UNIVERSIDAD NACIONAL AUTÓNOMA
DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES
ARAGÓN

"SISTEMA INFORMÁTICO INTEGRAL PARA LA ACADEMIA MEXICANA DE DERMATOLOGÍA, A.C."

T E S I S

QUE PARA OBTENER EL TÍULO DE:

INGENIERO EN COMPUTACIÓN

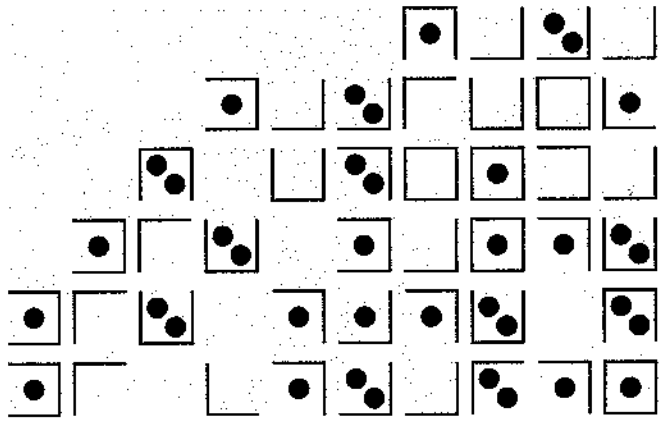
P R E S E N T A:

ALBERTO CÉSAR VILLANUEVA

ASESOR:

ING. SILVIA VEGA MUYTOY

MÉXICO 2005



Autorizo a la Universidad Señorial de Sinaloa de la UNISAL a difundir en Internet electrónico o impreso el contenido de mi trabajo recepcional.

NOMBRE: Alberto Cesar Valenzuela

FECHA: 13/9/05

FIRMA:

Índice

Introducción

Este trabajo trata acerca de los programas de un sistema informático que ha sido creado para mejorar la administración de la información en la Academia Mexicana de Dermatología, A.C. **(AMD).** El objetivo principal de este trabajo es mostrar paso a paso la realización de dicho sistema, el cual conjuga tecnología de la plataforma *.Net* y aplicaciones *Web* para poder satisfacer las necesidades de gestión de la AMD.

Dicho sistema es denominado como integral, ya que cada una de las partes que lo conforman son esenciales para su correcto funcionamiento y que en conjunto han evolucionado en aplicaciones que constituyen un sistema de información que relaciona la funcionalidad y el profesionalismo que la institución posee en la actualidad.

Se ha dividido el estudio de estas aplicaciones en cuatro capítulos para su mejor entendimiento; el primero muestra una visión general de la misión y los objetivos de la AMD, a través de ello se realiza el análisis del sistema basándose en las necesidades informáticas que han dado hincapié al desarrollo de un sistema de información llamado *AcadMexDerm .Net,* el cual es una aplicación de escritorio creada para guardar a través de una base de datos los registros e historiales de los miembros de la AMD.

Para el segundo capítulo se ha elaborado la documentación para el desarrollo del sistema de información antes mencionado. La metodología estructurada utilizada para dicho desarrollo tiende a seguir el estándar *CASE (Computer Aided Systems Engineering o Sistemas de Ingeniería Asistidos por Computadora)* y las técnicas basadas en el desarrollo de aplicaciones orientadas a datos no jerárquicos.

El estándar *CASE* plantea una secuencia de etapas basadas en cinco pasos, la definición y análisis de los requerimientos del usuario, el diseño del sistema y de la base de datos, la implantación y prueba de módulos, la integración y prueba del sistema y por último la operación y mantenimiento del mismo.

Dado que en el primer capítulo se desarrolla el análisis de las necesidades que serán satisfechas por la aplicación, el segundo capítulo comienza con el estudio de la base de datos del sistema de información; posteriormente se desarrollarán los diez módulos en los que se han dividido las principales funciones del sistema, basándose en sus principales capacidades y que han sido creadas desde versiones anteriores que fueron programadas en Visual Basic 6 y que han sido migradas a la plataforma .Net Framework hasta llegar a la sexta y última versión.

El sistema de información que se expone en el segundo capítulo es una migración de las versiones anteriores, se han tomado partes de la metodología orientada a datos no jerárquicos para su desarrollo, la cual divide su estándar en cuatro principales pasos a seguir, la planeación, el análisis, el diseño y la construcción.

El primero se refiere a la construcción de una arquitectura de la información y una estrategia que soporte los objetivos de la organización, el segundo se basa en comprender las áreas del negocio y determinar los requisitos del sistema, el tercero pretende establecer las funciones y comportamiento del sistema deseado y que sea alcanzable por la tecnología empleada; por ello se ha desarrollado un estudio de la plataforma utilizada para el programa destacando los segmentos principales requeridos para el funcionamiento de la aplicación y que han sido utilizados para la programación de la misma.

El último paso de la metodología, la construcción, se refiere a que el programa cumpla con los otros tres niveles antes mencionados; es por ello que gracias a la experiencia obtenida durante el desarrollo de las versiones anteriores se han podido adaptar estas dos metodologías para crear una propia; esta será expuesta a lo largo de este capítulo, en el cual es posible cumplir todos los objetivos requeridos por la AMD para sus sistema de información.

Adicionalmente a este desarrollo se ha incluido un apartado para el programador de nivel intermedio de *Visual Basic .Net* que muestra como se han programado cada una de las principales secciones de dicha aplicación a través de ejemplos prácticos e instrucciones sencillas y explicativas. El objetivo principal de este apartado es mostrar al programador avanzado puntos importantes a considerar para un correcto traslado del lenguaje Visual Basic 6 a la plataforma .Net y porque no es posible realizar una migración de forma transparente.

Se ha separado la parte del desarrollo en este capítulo para que el lector pueda conocer como puede aplicar estos ejemplos en sus propios desarrollos y programar de la mejor manera sus aplicaciones, dándoles un toque de vistosidad sin sacrificar la funcionalidad y al mismo tiempo proporcionarles el profesionalismo que el sistema de información *AcadMexDerm* posee.

El tercer capítulo muestra la estructura de la página de internet de la AMD y la forma en que se han creado sus partes para conformar un sitio profesional y que contiene la información más completa que podrá ser accedida por todas las personas que desean tener un mayor acercamiento a la institución, conocer todos la información de sus próximos eventos y poder contactar a alguno de sus miembros.

El objetivo principal de este capítulo es exponer como se ha distribuido la información más importante de la institución y explicar la estructura funcional de los contenidos distribuidos para que los diferentes usuarios accedan de forma lógica y natural dependiendo de su nivel de experiencia y el tipo de audiencia que tenga el sitio de internet.

El último capítulo trata acerca de una propuesta para un sistema automático de publicaciones dermatológicas creado para que los miembros de la AMD compartan sus ideas y experiencias en un foro abierto que podrá ser consultado y administrado desde sus computadoras personales de forma sencilla y con una interfaz agradable a través de la página de internet.

El objetivo de esta sección es presentar ante los miembros de la mesa directiva de la AMD una propuesta para crear un sistema de publicaciones hecho a partir de los insumos con que cuenta la página de internet sin tener que adquirir ni costear unidades adicionales a lo que ya se tiene, esto con el fin de crear un programa ideado para que los miembros de la institución compartan sus conocimientos para beneficio de sus pacientes y de la comunidad en general.

Capítulo 1

La Academia Mexicana de Dermatología, A.C.

Para este capítulo se proyecta una visión general de la misión de la *Academia Mexicana de Dermatología, A.C.* como una institución dedicada al conocimiento de la dermatología, su comunidad y la creación de una interactividad entre ella para su propia actualización; mediándola y convocándola a congresos y sesiones regulares.

La administración de la información para este propósito resulta ser un servicio indispensable para esta institución, pero para prestar dichos servicios se requieren de varios sistemas informáticos que de manera conjunta den respuesta a la urgente innovación en el modo de realizar las tareas de forma automatizada y eficaz.

En la actualidad se requiere tener una visión amplia acerca del poder de la información; sus características por si sola no dicen mucho, pero si existe una buena logística y se hace un buen uso de la misma se puede competir a cualquier nivel. Con esto se pretende decir que el poder que tiene cada empresa es directamente proporcional a la información con la que cuenta; por más pequeña que sea su magnitud, y dicho parámetro se puede medir desde cierto punto de vista; a veces se cuantifica por el número de empleados, que en el caso de la Academia es muy pequeño, ya que la mayor parte de la administración se realiza en una pequeña oficina conducida por cuatro personas. En cambio, desde otro punto de vista, la AMD cuenta con la mayor *Base de Datos* de dermatólogos a nivel nacional, y esto es gracias en parte al Sistema de Información que se implementó desde hace casi tres años. Y es en este punto que la Academia es fuerte, en cuanto a conocimiento de la gente que la conforma, lo es sumamente importante para la toma de decisiones.

Pero esto es sólo una fracción de todo un sistema de servicios que se necesitan para que la institución funcione de manera deseable. A continuación se muestra que es la AMD, su fundación, su misión, responsabilidades y los diferentes proyectos realizados en el ámbito informático, que en capítulos posteriores, se abordan de manera individual, pero que en forma conjunta crean un Sistema Integral para dar servicio en pos del conocimiento dermatológico.

1.1 Fundación y Misión de la Academia Mexicana de Dermatología, A.C.

Desde la época de nuestros antepasados indígenas, las enfermedades de la piel se conocían y trataban. Entre los *mexicas* o *aztecas* existía el Dios titular de la medicina, y aunque se conocía con distintos nombres, era de suma importancia en la vida diaria de los prehispánicos. Uno de los nombres de esta deidad era ***Xipe-Totec,*** "Nuestro Señor Desollado" ***(figura 1.1),*** Dios de la Primavera, de las Flores y de las Enfermedades de la Piel. Procedía de ***Tzapotlán*** y su manera de venganza era mandar a los hombres enfermedades tales como "el mal de ojo", la sarna y la postema[1].

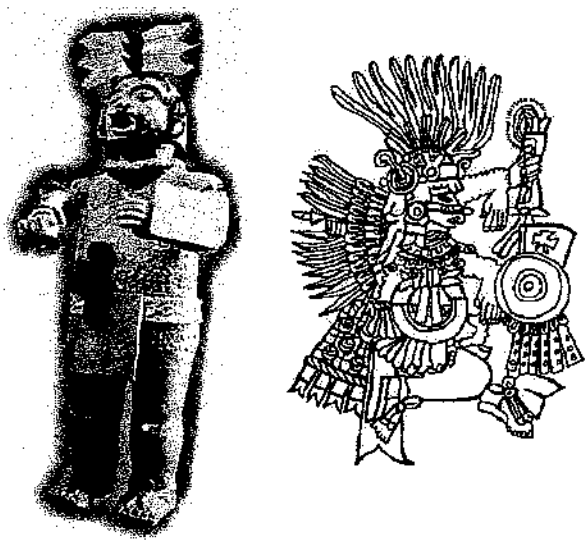


Figura 1.1 → Escultura del Dios Xipe-Totec

Escultura modelada en arcilla, representa el dios de los orfebres, cuyo significado es "Nuestro señor desollado", se trata de un sacerdote que viste la piel de un desollado, el cual luce el tocado característico de la deidad. Procede de Xolalpan, Estado de México. Cultura Teotihuacana de la época clásica (500-650 d.C.).

Desde entonces se han tratado enfermedades de la piel en nuestro país, pasando por la época colonial y hasta nuestros días, se han creado distintas instituciones encargadas de agrupar a los dermatólogos en México.

En el país existen dos grandes agrupaciones dermatológicas, **La Sociedad Mexicana de Dermatología**, fundada en 1936; y **la Academia Mexicana de Dermatología**, fundada el 20 de Junio de 1952 y cuyo primer presidente fue el

1.- Absceso Supurado

doctor *Oswaldo Arias Capetillo*. En la actualidad existen otras pequeñas sociedades como la Sociedad Mexicana de Cirugía Dermatológica y Oncológica, la Sociedad Mexicana de Dermatología Pediátrica, el Colegio Nacional de Dermatología, A.C., y la Sociedad Mexicana de Dermatología. En 1974 surgió el Consejo Mexicano de Dermatología que evalúa y acepta a las personas merecedoras de un título dermatológico.

Sin embargo la AMD es la agrupación más importante en cuanto a dermatología se refiere, ya que sus miembros pasan por una serie de requisitos previos necesarios para poder ser miembro de esta institución, y al final, resultan ser los mejores dermatólogos del país. La Academia Mexicana de Dermatología, A.C. tiene sesiones ordinarias, cursos anuales, congresos y jornadas en provincia con la finalidad de compartir conocimientos, proyectar la dermatología y engrandecerla.

Para tratar de explicar el propósito de la AMD se enuncian las palabras de bienvenida que dio la Ex-Presidenta *Dra. Graciela Guzmán Perera* para todos los asistentes al 50 aniversario de la misma en Oaxaca; las cuales fueron:

"La Academia Mexicana de Dermatología se fundo cuando un grupo de dermatólogos con visión de presente y de futuro creyeron en que la convivencia científica entre colegas de la especialidad era no sólo necesaria, sino imprescindible para darles la oportunidad de participación a todos por igual, sin importar el sitio, la filosofía y la escuela en la que hubieran sido adiestrados, siempre bajo el compromiso de actualizarse, individual, colectiva y permanentemente, incrementando su acervo científico y cultural, compartiendo experiencias que fueran útiles a todos, con el consecuente beneficio que necesariamente iban a recibir sus enfermos".

Dra. Graciela Guzmán Perera

El 20 de Junio de 1952, en el salón de actos del Hospital Concepción Bastegui, ubicado en las calle de Regina, se reunió un grupo de médicos e inquietos dermatólogos, deseosos de compartir y difundir ideas; y animados por un vehemente deseo de democracia que los llevó a construir **la Academia Mexicana de Dermatología.** El objetivo de esta institución era construir una asociación civil cuyo fin sería promover el mejor conocimiento de la dermatología en el país. Estos médicos con intereses compartidos, mutuas inquietudes y con el afán del diario batallar dentro del campo de la especialidad han compartido por más de 50 años logros como la intervención en la enseñanza dermatológica a través de sesiones mensuales clínicas y académicas, congresos y cursos anuales, jornadas a nivel nacional, etc. Todas estas actividades han formado parte importante para el postgrado y certificación en la especialidad para una gran comunidad de médicos que han encontrado en la academia el medio adecuado para compartir sus

experiencias y estar al corriente de los últimos adelantos en cuanto a la investigación se refiere; y cuyos beneficiarios serán los pacientes, quienes recibirán la mejor atención dado que los miembros de esta asociación siempre estarán a la vanguardia en la ciencia dermatológica.

La Academia Mexicana de Dermatología, A.C. es una institución dinámica, abierta al diálogo y al cambio, siempre a la vanguardia, que tiene entre sus objetivos ir al ritmo de los avances promoviendo reuniones interdisciplinarias tanto en el país como en el extranjero; entre las que encontramos las reuniones México-Centroamericanas y Canadienses.

La AMD agrupa a dermatólogos que ejercen a lo largo del territorio nacional y a quienes apoya con programas de becas para que asistan a las reuniones en la capital de nuestro país y a las de los diferentes estados para promover el conocimiento dermatológico.

Para dar cumplimiento medular de su propósito la AMD realiza sesiones mensuales ordinarias, extraordinarias, clínicas, coloquios, jornadas en provincia, congresos nacionales, cursos y sesiones conjuntas con otras sociedades científicas, entre otras actividades.

Para poder promover y desarrollar muchas de las actividades e investigaciones relacionadas con la especialidad hacía falta recursos financieros, por lo que se constituyó una organización que lleva por nombre **"Fundación Mexicana para la Dermatología, A.C."**, la cual promueve y fomenta como parte de la misma asociación, la investigación y la educación médica a la población en general. Gracias a ello se adquirió en propiedad un inmueble con el mobiliario correspondiente para el adecuado desempeño y funcionamiento de la asociación.

La habilidad, el entusiasmo, el coraje y la devoción de los catorce miembros fundadores, así como la constancia de quienes han seguido sus pasos, ha propiciado la integración de más socios a lo largo del país. El éxito de la AMD no ha dependido de la larga lista de miembros, sino de las cualidades científicas y humanas de sus candidatos; agrupando en su momento a los líderes de los diferentes servicios y centros de salud del país en un fin común.

Habiendo comenzado con catorce miembros en 1952, la AMD ahora cuenta con más de 300 miembros activos. Los dermatólogos en México en los albores del siglo XXI son hoy una especialidad médico–quirúrgica bien establecida relacionada con otras ramas de la medicina. La contribución de la AMD durante más de

cincuenta años ha sido el trabajo de equipo con todas las especialidades médicas, lo que asegura un crecimiento continuo tanto de la institución como de la especialidad.

Con orgullo en el pasado y confianza en el futuro, la Academia Mexicana de Dermatología no puede sino continuar siendo una luz para la especialidad que representa.

Los miembros de la AMD son la sociedad más preparada, certificada y especializada en la prevención, diagnóstico y tratamiento de todos los padecimientos de la piel, pelo, uñas y *mucosas* en México.

Esta sociedad ha crecido hasta llegar a ser una importante comunidad de dermatólogos integrada por casi 2000 médicos (miembros o no) especialistas en toda la República Mexicana. La AMD se ha facultado todos estos 52 años para llevar a cabo la integración de información y creación de un medio de comunicación entre los dermatólogos, quienes se benefician al poder estar al día en la materia y al mismo tiempo poder informar de sus investigaciones y experiencias en un foro abierto, pero al mismo tiempo filtrando para tener la certeza que el médico que desee formar parte de esta asociación haya tenido una apropiada educación previa, la cual, lo ha formado como un dermatólogo con ética y conocimientos extensos para poder ejercer con una conducta moral y profesional durante toda su vida. Y sin lugar a dudas los beneficiarios finales de todo este riguroso proceso de selección son los pacientes; los cuales, tendrán la seguridad que recibirán la mejor atención dado que los miembros de la AMD siempre estarán a la vanguardia en la ciencia dermatológica.

La AMD no sólo es un foro para dermatólogos, porque la medicina no sólo se hace de médicos, sino que también se integran medicamentos, químicos, laboratorios, cosméticos, tecnología, aparatos, etc. Todo esto también es tomado en cuenta por la AMD. Todos los laboratorios dedicados a la rama de la dermatología están relacionados y en contacto directo con la academia para poder mostrar sus últimos adelantos y los mejores medicamentos para la atención de la gente. También del mismo modo la comunidad dermatológica, a través de la AMD, hace llegar las necesidades que existen entre los pacientes para la creación de nuevos métodos de tratamiento para procurar la salud de la nación, por lo menos, en padecimientos dermatológicos clínicos y cosméticos.

En resumen la Academia cuenta con una larga experiencia en la organización de gente, laboratorios, tecnología, ciencia, investigación y enseñanza de la dermatología, la cual, está a la altura de cualquier país a nivel mundial.

La academia cuenta entre sus miembros a los mejores profesores, investigadores, especialistas, laboratoristas, académicos, cirujanos, residentes, etc. de todo el país; y de esta forma, ha tenido que organizar toda esta información durante años.

Anteriormente también se ha procurado mantenerse al día en cuestiones administrativas para llevar de la mejor forma sus registros, pero de todo este mundo de información, lo primordial es tener el mejor archivo de sus miembros, los cuales, dan vida a la institución.

1.2 Administración y Eventos Organizados por la AMD

En principio la administración se llevaba a papel, como cualquier empresa pequeña, pero el auge en la AMD se ha dado a pasos agigantados estos últimos años; y para mantenerse en contacto con toda la sociedad médica creciente de esta especialidad, es primordial tener toda la información posible de cada uno de los médicos sin descuidar a ninguno de ellos. Desde hace muchos años se ha mantenido un archivo en papel de los miembros de la institución, pero el archivo incluye también a los dermatólogos que no son miembros, es decir, pasantes o residentes que todavía no tienen una especialidad, pero que sin embargo, siguen ejerciendo la dermatología. A estas personas también es importante tenerlas en contacto, porque no sólo los miembros tienen derecho a obtener la última información referente a su medio de trabajo para desenvolverse y obtener más experiencia y conocimientos, sino que la academia esta abierta no sólo para dermatólogos, sino para cualquier médico que desee aumentar sus conocimientos y experiencias para su desarrollo profesional y personal.

La AMD organiza una serie de eventos continuos, entre ellos organiza sesiones mensuales, jornadas en provincia, cursos anuales y congresos. Todos estos eventos se aprovechan de forma recíproca. Por un lado, los laboratorios dan a conocer sus nuevos productos y también se informan de las nuevas investigaciones que se llevan a cabo en el medio. Los dermatólogos, aparte de aprender más, recaudan puntos. Estos puntos son primordiales para poder ser parte de la academia; cierta cantidad de puntos se obtienen simplemente por asistir, pero en mayor proporción se obtienen por presentar trabajos en las jornadas o en las sesiones mensuales.

La academia exige cumplir con una serie de requisitos previos para poder ser miembro activo, poder ser favorecido por las becas y demás beneficios que esto representa.

Para el ingreso de nuevos miembros numerarios a esta asociación, quedan los requisitos de la siguiente manera:

1. Con un trabajo de ingreso, en extenso y resumen, carta solicitud de ingreso dirigida al Presidente de la AMD, 2 cartas aval de miembros de esta asociación, currículo y certificaciones (título de médico cirujano, diploma de la especialidad y certificado del Consejo Mexicano de Dermatología).

   Presentar lo anterior en 2 carpetas; el resumen y el trabajo de ingreso en una carpeta y en la otra los documentos restantes.

2. El requisito a que se refiere el punto anterior podrá suprimirse si el interesado reúne quince puntos, que se obtendrán de la siguiente manera:

   - Un punto por cada asistencia a las sesiones mensuales de la Academia o a las Sesiones Mensuales de su localidad.
   - Tres puntos por cada caso presentado en Sesión Mensual, Jornada en Provincia o Congreso Bienal.
   - Cuatro puntos por participación en Simposio o trabajo libre presentado en Sesión Mensual, Jornada en Provincia o Congreso Bienal.
   - Tres puntos por asistencia a Jornada en Provincia o Congreso Bienal.

   Para poder participar en el esquema de puntuación antes citado, será necesario que presente a la Mesa Directiva la siguiente documentación:

- Carta solicitud de ingreso.
- Copia fotostática del título profesional, así como de la cédula profesional de especialista.
- Copia fotostática del certificado del Consejo Mexicano de Dermatología o cédula especialista.

En las ***Sesiones Mensuales*** se presentan los trabajos de los nuevos pretendientes a ingresar a la institución, estas sesiones mensuales son sumamente importantes, ya que en ellas se reúnen tanto la mesa directiva como los aspirantes y así, estos últimos pueden mostrar a la audiencia sus trabajos y pueden presentar al mismo tiempo sus currículos y demás cartas de méritos para su aceptación.

En las ***Jornadas en Provincia***, la institución trata de acercar a la academia a las diferentes localidades y la posibilidad de presentar trabajos a la mesa directiva, se exhiben simposios y cursos, se realizan foros y prácticas, etc. en fin, una serie de eventos que, sin lugar a dudas, evitan la centralización de oportunidades para los médicos que residen en provincia.

Tanto los ***Cursos Anuales*** como los ***Congresos*** son eventos magnos. Se realizan tanto en la capital como en provincia; estos eventos son a gran escala. Son para los dermatólogos en general y también para los miembros, todos nacionales y extranjeros. Para los miembros, que pagan una cuota anual para estar inscritos, estos eventos son gratuitos, pero para el público en general, se cobra una cuota para poder tener acceso a los mismos. Esta cuota es en verdad significativa comparada con los beneficios obtenidos por la asistencia a estos eventos.

Estos eventos se realizan, por lo general, en hoteles muy grandes y duran alrededor de 5 días; dichos eventos reflejan la magnitud de la influencia que tiene la AMD en la dermatología Mexicana y en la medicina en general, ya que asiste una cantidad considerable de gente, incluyendo muchas eminencias en la rama tales como profesores extranjeros que enriquecen el evento haciéndolos de nivel internacional.

En fin, la AMD es la asociación más prestigiada, en lo que a dermatología se refiere, en México; y es reconocida a nivel Latinoamericano y Mundial. El público en general puede tener la confianza de que, al asistir con un dermatólogo miembro de la AMD, está asistiendo con un médico certificado, especialista y sobre todo actualizado que es equiparable a cualquiera en el mundo.

1.3 Historia del Registro de Miembros

El registro de los miembros, y toda la información referente a la AMD se lleva a cabo en su oficina; inmueble propio ubicado en la dirección Georgia #114 - Despacho 503, Col. Nápoles en la Delegación Benito Juárez.

Una de las partes más importantes en la administración de la AMD es el registro de los miembros y de los que no lo son todavía; este archivo se registraba desde el comienzo en papel. En el año de 1999 se adquirieron 2 equipos de cómputo *(Pentium II)*, con los cuales se pudo llevar un control de los miembros y sus datos a través de tablas de *Microsoft Excel*. Estos registros eran meramente un intento de llevar un control más eficiente, pero el registro de los mismos no era muy eficiente, ya que el estar cambiando de programas para realizar las diferentes tareas con dicha información era complicado y desorganizado. No había un control de actualizaciones y mucho menos un respaldo de la información.

En ese entonces se registraron alrededor de 200 dermatólogos entre documentos y hojas de cálculo; también se contaba con un listado en papel que sumaban un total de casi 500 médicos, entre miembros y demás dermatólogos. Los diferentes registros diferían entre sí y no se podían integrar todos en un mismo libro ya que

muchos de éstos no contaban con todos los datos necesarios para su utilización; de hecho sólo los registrados en papel contenían la información mínima para poder ser útiles para los propósitos de la AMD.

La utilidad de los registros es muy amplia, se debe tener un mínimo de datos de los miembros para realizar actividades cotidianas como: registrar asistencia a los eventos, editar diplomas para expositores y asistentes, crear un historial de asistencias, enviar folletos y catálogos, etc. Algunos de los datos indispensables para estas actividades son:

- Nombre(s) y Apellidos.
- Dirección Completa (Incluyendo Ciudades, Municipios, Delegaciones, Calles, Colonias y en especial Código Postal).
- Teléfonos de Casa, Consultorios, Celulares, Beepers, Faxes, etc.
- Especialidades (Cirujanos, Micólogos, Residentes, Oncología, Cirugía Plástica, Internista, etc.).

A través de este conjunto de datos es posible llevar a cabo la administración de la gente en la AMD. Pero el llevar un enredado registro de las personas causaba muchos errores en la producción de servicios en la academia. Uno de los principales problemas era la duplicidad en los datos.

Los dermatólogos tenían hasta entonces tres medios de inscripción a los eventos de la academia; estos son relativamente simples, lo primero es tener conocimiento de algún evento que realiza la AMD. Si se es miembro activo, y obviamente se está al día en las cuotas, dicho evento es gratuito. Para eventos especiales como Cursos Anuales se anteceden diversos programas como *precurso* u *otros talleres*, que tienen un costo adicional para los miembros; pero para los dermatólogos externos a esta asociación, el evento en su totalidad tiene un costo significativo.

Ya que se ha tomado la decisión de asistir al evento (al cual no necesariamente hay que asistir a todos los talleres y cursos; siendo posible asistir a los que les causa más interés o simplemente que les atañe en sus especialidades) se procede a realizar el pago de la cuota. Después es posible realizar la inscripción de tres diferentes formas ***(figura 1.2)***.

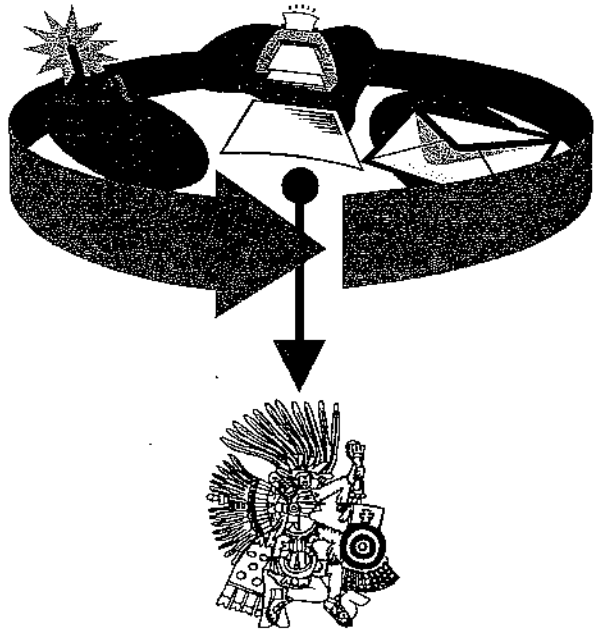


Figura 1.2 → Formas de Inscripción a algún evento de la Academia

La academia recibía únicamente de estas tres formas las inscripciones (Fax, Teléfono y Correo Postal); el problema de duplicidad en los datos de inscripciones se presentaba porque existen laboratorios que becan a algunos médicos para que asistan a los eventos de la AMD. Estos laboratorios se dedican exclusivamente a medicamentos dermatológicos, algunos de ellos son:

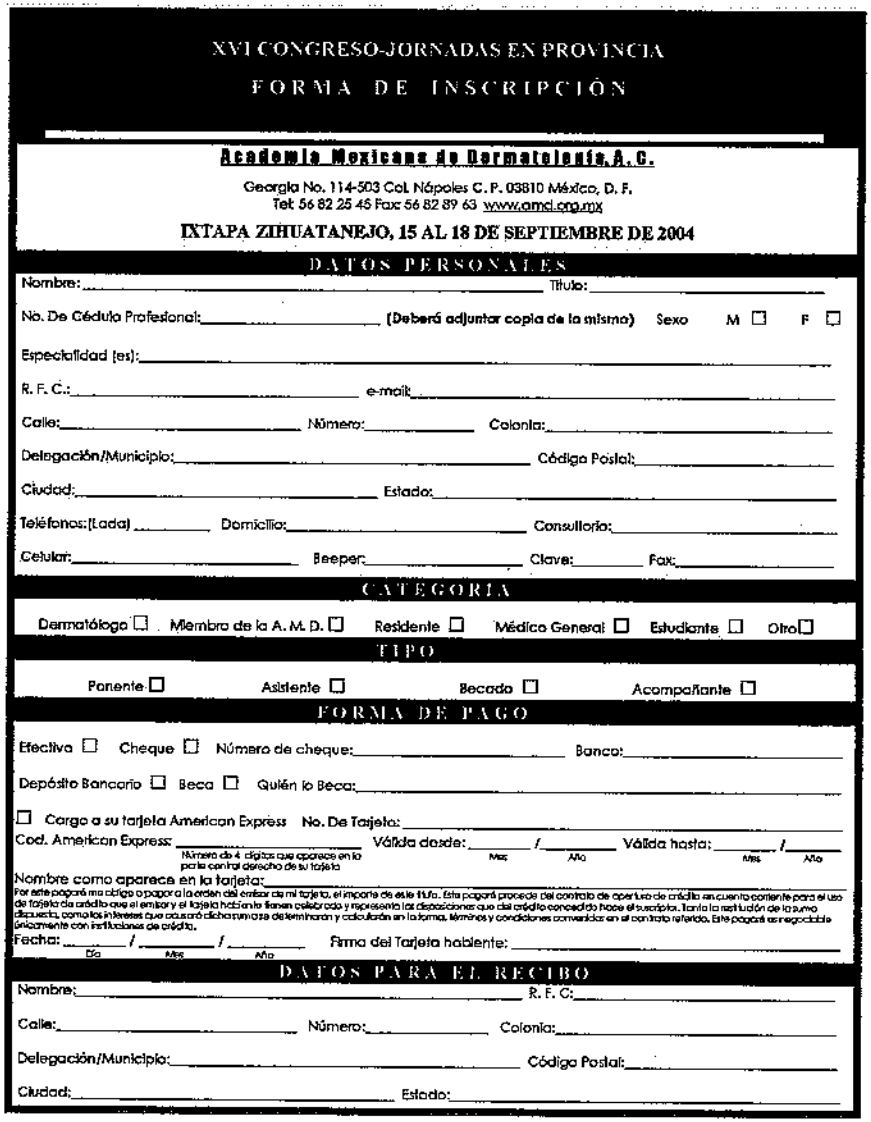
- Centro Internacional de Cosmiatría, S.A. de C.V.
- Galderma México, S.A. de C.V.
- Siegfried Rhein, S.A. de C.V.
- 3M México, S.A. de C.V.
- Frabel, S.A. de C.V.
- Boehringer Ingelheim Promeco, S.A. de C.V.
- Pierre Fabre México, S. A. de C. V.
- Novartis Farmacéutica, S. A. de C. V.

Estos laboratorios, entre otros, becan a algunos médicos para asistir a los eventos de la AMD; pero cada laboratorio enviaba un listado de las personas que asistirían al evento becados por este mismo y el pago obviamente es absorbido por el laboratorio.

Cuando los laboratorios mandan los listados de los becados, mandan también los datos de cada uno de ellos; la Academia recibe la información necesaria para realizar las inscripciones de todos, pero al mismo tiempo ocurría que, al ser miembros de la AMD, no se cobra por asistir a los eventos, y es entonces que desde las oficinas de la AMD se pedía una confirmación de asistencia a los consultorios de los médicos, y regularmente para éste tipo de transacciones administrativas sólo se tiene contacto con las secretarias de los dermatólogos, de tal manera, que ellas mismas no tienen conocimiento anterior de que algún laboratorio se había encargado de becarlo y es entonces que se presentaba la duplicación de inscripciones; sólo por mencionar un ejemplo.

Además de esto no sólo se hacía el registro de asistencia, sino que se ingresaban los datos del dermatólogo a los archivos de la academia. Los dermatólogos de manera individual llenan un formato de inscripción que reciben por correo postal y es enviado junto con su ficha de pago para ingresar a algún evento y tener al día sus datos.

La forma de inscripción se enviaba por correo postal a todos los dermatólogos registrados, fueran miembros o no; y una vez que se llenaba se enviaba de regreso a la academia por correo o por fax ***(figura 1.3)***.



XVI CONGRESO-JORNADAS EN PROVINCIA

FORMA DE INSCRIPCIÓN

Academia Mexicana de Dermatología, A. C.

Georgia No. 114-503 Col. Nápoles C. P. 03810 México, D. F.
Tel: 56 82 25 45 Fax: 56 82 89 63 www.amcl.org.mx

IXTAPA ZIHUATANEJO, 15 AL 18 DE SEPTIEMBRE DE 2004

DATOS PERSONALES

Nombre: _____ Título: _____

No. De Cédula Profesional: _____ **(Deberá adjuntar copia de la misma)** Sexo M ☐ F ☐

Especialidad (es): _____

R. F. C.: _____ e-mail: _____

Calle: _____ Número: _____ Colonia: _____

Delegación/Municipio: _____ Código Postal: _____

Ciudad: _____ Estado: _____

Teléfonos: (Lada) _____ Domicilio: _____ Consultorio: _____

Celular: _____ Beeper: _____ Clave: _____ Fax: _____

CATEGORIA

Dermatólogo ☐ Miembro de la A. M. D. ☐ Residente ☐ Médico General ☐ Estudiante ☐ Otro ☐

TIPO

Ponente ☐ Asistente ☐ Becado ☐ Acompañante ☐

FORMA DE PAGO

Efectivo ☐ Cheque ☐ Número de cheque: _____ Banco: _____

Depósito Bancario ☐ Beca ☐ Quién lo Beca: _____

☐ Cargo a su tarjeta American Express No. De Tarjeta: _____

Cod. American Express: _____ (Número de 4 dígitos que aparece en lo parte central derecho de su tarjeta) Válida desde: ___/___ (Mes / Año) Válida hasta: ___/___ (Mes / Año)

Nombre como aparece en la tarjeta: _____

Por este pagaré me obligo a pagar a la orden del emisor de mi tarjeta, el importe de este título. Este pagaré procede del contrato de apertura de crédito en cuenta corriente para el uso de tarjeta de crédito que el emisor y el tarjeta habiente tienen celebrado y representa las disposiciones que del crédito concedido hace el suscriptor. Tanto la restitución de la suma dispuesta, como los intereses que causará dicha suma se determinarán y calcularán en la forma, términos y condiciones convenidos en el contrato referido. Este pagaré es negociable únicamente con instituciones de crédito.

Fecha: ___/___/___ (Día / Mes / Año) Firma del Tarjeta habiente: _____

DATOS PARA EL RECIBO

Nombre: _____ R. F. C: _____

Calle: _____ Número: _____ Colonia: _____

Delegación/Municipio: _____ Código Postal: _____

Ciudad: _____ Estado: _____

Figura 1.3 → Forma de Inscripción del XVI Congreso-Jornadas en Provincia que se llevo a cabo del 15 al 18 de Septiembre del 2004.

Al no existir un archivo único de registro tanto de datos personales de todos los dermatólogos así como de sus asistencias a los eventos se presentó la necesidad de crear una Base de Datos que tuviera las siguientes capacidades mínimas de almacenamiento:

1. Poder registrar los datos personales de todos los dermatólogos, miembros o no, en un archivo único y estandarizado para todos. ***(Esto se refiere a que si son miembros o no, se describa que no todos los dermatólogos registrados en los archivos son miembros activos de la academia).***
2. Ser capaz de evitar la duplicidad en el proceso de registros.
3. Poder acceder de manera eficaz a los datos de los médicos registrados.
4. Realizar búsquedas por diferentes parámetros como:

    - Especialidad
    - Localidad (Estado o Delegación)
    - Miembros de la Academia, Academia–Fundación u Otro (que aplica para NO miembros)

5. Llevar un registro o historial de asistencia a los eventos.

1.4 Fundación Mexicana para la Dermatología, A.C.

No se pretende entrar muy a fondo al funcionamiento ni a la estructura de la Fundación como parte de los proyectos que atañen a la Academia, pero si es indispensable dar una breve explicación de su relación con la AMD.

La Fundación es una institución que trabaja de manera conjunta a la Academia para lograr un solo objetivo, y de manera sencilla se podría decir que la Fundación Mexicana para la Dermatología, A.C. lleva la administración financiera de la Academia; fuera de esto, la meta individual de la Fundación es fortalecer la información que llega al público en general mediante el impulso a la investigación, la formación de recursos humanos de alto nivel y el desarrollo tecnológico y de avalar e identificar programas y productos de excelencia académica y de trascendencia social.

Pero al llevar una relación tan estrecha, es importante resaltar que al ser miembro de la Fundación, automáticamente se es miembro de la AMD.

La Academia tiene tres clasificaciones para los dermatólogos que registra en el sistema:

1. Miembros de la Academia Mexicana de Dermatología, A.C. ***(Miembros AMD)***
2. Miembros de la Fundación Mexicana para la Dermatología, A.C. *(Automáticamente son Miembros de la Academia)* ***[Miembros FMD-AMD]***
3. Los que NO son Miembros de Ninguna de las Dos ***(Otros)***

1.5 Contacto a través del Correo Postal

La AMD mantiene un contacto continuo con sus miembros, y también con los que no lo son todavía, a través de un medio sumamente utilizado por la misma para poder anunciar sus eventos, publicaciones y demás entregas impresas. Este medio es el ***correo postal***.

El correo es muy utilizado por la AMD aún en la actualidad como medio principal de difusión de información a través de su comunidad a nivel nacional. Este medio es sumamente práctico para enviar folletos y programas impresos que inducen al lector, por medio de fotos e imágenes, a interesarse por un cierto evento, curso, simposio, fármaco, etc.

En el pasado para poder enviar información para todas las personas registradas, se tenían que hacer etiquetas a máquina de escribir para cada una de las cartas. Posteriormente se hizo un formato en un Documento de *Microsoft Word* para poder utilizar etiquetas Standard tamaño Carta (*Avery 5160*) para crear un tiraje de etiquetas ***(figura 1.4)***.

Figura 1.4 → Tirajes de Etiquetas

Dicho formato de *Word* resulta ineficiente, ya que hay que transcribir una a una las etiquetas con los datos de los destinatarios; al final se tenían cierta cantidad de Documentos de Word que tenían que ser actualizados a mano en el caso de que los datos de algún Dermatólogo cambiaran; además se iban insertando nuevos registros de manera desordenada, es decir, como se fueran recibiendo; esto causaba un problema, ya que al llevar cierta cantidad de sobres a las oficinas postales, el personal de la misma exigía que se ordenaran de acuerdo al código postal de las diferentes zonas de la República.

Es entonces que se requirió un sistema el cual automatizara la impresión de estas etiquetas y se cuestionó entonces el poder contar con un sistema que integrara el uso de los datos de los médicos ya registrados en una Base de Datos y utilizarlos para la impresión organizada y actualizada de etiquetas para enviar correos postales masivos cuando se requiriera sin necesidad de cambiar uno por uno los datos.

1.6 Internet como una Solución

Con el auge que ha tenido Internet en los últimos años, se ha incrementado el uso de las páginas de *Internet* y del *Correo Electrónico* como medio casi imprescindible de comunicación entre la gente; y los dermatólogos no son la excepción; desde tiempo atrás había surgido en la Academia la inquietud de que la AMD contara con estos servicios.

Para la época en que vivimos, por el modo de vida con tecnología avanzada, resulta engorroso enviar documentos por fax para algunos de estos médicos, y no sólo de envío sino de recepción de documentos, y es que mientras más pasa el tiempo, crece la cantidad de información que se maneja en cualquier oficina, y en la AMD no se contaba ni con el servicio de Internet y mucho menos del correo electrónico.

El intercambio de documentos y de información entre los dermatólogos y la Academia crecía, así como los laboratorios requerían tener un mayor contacto con la Academia, y viceversa. La demanda en la velocidad de intercambio de información se acrecentaba y la Academia no se podía quedar atrás. Es por esto que se tomó la decisión de contar con el servicio de internet y de correo electrónico; pero con los equipos con que se contaba (*dos equipos Pentium II*) no se podrían cumplir los proyectos que comenzaban a tomar forma en la Institución.

Es por todo esto que la AMD tomo la decisión de crear un departamento de *"Sistemas"*, esto con el fin de poder dar solución inmediata a los proyectos que requerían atención inmediata. Entre las soluciones esperadas está el poder contar con los servicios de internet y correo electrónico para intercambiar documentos electrónicos de manera rápida y eficiente; pero antes que nada se tenía que contar con una infraestructura mínima que soportara los requerimientos mínimos de equipos de cómputo, interconexión y comunicación para los proyectos.

1.7 Centro de Cómputo y RED Interna

Para comenzar se realizó la adquisición de tres equipos para el Centro de Cómputo; Dos equipos *Intel Celeron* y un *Pentium IV* que fungiría como servidor de *Proxy* para Internet. Es posible ver en la ***Tabla 1.1*** los servicios que se pretenden cubrir por equipo:

	Servicios Principales	Características Principales	Dispositivos Adicionales
Equipo Uno (PC 1)	• Proxy Server. • Administración Principal de Información de Entrada y Salida de la AMD • Comunicación a través del Correo Electrónico de la AMD con el Exterior • La mayoría de la documentación de la Academia se realiza en este Equipo	• Intel Pentium IV • Dirección IP Fija para Red Interna • Conexión Principal a Internet por Servicio ADSL 512 kbps. • Dirección IP Dinámica para Red Externa Internet	• Impresora de Inyección • Modem Ruteador Externo (Internet)
Equipo Dos (PC 2)	• Base de Datos • Servicios de Fax • Servicios de Impresión Principal	• Intel Celeron • Dirección IP Fija para Red Interna	• Impresora Láser • Fax
Equipo Tres (PC 3)	• Administración de la Fundación Mexicana para la Dermatología, A.C.	• Intel Celeron • Dirección IP Fija para red Interna	• Impresora Inyección. • Unidad ZIP

Tabla 1.1 → Servicios y Características para los Equipos en Centro de Cómputo

No son necesarios más equipos en el Centro de Cómputo ya que sólo hay tres personas laborando en ellos, pero para el funcionamiento del mismo, se interconectaron a través de un *Switch* para crear una RED LAN pequeña pero funcional en el aspecto de que los tres equipos cuentan con la salida a Internet, y para esto la puerta de enlace virtual es el equipo más rápido, es decir, el Pentium IV.

Además, la Administración de la Base de Datos esta a cargo de una sola persona, la cual, es la única con privilegios de administración; pero eso se explicará en el siguiente capítulo; pero para comprender de una manera más visual y clara los equipos con que se cuenta y los servicios que prestan, se puede observar la ***figura 1.5***.

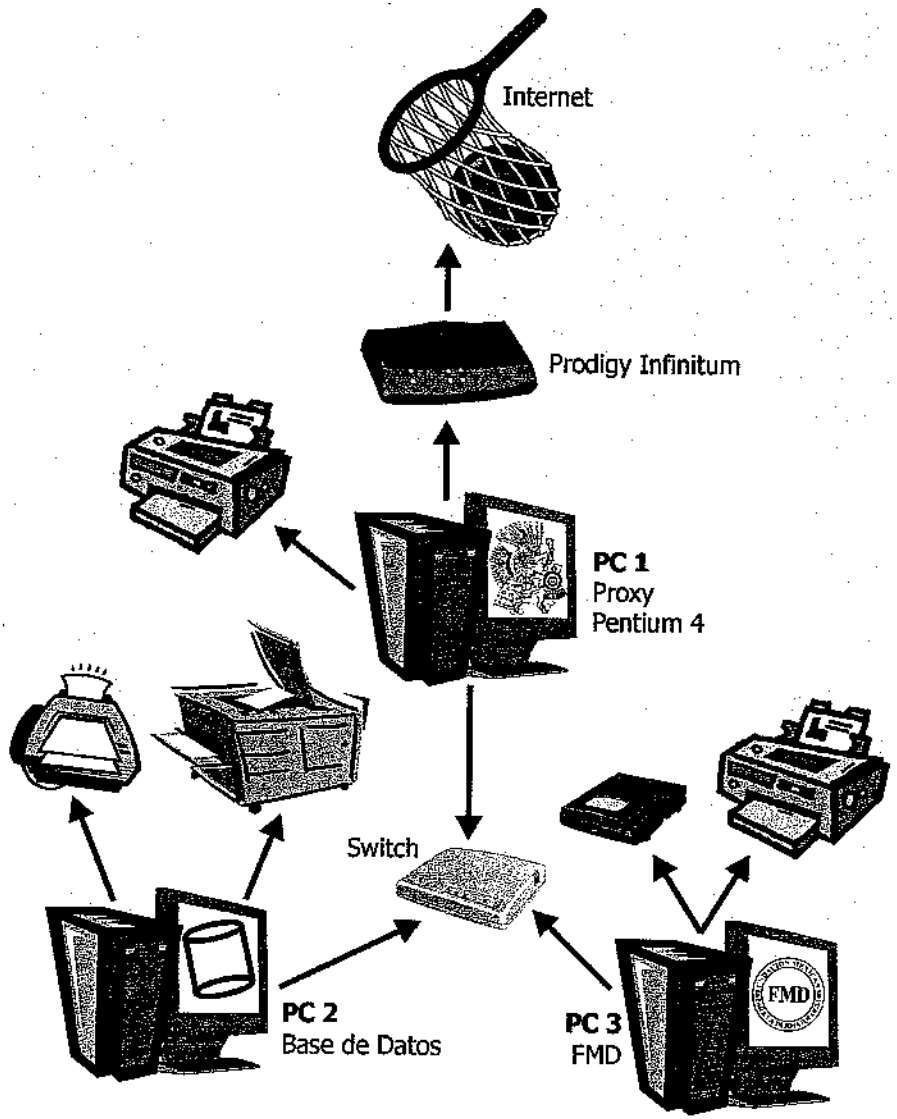


Figura 1.5 → Esquema del Centro de Cómputo de la AMD

1.8 Base de Datos de Miembros

Hablando de un archivo histórico único como solución indispensable de registros para el control de los miembros, surgió la necesidad de contar con una Base de Datos que fuera administrada a través de un ***sistema de información*** único que tuviera la capacidad de ***integrar*** diversas soluciones en un mismo programa, estas soluciones llegarían para satisfacer las dificultades que presenta el no contar con un sistema único de almacenamiento de información de los miembros de la AMD; entre otras cosas, las funciones más importantes a realizar por el sistema debían ser desarrollar un programa cuya infraestructura esté basada en *Bases de Datos*, la cual será capaz de:

- ✓ Organizar y aprovechar sus datos para la mejor toma de decisiones.
- ✓ El acceso a la información debe ser rápido, confiable y que esté disponible en cualquier momento.
- ✓ Que la interfaz de Usuario sea cordial, sencilla y accesible.
- ✓ Evitar Duplicidad en los miembros.
- ✓ Crear tirajes de etiquetas de correo postal.
- ✓ Poder enviar correos electrónicos también de manera masiva.
- ✓ Contener un historial personalizado para cada dermatólogo acerca de su asistencia a los distintos eventos de la AMD.
- ✓ Tener un generador de consultas dinámicas que afecten a todas las demás funciones integradas.
- ✓ Crear reportes personales y de conjunto para su organización y almacenamiento.

Se contaba hasta el momento con la información necesaria para crear una Base de Datos sencilla en cuanto a su tamaño, pero lo suficientemente compleja como para contener información trascendental de cada uno de los Miembros de la Academia.

El programa debe ser un sistema único de escritorio con una interfaz cordial y bajo la plataforma Windows, ya que en los equipos se contaba ya con dicho sistema operativo precargado desde su adquisición, y es por esto que en el capítulo dos profundizaremos en como se realizó dicho Sistema de Información y el impacto inmediato que tuvo dentro de la AMD.

1.9 Contacto a través de una Página de Internet

Toda empresa que mantenga un contacto directo con sus consumidores requiere actualmente contar con un servicio que en los últimos años se ha vuelto indispensable, una página de Internet. Anteriormente ya se tenía la inquietud de contar con una página dentro de la red mundial, y en efecto, se hizo el primer intento de tener una; antes de contar con el Departamento de Sistemas se adquirió el dominio **http://www.amd.org.mx** a través de una empresa llamada *Mexis*. Pero no se tuvo un orden tanto en la administración como en la realización del proyecto.

Al no haber una planeación previa del diseño de la página y los servicios que esta prestaría a la comunidad de la AMD, el proyecto fue bastante pobre en muchos aspectos, y uno de los principales es que no se pensó en el costo del hospedaje y los servicios que se daban a través de ella.

El dominio estaba adquirido; el hospedaje era caro e ineficiente, de hecho, no contaban con la información de los servicios que prestaba la empresa dentro del paquete de hospedaje, la página era pobre en cuanto a su contenido, en resumen, un desastre por la falta de planeación.

Entonces se realizó la planeación de un proyecto que se integrara a la solución informática que daría los servicios mínimos de una página de Internet; los puntos principales a cubrir eran los siguientes:

- ✓ Reducir los costos de hospedaje y administración
- ✓ Tener conocimiento completo de los servicios de hospedaje y de los derechos del dominio
- ✓ Incluir en la página información mínima indispensable como:
  - Información de la Academia y sus Rubros
  - Mesa Directiva de la AMD
  - Información de Eventos de la AMD
  - Agenda de Eventos
  - Información de Inscripciones a Eventos
  - Requisitos para ser Miembro de la AMD
  - Enlaces a otros sitios relacionados directamente a la AMD
  - Publicidad de Laboratorios
- ✓ Crear un servicio para el público en general que beneficie directamente a los Miembros de la Academia y de la Fundación.

1.10 Publicaciones Dermatológicas

La mayoría de los dermatólogos miembros de la AMD tienen en su currículum un amplio historial de publicaciones que van desde artículos científicos en revistas hasta libros completos.

Actualmente en la mayoría de las empresas se cuenta por lo menos con un sistema de publicación interna para dar a conocer un amplio rango de documentos, que van desde reglas internas de la empresa, como artículos de actualización y todo lo referente con la compañía. Esto existe desde hace mucho tiempo, la Universidad Nacional Autónoma de México tiene, por ejemplo, la *Gaceta Universitaria;* las empresas privadas también cuentan con *Intranets* en las cuales, cualquier persona que tenga acceso a una máquina dentro de la red de la empresa puede tener acceso a dicha información.

Al principio los encargados de realizar las tareas de publicación eran los programadores y administradores de los sitios de Internet o *Intranet*, pero actualmente hay un gran número de programas que se encargan de esta tarea de manera automática.

Uno de los trabajos principales del personal encargado del mantenimiento de los sitios de Internet es el de mantener actualizados y corregidos los contenidos de los diversos sitios de la red; sin embargo, muchas veces el delegar esta labor al usuario resulta muy útil, ya que la responsabilidad recae sobre el autor y no en el programador, esto sin contar que para el desarrollador es mucho más útil dedicar su tiempo a otras actividades si es el autor es el encargado de mantener y corregir lo que desea publicar en el sitio.

El punto es que el usuario no esta obligado de ninguna manera a saber programar ni hacer páginas de Internet, entonces ¿Cómo hacer esta tarea?; la respuesta es muy sencilla, ¿Por qué no hablar en un mismo idioma?; si el usuario esta en contacto directo con una computadora, lo menos que debe saber es como hacer un documento en un procesador de texto, y si sabe guardar un archivo de *Microsoft Word*, por ejemplo, y también tiene conocimiento de como navegar en Internet, entonces la solución es crear una interfaz en la que el mismo usuario sea el que realice la publicación desde su escritorio a través de un navegador.

Muchas empresas han lanzando al mercado diversas plataformas que funcionan como servidores de documentación a través de Internet. *Microsoft* posee el servidor *Microsoft SharePoint*, el cual, realiza esta labor de manera satisfactoria.

Para la AMD es importante la difusión de información, pero no posee una intranet a la cual puedan acceder todos los dermatólogos miembros de la academia para poder realizar sus artículos y para compartir sus conocimientos con los demás. Pero con lo que si cuenta es una página de Internet. Y es por esto, que la última parte de este trabajo se enfoca en dar solución a esta inquietud de contar con una Biblioteca Virtual de Publicaciones Dermatológicas que funcione con los recursos con que se cuenta de manera eficaz y completa. Posteriormente se abordará con más detalle este tema.

1.11 Integración de Soluciones

Para poder dar una solución informática integral, se debe contar con un sistema que interconecte sus partes como un todo, es decir, que divida las funciones y que integre las soluciones con un mismo fin, esto es lo que hacen todas las partes del sistema, tomando como pilar, las Bases de Datos, que son el medio actual más popular para el almacenamiento masivo de datos.

Al tener un sistema de información de escritorio con el cual se pueda administrar la entrada de información referente a los miembros de la AMD no es necesario acceder de manera directa a la base de datos, pero este sistema no sólo administra las entradas o salidas, crea reportes, envía correo electrónico masivo, crea tirajes de etiquetas de correo y muchas otras cosas más, eso sin contar las innumerables validaciones de datos.

El sistema también registra de manera eficaz los historiales de asistencia a eventos de los miembros de la AMD, y los registra en la Base de Datos. Y es esta misma base que se exporta a diversas instancias que se utilizan tanto en la página de Internet como en la información que se comparte con diversos laboratorios que dan un importante ingreso a la AMD por tener derecho a dicha información.

En la ***figura 1.6*** se puede apreciar como se integrarán todas las partes en un sistema único.

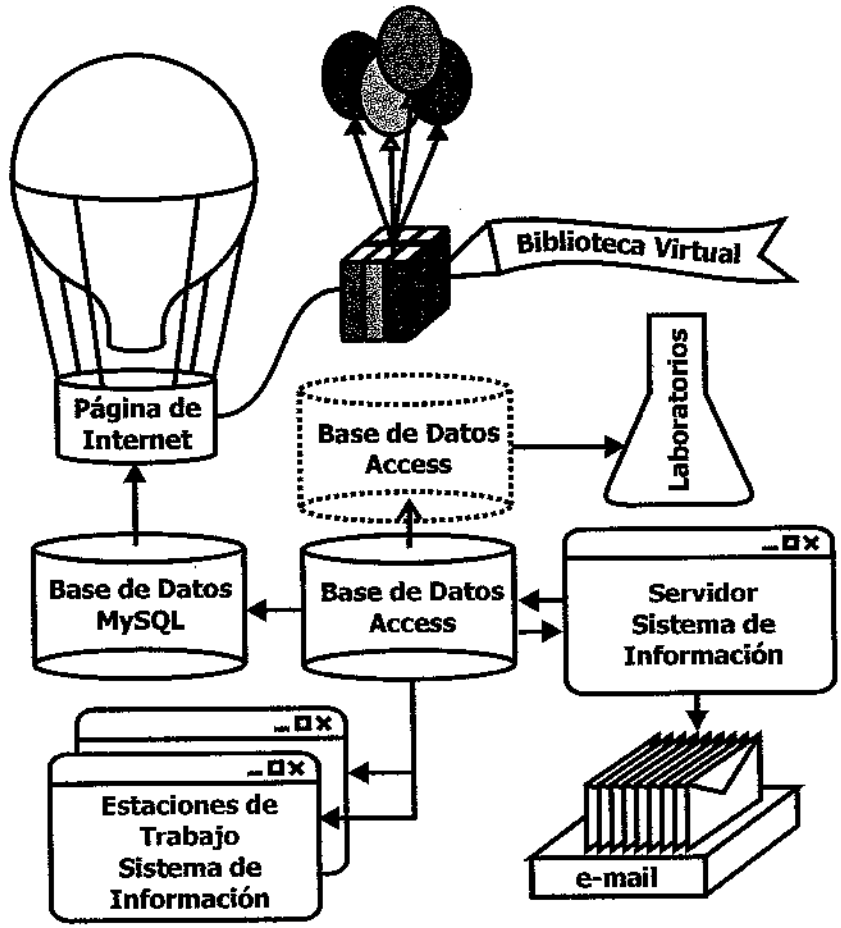
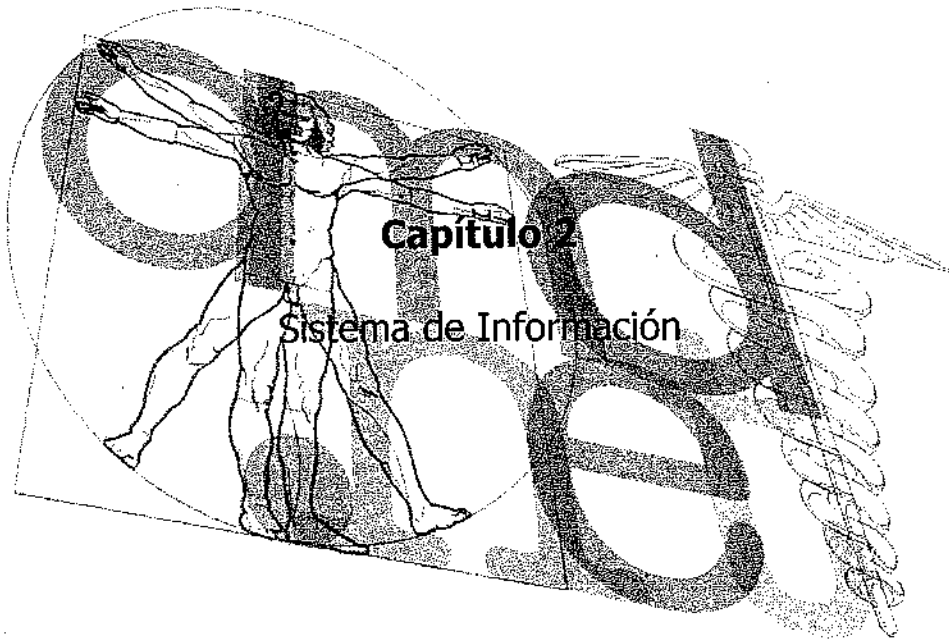


Figura 1.6 → Integración de todas la partes como una solución

Aunque el sistema informático sea la sección más amplia y fundamental, no implica que todo lo demás sea menos importante; al contrario, todas las secciones funcionan como un motor que no podría funcionar si algún engrane no está, aunque éste sea muy pequeño.

