h2><%= msg1 %></h2>
72 <html:form action="selsemestre">
73 <html:hidden name="SelsemestreForm" property="idusuario" value='<%= idu.toString() %>' />
74 <html:hidden name="SelsemestreForm" property="pagorigen" value='<%=pagorigen.toString() %>' />
75 <table width="730" border="1">
76  <tr>
77  <th scope="col">Semestre</th>
78     </tr>
79      <tr>
80       <th scope="col"><label>
81           <html:select property="uciclo1" name="SelsemestreForm">
82             <html:option value="2">2012-1</html:option>
83             <html:option value="3">2012-2</html:option>
84             <html:option value="4">2013-1</html:option>
85             <html:option value="5">2013-2</html:option>
86             <html:option value="6">2014-1</html:option>
87             <html:option value="7">2014-2</html:option>
88             <html:option value="8">2015-1</html:option>
89             <html:option value="9">2015-2</html:option>
90       </html:select>
91      </label></th>
92        </tr>
93        <tr>
94        <td colspan="5"><label>
95        <div align="center">
96        <html:submit value="Aceptar"/>
97        </div>
```

Figura 5.5 Código selsemestre.jsp

En el código del archivo selsemestre.jsp mostrado en la figuras 5.5 se hace uso del Form SelsemestreForm y de sus propiedades idusuario, pagorigen y uciclo1, en donde el idususario es la propiedad del Form que almacenara el identificador de usuario, la propiedad pagorigen que almacena el nombre de la pagina donde se encuentra el Form y la propiedad uciclo que almacena el ciclo escolar seleccionado por el usuario.

Para poder utilizar un Form y sus propiedades, previamente se debe haber creado dicho Form en el paquete formBean el cual es el paquete que contiene todos los forms que se utilizan en la aplicación tal como se muestra en la siguiente figura:

```
25  public class SelsemestreForm extends ActionForm{
26
27  private String uciclo1;
28  private String pagorigen;
29  private String idusuario;
30
31  public String getuciclo1(){
32                  return uciclo1;
33  }
34  public String getpagorigen(){
35                 return pagorigen;
36  }
37  public String getidusuario(){
38                 return idusuario;
39  }
40
41
42  public void setuciclo1(String puciclo1){
43                  uciclo1 = puciclo1.trim();
44  }
45  public void setpagorigen(String pagorigen1){
46                  pagorigen = pagorigen1.trim();
47  }
48  public void setidusuario(String pidusuario){
49                  idusuario = pidusuario.trim();
50  }
51
52
53  public SelsemestreForm(){
54  super();
```

Figura 5.6 Form SelsemestreForm.java

Una vez que se creó el form SelsemestreForm con sus propiedades uciclo1, pagorigen e idusuario (figura 5.6), se configura el archivo struts-config.xml que entrara en acción al presionar el botón aceptar en la figura 5.4 y del cual a continuación mostramos parte de su contenido para explicar su funcionamiento.

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2
3 <!DOCTYPE struts-config PUBLIC
4         "-//Apache Software Foundation//DTD Struts Configuration 1.3//EN"
5         "http://jakarta.apache.org/struts/dtds/struts-config_1_3.dtd">
6
7
8 <struts-config>
9   <form-beans>
10        <form-bean name="LoginForm" type="formBean.LoginForm" />
11        <form-bean name="SemestreForm" type="formBean.SemestreForm" />
12        <form-bean name="AperturaForm" type="formBean.AperturaForm" />
13        <form-bean name="CoordinacionForm" type="formBean.CoordinacionForm" />
14        <form-bean name="cAperturaForm" type="formBean.cAperturaForm" />
15        <form-bean name="CapturaGrupoForm" type="formBean.CapturaGrupoForm" />
16        <form-bean name="ConsejoForm" type="formBean.ConsejoForm" />
17        <form-bean name="SelsemestreForm" type="formBean.SelsemestreForm" />
18        <form-bean name="ProfesorForm" type="formBean.ProfesorForm" />
19        <form-bean name="UsuarioForm" type="formBean.UsuarioForm" />
20        <form-bean name="DepartamentoForm" type="formBean.DepartamentoForm" />
```

Figura 5.7 Archivo struts-config.xml

En la sección <form-beans> de la figura 5.7 que corresponde al archivo struts-config.xml podemos observar que todos los Form que se crean deben de definirse en esta sección del archivo, en el cual podemos observar, además de otros Form que se utilizan en la aplicación, el Form SelsemestreForm que tiene los atributos name (Identificador único para referenciarlo en los mapeos de acciones) y type (ruta completa de la clase Java del bean de formulario).

En la sección <action-mapping> que vernos en la figura 5.8 se especifica las definiciones de acciones que el archivo Struts-config.xml realizara.

```
87      <action input="/index-consejo.jsp" name="ConsejoForm" path="/consejo"
        type="actionBean.Consejo" >
88
89          <forward name ="successG" path ="/coordinacion-extendido.jsp"/>
90          <forward name ="successP" path ="/consejo_profesor.jsp"/>
91          <forward name ="failure" path ="/error.html"/>
92      </action>
93      <action input="/selsemestre.jsp" name="SelsemestreForm" path="/selsemestre"
        type="actionBean.Selsemestre" >
94          <forward name ="success" path ="/selsemestre.jsp"/>
95          <forward name ="failure" path ="/errorSelSem.html"/>
96      </action>
97      <action input="/departamento.jsp" name="DepartamentoForm" path="/departamento"
        type="actionBean.Departamento" >
98          <forward name ="success" path ="/index-seccion.jsp"/>
100         <forward name ="failure" path ="/error.html"/>
101     </action>
```

Figura 5.8 Sección <action-mapping>

En esta sección se usa el elemento <action> por cada acción y se compone de los siguientes atributos:

- Input: en el cual se especifica el nombre del archivo en el cual se utiliza el Form que invoca al struts-config.xml
- Name: indica el nombre del Form que se está utilizando
- Path: es el action que se declaro en el Form dentro del archivo origen, en este caso línea 72 de la figura 5.5 del código selsemestre.jsp;
- Type: el cual indica la acción que realizara el struts-config.xml y que en este caso es una llamada al archivo Selsemestre.java

En cada <action> se puede definir más de un forward, el cual tiene las siguientes propiedades:

- Name: nombre del forward que será utilizado al llamarlo desde la clase Action
- Path: ruta de la página JSP

Una vez que el <action-mapping> hace el llamado al código Selsemestre.java, este se encarga de verificar las fechas para las cuales el sistema estará abierto

para el usuario CT y si esta en un periodo valido, Selsemestre.java envía un forward success para permitir el acceso al sistema o de lo contrario enviar un forward failure (fig. 5.9).

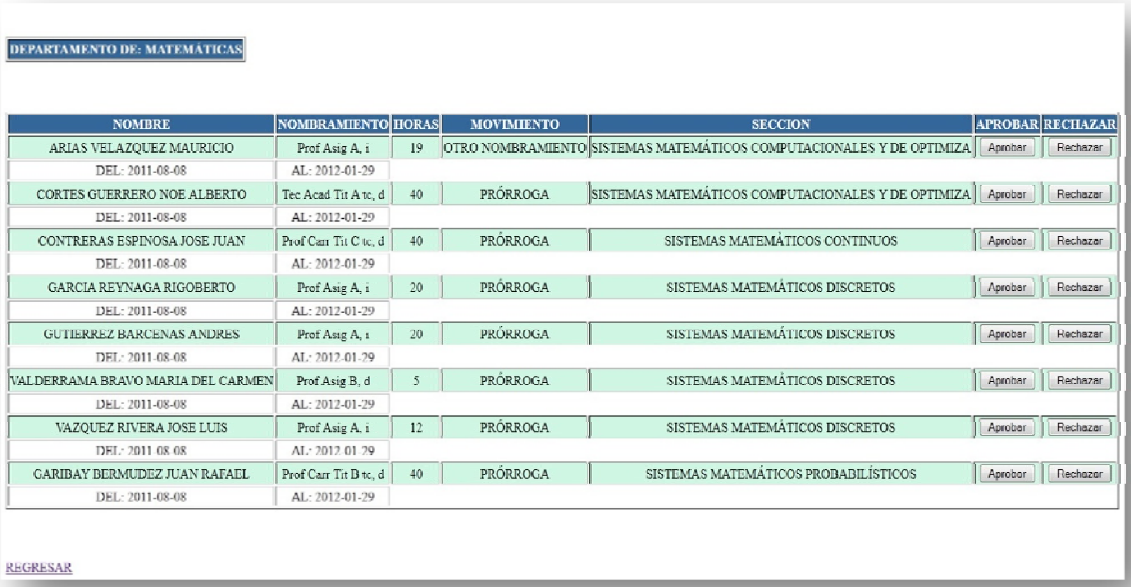
```
37      SelsemestreForm lf = (SelsemestreForm)form;
38      HttpSession hs=request.getSession();
39      hs.setAttribute("idcicloescolar",lf.getuciclo1());
40      querys query = new querys();
41      String qry = "Select * from apertura where idcicloescolar = "+lf.getuciclo1()+" and
        idmodulo="+hs.getAttribute("idmodulo")+ " and (usuario="+lf.getidusuario()+" or usuario=500)";
42       ResultSet rs = query.consultar(qry);
43          Integer tempo=0;
44          while (rs.next()) {
45              calendario calfi1 = new calendario(rs.getString("fechaapertura"));
46              calendario calft1 = new calendario(rs.getString("fechacierre"));
47              Date now = Calendar.getInstance().getTime();
48              if( now.after(calfi1.getdate()) && now.before(calft1.getdate())){
49                    tempo=tempo+1;
50               } else { tempo=tempo; }  }
51              if (tempo!=0){
52                 hs.setAttribute("mopen",1);
53                 response.sendRedirect(lf.getpagorigen());
54              }else{
55                          hs.setAttribute("mopen",0);
56                          response.sendRedirect(lf.getpagorigen());
57      }         }
58    return mapping.findForward("failure");
```

Figura 5.9 Selsemestre.java

Ya establecido el ciclo escolar sobre el que se desea trabajar, el usuario puede seleccionar el departamento.