En los siguientes capítulos hablaremos a fondo de cada una de las secciones; el Sistema Informático, la Página de Internet y la Biblioteca Virtual, que hasta el momento, se maneja como una propuesta viable a futuro de una implementación más a un sistema que ha evolucionado por más de dos años hasta ser un sistema informático integral que se complementa entre sus secciones para retroalimentarse y complementarse para dar una solución informática completa.



Capítulo 2

Sistema de Información

Desde el mes de Abril del año 2001 se ha desarrollado un programa que en su primera versión no podría ser llamado sistema de información, pero el crecimiento del mismo ha creado una aplicación que resulta independiente de otras y que al estar cimentada en las bases de datos ha madurado y crecido a tal grado que es una aplicación que hace uso de las mismas para no sólo guardar y mostrar información acerca de los dermatólogos registrados, sino que hace de esta información un objeto con el cual se pueden hacer un sin número de cosas muy útiles y hasta cierto punto complejas.

Desde el inicio la necesidad de un programa que administrara los datos de los médicos registrados fue plasmada por primera vez en una versión que utilizaba archivos secuenciales para guardarlos. La perspectiva de la Mesa Directiva de la AMD al tener contacto por primera vez con el programa fue el ser atraídos por la idea de tener un sistema único que pudiera conjuntar una serie de capacidades integradas para guardar dicha información, evitando las duplicidades durante el proceso de registro; pero eso no era todo, deseaban poder hacer más con esos datos que sólo tenerlos guardados.

Para cualquier empresa respetable, los datos resultan ser como el dinero, no sirve de nada si no genera ganancias; en cambio, al realizar inversiones y crear una retroalimentación en el mercado, podría producir una ganancia considerable. Para el caso, lo que se necesitaba era poder tener un mayor contacto con la comunidad y poder aprovechar el poseer un registro tan extenso de médicos especialistas.

Un Sistema de Información puede definirse técnicamente como un conjunto de componentes interrelacionados que permiten capturar, procesar, almacenar y distribuir la información para apoyar la toma de decisiones. La información por si sola no nos dice nada, pero al ser procesada de alguna manera las posibilidades de su uso crecen exponencialmente, y es por esto que el programa puede ser considerado por completo un sistema de información robusto en cuanto a sus capacidades individuales.

En este capítulo se presenta un esquema general del sistema de información llamado ***AcadMexDerm*** en su última versión (sexta); también se dará un bosquejo de como es que organiza de forma central los datos para ser exportados a las otras entidades del sistema integral, como la página de Internet.

También se expondrán de manera separada instrucciones de programación y migración para algunas secciones en específico, las cuales, en base a la experiencia que se ha obtenido durante el desarrollo del mismo, se ha podido comprobar que son sumamente útiles en el desarrollo bajo plataforma *.Net Framework*. Para este propósito se a separado una sección en especial, la cual se ha nombrado **"Apartado para el Programador Avanzado de Visual Basic .Net"**, en el cual se podrá estudiar como es que se han programado los módulos más importantes del sistema.

Los módulos en que se compone el programa son unidades específicas que ejemplifican de forma concreta la resolución de un problema en particular, pero que en general, se podrían aplicar como objetos que se integran a una solución empresarial compleja, es decir, de forma individual resuelven una serie de complicaciones que se presentarían en cualquier sistema corporativo de almacenamiento de datos, pero que de una manera sencilla pueden ser controlados profesionalmente a través de estas aplicaciones.

Sin duda alguna es imperativo destacar la importancia que ha tenido la selección del lenguaje de programación para el proyecto del sistema de información, se ha detallado también un simple pero minucioso estudio de la plataforma ***.Net Framework*** como plataforma de desarrollo; ésta se podría definir básicamente como la nueva e innovadora plataforma que *Microsoft* lanza al mercado como una herramienta indispensable para el desarrollo que vale la pena observar con más detalle, ya que presenta las armas necesarias para afrontar casi cualquier problema, y de la misma manera, solucionarla aplicando un nuevo método de programación que se centra en la distribución de servicios y no de programas.

Se comienza con un análisis a profundidad de la base de datos que se manifiesta como el pilar de la aplicación, su evolución desde el uso de *Archivos Secuenciales* hasta el diseño e implementación de la misma; así mismo, se conocerán los aspectos más importantes del contenido de sus campos y su propósito específico de existir, el cual se puede ostentar como el más completo registro de dermatólogos Mexicanos en el País.

2.1 De Archivos Secuenciales a Bases de Datos

Un archivo es un elemento de información conformado por un conjunto de registros. Estos registros a su vez están compuestos por una serie de bytes. Los archivos, alojados en dispositivos de almacenamiento pueden almacenarse de dos formas diferentes: *archivos convencionales* o ***bases de datos***.

Los archivos convencionales, pueden organizarse como *archivos secuenciales*; sin embargo, el almacenamiento de información a través de archivos secuenciales presenta una serie de limitaciones que restringen de manera importante la versatilidad que poseen las bases de datos para almacenar información.

En los *archivos secuenciales* los registros están almacenados en una secuencia que depende de algún criterio definido por el programa que lo crea; y es el mismo programa el único que conoce esta estructura o criterio, y únicamente él podrá acceder a los datos guardados; a diferencia de las bases de datos, las cuales se caracterizan por poseer una estructura estandarizada para cualquier manejador o plataforma en la que fueron creadas.

Cuando se desea *consultar* o modificar información en un archivo secuencial, también es necesario buscar uno por uno en los registros hasta encontrarla; en cambio, en una base de datos se define una *clave principal* que sirve para realizar dicha *consulta* sin tener que recorrer todos los registros.

Para definir como están estructurados y organizados los registros en una base de datos sería necesario un libro completo, y sería lo mismo para poder exponer todas las ventajas que tienen las bases de datos en contra de los archivos secuenciales. Sin duda alguna, depende mucho del tipo de desarrollo que se desee; pero para el caso del sistema de información *AcadMexDerm*, la mejor elección es la primera. El desarrollo y planeación de la estructura de la base de datos del sistema ha ido evolucionando a través del tiempo y se ha depurado de tal manera que se ha vuelto sumamente eficaz, y sin duda, seguirá cambiando, pero eso es otra ventaja de estas en contra de los archivos secuenciales, ya que no es necesario reescribir todos los registros, sino cambiar la estructura del depósito donde se guardarán.

Para la primera versión del sistema se creo un sistema de archivos secuenciales que podían guardar datos muy básicos de los dermatólogos; también se creaba otro archivo independiente que guardaba los datos de los eventos de la AMD;

este programa creaba internamente un vínculo entre dichos archivos como un intento de guardar un historial de las asistencias de los dermatólogos en los diferentes eventos.

El diseño del archivo de los congresos era muy simple, se trataba de guardar datos de los congresos y los diferentes talleres que se impartían en el mismo. Obviamente el número de talleres variaba en cada evento; al mismo tiempo, en el archivo de los miembros, se registraba únicamente el número de congreso al que asistía, además de guardar una serie de registros en forma de ***booleanos*** (verdadero o falso) para conocer su asistencia a los talleres; es decir, se guardaban 9 datos en total, el primero era el número de congreso, los otros ocho la serie de booleanos y este dato en conjunto conformaba el historial. El diseño del archivo de los congresos se puede apreciar mejor en la ***figura 2.1***:

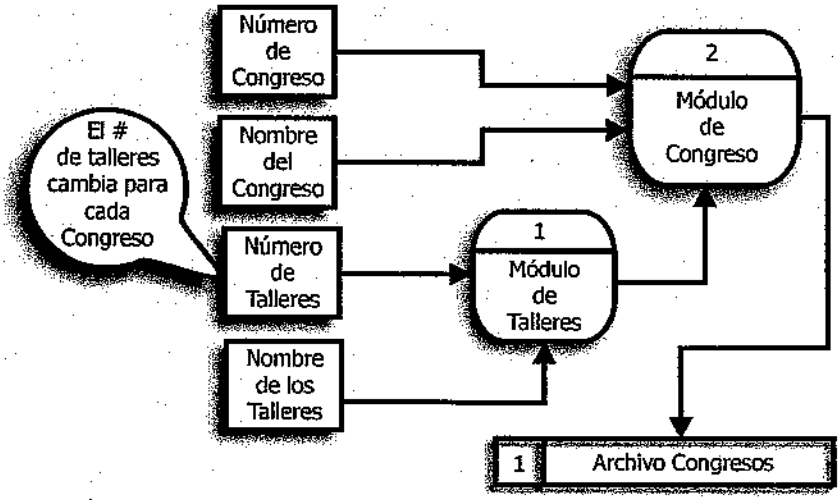


Figura 2.1→ Archivo secuencial de los congresos para la primera versión.

El programa poseía una interfaz gráfica muy sencilla que contenía pantallas para guardar los datos de los miembros y sus asistencias, así como los congresos de la AMD. De manera muy sencilla se vinculaba el arreglo de booleanos con ***casillas de verificación*** o ***checkbox's*** (☑) para capturar la asistencia a los diferentes talleres ***(figura 2.2)***. La desventaja era que sólo se podía registrar el último congreso al que asistió el miembro, de tal forma que no se podría llamar historial, ya que si fuera de esta forma, debería guardar los datos de todos los eventos a los que se asistía.

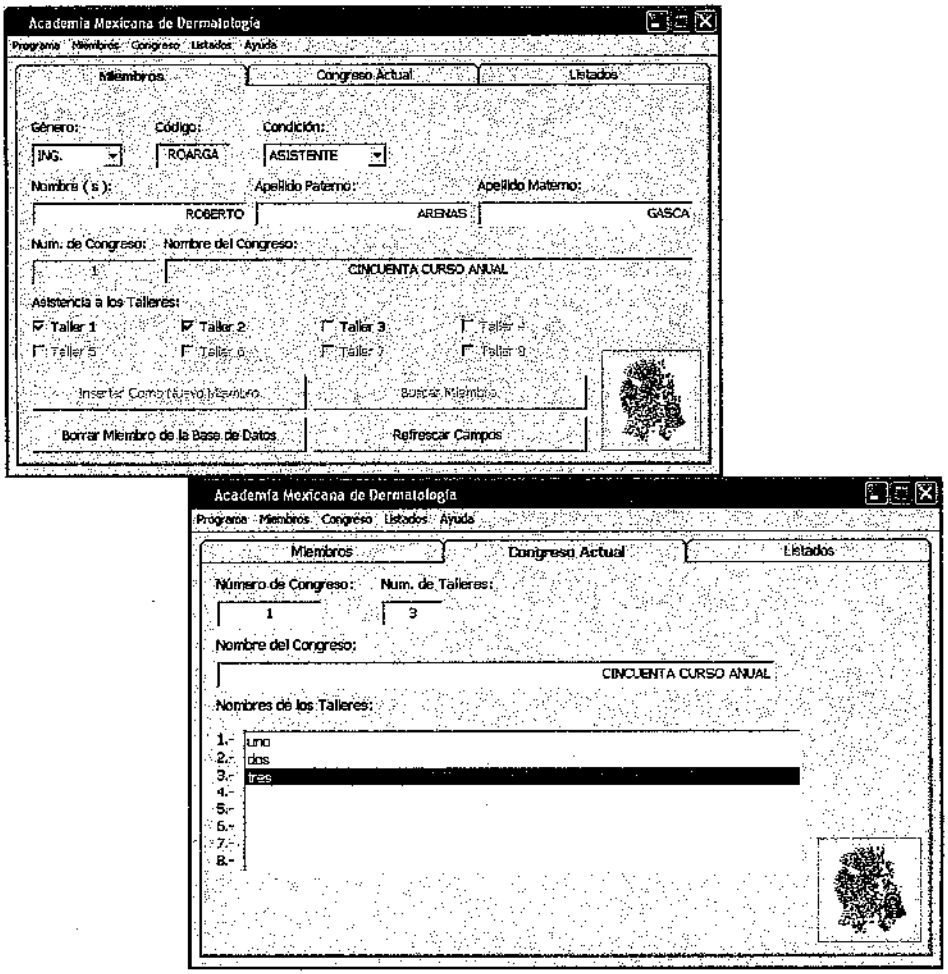


Figura 2.2 → Pantallas de captura de miembros y de congresos en la primera versión del programa AcadMexDerm.

Otro dato importante que guardaba el archivo de los miembros en un intento de no generar duplicidad, era un dato único que identificara a cada uno de los miembros para evitar la duplicidad al ingreso de los datos. Este dato fue denominado "***código***". El código sirve como una clave única que identifica a cada miembro durante los procesos como búsquedas, ordenamiento y evasión de duplicidad en la inserción de datos.

El código se generó en primera instancia tomando dos letras del nombre y dos de cada apellido ***(figura 2.3)***.

Figura 2.3 Código único e identificación.

La restricción principal para registrar a un miembro en el sistema era que existiera el archivo de los congresos, de otra forma no se podría guardar el registro de asistencia de cada uno. En conjunto se generaban dos archivos con más limitantes que usos; pero independientemente de éstas, la idea era sumamente útil, es decir, el poder poseer un registro único de miembros con el historial de sus asistencias sería un proyecto que valía la pena evaluar más a fondo. En la ***figura 2.4*** se puede observar un esquema general del registro de los miembros en el archivo del mismo nombre:

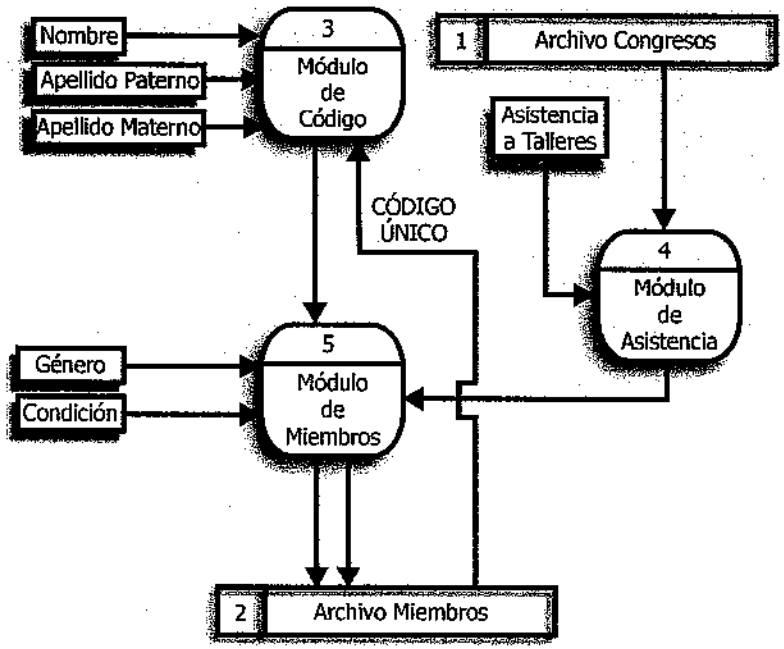


Figura 2.4 → Registro de los miembros en el archivo secuencial del mismo nombre.

2.2 Análisis de Necesidades para la Base de Datos

Antes de realizar el análisis del sistema, fue necesario realizar un estudio de la base de datos que fungiría como pilar para el programa; al comienzo se utilizó ***Access*** como manejador por ser la solución más sencilla y barata para utilizar, pero conforme fue pasando el tiempo se utilizó como una herramienta muy útil por estandarización al compartir el archivo con los Laboratorios Tipo "A", pero de esto se hablará más adelante. De tal forma, se eligieron las bases de datos para registrar los datos tanto de los miembros, sus historiales y los congresos; entonces se crearían dos tablas principales que compartirían una relación sencilla para poder obtener los nombres de los diferentes congresos y los diferentes talleres, y así poder guardar únicamente datos numéricos, que evidentemente son más controlables, exactos y ocupan mucho menos espacio. Para poder apreciar de manera más clara lo dicho ver la ***figura 2.5:***

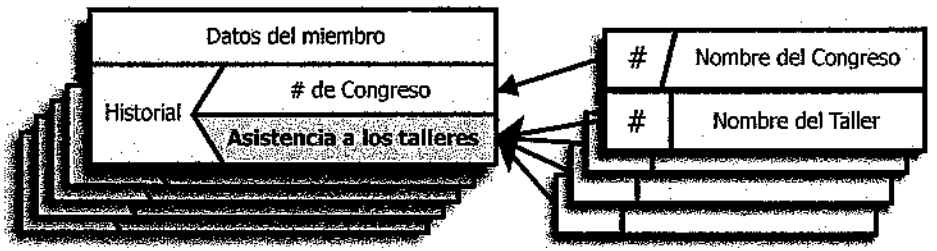


Figura 2.5 → Relación entre la tabla de los miembros (izquierda) y la tablado los congresos (derecha).

Como se aprecia en la figura anterior hay dos tablas principales, la de los *Miembros* y la de los *Congresos*, para la tabla *Miembros* hay una cantidad "n" de registros para historiales por cada uno de los miembros; en cambio para la tabla de los congresos hay un número máximo de 10 talleres por congreso; esto es porque nunca ha habido más de 8 talleres por congreso, en mayor parte por cuestiones de tiempo de duración del evento; pero de todas maneras existe la holgura de otros dos talleres por congreso, de hecho, no ha habido en la historia de la AMD más de 6 talleres.

También se puede observar que existe una relación simple para obtener el nombre del taller al que asistió el miembro, en cambio, la relación que existe al momento de obtener el nombre de los talleres a los que asistió es más compleja, dado que hay varios talleres y se convierte en una relación uno a "n" talleres. Pero como podríamos guardar un número limitado de confirmaciones de asistencia en un solo registro a la sección de historiales ***(sección sombreada del segmento izquierda de la figura 2.5)***.

Hay que recordar que en esta sección se guardaba un arreglo de booleanos, pero esta forma de guardar los datos era un poco, o bastante arcaica, en el sentido del espacio que ocupaba para guardar datos ***bool*** (verdadero o falso).

Se debe que hay un congreso con ocho talleres, y hay un miembro que asistirá a los primeros dos y también al cuarto. Lo que se tendría que guardar en el arreglo de su asistencia a los talleres sería algo muy parecido a lo que se muestra en la ***tabla 2.1:***

Taller 1	Taller 2	Taller 3	Taller 4	Taller 5	Taller 6	Taller 7	Taller 8
true	true	false	true	false	false	false	false

Tabla 2.1 → Arreglo de boolenos representando la asistencia a los talleres a un congreso

También se ha podido apreciar que el uso de los datos bool no es muy popular, pero se le ha encontrado que son sumamente útiles, en el sentido de que no hay margen de error, o es o **no** es. Pero el problema es que aunque se trate de un solo dato, por lo menos se necesita un byte para representarlo, direccionarlo y al mismo tiempo, guardarlo.

Al observar la tabla se puede apreciar una serie de elementos representados en forma de bits, es decir, un arreglo de ceros o unos, esto se puede ver a detalle en la ***tabla 2.2:***

Taller 1	Taller 2	Taller 3	Taller 4	Taller 5	Taller 6	Taller 7	Taller 8
1	1	0	1	0	0	0	0

Tabla 2.2 → Representación en binario del arreglo de booleanos de la tabla 2.1

Y no es necesario profundizar tanto para ver un arreglo de ocho ***bits***, es decir, un simple ***byte***, y esto sería sumamente útil para ahorrar espacio de almacenamiento, porque es claro que no es lo mismo ocho bytes que uno solo representando la misma información; y es de esta manera que se guardan en la base de datos los registros de asistencia para cada uno de los miembros. En base a lo anterior, no se guarda "**11010000-binario**", si no que se guarda "**208-decimal**" en el registro del miembro.

Aparte de los datos de historiales de los miembros, hay que almacenar en los registros datos trascendentales referentes a su información personal como dirección, incluyendo calles, colonias, números y cualquier dato de la manera más

detallada posible; porque es de vital importancia para la AMD el poder tener contacto a través de correo postal con todos sus miembros; el envío de información por medios postales es sumamente usado por la AMD y el tener una base de datos lo más actualizada posible es imperativo.

Otros datos han sido estandarizados para uso interno, pero al mismo tiempo pueden ser utilizados para poder referenciar por sus características a los miembros; esto es, referente al género de los médicos, se han dividido en Dr., Dra., Sr., Sra., etc. Y de esta misma forma se han dividido varios tipos de datos como son:

- Especialidad *(profesionalmente hablando, especialidad médica)*
- Estado
- País *(Por limitantes de miembros extranjeros se han dividido únicamente en México y Extranjero)*
- Delegación
- Miembro *(Recordando que para esta clasificación existen Miembros AMD, AMD-FMD y OTRO)*
- Tipo de Cédula *(Profesional o Especialidad)*
- Condición *(En este caso se hace referencia al tipo de asistencia a algún congreso, es decir, Asistente, Profesor, Staff, Acompañante, etc.)*

Teniendo en cuenta que todos estos datos son indispensables, y tomando como referencia los datos que se generaban en la primera versión, es posible generar un esquema de una base de datos que cubra las necesidades de la asociación.

Independientemente de los datos que se requieran guardar, un aspecto importante es el manejador de base de datos que se utilizaría en el programa; se acordó utilizar en primera instancia ***Access***, por ser un estándar para casi cualquier laboratorio al que se deseara exportar el archivo y ser enviado por correo electrónico. Independiente del manejador utilizado para el programa, la página de Internet utilizaría la base de datos pero en ***MySQL***, pero este tema se ampliará en el siguiente capítulo.

Por último, otra ventaja que tiene *Access* contra otros manejadores es que la Academia ya poseía las licencias de *Microsoft Office XP Professional*, así que no habría problemas por el lado de las licencias y la legalidad del software; así que por *viabilidad tecnológica* no habría restricciones; en el ámbito operacional y económico, las ganancias por la venta de la base de datos a los laboratorios es considerable.

Gracias a la interfaz sencilla del programa, el personal de la AMD debe ser autosuficiente para la operación del programa. Su mantenimiento sería mínimo, ya que el tamaño del archivo no es tan grande; es por este aspecto que es sumamente importante la validación de los datos, para que no entre información errónea a la base de datos y no tener que depurar a futuro, es decir, limpiar la *"basura"* de los registros.

2.3 Diseño e Implementación de la Base de Datos

Antes de realizar la implementación se ha desarrollado el diseño conceptual de la base de datos en dos tablas principales, la de los *Miembros* y la de los *Congresos*; se han evaluado los tipos de datos, sus tamaños y características dentro del sistema. La primera tabla es la de los *"Miembros"*; la especificación de cada uno de los campos y sus características principales se muestran en la ***tabla 2.3:***

Nombre del Campo	Tipo	Tamaño	Características	Descripción
Codigo	Texto	12	**Clave Principal**, Requerido, NO Duplicado	Código Único por Miembro
Nombre	Texto	20	Requerido	Nombre del Miembro
ApP	Texto	20	Requerido	Apellido Paterno
ApM	Texto	20	Requerido	Apellido Materno
Condicion	Texto	11	No Requerido	Condición del Miembro en el evento en el que está inscrito; únicamente podrá ser uno de los siguientes: • PROFESOR • ASISTENTE • ALUMNO • ACADEMICO • BECARIO • STAFF • PONENTE • EXPOSITOR • ACOMPAÑANTE
CP	Texto	10	Requerido	Código Postal
Genero	Texto	6	Requerido	Género del Miembro; únicamente podrá ser uno de los siguientes: • DR. • DRA. • SR. • SRA. • LIC. • EST. • ING.

Calle	Texto	40	Requerido	Domicilio (Calle)
Colonia	Texto	30	Requerido	Domicilio (Colonia)
Delegacion	Texto	25	Requerido	Domicilio (Delegación Política)
DelegacionCon	Texto	25	Requerido	Domicilio (Delegación Política del Consultorio Principal)
Estado	Texto	23	Requerido	Domicilio (Estado de la República Mexicana o del Extranjero); podrá ser uno de los 32 Estados de la República Mexicana o Cualquiera del Extranjero.
Pais	Texto	6	Requerido	Domicilio (País en el que radica), únicamente podrá ser uno de los siguientes: • MEXICO • EXTRAN
Especialidad_1	Texto	20	No Requerido	Especialidad Médica Uno
Especialidad_2	Texto	20	No Requerido	Especialidad Médica Dos
Cedula	Texto	12	No Requerido	Tipo de Cédula: • NO TIENE • PROFESIONAL • ESPECIALISTA
NoCedula	Texto	17	No Requerido	Número de Cédula (Numéricamente hablando)
CURP	Texto	18	No Requerido	Clave Única de Registro Poblacional
Tel_Casa	Texto	19	No Requerido	Teléfono Particular
Tel_Consu1	Texto	19	No Requerido	Teléfono Uno de Consultorio
Tel_Consu2	Texto	19	No Requerido	Teléfono Dos de Consultorio
Cel	Texto	19	No Requerido	Teléfono Celular Particular
Beeper	Texto	21	No Requerido	Número de Localizador, Beeper o Equivalente
EMail	Texto	60	No Requerido	Correo Electrónico Particular
Fax	Texto	19	No Requerido	Fax Consultorio o Particular
Miembro	Texto	8	Requerido	Indica si es miembro de la AMD, de la FMD* o ninguno de los dos, únicamente podrá ser uno de los siguientes: AMD AMD-FMD OTRO
RFC	Texto	16	No Requerido	Registro Federal de Contribuyente
NumCongAct	Entero	-	No Requerido	Número de Congreso al que se encuentra inscrito actualmente
AsisAct	Entero	-	No Requerido	Asistencia a Talleres del Congreso en el que se encuentra Inscrito Actualmente**
NumHis	Entero	-	No Requerido	Nos indica cuál es el último número de historial ocupado
NumHis_1	Entero	-	No Requerido	Historial 1 (No. Congreso)

AsisHis_1	Entero	-	No Requerido	Historial 1 (Asistencia Talleres)
NumHis_2	Entero	-	No Requerido	Historial 2 (No. Congreso)
AsisHis_2	Entero	-	No Requerido	Historial 2 (Asistencia Talleres)
NumHis_3	Entero	-	No Requerido	Historial 3 (No. Congreso)
AsisHis_3	Entero	-	No Requerido	Historial 3 (Asistencia Talleres)
NumHis_4	Entero	-	No Requerido	Historial 4 (No. Congreso)
AsisHis_4	Entero	-	No Requerido	Historial 4 (Asistencia Talleres)
NumHis_5	Entero	-	No Requerido	Historial 5 (No. Congreso)
AsisHis_5	Entero	-	No Requerido	Historial 5 (Asistencia Talleres)
NumHis_6	Entero	-	No Requerido	Historial 6 (No. Congreso)
AsisHis_6	Entero	-	No Requerido	Historial 6 (Asistencia Talleres)
NumHis_7	Entero	-	No Requerido	Historial 7 (No. Congreso)
AsisHis_7	Entero	-	No Requerido	Historial 7 (Asistencia Talleres)
NumHis_8	Entero	-	No Requerido	Historial 8 (No. Congreso)
AsisHis_8	Entero	-	No Requerido	Historial 8 (Asistencia Talleres)
NumHis_9	Entero	-	No Requerido	Historial 9 (No. Congreso)
AsisHis_9	Entero	-	No Requerido	Historial 9 (Asistencia Talleres)
NumHis_10	Entero	-	No Requerido	Historial 10 (No. Congreso)
AsisHis_10	Entero	-	No Requerido	Historial 10 (Asistencia Talleres)
NumHis_11	Entero	-	No Requerido	Historial 11 (No. Congreso)
AsisHis_11	Entero	-	No Requerido	Historial 11 (Asistencia Talleres)
NumHis_12	Entero	-	No Requerido	Historial 12 (No. Congreso)
AsisHis_12	Entero	-	No Requerido	Historial 12 (Asistencia Talleres)
NumHis_13	Entero	-	No Requerido	Historial 13 (No. Congreso)
AsisHis_13	Entero	-	No Requerido	Historial 13 (Asistencia Talleres)
NumHis_14	Entero	-	No Requerido	Historial 14 (No. Congreso)
AsisHis_14	Entero	-	No Requerido	Historial 14 (Asistencia Talleres)
NumHis_15	Entero	-	No Requerido	Historial 15 (No. Congreso)
AsisHis_15	Entero	-	No Requerido	Historial 15 (Asistencia Talleres)
NumHis_16	Entero	-	No Requerido	Historial 16 (No. Congreso)
AsisHis_16	Entero	-	No Requerido	Historial 16 (Asistencia Talleres)
NumHis_17	Entero	-	No Requerido	Historial 17 (No. Congreso)
AsisHis_17	Entero	-	No Requerido	Historial 17 (Asistencia Talleres)
NumHis_18	Entero	-	No Requerido	Historial 18 (No. Congreso)
AsisHis_18	Entero	-	No Requerido	Historial 18 (Asistencia Talleres)
NumHis_19	Entero	-	No Requerido	Historial 19 (No. Congreso)
AsisHis_19	Entero	-	No Requerido	Historial 19 (Asistencia Talleres)
NumHis_20	Entero	-	No Requerido	Historial 20 (No. Congreso)
AsisHis_20	Entero	-	No Requerido	Historial 20 (Asistencia Talleres)

Tabla 2.3 → Tabla "Miembros" de la base de datos

* Al ser miembro de la Fundación Mexicana para la Dermatología, automáticamente forma parte de la Academia Mexicana de Dermatología, y es por esto que se considera una única clave unificada dentro del campo *Miembro* para AMD-FMD.

** Este dato es únicamente para el registro de asistencia a los talleres del congreso al que se encuentra inscrito actualmente. Posteriormente, al inscribirse a un nuevo evento, ese dato junto con el registro ***NumCongAct*** pasarán a ser parte del historial y tomarán el siguiente espacio vacío dentro del área de historiales.

La segunda tabla es la de los *"Congresos"*, la especificación de cada uno de los campos y sus características principales se muestran en la ***tabla 2.4:***

Nombre del Campo	Tipo	Tamaño	Características	Descripción
NumCon	Entero	-	Llave Principal, Requerido, NO duplicado	Número de Congreso*
NomCon	Texto	80	Requerido	Nombre del Congreso
NumT	Entero	-	Requerido	Número de Talleres en el Evento**
Fecha	Fecha	-	Requerido	Fecha del Evento
Nt1	Texto	40	No Requerido	Nombre del Taller 1
Nt2	Texto	40	No Requerido	Nombre del Taller 2
Nt3	Texto	40	No Requerido	Nombre del Taller 3
Nt4	Texto	40	No Requerido	Nombre del Taller 4
Nt5	Texto	40	No Requerido	Nombre del Taller 5
Nt6	Texto	40	No Requerido	Nombre del Taller 6
Nt7	Texto	40	No Requerido	Nombre del Taller 7
Nt8	Texto	40	No Requerido	Nombre del Taller 8
Nt9	Texto	40	No Requerido	Nombre del Taller 9
Nt10	Texto	40	No Requerido	Nombre del Taller 10

Tabla 2.4 → Tabla "Congresos" de la base de datos

* El Número de Congreso se refiere únicamente como un número de control para el sistema, esto es por ejemplo, desde la puesta en marcha del sistema se han ingresado 5 congresos y el Número cuatro es el 50 Curso Anual Terapéutica Dermatológica, sin embargo, el número de congreso no es el 50, sino el 4, ya que fue el cuarto en registrarse en la Base.
** En el caso de que sólo fuera un simposio de exposiciones, por ejemplo, el número de talleres sería cero, por lo tanto, este dato siempre será requerido para generar procesos como la inserción del nombre de los talleres, y saber posteriormente como se pasará asistencia a los miembros en el caso de que sean inscritos al evento.

El ingreso de la información dentro del sistema es relativamente sencillo, hay restricciones que existen desde la primera versión del programa, y aunque han ido evolucionando, la esencia del ingreso de un nuevo miembro, su edición y el registro de sus historiales se mantiene como una secuencia controlada por el sistema, esto es, la base de datos no puede registrar por si sola datos como los historiales, sino que es el programa y únicamente él podrá, registrar, cambiar y realizar las consultas y las relaciones necesarias para poder mostrar resultados y procesar la información que da el usuario, interpretarla y guardarla de la manera más compacta posible.

En el caso de los historiales, tal vez habría otros métodos para poder registrarlos, porque es evidente que cada uno de los miembros posee una limitante de 20 registros individuales que representan las asistencias a los eventos;

y es que, aunque la AMD organiza sesiones mensuales, los eventos que se registran en los historiales son únicamente los congresos o reuniones naturalmente más grandes.

Se podría crear una tabla especial para los historiales, que fuera independiente de la tabla de sus datos personales, y en esencia tiene varias ventajas, siendo la más evidente el uso innecesario de espacio por registro; pero el problema viene cuando no se lleva un control estricto de la entrada de los registros; al tener una limitante de registros tampoco se desperdicia espacio, ya que cada uno de ellos sólo es representada por un número, y aunque sean veinte, siguen siendo números enteros, que ocupan muy poco espacio; sin embargo, si existiera una tabla independiente se tendría que vincular de alguna forma con la tabla de los miembros, y este dato sería la clave única por miembro.

La clave única por miembro también ocuparía espacio, y de la misma forma ocupa memoria. Adelantándose un poco a lo que se verá en la definición del programa, el sistema utiliza un nuevo método de conexión a bases de datos llamado *ADO .Net*, que es básicamente tener una base de datos completa en una memoria caché que se almacena de forma temporal y se trabaja sobre la misma, mientras que la base de datos original permanece cerrada. Esta es una forma muy innovadora de trabajar sobre las bases de datos, pero de ello se hablará más adelante; el caso es que la clave única ocuparía un mayor espacio en memoria que teniendo una sola tabla completa, con todos los historiales; de hecho así sería, los historiales ocuparían un espacio mayor, ya que si se acumularan de manera independiente, se almacenarían en caché como una tabla completa, ocupando más espacio, y al mismo tiempo, las transacciones que se realizarían sobre los datos y sus relaciones se tornarían más complejas; tal vez la forma más sencilla de hacer las cosas sea la más fácil de controlar, por lo menos para este caso.

Como se menciona anteriormente, el manejador de la base de datos es ***Access***, y esto es por ser el más comúnmente usado por los Laboratorios a los que se les exporta una consulta de la archivo original; esto como un servicio único para los Laboratorios inscritos a la institución y nombrados como Tipo "A", los cuales son privilegiadamente tratados por la AMD. Dichos laboratorios pagan una cuota por tener derecho a esta información actualizada. De hecho los laboratorios usan la base de datos de forma parecida al que le da la AMD; lo que les interesa más a ellos es enviar información de sus productos y medicamentos a los dermatólogos, para que ellos los receten a sus pacientes, pero no se entrará en ese campo, ya que los registros enviados a los laboratorios son datos confidenciales; lo importante es destacar que se hace buén uso de dicha información.

De esta manera es que se comparte la base de datos con otras empresas, y la manera más fácil para poder realizar un archivo especial para enviar por correo electrónico es exportar a través de una consulta desde el archivo original hasta otra tabla de *Access* o a una *Hoja de Cálculo* ***Excel.***

Los Laboratorios tipo "A" son siempre bienvenidos a los eventos de la AMD; instalan mostradores con sus productos, y también tienen un espacio especial en la página de Internet. Poseen ***banners*** especiales y documentos con sus productos, para que cualquiera que visite la página vea dicha información, pero de esto se hablará más a fondo en el siguiente capítulo; por lo pronto se puede mencionar que los laboratorios inscritos como tipo "A" de la AMD hasta ahora son:

- Allergan, S. A. de C. V.
- Boehringer Ingelheim Promeco, S. A. de C. V.
- Centro Internacional de Cosmiatria, S. A. de C. V.
- Compañía Procter & Gamble México, S. de R. L. de C. V.
- Galderma México, S. A. de C. V.
- Laboratorios Vichy
- La Roche Posay
- Novartis Farmacéutica, S. A. de C. V.
- Pierre Fabre México, S. A. de C. V.
- Siegfried Rhein, S. A. de C. V.
- 3M México, S. A. de C. V.

Como se ha visto, la base de datos sigue siendo una implementación muy sencilla; ciertamente no se han agregado desde el inicio muchos de los registros que contiene ahora, pero conforme han avanzado las versiones, se han agregado elementos de tal modo que todos los datos necesarios para la AMD han sido contemplados, tanto los datos personales, como los necesarios para que el sistema funcione y la página de Internet se mantenga actualizada. Cada una de las partes y registros de la base de datos han sido contempladas para uno o varios propósitos dentro del sistema en conjunto.

2.4 Mantenimiento y Respaldo de la Base de Datos

El mantenimiento que se le dará a la base de datos resulta ser mínimo, de hecho, casi inexistente; durante la evolución del programa han crecido ciertos datos como el código, datos personales de los médicos, más registros para los historiales, etc.; pero hasta hoy se ha experimentado un segmento amplio de tiempo en el que no se han tenido que hacer cambios trascendentales a la Base. En cambio, hablando de la depuración que se le ha dado ésta, es decir, los datos que se pueden denominar como "basura" dentro de la base de datos, han sido prácticamente

inexistentes, y esto es, como se verá más adelante, gracias a la validación que se le da a los datos antes de ser introducidos al archivo; por ejemplo, la duplicidad en los registros, que todos los registros se encuentren en mayúsculas, diferenciar nombres abreviados, no aceptar caracteres inválidos, entre muchos otros. Esta validación se ha perfeccionado a través de la evolución del programa, llegando casi a contemplar todos los errores que se pueden presentar al ingresar la información. Aunque al principio el sistema presentaba varias fallas que tal vez eran muy evidentes, se ha desarrollado un buen sistema, estable y autosuficiente, esto último al referirse a la interfaz con el usuario final, es decir, contra errores generados por el usuario al insertar los datos.

Gracias a la experiencia que se ha adquirido a través del tiempo, se ha comprobado que para evitar tener que depurar una base de datos es mejor prevenir, y es a la hora de su ingreso que hay que pensar en casi un sinfin de errores que podría cometer el usuario, y, como el usuario no esta obligado a saber que tipo de errores se podrían cometer al ingresar un registro, como por ejemplo, que para algunos sistemas no es lo mismo una letra minúscula que una mayúscula, hay que facilitarles el trabajo lo más posible, es decir, que no tengan que darse cuenta que están ingresando un registro de manera incorrecta, y cuando se comete el error, que el sistema sea, como lo hemos mencionado, autosuficiente para poder señalarle dónde y cuál fue su error en específico; y con esto no sólo facilitaremos su trabajo, sino el propio también.

Entonces, gracias a la validación previa de la entrada de información, referente a la depuración, sólo quedan los respaldos; al ser un simple archivo, se realizan respaldos de manera mensual, simplemente comprimiendo el archivo y guardándolo en un disco de 3 ½ al cual le basta y sobra espacio, ya que el archivo no rebasa los 2.5 ***MBytes*** sin compresión, y comprimiéndolo ocupa alrededor de 400 ***KBytes***; entonces, se puede afirmar que un archivo de disquetes es más que suficiente para respaldar la base; aunque actualmente se encuentra en producción un disco ***regrabable*** para tener un medio óptico y mucho más durable que un disquete.

La base de datos sólo sufre cambios trascendentales durante los dos meses anteriores a un evento y uno posterior al mismo, esto es porque durante este tiempo se reciben las inscripciones a los eventos y se pueden registrar las asistencias al mismo; también es cuando se actualizan los datos de cada uno de los miembros como sus direcciones y teléfonos, ya que los miembros mandan de nuevo sus datos por medio del formato de inscripción ***(figura 1.3)*** que reciben vía correo postal y que son recibidos y revisados minuciosamente uno por uno por el personal de la AMD.

Con respecto a la copia de la base de datos para la página de Internet, se realizan respaldos especiales que se almacenan en diferentes equipos, pero se ahondará más en este tema en el siguiente capítulo. Referente a la base que se envía a los diferentes laboratorios, es importante destacar que no todos los registros son enviados como archivo a los mismos, es decir, sólo se les envía una consulta especial que contiene únicamente los datos referentes a sus direcciones; los datos como teléfonos e historiales son exclusivos de la AMD, y esta consulta se guarda como la misma, una simple consulta dentro del archivos original; entonces, al hacer el respaldo de la base original, es incluida la que se envía a los laboratorios.

2.5 Planeación y Análisis Preliminar del Sistema de Información

Hasta el momento se han planteado una serie de necesidades para la AMD que necesitan ser resueltas de la mejor manera; sin saber si las necesidades podrán ser cubiertas de manera satisfactoria, se puede comenzar normalizando lo que necesita hacer el sistema; primero es necesario dividir todo el esquema en secciones como se han dividido a las tablas que anteriormente se estandarizaron e implementaron como una base de datos que servirá como pilar para la aplicación.

Las tablas, aún con datos, no sirven de nada, porque se podría decir que no saben hacer nada por si solas; y es por esto que es necesario un sistema de información que utilice estos datos, los procese y muestre en una salida que le sirva a la organización y que sea comprensible para cualquier usuario. Entonces, en primer lugar se necesita conocer lo que el sistema debe realizar con estos registros.

Lo primero es que la información este siempre disponible, debe estar en el momento que el usuario la requiera; debe conservar su integridad, la información debe ser consistente, fiable y no propensa a alteraciones no deseadas. La información debe ser vista y manipulada sólo por quienes tienen el derecho o la autoridad de hacerlo, es decir, se debe crear un sistema que tenga la capacidad de poseer una interfaz de edición y otra que sólo sirva para ver y analizar la información.

Poseer una interfaz simple y amable para ingresar los datos de los miembros es importante; esta interfaz debe ser capaz de validar los datos que se ingresan, por ejemplo; que no se ingrese en el sistema letras minúsculas ni datos que puedan incurrir en errores internos, como que el correo electrónico sea una dirección válida, es decir, que contenga una arroba (@), un punto y que no tenga espacios en blanco, ya que de otra forma, no se podrán enviar correos electrónicos.

Otra cosa que debe hacer la interfaz es mostrar los datos de manera ordenada, legible, coherente y precisa. Pero lo más importante de todo es que debe evitar la duplicidad en los datos; y la única manera de hacer esto es crear un código que sea único para cada uno de los miembros y que sea autosuficiente para que, aunque crezca mucho la base de datos, ningún dermatólogo posea un código que ya exista.

Este último punto es muy importante para la AMD, y sin lugar a duda, ha sido algo en lo que se ha puesto mucho énfasis y cuidado. Teniendo un código como el que se poseía en la primera versión es seguro que habría algún tipo de duplicidad. Para esto se ha pensado, por ejemplo, en cuantos Juan Pérez Pérez hay en el país, y aunque habrá alguna duda que haya dos Juanes Pérez Pérez que sean dermatólogos, se puede dar el caso o bien tal vez haya dos personas cuyas primeras dos primeras letras de su nombre sean las mismas, como *Mario, Maria* o *Maribel*, y para los apellidos sería lo mismo, pensando en *Hernández* o *Herrera*.

Se supone que las autoridades gubernamentales se enfrentaron a este mismo problema al crear la *Clave Única de Registro Poblacional* ***CURP***, y entonces quizás lo primero que se preguntaron fue: ¿Qué dato se posee de cada persona que sea único y lo diferencie de las otras que se llaman igual que ella?. Entonces alguien propuso que este dato sería la fecha de nacimiento. Para la AMD era igual, pero la AMD no posee la fecha de nacimiento de sus miembros, ya que este dato es irrelevante; tampoco se puede usar su ***RFC*** o ***CURP***, ya que no se posee este dato para todos los registros que se tenían archivados; pero entonces, ¿Cuál es el dato que se posee de todos los miembros ya registrados, y que también sea requerido obligatoriamente al inscribir a un nuevo miembro?. Pues este dato resulta ser el *Código Postal*.

Fue entonces que se tomó la decisión de crear un código único para cada dermatólogo usando las letras de sus nombres y apellidos en conjunción a su número de Código Postal para crear lo que se puede observar en la ***figura 2.6:***

Figura 2.6 → Nuevo Código Incrustando Código Postal

Se ha tomado únicamente las dos primeras letras del nombre porque hay registros con abreviaciones de nombres como María (Ma.) y que los médicos así llenan en los formatos de registro. También se han tomado sólo los últimos cuatro números del código postal para no provocar errores por ceros a la izquierda, los cuales son inválidos cuando se manejan como números, y aunque es precisamente por esto por lo que el código postal se manejará como una cadena de letras, es aquí donde se ha prevenido que el usuario no ponga el cero a la izquierda si el código postal no lo incluye en el formato de registro.

La interfaz también tendrá una sección especial en la que se podrán ingresar las asistencias a los eventos, y aunque el usuario no tendrá conocimiento de como se guardarán los registros de asistencias internamente, si es preciso mostrar de una manera sencilla la asistencia del miembro a algún evento. El sistema debe ser capaz de traducir los números decimales de los registros a una forma comprensible para el usuario, y esto no sólo para mostrar la concurrencia del médico, sino también para ingresar un nuevo registro o bien editar algún cambio.

Se debe poder realizar diversas consultas, dados ciertos parámetros; estos parámetros deben ser proporcionados por el usuario, pero esto sin generar confusión en cuanto al uso de una consulta en si, es decir, que el usuario no tenga conocimiento que se le esta mostrando una consulta de una tabla de la base de datos; en cambio, debe poder seleccionar de entre los parámetros específicamente importantes para la AMD para mostrar un subconjunto de los miembros; los parámetros estándar que se han propuesto son:

- Por Estado de la República Mexicana
- Por Condición
    - Profesor, Asistente, Alumno, Académico, Becario, Staff, Ponente, Expositor o Acompañante
- Por Miembro
    - AMD, AMD-FMD u OTRO
- Por Especialidad
    - Dermatología, Micología, Oncología, Internista, Cirugía Plástica, Cirugía, Medicina General o Laboratorio
- Por Número de Congreso
    - Asistencia al Congreso o Sin Filtro de Asistencia (Todos)

Ya teniendo los parámetros de las consultas, se debe tener en cuenta que la interfaz para realizarla debe de ser amigable y accesible para cualquier persona, de tal manera que sea lo más fácil posible de utilizar para el usuario pero limitar la posibilidad de hacerla conforme a los parámetros antes mencionados.

Se deben poder realizar búsquedas por nombre y apellidos; esto es muy importante, ya que al recibir una petición externa de datos de algún miembro, la AMD esta obligada a proporcionar los datos a quien lo requiera, incluyendo alguna persona que desee saber si en realidad algún médico es miembro o no de la Academia Mexicana de Dermatología.

Otra parte muy importante es referente a las impresiones que deberá realizar el sistema, comenzando por poder imprimir los datos de algún dermatólogo después de ubicarlo dentro del programa. Otro tipo de reportes que debe de realizar es el de un tiraje completo de los datos de cada uno de los dermatólogos, y aunque este tipo de archivo no es muy común, la idea es tener un archivo impreso del reporte general de todos los Miembros. Aún más importante que los dos anteriores, es el tiraje de etiquetas para los correos postales, ya sea que se desee enviar algún tipo de información impresa o folletos, se debe tener la capacidad de poder crear tirajes de etiquetas con todos los registros y sus direcciones actualizadas. Es muy importante que el tiraje de etiquetas este ordenado por código postal, de menor a mayor; ya que esto es requisito en las oficinas postales de la capital que cuando se rebasa un cierto número de envíos en masa, deben estar ordenadas de dicho modo; de otro forma, se tendría que ordenarla mano, y de hecho, antes del programa, el ordenado era a mano, de tal manera que esto retrasaba el envió de información urgente.

Por último, pero no menos importante, es poder enviar los correos electrónicos de manera masiva; pero hay un punto que no se ha mencionado, y que resulta interesante mencionar, y es el hecho de que las consultas deben de afectar al sistema de correos electrónicos, es decir, que cuando se realiza una consulta, el sistema de correos electrónicos busque, a partir del subconjunto de personas afectadas por la consulta, las que poseen correo electrónico y agregarlas a la lista de destinatarios del correo automáticamente, y esto sin afectar el contenido del correo en si, de tal forma que todas las personas reciban la misma información contenida dentro del correo electrónico.

Terminando con la sección referente a los miembros, se puede mencionar ahora la de los congresos. Por el momento, lo único que hay que resaltar al respecto, es el hecho de que una vez registrado un evento, es muy importante evitar realizar cambios en él, ya que al realizar un cambio imprevisto, el resultado sería que los historiales se verán afectados de manera directa, ya que no concordarían los registros de historiales con los cambios realizados al congreso; esto es muy importante, ya que al poder guardar datos de los congresos, estos están directamente vinculados a los registros de los miembros, y es por esto que se debe tener un mayor cuidado con los cambios y validaciones que se realicen en los congresos ya guardados, además de que los nuevos congresos que se vayan

registrando deben de pasar por un preproceso de validación, y aunque la AMD tiene conocimiento de sus eventos a realizar con un año de anticipación, los talleres son los que van cambiando; esto es porque los talleres son cursos pequeños que se dan en razón con lo que se esté utilizando en el momento; por ejemplo, un taller de ***Toxina Butolínicia***, que es un medicamento inyectado ampliamente conocido en esta época, tiene alta demanda en el mercado, y por esta razón se ofreció un taller especial para dicho propósito, causando gran interés entre los asistentes que acudieron para poder aplicarla de manera correcta, y al mismo tiempo, obtener el permiso para inyectarla y la autorización adecuada para adquirirla en el mercado, ya que no cualquier persona la puede comprar, y mucho menos aplicar.

En resumen, se puede observar en conjunto el planteamiento del análisis anterior de manera más gráfica observando la ***figura 2.7***, que muestra las facultades que debe poseer el sistema para procesar los datos de los Miembros:

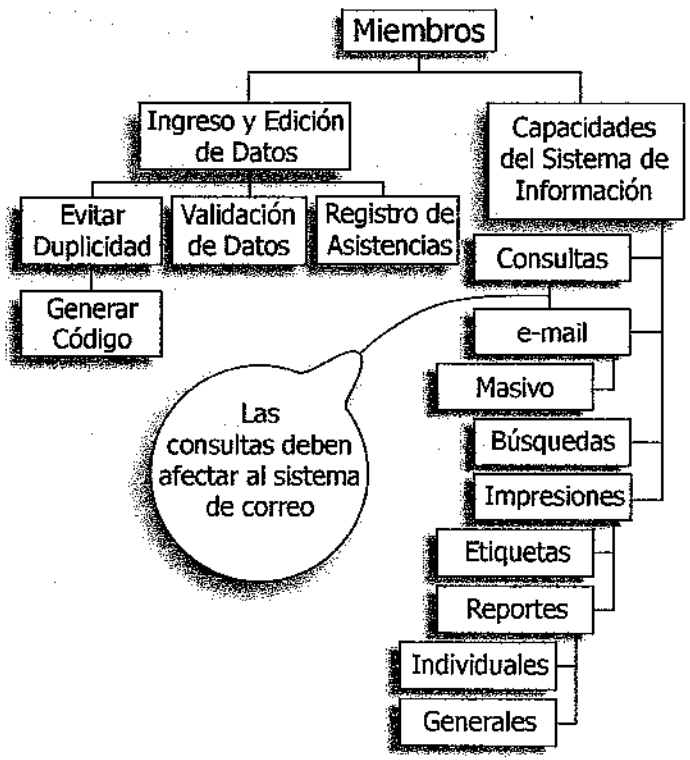


Figura 2.7 → Funciones del apartado de Miembros

El sistema será capaz de realizar todas las funciones antes mencionadas en un solo programa y de manera unificada; ya que el usuario debe poder realizar cualquier cosa que necesite en un solo programa, y aunque utilice otros programas mientras esta abierto el sistema, se debe recordar que un solo usuario tendrá la posibilidad de ingresar nuevos registros o editar los existentes; mientras que las otras dos personas de la oficina sólo lo utilizarán para realizar consultas y búsquedas de los diferentes médicos. La Mesa Directiva de la AMD tendrá los mismos permisos de consultas y búsquedas; y es que ellos lo tendrán instalado en sus consultorios particulares, y podrán bajar una versión de instalación del archivo de la base de datos con lo que tendrán el sistema actualizado con los últimos cambios; pero de esto se hablará en el siguiente capítulo.

Por último es importante mencionar que el número de registros para los historiales de cada uno de los miembros debe ser suficiente para un promedio de 10 a 15 años. Tal vez en este tiempo se realicen cambios en el sistema, que de cierta manera no sería muy difícil, pero mientras pasa el tiempo, podemos aclarar que los eventos que organiza la AMD son, en promedio, dos al año; y es muy raro que los miembros asistan a todos.

2.6 Análisis de Alternativas de Solución y Factibilidad Técnica

Las primeras versiones fueron realizadas en ***Visual Basic 6***, dada su facilidad para desarrollar aplicaciones bajo plataforma ***Windows (Formularios de Windows)***, e incluyendo la primera versión, el programa ha evolucionado hasta adoptar una forma sutil y elegante para ser una aplicación tan sencilla, que al mismo tiempo encierra una compleja estructura de secciones que se organizan para interactuar en beneficio de la organización.

Por desgracia, para el desarrollo de las primeras versiones no se tenían licencias de ningún tipo, excepto las de los Sistemas Operativos *Microsoft Windows XP Home Edition* para los tres equipos y únicamente una de *Microsoft Office Standard.*

Durante el desarrollo de las primeras versiones se hizo un estudio muy profundo para el desarrollo del sistema. El resultado fue un amplio conocimiento del compilador *Visual Basic* junto con tres de sus componentes que resultaban absolutamente imprescindibles y eficaces para las necesidades de desarrollo del programa.

Los tres componentes y sus funciones eran:

- ***Microsoft Visual Basic ADODB (Objeto de Datos Active X)***
  - La función principal de este componente es realizar una ***conexión permanente*** con una base de datos y enlazar las columnas de las tablas con algún control de formulario de Windows de tal forma que se puedan asociar los registros y poder editarlos o sólo moverse a través de ellos para examinarlos.
- ***Crystal Reports 8.5 para Visual Studio***
  - A través de este componente se puede crear una serie de reportes imprimibles a través de una conexión permanente a una base de datos y sus registros, y de manera precargada, contiene reportes para etiquetas estándar, a través de las cuales, se podrían crear las etiquetas para correo postal.
- ***Active X Microsoft Outlook 10.0 Object Library***
  - A través de esta referencia a ***Outlook*** es posible enviar correos electrónicos personalizados con todas las opciones estándar disponibles desde este pero implementadas en nuestro software. Esta herramienta también permite poner elementos especiales, como notas y citas, así como poder ingresar a los buzones de entrada y poder recibir los correos electrónicos desde cualquier buzón remoto hasta nuestro propio programa.

Estos tres elementos, en conjunto formaron un programa de alrededor de 3620 líneas que conformaban el registro, validación, edición, reporteo, correo masivo, y entre otros objetos, la interfaz gráfica de las anteriores versiones del sistema de información.

Existían componentes que no convivían cordialmente entre ellos, y por eso existían módulos independientes que funcionaban de manera separada, y de hecho sólo se hacía referencia a ellos como una llamada a un programa ejecutable que funcionaba independiente del programa principal, que se encargaba únicamente de realizar los registros y su edición.

Otro problema era que el programa sólo funcionaba si *Microsoft Visual Basic 6* y *Cristal Reports 8.5* (El cual era el único compatible con las bases de datos de *Microsoft Access XP*) estaban instalados en el mismo sistema operativo. Obviamente esto provocaba un conflicto de versiones de programas que no podían convivir de manera adecuada, y aunque *Visual Basic 6* contaba con un empaquetador, que en teoría, debía crear un empaquetado de cualquier compilación, incluyendo los componentes y referencias en los proyectos, el programa era demasiado grande como para poder crear un empaquetado sin

generar un error. Al parecer existe un componente incompatible de las referencias con *Cristal Reports* y el Objeto ***Active X*** de *Outlook* que provocaban el error al generar dicho empaquetado.

Esto era sin duda un complejo problema que requería atención inmediata; y la decisión fue tomar cartas en el asunto, comenzando por obtener licencias legales de todos los programas necesarios para poder crear el sistema de información y poder ser dueños de los derechos del software. Aunque se contaba con los códigos fuentes del programa, era imperativo que se contara con el compilador. Pero al desear obtenerlo, ya se habían liberado las versiones de los compiladores de la plataforma .Net de Microsoft, por lo tanto, ya no era posible comprar una licencia nueva de *Microsoft Visual Basic 6*, y fue entonces que se tomó la decisión de migrar el sistema a la plataforma *.Net*.

Visual Basic era sin duda la plataforma donde se tenía mayor experiencia en el desarrollo de interfaces vinculadas a bases de datos, además de muchos componentes de Formularios de *Windows*, y también el único donde se podía vincular al ***Active X*** de *Outlook*; pero antes de decidir de cambiar de plataforma, se realizó un estudio de factibilidad, el cual arrojaba que la mayoría de los componentes de los formularios de Visual Basic se conservaban en esencia, y además también la forma de crearlos y usarlos; el problema es que el componente ***ADODB*** había desaparecido casi por completo, este había sido desplazado por una nueva tecnología llamada ***ADO .Net***, que no mantenía nada de la forma en que se enlazaban los datos con ***ADODB*** y mucho menos en como se utilizaban los mismos dentro de un programa; por lo tanto, el hacer una migración resultaba casi imposible; de hecho, *Visual Basic .Net* posee un programa especial para migrar proyectos de la versión anterior, pero este programa es extremadamente limitado; tanto, que no se podría utilizar a nivel profesional, sólo para pequeños, pero muy pequeños proyectos limitados a unos cuantos botones que no tuvieran ningún componente *Active X*, y esto incluía los objetos *Cristal Reports* y los de *Outlook*.

De tal forma que la palabra *"migrar"* se podría descartar del proyecto; el trabajo a realizar era crear un nuevo sistema basado exclusiva y explícitamente para la plataforma *.Net de Visual Basic*. Pero entonces surgió la pregunta obvia de ¿Por que utilizar la plataforma *.Net* para crear un proyecto que en teoría resultaría completamente nuevo?. Era entonces viable hacer un estudio más profundo acerca de las plataformas existentes, dadas las necesidades y la tecnología con que se contaba para la implementación del nuevo sistema. En resumen resultaba un poco ostentoso poder cambiar de plataforma si se tenía tanta experiencia en *Visual Basic* y se había adquirido un razonamiento de como atacar algunas cuestiones especificas, como el manejo del programa a través de los diseños ya establecidos de las pantallas, la validación de datos y los formularios establecidos para la

plataforma Windows, que resultaban ser desarrollados casi obligatoriamente, ya que el único sistema operativo que conocía y manejaba el personal de la AMD era Windows, eso sin tomar en cuenta que ya se tenía un conocimiento del uso de la versión anterior de la aplicación AcadMexDerm.

Entonces el cambio de plataforma no era necesariamente un programa nuevo; la migración automática estaba completamente descartada, pero dado que ya se tenía un programa lo suficientemente avanzado como para tener que crear otro a partir de cero, entonces por lo menos se contaba con las pantallas, los elementos de programación para las validaciones y otros objetos que funcionarían de la misma forma que en la versión anterior, como los correos electrónicos y los reportes; por lo tanto, la nueva plataforma difería en mayor parte en el acceso a los datos a través de una nueva tecnología llamada ***ADO .Net.***

2.7 .Net Framework y ADO.Net como una Plataforma de Desarrollo

Microsoft .Net es una plataforma completa que comprende servidores, clientes y servicios como un conjunto de aplicaciones que se integran para conformar una infraestructura y las herramientas para crear y poner en funcionamiento una nueva generación de nuevos servicios como sistemas operativos *Windows .Net (Server, Advanced Server, Datacenter Server)*, servidores de bases de datos como *SQL Server*, correo electrónico con *Microsoft Exchange Server*, motores de transformación de datos basado en *XML* como *Microsoft Biz Talk Server*, servicios de comunicación como *MSN* y *Microsoft .Net Messanger Service*, servicios de autenticación como *Microsoft Passport* o herramientas de desarrollo incluidas en *Visual Studio .Net*, sólo por mencionar algunos.

La plataforma de desarrollo de Microsoft está basada en un nuevo sistema de programas cuyas bases están sustentadas como un grupo de aplicaciones que se distribuyen como servicio y no sólo como un software fijo y desconectado del mundo llamado ***.Net Framework.*** Esta nueva plataforma se basa en un nuevo modelo de programación y un completo conjunto de clases diseñadas para simplificar el desarrollo de aplicaciones para Windows, Web y dispositivos móviles que proporciona compatibilidad total para los sistemas de Microsoft, que incluyen sólidas características de seguridad y ofrecen nuevos niveles de capacidad de programación y flexibilidad a los programadores, cuyos códigos son convertidos a un lenguaje intermedio de Microsoft ***(MSIL: Microsoft Intermediate Language)*** y compilados en tiempo de ejecución por el ***CLR (Common Language Runtime o Entorno Común de Ejecución)*** para crear aplicaciones en código nativo de la plataforma local, como ***Intel***, para que el mismo código

nativo pueda ser revisado, verificar la seguridad del mismo y recolectar los objetos para los cuales no existe ninguna referencia *(recolección de basura)*, además de gestionar las excepciones, entre otras tareas.

Para comprender mejor este concepto, es necesario explicar que *Visual Studio .Net* es un sólo entorno de desarrollo que incluye diversos lenguajes de programación como *Visual Basic*, *C++*, *C#* y *Visual J#* (sólo por mencionar los más importantes). Cualquier código escrito en cualquiera de los lenguajes disponibles para el *.Net Framework* será compilado a un lenguaje intermedio llamado *MSIL (Microsoft Intermediate Language)*, el cual será posteriormente convertido a código nativo por el *CLR (Common Language Runtime)* y cuando se ejecute por primera vez, el resultado será un programa con mejor rendimiento, con el cual se tendrá más control sobre los errores *(excepciones)* y la seguridad del mismo.

El *Runtime de Lenguaje Común* o *CLR* es la capa del .Net Framework que se encarga de los servicios básicos de .Net, tales como la administración de memoria. recolección de los elementos no utilizados, el control estructurado de las excepciones y del subprocesamiento múltiple. A diferencia de los lenguajes convencionales de programación, los compiladores .Net no producen código nativo que pueda inyectarse de manera directa al ***CPU (Unidad Central de Proceso)*** y ser ejecutado por ésta; en su lugar, se produce el ***Lenguaje Intermedio de Microsoft (MSIL)***, que es una especie de lenguaje de máquina asociado con un procesador virtual que no corresponde con ningún *CPU* comercial. Este Lenguaje Intermedio es un lenguaje orientado a una pila, que no direcciona directamente los registros del *CPU*, sino que tiene en cuenta conceptos de más alto nivel como excepciones y creación de objetos, así como la posibilidad de ser desensamblado directamente desde cualquier ejecutable creado dentro del entorno .Net Framework.

Es así como *Microsoft* desea simplificar el desarrollo de aplicaciones distribuidas como servicios de manera tal que no importe en que lenguaje se programe, ni tampoco si se usan clases específicas de otro lenguaje, es decir, si se hiciera una clase basada en *C#* y se deseara usar como clase heredada desde Visual Basic .Net, no habría impedimento alguno; y para que dicho propósito sea cumplido, todos los lenguajes contenidos en .Net Framework deben de cumplir con la ***Especificación de Lenguaje Común o CLS (Common Language Specification)***, que es una especificación de los tipos de datos, estructuras y operaciones comunes a todos los lenguajes de programación; esto para que el código escrito en un lenguaje pueda utilizarse en otras aplicaciones escritas en otro diferente; lo que significa que *Microsoft* pone a nuestra disposición una serie de lenguajes que funcionan mejor de manera conjunta y que pueden acceder a

clases base incluidas dentro de .Net Framework como componentes comunes que realizan diversas funciones que van desde el manejo de cadenas o números hasta el cifrado y el acceso a redes, dicho conjunto es llamado *Biblioteca de Clases Base o BCL.*

Para poder apreciar lo anterior de manera más explícita, se puede hacer la gráfica mostrada en la ***figura 2.8***, que abarca el concepto de desarrollo en la plataforma .Net; en ella se han dividido las capas en que se fracciona el entorno .Net Framework.

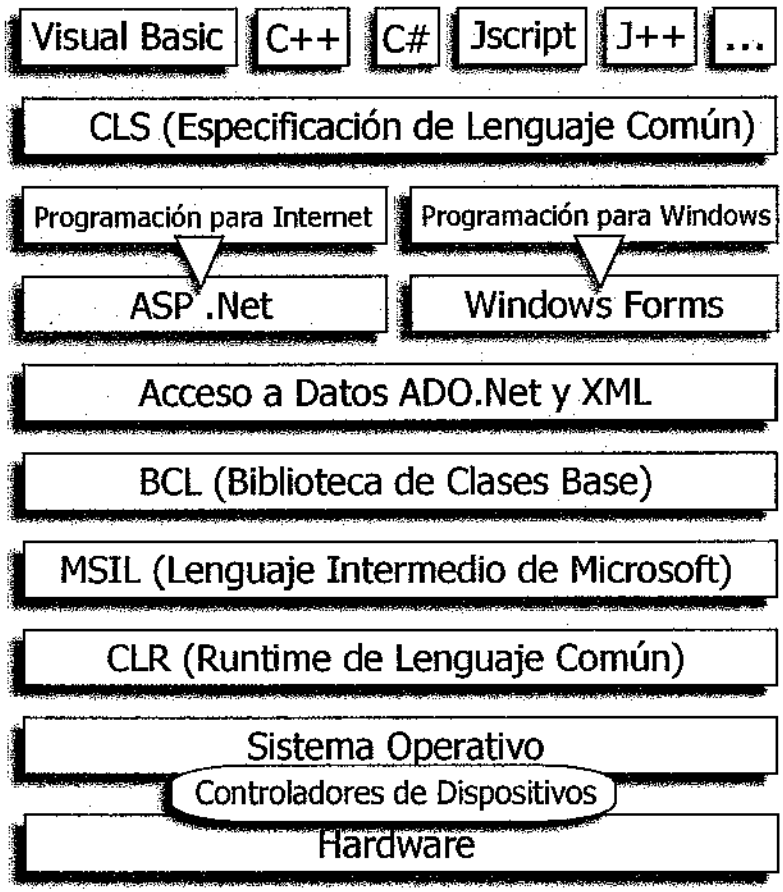


Figura 2.8 → Capas contenidas en .Net Framework

Todas las aplicaciones escritas con Visual Basic .Net se ejecutan en el contexto de *Windows .Net Framework* y pueden aprovechar la capacidad total de las clases que proporciona esta plataforma. En teoría *Microsoft* ha conservado la coherencia semántica dentro del lenguaje para garantizar que la mayor parte del código escrito con *Visual Basic 6.0* mantenga el mismo significado; esto quiere decir que los elementos del lenguaje se siguen conservando, tal como el manejo de los elementos simples, como las variables, sus declaraciones y asignaciones, el control de flujo de datos, operadores, funciones integradas, cálculos, procedimientos, etc.

Sin embargo, *Visual Basic* se ha convertido en un lenguaje completamente orientado a objetos, eso sin mencionar que actualmente *Visual Basic .Net,* así como todos los lenguajes incluidos en Visual Studio .Net, son lenguajes basados en ***CLI (Command Line Interface* o *Interfase de Línea de Comandos)*** y que amplía su funcionalidad a aplicaciones de escritorio con formularios Windows Forms y aplicaciones para Internet a través de formularios Web Forms, y también la posibilidad de crear aplicaciones para dispositivos inalámbricos compatibles con *Internet* y *Pocket PC.*

El acceso a datos con Visual Basic .Net ha cambiado radicalmente de como se accedía en su versión anterior; en Visual Basic 6.0 básicamente existían tres formas de acceder a datos:

- DAO (Objeto de Acceso a Datos)
- RDO (Objetos de Datos Remotos)
- ADO (Objeto de Datos Active X)

La manera más eficaz de acceder a los datos de una base era a través de *ADO*, ya que contaba con una tecnología superior a cualquiera de las otras dos. *ADO* accedía a los datos de forma remota y con una conexión permanente que se podía configurar para poder dar permisos diversos de acceso, dependiendo de las necesidades. A partir de esta base, se contaba con un cursor especial llamado ***RecordSet***; este cursor permitía recorrer los registros, recuperarlos, editarlos o generar uno nuevo a partir de una consulta simple de ***SQL (Structured Query Language* o *Lenguaje Estructurado de Consulta).***

ADO utilizaba una conexión directa a muchas de las bases de datos comerciales como ***Access***, ***Oracle***, ***Informix***, ***Sybase***, entre otras, utilizando un controlador especialmente diseñado para *ADO* conocido como ***controlador OLE DB***, el cual permitía conectarse directamente a una base sin necesidad de utilizar los controladores ***ODBC (Open DataBase Connectivity)*** de Windows, que al mismo tiempo, eran soportados por ADO; de tal forma que se podía realizar la conexión a prácticamente cualquier base de datos dentro de Windows.

Sin embargo, dicho método presentaba diversos inconvenientes en cuanto a su funcionamiento, el principal era que al estar siempre conectado, depende mucho del tipo de cursor que se elige, será el tipo de acceso a los registros, es decir, existen varios cursores que únicamente pueden recorrer la base de inicio a fin, sin poder retroceder, y al hacer cualquier transacción, se tiene que recorrer la base por completo. ADO también es incompatible con XML como un medio de intercambio de datos con otras plataformas. Otro inconveniente de ADO es que está basado en objetos COM; COM es el "modelo de objetos" fundamental sobre el que se generan controles Active X y *OLE (Object Linking and Embedding o Vinculación de Objetos e Incrustación; OLE es un mecanismo que permite a los usuarios crear y editar documentos que contengan elementos u "objetos" creados en distintas aplicaciones.)*. COM permite que un objeto exponga su funcionalidad a otros componentes y se aloje en otras aplicaciones, por lo que el acceso a datos a través de *Visual Basic Script* o *Visual C++* es más difícil de lo que debe.

Sin embargo ADO .Net es una capa de .Net Framework que permite a todos los lenguajes implantados en el mismo, acceder de la misma forma básica y así poder obtener la misma funcionalidad de acceso sin importar la plataforma en la que se este desarrollando. Desde la perspectiva del programador, la diferencia más importante entre ADO y ADO .Net es que este último ya no soporta los cursores de avance (*recordsets*); y es por esto que se puede dificultar la escalabilidad de una aplicación desarrollada en Visual Basic 6.0 a la plataforma .Net.

ADO .Net integra tres formas principales de conexiones con bases de datos, con las cuales extiende una nueva forma de conectarse directamente con Microsoft SQL Server:

- ***Proveedor de datos OLE DB .Net***, este proveedor permite acceder a una fuente de datos para la que existe un proveedor OLE DB, igual como lo hacía ADO.

- ***Proveedor de datos SQL Server .Net***, este proveedor ha sido escrito específicamente para acceder a SQL Server7.0 o posterior, utilizando ***Tabular Data Stream (TDS)*** como medio de comunicación. *TDS* es el protocolo nativo de *SQL Server*, por lo que obviamente ofrece mejores prestaciones que el proveedor de datos *OLE DB* que se utilizaba antes de ADO .Net.

- ***Proveedor de datos ODBC .Net***, este proveedor funciona como un puente a una fuente ODBC, por lo que se puede utilizar para acceder a cualquier origen para el que exista un proveedor ODBC.

ADO .Net incorpora un nuevo método para trabajar con bases de datos; lo más trascendental de este método es el hecho de que se puede almacenar una copia de una consulta completa a una base de datos, sin importar el tamaño, para trabajar directamente en un nuevo objeto llamado ***"dataset"***, el cual será el huésped de dicha consulta. A través de él se puede realizar cambios, adiciones, borrados y actualizaciones sobre los registros, agregando la posibilidad de deshacer el cambio realizado, sin importar cual sea. De esta manera se pueden realizar una serie infinita de transacciones sobre un caché de la base de datos estando desconectado del origen; y de igual manera, se pueden actualizar los cambios realizados sobre ésta a la original cuando se le indique.

Imaginando por un momento que el sistema requiere conectarse a una base de datos remota que se encuentra en Monterrey; al estar en el Distrito Federal se hace la conexión y se obtienen los registros necesarios (o las tablas completas si así se desea) por lo que la conexión es únicamente para obtener el caché, inmediatamente después de obtenerla, se permanecerá desconectado durante todo el día y sólo se necesitará la conexión para actualizar los cambios cuando se desee; esto significaría que no se necesita una conexión permanente a un servidor de datos para poder trabajar, sin mencionar la drástica reducción del uso de ancho de banda de la conexión si se requiriera permanecer conectado todo un día o más tiempo.

Aunque ADO .Net sigue teniendo un modo conservador de conexión permanente, el modo desconectado muestra una nueva forma sin precedente de trabajar. Al obtener una tabla, no sólo se obtienen los registros, también se obtiene toda la información de la tabla, como los datos de las columnas y sus tipos de registros, tamaños o cualquier especificación que se requiera para trabajar; sin mencionar que también se podrán obtener varias tablas dentro de un *DataSet* y relacionarlas para poder hacer transacciones más complejas, sin importar el tipo de relación que se necesite, por más complicada que ésta sea. ADO .Net ofrece pues el objeto *DataSet* para obtener un caché en memoria de una base de datos para poder realizar cambios o ediciones sobre la misma, pero también brinda otro objeto llamado *DataReader* que mantiene la misma funcionalidad de desconexión que el *DataSet* con la diferencia de que los registros insertados en este objeto son únicamente para lectura o consulta.

Para comprender mejor el funcionamiento de ADO .Net se puede observar la ***figura 2.9*** que muestra el contexto completo de la forma de trabajar de esta nueva tecnología.

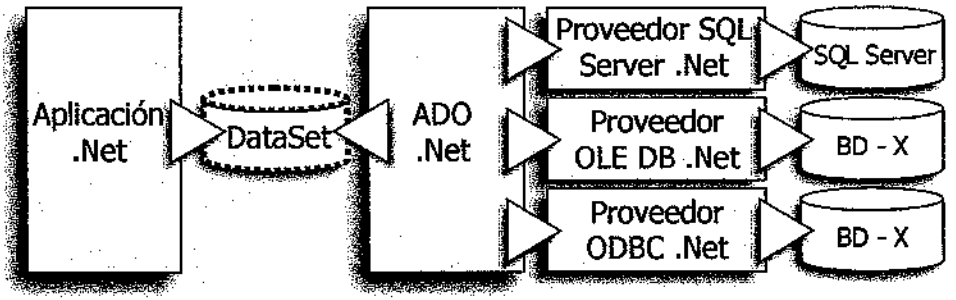


Figura 2.9 → Contexto Completo de ADO .Net

La idea detrás de ADO .Net y el *DataSet* es que la aplicación cuente con todos los datos necesarios para poder trabajar, y por ninguna razón requiera conectarse a la base de datos, salvo para cargar o actualizar los cambios generados en el *DataSet*. Temporalmente los datos pueden ser almacenados al disco duro donde se encuentre el programa directamente a un archivo XML, bastará con utilizar el método *WriteXml* y *WriteXmlSchema* del objeto *DataSet*. Con esto se estará volcando toda la base de datos, incluyendo su estructura, es decir, definición de tablas, columnas, tipos de datos, relaciones, etc. Básicamente no se necesita la base de datos para realizar transacciones completas dentro de la computadora cuando los datos se encuentren en un servidor remoto, sin importar que tan lejos este o cuanto tiempo este accesible para que el servicio esté activo.

En resumen, se puede afirmar que con los formularios Windows Forms, los programadores que utilicen Visual Basic .Net pueden crear aplicaciones basadas en Windows que aprovechen las características completas de interfaz de usuario disponibles en el sistema operativo Windows. Visual Basic .Net permite enfrentarse a cualquier escenario de acceso a datos con facilidad, proporcionando compatibilidad para obtener un acceso a cualquier base de datos de forma flexible y altamente escalable. Visual Basic .Net admite construcciones completas orientadas a objetos para permitir código con más componentes y más reutilizable. Las características del lenguaje incluyen implementación total de herencia, encapsulación y polimorfismo. Visual Basic .Net aporta también a sus aplicaciones seguridad integrada, acceso directo a .Net Framework y la capacidad de orientar las aplicaciones a una amplia gama de dispositivos móviles y el nuevo modelo de seguridad de .Net Framework proporciona un control exhaustivo sobre la seguridad de la aplicación y a las excepciones del mismo.

2.8 Factibilidad Operativa y Estimación de Recursos

Dadas las amplias funciones y opciones que ofrece Visual Basic .Net para facilitar el desarrollo del sistema, y cubriendo todas las necesidades mencionadas, la decisión final para la plataforma de desarrollo fue Visual Basic .Net. Esta plataforma ofrece una amplia gama de herramientas para un desarrollo más profesional que su antecesor, dado que se reunifica como parte de una plataforma mucho más robusta, que es Visual Studio .Net, que integra varios lenguajes con un mismo fin, y la idea de que se puede desarrollar en cualquier lenguaje, ya sea Visual Basic, C# o C++ para desarrollo de Aplicaciones Windows.

Dadas las necesidades, era imperativo que el programa contara con un sistema de reportes (etiquetas) integrado durante el desarrollo, para esto existían dos opciones, la adquisición de la licencia de *Visual Basic .Net* individualmente y adquirir el *Crystal Reports 8.5* o superior por separado, y por otro la versión de Visual Studio .Net Professional ofrecía una versión especial de Crystal Reports llamada *Cristal Reports para Visual Studio .Net* integrada al Visual Studio, y en este paquete que superaba por muy poco el costo de los programas por separado, con la ventaja de que el Visual Studio .Net incluía los lenguajes C#, C++ y J++ en un sólo entorno de programación, aparte de contar también con la librería de desarrollo de *Microsoft MSDN (Microsoft Developer Network)* incluida en un paquete de 7 Discos.

Cuando se presentó el costo del paquete a la mesa directiva fue bien recibida, dada la idea de obtener la legalidad y los derechos de distribución del sistema para los laboratorios, y dadas las ganancias que representaba poseer el sistema con una nueva versión mejorada en su funcionalidad interna y externa por lo que fue aceptada desde el comienzo. El costo de la versión Professional de Visual Studio .Net fue de $ 8,921.74 +IVA, una cantidad aceptable, dado que no sólo se podrían realizar desarrollos de manera interna para la AMD.

Los tiempos necesarios para terminar el proyecto del sistema de información pretendían cubrir los meses de Abril, Mayo y Junio de 2004, desde la adquisición del software, hasta la implementación, pruebas y su liberación, para que durante la Sesión Mensual de Julio del mismo año fuera presentado el proyecto ante la Mesa Directiva, la cual tendría acceso al mismo para una versión redistribuible para sus respectivos consultorios. La versión 6 de este sistema debía estar en completa operatividad para recibir la información del próximo evento a realizar por la AMD, el XVI Congreso-Jornadas en Provincia, el cual se llevaría a cabo del 15 al 18 de Septiembre del 2004 en Ixtapa Zihuatanejo.

El estudio y la programación de la nueva página de Internet de la AMD se llevo a cabo para su implementación en el mes de Abril. Así que después de desarrollar por completo la página de Internet, la atención se centraría por completo en el desarrollo de la nueva versión del sistema de información, y durante dicho tiempo, no se interrumpiría la producción en las oficinas, dado que la versión anterior del sistema estaba en completo funcionamiento.

2.9 Definición de Módulos

Para comenzar con el diseño y la programación del sistema de información, se inició por la división de los módulos más importantes que integran al programa. Estos módulos fueron concebidos a partir de las principales secciones de los sistemas de las versiones anteriores, basándose específicamente en la funcionalidad y capacidad de cada uno de ellos; éstos fueron ilustrados en la ***figura 2.7***, pero se ampliará más cada uno de ellos y se explicará su propósito individualmente. Posteriormente hay que avocarse al diseño, estructuración, normalización y programación de cada uno de ellos en el mismo orden que son presentados y divididos a continuación:

1. **Conexión a la base de datos:** Este módulo comprende la conexión a la base de datos, valga la redundancia, sus tablas, las relaciones que persisten entre ellas y su enlace con los formularios de Windows integrados para los Miembros y los Congresos respectivamente. Estos formularios deben de poseer la función de poder desplazarse a través de la base de datos y sus registros.

2. **Interfaz de Datos:** Este módulo abarca toda la interfaz Gráfica de la ventana principal del sistema de información, desde el diseño de las pantallas, hasta la edición, adición y eliminación de registros; incluyendo también el apartado para los Congresos.

3. **Evasión de Duplicidad:** Como se ha mencionado anteriormente, se ha generado un código único e induplicable para cada uno de los miembros que existen dentro del sistema, en este módulo se verá como el código interviene directamente en casi todos los procesos que tienen que ver con la edición, búsqueda, e impresión de los registros de los miembros dentro de la aplicación, y también se verá como es que a través de él podemos evitar la duplicidad del registro de los miembros, al momento de intentar registrar uno nuevo.

4. **Validación de Datos:** En este módulo se verá como poder validar los datos que se ingresan en los diferentes formularios antes de ser vaciados a la base de datos a partir de las nuevas herramientas disponibles para este propósito dentro de Visual Basic .Net, así como su diseño e implementación para nuestra aplicación.

5. **Registro de Asistencias:** El registro de las asistencias es parte fundamental para el programa, se basa en una interfaz de *casillas de verificación* (*checkbox's* ☑) que interactúan con el usuario para poder registrar y al mismo tiempo visualizar de una manera sencilla la asistencia de cada uno de los miembros; pero internamente, el programa debe tener la capacidad de interpretar un número real y transformarlo de tal forma que sea fácil de digerir para el usuario.

6. **Consultas:** En el mundo de las bases de datos es bien conocido que existe la posibilidad de generar un subconjunto especial del universo de los datos con una herramienta llamada *consulta*; dichas consultas se generan, dependiendo del gestor, a través del *Lenguaje Estructurado de Consultas SQL* o a través de un diseñador de consulta especial del mismo gestor de bases de datos que se utilice. ADO.Net ofrece un gestor de visualización de datos llamado *DataView* que permite de una manera sencilla realizar dichas consultas, pero se debe realizar una interfaz al usuario la cual se pueda programar para que genere dicha consulta a través de la interacción del usuario, entonces, se creará el propio generador de consultas gráfico y accesible para cualquiera que tenga conocimientos mínimos de Windows.

7. **E-Mail:** Uno de los propósitos principales del sistema es poder enviar correos electrónicos de forma masiva a todos los miembros que tengan una dirección de correo válida y que al mismo tiempo reciban la misma información generada por un editor especial de correo que contenga las mismas opciones que ofrecería cualquier proveedor de correo, es decir, poder emitir un asunto, un mensaje y adjuntar uno o más archivos de cualquier tipo. Una propiedad muy especial y específica del sistema es que al generar una consulta, ésta afecte al sistema de correos electrónicos, de tal forma que el sistema sea bastante flexible en el aspecto de a quién se quiere enviar un correo electrónico.

8. **Rejillas de Visualización y Clasificación:** El sistema debe contener un sistema especial de clasificación y ordenamiento con el cual se pueda ordenar y ubicar a los miembros contenidos en la base, el sistema de visualización que ofrece todas estas posibilidades es un objeto de Windows llamado ***DataGrid (Rejilla de Datos)*** con el cual se puede obtener la máxima operatividad de los registros de una forma clara y visiblemente sencilla. El sistema de consultas también debe de afectar de manera directa los datos que se mostrarán en el *DataGrid*.

9. **Búsquedas:** La aplicación debe ofrecer un sistema de búsquedas para los miembros a partir de sus nombres; este sistema debe ser muy sencillo, ya que por lo regular a los usuarios les genera confusión los sistemas de búsqueda complejos y muy parametrizados en sus métodos de consulta; es por esto que es de suma importancia que las búsquedas requieran de la menor información necesaria para realizar una exploración sin tener que sacrificar exactitud durante el proceso.

10. **Impresiones:** La aplicación debe contener una serie de sistemas de impresión que dependen directamente de la información contenida en la base de datos, estos documentos deben ser capaces de imprimir la información más recientemente ingresada en el sistema, es decir, si se acaba de registrar a un nuevo miembro, las impresiones deben de ser afectadas por dicho registro, y este mismo deberá aparecer dentro de los tirajes de impresiones generales con su información recientemente registrada. Sin duda alguna el documento más importante a imprimir es el tiraje de etiquetas; el cual debe de poder realizar cuatro principales tirajes, la de los Miembros AMD-FMD, los miembros AMD, OTROS y TODOS (Todos los Registros). Otros documentos son los de los "reportes generales", los cuales son, como antes se ha mencionado, reportes impresos de los registros de la base de datos. El programa deberá poder imprimir también documentos como los datos completos de un miembro que sea ubicado de forma individual, y por último pero no menos importante, la impresión de los historiales de los miembros; esta sección es también de suma importancia, ya que será este documento una forma de mostrar todo lo referente con los registros de asistencia de cada uno de los médicos, y dicha información debe de ser veraz y confiable, ya que de esto depende el ingreso de nuevos miembros a la AMD, y será de esta forma que se corroborará la asistencia de los miembros a los eventos, y cubrir con esto un requisito muy importante para poder ingresar a la AMD.

2.10 Diseño y Programación

El diseño de las pantallas obedece a las versiones anteriores, que han funcionado de manera aceptable hasta ahora; el diseño se integra en una única ventana que contiene los diez módulos antes mencionados, divididos entre si por cinco principales secciones:

- Miembros
- Congresos
- Consultas
- E-Mail
- Rejilla

Dichas secciones son separadas entre si por pestañas de un control especial llamado ***TabControl*** de Windows y contenidas en una sola ventana de Windows. Dicho formato había funcionado hasta ahora, ya que una de las metas es que exista una integración completa de los módulos.

Dentro de la programación en Windows existen dos formas de presentar a las ventanas, la primera es como una *Caja de Diálogo*, la cual se presenta como una única ventana que contiene botones, menús, cajas de diálogo, etc. Todos estos elementos contenidos en una ventana única, como por ejemplo, la ventana que contiene los elementos del Explorador de Windows o el *Windows Explorer*, el cual utiliza una sola ventana para contener los elementos que se visualizan mientras se navega por Internet, una barra de herramientas simple y un menú, pero no requiere más ventanas contenidas dentro del mismo Explorer para contener información o documentos especiales, por ejemplo, si se desea mostrar alguna información al usuario, entonces se muestra una ventana que esta separada de la ventana principal.

La otra forma de presentar a las ventanas es una serie de formularios contenidos dentro de una interfaz especial llamada ***MDI*** *(Multiple Document Interfaz* o *Interfaz de Múltiples Documentos)*. Las aplicaciones MDI permiten mostrar varios documentos al mismo tiempo, cada uno de ellos en su propia ventana y contenidas en una única ventana que funge como contenedor.

La interfaz del sistema será una ventana *MDI*, que contendrá una ventana principal que será en si la interfaz con la base de datos, es decir, el sistema de información; los demás documentos serán Reportes de Crystal Reports que contendrán los tirajes de etiquetas y los reportes generales. Dicha interfaz sustituirá el método de acceso a los reportes, el cual era hasta la versión anterior una simple llamada a una aplicación totalmente separada del sistema, es decir, anteriormente los

reportes y todos los sistemas de impresión estaban separados del programa principal, esto principalmente a causa de que no existía forma de mantener los reportes actualizados sin tener que salir del programa para que se actualizaran de nuevo, aunque se mantuviera una conexión permanente a la base de datos, pues simplemente era inevitable. Pero ahora es posible tener los reportes y las etiquetas actualizadas aún cuando no se cierre la aplicación, y esto es gracias a que en el nuevo sistema *ADO.Net* es posible crear reportes a partir de los registros contenidos en el *DataSet* que se mantienen en memoria caché sin tener que acceder a la base de datos original.

En base a un diseño de programa *MDI* junto con la integración de la Ventana del sistema de información, se puede comenzar a diseñar y programar los módulos del Sistema por separado, comenzando por realizar la conexión a la base de datos.

Dicha conexión obedece a un sistema que posteriormente será empaquetado para su distribución, pero es importante destacar que el programa depende directamente de una base de datos que se mantiene en constante actualización; dicha Base no estará incluida dentro del empaquetado, ya que este mismo archivo será actualizado bajando un programa especial desde la página de Internet; pero de esto se hablará más adelante, por lo pronto se inicia con la descripción de los módulos individualmente.

Para continuar con el estudio completo de la programación de los módulos, se ha documentado una sección especialmente creada para el programador que está interesado en saber como es que se han programado los diferentes módulos de la aplicación, esto es con el fin primordial de que la programación no quede perdida entre el texto del diseño, es decir, al poder consultar específicamente las secciones de programación a las que se hace referencia con ejemplos que se desarrollaron de la misma forma en que se programaron los módulos de la aplicación ***AcadMexDerm*** se podrá observar de manera detallada el proceso en que se creó el sistema, y de la misma forma, se podrán aplicar estos métodos para el beneficio de aplicaciones profesionales a través de ejemplos sencillos que reflejan perfectamente los pasos que se siguieron durante el desarrollo. Así pues, se pueden aplicar las diversas soluciones que se dieron dentro del sistema para crear los módulos aplicando los más nuevos métodos de programación que *Visual Basic .Net* ofrece para el desarrollo de aplicaciones de una manera sencilla y con el más alto profesionalismo a los proyectos reflejando lo que se hizo para crear ***AcadMexDerm .Net 6.0***.

2.10.1 Conexión a la base de datos

Las bases de datos son herramientas increíblemente útiles para almacenar grandes cantidades de datos de forma eficaz. Permiten modificar y trabajar con todo tipo de información y son una de las herramientas más importantes de la computación. Las bases de datos proporcionan la infraestructura requerida para los sistemas de apoyo a la toma de decisiones y para los sistemas de información estratégicos, ya que estos sistemas explotan la información contenida en las bases de datos para lograr mayores ventajas competitivas.

Las bases de datos ofrecen una serie de ventajas ineludibles en su forma de almacenar información, la globalizan de tal forma que permiten a los diferentes usuarios que acceden a un mismo sistema considerar la información como un recurso corporativo que carece de dueños específicos. Eliminan información redundante, es decir, evitan duplicidades. Eliminan información inconsistente y permiten compartir información entre varios sistemas o usuarios que las pueden utilizar al mismo tiempo como una misma entidad. Permiten mantener la integridad en la información y gozan de independencia de datos; dicha independencia implica un divorcio entre programas y datos; es decir, se pueden hacer cambios a la información que contiene la base de datos o tener acceso a las mismas de diferentes maneras, sin hacer cambios en las aplicaciones o en los programas.

Dichas ventajas se ven reflejadas directamente en los Sistemas de Información que acceden a los datos para utilizar su contenido y mostrarlos de una forma coherente y organizada al usuario; y al mismo tiempo este observará deducciones lógicas de los datos que son normalizados por el sistema de tal forma que le permitan comprenderlos de la manera más sencilla posible.

Las bases de datos han existido de una forma u otra desde antes que se generara Visual Basic como un lenguaje Visual. La larga historia de las bases de datos y su uso en programas de cómputo ha asegurado que Visual Basic tenga acceso a las bases de datos desde sus primeras versiones; por ello, la tecnología de acceso a datos en este ha evolucionado con el tiempo, y ha pasado por diversas versiones antes de la aparición de la tecnología .Net.

ADO.Net es la tecnología principal de acceso a datos que forma parte del .Net Framework; siendo este su medio primordial para obtener datos cuando se desarrolle en *Visual Basic .Net*, en el cual, se obtiene acceso mediante una ***clase*** especial del espacio de nombres del .Net Framework derivada de ***System.Data*** como se muestra en la ***figura 2.10***.

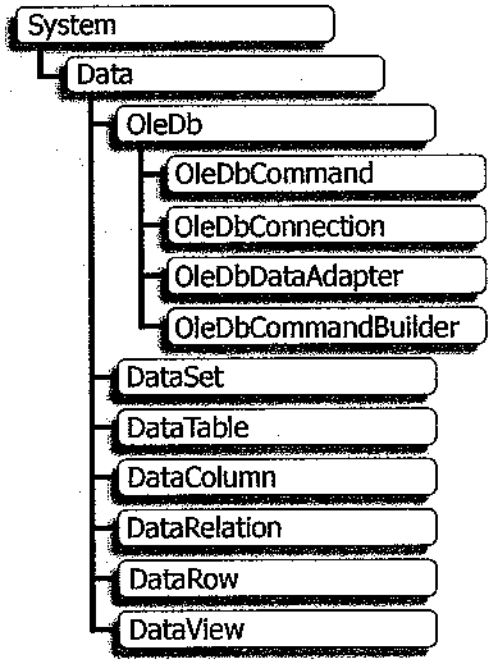


Figura 2.10 → Espacio de nombres del conjunto de datos derivados de la clase *System.Data*

Este espacio de nombres *(System.Data)* es una clase especial derivada del .Net Framework que se divide principalmente en dos áreas distintas: los conjuntos de clases ***System.Data.OleDB*** y ***System.Data.SQLClient.*** El primero, *System.Data.OleDB*, está diseñado para permitirle la conexión con cualquier base de datos para la que se tenga un proveedor **OLE DB** o (mediante el proveedor **OLE DB** para **ODBC**) un controlador ODBC, y es técnicamente el equivalente a la capa de acceso a datos **ADO** original de Visual Basic 6. El segundo, *System.Data.SQLClient*, esta diseñado para funcionar sólo con ***Microsoft SQL Server*** y ofrece una funcionalidad similar a las clases OLE DB.

Una de las mayores ventajas que ofrece ADO.Net con respecto otros tipos de conexiones a bases de datos es que no depende de conexiones continuamente activas. En las aplicaciones tradicionales ***cliente/servidor***, los componentes establecen una conexión con una base de datos y la mantienen abierta mientras la aplicación está en funcionamiento.

Existen diferentes razones por lo que dicho enfoque no resulta práctico en muchas aplicaciones como:

- Las conexiones abiertas de base de datos consumen valiosos recursos del sistema. En la mayoría de los casos, las bases de datos sólo pueden mantener un pequeño número de conexiones concurrentes. La sobrecarga que supone mantener estas conexiones reduce el rendimiento general de la aplicación.

- Por la misma razón, las aplicaciones que necesitan una conexión de base de datos abierta resultan extremadamente difíciles de ampliar. Una aplicación difícil de ampliar puede funcionar aceptablemente con cuatro usuarios, pero no es probable que lo haga con decenas. Las aplicaciones vinculadas a ADO.Net pueden ser fácilmente escalables, porque el tráfico con el servidor de bases de datos se realiza durante un periodo de tiempo muy corto.

- En un modelo basado en acceso a datos continuamente conectados se puede hacer difícil y poco práctico el intercambio de datos entre los límites de las aplicaciones y la organización por medio de una arquitectura conectada. Si dos componentes necesitan compartir los mismos datos, es necesario que estén conectados o idear un medio para que los componentes intercambien datos.

Por lo anterior, el acceso a datos con ADO.Net se diseñó en torno a una arquitectura que utiliza conexiones con moderación. Las aplicaciones se conectan a la base de datos sólo durante el tiempo necesario para extraer o actualizar los datos. La base de datos ya no contiene conexiones permanentes que la mayor parte del tiempo permanecen inactivas, así que pueden dar servicio a muchos más usuarios.

ADO.Net ofrece varias ventajas sobre las anteriores versiones de ***ADO*** y sobre otros componentes de acceso a datos. Estas ventajas se incluyen en las siguientes categorías:

- **Interoperabilidad con XML**: Las aplicaciones ADO.Net pueden aprovechar la flexibilidad y la amplia aceptación de ***XML (eXtensible Markup Language o Lenguage de Marcado Extensible),*** dado que XML es el formato de transmisión de conjuntos de datos a través de la red; cualquier componente que pueda leer el formato XML podrá procesar los datos generados por ADO.Net; esto quiere decir que los componentes de datos de una aplicación .Net podrán intercambiar datos con cualquier otro componente de cualquier otra aplicación, siempre que este componente

entienda XML. Se están escribiendo muchas aplicaciones que comprenden XML, lo que ofrece un nivel sin precedentes de intercambio de datos entre aplicaciones diferentes.

XML se basa en texto, por lo tanto, la representación XML de los datos no utiliza información binaria, lo que permite enviarla mediante cualquier protocolo, como por ejemplo HTTP. La mayor parte de los servidores de seguridad bloquean la información binaria; sin embargo, si el formato de la información es XML, los componentes pueden continuar intercambiando fácilmente la información; y para dicho efecto no es necesario saber XML para utilizar datos en ADO.Net, ya que este convierte automáticamente los datos a XML y viceversa según sea preciso; simplemente se interactúa con los datos utilizando métodos de programación ordinarios.

- **Programabilidad:** En Visual Studio, los componentes de datos ADO.Net encapsulan funcionalidad de acceso a datos de diversas formas que ayudan a programar de modo más rápido y con menos errores.

- **Disponibilidad:** Dado que en la ***Web*** puede incrementar en gran medida la demanda de datos, la dispinibilidad adquiere una importancia crucial. Las aplicaciones para Internet tienen un suministro ilimitado de usuarios potenciales. Aunque una aplicación pueda dar un buen servicio a una docena de usuarios, eso no significa que pueda dar un servicio igualmente bueno a cientos o cientos de miles de ellos. Una aplicación que consuma recursos tales como bloqueos y conexiones de base de datos no dará un buen servicio a un gran número de usuarios, porque la demanda de recursos limitados por parte de los usuarios acabará por superar su disponibilidad.

Los componentes de ADO.Net están diseñados para separar el acceso a los datos de su manipulación. ADO.Net tiene dos componentes principales que cumplen esta función: el ***DataSet*** y el proveedor de datos de .Net Framework, que es un conjunto de componentes entre los que se incluyen los objetos especiales de acceso a datos ***Connection***, ***Command***, ***DataReader*** y ***DataAdapter***.

El ***DataSet*** de ADO.Net es una representación de datos residente en memoria que proporciona un modelo de programación relacional coherente independientemente del origen de datos que contiene. Un *DataSet* representa un conjunto completo de datos, incluyendo las tablas que contienen, ordenan y restringen los datos, así como las relaciones entre las tablas. El *DataSet* es el componente central de la arquitectura sin conexión de ADO.Net; también está expresamente diseñado para el acceso a datos independientemente de su origen. Como resultado, se puede

utilizar con múltiples y distintos orígenes, como *XML*, un servidor de datos locales o en una red distante. El *DataSet* contiene una colección de uno o más objetos ***DataTable*** formados por filas y columnas de datos, así como información sobre claves principales, claves externas, restricciones y relaciones relativas a los datos incluidos en los objetos ***DataTable***. El ***DataTable*** representa una tabla de datos relacionales en la memoria, se puede crear y usar de manera independiente o lo pueden usar otros objetos de .Net Framework, normalmente como miembro de un objeto *DataSet*.

El objeto ***Connection*** proporciona conectividad con un origen de datos. El objeto ***Command*** permite tener acceso a comandos de base de datos para devolver datos, modificarlos, ejecutar procedimientos almacenados y enviar o recuperar información sobre parámetros. El objeto ***DataReader*** proporciona una secuencia de datos de alto rendimiento desde el origen de datos; y por último, el objeto ***DataAdapter*** proporciona el puente entre el objeto *DataSet* y el origen de datos. El ***DataAdapter*** utiliza objetos ***Command*** para ejecutar comandos ***SQL*** en el origen de datos tanto para cargar el *DataSet* con datos como para reconciliar en el origen los cambios aplicados a los datos incluidos en el ***DataSet***.

Existe un objeto especial llamado ***OleDbConnection*** que representa una conexión exclusiva a un origen de datos. En el caso de un sistema de bases de datos *cliente/servidor*, equivale a una conexión de red al servidor. La principal propiedad asociada a un objeto de conexión *OleDbConnection* es la propiedad ***ConnectionString***, que consiste en una cadena con pares de atributo/valor con la información necesaria para iniciar una sesión con la base de datos. Una propiedad *ConnectionString* típica podría tener el siguiente aspecto:

```
[oledb]
; Everything after this line is an OLE DB initstring
Provider=SQLOLEDB;Data Source=localhost;Integrated
Security=SSPI;Initial Catalog=Northwind
```

Los pares atributo/valor que *OleDbConnection* utiliza con más frecuencia están representados también por separado, por medio de propiedades individuales tales como *DataSource* y *DataBase*. Cuando se trabaja con un objeto de conexión, se puede establecer la propiedad *ConnectionString* como una sola cadena de conexión. También se puede establecer la propiedad *ConnectionString* en la ruta de acceso de un *Archivo de Vínculo de Datos Microsoft Data Link (con extensión *.udl)*.

Dicho método de conexión es utilizado en el sistema de información *AcadMexDerm* para el acceso a sus dos principales tablas: *Miembros* y *Congresos;* se conectan a la aplicación utilizando esta metodología de acceso a datos, como se muestra en la ***figura 2.11***.

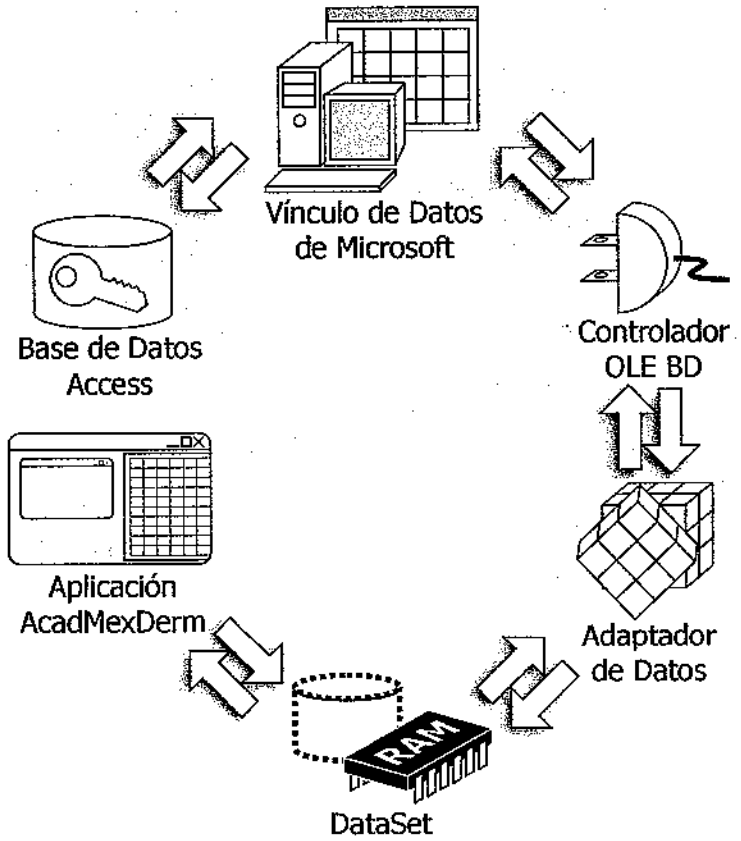


Figura 2.11 → Metodología de Acceso a Datos del sistema de información

En el ***apartado 2.12.1*** se muestra a través de un ejemplo muy sencillo pero significativo la forma en que se realiza esta conexión y como es que se puede hacer uso de un DataSet para una aplicación *Visual Basic.Net*. Dentro de la aplicación existen veintiun relaciones sencillas entre las dos tablas: *Miembros* y *Congresos* ***(ver tablas 2.3 y 2.4)***, como se ha mencionado con anterioridad,

dichas relaciones persisten con el propósito de conservar los historiales de cada uno de los miembros y relacionarlos con los nombres de los Congresos y Talleres a los que asistieron (***ver figura 2.5***).

El Adaptador de Datos (***DataAdapter***) representa un conjunto de comandos de datos y una conexión de base de datos que se utilizan para rellenar el *DataSet* y actualizar el origen de datos.

Cada proveedor de datos incluido en .Net Framework tiene un objeto *DataAdapter*. el proveedor de datos de .Net Framework para OLE DB incluye un objeto ***OleDbDataAdapter***, el proveedor de datos de .Net Framework para *SQL Server* incluye un objeto ***SqlDataAdapter*** y el proveedor de datos de .Net Framework para ODBC incluye un objeto ***OdbcDataAdapter***. Para recuperar datos de un origen de datos y llenar las tablas de un *DataSet*, se utiliza un *DataAdapter*. El *DataAdapter* sirve también para reflejar en el origen de datos los cambios efectuados en el *DataSet*.

Una clase derivada del espacio de nombres ***System.Data*** es la clase ***DataView,*** esta clase representa una vista personalizada que puede enlazar datos de un ***DataTable*** para ordenación, filtrado, búsqueda, edición y exploración de los registros; al mismo tiempo permite editar, adicionar y borrar elementos de las tablas vinculadas al mismo. Es este objeto el que se usa primordialmente para enlazar los registros de las tablas al sistema de información. Para poder mostrar los registros en el formulario del sistema de información se utilizan cajas de texto normales que sirven de contenedores de visualización y edición de dichos registros; en la ***tabla 2.5*** se puede ver el listado completo de los mismos.

Nombre	Tipo	Registro	Tabla	Descripción
ComboGen	ComboBox	Genero	Miembros	Género del Miembro
TxtNom	TextBox	Nombre	Miembros	Nombre
TxtApp	TextBox	ApP	Miembros	Apellido Paterno
TxtApm	TextBox	ApM	Miembros	Apellido Materno
TxtCod	TextBox	Codigo	Miembros	Código Único
TxtCalle	TextBox	Calle	Miembros	Calle
TxtCol	TextBox	Colonia	Miembros	Colonia
TxtCp	TextBox	CP	Miembros	Código Postal
ComboPa	ComboBox	Pais	Miembros	País
ComboEst	ComboBox	Estado	Miembros	Estado
ComboDel	ComboBox	Delegacion	Miembros	Delegación o Municipio
ComboDelCon	ComboBox	DelegacionCon	Miembros	Delegación Consultorio
ComboCon	ComboBox	Condicion	Miembros	Condición
ComboCedula	ComboBox	Cedula	Miembros	Tipo deCédula Profesional

TxtNoCedula	TextBox	NoCedula	Miembros	Número de Cédula Profesional
ComboMem	ComboBox	Miembro	Miembros	Tipo Miembro
TxtCURP	TextBox	CURP	Miembros	Clave Única de Registro Poblacional
ComboEsp1	ComboBox	Especialidad_1	Miembros	Primera Especialidad
TxtEsp2	TextBox	Especialidad_2	Miembros	Segunda Especialidad
TxtTelCasa	TextBox	Tel_Casa	Miembros	Teléfono Casa
TxtC1	TextBox	Tel_Consu1	Miembros	Primer Teléfono Consultorio
TxtC2	TextBox	Tel_Consu2	Miembros	Segundo Teléfono Consultorio
TxtFax	TextBox	Fax	Miembros	Teléfono Fax
TxtCel	TextBox	Cel	Miembros	Teléfono Celular
TxtBee	TextBox	Beeper	Miembros	Teléfono Beeper (Localizador)
TxtRfc	TextBox	RFC	Miembros	Registro Federal de Causantes
TxtEma	TextBox	EMail	Miembros	Correo Electrónico
TxtNumCon	TextBox	NumCon	Congresos	Número de Congreso
TxtNumT	TextBox	NumT	Congresos	Número de Talleres
ComboFecha	DateTimePicker	Fecha	Congresos	Fecha del Congreso
TxtNomCon	TextBox	NomCon	Congresos	Nombre del Congreso
TxtT1	TextBox	Nt1	Congresos	Nombre Taller 1
TxtT2	TextBox	Nt2	Congresos	Nombre Taller 2
TxtT3	TextBox	Nt3	Congresos	Nombre Taller 3
TxtT4	TextBox	Nt4	Congresos	Nombre Taller 4
TxtT5	TextBox	Nt5	Congresos	Nombre Taller 5
TxtT6	TextBox	Nt6	Congresos	Nombre Taller 6
TxtT7	TextBox	Nt7	Congresos	Nombre Taller 7
TxtT8	TextBox	Nt8	Congresos	Nombre Taller 8
TxtT9	TextBox	Nt9	Congresos	Nombre Taller 9
TxtT10	TextBox	Nt10	Congresos	Nombre Taller 10

Tabla 2.5 → Controles vinculados a los registros de las tablas

Todos los controles anteriormente mencionados estarán vinculados con los registros de la base de datos y fungirán como contenedores de visualización y edición de los diferentes registros. Los controles estarán distribuidos en la ventana del sistema de información contenida como una caja de dialogo dentro del recipiente *MDI* del programa principal. Para poder acomodar ordenadamente todos los elementos de la ventana, se usará un control de Windows muy útil para dicho propósito llamado ***TabControl***. El *TabControl* muestra múltiples fichas, similares a los divisores de un cuaderno o a las etiquetas de las carpetas de un archivador. Las fichas pueden contener imágenes y otros controles, que para este caso, contendrán los elementos de los formularios para *Miembros* y *Congresos*; se puede

utilizar el control de fichas para crear cuadros de diálogo con varias páginas como los que suelen aparecer en Windows, por ejemplo, en el control *Pantalla* del *Panel de Control* de *Windows*. La parte más importante del *TabControl* son los *TabPages*, que contienen las fichas individuales. Cada ficha individual es un objeto *TabPage*. Para el sistema de información se utilizaron seis *TabPages* en los cuales se dividieron las seis principales funciones del sistema de información (***ver tabla 2.6***).

Nombre del TabPage	Sección	Función Principal
TabMiembros	Miembros	Contenedor para el formulario de los Miembros, en este recipiente se podrán realizar cualquier acción referente a la tabla de los miembros, desde búsquedas hasta ediciones, eliminación y adición de nuevos registros.
TabCongresos	Congresos	Contenedor para el formulario de los Congresos, en este recipiente se podrán realizar cualquier acción referente a la tabla de los congresos, desde búsquedas hasta ediciones, eliminación y adición de nuevos congresos.
TabConsultas	Consultas	Desde este TabPage se podrán realizar consultas especiales, para poder examinar a los Miembros seleccionados por: ✓ Estado (Estado de la República Mexicana) ✓ Condición ✓ Miembro (AMD, AMD-FMD, OTRO) ✓ Especialidad (Primera Especialidad) ✓ Número de Congreso (Asistencia o No Asistencia a cualquier congreso ingresado en el sistema)
TabEmail	e - mail	Contenedor del Formulario para Enviar Correos Electrónicos Individuales o Masivos (Debemos recordar que el sistema de Correos Electrónicos es afectado por las Consultas).
TabRejilla	Rejilla	Contenedor del Control *DataGrid* con los registros de la Tabla de los Miembros.
TabImpresiones	Impresiones	Contenedor del Sistema de Impresiones.

Tabla 2.6 → Funciones principales de los TabPages del sistema de información

Los controles de edición del *TabPage* de Miembros se enlazan con los registros de las tablas de la base de datos como se muestra en el ***apartado 2.12.2***. Es de este modo que se realiza la conexión a la base de datos y se les ha vinculado a los controles de la ventana. Esta vinculación es directa y no necesita que se realice un pre-proceso especial para que los datos sean mostrados, pero esto no aplica para los registros de los historiales.

Se debe recordar que el almacenamiento de los registros de los historiales de cada uno de los miembros es a través de una serie de columnas especiales que contienen dicha información de forma numérica, pero dichos registros no pueden

ser mostrados al usuario en forma de números, ya que no podrían ser analizados ni editados. Para este propósito se ha creado una zona especial en el *TabPage* de *Miembros* con los controles listados en la ***tabla 2.7***.

Nombre	Tipo	Descripción
TxtNConAct	TextBox	Contenedor del registro **NumCongAct** de la tabla *Miembros*.
Ch	CheckedListBox	Contenedor en el que se representará a través de casillas de verificación el registro **AsisAct** de la tabla *Miembros*.

Tabla 2.7 → Descripción de los controles para el registro de asistencia

La zona para el registro de asistencia contiene dos registros, el del *número de congreso actual* y el de la *asistencia actual;* el registro de la asistencia actual se realiza a través de once casillas de verificación y son únicamente para el registro de asistencia a los talleres del congreso al que se encuentra inscrito actualmente; posteriormente, al inscribirse a un nuevo evento, este dato junto con el registro *NumCongAct* pasarán a ser parte del historial y tomarán el siguiente espacio vacío dentro del área de los historiales, campo que será señalado por el registro *NumHis*, el cual cambiará automáticamente aumentando e indicando el siguiente espacio vacío para el próximo historial editado. Para poder ver el proceso del registro de los historiales se puede seguir el diagrama de flujo en la ***figura 2.12***, que muestra el procedimiento que se sigue para registrar las asistencias.

Existen once casillas de verificación para registrar la asistencia actual, este registro es convertido a forma binaria como se muestra en las ***tablas 2.1 y 2.2***, posteriormente es convertido a un número entero y guardado en el registro ***AsisAct*** de la tabla *Miembros*. Es importante mencionar que dicho registro guarda la asistencia a los talleres que se imparten en cierto congreso, pero cada congreso puede tener un máximo de diez talleres dentro del sistema; pero son guardados once registros por una sencilla razón, también es guardado el registro de la asistencia al propio congreso, es decir, para la AMD no basta haber cubierto la cuota de asistencia, sino que es necesario haber asistido personalmente al evento para poder obtener la retribución por dicha asistencia, retribución que se traduce en puntos para ingresar a la asociación; de esta forma es que la primera casilla de verificación mostrará si se ha asistido al evento o no.

Dentro de las consultas debe de existir la posibilidad de examinar a los miembros que hayan asistido o no a cierto evento, y para ello simplemente habría que buscar los miembros que tengan como registro de asistencia al evento un número cuyo primer ***bit*** sea un dato positivo (o uno lógico), es decir, un número que en forma binaria sea mayor que $\#01111111111_b$, dicho número en decimal es $\#1023_d$, por lo tanto, si se desea consultar a los miembros que hayan asistido efectivamente al

congreso, hayan o no acudido a algún taller, simplemente se buscarán los registros que tengan en *AsisAct* un número decimal igual o mayor a $\#1024_d$.

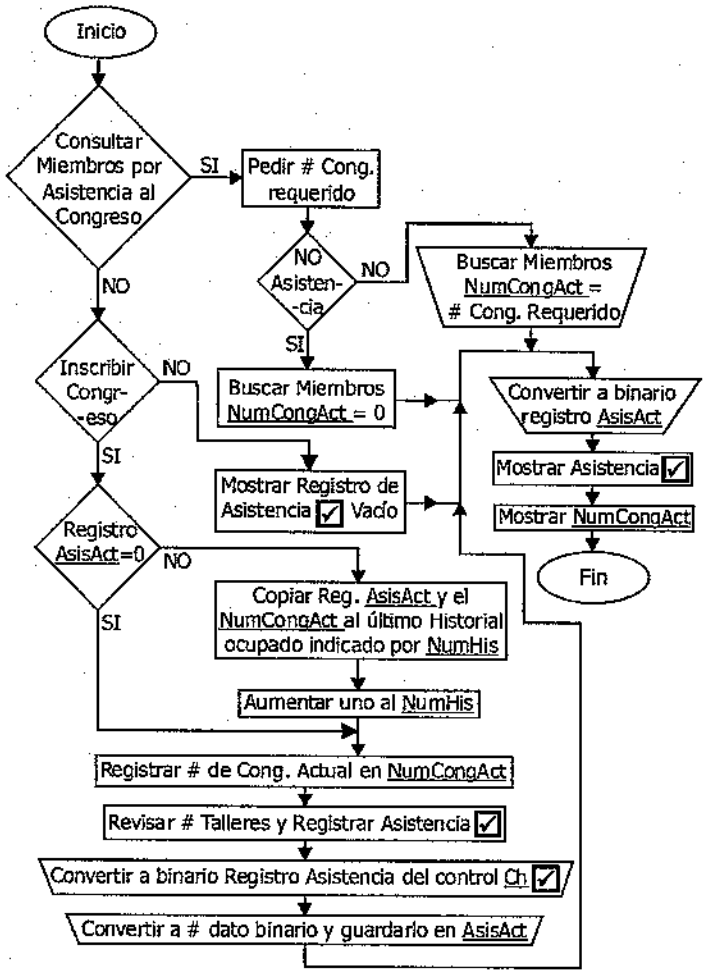


Figura 2.12 → Diargrama de flujo del registro de asistencias

Para finalizar el módulo de conexión, el sistema deberá ser capaz de desplazarse a través de los registros del *DataView;* para este propósito se debe crear una sección especial con botones para poder recorrer los registros uno por uno y de principio a fin; también se necesitará un indicador que muestre el registro en el que se está posicionado actualmente. Para poder gozar de dichos mandos, se debe integrar a la aplicación los controles listados en la ***tabla 2.8.***

Nombre	Tipo	Propiedad	Valor	Descripción
BInicio	Button	Text	Inicio	Botón para desplazarse al primer registro del *DataView*.
BPrev	Button	Text	Anterior	Botón para desplazarse al registro Anterior del *DataView*.
BNext	Button	Text	Siguiente	Botón para desplazarse al Siguiente registro del *DataView*.
BFinal	Button	Text	Final	Botón para desplazarse al último registro del *DataView*.
LabelNumReg	Label	Text	(Vacío)	Etiqueta que muestra el número de registro actual con respecto al total de registros.