Figura 5.10 seldepto_ct.jsp

Una vez seleccionado el departamento el sistema muestra todos los nombramientos que la Unidad de Asuntos del Personal Académico a enviado a consejo técnico ya que cumplieron con las especificaciones del Estatuto del Personal Académico y pueden pasar a su revisión para ser aprobados o rechazados según el criterio del consejo técnico, lo anterior lo podemos observar en la siguiente figura:



DEPARTAMENTO DE: MATEMATICAS

NOMBRE	NOMBRAMIENTO	HORAS	MOVIMIENTO	SECCION	APROBAR	RECHAZAR
ARIAS VELAZQUEZ MAURICIO	Prof Asig A, i	19	OTRO NOMBRAMIENTO	SISTEMAS MATEMÁTICOS COMPUTACIONALES Y DE OPTIMIZA	Aprobar	Rechazar
DEL: 2011-08-08	AL: 2012-01-29					
CORTES GUERRERO NOE ALBERTO	Tec Acad Tit A tc, d	40	PRÓRROGA	SISTEMAS MATEMÁTICOS COMPUTACIONALES Y DE OPTIMIZA	Aprobar	Rechazar
DEL: 2011-08-08	AL: 2012-01-29					
CONTRERAS ESPINOSA JOSE JUAN	Prof Carr Tit C tc, d	40	PRÓRROGA	SISTEMAS MATEMÁTICOS CONTINUOS	Aprobar	Rechazar
DEL: 2011-08-08	AL: 2012-01-29					
GARCIA REYNAGA RIGOBERTO	Prof Asig A, i	20	PRÓRROGA	SISTEMAS MATEMÁTICOS DISCRETOS	Aprobar	Rechazar
DEL: 2011-08-08	AL: 2012-01-29					
GUTIERREZ BARCENAS ANDRES	Prof Asig A, i	20	PRÓRROGA	SISTEMAS MATEMÁTICOS DISCRETOS	Aprobar	Rechazar
DEL: 2011-08-08	AL: 2012-01-29					
VALDERRAMA BRAVO MARIA DEL CARMEN	Prof Asig B, d	5	PRÓRROGA	SISTEMAS MATEMÁTICOS DISCRETOS	Aprobar	Rechazar
DEL: 2011-08-08	AL: 2012-01-29					
VAZQUEZ RIVERA JOSE LUIS	Prof Asig A, i	12	PRÓRROGA	SISTEMAS MATEMÁTICOS DISCRETOS	Aprobar	Rechazar
DEL: 2011-08-08	AL: 2012-01-29					
GARIBAY BERMUDEZ JUAN RAFAEL	Prof Carr Tit B tc, d	40	PRÓRROGA	SISTEMAS MATEMÁTICOS PROBABILÍSTICOS	Aprobar	Rechazar
DEL: 2011-08-08	AL: 2012-01-29					

REGRESAR

Figura 5.11 Nombramientos propuestos a CT

Para poder mostrar la vista de la figura 5.11 es necesario hacer una consulta a la base de datos que nos permita recabar dicha información lo cual podemos observar técnicamente en la figura siguiente:

```
125  querys prorrogas = new querys();
126  String qprorrogas = "select pg.apaterno, pg.amaterno, pg.nombre,nombramiento.nombramiento,
127  datos.numero_horas,"
128  + " causa.nombre_causa, seccion.seccion, datos.fechainicio_nomb, datos.fechatermino_nomb, "
129  + "datos.profesor_idnombramiento, datos.profesor_idprofesor from causa,seccion,profesor_generales
       pg,"
130  + " nombramiento,(select * from profesor_nombramiento where profesor_idnombramiento "
131  + "in(select profesor_idnombramiento from oficio_asignacion where usuario_idusuario "
132  + "in(select idusuario from useccion where iddepartamento="+tempo2+") and status ='4' "
133  + "and cicloescolar_idcicloescolar="+ciclo+")) as datos where
       pg.profesor_idprofesor=datos.profesor_idprofesor "
134  + "and nombramiento.idnombramiento=datos.nombramiento_idnombramiento "
135  + "and causa.idcausa=datos.causa_idcausa "
136  + "and seccion.idseccion=datos.seccion_idseccion order by seccion,apaterno";
137  try{
138  ResultSet rsprorrogas = prorrogas.consultar(qprorrogas);
```

Figura 5.12 aprueba_ct.jsp

La consulta es almacenada en una variable de tipo String de nombre qprorrogas (línea 126, Fig. 5.12), dicha consulta está compuesta de una serie de subconsultas de manera que se recaba la información necesaria desde varias tablas de la base de datos en la misma consulta.

Para poder ejecutar la consulta es necesario en primera instancia crear un objeto de tipo de la clase querys de nombre prorrogas (línea 125, Fig. 5.12), además de un ResultSet de nombre rsprorrogas (línea 139 Fig. 5.12) donde se almacenara el resultado de la consulta para su posterior despliegue en pantalla.

Una vez que el usuario CT está en la pantalla de nombramientos propuestos al CT (fig. 5.11) puede aprobar o rechazar los nombramientos que se muestran, para eso el usuario CT solo presiona el botón Aprobar o en el caso contrario el botón rechazar para realizar dicha acción; una vez que se presiona el botón Aprobar se ejecuta internamente una función java script, la cual se encarga de redirigirnos al archivo aprueba_consejo.java; a continuación se muestra parte del código con la función java script.

```
<script type="text/javascript" language="javascript">
        function update_ct(reu,ses,idprof,idnom,depa,dia,mes,anio){
        var reunion=reu;
        var sesion=ses;
        var profe=idprof;
        var nomb=idnom;
        var depto=depa;
        var id="1";
        var dia=dia;
        var mes=mes;
        var anio=anio;

location.href='./aprueba_consejo?reunion='+reunion+'&sesion='+sesion+'&prof='+profe+'&nomb='+nomb+'&
depto='+depto+'&id='+id+'&dia='+dia+'&mes='+mes+'&anio='+anio;
  }
 </script>
```

Figura 5.13 aprueba_ct.jsp

Como podemos observar en el fragmento de código de la figura 5.13, la función recibe una serie de parámetros, los cuales a su vez reenvía mediante un llamado al archivo aprueba_consejo.java, el cual es el encargado de cambiar el status del nombramiento en la base de datos como veremos en el código siguiente:

```
}else if((reunion.compareTo("")!=0)&&sesion.compareTo("")!=0){
     //actualizamos oficio_asignacion y profesor_nombramiento para cambiar el estado aprobado por ct
    String act="update oficio_asignacion set status='5' where profesor_idprofesor="+prof+" and
    profesor_idnombramiento="+nomb+" and status='4' ";
     try{
            Integer rsupdate=update.actualiza(act);
     }catch (Exception e) {System.out.print("El update en apueba_consejo fallo");}

    String act2="update profesor_nombramiento set movimiento_aprobado='APROBADO',
    fecha_mov_aprob="'+dia+"-"+mes+"-"+anio+"', estadoct='5', sesion=\'"+sesion+"\', reunion=\'"+reunion+"\'
    where profesor_idprofesor="+prof+" and profesor_idnombramiento="+nomb+"";

    try{
         Integer rsupdate2=update.actualiza(act2);
    }catch(Exception e){System.out.print("El update en apueba_consejo fallo");}

 }else{
        JOptionPane.showMessageDialog(null,"Olvido llenar el campo de Oficio/Sesion/Reunion");
 }
```

Figura 5.14 aprueba_consejo.java

En el código de la figura 5.14 podemos observar que se actualizan las tablas oficio_asignacion y profesor_nombramiento para poner el estatus APROBADO por CT, al igual que los datos de la reunión, la sesión y la fecha de aprobación del CT sobre el nombramiento en cuestión; una vez actualizadas las tablas el archivo aprueba_consejo.java regresa el control al archivo aprueba_ct.jsp para que el CT siga aprobando o rechazando nombramientos; en caso de los nombramientos rechazados el procedimiento es idéntico.

Toda aprobación o rechazo realizado por el CT se ve reflejado en el modulo de secciones, el modulo de UAPA y el modulo de personal, indicando a los distintos usuarios involucrados en el proceso que el nombramiento ha terminado de manera satisfactoria el proceso de aprobaciones del CT.

5.5 Pruebas

Se realizaron una serie de pruebas para comprobar la calidad del software que se construyo, entre las pruebas realizadas se encuentran las siguientes:

- **Pruebas integrales:** se realizaron una seria de pruebas para confirmar que el nuevo modulo se adaptara a los módulos ya existentes ya que el modulo de consejo técnico trabaja en estrecha relación con el modulo de la Unidad de Asuntos del Personal Académico y el modulo de personal, por lo cual en estas pruebas se verifico la veracidad de la información que se comparte entre los módulos así como interacción entre los mismos.
- **Casos de prueba:** se realizaron pruebas para verificar el funcionamiento del sistema en las cuales se incluyeron entradas reales para validar los resultados esperados y se hicieron sobre condiciones y casos reales aplicables al sistema.
- **Pruebas beta:** se realizaron pruebas con los usuarios finales del software directamente en su lugar de trabajo para validar que el software construido cumpliera con los requisitos establecidos por los mismos usuarios del modulo de consejo técnico, además de verificar la usabilidad del mismo, estas pruebas se realizaron durante una semana obteniendo resultados favorables.