Tabla 2.8 → Controles para cambiar de posición de registro

Los controles de la ***tabla 2.8*** tienen la facultad de desplazarse a través de los miembros como se ejemplifica en el ***apartado 2.12.3***, donde se muestra el funcionamiento explicito de los botones y su funcionamiento interno.

2.10.2 Interfaz de Datos

Hasta ahora se ha visto como está vinculada la base de datos al Sistema de Información; es momento de mostrar como es que se puede interactuar con ella. En la interfaz de los formularios se han dividido a través de los *TabPages* los apartados para los *Miembros* y para los *Congresos*; en las pantallas para cada uno de ellos se integran la mayor parte de las funcionalidades que existen para crear nuevos registros, editarlos, borrarlos, realizar búsquedas, etc. Para dicho propósito, el elemento primordial para interactuar con el usuario son botones.

Los botones representan una sencilla forma de limitar las funciones que integra un programa, dicha funcionalidad es indicada de forma clara y restringida en cuanto a cometer algún error, es decir, en la versión anterior de la aplicación existían muchas pantallas que carecían de la funcionalidad para *Cancelar* alguna operación, pero las ventanas nos permiten saber que es lo que desea exactamente el usuario, al tener conocimiento de que es lo que realiza en la pantalla, como presionar un cierto botón con el cursor del Mouse, presionar la tecla *Escape* o *Enter*. Las funciones presentadas por formularios Windows nos permiten limitar las posibilidades de respuesta del usuario, ya que precisamente es esta funcionalidad la que nos deja mostrarle únicamente las opciones que existen, y no podrá realizar otra acción, delimitándose a presionar cierto botón o cierta tecla.

La opción de cancelar alguna operación se ha integrado como una funcionalidad más a esta nueva versión, ya que es posible cancelar alguna edición o búsqueda; y para integrar todas las funciones para el apartado de los *Miembros* se han agregado los controles listados en la ***tabla 2.9***.

Nombre	Tipo	Propiedad	Valor	Descripción
BNew	Button	Text	Nuevo	Crea un nuevo registro para un nuevo Miembro.
BEdit	Button	Text	Editar	Edita los datos del Miembro actual.
Bsave	Button	Text	Grabar	Guarda los cambios generados al Miembro.
BSeek	Button	Text	Buscar	Busca un Miembro por Nombre y Apellidos.
BDel	Button	Text	Borrar	Elimina el registro del Miembro.
BRefr	Button	Text	Refrescar	Refresca los registros de los Miembros y los ordena alfabéticamente por Apellido Paterno.
BPrint	Button	Text	Imprimir	Imprime los datos completos del Miembro.
BCancel	Button	Text	Cancelar	Cancela una operación de adición o edición de un registro.
BAct	Button	Text	Actualizar	Actualiza los cambios generados en el *DataSet* directamente a la base de datos.
BHistorial	Button	Text	Historial	Examina el historial del Miembro y los muestra en el *TabPage* de Impresiones.
MMenuMiembros	MenuItem	Text	Miembros	Muestra el menú de las opciones para el apartado de los Miembros.
MMiembroActualizar	MenuItem	Text	Actualizar Miembro al Congreso Actual	Inscribe al Miembro al congreso actual y los datos anteriores los traslada a su historial.
MMiembroHistorial	MenuItem	Text	Examinar Historial del Miembro	Realiza la misma acción que el botón **BHistorial.**
MMiembroClipboard	MenuItem	Text	Copiar Datos del Miembro al Portapapeles	Copia los datos personales de dirección para crear una etiqueta de correo y la pone en el portapapeles de Windows.

Tabla 2.9 → Controles para la Interfaz de Datos del *TabPage* de los Miembros.

Los controles de la ***tabla 2.9*** nos muestran todas las funciones que se pueden realizar para la edición, adición y eliminación de los registros de los Miembros.

Dichas opciones siguen un patrón ordenado para realizar cada una de sus acciones, este orden es formal y sigue un proceso muy simple para poder realizar algún cambio dentro de la tabla principal. Estos cambios requieren una validación preliminar antes de poder grabar los datos ingresados por el usuario a la base de datos original.

Las opciones que puede realizar el usuario dentro del apartado de los Miembros están registradas en el Diagrama de Flujo de la ***figura 2.13***. Este diagrama nos presenta un panorama más amplio de las funciones que se pueden realizar para la edición y actualización de los registros de los Miembros.

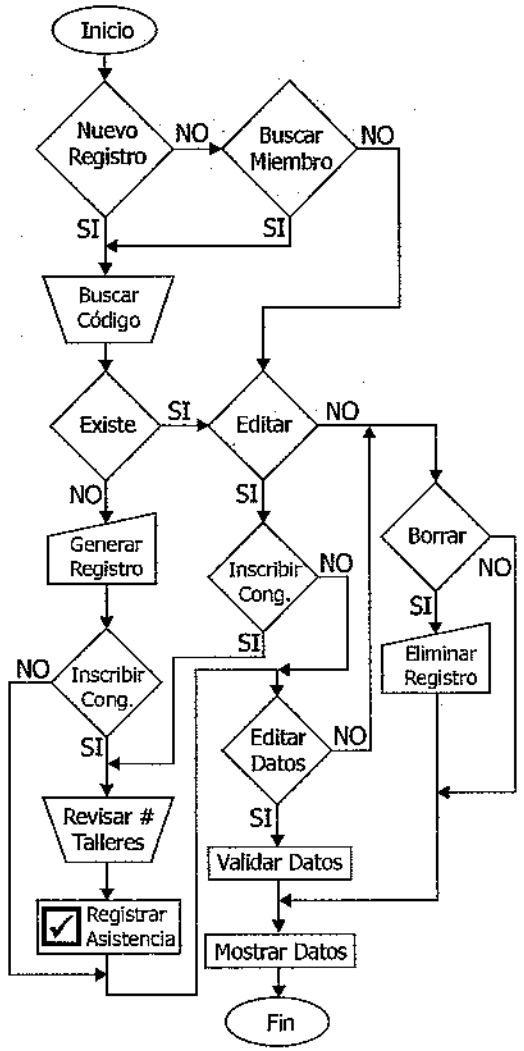


Figura 2.13 → Diagrama de Flujo para las opciones dentro del apartado de los Miembros.

La edición de los registros para el apartado de *Miembros* se realiza de la misma forma que se ejemplifica en el ***apartado 2.12.4***, en donde se puede observar detalladamente como interactuar con los registros del *DataSet* de las aplicaciones y es de esta forma que el *sistema de información* realiza cambios en los registros de la tabla *Miembros*. La ventana principal del sistema de información se encuentra ilustrada en la ***figura 2.14:***

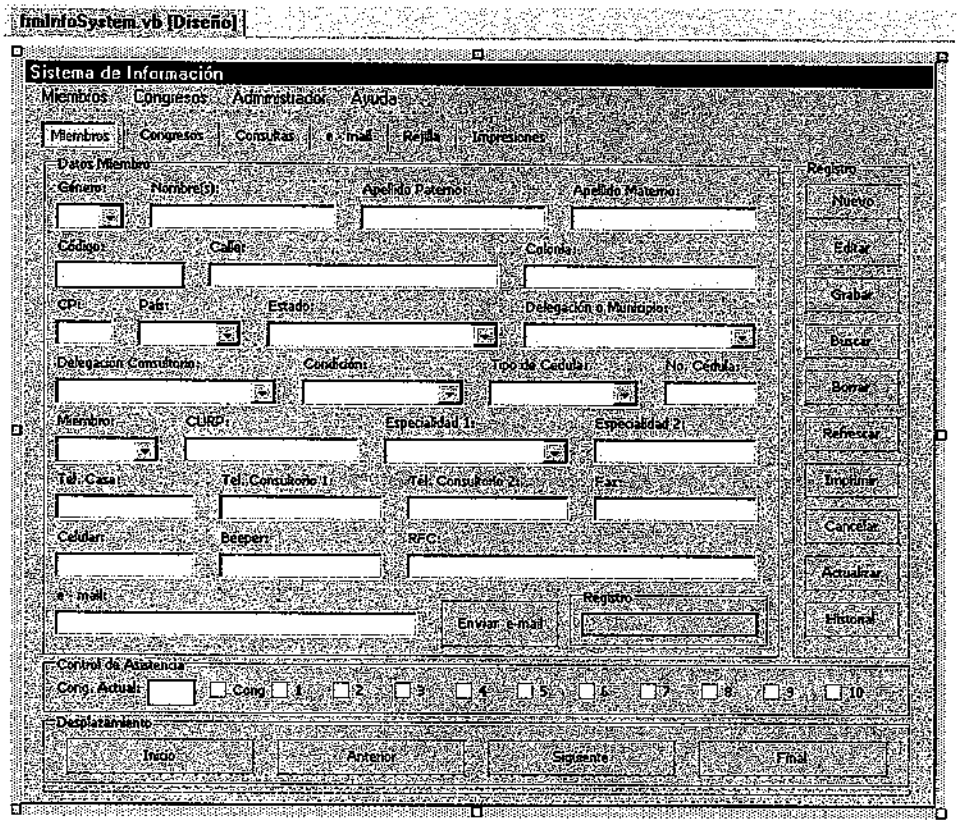


Figura 2.14 → Ventana Principal del TabPage-Miembros del sistema de información

Dentro del sistema de información se utiliza un objeto llamado *DataView* que permite crear diferentes vistas de los datos almacenados en el *DataSet*; este *DataView* tiene la capacidad de exponer los datos de una tabla con distintos criterios de ordenación y al mismo tiempo poder filtrar los datos por el estado de alguna fila o basándose en una expresión de filtro. Todos las cajas de texto de la ***tabla 2.5*** están vinculadas al *DataView* y no al *DataSet* por varias razones; la primordial estriba en que al realizar una consulta especial no se tiene que

programar un procedimiento especial para que este formulario sea afectado por la consulta, y como se puede apreciar en el ***apartado 2.12.4*** sirven también para poder tener un conducto de visualización de los registros y al mismo tiempo tener la capacidad de editarlos sobre el mismo formulario.

Los procedimientos de adición, edición y eliminación de registros para el apartado de los *Congresos* se asemeja casi idénticamente al de los miembros, con la peculiar diferencia que mientras se editen, agreguen o eliminen congresos no se podrán realizar cambios en la tabla de *Miembros*, hecho que se deriva de que al realizar cambios a los registros de los *Congresos* se afectará de manera directa a los historiales de los miembros; y es por esta razón que existe una protección simple pero efectiva para dicho propósito, simplemente se evita el pueda cambiar de *TabPage* durante la edición de los congresos. Se han creado dos menús especiales para poder entrar en modo de edición de los congresos, estas opciones están listadas en la ***tabla 2.10***.

Nombre	Tipo	Pro-piedad	Valor	Descripción
MMenuCongresos	MenuItem	Text	Congresos	Muestra el menú de las opciones para el apartado de los Congresos.
NTC	MenuItem	Text	Navegar a través de los congresos	Entra en modo Edición para el apartado de Congresos.
NoNTC	MenuItem	Text	Dejar de navegar a través de los congresos	Termina el modo Edición para el apartado Congresos.

Tabla 2.10 → Menú de opciones para el apartado Congresos

Al impedir que el usuario edite los registros de los miembros mientras se realizan cambios sobre los congresos, se evita crear conflictos posteriores en los historiales de los miembros. Al terminar de editar los registros de los *Congresos*, se finalizará el modo de edición de los mismos y automáticamente se podrá ubicar como *Congreso Actual* al último congreso registrado. La ventana para el apartado de los *Congresos* se ilustra en la ***figura 1.15*** en la cual se puede observar que este apartado es accesible solo dentro del *TabPage-Congresos*.

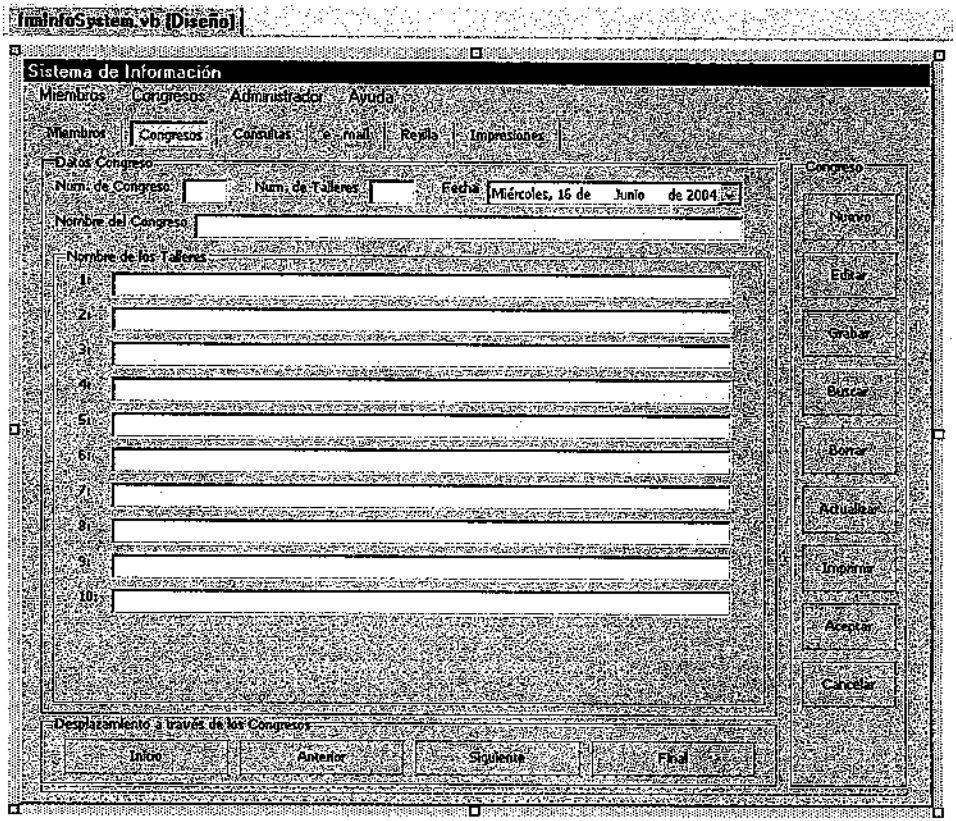


Figura 2.15 → Ventana Principal del apartado Congresos *(TabPage-Congresos)*

Por último se abordará la eliminación de registros de los *Miembros*; este procedimiento resulta sumamente sencillo, sin embargo, existen algunas precauciones que es posible tomar antes de eliminar un registro; el objeto *DataSet* permite restaurar un elemento borrado e incluso interactuar con él como si fuera un elemento mas, pero con una diferencia, una simple propiedad cuyo valor indica que es un registro eliminado, esta propiedad se llama *DataRowState*. Los valores para esta propiedad se encuentran enumerados en la ***tabla 2.11.***

Nombre de Miembro	Descripción	Valor
Added	La fila se ha agregado a *DataRowCollection* y no se ha llamado a *AcceptChanges*.	4
Deleted	La fila se ha eliminado mediante el método *Delete* del *DataRow*.	8
Detached	Se ha creado la fila, pero no forma parte de ningún *DataRowCollection*. *DataRow* se encuentra en este estado inmediatamente después de haber sido creado y antes de que se agregue a una colección, o bien si se ha quitado de una colección.	1
Modified	La fila se ha modificado y no se ha llamado a *AcceptChanges*.	16
Unchanged	La fila no ha cambiado desde que se llamó a *AcceptChanges* por última vez.	2

Tabla 2.11 → Valores para la propiedad *DataRowState* de un *DataRow*

Al saber exactamente el estado de cada uno de los registros en el *DataSet* las posibilidades de hacer interacciones entre ellas aumenta drásticamente, sin embargo, para el sistema de información AcadMexDerm no es necesario utilizar esta propiedad, sin embargo, es necesario cerciorarse que el usuario desea en realidad borrar el registro; para este propósito se agregó un procedimiento de borrado de registros prácticamente idéntico al mostrado en el ***apartado 2.12.4*** donde es posible apreciar lo sencillo que resulta eliminar un registro, y de la misma forma, este procedimiento existe para el apartado de los *Congresos*. Sin embargo, existe la posibilidad de crear una bandeja especial para los elementos borrados y poder restaurarlos ya que los registros eliminados permanecerán accesibles hasta que se actualice en el origen de datos, que en este caso, es la base de datos *Access*.

2.10.3 Evasión de Duplicidad

Hasta ahora se ha podido apreciar un amplio panorama de como es que el código interviene en una serie de procesos dentro del sistema de información, siendo su principal motivo de existencia evitar la duplicidad al ingresar nuevos registros dentro de la tabla de los Miembros; sin embargo, ésta no es su única utilidad.

Se ha mencionado, el código es una colección de caracteres y números comprendidos por las dos primera letras el Nombre, las tres primeras letras del Apellido Paterno, las tres primeras letras del Apellido Materno y los últimos cuatro números del Código Postal ***(ver figura 2.16);*** la razón principal de haber

escogido estos datos para extraer de ellos dichos caracteres es simplemente que para la AMD es imperativo contar con estas referencias para ingresar al médico en sus registros.

Figura 2.16 → Código Único para cada Miembro

En particular, se escogió el Código Postal por ser un dato numérico con el cual podemos generar un código único y casi induplicable para cada uno de los Miembros dentro del sistema. El código es parte fundamental dentro del programa y es a través de el que se pueden organizar los registros de manera más rápida en los procesos de búsqueda y ordenación.

Dentro de las bases de datos existe un campo especial llamado ***Clave Principal*** que se define como una columna (o combinación de columnas) que proporciona un valor único para cada fila de la tabla que permite identificar de forma inequívoca cada fila de la tabla, por lo que no pueden haber en una tabla dos filas con el mismo valor en la columna definida como *Clave Principal.* Existen esencialmente tres tipos de claves principales:

- Auto-numérica
- Campo Simple
- Compuesta (Campos Múltiples)

De no poder crear una clave principal a partir de un campo simple se puede crear una compuesta por más de un campo; los campos que representan una *Clave Principal* no contendrán valores nulos y por lo tanto deberán contener algún valor que no se duplicará a lo largo de la vida útil de la base de datos.

Tal vez el usar una clave compuesta sea útil para un sistema en el cual no se debe llevar un control tan exhaustivo para la duplicidad en los registros, pero en el caso de esta aplicación, la evasión de duplicidad es imperativa. El uso dentro del sistema de una clave principal de un único campo facilita y agiliza los procesos de ordenación y búsqueda, ya que si fuera una clave compuesta, el sistema debería combinar los campos durante una búsqueda u otro proceso.

Al insertar un nuevo registro se muestra al usuario una caja de diálogo especial creada para este propósito con la cual podremos realizar una búsqueda previa del código generado para avisarle al usuario que el registro ya existe ***(ver figura 2.17).***

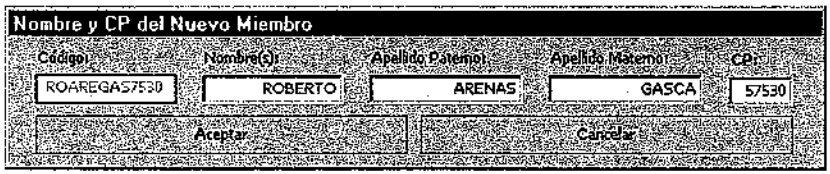


Figura 2.17 → Caja de Diálogo para un nuevo registro

Esta caja de diálogo genera automáticamente el código al ir insertando los campos requeridos, es decir, al ir tecleando los caracteres de los campos, la caja de dialogo iluminada del lado izquierdo le muestra al usuario el código para este nombre.

Esta caja de diálogo también valida los datos y solo acepta como mínimo los caracteres necesarios para generar el código, de lo contrario avisará al usuario que falta algún campo o que la longitud de alguna es menor que la requerida.

Se han escogido únicamente dos caracteres del nombre para abreviaturas como Ma. de María. El género de cada uno de los miembros se insertarán posteriormente dentro del *TabPage* de *Miembros*.

Se han elegido los últimos cuatro números para evitar algún error si el usuario no inserta un cero a la izquierda del código en caso de que el mismo solo estuviera conformado por cuatro números.

Al intentar un Miembro que ya exista, el sistema avisará con una *Caja de Mensajes* ***(MessageBox)*** que el usuario ya existe ***(ver figura 2.18)***, al aceptar dicho mensaje el sistema automáticamente se posicionará en el registro con el código duplicado.

Figura 2.18 → MessageBox de aviso de duplicidad

El código sirve para otras funciones como la búsqueda de algún miembro o para obtener los registros de los diferentes campos del mismo para otros procesos como la impresión de algún reporte individual para dicho Miembro.

Para poder apreciar de manera más gráfica los procesos en los que interviene directamente el código es posible ver el Diagrama de Flujo de la ***figura 2.19;*** dicho diagrama muestra que un código único de identidad para los registros de una tabla facilita procesos que de otra forma generarían errores que se pueden prevenir con el uso de este simple método de identificación que sería ineficaz si tuviera un diferente método de tipificación, como auto-numeración, la cual, genera ambigüedad en un sistema en el que la eliminación de registros es algo cotidiano.

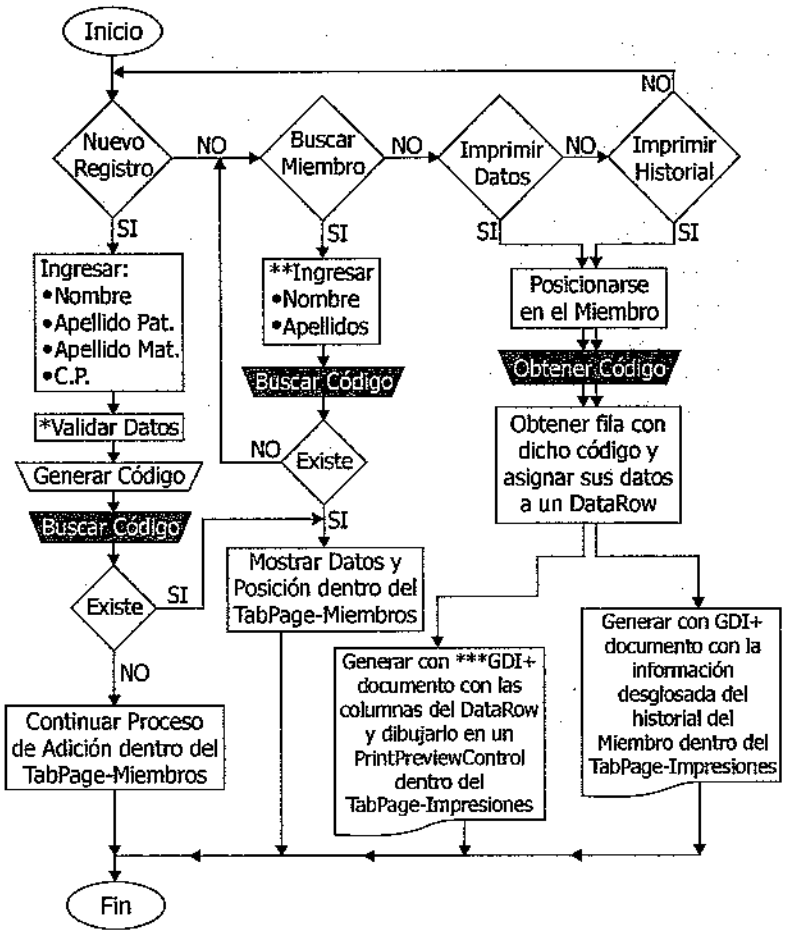


Figura 2.19 → Diagrama de Flujo de los procesos en los que interviene directamente el código único de identificación.

* La validación de datos se realiza previamente al ingreso del registro a la tabla en la Caja de Diálogo de *Nuevo Registro*.

** Para la búsqueda de algún Miembro se muestra una Caja de Diálogo especial que requiere al menos los dos primeros caracteres del Nombre y tres del Apellido Paterno para buscar algún registro que contenga dichos caracteres dentro del código, para más detalles ver subcapítulo 2.10.9.

*** GDI+ es una interfaz de dispositivo gráfico que proporciona gráficos vectoriales de dos dimensiones, imágenes y tipografía y que permite desplegarla independientemente del dispositivo, para más detalles ver subcapítulo 2.10.10.

Un objeto *DataSet* está formado por un conjunto de tablas, relaciones y restricciones. En ADO .Net, los objetos *DataTable* se usan para representar las tablas de un *DataSet*. Un objeto *DataTable* representa una tabla de datos relacionales de la memoria; los datos son locales de la aplicación basada en .Net en la que residen que son llenadas desde un origen de datos mediante un *DataAdapter*.

La clase *DataTable* es miembro del espacio de nombres *System.Data* dentro de la biblioteca de clases de *.Net Framework*. Se puede crear y usar *DataTables* de manera independiente o como miembro de un *DataSet*, y los objetos *DataTable* se pueden utilizar también en combinación con otros objetos de *.Net Framework*, incluido *DataView*. Al conjunto de tablas de un *DataSet* se puede tener acceso mediante la propiedad *Tables* del objeto *DataSet*.

Además de este esquema de trabajo, un objeto *DataTable* debe tener también filas en las que albergar y ordenar los datos. La clase *DataRow* representa los datos reales que contiene una tabla. La clase *DataRow*, sus propiedades y métodos se usan para recuperar, evaluar y manipular los datos de una tabla. Cuando se tiene acceso a los datos de una fila y se desea acceder a ellos de manera individual, el objeto *DataRow* representa una fila única de datos en un *DataTable* y mantiene sus registros en el mismo estado y con las mismas características de la fila original, pero con la diferencia que se puede trabajar con ella de manera individual. Es este formato de trabajo que se utiliza para obtener un registro específico dentro del marco de ADO .Net, pero para obtener una fila en específico habrá que ubicarse en ella si se esta trabajando con un *DataView* o buscarla usando su Clave Principal como referencia de identificación, que en este caso es el Código Único de Identificación.

Es así que se obtienen los datos de un registro y se envía la fila en forma de *DataRow* al procedimiento especial de impresión de datos para que a través de GDI+ genere el reporte listo para imprimirse en un medio externo al sistema, como una impresora o fax que el *Sistema Operativo* tenga configurado.

2.10.4 Validación de Datos

Una parte importante en los Sistemas de Información es la validación de los datos que ingresa el usuario en el formulario antes de ser guardados a la base de datos, dicha validación es importante ya que, dependiendo del sistema, se pueden realizar comparaciones o búsquedas en las que persiste la diferenciación de los caracteres en minúsculas o mayúsculas.

En la mayoría de los sistemas de registro de nombres, existe mucha diferencia entre un nombre escrito con mayúsculas y minúsculas que uno escrito solo con minúsculas, o mayúsculas. También existiría una diferencia abismal en el caso de que no se respetaran los acentos o que se abreviaran los nombres o apellidos.

Las opiniones se dividen en que si el usuario debe o no tener conocimiento de que podría causar un error al introducir cierto dato sin imaginarse que en computación existe un código ascii[1] que representa numéricamente una letra de otra y que no es lo mismo una letra minúscula y una mayúscula. Pero informáticamente hablando, el usuario no tiene la obligación de ingresar los datos de cierta forma o en cierto formato, ya que este no es su trabajo; es pues, trabajo del programador prever cualquier fallo potencial durante el ingreso de datos al sistema.

Existirá también entre los usuarios el conocido debate de que si las letras mayúsculas se acentúan o no, pero sin duda alguna, ortográficamente hablando, si deben de acentuarse todas las vocales que así lo requieran, sean mayúsculas o no; informáticamente hablando, no existe impedimento alguno para no acentuarlas, como era el caso de las máquinas de escribir.

Todos estos casos deben de ser contemplados, en el caso de que se desee hacer un depurado previo al ingreso de registros a la base de datos; para la AMD es importante que se evite la duplicidad en los registros, y por ello se tomó la decisión de registrar únicamente letras mayúsculas en todos los campos de la base de datos. Al usuario del sistema de información simplemente se le podrá recomendar que respete los acentos y que no abrevie los nombres; pero en el caso de que así lo desee, el sistema esta previendo que es posible abreviar el nombre con dos caracteres, y por ello únicamente se toman las dos primeras letras del nombre para vaticinar abreviaturas como Ma. de María.

1.- ASCII. Acrónimo de American Standard Code for Information Interchange *(Código Normalizado Americano para el Intercambio de Información)*. En computación, es un esquema de codificación que asigna valores numéricos a las letras, números, signos de puntuación y algunos otros caracteres. ASCII incluye 256 códigos que representan todas las combinaciones posibles en 8 bits o un byte.

Es así que con mucha frecuencia se comprueba si la información que especifican los usuarios en un formulario Windows o uno Web es válida o no. Por ejemplo, si tiene un control TextBox para un número de teléfono, es posible comprobar que contiene sólo los caracteres correctos (números, paréntesis, guiones, etc.).

En el caso de los sistemas realizados en Visual Basic (hasta la versión 6) era muy complicado realizar una validación a los datos; en la versión anterior del sistema de información AcadMexDerm (5) se realizaba una validación carácter por carácter a través de un método llamado *KeyPress* que ocurría cuando era presionada una tecla cuando algún control en específico tenía el foco, en este caso, las Cajas de Texto.

El sistema tenía desde entonces la fuente *Tahoma* como fuente predeterminada para todos los controles, botones, ventanas, cajas de texto, etc. En la ***tabla 2.12*** se puede revisar el código *ascii* del alfabeto en español para esta fuente.

Dec.	Hex.	Car.	Dec.	Hex.	Car.	Dec.	Hex.	Car.	Dec.	Hex.	Car.
65	41	A	97	61	a	82	52	R	114	72	r
66	42	B	98	62	b	83	53	S	115	73	s
67	43	C	99	63	c	84	54	T	116	74	t
68	44	D	100	64	d	85	55	U	117	75	u
69	45	E	101	65	e	86	56	V	18	76	v
70	46	F	102	66	f	87	57	W	119	77	w
71	47	G	103	67	g	88	58	X	120	78	x
72	48	H	104	68	h	89	59	Y	121	79	y
73	49	I	105	69	i	90	5A	Z	122	7A	z
74	4A	J	106	6A	j	193	C1	Á	225	E1	á
75	4B	K	107	6B	k	201	C9	É	233	E9	é
76	4C	L	108	6C	l	205	CD	Í	237	ED	í
77	4D	M	109	6D	m	211	D3	Ó	243	F3	ó
78	4E	N	110	6E	n	218	DA	Ú	250	FA	ú
79	4F	O	111	6F	o	220	DC	Ü	252	FC	ü
80	50	P	112	70	p	209	D1	Ñ	241	F1	ñ
81	51	Q	113	71	q						

Tabla 2.12 → Código ascii del alfabeto en español en fuente Tahoma

Tal vez a simple vista parezca el mismo código ascii de cualquier fuente, pero la diferencia estriba en que para todas las letras, incluyendo las letras con acento, se puede restar 32 números en decimal al código ascii de la letra para obtener su semejante pero en mayúscula; es así pues que se puede acceder al método *KeyPress* que entrega como parámetro el número en ascii de la tecla que se había presionado, posteriormente se verificaba si era una letra mayúscula, es decir, si estaba en el rango ascii de *97* a *122* o si era *á* o *é* o *í* u *ó* o *ú* o *ñ* y se le restaban

32 a dicho valor para convertirla en mayúscula. Se puede comprobar el código para la fuente Tahoma desde el mapa de caracteres de Windows como se muestra en la ***figura 2.20***.

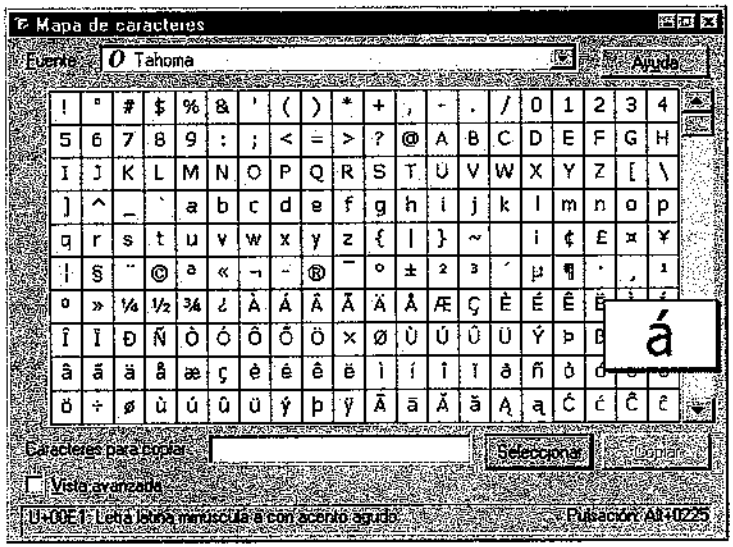


Figura 2.20 → Mapa de Caracteres de Windows para la fuente Tahoma

Pero no es únicamente el hecho de escribir minúsculas o no lo que se debía de validar, también era necesario verificar que no se escribieran espacios en blanco a los costados de las cadenas que se ingresaban en los diferentes campos; también se debe de verificar que no se ingresen caracteres inválidos como signos de interrogación, admiración, de pesos, de porcentajes, llaves, signos de operaciones, etc.

También es común que el usuario desee que exista la capacidad de pasar al siguiente campo con un ENTER en vez de la tabulación normal o un click del mouse. Para este propósito se crearon para cada una de las cajas de texto del sistema las funciones necesarias para validar cada una de las entradas que realizaba el usuario durante el registro o edición de los Miembros y de los Congresos en el programa.

Para comprender mejor el proceso completo que se realizaba en la versión anterior del sistema de información se debe apreciar la ***figura 2.21*** que contiene el diagrama de flujo completo para un campo que contenga nombres o direcciones.

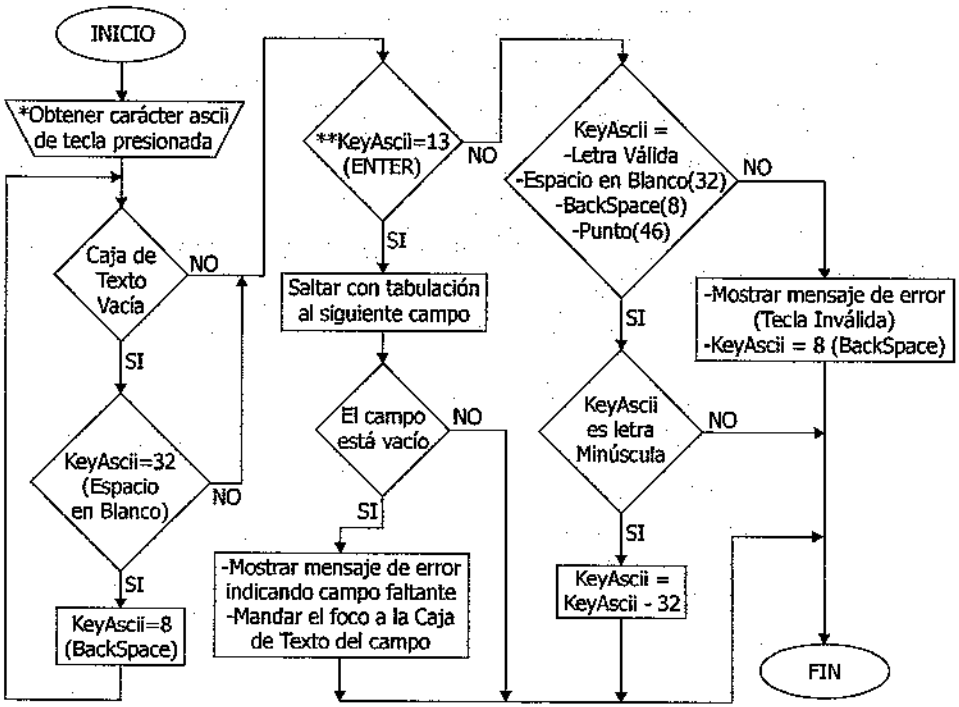


Figura 2.21 → Diagrama de Flujo para la Validación de Datos en Visual Basic 6

*Para obtener el valor en código Ascii de la tecla presionada accedemos al evento *KeyPress* de la Caja de Texto; dicho evento se produce cuando se presiona una tecla mientras el control tiene el foco.

**KeyAscii representa al número del carácter Ascii correspondiente a la tecla presionada.

El programar de esta forma la validación de datos representaba varias dificultades; la primera era que se debía de programar un módulo para cada uno de los campos del formulario de entrada; esto causa una saturación considerable dentro del código fuente, y representa también más desgaste del sistema.

Por otra parte no se contemplaba el hecho de que el usuario copiará desde otra fuente el dato a insertar y que lo pegará en el campo del registro. Definitivamente se podría programar un módulo especial que lo contemplara, pero de la misma forma se tendría que programar un módulo para cada uno de los campos.

Dentro de la plataforma .Net existe una nueva forma de validar los datos de una manera sencilla y muy accesible para cualquier desarrollador de cualquier nivel, y este método es representado a través de un componente llamado. ***ErrorProvider.***

El componente ***ErrorProvider*** de formularios *Windows Forms* permite indicar al usuario, de forma no intrusiva, que algo va mal; habitualmente, se utiliza junto con la validación de entrada del usuario en un formulario o con la presentación de errores dentro de un conjunto de datos. Un proveedor de errores es una alternativa mejor que mostrar un mensaje de error en un cuadro de mensaje, porque una vez que se descarta un cuadro de mensaje, el mensaje de error deja de estar visible.

El componente ***ErrorProvider*** muestra un icono de error () junto al control correspondiente, por ejemplo, un cuadro de texto; cuando el usuario coloca el puntero del ratón sobre el icono de error aparece una información sobre herramientas que muestra la cadena del mensaje de error. Dicho icono no desaparece hasta que es corregido el error, y al mismo tiempo deshabilita cualquier función (como botones u otros campos de llenado) hasta que el error sea corregido. La deshabilitación del formulario se realiza de manera automática y no se debe de programar una sola línea para dicho propósito, exceptuando el caso de que se permita cancelar la operación a través de un botón de cancelar o cerrando la ventana del formulario.

Con frecuencia, se utilizan expresiones regulares para validar datos especificados por el usuario; las expresiones regulares son una forma textual de expresar lo que debe o no debe de contener un campo, es decir, lo que deseamos o no que el usuario escriba dentro de una Caja de Texto; por ejemplo, podemos especificar que deseamos que se escriban únicamente las letras del alfabeto utilizando la siguiente expresión regular descrita simplemente como una cadena:

```
[a-zA-Z]|Ñ|ñ|[á-úÁ-Ú]
```

También existe una nueva propiedad para algunos controles de *Windows Forms* (incluyendo las *Cajas de Texto*) llamada ***CharacterCasing*** que modifica la condición de mayúscula o minúscula de los caracteres a medida que se escriben dentro del control. Dicha propiedad puede tomar tres valores, dichos valores están descritos en la ***tabla 2.13.***

Valor	Descripción
Lower	Convierte todos los caracteres en minúsculas.
Normal	La mayúscula o minúscula de los caracteres permanece sin cambios.
Upper	Convierte todos los caracteres en mayúsculas.

Tabla 2.13 → Valores para la propiedad CharacterCasing de un control TextBox (Caja de Texto)

Con la propiedad ***CharacterCasing*** se puede controlar que tipo de caracteres se desea que ingresen en una Caja de Texto en específico, letras mayúsculas, minúsculas o una mezcla de las dos. ***CharacterCasing*** actúa de forma automática y no se debe programar una sola línea para su funcionamiento, afectará también a los textos que son copiados desde otro origen y pegados en la campo de texto.

Una vez que se haya indicado a través de expresiones regulares lo que debe o no debe contener algún campo, el icono de error del *ErrorProvider* aparecerá al costado del control que se encuentre en "estado de error", bloqueará todos los controles excepto los que le se haya dicho que ignore (como algún procedimiento o botón para cancelar la operación) sin dejarnos hacer nada hasta que se corrija dicho error, el cual, será indicado a través de una etiqueta que aparecerá al pasar el cursor del mouse sobre el icono, como se muestra en la ***figura 2.22.***

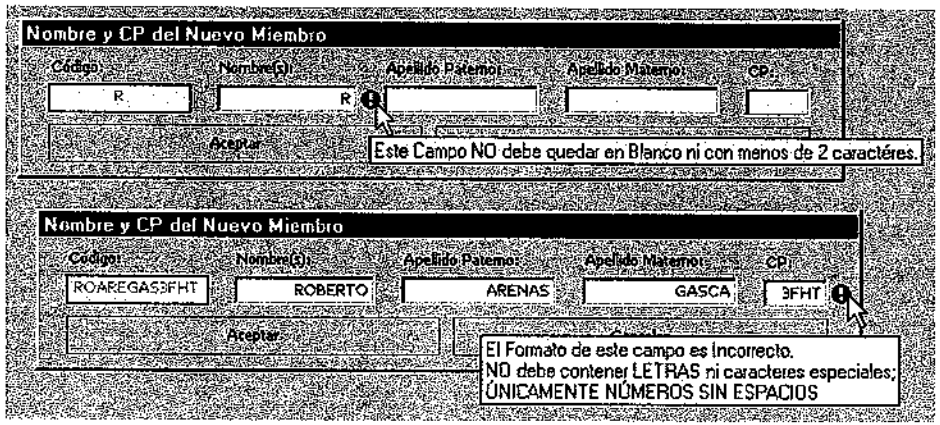


Figura 2.22 → Ilustración de muestra del funcionamiento del ErrorProvider

El icono de error del *ErrorProvider* desaparecerá automáticamente cuando se corrija el error o aparecerá de nuevo con otra etiqueta en el caso de que se incurra en un error diferente sobre el mismo campo, y así mismo, desaparecerá hasta que se corrija el segundo error y así sucesivamente.

Otra de las novedades dentro de la plataforma Visual Basic .Net es que existe la posibilidad de controlar varios eventos dentro de los programas con un solo procedimiento. Esto quiere decir que si se tienen varios controles que funcionan de manera exacta y que tienen las mismas respuestas y parámetros dentro del código que controla algún evento en específico, se puede programar un solo procedimiento e indicarle al mismo que él será el único que controlará dicho evento para todos los controles a los que se les haya indicado. De esta forma, no se deberá programar una y otra vez la misma respuesta para algún evento, como validar los datos; en vez de ello, se debe simplemente programar un solo procedimiento para todos los controles que se deseen y así no se saturarán de líneas duplicadas los códigos para los procedimientos.

Esta forma de validación de datos se ha implementado como un accesorio nuevo e innovador para la plataforma .Net Framework que facilita la admisión de datos, su programación e implementación. Cuenta con una interfaz sencilla, accesible y sumamente explicativa para el usuario, informándole un error y también previniendo depuraciones posteriores de la base de datos.

Para poder ver más a detalle el funcionamiento del control *ErrorProvider* se ha preparado un ejemplo concreto en el ***apartado 2.12.5*** llamado *Validación de Datos con el Control ErrorProvider* con el que se puede agregar dicho componente a las aplicaciones, aprovechar al máximo sus funciones y poder validar prácticamente cualquier entrada y posible causante de error por parte del usuario; previniendo así fallas posteriores.

2.10.5 Registro de Asistencias

Los registros de asistencia se guardan dentro de la base de datos de forma numérica, los registros son convertidos de forma decimal a binaria para ser posteriormente traducidos a casillas de verificación para que el usuario pueda analizarlos y editarlos. Dentro del *TabPage* de *Miembros* existen dos controles especialmente dedicados para dicho propósito, estos controles están listados en la ***tabla 2.14.***

Nombre	Tipo	Descripción
TxtNConAct	TextBox	Contenedor del registro **NumCongAct** de la tabla *Miembros.*
Ch	CheckedListBox	Contenedor en el que se representará a través de casillas de verificación el registro **AsisAct** de la tabla *Miembros.*

Tabla 2.14 → Descripción de controles para el registro de asistencia

La zona para el registro de asistencia contiene dos registros, el del *número de congreso actual* y el de la *asistencia actual;* el registro de la asistencia actual se realiza a través de once casillas de verificación y son únicamente para el registro de asistencia a los talleres del congreso al que se encuentra inscrito actualmente; posteriormente, al inscribirse a un nuevo evento, este dato junto con el registro *NumCongAct* pasarán a ser parte del historial y tomarán el siguiente espacio vacío dentro del área de los historiales, campo que será señalado por el registro *NumHis*, el cual cambiará automáticamente aumentando e indicando el siguiente espacio vacío para el próximo historial editado. Para poder ver el proceso del registro de los historiales se puede seguir el diagrama de flujo en la ***figura 2.23***, que muestra el procedimiento que se sigue para registrar las asistencias.

El registro de asistencias guarda la concurrencia a los talleres que se imparten en un congreso, pero cada congreso puede tener un máximo de diez talleres dentro del sistema; sin embargo, son guardados once registros porque también se guarda la asistencia al congreso aunque el miembro no asista a ningún taller, ya que existen miembros que se inscriben al evento y no asisten.

Dentro de las consultas debe de existir la posibilidad de examinar a los miembros que hayan asistido o no a cierto evento, y para ello simplemente habría que buscar los miembros que tengan como registro de asistencia al evento un número cuyo primer ***bit*** sea un dato positivo (o uno lógico), es decir, un número que en forma binaria sea mayor que $\#01111111111_b$, dicho número en decimal es $\#1023_d$, por lo tanto, si se desea consultar a los miembros que hayan asistido efectivamente al congreso, hayan o no acudido a algún taller, simplemente se buscarán los registros que tengan en *AsisAct* un número decimal igual o mayor a $\#1024_d$.

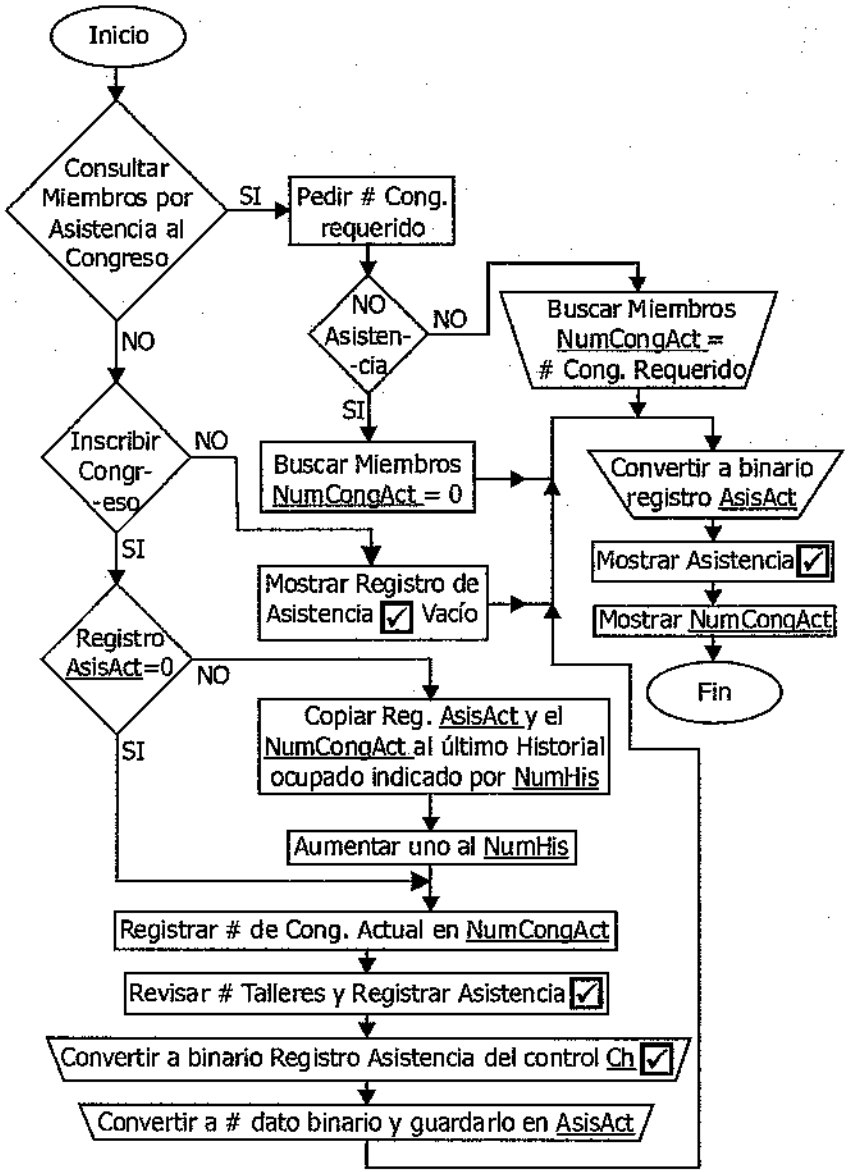


Figura 2.23 → Diagrama de flujo del registro de asistencias

En resumen, el registro de asistencias es un sistema de almacenamiento eficaz que traduce un registro numérico a una matriz de casillas de verificación con las cuales el usuario puede tener control total sobre la edición y la visualización de dicha información. Como se ha mencionado, los registros más importantes dentro del conjunto de los historiales son *NumCongAct* y *AsisAct*, los cuales, representan tanto la asistencia al evento actual, así como la información del último evento al que asistió el Miembro; por ejemplo, un registro que contenga en *NumCongAct* el valor **5** y en *AsisAct* el número $\#1344_d$ ($\#1010100000_b$) se visualizarán en los controles de asistencia de la ventana del sistema de información como se muestra en la figura 2.24.

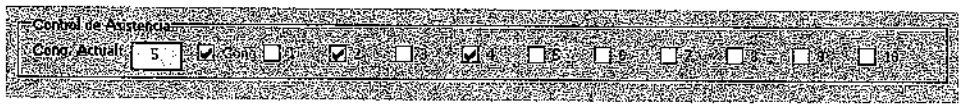


Figura 2.24 → Controles para el Registro de Asistencia de los Miembros

Las casillas de verificación son una forma digerible de ver un dato que de otra manera sería imposible de estudiar. La conversión de binario a decimal es un método sencillo que percibe cuando las casillas de verificación están o no activadas, es decir, palomeadas. Como el control *CheckedListBox* es un arreglo de dichas casillas, es posible acceder a los diferentes elementos de la lista de casillas y obtener su estado actual. De esta forma se puede transformar un dato que encierra cierta complejidad en un número que es fácil de almacenar y que al mismo tiempo no genera espacio innecesario dentro de la base de datos.

Los registros de los historiales son también escalables a cualquier modificación posterior al sistema, ya que siguen siendo datos que almacenan de manera ineludible y casi infalible los registros de los historiales para que la AMD tenga un registro conciso de la concurrencia que existe a sus eventos, y al mismo tiempo, cada uno de los miembros tendrá la certeza de que su asistencia se podrá almacenar de la mejor manera para que posteriormente dicho recuento sea recompensado en forma de puntos para su ingreso a esta institución.

Posteriormente se podrá estudiar como es que los registros de las asistencias son volcados a un reporte especial, el cual podrá ser impreso y que contendrá toda la información de sus asistencias a lo largo de los eventos registrados dentro del sistema de información.

2.10.6 Consultas

Una base de datos consta de una colección de tablas con datos y otros objetos como vistas, índices, procedimientos almacenados y desencadenadores, que se definen para poder llevar a cabo distintas operaciones con datos. Los datos almacenados en una base de datos suelen estar relacionados con un tema o un proceso determinados como, por ejemplo, la información de inventario para el almacén de una fábrica.

Las tablas son objetos de la base de datos que contienen todos sus datos. Una tabla se define mediante una colección de columnas. En las tablas, los datos se organizan con arreglo a un formato de filas y columnas, similar al de una hoja de cálculo. Cada fila representa a un registro único, y cada columna representa a un campo dentro de un registro.

Una consulta es simplemente una petición específica de recuperación, modificación o eliminación de los datos contenidos en una tabla. Una consulta típica dentro de un manejador de base de datos de realiza a través del *Lenguaje Estructurado de Consulta* o *SQL (Structured Query Language)*. Cuando únicamente se desea obtener un subconjunto del universo de los registros contenidos en una tabla usualmente se debe hacer referencia al procedimiento almacenado ***SELECT*** del lenguaje SQL. Este método obtiene filas de la base de datos y permite realizar la selección de una o varias filas o columnas de una o varias tablas. La sintaxis completa de la instrucción *SELECT* es compleja, aunque las cláusulas principales se pueden resumir como sigue:

SELECT listaSelección
[INTO nuevaTabla_]
FROM origenTabla
[WHERE condiciónBúsqueda]
[GROUP BY expresiónAgruparPor]
[HAVING condiciónBúsqueda]
[ORDER BY expresiónOrden [*ASC* | *DESC*]]

Ésta instrucción es una forma estándar en la que la mayoría de los manejadores, lenguajes y programas acceden a los diferentes registros almacenados en una base de datos. Incluso dentro de la aplicación ***AcadMexDerm*** se realiza la primera petición al Adaptador de Datos para que llene el *DataSet* a través de una instrucción simple de SQL.

En la versión anterior de la aplicación, para cualquier petición a una consulta diferente, esta se debía realizar a través de una sentencia *SELECT* para poder filtrar los registros u ordenarlos en cierta disposición. En cambio ahora, este filtrado se realiza con ayuda del *DataView*, el cual representa una vista personalizada que puede enlazar datos de un *DataTable* para ordenación, filtrado, búsqueda, edición y exploración de registros.

Un *DataView* permite crear diferentes vistas de los datos almacenados en una *DataTable*, una capacidad que suele utilizarse prácticamente todas las aplicaciones de enlace a datos. Mediante un *DataView* se pueden exponer los datos de una tabla con distintos criterios de ordenación y se pueden filtrar los datos por el estado de una fila o basándose en una expresión de filtro.

Un *DataView* proporciona una vista dinámica de los datos cuyo contenido, ordenación y pertenencia reflejan cambios en el *DataTable* subyacente a medida que se producen. Esto es distinto del método *SELECT* ordinario de los sistemas de bases de datos, que devuelve una matriz registros de una tabla por un criterio de ordenación determinado y cuyo contenido no refleja cambios en la tabla subyacente, pero cuya pertenencia y ordenación siguen siendo estáticas. Las capacidades dinámicas de la *DataView* hacen que resulte ideal para las aplicaciones de enlace a datos.

Un *DataView* proporciona una vista dinámica de un único conjunto de datos a la que se puede aplicar distintos criterios de ordenación y filtrado, de manera similar a la vista suministrada por una base de datos. Sin embargo, un *DataView* difiere considerablemente de una vista de base de datos en el sentido de que el *DataView* no se puede tratar como una tabla y no puede proporcionar una vista de tablas combinadas. Tampoco puede excluir columnas que existen en la tabla de origen ni puede anexar columnas, como columnas de cálculo, que no existen en la tabla de origen. El *DataView* ofrece varias posibilidades para ordenar y filtrar datos de una *DataTable*:

- Con la propiedad *Sort* se pueden especificar criterios simples o múltiples de ordenación de columnas e incluir parámetros *ASC* (*ascendente*) y *DESC* (*descendente*).

- Puede utilizar la propiedad *ApplyDefaultSort* para crear automáticamente un criterio de ordenación, en sentido ascendente, basándose en la columna o las columnas de clave principal de la tabla. *ApplyDefaultSort* sólo se aplica cuando la propiedad *Sort* es una referencia nula o una cadena vacía y cuando la tabla tiene definida una clave principal.

- Con la propiedad *RowFilter* puede especificar subconjuntos de filas basándose en sus valores de columna a través de una expresión utilizada para filtrar filas, calcular los valores de una columna o crear una columna agregada.

- Si desea devolver los resultados de una consulta determinada en los datos, en lugar de proporcionar una vista dinámica de un subconjunto de los datos, para conseguir el máximo rendimiento se utilizan los métodos *Find* o *FindRows* de la *DataView* en lugar de establecer la propiedad *RowFilter*. El establecimiento de la propiedad *RowFilter* hace que se vuelva a generar el índice de los datos, lo que agrega sobrecarga a la aplicación y reduce el rendimiento. La propiedad *RowFilter* es más idónea en una aplicación enlazada a datos donde un control enlazado muestra resultados filtrados. Los métodos *Find* y *FindRows* aprovechan el índice actual, sin necesidad de volver a generarlo. Con los métodos *Find* y *FindRows* del *DataView* se pueden buscar filas según sus valores de clave de ordenación. La diferenciación entre mayúsculas y minúsculas de los valores de búsqueda en los métodos *Find* y *FindRows* está determinada por la propiedad *CaseSensitive* del *DataTable* subyacente. Los valores de búsqueda deben coincidir en su totalidad con los valores de clave de ordenación existentes para que se devuelva un resultado. El método *Find* devuelve un entero con el índice de la *DataRowView* que coincide con los criterios de búsqueda. Si más de una fila coincide con los criterios de búsqueda, sólo se devolverá el índice de la primera *DataRowView* coincidente. Si no se encuentra ninguna coincidencia, *Find* devuelve -1. Para devolver resultados de la búsqueda que coincidan con varias filas, se puede utilizar el método *FindRows*. *FindRows* funciona igual que el método *Find*, excepto en que devuelve una matriz de *DataRowView* que hace referencia a todas las filas coincidentes de la *DataView*. Si no se encuentra ninguna coincidencia, la matriz de *DataRowView* estará vacía.