El objetivo de realizar las pruebas antes mencionadas fue mejorar la calidad de la aplicación y cumplir al 100% con los requerimientos esperados por el usuario.

5.6 Capacitación

Es claro que siempre que se utiliza un software por primera vez es necesario de una capacitación para poder utilizar el mismo y sacar el mayor provecho de la nueva aplicación; una vez que se termino el modulo de Consejo Técnico se realizó una reunión con los usuarios del mismo, además de los usuarios de los módulos relacionados directamente, como los usuarios de la Unidad de Asuntos del personal Académico y Personal para que pudiesen tener un enfoque global del funcionamiento del sistema y lograr entender así lo importante de sus funciones para el buen desempeño de los módulos en su conjunto.

Posteriormente, se realizó una reunión de capacitación exclusiva de los usuarios del modulo de Consejo Técnico, en la cual los usuarios interactuaron por primera vez sobre el sistema, logrando familiarizarse con la aplicación y validando que sus requerimientos se cumplieran al 100%.

5.7 Liberación

Una vez concluido el periodo de pruebas y la capacitación necesaria a los usuarios se procedió con la entrega de nombres de usuarios y contraseñas para las diferentes personas que interactúan con el sistema, posteriormente el usuario accedió al sistema desde su equipo validando el funcionamiento de sus cuentas.

Una vez terminado el proceso de entrega de cuentas de usuario se firmo el oficio de entrega y aceptación de las cuentas y funcionamiento del modulo en general.

Actualmente el sistema se encuentra funcionando y con resultados satisfactorios para el proceso de asignación de la docencia y la facultad en general.

5.8 Conclusiones y trabajo futuro

El Sistema de Asignaciones de la Facultad de Estudios Superiores Cuautitlán “SAFESC” es una aplicación web que se desarrollo con la finalidad de facilitar, automatizar, reducir tiempos, evitar errores y ayudar a las diversas coordinaciones, departamentos, secciones, Unidad de Asuntos del Personal Académico y Consejo Técnico en el proceso de asignación de la docencia de la facultad.

El desarrollo del SAFESC fue una tarea complicada debido al gran tamaño y complejidad del sistema, pero con la ayuda de métodos como el ciclo de vida del software, especialmente el ciclo de vida en espiral, se logro estructurar de manera solida las bases para su desarrollo.

La etapa de análisis fue una de las más complicadas, debido a que a la mayoría de los usuarios les era difícil describir el proceso que realizaban, creando confusión para poder establecer los requerimientos básicos de los usuarios, pero a pesar de todo se pudo realizar un consenso y establecer los requerimientos y alcances para los módulos del sistema.

La etapa de codificación y prueba fue un proceso muy complicado, debido a que los tiempos eran muy reducidos, por lo cual se trabajo a marchas forzadas en estas etapas y realizando modificaciones, producto del resultado de alguna prueba, prácticamente al momento de descubrir el error.

El trabajo en equipo, los conocimientos y las habilidades técnicas que obtuve a lo largo de mi formación universitaria fueron fundamentales para el éxito del desarrollo del SAFESC.

El SAFESC actualmente el es sistema sobre el cual se trabaja toda la administración de la docencia de la facultad, logrando así una implementación exitosa, además de ser una valiosa herramienta para todas aquellas personas que intervienen en el proceso.

Considero que gracias al SAFESC la gestión del proceso de asignación de la docencia tiene un mayor control, su grado de automatización permite reducir los tiempos de forma considerable y la recuperación de la información por medio de reportes de la base de datos permite a la institución tomar decisiones adecuadas en el momento preciso.

Dentro de las fortalezas y trabajo futuro del SAFESC cabe mencionar que este proyecto está conformado de manera que puede expandir su alcance mediante actualizaciones necesarias para seguir automatizando el proceso o funciones

relacionadas con el mismo; dentro de las actualizaciones, se está trabajando en el modulo de Administración del sistema para poder generar una serie de reportes que se le solicitan a la facultad anualmente y en los cuales actualmente se invierte aproximadamente una semana de tiempo generar, pero una vez terminada la actualización dichos reportes se generaran de forma instantánea, facilitando así las labores del administrador del sistema y la carga de trabajo de la Secretaria de Planeación.

Actualmente el sistema se encuentra funcionando y con resultados satisfactorios para realizar el proceso de asignación de la docencia y con buena aceptación por parte de los usuarios que intervienen en el proceso ya que han notado que el sistema facilita el proceso y recorta el tiempo para realizarlo.

GLOSARIO

Grupo: Conjunto de estudiantes que asisten al mismo grado y aula de clase

Académico: Individuo que pertenece a una corporación académica

Docente: Relativo a quien enseña

Cátedra de Investigación: Proyecto de investigación realizado por académicos de la FESC

Software libre: Es un tipo de software que se distribuye y desarrolla libremente. Es mejor conocido por su equivalente en ingles (Open Source). Se caracteriza por que el código fuente puede ser mejorado y redistribuido.

GNU: La Licencia Pública General de GNU o más conocida por su nombre en inglés *GNU General Public License* o simplemente sus siglas del inglés GNU GPL, es una licencia creada por la Free Software Foundation en 1989 (la primera versión), y está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

Servidor: Computadora que provee servicios a otras denominadas clientes.

Framework: Estructura de soporte definida mediante el cual otro proyecto de software puede ser organizado y desarrollado. Puede incluir soporte de programas, bibliotecas y un lenguaje interpretado para ayudar a la unión y desarrollo de los diferentes componentes de un proyecto.

Servlet: Programa que se ejecuta en un servidor

JSP: Acrónimo de Java Server Pages. Es una tecnología orientada a crear páginas web con programación en Java.

Internetwork: Es la posibilidad de trabajo en redes a través de medios masivos de comunicación.

Apéndice A: Tabla de Nombramientos de la Faculta de Estudios Superiores Cuautitlán

NOMBRAMIENTO	NIVEL	CONTRATO	TIEMPO	CATEGORIA	ABREVIATURA
Ayudante de Profesor	A	Interino	Asignación de horas		Aydte Profr A
Ayudante de Profesor	B	Interino	Asignación de horas		Aydte Profr B
Profesor de Asignatura	A	Definitivo	Asignación de horas		Prof Asig A, d
Profesor de Asignatura	A	Interino	Asignación de horas		Prof Asig A, i
Profesor de Asignatura	B	Definitivo	Asignación de horas		Prof Asig B, d
Profesor de Asignatura	B	Interino	Asignación de horas		Prof Asig B, i
Profesor de Carrera	A	Definitivo	Medio tiempo	Asociado	Prof Carr Asoc A mt, d
Profesor de Carrera	A	Interino	Medio tiempo	Asociado	Prof Carr Asoc A mt, i
Profesor de Carrera	B	Definitivo	Medio tiempo	Asociado	Prof Carr Asoc B mt, d
Profesor de Carrera	B	Interino	Medio tiempo	Asociado	Prof Carr Asoc B mt, i
Profesor de Carrera	C	Definitivo	Medio tiempo	Asociado	Prof Carr Asoc C mt, d
Profesor de Carrera	C	Interino	Medio tiempo	Asociado	Prof Carr Asoc C mt, i
Profesor de Carrera	A	Definitivo	Tiempo completo	Asociado	Prof Carr Asoc A tc, d
Profesor de Carrera	A	Interino	Tiempo completo	Asociado	Prof Carr Asoc A tc, i
Profesor de Carrera	B	Definitivo	Tiempo completo	Asociado	Prof Carr Asoc B tc, d
Profesor de Carrera	B	Interino	Tiempo completo	Asociado	Prof Carr Asoc B tc, i
Profesor de Carrera	C	Definitivo	Tiempo completo	Asociado	Prof Carr Asoc C tc, d
Profesor de Carrera	C	Interino	Tiempo completo	Asociado	Prof Carr Asoc C tc, i
Profesor de Carrera	A	Definitivo	Medio tiempo	Titular	Prof Carr Tit A mt, d
Profesor de Carrera	A	Interino	Medio tiempo	Titular	Prof Carr Tit A mt, i
Profesor de Carrera	B	Definitivo	Medio tiempo	Titular	Prof Carr Tit B mt, d
Profesor de Carrera	B	Interino	Medio tiempo	Titular	Prof Carr Tit B mt, i
Profesor de Carrera	C	Definitivo	Medio tiempo	Titular	Prof Carr Tit C mt, d
Profesor de Carrera	C	Interino	Medio tiempo	Titular	Prof Carr Tit C mt, i
Profesor de Carrera	A	Definitivo	Tiempo completo	Titular	Prof Carr Tit A tc, d
Profesor de Carrera	A	Interino	Tiempo completo	Titular	Prof Carr Tit A tc, i
Profesor de Carrera	B	Definitivo	Tiempo completo	Titular	Prof Carr Tit B tc, d
Profesor de Carrera	B	Interino	Tiempo completo	Titular	Prof Carr Tit B tc, i
Profesor de Carrera	C	Definitivo	Tiempo completo	Titular	Prof Carr Tit C tc, d
Profesor de Carrera	C	Interino	Tiempo completo	Titular	Prof Carr Tit C tc, i
Técnico Académico	A	Definitivo	Medio tiempo	Auxiliar	Tec Acad Aux A mt, d
Técnico Académico	A	Interino	Medio tiempo	Auxiliar	Tec Acad Aux A mt, i
Técnico Académico	B	Definitivo	Medio tiempo	Auxiliar	Tec Acad Aux B mt, d
Técnico Académico	B	Interino	Medio tiempo	Auxiliar	Tec Acad Aux B mt, i
Técnico Académico	C	Definitivo	Medio tiempo	Auxiliar	Tec Acad Aux C mt, d
Técnico Académico	C	Interino	Medio tiempo	Auxiliar	Tec Acad Aux C mt, i
Técnico Académico	A	Definitivo	Tiempo completo	Auxiliar	Tec Acad Aux A tc, d
Técnico Académico	A	Interino	Tiempo completo	Auxiliar	Tec Acad Aux A tc, i