- Con la propiedad *RowStateFilter* se puede especificar que versiones de fila desea ver. EL *DataView* administra implícitamente que versión de fila exponer, dependiendo del *RowState* de la fila subyacente. Por ejemplo, si el *RowStateFilter* está establecido como *DataViewRowState.Deleted*, el *DataView* expondrá la versión de fila *Original* de todas las filas *Deleted* porque no hay ninguna versión de fila *Current*. Con la propiedad *RowVersion* del *DataRowView* se puede determinar que versión de una fila se está exponiendo.

En la ***tabla 2.15*** se muestran los valores que puede tomar el *DataViewRowState*.

Nombre de miembro	Descripción	Valor
Added	Fila nueva.	4
CurrentRows	Filas actuales, incluidas las filas sin modificar, las nuevas y las modificadas.	22
Deleted	Fila eliminada.	8
ModifiedCurrent	Versión actual, que es una versión modificada de los datos originales (vea ModifiedOriginal).	16
ModifiedOriginal	Versión original (aunque se haya modificado y esté disponible como ModifiedCurrent).	32
None	Ninguna.	0
OriginalRows	Filas originales, incluidas las filas sin modificar y las eliminadas.	42
Unchanged	Fila sin modificar.	2

Tabla 2.15 → valores para la propiedad DataViewRowState

La propiedad *RowFilter* obtiene o establece la expresión utilizada para filtrar las filas que se ven en el *DataView*. Para formar un valor *RowFilter*, se especifica el nombre de una columna seguido de un operador y un valor que filtrar; por ejemplo:

Estado = "DISTRITO FEDERAL"

El sistema de información utiliza el método *RowFilter* para poder filtrar las filas de la tabla Miembros que se desean mostrar. Par ello se ha creado especialmente una sección dedicada únicamente para que el usuario genere diferentes consultas a partir de cinco principales parámetros:

1. Estado
2. Condición
3. Miembro
4. Especialidad
5. Número de Congreso

Estos parámetros estan divididos en sub-parámetros como se muestra en la ***figura 2.25***, en la cual se puede apreciar la gama de selecciones con que cuenta el usuario para poder filtrar los registros que se mostrarán dentro del sistema.

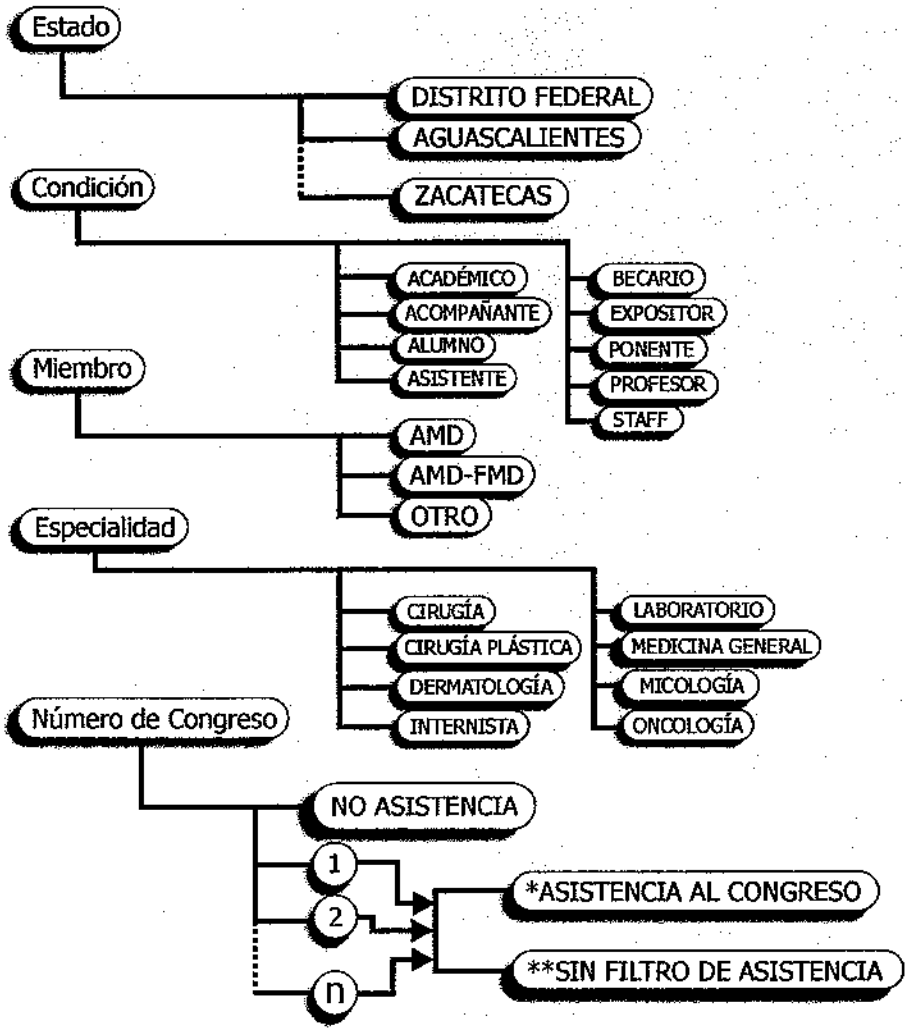


Figura 2.25 → Parámetros de selección para generar una consulta

*Como se ha mencionado antes, si se deseara buscar una persona que haya asistido presencialmente al evento, se deben filtrar los registros que en el campo *AsisAct* tengan un valor numérico mayor o igual a $\#1024_d$.

**Si el usuario no desea tener un filtro de asistencia, es decir, es indiferente el hecho de que haya asistido o no al evento en cuestión, el filtro de búsqueda aplicará a cualquier valor contenido en *AsisAct*.

La aplicación utiliza tres controles ***ComboBox*** de *Windows Form* para que el usuario genere una consulta. Un *ComboBox* es una lista desplegable muy común en ambiente Windows con el cual se permite desplegar una lista de opciones al usuario para que seleccione una y se realiza cierta acción, en este caso, seleccionar un parámetro de consulta. Los controles utilizados para las consultas están listados en la ***tabla 2.16.***

Nombre	Tipo	Descripción
ComboParametro1	ComboBox	Lista Desplegable donde están contenidos los primeros parámetros de búsqueda como: • Estado • Condición • Miembro • Especialidad • Número de Congreso
ComboParametro2	ComboBox	Lista desplegable donde están contenidos los primeros subparámetros de búsqueda. El contenido de esta lista desplegable depende del parámetro que haya seleccionado en el primer parámetro, es decir, si el usuario hubiera seleccionaco como primer parámetro el número de congreso, las opciones dentro de este *ComboBox* serían los números de congresos agregados en el apartado de Congresos además de la opción "NO ASISTENCIA", la cual, nos ofrece la alternativa de filtrar a los miembros que no hayan asistido a ningún evento de la AMD.
ComboParametro3	ComboBox	Lista desplegable donde están contenidos los últimos subparámetros de búsqueda. Si el usuario ha seleccionado como subparámetro cualquier número de congreso, ésta lista desplegable contendrá las opciones: • ASISTENCIA AL CONGRESO • SIN FILTRO DE ASISTENCIA
BConsulta	Button	Botón para generar la Consulta.
BRestaurarConsulta	Button	Botón para Cancelar la Consulta y Restaurar los valores originales de los parámetros originales en los ComboBox.

Tabla 2.16 → Descripción de los controles utilizados para generar la Consulta

Como se aprecia en la ***figura 2.25*** existe un máximo de 3 subparámetros a escoger, los *ComboBox* van tomando los valores de la selección de los parámetros anteriores; este método es ampliamente utilizado, por ejemplo, en los formularios de Internet donde se indica que se seleccione el país de procedencia en una lista desplegable y dinámicamente una segunda lista toma los valores de los estados del país que se haya seleccionado en primera instancia. La aplicación utiliza este

método para poder ramificar las opciones de filtrado y así generar la consulta que afectará al sistema. En la ***figura 2.26*** se puede apreciar el aspecto de las listas de selección contenidas dentro del *TabPage* de consultas.

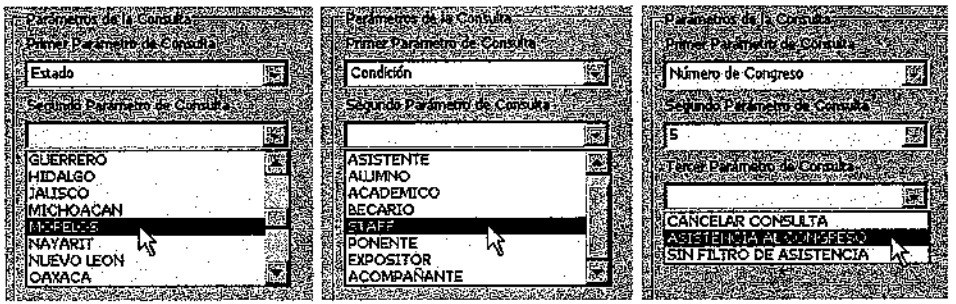


Figura 2.26 → ComboBox para diferentes parámetros de Consulta

Cuando el usuario haya generado la consulta, el *DataView* afectará al *TabPage* de *Miembros* por lo que únicamente aparecerán los registros con las características que se hayan seleccionado como filtro de consulta. Al mismo tiempo, en la *Rejilla de Visualización*, como se verá más adelante, sólo aparecerán los registros filtrados por la consulta.

El sistema de correos electrónicos en el *TabPage* de *E-Mail*, que será descrito más adelante, será afectado por la consulta, es decir, si se deseara enviar correos electrónicos únicamente a los miembros del Distrito Federal, lo único que deberá hacer el usuario es generar una consulta con dicho parámetro y el sistema automáticamente enlistará a los *Miembros* del D. F. que tengan una dirección de correo electrónico válida dentro del *TabPage* de *E-Mail*.

Así pues, el sistema complementa la opción de generar consultas para facilitar la búsqueda de miembros con ciertas características, las más importantes; y son afectados los diferentes apartados automáticamente siendo este método de consulta fácil de usar y sin cavidad para confusiones.

2.10.7 E - Mail

Para la mesa directiva de la AMD una de las funciones principales que debía de contener el sistema de información es la capacidad de enviar correos electrónicos de forma masiva a todos los miembros inscritos al sistema. Obviamente el requerimiento principal sería tomar únicamente los registros que contaran con una dirección de correo válida; en segundo lugar se tenía que tomar en cuenta que al realizar una consulta, el sistema de correos sería afectado por dicha consulta y de esta manera se podría también filtrar a los miembros a los que se deseara enviar un correo.

Al enviar correos electrónicos no se debía sacrificar ninguna de las funciones que este conlleva, como son contener asunto, un cuerpo de mensaje y uno o más archivos adjuntos. Desde la versión anterior de la Aplicación AcadMexDerm existe la posibilidad de enviar correos electrónicos de manera masiva, sin embargo, la lista de destinatarios comenzó a aumentar a niveles de cientos de contactos con correo electrónico, y dada la cantidad tan grande de correos que se debían enviar en un solo e-mail el servicio de ***Protocolo Simple de Transferencia de Correo (SMTP, Simple Mail Transfer Protocol)*** del servidor *Prodigy* (que es el proveedor de dicho servicio para la AMD) comenzó a regresar los correos de salida dada la lista tan grande de destinatarios.

Fue entonces que se decidió que para la nueva versión se dividirían los correos en una lista de receptores dividida de 50 en 50 destinatarios para no provocar una sobrecarga en el servidor de salida de los correos electrónicos. Para dicho fin se han agregado los controles listados en la ***tabla 2.17*** que se encuentran dispuestas en el *TabPage* llamado *e-mail* especialmente diseñado para enviar los correos electrónicos.

Nombre	Tipo	Descripción
Lista_Mail	ListBox	Lista contenedora de los destinatarios del correo electrónico.
TxtAsuntoMail	TextBox	Caja de texto contenedora del campo Asunto del correo electrónico.
TxtMensajeMail	TextBox	Caja de Texto contenedora del cuerpo del mensaje del correo electrónico.
Lista_Adjuntos	ListBox	Lista contenedora de los archivos que se adjuntarán al correo electrónico.
btAdicionarArchivo	Button	Botón para agregar la ruta de un archivo a la lista de archivos adjuntos.

btQuitarArchivo	Button	Botón para quitar de la lista la ruta del archivo que el usuario haya seleccionado de la lista de archivos adjuntos.
btEnviarMail	Button	Botón para enviar el correo electrónico actual.
btLimpiarCampos	Button	Botón para borrar el contenido de los contenedores de los campos Asunto, Mensaje y Archivos Adjuntos del correo electrónico actual.

Tabla 2.17 → Descripción de los controles para el *TabPage* de e-mail.

En la ***figura 2.26*** se puede apreciar el aspecto de los controles antes mencionados dentro del *TabPage* de e-mail.

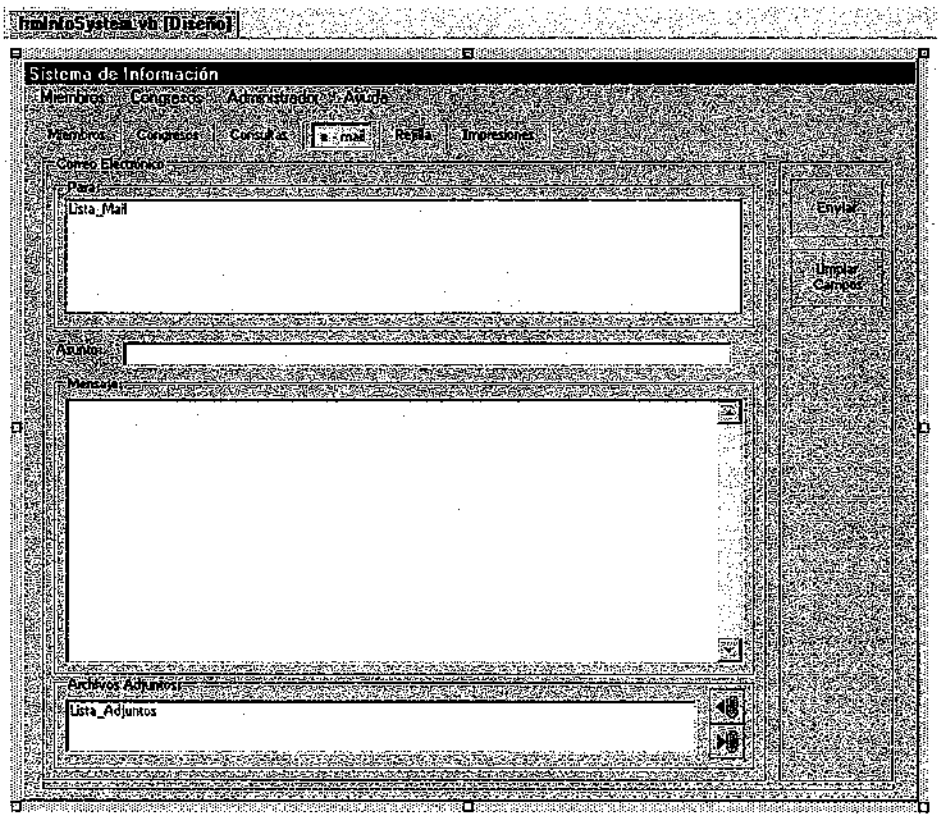


Figura 2.26 → Aspecto de los controles del *TabPage* e-mail

En la ***figura 2.27*** se encuentra el diagrama de flujo del proceso completo que se sigue para enviar correos electrónicos de manera masiva o individualmente para la versión actual del sistema de información.

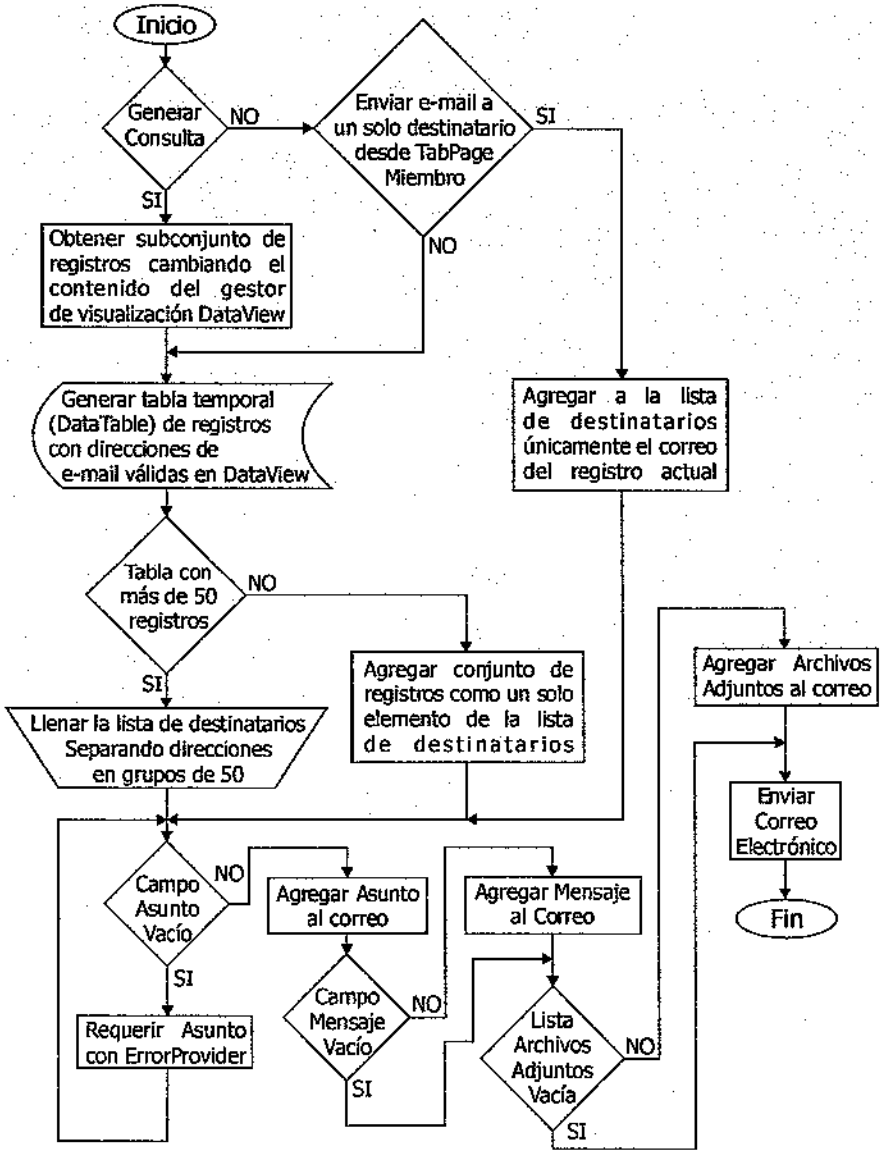


Figura 2.27 → Diagrama de flujo del proceso de envío de correos electrónicos

El problema de la saturación de direcciones en un solo correo electrónico fue resuelto sin mayor problema al dividir los destinatarios en grupos de cincuenta correos, sin embargo, todos los correos que son enviados, sin importar en número, contendrán la misma información, como el asunto, mensaje y los archivos adjuntos.

En el ***apartado 2.12.6 (Correos Electrónicos a través de Outlook Object Library)*** es posible ver como se pueden envían correos electrónicos de la misma forma en que lo hace la aplicación ***AcadMexDerm .Net.***

2.10.8 Rejillas de Visualización y Clasificación

Una forma común de mostrar una sucesión de datos o registros relacionados entre sí y divididos por columnas es a través de celdas en forma de una cuadrícula desplazable tipo hoja de cálculo. Existen muchos programas que realizan operaciones con registros en forma de hoja de cálculo, el más común en ambientes *Windows* es *Microsoft Excel*. La mayoría de los manejadores de bases de datos utilizan un método similar para mostrar los registros; esta vista se muestra en una ventana que expone las filas de las tablas originales y también las consultas generadas dentro del manejador en forma de tablas de datos. En esta vista es posible editar los campos, ordenarlos alfabéticamente ascendente y descendentemente, buscar, agregar y eliminar los datos; de tal forma que esta vista de datos en forma de cuadrícula proporciona las herramientas necesarias para trabajar con los datos de una manera sencilla e interactiva. El manejador *Microsoft Access* utiliza una vista en forma de tabla dinámica para las hojas de datos como se muestra en la ***figura 2.28***.

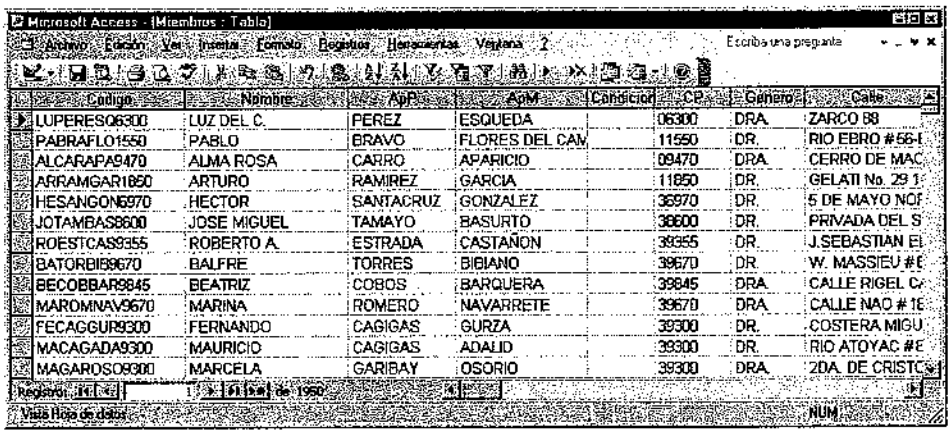


Figura 2.28 → Vista de los datos en Microsoft Access

Los usuarios avanzados relacionan casi siempre los registros de las bases de datos en este tipo de vistas; para ellos, es una forma común de ver los datos, relacionarlos, ordenarlos y ubicar un cierto registro desplazándose a través de la tabla y posicionando el cursor dentro de dicho registro. Para los usuarios del sistema de información ***AcadMexDerm*** este tipo de vista resulta muy útil para poder ubicar a un miembro relacionando su registro por cierto parámetro, es decir, a veces resulta más fácil ubicar a un miembro por su dirección, por su condición o tal vez hasta por su *RFC* o *CURP*.

Hablando del lado de la programación, resultaría muy engorroso crear un procedimiento especial para la búsqueda de un Miembro para todos y cada uno de los parámetros contenidos dentro de la tabla, u otra forma de decirlo, sería complicado realizar un método de búsqueda para todas las columnas de la tabla.

Tal vez la tabla de miembros de la base de datos *AcadMexDerm* sea pequeña en comparación a otras bases de datos, pero el fondo del argumento sigue siendo el mismo, si las tablas poseen una serie de *"n"* columnas, sería prácticamente imposible saber para cuál de estas columnas resultaría útil generar dicho procedimiento de búsqueda, ya que esto depende de en que situación se encuentre el usuario y que dato posea de un cierto miembro para poder ubicarlo.

Para no tener que programar muchos tipos de procedimientos de búsqueda es posible mostrarle al usuario los datos de las tablas en forma de una rejilla de visualización, tal y como lo hace *Microsoft Access* para sus bases de datos. Para ello la plataforma *.Net Framework* proporciona un control especial para mostrar los datos contenidos en un *DataSet* llamado ***DataGrid***.

El control *DataGrid* muestra datos de *ADO.Net* en una rejilla de visualización que se puede vincular a cualquiera de los siguientes orígenes de datos:

- ✓ DataTable
- ✓ DataView
- ✓ DataSet
- ✓ DataViewManager

Dentro del sistema de información existen dos Rejillas de visualización, la primera aparece como una sección siempre visible del contenedor MDI de la aplicación como un control acoplado a la ventana principal. Esta rejilla se puede examinar a través de un control especial de Windows llamado ***splitter***. El control *splitter* de formularios *Windows Forms* se utiliza para cambiar en tiempo de ejecución el tamaño de los controles acoplados, en este caso, el *DataGrid*.

El control *splitter* suele utilizarse en formularios con controles que contienen datos de longitud variable, como el *Explorador de Windows*, cuyos paneles de datos contienen información cuyo ancho cambia en función de las acciones del usuario.

Esta primera rejilla de visualización no tiene ninguna inferencia en los procesos internos del sistema de información, su propósito de existir es simplemente tener una forma de ubicar a los miembros, y al estar en la ventana principal del contenedor MDI, permite tener una rejilla más grande que la segunda, la cual, está contenida dentro de la ventana del sistema de información. Para poder entender mejor lo anterior es posible ver la ***figura 2.29***, la cual muestra como está distribuida la primera rejilla de visualización y su proporción en comparación con la segunda.

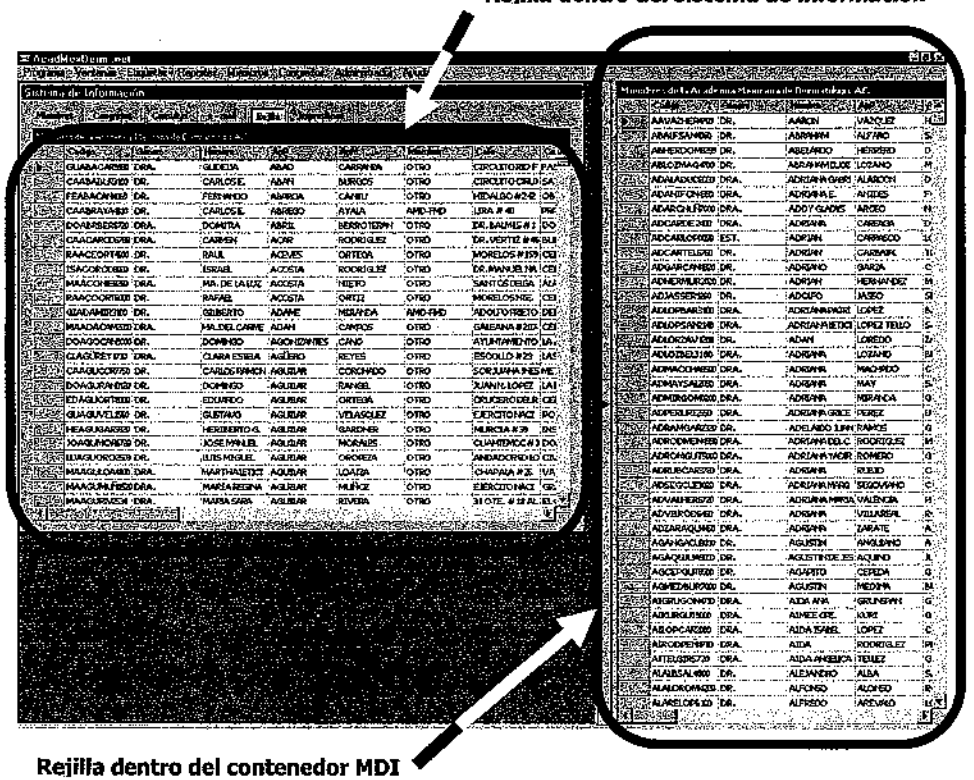


Figura 2.29 → Rejillas de Visualización dentro del sistema de información

Cuando el contenedor MDI se maximiza o cambia de tamaño, la rejilla de visualización cambiará automáticamente en proporción a su contenedora. Cuando el usuario sitúa el puntero del *mouse* sobre el borde del *splitter* el puntero cambia

de apariencia para indicar que es posible cambiar su tamaño. El control splitter permite al usuario cambiar el tamaño de la rejilla que se encuentra acoplada; pero también se ha agregado un botón especial para poder ampliarla a su tamaño máximo dentro del contenedor *MDI*, que es en proporción al 80% del tamaño horizontal de la ventana, de tal forma que no cubra por completo las ventanas contenidas dentro de la ventana principal; al mismo tiempo, al hallarse en su tamaño máximo, es posible regresarla con el mismo botón a su estado original (un 10% del tamaño del contenedor *MDI*), el cual, se encuentra al costado izquierdo del *splitter*, como se muestra en la ***figura 2.30***.

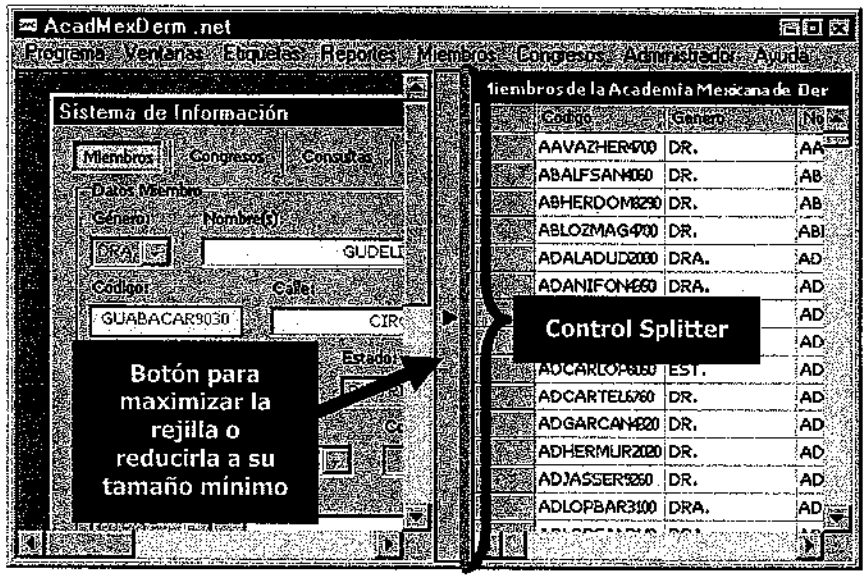


Figura 2.30 → Botón para cambiar el tamaño de la rejilla grande y el control *splitter*

La segunda rejilla se encuentra dentro de un *TabPage* del sistema de información especialmente creado para contenerla. Esta rejilla está diréctamente vinculada a muchos de los procesos que realiza la aplicación, y al mismo tiempo, esta posee un proceso que afecta al resto de los módulos del programa. Esta rejilla ha sido incluida desde la segunda versión de la aplicación *AcadMexDerm* y ha sido parte esencial de su contenido y forma de mostrar los datos y ubicarlos dentro del esquema completo de registros; sin embargo, en ninguna de las versiones anteriores ha poseído la versatilidad que goza bajo la plataforma .Net.

Para tener un panorama más amplio del funcionamiento de las dos rejillas y como afectan y son afectadas dentro de la aplicación, se puede analizar la ***figura 2.30***, la cual muestra como afectan estos procesos a los diferentes módulos del sistema, dependiendo de la acción que se realice dentro o fuera de las rejillas de visualización.

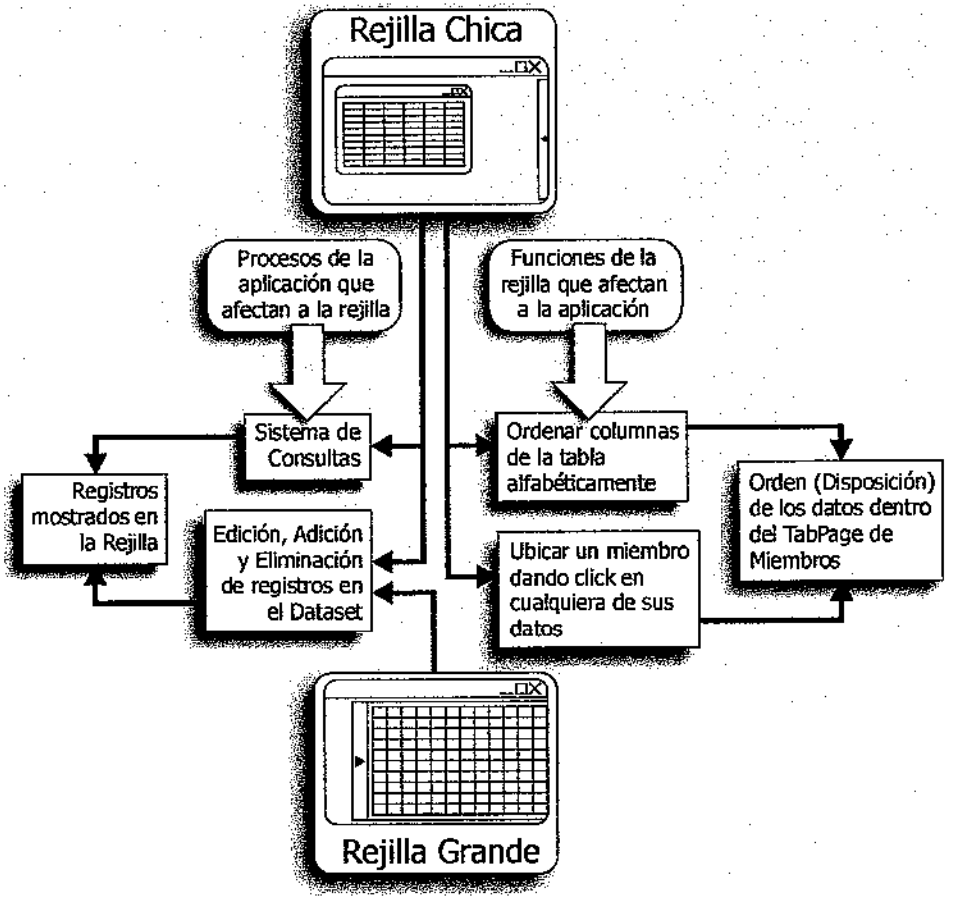


Figura 2.30 → Funcionamiento de las rejillas de visualización

La rejilla chica es parte importante dentro del sistema de información, es el único control en el cual se pueden ordenar a los miembros alfabéticamente por cualquiera de las columnas de la tabla, es decir, de manera predeterminada el sistema organiza los registros de los miembros por el apellido paterno,

sin embargo, es posible ordenarlos por cualquier campo, como nombre, dirección, especialidad, correo electrónico o cualquiera que requiera el usuario para poder ubicar algún miembro; como se muestra en la ***figura 2.31***.

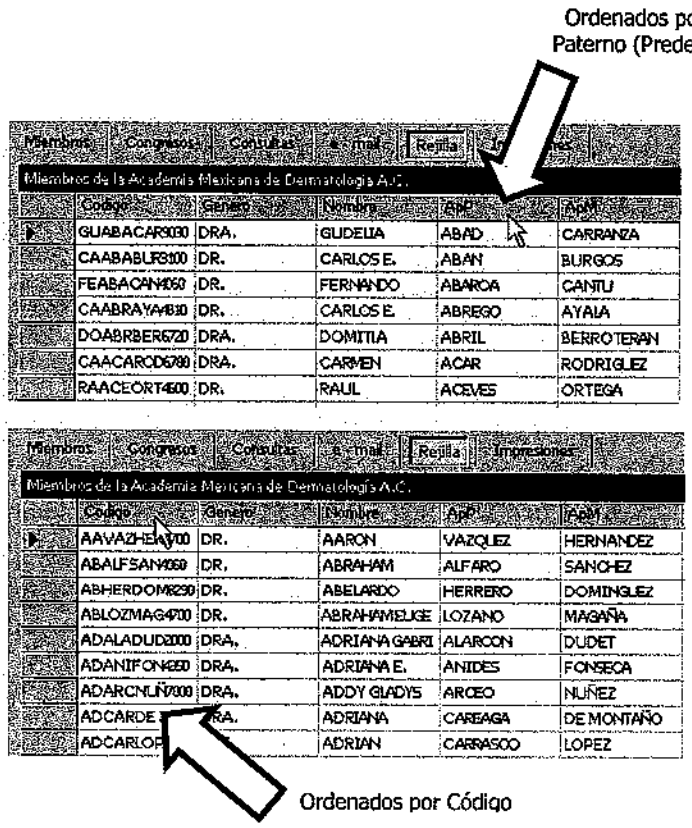


Figura 2.31 → Cambiar el orden en como se muestran los registros con la Rejilla de Visualización

En el ejemplo de la ***figura 2.31*** se cambió el orden de los registros de los *miembros* y se han dispuesto por su *código* en orden alfabético. Cuando se cambia el orden de los registros, la disposición de los registros en el *TabPage* de Miembros cambia y los muestra dependiendo de como estén organizados en la rejilla de visualización.

Cuando se realizan cambios en los registros, como editarlos, eliminarlos o agregar uno nuevo, estos cambios se ven reflejados de manera inmediata dentro de la rejilla de visualización; también el sistema de consultas afecta directamente los registros que se muestran en la rejilla, es decir, cuando el usuario realice una

consulta, sólo los registros que cumplan con los requisitos de ésta serán los que se vean dentro del *DataGrid*, como se muestra en la ***figura 2.32***, en la que se puede observar la rejilla de visualización después de realizar una consulta de los miembros cuyo estado de la República Mexicana sea *Baja California Sur*.

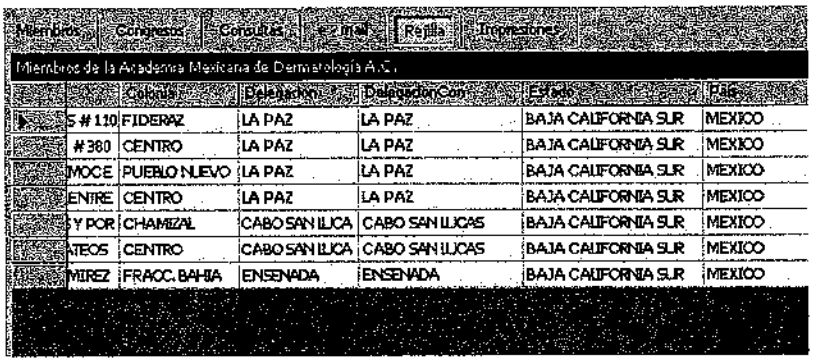


Figura 2.32 → El sistema de consultas afecta directamente la rejilla de visualización

Cuando el usuario da click en un cualquier miembro del *DataGrid*, se pasará automáticamente el *TabPage* de *miembros* al frente y estará ubicado en la posición de dicho miembro. Esto da toda la funcionalidad de poder hallar un miembro por cualquiera de sus campos y editarlo de manera normal simplemente dando un click del Mouse.

El potencial del control *DataGrid* es explotado dentro del sistema de información con el único propósito de facilitar la ubicación de los miembros dentro de la aplicación. Es posible ver como se puede adicionar el control *DataGrid* a un formulario *Windows Forms* en el ***apartado 2.12.8 (Rejilla de Visualización a través del control DataGrid)***, en el cuál, se explica de manera detallada el funcionamiento de este control y la forma de vincularlo a casi cualquier tipo de origen proveniente de *ADO .Net*.

2.10.9 Búsquedas

En una base de datos relacional, el ***índice*** de una tabla es el elemento que proporciona un acceso rápido a los datos de las filas de ésta. Estos proporcionan acceso rápido a los datos y pueden evitar la duplicidad de los registros de una tabla.

Los índices de una base de datos son similares a los índices que hay en los libros. En un libro, un índice permite encontrar información rápidamente sin necesidad de leer todo el libro. En una base de datos, un índice permite que el administrador de la base busque registros en una tabla sin necesidad de examinarla toda. El índice de un libro es una lista de palabras con los números de las páginas en las que se encuentra cada palabra. Un índice de una base de datos es una lista de los valores de una tabla con las posiciones de almacenamiento de las filas de la tabla donde se encuentra cada valor. Se pueden crear índices en una sola columna o en una combinación de columnas de una tabla; los índices se implementan en forma de árboles.

Un índice contiene una entrada con una o varias columnas (la clave de búsqueda) de cada fila de una tabla. Un árbol se ordena con la clave de búsqueda y se puede buscar de forma eficaz en cualquier subconjunto principal de la clave de búsqueda. Mientras que la mayor parte de los libros contienen un índice general de palabras, nombres, lugares, etc., las bases de datos contienen índices individuales para tipos o columnas de datos seleccionados. Es como un libro con un índice para los nombres de las personas y otro índice para los lugares.

Con frecuencia, en una base de datos hay una columna o combinación de columnas cuyo único propósito es identificar cada una de las filas de la tabla. A este tipo de columna se le denomina ***clave principal*** o ***llave primaria***.

Cuando se define la clave principal de una tabla en una base de datos se crea, automáticamente, un índice de clave principal, es decir, un tipo específico de índice único. Es necesario que en este índice cada valor de la clave principal sea único. Asimismo, se utiliza el índice de clave principal en las consultas para tener acceso rápidamente a los datos.

Un índice único es aquel en el que dos filas no pueden tener el mismo valor de índice. La mayoría de las bases de datos evitan que se guarde una tabla con un índice único que se acaba de crear si encuentran valores de clave duplicados en los datos existentes. La base de datos también puede evitar que se agreguen datos nuevos que crearían valores de clave duplicados en la tabla.

Se ha definido anteriormente que la clave principal en la tabla de miembros de la base de datos del sistema es una composición de letras del nombre, apellidos y código postal de los miembros ***(ver figura 2.6)***.

El estándar de búsquedas para todas las versiones de la aplicación *AcadMexderm* ha sido a través de la llave principal, es decir, el código único de identificación. En versiones anteriores, se contaba con un formulario especial para poder generar, en tiempo de ejecución, el código del miembro y así poder ejecutar el proceso de búsqueda del miembro ***(ver figura 2.33)***.

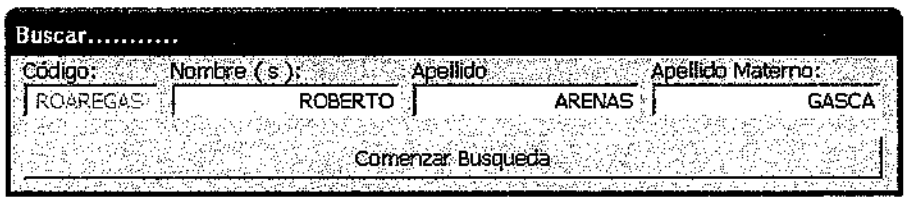


Figura 2.33 → Formulario para búsquedas de la versión anterior

Sin embargo, para las versiones anteriores se debía generar forzosamente el código completo para realizar la búsqueda, es decir, al ingresar el nombre del miembro eran necesarios las primeras letras del nombre y los apellidos, de lo contrario, no se podría generar la consulta; cuando no se ingresaran estos campos obligatorios y se intentara generar la búsqueda, aparecería la pantalla de la ***figura 2.34.***

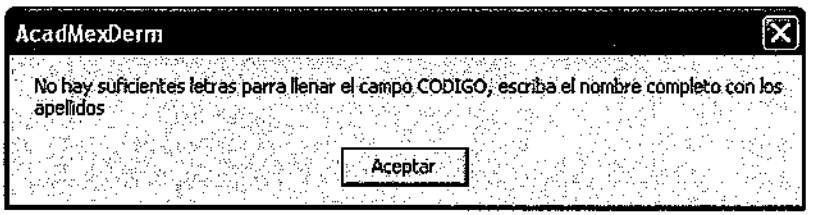


Figura 2.34 → Mensaje de error en las búsquedas para la versión anterior del programa

Otro problema existente hasta la versión anterior es que, dentro del procedimiento de búsqueda, una vez presionado el botón para buscar a un miembro, el programa no poseía la capacidad de cancelar la operación, es decir, si se deseara cancelar la búsqueda habría que ingresar datos basura en el formulario para poder generar una búsqueda sin sentido y que, obviamente, no devolvería ningún resultado.

Es obvio entonces el problema que existe si el usuario en algún caso, no recordara o no tubiera alguno de los campos de búsqueda. Para la nueva versión se han incluido opciones para solucionar estas discrepancias; la primera es que existe la posibilidad de cancelar la búsqueda en cualquier momento, como se muestra en la ***figura 2.35.***

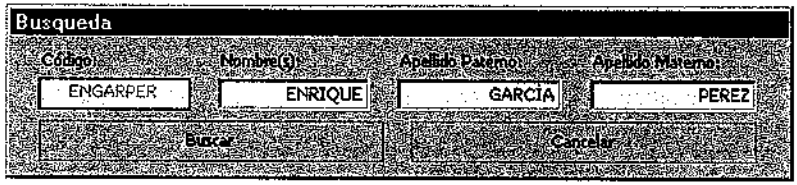


Figura 2.35 → Formulario de búsqueda para la última versión

Otra adición importante para esta nueva versión es que no es necesario tener los tres datos para realizar la búsqueda, al poder ingresar únicamente los dos primeros campos es posible generar una búsqueda general y en el caso de encontrar más de un miembro que concuerden con los datos ingresados, se le mostrará al usuario una nueva pantalla donde podrá seleccionar al miembro que en verdad está buscando, es decir, si el usuario sólo recuerda que el miembro que está buscando es, por ejemplo, Enrique García; existen para este caso más de un Enrique García, de tal forma que al confirmar la búsqueda, al usuario se le mostrará la pantalla de la ***figura 2.36.***

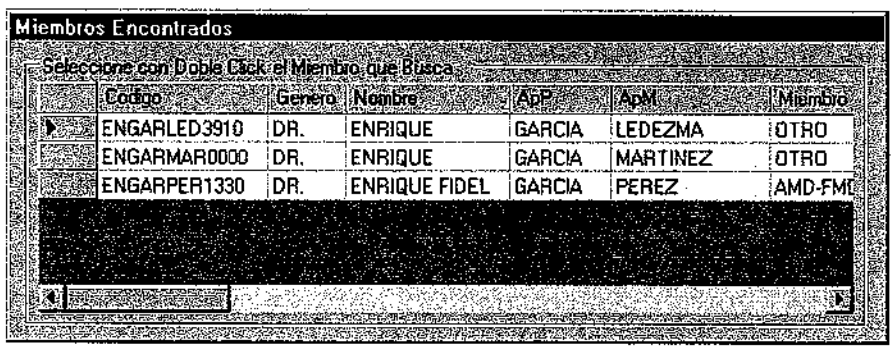


Figura 2.36 → Pantalla de miembros encontrados

Cuando el usuario desee confirmar cuál es el miembro que esta buscando, simplemente deberá dar doble click en el mismo y sus datos aparecerán en el apartado de miembros dentro del *TabPage* con el mismo nombre.

Las búsquedas dentro del sistema se realizan a través de la clave principal de la tabla de los miembros, y conjunto al hecho de que todo el tiempo el sistema se encuentra trabajando con el *DataSet* en memoria, el sistema de búsquedas es un procedimiento óptimo en su funcionamiento y en extremo rápido en su ejecución.

Para poder ver como es que el sistema de información realiza el procedimiento de búsqueda de registros se puede examinar el ***apartado 2.12.9 (Búsqueda de registros en un DataView)*** en el cual se puede observar la forma precisa de el método de búsquedas casi de la misma forma en que se realiza en el programa ***AcadMexDerm .Net.***

2.10.10 Impresiones

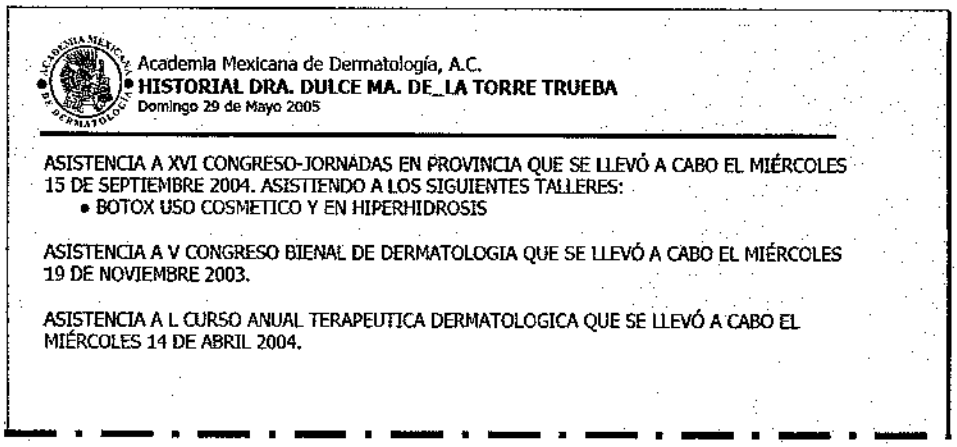
La última forma de refrendar la información contenida dentro de un sistema de información son los reportes impresos. A través de los reportes impresos se puede tener una constancia de los datos contenidos dentro del sistema, y a partir de ellos, tomar mejores decisiones o simplemente tener una evidencia escrita de cierta información.

Para la AMD lo más importante dentro de los reportes impresos del sistema es tener un tiraje de etiquetas que sirvan para poder rotular cualquier tipo de información, incluyendo panfletos, volantes o manifiestos, para informar a sus miembros acerca de cualquier evento próximo a realizarse dentro de la institución.

Otro reporte que contiene muchos documentos son los reportes generales de los miembros, los cuales incluyes todos los datos de todos los miembros contenidos dentro de la base de datos. Este reporte es utilizado para tener un historial de los registros guardados a lo largo de la historia desde la implementación del sistema, y así poder contar con un registro histórico de los miembros, sus adiciones y cambios.

Existen también reportes que están contenidos en una única página; los cuales constituyen un medio adicional para poder ver los datos de los registros contenidos en las tablas. Básicamente se dividen dependiendo de las dos tablas, la de los *congresos* y la de los *miembros*. Estos reportes incluyen todos los campos contenidos dentro de cualquiera de los registros o filas de las dos tablas antes mencionados. Pero sin lugar a dudas, la impresión más importante dentro de los reportes individuales es el reporte del historial de los miembros, el cual, es un documento que incluye de forma escrita la historia de las asistencias para todos los miembros dentro de los eventos registrados en el sistema; esto es, dentro del sistema es posible registrar las asistencias de los miembros, y como se ha mencionado con anterioridad, al registrar al miembro a un nuevo evento, el

registro actual pasa a ser parte de su historial. Para la AMD es importante tener como constancia de estas asistencias una impresión de un documento que funja como un medio único para validar dichas asistencias; así pues, este documento es la única constancia y también la única forma de ver y corroborar los historiales de las asistencias de los miembros. En la ***figura 2.37*** se muestra el formato de este reporte.



Academia Mexicana de Dermatología, A.C.
HISTORIAL DRA. DULCE MA. DE_LA TORRE TRUEBA
Domingo 29 de Mayo 2005

ASISTENCIA A XVI CONGRESO-JORNADAS EN PROVINCIA QUE SE LLEVÓ A CABO EL MIÉRCOLES 15 DE SEPTIEMBRE 2004. ASISTIENDO A LOS SIGUIENTES TALLERES:
- BOTOX USO COSMETICO Y EN HIPERHIDROSIS

ASISTENCIA A V CONGRESO BIENAL DE DERMATOLOGIA QUE SE LLEVÓ A CABO EL MIÉRCOLES 19 DE NOVIEMBRE 2003.

ASISTENCIA A L CURSO ANUAL TERAPEUTICA DERMATOLOGICA QUE SE LLEVÓ A CABO EL MIÉRCOLES 14 DE ABRIL 2004.

Figura 2.37 → Ejemplo del reporte de historial de un miembro

Para este apartado se han dividido los reportes en masivos e individuales para su mejor exposición, siendo el reporte del historial y el tiraje de etiquetas los más importantes. En la ***figura 2.38*** se puede realizar un análisis a fondo de todos los reportes impresos con que cuenta el sistema de información ***AcadMexDerm*** para su última versión.

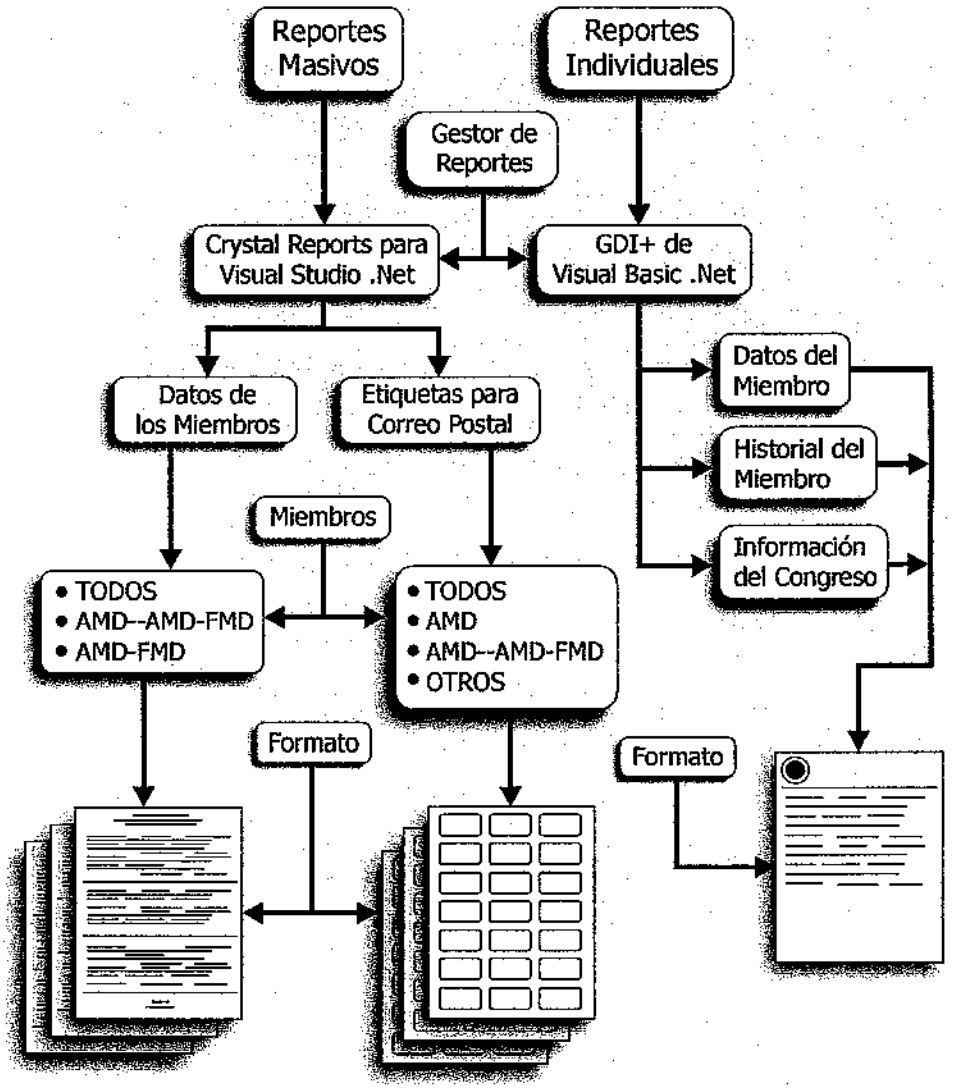


Figura 2.38 → Reportes Impresos

Se utilizan dos gestores diferentes para la creación de los documentos, dependiendo en mayor parte al número de páginas que se necesitan para cada reporte; pero lo que comparten en común es que la fuente de datos para cualquiera sigue siendo el *DataSet*, y por lo tanto, la información que se está imprimiendo son los registros que se encuentran en memoria y donde se incluyen los registros que se encuentren en producción desde que se ha iniciado la aplicación.

Los tirajes de etiquetas se imprimen en un tipo de hoja especial llamada ***Avery 5160,*** la cual constituye una serie de etiquetas en blanco contenidas en una hoja encerada en tamaño carta, la cual se puede imprimir como si fuera una hoja de *papel bond* cualquiera. Esta impresión esta ordenada por código postal de forma ascendente, ya que esto es requisito dentro de las oficinas postales cuando se desea enviar una cantidad considerable de sobres, y de esta manera, realizar el tramite de envío con mayor rapidez.

En resumen, es posible afirmar que la aplicación cuenta con los medios necesarios para reportar en forma de impresión todos los registros contenidos en la base de datos, mostrándolos de forma coherente y organizada además de poder imprimir la información más recientemente ingresada en el sistema de información.

2.11 Implementación y Mantenimiento de Versiones

La implementación es el proceso mediante el cual se distribuye una aplicación o un componente para su instalación en diferentes equipos de cómputo. En la aplicación *AcadMexDerm* la implementación se basa en la tecnología de ***Microsoft Windows Installer.***

Microsoft Windows Installer es un servicio de instalación y configuración que se incluye como parte de *Windows 2000, Windows Me y Windows XP.* Además, está disponible para *Windows 95, Windows 98* y *Windows NT 4.0.*

Con *Windows Installer*, todos los equipos mantienen una base de datos de información sobre cada aplicación que se instala, incluidos los archivos, las claves del Registro y los componentes. Cuando se desinstala una aplicación, se comprueba la base de datos para asegurarse de que ninguna otra aplicación precisa un archivo, clave de registro o componente antes de suprimirlo, lo que evita que la eliminación de una aplicación afecte al buen funcionamiento de otra.

Windows Installer es además compatible con la reparación automática, es decir, la posibilidad de que una aplicación reinstale automáticamente los archivos que falten en el caso de que el usuario los haya borrado sin darse cuenta.

Además, *Windows Installer* proporciona la posibilidad de dar marcha atrás en una instalación. Por ejemplo, si una aplicación precisa una base de datos determinada y ésta no se encuentra en el equipo, la instalación puede anularse y el equipo volverá al estado que tenía antes de comenzar el proceso.

Las herramientas de implementación de *Visual Studio .NET* se basan en los fundamentos de *Windows Installer*: proporcionar un rico conjunto de posibilidades para una rápida implementación y mantenimiento de aplicaciones generadas con *Visual Studio .NET.*

Para la aplicación *AcadMexDerm* se ha creado un instalador especial para todos los sistemas operativos basados en Windows; creando así un paquete instalador *Microsoft Windows Installer (*.msi)*, Este proyecto de instalación permite crear un instalador para distribuir la aplicación junto con cualquier archivo dependiente, información sobre la aplicación, como las entradas en el Registro de Windows (que para el sistema de información son innecesarias), e instrucciones para la instalación. Cuando el archivo *.msi* se distribuye y ejecuta en otro equipo, se puede garantizar que todos los elementos necesarios para la instalación se han incluido; si por cualquier razón la instalación falla (por ejemplo, si el equipo de destino no tiene la versión requerida del sistema operativo) el proceso anulará los pasos realizados para que el equipo quede en el mismo estado en que se encontraba antes de iniciar la instalación.

Todas las aplicaciones y componentes de *Visual Studio .NET* que utilizan *.NET Framework* requieren la instalación de la versión correcta de *Common Language Runtime* en el equipo donde se ejecuta la aplicación. Todos los instaladores creados mediante la implementación de *Visual Studio .NET* también requieren tener instalado *.NET Framework;* por tanto, no es posible instalar *.NET Framework* como parte de un proyecto de implementación. *.NET Framework* se debe instalar mediante su elemento redistribuible antes de instalar la aplicación *AcadMexDerm.*

Toda aplicación o componente de *Visual Studio .NET* que incluya acceso a datos tendrá una dependencia en ***Microsoft Data Access Components (MDAC) 2.7*** o posterior. La dependencia no puede ser detectada por el proyecto de implementación; si no se instala *MDAC* en el equipo de destino antes de instalar la aplicación, la aplicación se instalará correctamente, pero se producirá un error al ejecutarse.