Técnico Académico	B	Definitivo	Tiempo completo	Auxiliar	Tec Acad Aux B tc, d
Técnico Académico	B	Interino	Tiempo completo	Auxiliar	Tec Acad Aux B tc, i
Técnico Académico	C	Definitivo	Tiempo completo	Auxiliar	Tec Acad Aux C tc, d
Técnico Académico	C	Interino	Tiempo completo	Auxiliar	Tec Acad Aux C tc, i
Técnico Académico	A	Definitivo	Medio tiempo	Asociado	Tec Acad Asoc A mt, d
Técnico Académico	A	Interino	Medio tiempo	Asociado	Tec Acad Asoc A mt, i
Técnico Académico	B	Definitivo	Medio tiempo	Asociado	Tec Acad Asoc B mt, d
Técnico Académico	B	Interino	Medio tiempo	Asociado	Tec Acad Asoc B mt, i
Técnico Académico	C	Definitivo	Medio tiempo	Asociado	Tec Acad Asoc C mt, d
Técnico Académico	C	Interino	Medio tiempo	Asociado	Tec Acad Asoc C mt, i
Técnico Académico	A	Definitivo	Tiempo completo	Asociado	Tec Acad Asoc A tc, d
Técnico Académico	A	Interino	Tiempo completo	Asociado	Tec Acad Asoc A tc, i
Técnico Académico	B	Definitivo	Tiempo completo	Asociado	Tec Acad Asoc B tc, d
Técnico Académico	B	Interino	Tiempo completo	Asociado	Tec Acad Asoc B tc, i
Técnico Académico	C	Definitivo	Tiempo completo	Asociado	Tec Acad Asoc C tc, d
Técnico Académico	C	Interino	Tiempo completo	Asociado	Tec Acad Asoc C tc, i
Técnico Académico	A	Definitivo	Medio tiempo	Titular	Tec Acad Tit A mt, d
Técnico Académico	A	Interino	Medio tiempo	Titular	Tec Acad Tit A mt, i
Técnico Académico	B	Definitivo	Medio tiempo	Titular	Tec Acad Tit B mt, d
Técnico Académico	B	Interino	Medio tiempo	Titular	Tec Acad Tit B mt, i
Técnico Académico	C	Definitivo	Medio tiempo	Titular	Tec Acad Tit C mt, d
Técnico Académico	C	Interino	Medio tiempo	Titular	Tec Acad Tit C mt, i
Técnico Académico	A	Definitivo	Tiempo completo	Titular	Tec Acad Tit A tc, d
Técnico Académico	A	Interino	Tiempo completo	Titular	Tec Acad Tit A tc, i
Técnico Académico	B	Definitivo	Tiempo completo	Titular	Tec Acad Tit B tc, d
Técnico Académico	B	Interino	Tiempo completo	Titular	Tec Acad Tit B tc, i
Técnico Académico	C	Definitivo	Tiempo completo	Titular	Tec Acad Tit C tc, d
Técnico Académico	C	Interino	Tiempo completo	Titular	Tec Acad Tit C tc, i
Pago por Honorarios			Asignación de horas		Pago por Honorarios
Profesor Emérito	C		Tiempo completo		Profesor Emérito
Investigador	C	Definitivo	Tiempo completo	Asociado	Inv Asoc C tc, d
Investigador	C	Interino	Tiempo completo	Asociado	Inv Asoc C tc, i
Investigador	B	Definitivo	Tiempo completo	Titular	Inv Tit B tc, d
Investigador	B	Interino	Tiempo completo	Titular	Inv Tit B tc, i
Investigador	C	Definitivo	Tiempo completo	Titular	Inv Tit C tc, d
Investigador	C	Interino	Tiempo completo	Titular	Inv Tit C tc, i
Técnico Académico	C	Definitivo	Tiempo completo	Asociado	Tec Acad Asoc C tc, d

BIBLIOGRAFIA

33. Naur P., Randell B., Software Engineering: A Report on a Conference Sponsored by
the NATO Scienc, 1969.

6. James Martin, Computer Data Base Organization, 2ª edition, Ed. Prentice Hall, 1975.

18. Pressman, R, Ingeniería del Software: Un enfoque práctico, McGraw Hill 1997.

16. Jacobson, I., Booch, G., Rumbaugh J., El Proceso Unificado de Desarrollo de Software, Addison Wesley 2000

17. Sommerville, I., Ingeniería de Software, Pearson Educación, 2002.

38. Korth, Henry F., Silberschatz, Abraham. Fundamentos de bases de datos, 4ª edición,
Ed. McGraw-Hill. España (2002).

8. Silberschatz, Abraham, Henry F. Korth, S. Sudarshan, Fundamentos de Bases de
Datos, 4ª ed., TR: Fernando Sáenz Pérez [et. al], McGraw Hill, España, 2002.

2. Piattini Velthuis, Mario G., [et al.], "Análisis y diseño detallado de aplicaciones Informáticas de gestión: Primera perspectiva de ingeniería del software", México,
Alfaomega Ra-Ma, 2004.

12. Cesar Perez, MySQL para Windows y Linux, Editorial Alfaomega Grupo Editor Ra-Ma, 2004.

3. Laudon Kennet C., Laudon Jane P., "Sistemas de información gerencial", 8ª ed., Prentice Hall, p.8

7. Mario G. Piattini, Esperanza Marcos, Coral Calero, Belén Vela, Tecnología y diseño de
Bases de Datos", México, Alfaomega Ra-Ma, 2007.

10. Peter Rob, Coronel Carlos. Sistemas de Bases de Datos. Diseño, Implementación y
Administración, 5ª edición, Ed. Thomson.

29. Antonio J., Martin Sierra, STRUTS, 2ª edición, Ed. Ra-Ma.

1. Innovación tecnológica
Recuperado: http://ingsw.ccbas.uaa.mx/sct/course/view.php?id=27

4. El ciclo de vida del software
Recuperado: http://www.slideshare.net/juliopari/3-clase-ciclo-de-vida-del-software

5. Desarrollo en espiral
Recuperado: http://es.wikipedia.org/wiki/Desarrollo_en_espiral

9. Bases de datos
Recuperado: http://www.monografias.com/trabajos5/tipbases/tipbases.shtml

11. Modelo relacional
Recuperado: http://es.wikipedia.org/wiki/Modelo_relacional

13. Introducción a los sistemas distribuidos
Recuperado: http://www.augcyl.org/?q=glol-intro-sistemas-distribuidos

14. Cliente – Servidor
Recuperado: http://es.wikipedia.org/wiki/Cliente-servidor#Arquitecturas_multi-capas

15. Multitier Architecture
Recuperado: http://en.wikipedia.org/wiki/Multitier_architecture

19. Dirección general de asuntos del personal académico
Recuperado: http://dgapa.unam.mx/epa/epa.html

20. Modelado de sistemas con UML
Recuperado: http://es.tldp.org/Tutoriales/doc-modelado-sistemas-UML/doc-modelado-
sistemas- uml.pdf

21. Distribución Linux
Recuperado: http://es.wikipedia.org/wiki/Distribuci%C3%B3n_Linux

22. GlassFish
Recuperado: http://es.wikipedia.org/wiki/GlassFish

23. GlassFish community
Recuperado: http://glassfish.java.net/downloads/3.0.1-final.html

24. PostgreSQL
Recuperado: http://www.postgresql.org/

25. Oracle
Recuperado:
http://www.oracle.com/technetwork/java/javase/overview/index.html
26. Struts: MVC en Java

Recuperado: http://www.slideshare.net/siis/struts-en-java

27. Struts tutorial
Recuperado: http://www.roseindia.net/struts/

28. Struts
Recuperado: http://www.slideshare.net/siis/struts-en-java

30. The Apache Software Foundation
Recuperado: http://struts.apache.org/

31. The Apache Software Foundation
Recuperado: http://struts.apache.org/2.x/

32. The Apache Software Foundation
Recuperado: http://struts.apache.org/1.x/

34. Universidad Nacional Autónoma de México.
Recuperado: http://www.cuautitlan.unam.mx/misionyvision.html

35. Bases de datos jerárquicas.
Recuperado: http://es.wikipedia.org/wiki/Base_de_datos_jerárquica

36. Base de datos orientada a objetos.
Recuperado: http://es.wikipedia.org/wiki/Base_de_datos_orientada_a_objetos

37. International Organization for Standardization
Recuperado:
http://www.iso.org/search.html?qt=12207&searchsubmit=Search&sort=rel&type=simple&published=on