Para el uso de la aplicación de han creado dos tipos de versiones, una versión para la edición, adición y eliminación de registros, llamada "SERVER" y otra que es exclusivamente para examinar y buscar registros, crear e imprimir reportes llamada "WORKSTATION". La versión *SERVER* se instala únicamente en un equipo

de las instalaciones de la AMD, en donde se lleva todo el control de los registros, sus actualizaciones y la inserción de nuevos miembros dentro del sistema. Para los otros dos equipos así como los equipos personales de la mesa directiva, se instala la versión *WORKSTATION* la cual cuenta con todo lo necesario para visualizar todos los registros que se ingresan en el equipo con la versión *SERVER*; es decir, cuando se inicia la aplicación en los equipos remotos, la base de datos con la que trabajan es una copia en memoria de la base original alojada en el equipo que funge como servidor de datos, como se mostró en la ***figura 1.5***, en donde se apreció el esquema de usuarios dentro de la oficina de la *AMD*.

Una vez instalada la aplicación en los equipos personales de la mesa directiva es necesaria la implementación de la base de datos dentro de sus equipos, ya que no todos pueden estar conectados remotamente al servidor de datos, para esto se ha creado un apartado especial dentro de la página de Internet en donde la mesa directiva de la *Academia Mexicana de Dermatología, A.C.* puede bajar un instalador con la última versión de la base de datos que incluye los últimos registros adicionados al sistema. Pero de esto se hablará más adelante, basta decir que el instalador de la versión *WORKSTATION* incluye un acceso directo a un vínculo de acceso a datos *(como se mostró en la **figura 2.11**)* para que el usuario pueda vincular el archivo de *Access* instalado al programa y así poder ejecutarlo de manera correcta.

Para cada uno de los cambios que se han realizado desde la liberación de la primera subversión de la aplicación *AcadMexDerm 6 .NET* se han creado empaquetados compresos que incluyen la versión y su fecha de creación. Estos empaquetados se almacenan en dos equipos diferentes y aparte se agrega dentro de un disco compacto reescribible para su almacenamiento y clasificación. Es de esta forma que se han mantenido un archivo de versiones de la aplicación y hasta hoy no ha sido necesario llevar un control más complejo.

2.12 Apartado para el Programador Avanzado de Visual Basic .Net

En este pequeño apartado se demuestra a través de ejemplos sencillos y prácticos el diseño de aplicaciones Windows vinculadas a Bases de Datos con ADO .NET. Se demuestra la mejor forma de conectar una base de datos a cualquier sistema de una forma clara y concisa, demostrando el uso de un ***DataSet***; se indicará la forma de acceder a los datos utilizando varios formularios y objetos a través de un único ***DataSet***, se utiliza un objeto ***DataView*** para poder vincular los elementos de los formularios para crear vistas con diferentes criterios de ordenación, con lo cual se podrán filtrar los datos y presentar su contenido de manera ordenada y dinámica; se demuestra como declarar los objetos derivados de la clase ***System.Data*** del .Net Framework, parametrizar sus constructores y vincularlos con los componentes habituales de los formularios de Windows. Se utiliza también el control ***DataGrid*** de Windows para ordenar y seleccionar los datos de un ***DataTable*** a través del ***DataView*** de la aplicación y manipula el objeto ***ActiveX Outlook*** para enviar correos electrónicos a través de formularios de *Windows Forms*; todo esto con el firme propósito de aprender la mejor forma de programar dichos módulos e integrarlos a las aplicaciones de forma profesional y que contribuyan a que los programas sean completamente funcionales y competitivos.

2.12.1 Conexión a la Base de Datos y Creación del DataSet

Se comenzará creando una conexión sencilla a una base de datos de ejemplo. Para todos los ejemplos expuestos, se utilizará una base de datos llamada ***"directorio.mdb"*** creada en *Access 2003*; esta base de datos contiene una única tabla llamada también ***"directorio"*** que contiene tres columnas y sus correspondientes descripciones expuestas en la ***tabla 2.18:***

Nombre del Campo	Tipo	Tamaño	Características	Descripción
nombre	Texto	50	Llave Principal, Requerido, NO duplicado	Nombre del contacto del directorio.
telefono	Texto	30	No Requerido	Número telefónico
email	Texto	30	No Requerido	Correo Electrónico

Tabla 2.18 → Descripción de Columnas de la Tabla "directorio"

Cabe mencionar que se ha creado un proyecto de *Visual Basic .Net* de *Visual Studio .Net versión 2003* en una carpeta especialmente creada para este propósito con ubicación:

C:\Proyecto\...

En dicha dirección se ubican todas las partes de la aplicación, así como la base de datos que se ha descrito anteriormente.

Cuando se crea un proyecto con más de un formulario interactuando entre si no es posible acceder a la información privada de las clases derivadas de dichos formularios, para este propósito se puede crear un *Módulo Global* que contenga todos los objetos o variables que se deseen acceder desde cualquier formulario del proyecto; este modo de trabajar sirve también para proteger los códigos de una manera simple y sencilla, ya que si no se adjunta el *módulo global* al proyecto no se podrá compilar, dado que le habrán quitado los objetos más importantes para que la aplicación funcione de manera correcta.

Lo primero es crear una Aplicación Windows de Visual Basic .Net llamada ***"Directorio"*** como se muestra en la ***figura 2.39***:

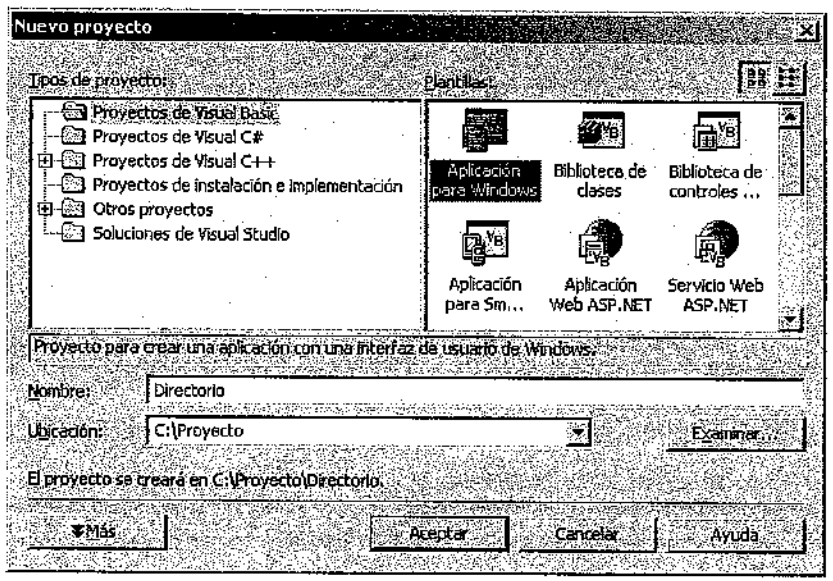


Figura 2.39 → Creación de una nueva aplicación de Windows

Una vez creado el proyecto, se le ha cambiado el nombre del formulario *Form1.vb* por *frmDirectorio.vb*, así mismo se ha cambiado la propiedad ***Name=****Form1* por ***Name=****frmDirectorio*, y por último se ha sustituido la propiedad ***Text=****Form1* por ***Text=****MyDirectorio* como se muestra en la ***figura 2.40:***

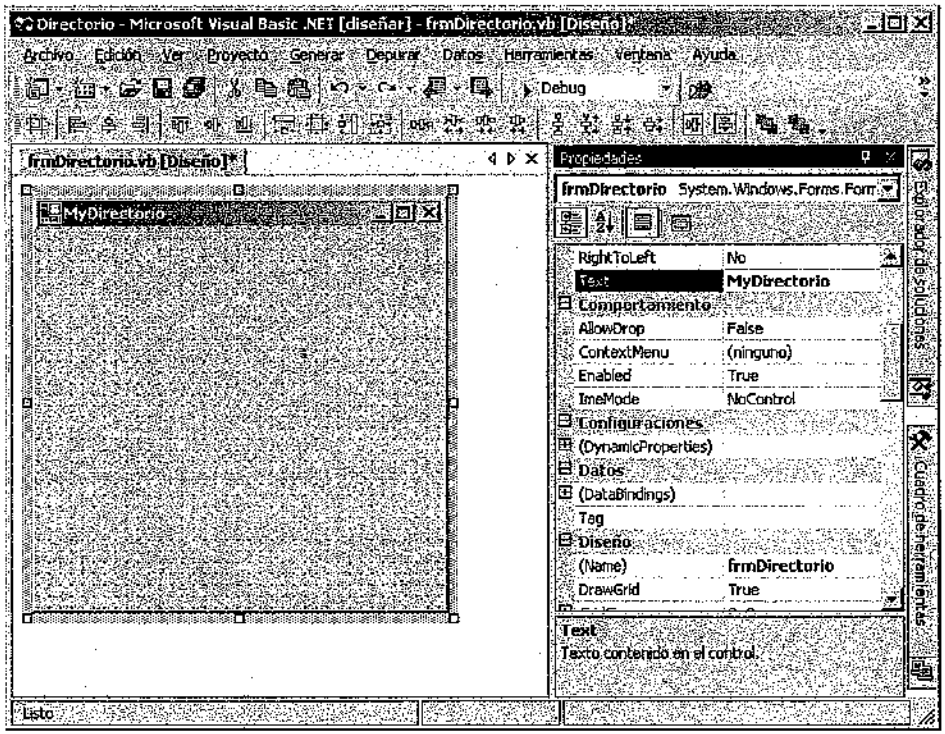


Figura 2.40 → Propiedades del Formulario "frmDirectorio"

Una vez creado el proyecto y creadas las carpetas del mismo, se debe crear un archivo de ***"Vínculo de Datos de Microsoft"*** en la carpeta ***"bin"*** del proyecto. Para crear este vínculo simplemente se crea un archivo llamado ***"vinculo.udl"*** con cualquier editor de texto simple. Este archivo contendrá la información del controlador *OLE DB* para conectarse a la base de datos, que en este caso, es un *Archivo de Access 2003*. Este vínculo resulta realmente versátil cuando se piensa en redistribuir la aplicación, ya que a través de él se puede cambiar de proveedor *OLE DB* sin tener que cambiar una sola línea de código fuente, por lo que se tendría que recompilar el proyecto cada vez que se cambiara proveedor, es decir, si se cambiara de base de datos, siempre y cuando se conserven las tablas con sus respectivas columnas, no habría necesidad de cambiar nada del programa.

También este tipo de versatilidad se aplica a la ubicación de la base de datos, es decir, se puede cambiar la ubicación de la base de datos a un servidor remoto y simplemente se cambiaría la dirección de la conexión directamente en este archivo sin tener que cambiar nada a la aplicación; también pasaría lo mismo si se estuviera conectado a un servidor de base de datos y este cambiara su ubicación en la Red.

Para utilizar el *Vínculo a Datos de Microsoft* se da doble click en el archivo ***vinculo.udl*** creado anteriormente y se observará la pantalla de la ***figura 2.41:***

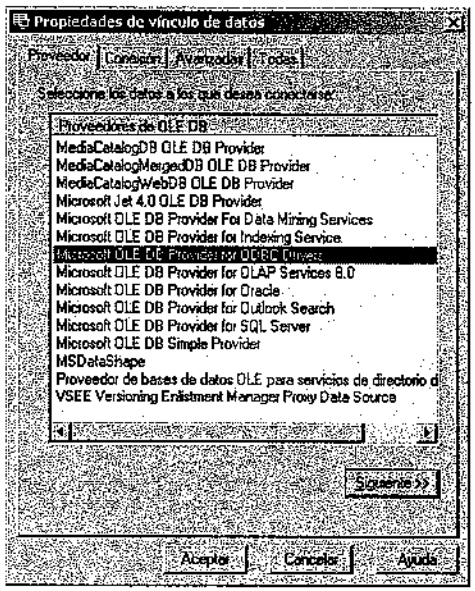


Figura 2.41 → Propiedades del Vínculo a Datos de Microdoft

En la pestaña *"Proveedor"* se selecciona: ***"Microsoft Jet 4.0 OLE DB Provider"***, se da click en el botón → ***Siguiente*** y en la sección de ***"selección de base de datos"*** se ubicará la posición del archivo *directorio.mdb*, que para este ejemplo, la ubicación completa sería:

C:\Proyecto\Directorio\bin\directorio.mdb

A continuación se presiona el botón *"Probar Conexión"* para que aparezca un aviso de: ***"La prueba de conexión fue satisfactoria".*** En la pestaña de *Avanzadas* se habilitará la opción ***"ReadWritey"*** para poder realizar cambios sobre la base de datos y tener permisos para poder realizar tales acciones.

Con lo anterior, lo único que se intenta generar es una *Cadena de Conexión OLE DB*, la cual se podría analizar si se abre el archivo en un editor de texto cualquiera y se observaría el contenido, que sería muy parecido a lo siguiente:

```
[oledb]
; Everything after this line is an OLE DB initstring
Provider=Microsoft.Jet.OLEDB.4.0;
Data Source=C:\Proyecto\Directorio\bin\directorio.mdb;
Mode=ReadWrite|Share Deny None;
Persist Security Info=False
```

Lo anterior se ha separado en varias líneas para un mejor análisis, pero corresponde a una sola línea, la cual representa únicamente la ubicación del archivo *.mdb, el proveedor OLE DB que se ha elegido para la conexión y los permisos que se tienen para realizar cambios sobre el mismo; como se verá más adelante, se podrá cambiar dentro del código fuente la línea de conexión e introducir directamente la cadena de conexión OLE DB, pero se inhibiría una parte muy importante del sistema, que es el poder realizar cambios externos sobre la base de datos sin tener que afectar a la aplicación.

Regresando al proyecto de *Visual Basic*, se accederá a la opción ***"Agregar nuevo elemento..."*** del menú ***"Proyecto"*** o se presiona ***"Ctrl+Mayús+A"*** para agregar un ***"Módulo"*** llamado ***"ModuloGlobal.vb"*** al proyecto como se muestra en la ***figura2.42:***

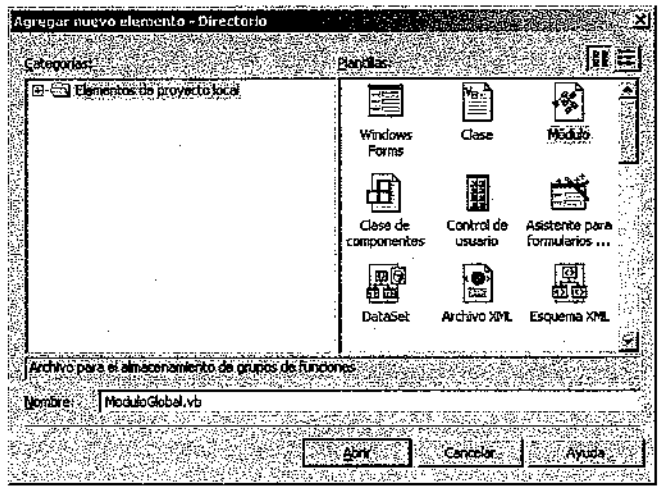


Figura 2.42 → Adición del Módulo Global al Proyecto

A continuación se agregará la línea del ***listado 2.1*** al Módulo Global:

```
'Conexión
Public myConnection As New System.Data.OleDb.OleDbConnection
```

Listado 2.1 → Declaración del OleDbConnection

Es importante declarar a *"myConnection"* como un objeto público, para que pueda ser accedido desde cualquier formulario del proyecto; *myConnection* representa una conexión abierta a un origen de datos, la cual, conecta al sistema a través de la *cadena de conexión OLE DB* del archivo *vinculo.udl*, esto se puede realizar agregando las líneas del ***listado 2.2*** en la llamada al evento ***LOAD*** del formulario principal *frmDirectorio*:

```
'Con la siguiente línea se define al directorio actual como el directorio de la aplicación
ChDir(Application.StartupPath)
'Controlando algún probable error, se genera la conexión a la base de datos
Try
        myConnection.ConnectionString = "File Name = vinculo.udl;"
Catch ex As Exception
        MessageBox.Show(ex.Message)
End Try
```

Listado 2.2 → Vinculación de *myConnection* al archivo *vinculo.udl*

A continuación, en la misma sección de código, se generará una cadena que contendrá la consulta simple SQL con la cual se podrá obtener los datos que se necesiten de la tabla *directorio* de la base de datos agregando las líneas del ***listado 2.3*** al evento ***LOAD*** del formulario principal *frmDirectorio:*

```
'Query de Consulta
Dim myQuery As String
myQuery = "SELECT * FROM directorio ORDER BY Nombre"
```

Listado 2.3 → Consulta SQL

Si se vincularan los datos a un control *DataGrid* y se deseara presentar las columnas de la tabla en cierto orden, es en esta sección en donde se deben realizar los cambios para poder mostrarlos en dicho orden, por ejemplo, si se dejara la consulta tal cual, el orden sería primero el *Nombre*, luego el *Teléfono* y por último el *Correo Electrónico* del *directorio*, pero si se deseara invertir el orden en el *DataGrid* la consulta debería ser con las líneas presentadas en el ***listado 2.4.***

```
'Query de Consulta
Dim myQuery As String
myQuery = "SELECT email, telefono, nombre FROM directorio ORDER BY Nombre"
```

Listado 2.4 → Consulta SQL en distinto orden para un DataGrid

A continuación se adicionará otra declaración al *Módulo Global*, el cual representa un *Adaptador de Datos*; este adaptador es simplemente un conjunto de comandos de datos y una conexión de base de datos que se utilizan para rellenar un *DataSet* y actualizar el origen de datos, se adicionarán las líneas del ***listado 2.5*** al evento *LOAD* del formulario.

```
'Adaptador de datos
Public myAdapter As System.Data.OleDb.OleDbDataAdapter = New OleDb.OleDbDataAdapter
```

Listado 2.5 → Declaración del Adaptador de Datos

Es importante resaltar que se ha declarado a *myAdapter* como un objeto público, además de que es importante declararlo como un nuevo objeto de tipo *OleDb.OleDbDataAdapter*, de lo contrario, no funcionará. Regresando al código del formulario *frmDirectorio*, se agregarán las líneas del ***listado 2.6*** a la ya mencionada sección del evento ***LOAD*** de dicho formulario:

```
'Controlando algún probable error, se genera el Adaptador de Datos para frmDirectorio
Try
        myAdapter.SelectCommand = New OleDb.OleDbCommand(myQuery, myConnection)
Catch ex As Exception
        MessageBox.Show(ex.Message)
        Exit Sub
End Try
```

Listado 2.6 → Instauración del Adaptador de Datos

En el evento ***CLOSING*** del formulario *frmDirectorio* se agregarán las líneas del ***listado 2.7*** para cerrar la conexión antes de que se cierre la aplicación, en el caso de que se conserve abierta:

```
'Si la conexión sigue abierta, se cerrará
If myConnection.State = ConnectionState.Open Then
        Try
                myConnection.Close()
        Catch ex As Exception
                MessageBox.Show(ex.Message)
        End Try
End If
```

Listado 2.7 → Cierre de Conexión

Ahora se está a punto de generar el *DataSet*, para esto agregará la declaración del ***listado 2.8*** al *módulo global*.

```
'Declaración del DataSet
Public myData_set As System.Data.DataSet
'El Constructor de Comandos genera automáticamente comandos de tabla única utilizados para
conciliar los cambios realizados en el DataSet con la base de datos asociada. Esta clase no se
puede heredar y es necesaria para las actualizaciones
Public myCommand_builder As System.Data.OleDb.OleDbCommandBuilder
```

Listado 2.8 → Declaración del *DataSet* y del *Constructor de Comandos*

El objeto *DataSet* es esencial para admitir contextos de datos distribuidos de ADO.NET sin mantener una conexión. El objeto *DataSet* es una representación residente en memoria de datos que proporciona un modelo de programación relacional coherente independientemente del origen de datos. Se puede utilizar con múltiples y distintos orígenes de datos, como XML. El *DataSet* representa un conjunto completo de datos entre los que se incluyen tablas relacionadas, restricciones y relaciones entre las tablas. En la ***figura 2.43*** se muestra el modelo de objeto *DataSet*.

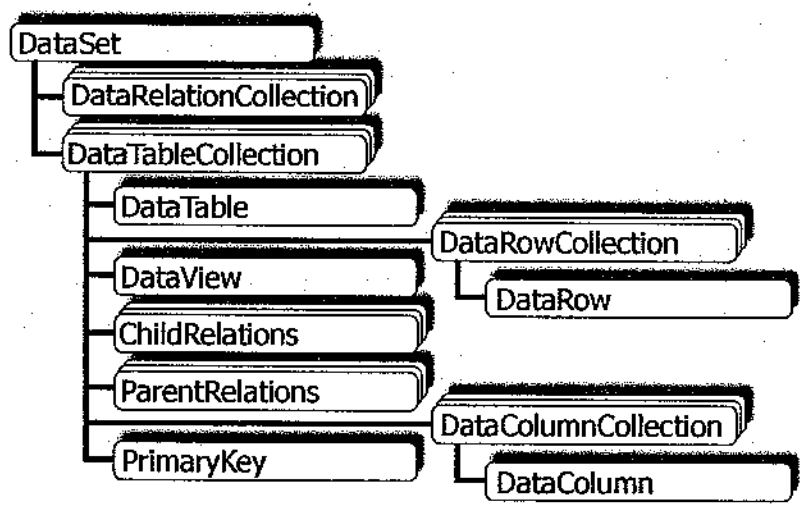


Figura 2.43 → Modelo de objeto *DataSet*

Los métodos y objetos contenidos en un *DataSet* son coherentes con los del modelo de base de datos relacional. El *DataSet* también puede persistir y volver a cargar su contenido como ***XML*** y su estructura como esquema *XSD (Lenguaje de definición de esquemas XML).*

Un *DataSet* de ADO.NET contiene una colección de cero o más tablas representadas por objetos *DataTable.* La *DataTableCollection* contiene todos los objetos *DataTable* de un *DataSet.*

Un *DataTable* se define en el espacio de nombres *System.Data* y representa una única tabla de datos residentes en memoria. Contiene una colección de columnas representadas por una *DataColumnCollection* y restricciones representadas por una *ConstraintCollection* que, juntas, definen el esquema de la tabla. Un *DataTable* también contiene una colección de filas representadas por la *DataRowCollection,* que contiene los datos de la tabla. Junto con su estado actual, un *DataRow* conserva tanto la versión original como la actual para identificar los cambios realizados en los valores almacenados en la una fila.

Un *DataSet* contiene relaciones en su objeto *DataRelationCollection.* Una relación, representada por el objeto *DataRelation,* asocia las filas de un *DataTable* con las filas de otro *DataTable.* Es análogo a una ruta de unión que podría existir entre las columnas de claves externas y principales en una base de datos relacional. Un *DataRelation* identifica columnas coincidentes en dos tablas de un *DataSet.*

Las relaciones permiten pasar de una tabla a otra dentro de un mismo *DataSet.* Los elementos esenciales de un *DataRelation* son el nombre de la relación, el nombre de las tablas relacionadas y las columnas relacionadas de cada tabla. Se pueden establecer relaciones con más de una columna por tabla, para lo que se debe especificar una selección de objetos *DataColumn* como columnas clave. Cuando se agrega una relación al *DataRelationCollection,* se puede agregar también un *UniqueKeyConstraint* y un *ForeignKeyConstraint* para imponer restricciones de integridad cuando se realicen cambios en los valores de las columnas relacionadas.

Dentro del evento ***LOAD*** de *frmDirectorio* se agregarán las líneas del ***listado 2.9*** para generar el *DataSet* y llenarlo con los datos a través del *Adaptador de Datos* y el *Constructor de Comandos.*

```
'Como parámetro del constructor del DataSet se debe proporcionar el nombre
'del mismo
Try
        myData_set = New DataSet("DataSetDirectorio")
Catch ex As Exception
        MessageBox.Show(ex.Message)
        Exit Sub
End Try

'Se genera el Constructor de Comandos agregando en su constructor el Adaptador de Datos
Try
        myCommand_builder = New OleDb.OleDbCommandBuilder(myAdapter)
Catch ex As Exception
        MessageBox.Show(ex.Message)
        Exit Sub
End Try

'Se abre la Conexión para llenar el DataSet
Try
        myConnection.Open()
Catch ex As Exception
        MessageBox.Show(ex.Message & ControlChars.CrLf & "Diríjase al archivo vinculo.udl y
seleccione la ubicación correcta de la base de datos directorio.MDB")
        Exit Sub
End Try

'Se llena el DataSet
If myConnection.State = ConnectionState.Open Then
        Try
            'Se agregan las columnas necesarias e información sobre la clave principal para
            completar el esquema.
            myAdapter.MissingSchemaAction = MissingSchemaAction.AddWithKey
            'Se llena el DataSet con la tabla completa "directorio" de la base de datos
            myAdapter.Fill(myData_set, "directorio")
            'Se cierra la Conexión, ya que no será necesaria porque ya se tienen todos los
            datos dentro del DataSet
            myConnection.Close()
        Catch ex As Exception
            MessageBox.Show(ex.Message)
            Exit Sub
        End Try
End If
```

Listado 2.9 → Instauración del *DataSet* a través del *Adaptador de Datos* y del *Constructor de Comandos*

Hasta ahora se ha logrado generar el *DataSet* con éxito, a continuación se adicionará un *DataTable*, un *DataRowCollection* y un *DataView* dentro del *Módulo Global* agregándole las líneas del ***listado 2.10:***

```
'El DataTable
Public myTable As DataTable
'El RowCollection
Public myRow_collection As DataRowCollection
'El DataView
Public myData_view As DataView
```

Listado 2.10 → Declaración del *DataTable*, *DataRowCollection* y del *DataView*

Se debe recordar que el *DataTable* simplemente representa una tabla de datos en memoria, el *DataRowCollection* representa una colección de filas para un *DataTable* y el *DataView* representa una vista personalizada que puede enlazar datos de un *DataTable* para ordenación, filtrado, búsqueda, edición y exploración.

Para poder generar estos elementos se deben agregar las líneas del ***listado 2.11*** al evento ***LOAD*** del formulario *frmDirectorio:*

```
'Se genera la Tabla a partir del DataSet
Try
        myTable = myData_set.Tables("directorio")
Catch ex As Exception
        MessageBox.Show(ex.Message)
        Exit Sub
End Try

'Se genera la colección de tablas para ediciones y cambios posteriores
Try
        myRow_collection = myTable.Rows
Catch ex As Exception
        MessageBox.Show(ex.Message)
        Exit Sub
End Try
```

```
'Se crea el Generador Vistas Personalizadas
Try
        myData_view = New DataView(myData_set.Tables("directorio"))
        'Se debe dar permisos al DataView para Borrar, Editar y Crear Nuevos Registros
        myData_view.AllowDelete = True
        myData_view.AllowEdit = True
        myData_view.AllowNew = True
        myData_view.ApplyDefaultSort = True
Catch ex As Exception
        MessageBox.Show(ex.Message)
        Exit Sub
End Try
```

Listado 2.11 → Instauración del *DataTable*, *DataRowCollection* y del *DataView*

Para terminar con la conexión a la base de datos, se puede ubicar en un evento especial del programa el código necesario para actualizar los cambios que se realicen en la aplicación a los registros de la base de datos. Se debe recordar que mientras el programa este funcionando se estará trabajando en los registros del *DataSet* y el sistema se mantendrá desconectado de la base de datos original, por lo que lo que se debe crear el procedimiento para actualizar los datos en el origen; en el ejemplo se ubicará dicho procedimiento dentro del evento ***CLOSING*** del formulario principal *frmDirectorio* agregando las líneas del ***listado 2.12*** a dicho evento y con esto se actualizarán todos los cambios antes de cerrar la aplicación:

```
'Antes de cerrar la aplicación, se actualizarán los cambios
'directamente en la base de datos
If myConnection.State = ConnectionState.Closed Then
    Try
        'Se reabrirá la conexión, ya que ya que ésta ha permanecido
        'desconectada mientras el programa ha estado activo
        myConnection.Open()
        'Con esta línea se actualizan los cambios
        myAdapter.Update(myData_set, "directorio")
        'Ahora se vuelve a cerrar la conexión
        myConnection.Close()
    Catch ex As Exception
        MessageBox.Show(ex.Message)
    End Try
End If
```

Listado 2.12 → Actualización de los datos del DataSet antes de cerrar la aplicación

En este momento la aplicación esta lista para ser compilada (presionando **F5**), y si todo ha funcionado bien, únicamente se observará el formulario vacío, pero esto no significa que no ha pasado nada, si no que NO se ha mostrado ningún mensaje de error, entonces se podrá afirmar que la conexión se ha realizado de manera exitosa.

2.12.2 Vinculación de Datos a Formularios Windows Forms

La aplicación se ha conectado exitosamente a la base de datos; es hora de mostrar los datos vinculándolos a controles de formularios Windows Forms. Para esto se agregará al formulario los controles listados en la ***tabla 2.19:***

Objeto	Propiedad	Valor
Label	Name Text	**lblNombre** Nombre:
TextBox	Name Text MaxLength	**txtNombre** (Vacío) 50
Label	Name Text	**lblTelefono** Teléfono:
TextBox	Name Text MaxLength	**txtTelefono** (Vacío) 30
Label	Name Text	**lblEmail** Correo Electrónico:
TextBox	Name Text MaxLength	**txtEmail** (Vacío) 30

Tabla 2.19 → Controles agregados al formulario frmDirectorio

Cuando se hayan agregadó los controles, se podrá observar el formulario tal y como se muestra en la ***figura 2.44:***

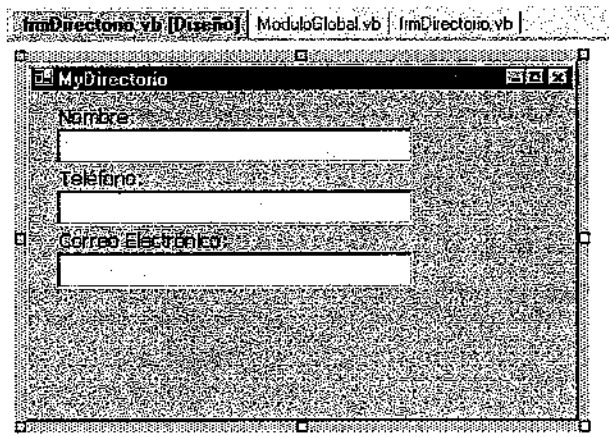


Figura 2.44 → Formulario frmDirectorio

Se ha creado con anterioridad un objeto *DataView*; este objeto permite crear diferentes vistas de los datos almacenados en una *DataTable*, una capacidad que suele utilizarse en aplicaciones de enlace a datos.

Mediante un *DataView* se pueden exponer los datos de una tabla con distintos criterios de ordenación y se pueden filtrar los datos por el estado de una fila o basándose en una expresión de filtro. Un *DataView* proporciona una vista dinámica de los datos cuyo contenido, ordenación y pertenencia reflejan cambios en el *DataTable* subyacente a medida que se producen.

Como un ejemplo se utilizará el *DataView* que se ha instaurado en el ***listado 2.11*** por su facilidad para diversificar las vistas de los datos. Para este propósito, se ha creado una función llamada: ***Enlace_Datos_Cajas_de_Texto***. Dicha función servirá para enlazar los datos de la tabla *directorio* a los controles establecidos anteriormente; en el caso de que se presente un error, dicha función devolverá un valor negativo para poder realizar alguna acción en especifico, que para éste ejemplo, será cerrar la aplicación; es posible ver el código de dicha función en el ***listado 2.13:***

```
Private Function Enlace_Datos_Cajas_de_Texto() As Boolean
    Dim bRespuesta As Boolean = False
    Try
        'El enlace sencillo a un control de Windows Forms
        'se realiza con los siguientes parámetros:
        '(propiedad_del_control, _
        'El_data_view, _
        'nombre_de_la_columna_a_enlazar)
        txtNombre.DataBindings.Add("Text", myData_view, _
        "nombre")
        txtTelefono.DataBindings.Add("Text", myData_view, _
        "telefono")
        txtEmail.DataBindings.Add("Text", myData_view, _
        "email")
        bRespuesta = True
    Catch ex As Exception
        bRespuesta = False
        MessageBox.Show(ex.Message)
    End Try
    Return bRespuesta
End Function
```

Listado 2.13 → Función *Enlace_Datos_Cajas_de_Texto*

La sobrecarga de la función *Add* de la propiedad *DataBinding* de las cajas de texto recibe tres parámetros, el nombre de la propiedad del control al que se va a enlazar, el objeto que representa el origen de datos (el *DataView*) y el miembro con el que se va a enlazar. Se agregará al evento ***LOAD*** del formulario

frmDirectorio la llamada a la función ***Enlace_Datos_Cajas_de_Texto()*** como se muestra en el ***listado 2.14*** para poder enlazar los registros del *DataSet* a los controles del formulario.

```
'Se enlazan los datos de la tabla Miembros con los controles del
formulario
If Enlace_Datos_Cajas_de_Texto() = False Then
      End
End If
```

Listado 2.14 → Llamada a la Función de Enlace a los controles del formulario

Ahora se puede compilar la aplicación y ver el resultado, el cual será algo muy parecido a lo mostrado en la ***figura 2.45***. Cabe aclarar que se ha agregado con anterioridad algunas filas a la tabla *directorio* para poder realizar las pruebas necesarias.

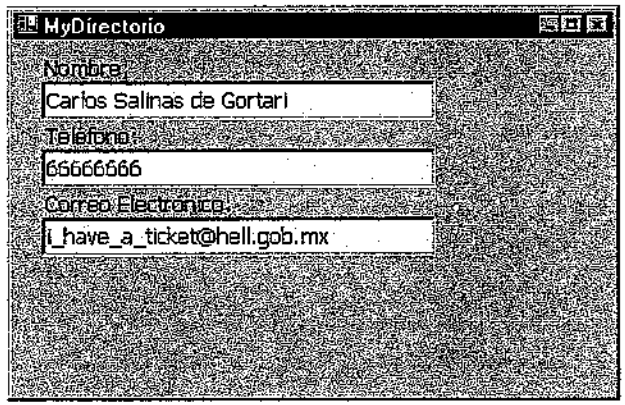


Figura 2.45 → Formulario Compilado

De esta forma se tendrán enlazados las diferentes columnas de la tabla a las Cajas de Texto del formulario. Se han enlazado dichos controles al *DataView* por una simple razón, si se deseara generar una consulta, o dicho de otra forma, mostrar un subconjunto específico de los registros de la tabla, el *DataView* permite hacerlo de forma sencilla, aparte de que cualquier cambio que se realice sobre los registros se verá reflejado inmediatamente en el *DataSet*, y viceversa. El *DataView* permite también ordenar los registros para mostrarlos en cierta disposición y regresarlos al estado original. Para todos estos casos se debe dar permisos al *DataView* para poder realizar cualquier cambio sobre los registros, pero esto ya se ha realizado como se muestra al final del ***listado 2.11***, si no se dieran dichos privilegios, el realizar cambios, eliminar o adicionar registros sería imposible.

2.12.3 Desplazamiento a través de los Registros

Para desplazarse a través de los registros de la base de datos se deben agregar varios elementos a la ventana, es posible ver la lista de los mismos en la ***tabla 2.20.***

Objeto	Propiedad	Valor
Button	Name Text	**btAnterior** Anterior
Button	Name Text	**btSiguiente** Siguiente
Button	Name Text	**btInicio** Inicio
Button	Name Text	**btFinal** Final
GroupBox	Name Text	**GroupBoxNoRegistro** Registro
Label	Name Text TextAlign BorderStyle	**lbNoRegistro** (Vacío) MiddleCenter FixedSingle

Tabla 2.20 → Controles agregados al formulario frmDirectorio

Cuando se hayan agregado los controles, se podrá observar el formulario tal y como se muestra en la ***figura 2.46:***

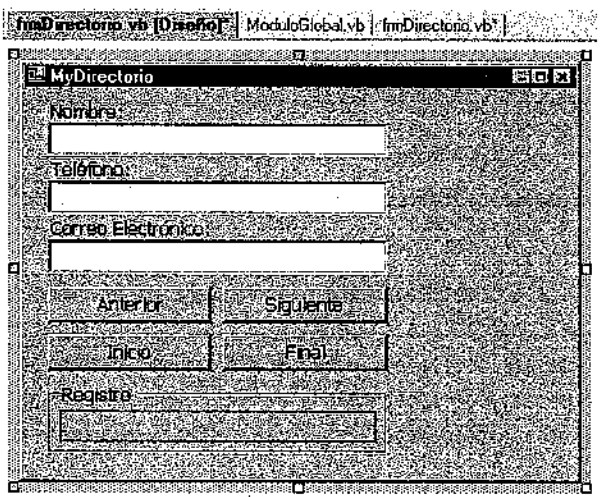


Figura 2.46 → Formulario frmDirectorio

Para poder desplazarse a través de los registros del *DataView* simplemente se accederá a la clase ***BindingContext*** que posee el *DataView*. La clase ***Binding*** representa un enlace sencillo entre el valor de propiedad de un objeto y el valor de propiedad de un control. ***BindingManagerBase*** administra todos los objetos *Binding* enlazados al mismo origen y miembro de datos. *BindingContext* administra la colección de objetos *BindingManagerBase* para cualquier objeto que herede de la clase ***Control***, en este caso, el *DataView*.

Lo primero a realizar es una función que ayude a mostrar la posición en la que se encuentra actualmente. Para esto se agrega la función ***Muestra_Posicion*** del ***listado 2.15***.

```
Private Function Muestra_Posicion()
      lbNoRegistro.Text = BindingContext(myData_view).Position + 1 _
      & " de " & BindingContext(myData_view).Count
End Function
```

Listado 2.15 → Función Muestra_Posición

Lo segundo será ubicarse en el primer registro al iniciar el programa, por lo que se agregará al evento ***LOAD*** del formulario las líneas del ***listado 2.16*** y al mismo tiempo se llamará a la función ***Muestra_Posicion*** desde el inicio.

```
'Se ubicará en el primer registro al inicio de la aplicación
BindingContext(myData_view).Position = 0
Muestra_Posicion()
```

Listado 2.16 → Código para ubicarse en el primer registro

Lo siguiente será adicionar la funcionalidad de moverse a través de los registros a cada uno de los botones que se han agregado con anterioridad, para esto, se pueden observar los códigos del evento ***click*** de cada uno de los botones del ***listado 2.17***.

```
Private Sub btAnterior_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btAnterior.Click
      BindingContext(myData_view).Position -= 1
      Muestra_Posicion()
End Sub

Private Sub btSiguiente_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btSiguiente.Click
      BindingContext(myData_view).Position += 1
      Muestra_Posicion()
End Sub
```

```
Private Sub btInicio_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btInicio.Click
      BindingContext(myData_view).Position = 0
      Muestra_Posicion()
End Sub

Private Sub btFinal_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btFinal.Click
      BindingContext(myData_view).Position = _
      BindingContext(myData_view).Count - 1
      Muestra_Posicion()
End Sub
```

Listado 2.17 → Llamadas al evento click de los botones de desplazamiento

Ahora es posible compilar la aplicación (presionando **F5**) y probar los resultados; de esta forma es posible desplazarse a través de los registros de una manera sencilla y a prueba de errores.

2.12.4 Edición de Registros (Adición, Edición y Eliminación)

Para poder realizar cambios en los registros del *DataSet* se agregarán los elementos listados en la ***tabla 2.21*** en nuestro formulario.

Objeto	Propiedad	Valor
GroupBox	Name Text	**GroupBoxEdicion** Edición
Button	Name Text	**btNuevo** Nuevo Reg.
Button	Name Text Enabled	**btAceptar** Aceptar False
Button	Name Text	**btBorrar** Borrar
Button	Name Text	**btEditar** Editar

Tabla 2.21 → Formulario para la edición de registros

Dentro del *GroupBoxEdicion* se agregarán los botones; el botón *btNuevo* nos servirá para poder agregar nuevos registros a nuestro *DataSet*, el botón *btAceptar* nos servirá para poder admitir los cambios realizados en el registro cuando se agregue uno nuevo o se este editando uno existente, el botón n *btBorrar* tendrá la función de eliminar el registro en el que se esté posicionado, y por último, el botón *btEditar* pondrá al registro actual en modo de edición, y por lo tanto, se podrá

editar y cambiar sus elementos individualmente. Una vez agregados estos elementos se tendrá un formulario muy parecido al de la ***figura 2.47:***

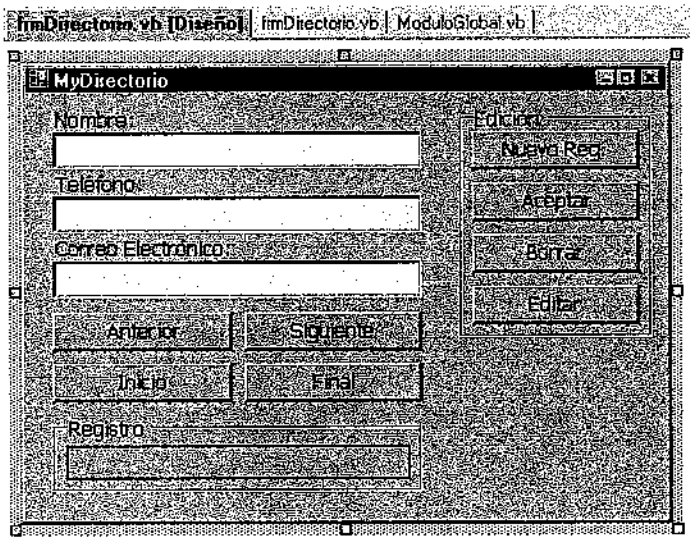


Figura 2.47 → formulario frmDIrectorio

Lo primero que se hará será crear dos procedimientos que nos permitan deshabilitar y habilitar los botones que no se requieran mientras se realice cierta acción y permanecerán en este estado hasta que el usuario confirme que ha terminado de agregar un nuevo registro o editado otro. Para dicho fin se agregará el código del ***listado 2.18:***

```
Private Sub Deshabilita_Botones_Edicion()
        'Con esta función se deshabilitan
        'los botones para evitar los cambios
        'mientras termina la edición del
        'registro actual
        btAceptar.Enabled = True
        btNuevo.Enabled = False
        btAnterior.Enabled = False
        btSiguiente.Enabled = False
        btInicio.Enabled = False
        btFinal.Enabled = False
        btBorrar.Enabled = False
        btEditar.Enabled = False
End Sub
```

```
Private Sub Habilita_Botones_Edicion()
        'Con esta función se habilitan
        'los botones que se deshabilitaron
        'mientras se realizaban cambios de
        'edición en algún registro
        btAceptar.Enabled = False
        btNuevo.Enabled = True
        btAnterior.Enabled = True
        btSiguiente.Enabled = True
        btInicio.Enabled = True
        btFinal.Enabled = True
        btBorrar.Enabled = True
        btEditar.Enabled = True
End Sub
```

Listado 2.18 → Procedimientos para habilitar y deshabilitar los botones de edición

El procedimiento al evento click del botón *btNuevo* se encuentra en el ***listado 2.19***, dicho procedimiento permite responder de manera eficaz cuando el usuario desee agregar un nuevo registro; lo primero será pedir el nombre del nuevo contacto del directorio; el tener conocimiento del valor de este registro antes de insertarlo al *DataSet* es primordial si se desea evitar ingresar un registro ya existente dentro de la tabla. De esta manera, se puede controlar un posible error dentro de la aplicación, este método es útil para un usuario experto, pero si se estuviera afrontando un usuario principiante, se debería usar una caja de diálogo propiamente programada para este fin en vez de usar un *Inputbox*, pero se puede ver primero el código para observar de que se está hablando.

```
Private Sub btNuevo_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs)
        'Obtenemos el nombre completo del nuevo contacto a _
        'través de un InputBox
        Dim strNombre As String = InputBox _
         ("Inserte el Nombre Completo del Contacto", _
          "Nuevo Registro", "nuevo_registro")
        'La función TRIM sirve para quitar los espacios en blanco
        'de los costados de la cadena que hemos obtenido
        strNombre = strNombre.Trim
        If Len(strNombre) = 0 Then
            MsgBox("Debe Insertar el Nombre del Contacto", _
            MsgBoxStyle.OKOnly, "Error")
        Else
            Try
                'El DataRowView nos representa una
                'vista personalizada de un DataRow, que
                'en este caso nos servirá como
                'una nueva fila de nuestro DataView.
                Dim NuevaFilaContactos As DataRowView
```

```
                'Creamos el nuevo registro
                NuevaFilaContactos = myData_view.AddNew
                'Comenzamos la edición
                NuevaFilaContactos.BeginEdit()
                'Insertamos el nombre del nuevo contacto
                NuevaFilaContactos("nombre") = strNombre
                'Terminamos de editar
                NuevaFilaContactos.EndEdit()
                'En este momento el nuevo registro
                'ya existe dentro de la tabla
                '"directorio" del DataView, así que
                'tendremos que buscar
                'el registro y posicionarnos en él.
                Dim Numero_Fila As Integer = _
                myData_view.Find(strNombre)
                'Nos ubicamos en el registro que creamos
                'anteriormente
                BindingContext(myData_view).Position = Numero_Fila
                'Ponemos a foco la caja de texto del teléfono
                txtTelefono.Focus()
                'Mostramos la posición del registro y
                'el contador actualizado
                Muestra_Posicion()
                'Con esta función, deshabilitaremos
                'los botones de desplazamiento
                'y otros hasta que el usuario presione
                'el botón btAceptar para
                'que el programa sepa que ha terminado
                'de editar los datos del
                'nuevo contacto.
                Call Deshabilita_Botones_Edicion()
            Catch ex As Exception
                MessageBox.Show(ex.Message)
                'El programa deberá poseer un
                'proceso especial para evitar
                'la duplicidad, en caso contrario,
                'al insertar un nombre
                'que ya este ocupado por un registro
                'se provocará un error
                'y la aplicación cerrará.
                End
            End Try
        End If
    End Sub
```

Listado 2.19 → Agregar un nuevo registro

Como se ha visto, este procedimiento resulta simple pero eficiente hasta cierto punto; se ha utilizado el *DataView* como medio para ingresar el nuevo registro, esto es porque se están utilizando las cajas de texto para que el usuario ingrese los valores de los diferentes campos de la tabla. Se podría insertar los registros

directamente al *DataSet*, pero como se han vinculado las cajas de texto a las columnas de la tabla, estas sirven como un conducto para poder realizar cambios directamente sobre los registros y poder visualizarlos de manera directa.

Evidentemente es necesario un procedimiento especial para poder detectar si ya existe un registro con el mismo nombre, pero al tener conocimiento de como poder buscar un registro existente, este pues no será un impedimento para poder realizar un formulario más experto para el ingreso de nuevos registros.

Para poder registrar los cambios al registro del nuevo contacto, se ha agregado el botón *btAceptar*, con el cual el usuario confirmará que ha terminado de realizar los cambios sobre el mismo, y el procedimiento para este botón también servirá para poder aceptar los cambios cuando edite a otro registro. El código para el evento *click* de este botón se muestra en el ***listado 2.20***.

```
    Private Sub btAceptar_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btAceptar.Click
        'En este procedimiento aceptamos los cambios _
        'de ediciones realizados en el registro y los _
        'actualizamos en el origen de los datos, que _
        'en este caso es el DataView.
        '
        'La clase Binding representa el enlace sencillo _
        'entre el valor de propiedad de un objeto y el _
        'valor de propiedad de un control.
        '
        'La clase BindingManagerBase administra todos _
        'los objetos Binding enlazados al mismo origen _
        'y miembro de datos.
        '
        'La clase BindingContext administra la colección _
        'de objetos BindingManagerBase para cualquier _
        'objeto que herede de la clase Control, como por _
        'ejemplo una Caja de Texto.
        '
        'BindingManagerBase permite la sincronización de _
        'controles con enlace a datos de un formulario _
        'Windows Forms que estén enlazados al mismo _
        'origen de datos, por ejemplo, un DataView.
        Dim Cambios_Realizados As BindingManagerBase = _
        BindingContext(myData_view)
        'El siguiente elemento representa el registro _
        'actual al que le realizamos los cambios
        Dim Fila_Cambios As DataRowView
        'El objeto Current contiene el valor del _
        'elemento actual en el origen de datos.
        Fila_Cambios = Cambios_Realizados.Current
```

```
        'Comenzamos y terminamos la edición; por _
        'supuesto podemos realizar procedimientos _
        'intermedios para controlar de manera mas _
        'eficaz las ediciones.
        Fila_Cambios.Row.BeginEdit()
        Fila_Cambios.Row.EndEdit()
        'Habilitamos la caja de texto txtNombre _
        'ya que así evitamos que cambie el nombre _
        'y provoque un error en el programa _
        'por duplicidad
        txtNombre.Enabled = True
        'Habilitamos los botones después de _
        'la edición.
        Call Habilita_Botones_Edicion()
    End Sub
```

Listado 2.20 → Procedimiento para el evento *click* del botón *btAceptar*

Al terminar de actualizar los cambios en el origen, en este caso el *DataView*, se llama al procedimiento *Habilita_Botones_Edición* para poder seguir agregando registros o editar los ya existentes.

Para poder editar los registros lo primero que el usuario debe de hacer es ubicarse en el registro en el que desea realizar cambios, posteriormente presionar el botón editar, el cual procederá a deshabilitar la caja de texto *txtNombre* para que el usuario no pueda realizar cambios sobre este dato, ya que si se ingresara un nombre ya existente, se provocaría un error de duplicidad en la columna llave de la tabla. Por lo tanto, como se ha mencionado, el procedimiento *Aceptar* servirá para aceptar los cambios cuando se edite un registro; al final de dicho procedimiento se puede observar que se volverá a habilitar la caja de texto *txtNombre*. El código para el evento *click* del botón *btEditar* se encuentra en el ***listado 2.21***.

```
Private Sub btEditar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btEditar.Click
        'Deshabilitamos los botones de navegación para _
        'poder editar sin problemas, eso hasta que el _
        'usuario acepte los cambios realizados.
        Call Deshabilita_Botones_Edicion()
```

```
        'Deshabilitamos la caja de texto del nombre _
        'para que no provoque un error en la _
        'duplicidad; se deve recordar que hemos _
        'tomado el registro del nombre como _
        'llave principal y por lo tanto no deben _
        'existir registros duplicados para este _
        'campo en específico; si lo que el usuario _
        'desea es cambiar el nombre, entonces se deberá _
        'crear un procedimiento especial para dicho _
        'propósito.
        txtNombre.Enabled = False
        'Ubicamos el foco de la ventana en la caja _
        'de texto del teléfono.
        txtTelefono.Focus()
    End Sub
```

Listado 2.21 → Procedimiento para el evento *click* del botón *btEditar*

Por último se verá el procedimiento de eliminación de registros, y para ello se agregará el código del ***listado 2.22*** al evento click del botón *btBorrar*.

```
Private Sub btBorrar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btBorrar.Click
        'Debemos estar completamente seguros que el usuario desea _
        'borrar el registro actual, por lo tanto, le preguntaremos _
        'dos veces para estar seguros.
        Dim Resultado As DialogResult
        Resultado = MessageBox.Show _
        ("¿Desea Borrar el Registro Actual?", _
        "Borrando", MessageBoxButtons.YesNo, MessageBoxIcon.Question, _
        MessageBoxDefaultButton.Button2)
        If Resultado = DialogResult.No Then
            Exit Sub
        ElseIf Resultado = DialogResult.Yes Then
            Resultado = MessageBox.Show( _
            "¿Desea en Verdad Borrar el Registro Actual?", _
            "Borrando", MessageBoxButtons.YesNo, _
            MessageBoxIcon.Question, _
            MessageBoxDefaultButton.Button2)
            If Resultado = DialogResult.No Then
                Exit Sub
            ElseIf Resultado = DialogResult.Yes Then
                'Ahora estamos seguros que en realidad desea
                'borrar el registro, así que simplemente nos ubicaremos
                'en el registro actual y lo borraremos.
                Dim Borrador As BindingManagerBase = _
                BindingContext(myData_view)
```

```
                'Fila_Borrador representará al registro en el que
                'estamos posicionados actualmente.
                Dim Fila_Borrador As DataRowView
                Fila_Borrador = Borrador.Current
                'Lo borramos directamente del DataView y el
                'cambio se reflejará _
                'inmediatamente en el DataSet.
                Fila_Borrador.Row.Delete()
                'Mostramos la posición del registro
                'y el contador actualizado.
                Muestra_Posicion()
            End If
        End If
End Sub
```

Listado 2.22 → Código para el evento *click* del botón *btBorrar*

Para estar completamente seguros que el usuario va a borrar el registro, se utilizan dos cajas de diálogo para poder salir del procedimiento si el usuario desea cancelar la operación. Una vez borrado el registro, no se debe olvidar actualizar el contador de registros para mostrar que el registro efectivamente fue eliminado.

2.12.5 Validación de Datos con el Control ErrorProvider

La validación de datos se ha vuelto muy popular entre los formularios de Internet; para este ejemplo se verá como se puede integrar a los formularios el control ***ErrorProvider.***

El componente *ErrorProvider* de formularios Windows Forms permite indicar al usuario, de forma no intrusiva, que algo va mal; habitualmente, se utiliza junto con la validación de entrada del usuario en un formulario o con la presentación de errores dentro de un conjunto de datos.

Para agregar este control es posible dirigirse al *Cuadro de Herramientas* de Visual Studio y se añadirá este control al formulario, como se indica en la ***figura 2.48.***

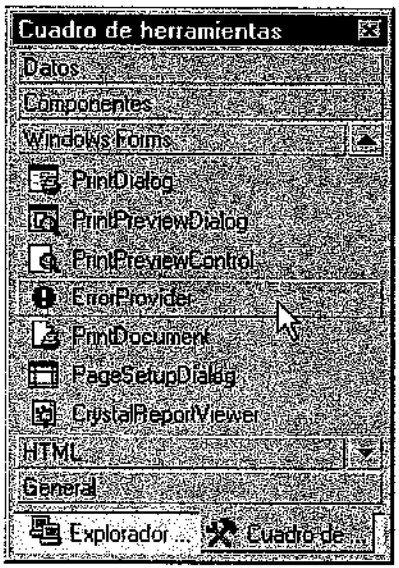


Figura 2.48 → Control ErrorProvider

El control *ErrorProvider* es un componente invisible, este tipo de controles existían desde la versión anterior de Visual Basic; un control invisible proporciona funcionalidad a la aplicación, pero a diferencia de otros controles, los componentes no proporcionan una interfaz de usuario y, por lo tanto, no es necesario mostrarlos en la superficie del diseñador de *Windows Forms* ya que no serán visibles en tiempo de ejecución.

En Visual Basic 6 existían muchos de estos controles, el más popular era el control ***Timer***, que proporcionaba la funcionalidad de un temporizador; al agregarlo al formulario, aparecía un relojito dentro del formulario, como se muestra en la ***figura 2.49;*** sin embargo, cuando se ejecutaba la aplicación, dicho control era invisible.

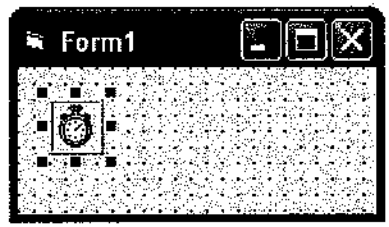


Figura 2.49 → Control *Timer* en un formulario de Visual Basic 6

Cuando se agrega un control invisible a un formulario en plataforma .Net, el diseñador de Windows Forms muestra una bandeja de tamaño variable en la parte inferior del formulario, donde se ven todos los componentes de este tipo, como se muestra en la ***figura 2.50.***

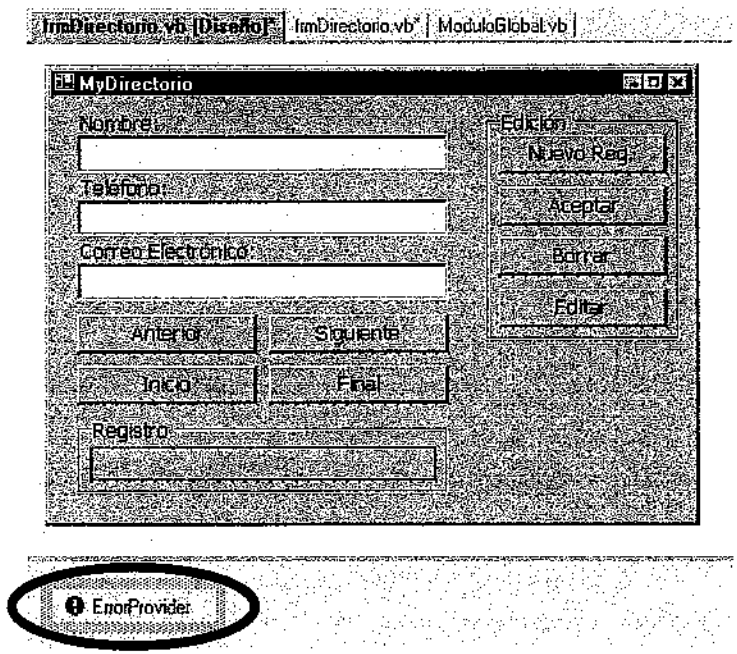


Figura 2.50 → Control ErrorProvider agregado a nuestro formulario frmDirectorio

Una vez que se agrega un control a la bandeja de componentes, puede ser seleccionado y establecer sus propiedades igual que con cualquier otro control en el formulario. Para el ejemplo no se cambiará ninguna propiedad al control excepto el nombre, el cual será simplemente ***ErrorProvider.***

La programación para este control resulta sumamente sencilla; simplemente se esperan dos eventos que suceden dentro de controles como las cajas de texto de Windows Forms, las cuales recibirán las entradas a validar. Estos dos eventos suceden durante y después de la validación de datos, el evento ***Validating*** y el evento ***Validated***. El primer evento sucede mientras el control esta siendo validado; mientras el segundo sucede cuando finaliza la validación; dentro de estos eventos se utilizan *expresiones regulares* para validar datos especificados por el usuario.

Las *expresiones regulares* proporcionan un método eficaz y flexible para procesar texto. La notación extensiva de búsqueda de patrones coincidentes de las expresiones regulares permite analizar con rapidez grandes cantidades de texto para buscar patrones de caracteres específicos para extraer, modificar, reemplazar o eliminar subcadenas de texto o para agregar las cadenas extraídas a una colección con el fin de generar un informe, que en este caso, es simplemente saber que contiene o que no contiene la cadena introducida por el usuario.

Para el ejemplo, simplemente se validará que la dirección de correo electrónico sea correcta; para este propósito se agregará el código del ***listado 2.23*** al evento ***Validating*** del control ***txtEmail.***

```
Private Sub txtEmail_Validating(ByVal sender As Object, ByVal e As
System.ComponentModel.CancelEventArgs) Handles txtEmail.Validating
        'El evento Validating Se produce cuando el
        'control se está validando.
        'El parámetro de entrada "sender" representa
        'el origen del evento en si,
        'que en este caso es un control Caja de Texto,
        'El tipo de dato "Control" define la clase
        'base para los controles, que son
        'componentes con una representación visual.
        Dim ctlEmisor As Control
        'La función CType devuelve el resultado
        'de convertir explícitamente una expresión
        'a un tipo de datos, objeto, estructura,
        'clase o interfaz.
        '"sender" representa el objeto origen
        'y con el parámetro Control especificamos
        'que es un "Control". De tal forma
        'que ctlEmisor es ahora una Caja de Texto
        ctlEmisor = CType(sender, Control)
```

```
'De esta forma nos podremos referir a este control
'como ctlEmisor, y de este modo podremos controlar
'o validar varios controles con un solo procedimiento.
'
'Trim devuelve la cadena sin espacios en blanco
'al inicio y al final, se puede usar
'RTrim o LTrim; Trim es la combinación de ambas
ctlEmisor.Text = ctlEmisor.Text.Trim
'"e" contiene los datos del evento (Validating),
'si "e" se establece en true en el delegado del
'evento Validating, se suprimen todos los eventos
'que normalmente se producirían después del evento
'Validating; lo que significa que si el control
'no tiene texto, asignaremos a "e" el valor false
'y por lo tanto suprimimos los eventos validating
'y validated, de tal forma que no se validará
'para este caso.
If ctlEmisor.Text = "" Then
    e.Cancel = False
    Exit Sub
End If
'La función Regex.IsMatch indica si la expresión
'regular busca una coincidencia en la cadena de
'entrada devolviendo true si es encontrada.
'Para este caso revisamos que el contenido
'NO contenga espacios en blanco.
If System.Text.RegularExpressions.Regex.IsMatch _
(ctlEmisor.Text, " ") Then
    'En caso de encontrar la coincidencia,
    'habilitamos el error hasta que sea
    'corregido y se produzca el evento
    'validated cuando finalice la validación
    'del control.
    e.Cancel = True
    'SetError establece la cadena de
    'descripción del error para el
    'control especificado.
    ErrorProvider.SetError(ctlEmisor, _
    "El Formato de este campo es Incorrecto." _
    & ControlChars.CrLf & _
    "NO debe contener ESPACIOS EN BLANCO.")
End If
'Revisamos que el formato en este campo
'contenga al menos un punto (.)
If System.Text.RegularExpressions.Regex.IsMatch _
(ctlEmisor.Text, "[.]") = False Then
    e.Cancel = True
    ErrorProvider.SetError(ctlEmisor, _
    "El Formato de este campo es Incorrecto." _
    & ControlChars.CrLf & _
    "DEBE contener un PUNTO -> .")
End If
```

```
        'Revisamos que el formato en este campo
        'contenga al menos una arroba @
        If System.Text.RegularExpressions.Regex.IsMatch _
        (ctlEmisor.Text, "[@]") = False Then
            e.Cancel = True
            ErrorProvider.SetError(ctlEmisor, _
            "El Formato de este campo es Incorrecto." _
            & ControlChars.CrLf & _
            "DEBE contener el carácter ARROBA -> @")
        End If
    End Sub
```

Listado 2.23 → Código para el evento Validating de txtEmail

A continuación se agregará el código del ***listado 2.24*** al evento ***Validated*** del control ***txtEmail*** para finalizar la programación de este módulo.

```
Private Sub txtEmail_Validated(ByVal sender As Object, ByVal e As
System.EventArgs) Handles txtEmail.Validated
        'El evento validated se produce
        'cuando finaliza la validación
        'del control; por lo tanto
        'mandaremos al ErrorProvider
        'una cadena vacía para que
        'desaparezca, es decir, si la
        'longitud de la cadena que se
        'envía como parámetro a SetError
        'es mayor que cero, se muestra
        'el icono de error y la
        'información en forma de
        'descripción del error.
        'Si la longitud de esta cadena
        'es cero, el icono del error
        'se oculta.
        Dim ctlEmisor As Control
        ctlEmisor = CType(sender, Control)
        ErrorProvider.SetError(ctlEmisor, "")
    End Sub
```

Listado 2.24 → Código para el evento Validated de txtEmail

Ahora se puede compilar la aplicación y ver los resultados; cabe hacer notar que cuando aparece el icono de error del ***ErrorProvider*** al costado de ***txtEmail*** se puede pasar el cursor encima de él para poder ver la descripción del error, y al mismo tiempo se deshabilitarán las funciones de los botones habilitados hasta que el error sea corregido; para apreciar el resultado se puede ver la ***figura 2.51.***

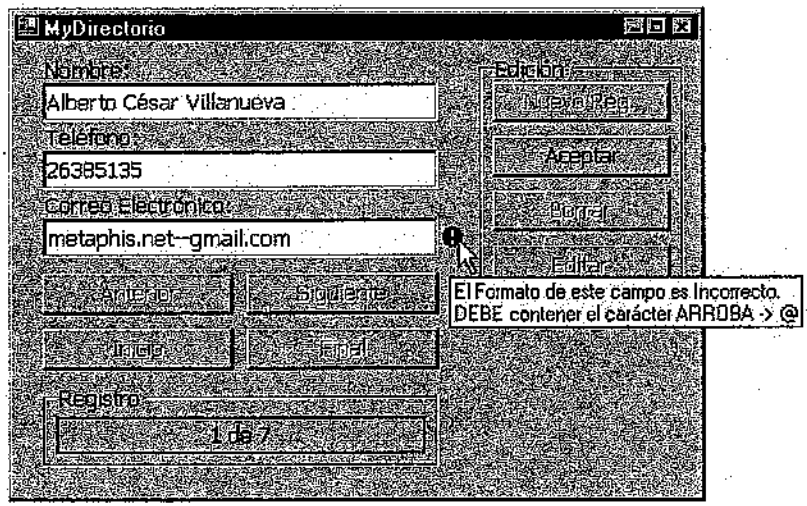


Figura 2.51 → Control ErrorProvider en funcionamiento

La validación es simplemente una serie de eventos que tienen lugar durante el procesamiento de la entrada del usuario; para validar el contenido de un control, se ha escrito el código que controla al evento ***Validating***. Este controlador de eventos debe probar una condición determinada, que para este ejemplo es que la cadena contenga una arroba, un punto y que no contenga espacios en blanco. Si la prueba devuelve error, se define la propiedad ***Cancel*** del ***CancelEventArgs*** del evento ***Validating*** como *True;* esto cancela el evento ***Validating*** y devuelve el foco al control. El efecto práctico es que el usuario no podrá dejar el control hasta que los datos sean válidos, por supuesto, corrigiendo el error.

2.12.6 Correos Electrónicos a través de Outlook Object Library

Outlook ofrece una serie de objetos para gestionar el correo a través de lenguajes como *Visual Basic .Net*. En este apartado se verá como es posible enviar correos electrónicos desde las aplicaciones desarrolladas bajo esta plataforma.

Outlook se compone de una biblioteca de objetos que se utilizan para tareas específicas dentro de su propio entorno. Entre estos objetos se establece una relación jerárquica encabezada por el objeto principal ***Application***. De este objeto dependen todos los demás, por este motivo será condición imprescindible haber creado el objeto *Application* para tener acceso al resto.

Los principales componentes del modelo de objetos de *Outlook* son los siguientes:

- ***Application:*** Es el objeto superior de la jerarquía que representa la aplicación completa. Con él es posible hacer referencias a otros objetos de la aplicación y crear nuevos elementos y objetos.

- ***NameSpace:*** Representa el almacén de mensajes ***MAPI (Interfaz de Programación de Aplicaciones de Mensajería)*** donde se almacenan todos los elementos de *Outlook*. Proporciona métodos para iniciar y cerrar la sesión en Outlook y para hacer referencia a las carpetas predeterminadas, como la Bandeja de Entrada, Bandeja de Salida, Contactos, etc.

- ***Explorer:*** Representa la ventana de *Outlook*. Permite mostrar, volver y cerrar la ventana activa.

- ***Folders:*** Hay dos objetos de carpetas, el objeto colección *Folders* permite trabajar con colecciones de carpetas y el objeto *MAPIFolder* sirve para trabajar con una carpeta única.

- ***Inspector:*** Hace referencia a formularios. Se usa para mostrar formularios y páginas.

- ***AddressEntry:*** Cada objeto *AddressEntry* de la colección *AddressEntries* mantiene información que representa a una persona o proceso al cual el sistema de mensajería puede entregar mensajes.

- ***AddressList:*** Representa la librería de direcciones que contiene un conjunto de objetos *AddressEntry*. Es posible acceder a la jerarquía completa mediante la colección primaria *AddressLists*.

- ***Exception:*** Mantiene información sobre la instancia o acumulación de un objeto *AppontmentItem*. A diferencia de los demás objetos de *Outlook*, el objeto *Exception* es un objeto de sólo lectura.

- ***Control:*** El objetos *Controls* permite trabajar con todos los controles de una página, y el objeto de control específico.

Los elementos de *Outlook* son tan importantes como los objetos, pues permiten acceder a los correos electrónicos, entre otras cosas. Hay dos objetos, el objeto de colección *Items*, que permite trabajar con un elemento dentro de una carpeta, y los objetos que representan los tipos de elementos estándar de *Outlook*, como por ejemplo *MilItem*, que representa un mensaje de correo.

Para continuar con el ejemplo, se debe adicionar un nuevo botón para enviar un correo electrónico al registro en el que la aplicación se encuentre posicionado actualmente; para ello se debe adicionar al formulario los controles descritos en la ***tabla 2.22:***

Objeto	Propiedad	Valor
GroupBox	Name Text	**GroupBoxNuevoEmail** E - Mail
Button	Name Text	**btNuevoEmail** Nuevo E - Mail.

Tabla 2.22 → Botón para crear un nuevo correo electrónico

Los controles listados anteriormente deberán tener una apariencia parecida a la que se muestra en la ***figura 2.52:***

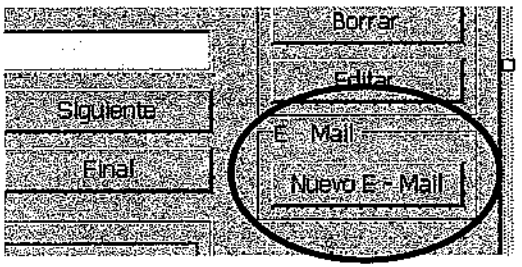


Figura 2.52 → Controles para crear un nuevo correo electrónico

El botón que se ha agregado simplemente servirá para abrir un nuevo formulario que mostrará como destinatario del correo electrónico dirigido al contacto del directorio donde se encuentre posicionado actualmente. Para ello se debe agregar un nuevo formulario, posteriormente se accede al panel del *Explorador de Soluciones* y se da click derecho sobre la raíz del proyecto para agregar un nuevo formulario, como se muestra en la ***figura 2.53:***

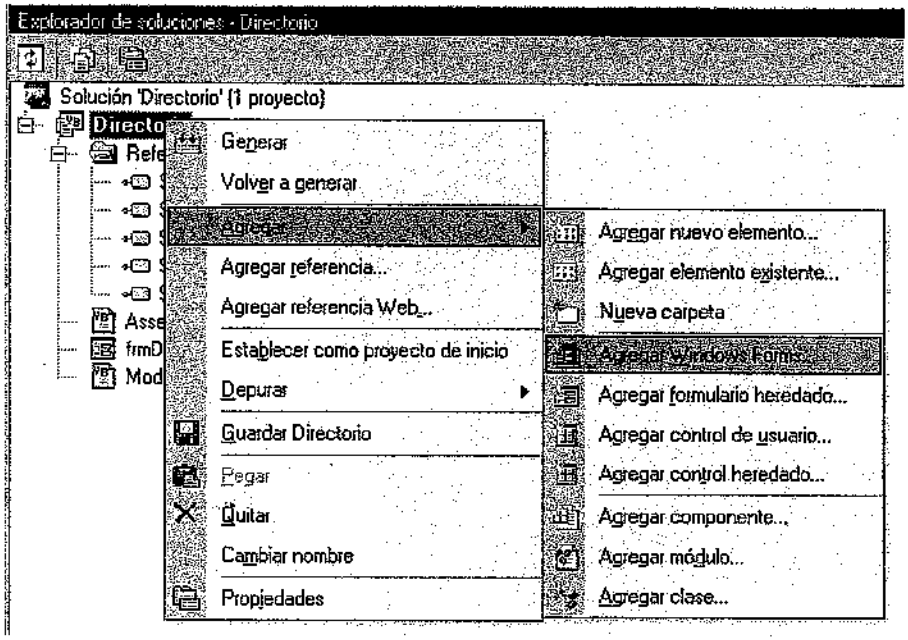


Figura 2.53 → Adición de un nuevo formulario

Posteriormente se nombrará al formulario ***frmEmail*** como se muestra en la ***figura 2.54:***

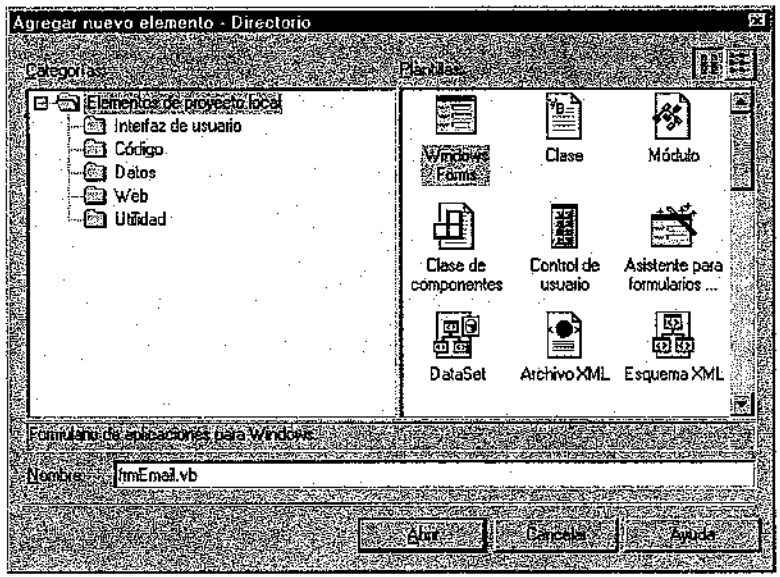


Figura 2.54 → Nombramiento del formulario *frmEmail*

Después de haber agregado el formulario al proyecto, se cambiará la propiedad ***text*** del formulario para que tome el valor ***"Nuevo Correo Electrónico"***; ahora se agregarán los controles listados en la ***tabla 2.23*** al formulario *frmEmail:*

Objeto	Propiedad	Valor
GroupBox	Name Text	**GroupBoxDestinatario** Destinatario:
GroupBox	Name Text	**GroupBoxAsunto** Asunto:
GroupBox	Name Text	**GroupBoxMensaje** Mensaje:
GroupBox	Name Text	**GroupBoxAdjuntos** Archivos Adjuntos:
Label	Name Text TextAlign BorderStyle	**LabelDestinatario** (Vacío) MiddleCenter FixedSingle
TextBox	Name Text	**txtAsunto** (Vacío)

Objeto	Propiedad	Valor
TextBox	Name Text Multiline ScrollBars	**txtMensaje** (Vacío) true Both
ListBox	Name SelectionMode	**ListBoxAdjuntos** One
Button	Name Text	**btMas** +
Button	Name Text	**btMenos** -
Button	Name Text	**btEnviar** Enviar

Tabla 2.23 → Controles para el formulario *frmEmail*

Una vez agregados estos controles el formulario *frmEmail* deberá tener una apariencia como la que se muestra en la ***figura 2.55:***

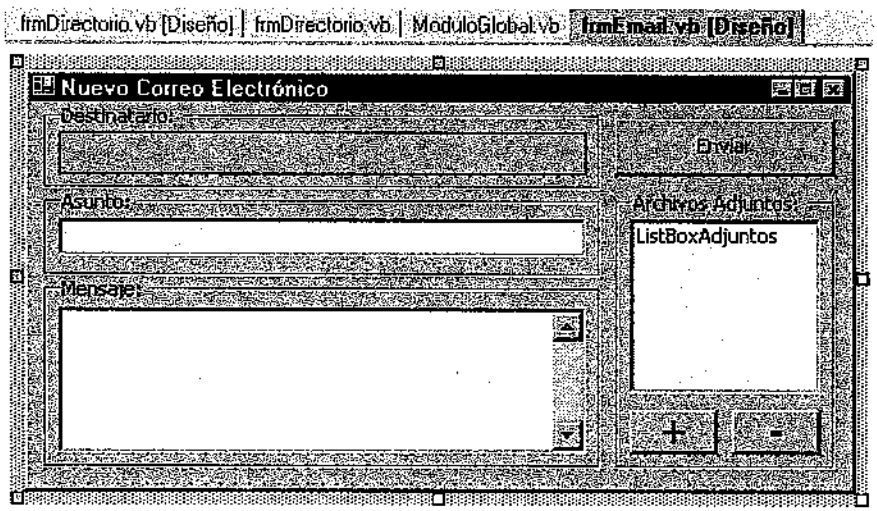


Figura 2.55 → Aspecto del formulario *frmEmail*

Para mostrar el formulario *frmEmail* como una caja de diálogo se debe crear un objeto de tipo *frmEmail* y acceder al método ***ShowDialog*** del objeto *Form* de *Windows Forms*. Para ello se debe agregar el código del ***listado 2.25*** al evento *click* del botón *btNuevoEmail* del formulario *frmDirectorio*.

```
Private Sub btNuevoEmail_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btNuevoEmail.Click
        If Len(txtEmail.Text) <= 0 Then
            'Es preciso que el contacto posea un correo electrónico
            MessageBox.Show("El contacto actual NO posee correo
electrónico", _
            "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
        Else
            'Se crea una nueva instancia de tipo frmEmail
            Dim myFrmEmail As New frmEmail
            'La Caja de Diálogo mostrará como destinatario
            'la dirección de correo del contacto actual
            myFrmEmail.LabelDestinatario.Text = txtEmail.Text
            'Se muestra el formulario en modo MODAL
            '
            'Modo MODAL se refiere al hecho de que la caja de
            'diálogo espera una respuesta del usuario, en este caso,
            'enviar un correo electrónico o cancelar el envío
            'cerrando la ventana.
            myFrmEmail.ShowDialog()
        End If
End Sub
```

Listado 2.25 → Procedimiento para el evento *click* del botón *btNuevoEmail* de *frmDirectorio*

Ahora se procederá a programar todos los procedimientos para que el formulario *frmEmail* funcione correctamente, pero antes de ello se debe agregar la referencia a la librería de *Outlook* para que se pueda crear una instancia a dicho objeto y acceder a sus elementos y procedimientos para poder enviar correos electrónicos.

Cuando se agrega una referencia a los proyectos de Visual Basic .Net se muestra una serie de paneles con fichas que contienen listas de varios tipos de componentes y proyectos que se pueden examinar. Para ver el cuadro de diálogo ***Agregar Referencia***, se selecciona *Agregar Referencia* en el menú *Proyecto* o se da click con el botón derecho del *Mouse* en el nodo *Referencias* del *Explorador de Soluciones*. A continuación, se selecciona la ficha del tipo de componente que desea examinar. El número de fichas disponibles en la parte superior del cuadro de diálogo *Agregar Referencia* varía en función del tipo de proyecto abierto y de los recursos que este utiliza.

Este cuadro de diálogo edita el archivo de referencias del proyecto seleccionado. Un archivo de referencias del proyecto contiene información sobre los requisitos en tiempo de ejecución de una aplicación o componente, el modo en que se van a registrar y el lugar del equipo del usuario en el que se deben instalar.

Las fichas contenidas en la caja de diálogo *Agregar Referencia* son:

- **.NET Framework:** Enumera todos los componentes de .NET Framework disponibles para hacer referencias.
- **COM:** Enumera todos los componentes COM disponibles para hacer referencias.
- **Proyectos:** Enumera todos los proyectos disponibles para hacer referencias.
- **Nombre de Componente:** Nombre completo o nombre descriptivo del componente.
- **Versión:** Número de versión que se muestra del componente.
- **Ruta de Acceso:** Nombre de archivo del componente y ruta de acceso en la que se encuentra.
- **Nombre de Proyecto:** (Ficha Proyecto solamente). Muestra los nombres de los proyectos a los que se hace referencia.
- **Directorio del proyecto:** (Ficha Proyecto solamente). Muestra la ubicación del directorio para proyectos a los que se hace referencia.
- **Examinar:** Muestra el cuadro de diálogo *Seleccionar componente*, que permite examinar archivos de los componentes adicionales para buscar los que no están mencionados en la ficha actual y agregarlos al campo de los que están seleccionados.
- **Seleccionar:** Agrega el componente seleccionado con el ratón a la lista *Componentes seleccionados*.
- **Componentes Seleccionados:** Muestra los componentes seleccionados que se van a agregar al ámbito de exploración del proyecto.
- **Tipo:** Tipo de componente que se muestra.
- **Origen:** Nombre de archivo del componente y ruta de acceso en la que se encuentra.
- **Quitar:** Quita una referencia de componente de la lista Componentes seleccionados. Esta acción no elimina del sistema el componente real.

Regresando al ejemplo, se debe agregar la referencia al objeto ***Microsoft Outlook 11.0 Object Library***, para ello, como se ha mencionado, se debe seleccionar la opción *Agregar Referencia* del menú *Proyecto* para que aparezca la ventana que se muestra en la ***figura 2.56:***

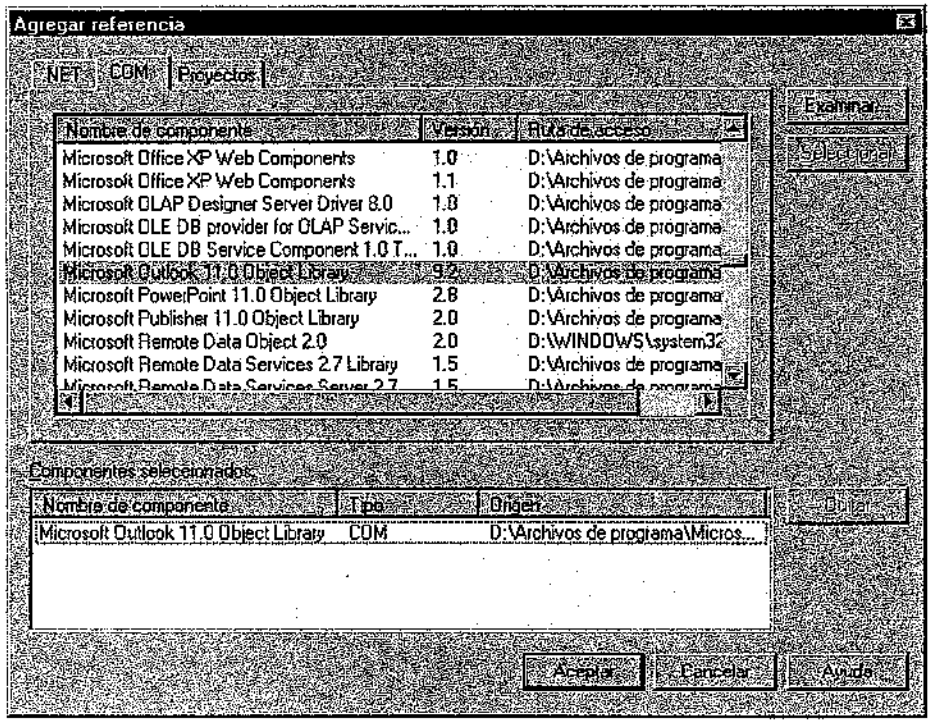


Figura 2.56 → Ventana para Agregar una Referencia al Proyecto

Es importante resaltar el hecho de que ***Microsoft Office 2000 Professional (con Outlook)*** o superior debe estar instalado en el sistema para poder agregar dicha referencia, ya que de lo contrario no se encontrará en el listado. También es necesario que el equipo donde sea instalada la aplicación posea la misma versión de *Office* o una superior para que el programa funcione correctamente.

Una vez seleccionada la referencia, se presiona el botón de *Seleccionar* y posteriormente *Aceptar*. Después se podrá verificar que efectivamente se haya agregado la referencia dentro del *Explorador de Soluciones* como se muestra en la ***figura 2.57.***

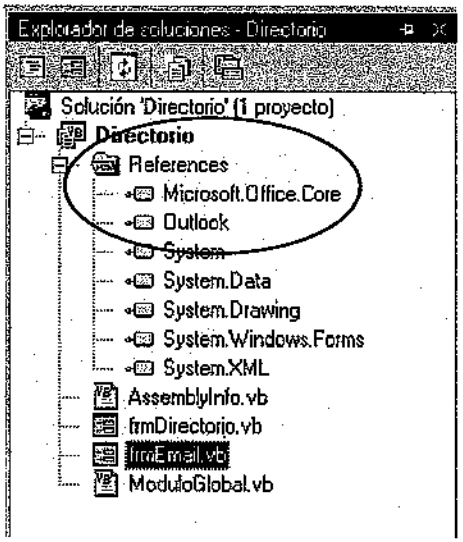


Figura 2.57 → Referencias agregadas al proyecto

Una vez agregada la referencia, lo primero a programar es la adición y eliminación de archivos adjuntos al e-mail, para ello el usuario podrá presionar el botón *btMas* para adicionar un archivo al correo, el cual aparecerá como un elemento de la lista *ListBoxAdjuntos* en forma de ruta, incluyendo el nombre con extensión de dicho archivo. Posteriormente podrá eliminar cualquier archivo agregado seleccionándolo en la lista y presionando el botón *btMenos*.

Para poder agregar un archivo al correo electrónico se debe usar un cuadro de diálogo prediseñado para dicho propósito, esta caja de diálogo se llama ***OpenFileDialog***, este control es un componente invisible parecido al *ErrorProvider* antes mencionado. Para agregar dicho elemento se selecciona el componente *OpenFileDialog* del *Cuadro de Herramientas* como se muestra en la ***figura 2.58.*** Las propiedades de este control se conservan integras excepto por su nombre, el cual será para este ejemplo *myOpenFileDialog.*

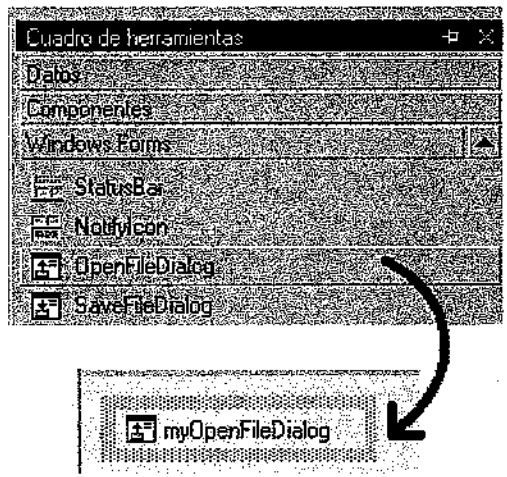


Figura 2.58 → Adición del componente invisible OpenFileDialog

Los procedimientos para los botones *btMas* y *btMenos* del formulario *frmEmail* se encuentran en el ***listado 2.26:***

```
Private Sub btMas_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btMas.Click
        'Cadena en donde se almacenará la ruta completa
        'del archivo a adjuntar, incluyendo el nombre
        'del archivo y su extensión
        Dim strFile As String
        'Una instrucción With permite múltiples referencias
        'a los miembros de una expresión sin tener que
        'especificar la expresión varias veces.
        'La expresión se evalúa una vez, cuando entra
        'en el bloque. En el bloque de instrucción With,
        'una expresión que empiece con un punto o una
        'exclamación se evalúa como si la expresión
        'With lo precediera.
        '
        'No es válido bifurcarse a un bloque de
        'instrucción With desde fuera del bloque.
        With myOpenFileDialog
            'Obtiene o establece el
            'título del cuadro de diálogo
            'de archivo.
            .Title = "Adjuntar Archivo"
```

```
            'Obtiene o establece la cadena
            'actual de filtro de nombres de
            'archivo, que determina las
            'opciones que aparecen en los
            'cuadros "Guardar como archivo
            'de tipo" o "Archivos de tipo"
            'del cuadro de diálogo.
            .Filter = "Todo Tipo (*.*)|*.*"
            'Obtiene o establece un valor que
            'indica si el cuadro de diálogo
            'agrega automáticamente una
            'extensión a un nombre de archivo
            'en caso de que el usuario omita
            'dicha extensión.
            .AddExtension = True
            'Obtiene o establece un valor que
            'indica si el cuadro de diálogo
            'muestra una advertencia cuando
            'el usuario especifica un nombre
            'de archivo que no existe.
            .CheckFileExists = True
            'Obtiene o establece un valor
            'que indica si el cuadro de diálogo
            'muestra una advertencia cuando el
            'usuario especifica una ruta de
            'acceso que no existe.
            .CheckPathExists = True
            'Obtiene o establece el directorio
            'inicial que muestra el cuadro de
            'diálogo de archivo.
            '
            'Para este ejemplo se ubicará el
            'directorio que sirve de
            'repositorio común para documentos.
            .InitialDirectory = Environment. _
            GetFolderPath(Environment. _
            SpecialFolder.Personal)
            'Obtiene o establece un valor que
            'indica si el cuadro de diálogo
            'permite seleccionar varios archivos.
            .Multiselect = False
            'Si la selección del archivo fué correcta
            If .ShowDialog() = DialogResult.OK Then
                'Se adiciona la ruta y el nombre
                'del archivo a la lista
                ListBoxAdjuntos.Items.Add(.FileName())
            End If
        End With
    End Sub

    Private Sub btMenos_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btMenos.Click
```

```
        'Si hay al menos un elemento seleccionado
        If ListBoxAdjuntos.SelectedItems.Count > 0 Then
            Try
                'Removemos de la lista el elemento
                'seleccionado
                ListBoxAdjuntos.Items. _
                Remove(ListBoxAdjuntos.SelectedItem)
            Catch ex As Exception
                'En caso de error, mostrar el
                'origen de la excepción
                MessageBox.Show(ex.Message)
            End Try
        End If
    End Sub
```

Listado 2.26 → Procedimientos para los botones *btMas* y *btMenos*

Para finalizar, el procedimiento más importante de esta sección cae sobre el evento click del botón *btEnviar* del formulario *frmEmail*. Este procedimiento encierra toda la gestión del correo, tanto su contenido textual como los archivos adjuntos. El formulario *frmEmail* contiene cajas de texto especiales para cada uno de los campos del correo como el asunto y mensaje del mismo.

Existen dos variables principales para enviar un correo electrónico a través de Outlook que se derivan a su vez del objeto ***Outlook Application***, la primera es ***MilItem*** que representa al correo electrónico, la segunda es ***Attachment***, la cual representa los archivos adjuntos al e-mail.

Los parámetros recibidos por la propiedad ***Attachment*** son:

- ***Origen:*** Especifica el archivo (representado por la ruta de acceso completa y el nombre del archivo) o el elemento que constituye el adjunto.
- ***Tipo:*** Especifica el tipo del adjunto. Puede tomar uno de tres valores de tipo *OlAttachmentType*: *olByReference*, *olByValue* u *olEmbeddedItem*.
- ***Posición:*** Especifica la posición del archivo adjunto dentro del mensaje.
- ***Nombre Completo:*** Especifica el nombre del archivo adjunto. Se omite, a menos que el valor de *Tipo* sea *olByValue*.

El parámetro Nombre Completo es una descripción del adjunto que reemplazará el nombre original del archivo dentro del mensaje, aunque intérnamente siga conservando su nombre original.

Respecto al tipo de archivo que se va a adjuntar, es posible seleccionar uno de los siguientes valores:

- ***OlByReference:*** Crea un acceso directo a un archivo externo.
- ***OlByValue:*** Incrusta un archivo en el elemento (en este caso el correo).
- ***OlEmbeddedItem:*** Crea un acceso directo a un elemento de Outlook.

Si se desea ver el correo que se va a enviar, se usará el método ***Display*** del objeto *Application*. Para enviar el mensaje se usará el método ***Send*** de ese mismo objeto. Ahora es posible adicionar el procedimiento para el evento click del botón *btEnviar* del formulario *frmEmail* desglosado en el ***listado 2.27:***

```
Private Sub btEnviar_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btEnviar.Click
        'Si no posee nada en el campo Asunto,
        'se muestra un aviso
        'de que falta dicho campo
        If Len(txtAsunto.Text) <= 0 Then
            MessageBox.Show( _
            "El correo electrónico no contiene un Asunto, favor de
ingresarlo antes de enviarlo", _
            "Falta campo Asunto", MessageBoxButtons.OK, _
            MessageBoxIcon.Exclamation)
            txtAsunto.Focus()
        Else
            'Se crea la instancia al objeto Application de Outlook
            Dim myOutlookApplication As New _
            Microsoft.Office.Interop.Outlook.Application
            'Se crea el elemento MailItem que representa al e-mail
            Dim myMail As Microsoft.Office.Interop.Outlook.MailItem
            'Se crea un arreglo de archivos adjuntos,
            'ya que no se sabe
            'si el usuario adjuntará uno o "n" correos.
            Dim myAttachment As _
            Microsoft.Office.Interop.Outlook.Attachments
            'Primero se crea el nuevo correo electrónico
            myMail = myOutlookApplication.CreateItem _
            (Microsoft.Office.Interop.Outlook.OlItemType.olMailItem)
            'Agregamos el destinatario
            myMail.To = Trim(LabelDestinatario.Text)
            'Agregamos el Asunto
            myMail.Subject = Trim(txtAsunto.Text)
            'Agregamos el mensaje, si existe
            If (Len(txtMensaje.Text) > 0) Then
                myMail.Body = Trim(txtMensaje.Text)
            End If
```

```
            'Se agregan los n archivos
            'adjuntos
            '
            'Si hay al menos un archivo dentro de la
            'lista de adjuntos, se agregan al correo
            Dim j As Integer
            If ListBoxAdjuntos.Items.Count > 0 Then
                myAttachment = myMail.Attachments
                For j = 0 To ListBoxAdjuntos.Items.Count - 1
                    myAttachment.Add _
                    (ListBoxAdjuntos.Items(j), _
                    Microsoft.Office.Interop.Outlook. _
                    OlAttachmentType.olByValue, j + 1)
                Next
            End If
            'Ahora es posible ver el correo electrónico
            'creado con las siguientes líneas
            '*****************************************
            'Try
            '    myMail.Display() .
            'Catch ex As Exception
            '    MessageBox.Show(ex.Message)
            'End Try
            '*****************************************
            'Por último es posible enviar directamente
            'el correo a la bandeja de salida de Outlook
            '*****************************************
            Try
                myMail.Send()
            Catch ex As Exception
                MessageBox.Show(ex.Message)
            End Try
            '*****************************************
        End If
    End Sub
```

Listado 2.27 → Procedimiento para el evento click del botón btEnviar

Para probar que se estén enviando los correos correctamente, es posible enviar ahora uno de prueba, como se muestra en la ***figura 2.59.*** Es importante verificar que efectivamente los datos ingresados en el formulario creado sean los que se estén enviando a la bandeja de salida de Outlook.

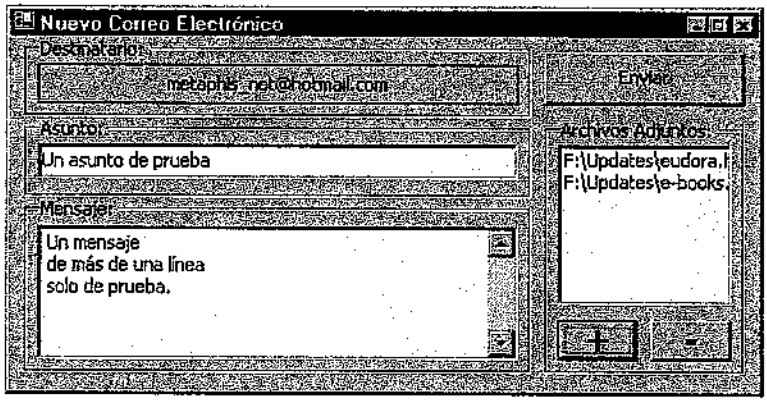


Figura 2.59 → Apariencia formulario *frmEmail* antes de enviar un correo electrónico

Cuando se presione el botón *btEnviar* del formulario se mostrará un mensaje como el de la ***figura 2.60,*** donde se pide una confirmación para poder enviar el correo electrónico.

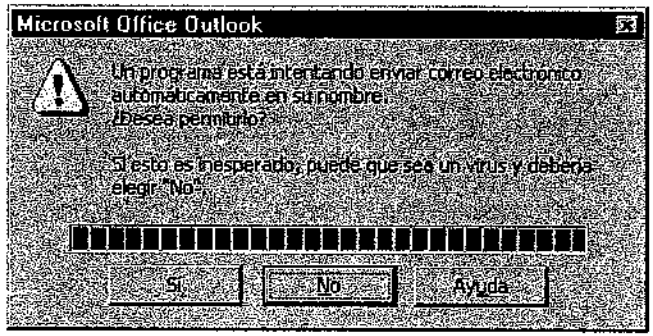


Figura 2.60 → Ventana de confirmación de envío de correo electrónico

Cuando se presione el botón "Sí" el correo electrónico pasará a la bandeja de salida de *Outlook* como se muestra en la ***figura 2.61:***

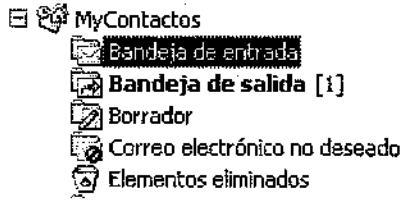


Figura 2.61 → Bandeja de Salida de *Outlook*

Por último es posible verificar los datos del correo que se ha enviado desde el formulario que se ha creado, también es importante resaltar el hecho de que se han enviado más de un archivo adjunto en el correo electrónico. Todo lo anterior se puede cotejar en la ***figura 2.62:***

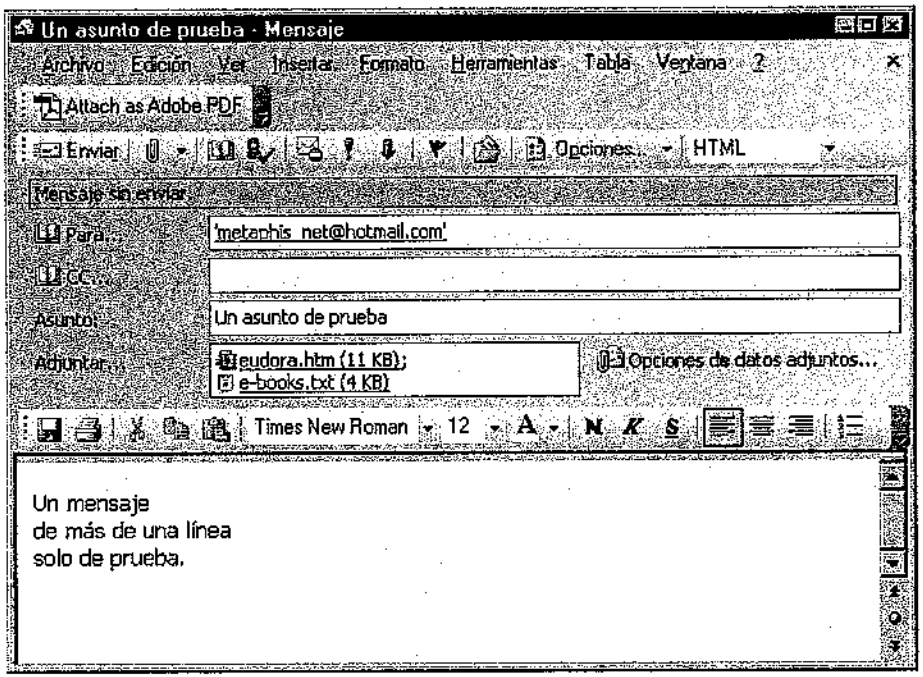


Figura 2.62 → Correo enviado en el editor de Microsoft Outlook 2003 Professional

Ahora es posible agregar esta funcionalidad para aplicaciones más complejas, y no sólo eso, si no que es posible gestionar cualquier tipo de objeto de *Outlook* a través de las aplicaciones ***Visual Basic .Net*** sin mayor dificultad.

Tal vez no haya diferencia en el hecho de enviar un solo correo, si no que la diferencia estriba en que es posible enviar una sucesión de ellos con las características que se deseen como un común denominador para todos y cada uno de ellos; estas características comunes como un solo asunto o un solo archivo adjunto extienden la funcionalidad de guardar correos electrónicos en los registros de las Bases de Datos de los Sistemas Empresariales.

2.12.7 Crear Relaciones entre dos Tablas con DataSet

En una base de datos relacional, las relaciones evitan la redundancia de datos. Por ejemplo, si se diseña una base de datos que realice un seguimiento de la información sobre libros, se podría disponer de una tabla, denominada títulos, que almacene toda la información sobre cada libro, por ejemplo, el título, la fecha de publicación y el editor. Puede que también se desee almacenar información sobre el editor, por ejemplo, el número de teléfono, la dirección y el correo electrónico. Si se deseara almacenar toda esta información en la tabla títulos, el número de teléfono del editor se duplicaría por cada uno de los títulos que publique.

La mejor solución es almacenar la información del editor una sola vez en una tabla editores independiente. Después, se podría situar un puntero en la tabla títulos que haga referencia a una entrada de la tabla "editores" como se muestra en la ***figura 2.63.***

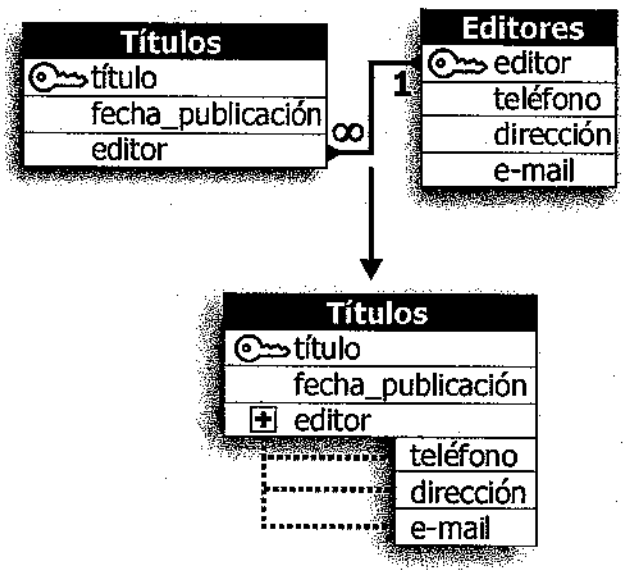


Figura 2.63 → Relaciones entre dos tablas de una base de datos

Para asegurarse de que los datos están sincronizados, se debe exigir integridad referencial entre las tablas títulos y editores. Las relaciones de integridad referencial garantizan que la información de una tabla coincida con la información de otra. Por ejemplo, cada título de la tabla títulos debe estar asociado con un editor específico de la tabla editores. No se puede agregar un título a un editor que no existe en la base de datos.

La integridad referencial es un sistema de reglas que garantiza que las relaciones entre las filas de tablas relacionadas son válidas y que no se eliminan ni modifican accidentalmente los registros relacionados.

Una relación hace coincidir los datos de las columnas clave (normalmente, se trata de columnas con el mismo nombre en ambas tablas). En la mayoría de los casos, la relación hace coincidir la clave principal de una tabla, que proporciona un identificador único para cada fila, con una entrada de la clave externa de la otra tabla.

Existen tres tipos de relaciones entre tablas. El tipo de relación que se crea depende de la definición de las columnas relacionadas.

1. **Relación uno a varios:** La relación uno a varios es el tipo de relación más común. En este tipo de relación, la fila de una tabla A puede coincidir con muchas filas de una tabla B, pero una fila de la tabla B sólo puede coincidir con una fila de la tabla A. Por ejemplo, las tablas editores y títulos poseen una relación uno a varios: cada editor publica muchos títulos, pero cada título sólo corresponde a un editor.

2. **Relación varios a varios:** En las relaciones varios a varios, la fila de una tabla A puede coincidir con muchas filas de una tabla B y viceversa. Para crear una relación de este tipo hay que definir una tercera tabla, denominada *"tabla de unión"*, cuya clave principal se compone de las claves externas de la tabla A y de la tabla B.

3. **Relación uno a uno:** En las relaciones uno a uno, la fila de una tabla A sólo puede coincidir con una fila de una tabla B y viceversa. Se crea una relación uno a uno si las dos columnas relacionadas son claves principales o tienen restricciones únicas. Este tipo de relación no es habitual debido a que la mayoría de la información relacionada de esta forma estaría en una sola tabla. Se puede utilizar una relación uno a uno para:

    - Dividir una tabla con muchas columnas.
    - Aislar parte de una tabla por razones de seguridad.
    - Almacenar datos con una vida muy corta que podrían eliminarse fácilmente con sólo eliminar la tabla.
    - Almacenar información que sólo se aplica a un subconjunto de la tabla principal.

Un conjunto de datos dentro de *ADO.Net* puede contener varias tablas, incluidas las que tengan relaciones implícitas entre ellas, con campos de clave en común. Para aprovechar estas relaciones implícitas, se puede utilizar un objeto ***DataRelation*** de *ADO.Net.*

En muchas ocasiones las aplicaciones tienen que trabajar con tablas relacionadas. Aunque un conjunto de datos contiene tablas y columnas al igual que una base de datos, no incluye intrínsecamente la capacidad de las bases de datos para relacionar tablas; no obstante, se pueden crear objetos *DataRelation* que establezcan una relación entre una tabla primaria (maestra) y una tabla secundaria (de detalle) sobre la base de una clave común.

El objeto *DataRelation* puede ejecutar dos funciones:

1. Puede poner a su disposición los registros relacionados al registro con el que esté trabajando; proporcionando los datos secundarios cuando se está en un registro primario y un registro primario si se está trabajando con uno secundario.

2. Puede exigir restricciones de integridad referencial, como la eliminación de los registros secundarios relacionados cuando se elimina un registro primario.

Es importante comprender la diferencia entre una combinación auténtica y la función de un objeto *DataRelation*. En una combinación auténtica, los registros se toman de tablas primarias y secundarias y se ponen en un único conjunto de registros plano. Cuando se utiliza un objeto *DataRelation* no se crea ningún conjunto de registros nuevo. En su lugar, se hace un seguimiento de la relación entre las tablas y se mantienen sincronizados los registros primarios y secundarios.

Si las tablas de un conjunto de datos tienen una relación lógica, un objeto *DataRelation* puede poner a su disposición los registros relacionados en otra tabla. Se puede utilizar el objeto *DataRelation* para obtener registros relacionados. El proceso es indirecto, puesto que no se combinan tablas. En su lugar, se llama al método ***GetChildRows*** de una fila de datos de la tabla primaria y se le pasa como parámetro el objeto *DataRelation* que define la relación entre primario y secundario. El método devuelve una matriz de registros secundarios relacionados.

De forma similar, se puede obtener la fila primaria de un registro secundario dado; para ello, se hace referencia al método ***GetParentRow*** de una fila de datos de la tabla secundaria. En este caso, el método no devuelve una matriz, sino una sola fila de datos.

Los objetos *DataRelation* se utilizan también para crear y exigir las siguientes restricciones:

- Una restricción ***UNIQUE***, que garantiza que una columna de una tabla no contenga duplicados.

- Una restricción ***FOREIGN KEY***, que puede utilizarse para mantener la integridad referencial entre una tabla primaria y una secundaria en un conjunto de datos.

Las restricciones que se especifican en objetos *DataRelation* se implementan mediante la creación automática de los objetos adecuados o el establecimiento automático de propiedades. Si se crea una restricción **FOREIGN KEY** mediante el objeto *DataRelation*, se agregan instancias de la clase ***ForeignKeyConstraint*** a la propiedad ***ChildKeyConstraint*** de la relación de datos.

Para implementar una restricción **UNIQUE** sólo hay que establecer la propiedad ***Unique*** de una columna de datos en *true* o agregar una instancia de la clase ***UniqueConstraint*** a la propiedad ***ParentKeyConstraint*** del objeto *DataRelation*.

Como parte de la restricción **FOREIGN KEY**, se pueden especificar reglas de integridad referencial que se aplican en tres situaciones:

1. Cuando se actualiza un registro primario
2. Cuando se elimina un registro primario
3. Cuando se acepta o se rechaza un cambio

Al crear un objeto *DataRelation*, se tiene la opción de especificar que la relación se utilice únicamente para exigir restricciones, es decir, que no se utilice también para el acceso a registros relacionados. Esta opción permite generar un conjunto de datos algo más eficaz y que contenga menos métodos que uno que tenga la capacidad de relacionar registros. Sin embargo, no se tendrá acceso a registros relacionados.

A través de la propiedad ***DataSet.Relations*** es posible obtener la colección de relaciones que vincula las tablas y permite el desplazamiento desde las tablas primarias a las secundarias. ***DataRelationCollection*** representa la colección de objetos *DataRelation* de un *DataSet*.

En resumen se puede decir que la clase ***DataRelation*** representa una relación primaria-secundaria entre dos objetos *DataTable*. Se utiliza un *DataRelation* para

relacionar dos objetos *DataTable* entre sí mediante objetos *DataColumn*. Las relaciones se crean entre columnas coincidentes de las tablas primarias y secundarias. Es decir, el valor ***DataType*** para ambas columnas debe ser idéntico.

Las relaciones también pueden originar varios cambios en cascada desde el *DataRow* primario hasta sus filas secundarias. Para controlar el cambio de valores en las filas secundarias, se agrega un ***ForeignKeyConstraint*** al ***ConstraintCollection*** del objeto *DataTable*. La ***ConstraintCollection*** determina la acción que se debe realizar cuando se elimina o actualiza un valor de una tabla primaria.

Cuando se crea un *DataRelation*, primero se comprueba que se puede establecer la relación. Después de agregarlo al *DataRelationCollection*, se mantiene la relación al no permitir ningún cambio que la invalide. Entre el periodo en que se crea un *DataRelation* y en que se agrega al *DataRelationCollection*, se pueden realizar cambios adicionales en las filas primarias o secundarias. Se generará una excepción si el resultado es una relación que ya no es válida.

Los objetos *DataRelation* se incluyen en un *DataRelationCollection*, al que se puede obtener acceso mediante la propiedad *Relations* del *DataSet*; y las propiedades ***ChildRelations*** y ***ParentRelations*** del *DataTable*.

Para mostrar como se crean las relaciones en un *DataSet* es preciso agregar una segunda tabla a la base de datos Directorio (contenida dentro del archivo *directorio.mdb* creada para este ejemplo). La nueva tabla se llamará ***"direcciones"*** y contendrá los campos citados en la ***tabla 2.24.***

Nombre del Campo	Tipo	Tamaño	Características	Descripción
telefono	Texto	30	Llave Principal, Requerido, NO duplicado	Teléfono de la persona registrada en la tabla "directorio".
calle	Texto	30	No Requerido	Nombre de la calle de su dirección.
colonia	Texto	30	No Requerido	Nombre de la colonia de su dirección.

Tabla 2.24 → Campos de la tabla *direcciones*

A continuación se deberán agregar algunos registros en la tabla, los cuales, deberán coincidir en el campo teléfono de la tabla directorio. Es importante que al menos exista una dirección que coincida en esta relación para poder ver los

resultados de este ejemplo, además de que servirán de la misma forma en el siguiente apartado, donde se podrán ver por completo las relaciones que se creen en esta sección.

Antes de poder crear las relaciones dentro del *DataSet* es necesario que se haya agregado la tabla "direcciones" como un *DataTable* dentro del mismo; para ello se agregará el código del ***listado 2.28*** dentro del *Módulo Global* del proyecto.

```
'*************************************************
'Declaración de objetos para la tabla Direcciones
'Adaptador de Datos
Public myAdapter_Direcciones As _
System.Data.OleDb.OleDbDataAdapter = _
New OleDb.OleDbDataAdapter
'Constructor de comandos para la tabla Direcciones
Public myCommand_builder_Direcciones As _
System.Data.OleDb.OleDbCommandBuilder
'Tabla Direcciones
Public myTable_Direcciones As DataTable
'RowCollection para la tabla myTableDirecciones
Public myRow_collections_Direcciones As _
DataRowCollection
'*************************************************
```

Listado 2.28 → Declaración del nuevo *DataTable*

Posteriormente se deberá agregar el código del ***listado 2.29*** al final del evento ***LOAD*** del formulario ***frmDirectorio*** para poder agregar la nueva tabla al *DataSet*.

```
'-------------------------------------------------------------
'Se genera el Adaptador de Datos para la tabla direcciones
Dim QueryDirecciones As String
'Se genera la  consulta
QueryDirecciones = "SELECT * FROM direcciones"
Try
    'antes que nada abrimos la conexión
    myConnection.Open()
    myAdapter_Direcciones.SelectCommand = _
    New OleDb.OleDbCommand(QueryDirecciones, myConnection)
Catch ex As Exception
    MessageBox.Show(ex.Message)
    Exit Sub
End Try
```

```
'Se genera el constructor de comandos
Try
    myCommand_builder_Direcciones = New _
    OleDb.OleDbCommandBuilder(myAdapter_Direcciones)
Catch ex As Exception
    MessageBox.Show(ex.Message)
    Exit Sub
End Try
'Se agrega al DataSet la nueva tabla
If myConnection.State <> ConnectionState.Open Then
    Try
        myConnection.Open()
    Catch ex As Exception
        MessageBox.Show(ex.Message)
        Exit Sub
    End Try
Else
    Try
        'Se Agregan las columnas necesarias e información
        'sobre la clave principal para completar el esquema.
        myAdapter_Direcciones.MissingSchemaAction = _
        MissingSchemaAction.AddWithKey
        'Se llena el DataSet con la tabla completa "direcciones"
        myAdapter_Direcciones.Fill(myData_set, "direcciones")
        'Se cierra la conexión
        myConnection.Close()
    Catch ex As Exception
        MessageBox.Show(ex.Message)
        Exit Sub
    End Try
End If
'Se genera la nueva tabla a partir del DataSet
Try
    myTable_Direcciones = myData_set.Tables("direcciones")
Catch ex As Exception
    MessageBox.Show(ex.Message)
    Exit Sub
End Try
'Por último se genera la colección de tablas para ediciones
'y cambios posteriores
Try
    myRow_collection_Direcciones = myTable_Direcciones.Rows
Catch ex As Exception
    MessageBox.Show(ex.Message)
    Exit Sub
End Try
'------------------------------------------------------------
```

Listado 2.29 → Adición de la tabla *direcciones* al *DataSet*

Lo siguiente será generar la relación en si, y para ello se deberá agregar el código del ***listado 2.30*** al final del evento ***LOAD*** del formulario ***frmDirectorio.***

```
'#########################################################
'Se agrega la nueva relación entre las dos
'tablas en el DataSet
Try
    myData_set.Relations.Add _
    ("myRelacion", myTable.Columns("telefono"), _
    myTable_Direcciones.Columns("telefono"))
Catch ex As Exception
    MessageBox.Show(ex.Message)
End Try
'#########################################################
```

Listado 2.40 → Adición de la relación dentro del DataSet

Para probar que efectivamente se haya agregado la relación entre las tablas se deberá agregar un botón con las propiedades que se enlistan el la ***tabla 2.25.***

Propiedad	Valor
Name	btMostrarRelaciones
Text	Mostrar Relaciones

Tabla 2.25 → Propiedades del botón para mostrar las relaciones entre las tablas

Ahora es necesario agregar el procedimiento para el botón agregado, y así poder ver como se puede acceder a los registros vinculados por la relación que se acaba de crear. Para ello se debe agregar el código del ***listado 2.41*** al evento click del botón *btMostrarRelaciones*.

```
Private Sub btMostrarRelaciones_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btMostrarRelaciones.Click
        'Si la relación es de uno a varios,
        'la consulta de la relación devolverá
        'un arreglo de DataRows
        Dim FilasRelacionadas_Padre() As DataRow
        'Sólo para conocer el tamaño del
        'arreglo que se nos ha devuelto
        Dim i As Integer
        Try
            'La manera correcta de acceder a los datos
            'de una relación es buscando dentro de la tabla
            'padre la fila a través de su llave principal y
            'accediendo al método GetChildRows
            FilasRelacionadas_Padre = myTable.Rows.Find _
            (txtNombre.Text.Trim).GetChildRows("myRelacion")
```

```
            'Se debe verificar que la relación haya devuelto
            'al menos una fila
            i = FilasRelacionadas_Padre.GetUpperBound(0)
            If i = 0 Then
                'En este momento se puede tener la certeza
                'de que efectivamente existe al menos
                'un registro de la tabla hija
                Dim szCalle As String = _
                FilasRelacionadas_Padre(0)("calle")
                Dim szColonia As String = _
                FilasRelacionadas_Padre(0)("colonia")
                Dim szTelefono As String = _
                FilasRelacionadas_Padre(0)("telefono")
                'Como ejemplo podemos acceder a la
                'relación pero de manera inversa,
                'es decir, a través de la tabla
                'hija
                Dim FilaRelacionada_Hija As DataRow = _
                myTable_Direcciones.Rows.Find _
                (szTelefono.Trim).GetParentRow("myRelacion")
                'Ahora es posible acceder a los datos
                'de la fila devuelta por la relación
                'hija a padre
                Dim szNombre As String = _
                FilaRelacionada_Hija("nombre")
                Dim szEmail As String = _
                FilaRelacionada_Hija("email")
                'Ahora es posible realizar cualquier acción con los
                'datos obtenidos a través de las relaciones,
                'para este ejemplo simplemente se mostrarán
                'los datos obtenidos en una caja de texto
                MsgBox( _
                "Nombre: " & szNombre & ControlChars.CrLf & _
                "Teléfono: " & szTelefono & ControlChars.CrLf & _
                "E-Mail: " & szEmail & ControlChars.CrLf & _
                "Calle: " & szCalle & ControlChars.CrLf & _
                "Colonia: " & szColonia, MsgBoxStyle.OKOnly, _
                "Relaciones Existentes")
            Else
                MessageBox.Show("El Registro actual NO posee " & _
                "datos dentro de la tabla 'direcciones'" _
                , "Relaciones", _
                MessageBoxButtons.OK, _
                MessageBoxIcon.Information)
            End If
        Catch ex As Exception
            MsgBox(ex.Message)
        End Try
    End Sub
```

Listado 2.41 → Código del evento click del botón btMostrarRelaciones

Cuando se presione el botón para mostrar las relaciones, si existe un registro en las dos tablas y que contengan el mismo teléfono, se deberá mostrar un mensaje muy parecido al de la ***figura 2.64.***

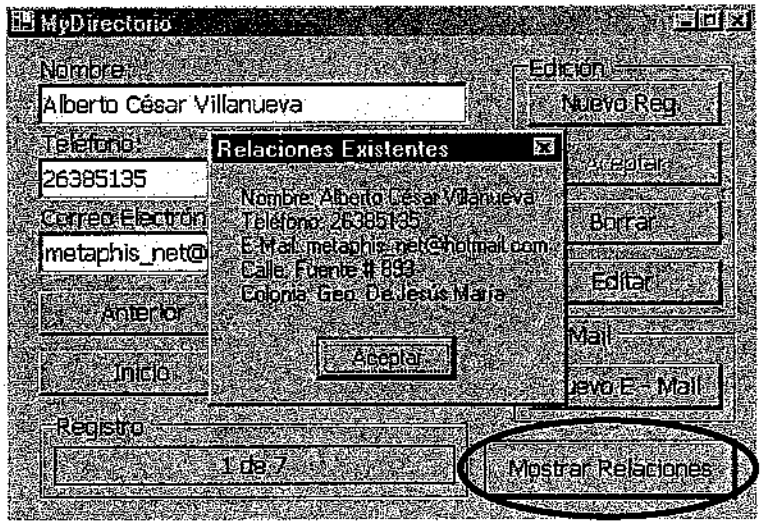


Figura 2.64 → Mensaje que muestra los datos extraídos a través de la relación

Es muy importante recordar que cuando se accede al método ***GetChildRows*** de una fila de datos de la tabla padre, el método devuelve una matriz de registros secundarios relacionados, cuando la relación creada es de uno a varios; de manera inversa es posible acceder a la fila primaria de un registro secundario accediendo al método ***GetParentRow*** de una fila de datos de una tabla secundaria, pero en este caso, el método no devolverá una matriz de filas, sino una sola fila de datos.

2.12.8 Rejilla de Visualización a través del Control DataGrid

El control ***DataGrid*** es una lista enlazada a datos que muestra los elementos del origen de datos en una rejilla de visualización separando las columnas y las filas de las tablas en cuadrículas. El control *DataGrid* permite seleccionar, ordenar y editar los registros contenidos en cualquiera de los siguientes orígenes de datos:

- DataTable
- DataView
- DataSet
- DataViewManager

Se utiliza el control *DataGrid* para mostrar los campos de un origen de datos como columnas en una tabla. Cada fila del control *DataGrid* representa un registro del origen de datos. El control *DataGrid* admite las operaciones de seleccionar, editar, eliminar, paginar y ordenar.

Es importante recordar que el orden en que se muestran las columnas en el control *DataGrid* se rige por el orden en que aparecen las columnas en el origen de los datos. Si bien se puede cambiar mediante programación el orden de las columnas, resulta más fácil mostrarlas en el orden del origen, como se mencionó en el ***apartado 2.12.1 (listado 2.4)***.

Los objetos *BindingManagerBase* administran asimismo los orígenes de datos. Para cada tabla de un origen de datos, se puede devolver un objeto *BindingManagerBase* desde *BindingContext* del formulario. Por ejemplo, se puede determinar el número de filas de un origen de datos devolviendo la propiedad *Count* del objeto *BindingManagerBase* asociado.

Para validar los datos, se utilizan los objetos subyacentes que representan los datos y sus eventos. Por ejemplo, si los datos proceden de un objeto *DataTable* de *DataSet*, se utilizan los eventos *ColumnChanging* y *RowChanging*.

Dado que es posible personalizar el número de columnas (agregando o eliminando miembros de *GridColumnStylesCollection*) y que las filas pueden ordenarse por columna, no es posible garantizar que los valores de las propiedades *RowNumber* y *ColumnNumber* se correspondan con los índices de *DataRow* y *DataColumn* en *DataTable*. Por este motivo, se ha de evitar el uso de dichas propiedades en el evento *Validating* para validar los datos.

Para determinar la celda seleccionada, se utiliza la propiedad *CurrentCell*. Para cambiar el valor de cualquier celda, se utiliza la propiedad *Item*, que puede adoptar los índices de fila y columna de la celda o un solo objeto *DataGridCell*. Se debe supervisar el evento *CurrentCellChanged* para detectar cuándo el usuario selecciona otra celda.

Para determinar en que parte del control ha hecho click el usuario, se utiliza el método *HitTest* en el evento *MouseDown*. El método *HitTest* devuelve un objeto *DataGrid.HitTestInfo*, que contiene la fila y la columna de un área en la que se ha hecho click.

Para administrar la apariencia del control en tiempo de ejecución, existen varias propiedades que permiten establecer los atributos de color y título, incluidas las propiedades *CaptionForeColor*, *CaptionBackColor*, *CaptionFont*, etc.

La apariencia de la cuadrícula o cuadrículas mostradas puede modificarse también creando objetos *DataGridTableStyle* y agregándolos a la colección *GridTableStylesCollection*, a la que se puede obtener acceso mediante la propiedad *TableStyles*. Por ejemplo, si la propiedad *DataSource* está establecida en una clase *DataSet* que contiene tres objetos *DataTable*, se pueden agregar tres objetos *DataGridTableStyle* a la colección, uno por cada tabla. Para sincronizar cada objeto *DataGridTableStyle* con un objeto *DataTable*, se bebe establecer la propiedad *MappingName* de *DataGridTableStyle* en *TableName* de *DataTable*.

Para crear una vista personalizada de una tabla, hay que crear una instancia de una clase *DataGridTextBoxColumn* o *DataGridBoolColumn* y agregar el objeto a la colección *GridTableStylesCollection*, a la que se obtiene acceso a través de la propiedad *TableStyles*. Ambas clases se heredan de *DataGridColumnStyle*. Para cada estilo de columna, hay que establecer la propiedad *MappingName* en *ColumnName* de una columna que se desee mostrar en la cuadrícula. Para ocultar una columna, se establece su propiedad *MappingName* en un valor que no sea una propiedad *ColumnName* válida.

Para aplicar formato al texto de una columna, hay que establecer la propiedad *Format* de *DataGridTextBoxColumn* en uno de los valores de *cadenas de formato de fecha y hora* o *cadenas de formato numérico estándar*.

Para enlazar *DataGrid* a una matriz de objetos con establecimiento inflexible de tipos, el objeto debe contener propiedades públicas. Para crear un *DataGridTableStyle* que muestre dicha matriz, hay que establecer la propiedad *DataGridTableStyle.MappingName* en *classname[]*, donde *classname* se reemplaza por el nombre de clase. Además, hay que tener en cuenta que la propiedad *MappingName* distingue mayúsculas de minúsculas.

Para cada *DataGridTableStyle*, se pueden establecer los atributos de color y título que reemplazan los valores del control *System.Windows.Forms.DataGrid*. Sin embargo, si no se establecen dichas propiedades, se utilizan los valores del control de manera predeterminada.

Las propiedades *DataGridTableStyle* pueden reemplazar las siguientes propiedades:

- *AllowSorting*
- *AlternatingBackColor*
- *BackColor*
- *ColumnHeadersVisible*
- *ForeColor*
- *GridLineColor*
- *GridLineStyle*
- *HeaderBackColor*
- *HeaderFont*
- *HeaderForeColor*
- *LinkColor*
- *PreferredColumnWidth*
- *PreferredRowHeight*
- *ReadOnly*
- *RowHeadersVisible*
- *RowHeaderWidth*
- *SelectionBackColor*
- *SelectionForeColor*

Para personalizar la apariencia de columnas individuales, se agregan objetos *DataGridColumnStyle* a la colección *GridColumnStylesCollection*, a la que se obtiene acceso mediante la propiedad *GridColumnStyles* de cada *DataGridTableStyle*. Para sincronizar cada objeto *DataGridColumnStyle* con un objeto *DataColumn* en *DataTable*, se debe establecer la propiedad *MappingName* en la propiedad *ColumnName* de un objeto *DataColumn*. Al construir un objeto *DataGridColumnStyle*, también se puede establecer una cadena de formato que especifica el modo en que se muestran los datos en la columna. Por ejemplo, se puede especificar que la columna utiliza un formato de fecha corta para mostrar las fechas que contiene una tabla.

Primero deben crearse los objetos *DataGridColumnStyle* y, después, deben agregarse a *GridColumnStylesCollection* antes de agregar los objetos *DataGridTableStyle* a *GridTableStylesCollection*. Cuando se agrega un *DataGridTableStyle* vacío a la colección, se generan automáticamente objetos *DataGridColumnStyle*. Por lo tanto, se iniciará una excepción si se intentan agregar nuevos objetos *DataGridColumnStyle* a *GridColumnStylesCollection* con valores de *MappingName* duplicados.

Para mostrar una tabla en un ***System.Windows.Forms.DataGrid*** en tiempo de ejecución se debe acceder al método *SetDataBinding* para establecer las propiedades *DataSource* y *DataMember* en un origen de datos válido.

Para continuar el ejemplo, se agregará un control *DataGrid* al formulario *frmDirectorio* con las propiedades listadas en la ***tabla 2.26:***

Propiedad	Valor
Name	DataGridDirectorio
CaptionFont	Tahoma, 8.25pt, style=Bold
CaptionText	Directorio de Contactos
Font	Tahoma, 8.25pt
ReadOnly	True*

Tabla 2.26 → Propiedades del control DataGridDirectorio

*La propiedad *ReadOnly* del control *DataGrid* obtiene o establece un valor que indica si la cuadrícula se encuentra en modo de sólo lectura. Si el valor de la propiedad se establece en *true* la cuadrícula estará en modo de sólo lectura; en caso contrario, tomará el valor predeterminado, *false*.

Cuando el *DataGrid* se encuentra en modo de sólo lectura, la cuadrícula puede desplazarse, los nodos pueden expandirse o contraerse, etc. Sin embargo, no es posible realizar adiciones, ediciones ni eliminaciones. ***DataGridColumnStyle*** también tiene una propiedad *ReadOnly* que puede establecerse en *true* para evitar que se editen los datos columna por columna. La propiedad *ReadOnly* puede establecerse en *true* si se desea impedir que el usuario edite directamente los datos en el *DataGrid*. Por ejemplo, es posible hacer que los usuarios vean todas las columnas de una tabla pero que sólo puedan editar determinados campos mediante los controles *TextBox* de algún formulario.

Cuando se haya agregado el control al formulario *frmDirectorio* es posible vincularla a los datos de las tablas de dos formas, la primera es directamente al *DataSet* y la otra es vincularla al *DataView*. Existe una diferencia primordial, y esta estriba en que si se vincula con un *DataView*, cualquier cambio que se realice sobre la vista personalizada del *DataView*, ésta afectará a los datos mostrados en el *DataGrid*, y lo mismo sucederá de forma inversa, es decir, cualquier cambio que se realice dentro del *DataGrid* se verá reflejado en los controles a los que esté vinculado dicha vista personalizada. En este ejemplo es posible vincularla por cualuiera de los dos métodos, y se dejará como ejercicio al usuario que realice la conexión de las dos formas para poder ver esta diferencia; por el momento se necesitará agregar el código del ***listado 2.42*** al evento ***LOAD*** del formulario *frmDirectorio* para vincular el *DataGrid* con los datos contenidos en el *DataView*.

```
'-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
'Las siguientes líneas son para indicar que el origen
'de los datos es el DataSet
'
'DataGridDirectorio.DataSource = myData_set
'DataGridDirectorio.DataMember = "Directorio"
'
'Se accede a la propiedad DataSource para establecer
'el origen de datos para el que muestra datos
'la cuadrícula, que en este caso, es el DataView
DataGridDirectorio.DataSource = myData_view
'-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

Listado 2.42 → Estableciendo el origen de los datos del *DataGrid*

Cuando se haya agregado este código al evento ***LOAD***, es posible ver el resultado, el cual debe ser muy parecido al de la ***figura 2.65,*** en la cual se muestra el aspecto de la rejilla de visualización que se ha agregado.

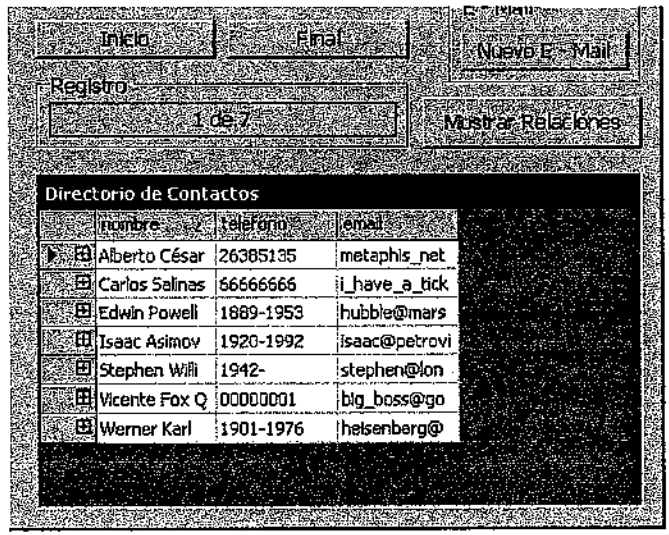


Figura 2.65 → Rejilla de visualización

En este momento se puede observar que todos los datos contenidos en el *DataView* son mostrados dentro de la rejilla de visualización. También es importante destacar el hecho de que es posible ubicar un registro en la parte superior simplemente dando click en el registro en que se desea posicionarse.

También es posible ordenar los registros por su teléfono, email o cualquier columna agregada al *DataView*, ya sea de forma ascendente o descendente.

También se habrá notado que automáticamente se ha agregado la relación que se ha creado con anterioridad ***(ver el apartado 2.12.7),*** y es posible ver los datos contenidos en la tabla secundaria simplemente dando click en la relación como se muestra en la ***figura 2.66.***

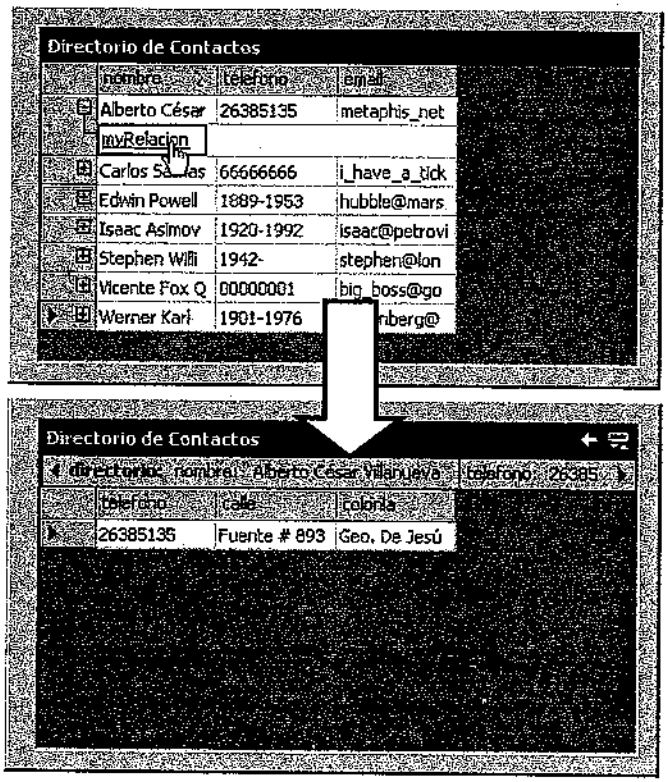


Figura 2.66 → Vista de las relaciones dentro del *DataGrid*

Es posible ocultar las relaciones en el *DataGrid* para que no sean accesibles para el usuario; esto simplemente accediendo a la propiedad *DataGrid.AllowNavigation*, la cual establece el valor *true* o *false* que indica si se permite el desplazamiento a través de las tablas relacionadas. Para ejemplificar lo anterior se puede agregar el código del ***listado 2.43*** al evento ***LOAD*** del formulario *frmDirectorio*.

```
DataGridDirectorio.AllowNavigation = False
```

Listado 2.43 → Deshabilitar la navegación de las tablas relacionadas en el *DataGrid*

También es posible controlar la apariencia de la cuadrícula dibujada en el *DataGrid* accediendo a los métodos y propiedades de la clase ***DataGridTableStyles***. Dicha clase representa sólo la cuadrícula dibujada; esta cuadrícula no debe confundirse con la clase *DataView*, que es uno de los posibles orígenes de datos de la cuadrícula; en cambio, *DataGridTableStyle* representa estrictamente la cuadrícula tal como está dibujada en el control. Por consiguiente, mediante *DataGridTableStyle* se puede controlar la apariencia de la cuadrícula de cada *DataView* o *DataSet* representado en la cuadrícula. Para especificar que *DataGridTableStyle* se utiliza al mostrar datos de un objeto *DataTable*, se establece la propiedad ***MappingName*** en la propiedad ***TableName*** de un objeto *DataTable*.

GridTableStylesCollection contiene todos los objetos *DataGridTableStyle* utilizados por un control *System.Windows.Forms.DataGrid*. La colección puede contener tantos objetos *DataGridTableStyle* como se precisen, si bien la propiedad *MappingName* de cada uno debe ser única. En tiempo de ejecución, esto permite sustituir un objeto *DataGridTableStyle* diferente por los mismos datos, dependiendo de la preferencia del usuario. Para ello es posible:

- Rellenar *GridTableStylesCollection* con objetos ***DataGridTableStyle***. Si existe un objeto *DataGridTableStyle* en la colección ***GridTableStylesCollection*** cuya propiedad *MappingName* tiene un valor que equivale a la propiedad *TableName* del objeto *DataTable*, *DataTable* se muestra con este objeto *DataGridTableStyle*. Si no existe ningún objeto *DataGridTableStyle* con una propiedad *MappingName* coincidente, *DataTable* se muestra con el estilo predeterminado para las tablas de la cuadrícula de datos.

- Cuando se requiera un estilo de cuadrícula diferente, se deberá la propiedad ***Item*** para seleccionar el objeto *DataGridTableStyle* apropiado (pase *TableName* a la propiedad *Item*) y se deberá establecer la propiedad *MappingName* del objeto devuelto en un valor nuevo.

- Se puede utilizar la propiedad *Item* para seleccionar el objeto *DataGridTableStyle* deseado y establezca su propiedad *MappingName* en la propiedad *TableName* de *DataTable*.

Si el *DataSet* origen contiene objetos *DataTable* relacionados mediante objetos *DataRelation* y el objeto *DataTable* que se muestra actualmente es una tabla secundaria, la propiedad *DataMember* devolverá una cadena en forma de ***NombreDeTabla.NombreDeRelación*** (en el caso más sencillo). Si *DataTable* se encuentra en un nivel inferior de la jerarquía, la cadena se compondrá del nombre de la tabla primaria seguido de los valores de *RelationName* necesarios para alcanzar el nivel de la tabla.

La colección de objetos *DataGridTableStyle* se devuelve mediante la propiedad *TableStyles* de *System.Windows.Forms.DataGrid.*

Cuando se muestra un objeto *DataGridTableStyle*, los valores de *DataGridTableStyle* reemplazarán los valores del control *DataGrid.* Si no se establece ningún valor para una propiedad de *DataGridTableStyle* determinada, se utilizará el valor del control *DataGrid.* En la siguiente lista se muestran las propiedades de *DataGridColumnStyle* que pueden establecerse para que reemplacen las propiedades del control *DataGrid:*

- *AllowSorting*
- *AlternatingBackColor*
- *BackColor*
- *ColumnHeadersVisible*
- *ForeColor*
- *GridLineColor*
- *GridLineStyle*
- *HeaderBackColor*
- *HeaderFont*
- *HeaderForeColor*
- *LinkColor*
- *PreferredColumnWidth*
- *PreferredRowHeight*
- *ReadOnly*
- *RowHeadersVisible*
- *RowHeaderWidth*
- *SelectionBackColor*
- *SelectionForeColor*

La propiedad *DataGrid.TableStyles* obtiene la colección de objetos *DataGridTableStyle* de la cuadrícula. Se utiliza *GridTableStylesCollection* para crear vistas personalizadas de cada tabla mostrada por el control *DataGrid.* De manera predeterminada, la colección que devuelve la propiedad *TableStyles* no contiene ningún objeto *DataGridTableStyle.* Para crear un conjunto de vistas personalizadas se debe seguir el siguiente procedimiento:

1. Se crea un objeto *DataGridTableStyle.*

2. Se establece la propiedad *MappingName* del objeto de tabla de la cuadrícula en la propiedad *TableName* de un objeto *DataTable.*

3. Se agregan objetos *DataGridColumnStyle*, uno por cada columna de cuadrícula que desee mostrar, a *GridColumnStylesCollection* que devuelve la propiedad *GridColumnStyles*.

4. Se establece la propiedad *MappingName* de cada objeto *DataGridColumnStyle* en la propiedad *ColumnName* de un objeto *DataColumn*.

5. Se agrega el objeto *DataGridTableStyle* a la colección que devuelve la propiedad *TableStyles*.

Es muy importante tener precaución, primero deben crearse los objetos *DataGridColumnStyle* y, después, deben agregarse a *GridColumnStylesCollection* antes de agregar los objetos *DataGridTableStyle* a *GridTableStylesCollection*. Cuando se agrega un *DataGridTableStyle* vacío a la colección, se generan automáticamente objetos *DataGridColumnStyle*. Por lo tanto, se iniciará una excepción si se intentan agregar nuevos objetos *DataGridColumnStyle* a *GridColumnStylesCollection* con valores de *MappingName* duplicados.

Como ejemplo es posible ocultar la última columna del *DataGrid* del proyecto agregando el código del ***listado 2.43*** al evento ***LOAD*** del formulario *frmDirectorio*.

```
'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
'Se obtiene la colección de objetos DataGridTableStyle
Dim EstilosRejilla As DataGridTableStyle = _
New DataGridTableStyle
'La siguiente línea verifica que coincida la propiedad del
'nombre de la tabla asignada como DataSource en el
'DataGrid con los datos que se muestran, es decir,
'se establece el nombre utilizado para asignar esta
'tabla a un origen de datos específico
EstilosRejilla.MappingName = "directorio"
'La siguiente línea es equivalente a la anterior
'EstilosRejilla.MappingName = myData_view.Table.TableName
'
'Si el DataGrid estuviera vinculado al DataSet y no al
'DataView, lo correcto sería vincular el nombre de la
'tabla al DataGrid con la siguiente línea en vez de
'cualquiera de las dos anteriores
'EstilosRejilla.MappingName = DataGridDirectorio.DataMember
'
'Se agrega el objeto DataGridTableStyle a la colección
'del DataGrid
DataGridDirectorio.TableStyles.Add(EstilosRejilla)
```

```
'Ahora es posible acceder a la colección de columnas
'dibujadas en el DataGrid, como por ejemplo, RemoveAt,
'que quita el objeto DataGridColumnStyle con el indice
'especificado de la colección GridColumnStylesCollection.
Try
    DataGridDirectorio.TableStyles(0). _
    GridColumnStyles.RemoveAt(2)
Catch ex As Exception
    MsgBox(ex.Message)
End Try
'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Listado 2.43 → Ejemplo de acceso a los estilos de la cuadrícula dibujada en el *DataGrid*

Es posible ver el resultado de este cambio en la ***figura 2.67,*** la cual muestra el *DataGrid* con los cambios realizados en su estilo.

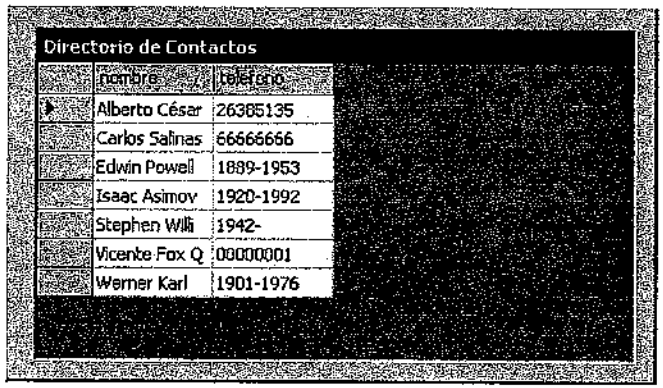


Figura 2.67 → Vista del control *DataGrid* después de quitar la última columna

También es posible acceder a las propiedades de color, ancho y formato de las celdas dibujadas en el *DataGrid* y cambiar su estado predeterminado en tiempo de ejecución, todo esto para hacer más flexible la forma en que se muestran los datos dentro de este control.

En resumen, se puede afirmar que el control *DataGrid* es la mejor manera de mostrar los datos de *ADO.Net* en una cuadrícula desplazable, flexible y sumamente versátil. Es posible acceder a sus propiedades y cambiar el estilo en que muestra los datos y también se puede visualizar a través de ella las relaciones creadas dentro de un *DataSet*; cambiar, editar y agregar nuevos registros y al mismo tiempo, validar las entradas del usuario.

2.12.9 Búsqueda de Registros en un DataView

Si ha leído por completo este apartado, y ha sido observador en los códigos de los listados, habrá notado que ya se ha programado un procedimiento que incluye un método para buscar registros en un *DataRowCollection* o *DataView*, ya que este método esta incluido para cualquiera de las dos clases.

Existen dos métodos relacionados con búsquedas de registros en un *DataView*, estos son ***Find*** y ***FindRows***, los cuales puede buscar filas según sus valores de clave de ordenación. La diferenciación entre mayúsculas y minúsculas de los valores de búsqueda en los métodos *Find* y *FindRows* está determinada por la propiedad ***CaseSensitive*** del *DataTable* subyacente. Los valores de búsqueda deben coincidir en su totalidad con los valores de clave de ordenación existentes para que se devuelva un resultado.

El método *Find* devuelve un entero con el índice de la ***DataRowView*** que coincide con los criterios de búsqueda. Si más de una fila coincide con los criterios de búsqueda, sólo se devolverá el índice de la primera *DataRowView* coincidente. Si no se encuentra ninguna coincidencia, *Find* devolverá **-1.**

Para devolver resultados de la búsqueda que coincidan con varias filas, se puede utilizar el método *FindRows*. *FindRows* funciona igual que el método *Find*, excepto en que devuelve una matriz de *DataRowView* que hace referencia a todas las filas coincidentes de la *DataView*. Si no se encuentra ninguna coincidencia, la matriz de *DataRowView* estará vacía.

Para utilizar los métodos *Find* o *FindRows* se debe especificar un criterio de ordenación; para ello, se establece *ApplyDefaultSort* como *true* o se podrá utilizar la propiedad ***Sort***. Si no se especifica ningún criterio de ordenación, se inicia una excepción.

Los métodos *Find* y *FindRows* toman como entrada una matriz de valores cuya longitud coincide con el número de columnas del criterio de ordenación. En el caso de una ordenación por una única columna, puede pasar un único valor. Para los criterios de ordenación que contienen varias columnas, debe pasar una matriz de objetos. Se debe tener en cuenta que para una ordenación según varias columnas, los valores de la matriz de objetos deben coincidir con el orden de las columnas especificado en la propiedad *Sort* del *DataView*.

En resumen el método *Find* encuentra una fila en el *DataView* mediante el valor de clave de ordenación especificado; y el método *FindRows* devuelve una matriz de objetos *DataRowView* cuyas columnas coinciden con el valor de clave de

ordenación especificado. Como ejemplo, se adicionará un botón para realizar búsquedas según el nombre del contacto dentro del formulario *frmDirectorio*. Para ello se agregarán los controles listados en la ***tabla 2.27*** al formulario antes mencionado.

Objeto	Propiedad	Valor
TextBox	Name Text	**txtBusquedaRegistro** (Vacío)
Button	Name Text	**btBuscarRegistro** Buscar

Tabla 2.27 → Controles especiales para búsqueda de registros

Después de haber agregado los controles antes mencionados, el aspecto del formulario debería ser algo parecido a la ***figura 2.68***, la cual muestra el nuevo aspecto del formulario *frmDirectorio*.

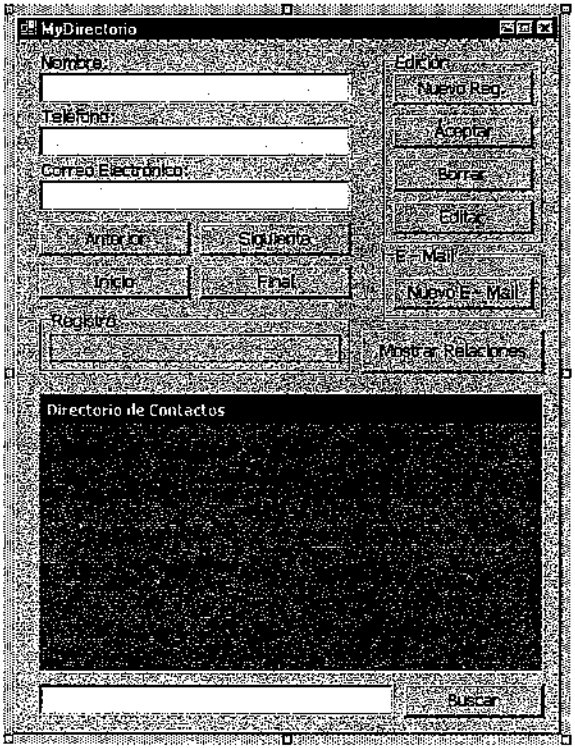


Figura 2.68 → Formulario frmDirectorio

Después de haber agregado los controles, se podrá agregar el código del ***listado 2.44*** a la llamada del evento click del botón *btBuscarRegistro*.

```
Private Sub btBuscarRegistro_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btBuscarRegistro.Click
        'La caja de texto es el campo obligatorio
        'para la búsqueda
        If txtBusquedaRegistro.Text.Length > 0 Then
            'Generamos un arreglo de filas
            Dim FilasEncontradas() As DataRow
            'Creamos el criterio para la búsqueda
            Dim szCriterio As String = "Nombre LIKE '*" & _
            txtBusquedaRegistro.Text.Trim & "*'"
            'El método Select de DataTable obtiene
            'una matriz de objetos DataRow que
            'coinciden con los criterios
            'de filtro por orden de clave
            'principal (o si ésta no existe,
            'por orden de adición).
            FilasEncontradas = _
            myTable.Select(szCriterio, "nombre")
            'La siguiente línea es equivalente a la anterior
            'FilasEncontradas = _
            'myData_view.Table.Select(szCriterio, "nombre")
            '
            'Se revisa cuántos elementos hay en el arreglo
            Dim i As Integer = FilasEncontradas.GetUpperBound(0)
            'Si no se ha encontrado ningún registro
            If i = -1 Then
                MessageBox.Show( _
                "No se ha encontrado el nombre que busca", _
                "Búsqueda sin Resultado", MessageBoxButtons.OK, _
                MessageBoxIcon.Information)
                'Se vacía la caja de texto
                txtBusquedaRegistro.Text = ""
                Exit Sub
                'Se se ha encontrado un contacto con este nombre
            ElseIf i = 0 Then
                'Se obtiene el nombre completo del
                'contacto
                Dim szNombreCompleto As String = _
                FilasEncontradas(0)("nombre")
                'Se ordenan los contactos por nombre
                myData_view.Sort = "nombre"
                'Se refresca el DataGrid
                DataGridDirectorio.Refresh()
                'Se obtiene entonces la posición
                'del contacto dentro del DataGrid
                Dim j As Integer = myData_view.Find(szNombreCompleto)
```

```
                'Por último se posiciona al contacto
                'en el formulario para ver sus datos
                BindingContext(myData_view).Position = j
                'Se muestra la posición actual
                Muestra_Posicion()
                'Se vacía la caja de texto
                txtBusquedaRegistro.Text = ""
                Exit Sub
            End If
        End If
    End Sub
```

Listado 2.44 → Código para el evento dlick del botón btBuscarRegistro

Ahora es posible realizar búsquedas de los contactos por su nombre completo o sólo ingresando parte del mismo; pero como habrá notado, este procedimiento funciona para cuando la búsqueda sólo encuentra un contacto con el criterio escrito; por lo tanto la pregunta obvia es ¿Qué pasa si la búsqueda devuelve más de un registro?; pues la respuesta es muy simple, se deberá adicionar un procedimiento para la condición: ***"ElseIf i>0 Then".*** Este procedimiento no se ha escrito en este apartado deliberadamente, y como ejercicio final de este apartado, se deja al lector que realice tal procedimiento por si mismo, ya que hasta ahora cuenta con los conocimientos suficientes para realizar tal programa de manera satisfactoria.

Capítulo 3

Página de Internet

Internet ha creado una revolución sin precedentes en el mundo de la informática y de las comunicaciones. Los inventos del telégrafo, teléfono, radio, televisión y computadora sentaron las bases para esta integración. Internet es a la vez una oportunidad de difusión mundial, un mecanismo de propagación de la información y un medio de colaboración e interacción entre los individuos y sus computadoras independientemente de su localización geográfica.

Internet representa uno de los ejemplos más exitosos de los beneficios de la investigación y desarrollo en infraestructuras informáticas. A raíz de la ***ARPANET (Advanced Research Projects Agency Network o Red de la Agencia de Proyectos de Investigación Avanzada)***, el gobierno, la industria y el mundo académico han sido copartícipes de la evolución y desarrollo de esta nueva y excitante tecnología. Hoy en día, términos como alguien@mi_empresa.com y http://www.mi_empresa.com.mx fluyen fácilmente en el lenguaje común de las personas.

El objetivo principal de una página de Internet es facilitar la comunicación entre las asociaciones, el gobierno, las instituciones educativas, las empresas y sus clientes o cualquier tipo de persona sea más estrecha entre ellos, particularmente en el aspecto comercial, el cual estriba en fomentar una mayor comunicación entre clientes y empresas estableciendo un modelo de operación de negocios orientado a una experiencia de satisfacción y personalización en el cliente durante sus adquisiciones en línea.

No existe hoy en día un medio de comunicación que por su dinamismo e inmensurable crecimiento pueda equipararse a Internet. A diferencia de otros medios tradicionales de información, un sitio web permite mantener una comunicación en ambos sentidos y puede sustituir y hacer más eficientes algunos canales actuales de información. Cualquier negocio puede ahora atender en línea las necesidades específicas de sus clientes que anteriormente era imposible atender de una forma más personal. Además de reducir costos operativos en la mayoría de los casos, los clientes se sienten mucho más cerca de su empresa y atraídos por la seguridad de poder realizar cualquier tipo de transacción desde la comodidad de sus casas u oficinas.

La AMD cuenta con su página de Internet, la cual, posiciona a esta institución a la vanguardia de la tecnología y pone a disposición de sus miembros la información más relevante acerca de sus eventos. Además de dar al público en general un listado de sus miembros para que puedan acudir, de así requerirlo, con un médico especialista en prevención, diagnóstico y tratamiento de todos los padecimientos y tratamientos de la piel, pelo, uñas y mucosas con la seguridad de ser atendidos por los mejores dermatólogos certificados del país.

3.1 Antecedentes

Durante la administración anterior, la AMD ya contaba con una página de Internet; sin embargo, ésta no contaba con la funcionalidad necesaria para que se desempeñe acorde a las expectativas de la mesa directiva; fue hasta el arribo de la nueva administración que se tomó la decisión de realizar una página más práctica y coherente a las necesidades de la AMD.

Hasta abril del 2004 la administración del dominio y el hospedaje de la página lo gestionaba una empresa llamada ***"Mexis"***. Cuando la nueva administración tomó protesta de la dirección de la AMD, se tomó la decisión de no seguir pagando la asistencia de dicha empresa, ya que la misma no proporcionaba el servicio al que se había comprometido, además de no brindar la información necesaria para la manutención de la página; eso además de ser sumamente caro y deficiente.

Fue entonces que se presentó la propuesta de que sea la AMD la que lleve la gestión tanto del nombre del dominio como de un hospedaje adecuado a las necesidades de la academia. Así pues, la AMD lleva ahora la administración de todas las partes que forman en conjunto su página de Internet.

3.2 Planeación y Definición de Requerimientos

La AMD requiere tener una forma de dar a conocer a sus miembros toda la información acerca de los eventos que realiza de forma continua y oportuna; también necesita dar a conocer los datos de la institución para que los médicos interesados en ingresar a ésta, cuenten con la información necesaria para contactar a la academia, así como estar al tanto de los requisitos de ingreso a la AMD.

Es importante dar a conocer a los miembros de la AMD las personas que conforman la Mesa Directiva, la agenda de la academia, la información acerca de las sesiones mensuales, la información de las inscripciones del siguiente evento y también es necesario un pequeño espacio para dar a conocer la Fundación Mexicana para la Dermatología, A.C.

Para la AMD es importante dar un espacio publicitario a todos los laboratorios suscritos como laboratorios tipo "A" para que den a conocer sus productos y medicamentos, y por supuesto, la página de Internet no es la excepción. Por esto es importante destinar un espacio especial para estos laboratorios y que puedan presentar de manera equitativa sus espacios publicitarios.

La imagen que la AMD desea proyectar a través de su página de Internet, es la de una institución solemnemente creada para promover el mejor conocimiento, investigación y enseñanza de la dermatología entre la comunidad médica. Sin embargo, un propósito importante es dar a conocer a la comunidad en general que existe una institución que reúne a la comunidad de médicos dermatólogos mejor preparados del país, y de esta forma, la gente pueda acudir a realizar cualquier tipo de consulta con dichos médicos; este beneficio es exclusivamente para los miembros de la AMD, ya que de otra forma, no podrán aparecer publicados en la página de Internet.

Los servicios interactivos que se prestarán en la página de Internet son:

- Búsqueda de dermatólogos miembros de la AMD por situación geográfica
- Libro de visitas
- Servicios exclusivos para la Mesa Directiva
    - Administración de comentarios registrados en el libro de visitas
    - Administración actualizaciones de la Base de Datos del Sistema de Información

Hablando de requerimientos era necesario cubrir los siguientes puntos para poder instaurar una nueva versión del sitio de Internet de la AMD:

- Contratar un plan de hospedaje acorde a las necesidades del sitio.
- Obtener la custodia completa de la administración del nombre del dominio.
- Seleccionar la mejor plataforma de desarrollo tomando como punto de partida el aspecto económico y administrativo.
- Seleccionar un administrador de bases de datos seguro para alojar los registros de los miembros en el sistema del sitio de Internet.

De esta forma se presenta un proyecto sencillo en su contenido pero con una finalidad muy profunda, que es primordialmente realizar un acercamiento a toda la comunidad dermatológica de México y proyectando la imagen de una institución vanguardista y de alto profesionalismo a nivel internacional.

3.3 Selección del Lenguaje de Programación para los Servicios Interactivos

Existen en el mercado muchos lenguajes de programación para páginas dinámicas de Internet; sin embargo, el requerimiento primordial para la AMD era seleccionar un lenguaje que por sus características fuera accesible en costos de mantenimiento e implementación, pero que tuviera un alto rendimiento en cuanto a sus funciones y desempeño en base a las necesidades del sitio, especialmente hablando en el acceso a bases de datos. Fue entonces que sin duda se eligió a PHP como la plataforma de desarrollo que cubriría todas las necesidades requeridas para este proyecto.

PHP es un lenguaje de programación basado en el modelo de preprocesamiento de páginas HTML. Cuando el preprocesador de PHP en su servidor Web nota una etiqueta de una página en lenguaje PHP, el ***engine*** convierte el código PHP y lo ejecuta creando una salida en formato HTML de forma dinámica.

PHP, acrónimo de "PHP: Hypertext Preprocessor", es un lenguaje ***"Open Source"*** interprete de alto nivel, especialmente pensado para desarrollos web y el cual puede ser incrustado en páginas HTML. La mayoría de su sintaxis es similar al lenguaje ***C***, ***Java*** y ***Perl***.

PHP resulta muy familiar a cualquier programador que haya trabajado con el mejor y más completo lenguaje de programación, el lenguaje C, e incluso existe una gran similitud sintáctica con lenguajes explícitamente creados para Internet como Java o Perl.

Hablando estrictamente de programación orientada a objetos, PHP toma prestada la estructura basada en objetos y conceptos de Java, así como del lenguaje C para los compilados de su última versión, la quinta.

La selección del lenguaje de programación se basó primordialmente en los costos de implementación, el acceso a diferentes fuentes de datos, la compatibilidad con servidores de hospedaje en México y por último, el conocimiento previo para el desarrollo de la página, ya que el proyecto tendría un plazo de alrededor de dos semanas en su programación e implementación.

3.4 PHP como Plataforma de Desarrollo

PHP es el heredero de un producto anterior, llamado PHP/FI. PHP/FI fue creado por ***Rasmus Lerdorf*** en 1995, inicialmente como un simple conjunto de ***scripts*** de Perl para controlar los accesos a su trabajo online. Llamó a ese conjunto de scripts ***'Personal Home Page Tools'***.

Según se requería más funcionalidad, Rasmus fue escribiendo una implementación C mucho mayor, que era capaz de comunicarse con bases de datos, y permitía a los usuarios desarrollar sencillas aplicaciones Web dinámicas. Rasmus eligió liberar el código fuente de PHP/FI para que cualquiera pudiese utilizarlo, así como arreglar errores y mejorar el código.

PHP/FI se mantuvo para páginas personales y como intérprete de formularios, incluía algunas de las funcionalidades básicas de PHP tal y como se conoce hoy. Tenía variables como las de Perl, interpretación automática de variables de formulario y sintaxis embebida HTML. La sintaxis por sí misma era similar a la de Perl, aunque mucho más limitada, simple y algo inconsistente.

Por 1997, PHP/FI 2.0, la segunda escritura de la implementación en C, tuvo un seguimiento estimado de varios miles de usuarios en todo el mundo, con aproximadamente 50.000 dominios informando que lo tenían instalado, sumando alrededor del 1% de los dominios de Internet. Mientras había mucha gente contribuyendo con bits de código a este proyecto, era todavía en su mayor parte el proyecto de una sola persona.

PHP/FI 2.0 no se liberó oficialmente hasta Noviembre de 1997, después de gastar la mayoría de su vida en desarrollos beta. Fue sucedido en breve tiempo por las primeras versiones alfa de PHP 3.0.

PHP 3.0 era la primera versión que se parecía fielmente al PHP tal y como se programa hoy en día. Fue creado por ***Andi Gutmans*** y ***Zeev Zuraski*** en 1997 reescribiéndolo completamente, después de que encontraran que PHP/FI 2.0 tenía pocas posibilidades para desarrollar una aplicación comercial que estaban desarrollando para un proyecto universitario. En un esfuerzo para cooperar y empezar a construir sobre la base de usuarios de PHP/FI existente, Andi, Rasmus y Zeev decidieron cooperar y anunciar PHP 3.0 como el sucesor oficial de PHP/FI 2.0, interrumpiéndose en su mayor parte el desarrollo de PHP/FI 2.0.

Una de las mejores características de PHP 3.0 era su gran extensibilidad. Además de proveer a los usuarios finales de una sólida infraestructura para muchísimas bases de datos, protocolos y APIs, las características de extensibilidad de PHP 3.0

atrajeron a docenas de desarrolladores a unirse y enviar nuevos módulos de extensión. Sin duda, ésta fue la clave del enorme éxito de PHP 3.0. Otras características clave introducidas en PHP 3.0 fueron el soporte de sintaxis orientado a objetos y una sintaxis de lenguaje mucho más potente y consistente.

Todo el nuevo lenguaje fue liberado bajo un nuevo nombre, que borraba la implicación de uso personal limitado que tenía el nombre PHP/FI 2.0. Se llamó 'PHP' a secas, con el significado de ser un acrónimo recursivo - PHP: Hypertext Preprocessor.

A finales de 1998, PHP creció hasta una base de instalación de decenas de millares de usuarios (estimados) y cientos de miles de sitios Web informando de su instalación. En su apogeo, PHP 3.0 estaba instalado en aproximadamente un 10% de los servidores Web en Internet.

PHP 3.0 se liberó oficialmente en Junio de 1998, después de haber gastado unos 9 meses en pruebas públicas.

En el invierno de 1998, poco después del lanzamiento oficial de PHP 3.0, Andi Gutmans y Zeev Suraski comenzaron a trabajar en la reescritura del núcleo de PHP. Los objetivos de diseño fueron mejorar la ejecución de aplicaciones complejas, y mejorar la modularidad del código base de PHP. Estas aplicaciones se hicieron posibles por las nuevas características de PHP 3.0 y el apoyo de una gran variedad de bases de datos y APIs de terceros, pero PHP 3.0 no fue diseñado para el mantenimiento tan complejo de aplicaciones eficientemente.

El nuevo motor, apodado ***'Motor Zend'*** (compreso de los apellidos, Zeev y Andi), alcanzó estos objetivos de diseño satisfactoriamente, y se introdujo por primera vez a mediados de 1999. PHP 4.0, basado en este motor, y acoplado con un gran rango de nuevas características adicionales, fue oficialmente liberado en Mayo del 2000, casi dos años después que su predecesor, PHP 3.0. Además de la mejora de ejecución de esta versión, PHP 4.0 incluía otras características clave como el soporte para la mayoría de los servidores Web, sesiones HTTP, *buffers* de salida, formas más seguras de controlar las entradas de usuario y muchas nuevas construcciones de lenguaje.

PHP 5 es actualmente la última versión liberada de PHP; se ha mejorado para esta versión el motor *Zend* para integrar, en su mayor propósito, la programación orientada a objetos.

Hoy, se estima que PHP es usado por cientos de miles de programadores y muchos millones de sitios informan que lo tienen instalado, sumando más del 20% de los dominios en Internet.

El equipo de desarrollo de PHP incluye docenas de programadores, así como otras docenas de personas trabajando en proyectos relacionados con PHP como ***PEAR*** y el proyecto de documentación.

No es el propósito de este trabajo aunar en comparativas con otros lenguajes de programación, sin embargo, es interesante ver una tabla de las mejores características de este lenguaje, y de esta forma, ver por que es la mejor elección de la AMD para sus desarrollos en Internet; para ello se presenta la ***tabla 3.1*** en donde es posible desglosar estas características en sus dos últimas versiones.

	PHP 4	PHP 5
Costo del Software	Gratis	Gratis
Costo de la Plataforma	Gratis	Gratis
Velocidad	Excelente	Excelente
Eficacia	Excelente	Excelente
Seguridad	Excelente	Excelente
Multi Plataforma (Sistema Operativo)	Excelente	Excelente
Multi Plataforma (Servidor de Internet)	Cualquiera	Cualquiera
Disponibilidad de Fuentes	Si	Si
Control de Excepciones	Si	Si
Programación Orientada a Objetos	No	Si
Acceso a Bases de Datos	• MySQL • PostrgreSQL • Oracle • Informix • Microsoft SQL Server • SQLite • SyBase • ODBC (Open DataBase Connectivity)	• MySQL • PostrgreSQL • Oracle • Informix • Microsoft SQL Server • SQLite • SyBase • ODBC (Open DataBase Connectivity)

Tabla 3.1 → Características del Lenguaje PHP

Como se ha visto, el ***Open Source*** no es sólo alguna antorcha filosófica para los programadores idealistas, o compañías que sólo quieren ahorrarse unos pesos en los costos de implementación de sistemas de computo; es una opción real y funcional que no sacrifica funcionalidad ni fortaleza en sus implementaciones.

3.5 Hospedaje y Administración del Nombre de Dominio

El hospedaje en Internet o ***Hosting*** se refiere a donde se van a alojar los scripts, documentos, imágenes, bases de datos, multimedia, etc. que conformen el sitio que se publicará en la red pública. Es posible contratar este servicio en diferentes empresas de México que básicamente dividen el servicio en *Servidores Dedicados* o *Servidores Compartidos*, dependiendo del servicio que la empresa necesite.

En el caso de la página de Internet de la AMD no es necesario contar con un servidor dedicado, ya que el proyecto, por su tamaño, no requiere una mayor cantidad de almacenamiento que cualquier plan básico ofrecido por los principales representantes de este servicio en nuestro país. En la ***tabla 3.2*** se pueden observar las empresas que fueron tomadas como candidatas para alojar el sitio de la AMD.

castleSYSTEMS TU DEPARTAMENTO DE SISTEMAS	Castle Systems México, S.A. de C.V.
elenlace.com.mx HOSTING I DOMINIOS I E-COMMERCE	elenlace.com, S.A. de C.V.
hostingbara	Hostingbara Com, S.A. de C.V.
internetworks	Internetworks.com.mx
www.SuEmpresa.com.mx	InterPlanet, S.A. de C.V.
prodigy hosting	Prodigy Hosting
mexis Seguridad Administrada	Servicios Administrados Mexis S.A. de C.V.
SITIOS REGIOS	Sitios Regios, S.A. de C.V.
Tu Seguro Servidor! Web Hosting - Co-Location	Teledesic Broadband Networks, S.A. de C.V.
terra	Terra Networks México
uservers web hosting	uServers Comunicaciones, S.C.
Web Comunicaciones	Web Comunicaciones

Tabla 3.2 → Principales empresas de hospedaje en Internet de México

Después de estudiar los diferentes planes de cada una de las empresas, se tomó la decisión de contratar el plan básico de ***Prodigy Hosting***, ya que cubre con los requerimientos necesarios para alojar el sitio de la AMD, aparte de contar con un buen costo de mantenimiento, proporcionar el servicio de bases de datos ***MySQL*** y scripts PHP sin costo adicional, un nivel de disponibilidad del 99.95%, soporte técnico telefónico y online las 24 horas de los 365 días del año; además de ofrecer la facilidad de pagar en el mismo recibo telefónico que el servicio de *Prodigy Infinitum*, con el cual ya se contaba en la oficina la AMD. En la ***tabla 3.3*** se puede observar el detalle de todos los servicios que ofrece este hospedaje.

Plan Básico Servidor Compartido		**Servicios Básicos**		**Herramientas de Administración del Sitio**	
Plataformas	Windows 2000 / UNIX	Almacenamiento de información	250 MB	Panel de Control	Si
Precio		Transferencia de información al mes	8,000 MB	FTP para subir archivos al servidor	Si
Renta Mensual	$199 pesos + IVA	Uso de Dominio Propio	Si	Soporte a Extensiones Front Page	Si
Costo de Instalación – Cargo Único	$ 350 pesos + IVA	**Correo Electrónico**		MySQL (base de datos)	Si
Seguridad, Respaldo y Desempeño		Cuentas de Correo Electrónico	10	Acceso directo a archivos de bitácora	Si
Respaldos diarios de información	Incluido	Almacenamiento por cuenta e-mail	25 MB	Perl / PHP / SSI	Si
Restauración de información	Disponible	Acceso POP3 / SMTP	Si	ASP (Active Server Pages)	Sólo Windows 2000
Servidores en espejo para mejorar desempeño	2 Servidores	Acceso vía WebMail (Acceso Web)	Si	**Soporte a Clientes**	
Soporte a SSL para seguridad de visitantes	Disponible	Creación de Alias	Si	Soporte telefónico 24x7	Incluido
		Reenvío ilimitado	Si	Soporte vía Correo Electrónico 24x7	Incluido
		Autorrespuesta	Si		

Tabla 3.3 → Servicios del plan básico de hospedaje de Prodigy Hosting

Un dominio o nombre de dominio es una forma simple de dirección de Internet que está formado por un conjunto de caracteres (letras, números, guiones). Es utilizado para localizar de una manera fácil los sitios en Internet ya que se puede asociar a la identidad de una persona, organización, empresa, idea, grupo, o a algún otro concepto.

Como ya se ha mencionado antes, la AMD deseaba tener el control absoluto de la administración del nombre de dominio ***www.amd.org.mx.*** Para ello terminó el contrato con la empresa "Mexis" y de esta forma dejó de utilizar una tercera empresa para la administración de su dominio. Entonces acudió a la empresa NIC de México para obtener los derechos por el dominio y cambiar los ***DNS's (Servidor de Nombres de Dominio)*** y direccionarlos a los DNS's de Triara, empresa que le presta este servicio a Prodigy.

Existen diferentes tipos de clasificaciones para los dominios en México, pero todas tienen terminación **.mx**, en la tabla 3.4 es posible ver las más comunes.

Tipos de nombres de dominio	Dirigido a:
.com.mx	Cualquier entidad o empresa
.net.mx	Proveedores de servicios de Internet localizados en México.
.org.mx	Organizaciones sin fines de lucro.
.edu.mx	Instituciones mexicanas de educación o investigación.
.gob.mx	Instituciones u oficinas del Gobierno Mexicano (Federal, Estatal o Local).

Tabla 3.4 → Tipos de dominios en México

La AMD entra por supuesto en la categoría de organización no lucrativa; por lo tanto no se presentó ningún problema para cambiar la propiedad de los derechos del dominio en NIC de México.

El ***Network Information Center-México (NIC-México)*** es la organización encargada de la administración del nombre de dominio territorial ***(ccTLD, country code Top Level Domain) .MX,*** el código de dos letras asignado a cada país según el ***ISO 3166.*** Entre sus funciones están el proveer los servicios de información y registro para .MX así como la asignación de direcciones de ***IP (Internet Protocol o Protocolo de Internet)*** y el mantenimiento de las bases de datos respectivas a cada recurso. En la ***tabla 3.5*** se pueden estudiar las diferentes tarifas que NIC México da por este servicio (precios válidos durante Junio del 2005).

CUOTAS	CANTIDAD A PAGAR	PERIODO DE COBERTURA	NOMBRES DE DOMINIO REQUERIDOS A SALDAR CUOTA	PERIODO DE GRACIA PARA EL PAGO
Cuota de Registro	35 USD	1 año	Solamente los nombres de dominio recién registrados bajo ***.com.mx, .net.mx*** y ***org.mx.***	Los siguientes 30 días después de la fecha de registro.
	66 USD	2 años		
	98 USD	3 años		
	128 USD	4 años		
	155 USD	5 años		
Cuota de Mantenimiento	35 USD	1 año	Aquellos nombres de dominio para los que expiró el PERIODO DE COBERTURA de la Cuota de Registro o de alguna Cuota de Mantenimiento.	Los últimos 30 días del PERIODO DE COBERTURA del pago anterior.
	66 USD	2 años		
	98 USD	3 años		
	128 USD	4 años		
	155 USD	5 años		

Tabla 3.5 → Tarifas de NIC México para el mantenimiento de dominios.

El registro de nombres de dominio en .edu.mx y .gob.mx no requieren saldar cuota de registro o mantenimiento. Todos los pagos que se realizaron para el trámite se llevaron a cabo por Internet y no representó ningún problema, incluso estos pagos se remitieron con factura desde Monterrey, dado que las oficinas de esta empresa están ubicadas en esta entidad.

Para la gestión del hospedaje, Prodigy proporciona todas las herramientas necesarias para administrar de forma remota los archivos que se alojarán en el sitio desde ***FTP (File Transfer Protocol o Protocolo de Transferencia de Archivos),*** también proporciona aplicaciones Web para la administración de las cuentas de correo y las bases de datos ***MySQL***, la cual es la que se utiliza para alojar los registros que son accedidos desde todos los contenidos dinámicos de la página, ya que es este administrador el que cubre de manera satisfactoria todas las necesidades de dichas aplicaciones.

3.6 MySQL como Administrador DBMS

Al dar de alta el hospedaje en Prodigy Hosting, se crea automáticamente una base de datos *MySQL* para generar las tablas que se requieran en el site. Especialmente este ***Sistema de Administración de Bases de Datos (DBMS)*** fue seleccionado porque el costo de este servicio se incluye dentro del plan básico de alojamiento de Prodigy.

Sin embargo, no se ha seleccionado este administrador únicamente por ser gratuito, también ofrece otras ventajas con respecto a otros administradores, las cuales, son requeridas para este proyecto:

- ✓ Originalmente MySQL es un administrador de bases de datos para Linux, sin embargo, tiene un comportamiento estable dentro de ambientes Windows.
- ✓ Licencia Gratis. Administración remota incluida en el servicio de Prodigy Hosting.
- ✓ Conexión directa a través de PHP.
- ✓ Alto desempeño en velocidad y seguridad.
- ✓ Permite almacenar todo tipo de registros, incluyendo imágenes y archivos de cualquier tipo y casi cualquier tamaño.
- ✓ Posee integridad referencial.
- ✓ Seguridad a nivel aplicación por esquemas de sistema de usuarios y permisos, los cuales ya están creados cuando se registra el hospedaje.

Prodigy Hosting ofrece a la AMD prácticamente todos los servicios necesarios para realizar de manera satisfactoria la publicación de su sitio de manera remota desde la oficina y sin ningún costo adicional. Por todas sus características y prestaciones, este servicio es la mejor opción para alojar la página de Internet.

3.7 Gestión y Arquitectura de Contenido

La gestión y la arquitectura de contenido de la página de Internet se refiere al conjunto de métodos y herramientas que permiten organizar los contenidos para ser encontrados y utilizados por los usuarios, de manera simple y directa.

La arquitectura de la información contenida en el sitio estará cumpliendo su objetivo cuando un usuario entre por primera vez al sitio y pueda reconocer a quién pertenece; lo pueda entender en forma rápida y sin esfuerzo, encontrar la información ofrecida fácilmente y, adicionalmente, entregar el beneficio de que quienes producen el sitio podrán ubicar la nueva información sin tener que crear nuevas estructuras y al mismo tiempo se tendrá la libertad de incorporar nuevas iniciativas al sitio sin tener que partir de cero.

3.7.1 Tipos de Audiencia

Se deben integrar al sitio diferentes tipos de accesos y herramientas para poder atender mejor a las diferentes tipos de audiencias que visiten la página. Para ello se deben de tomar acciones tomando en cuenta diferentes tipos de usuarios:

- **Capacidad Técnica:** existen diferentes niveles de usuarios, dependiendo de su experiencia y habilidades técnicas; es importante tomar en cuenta esto y crear accesos simples mediante enlaces sencillos de ubicar y otros más complejos para usuarios más experimentados.

- **Conocimiento de la Institución:** existen entre los visitantes usuarios que conocen a la institución y quienes no; por lo anterior, los primeros siempre sabrán donde buscar lo que necesitan usando la terminología y nombres comunes usados en la AMD; los segundos, en tanto, no entenderán nada de la nomenclatura interna y les será más difícil acceder a la información que se les ofrezca de esa manera; por lo tanto, es importante que todos los usuarios ubiquen las secciones que les interese de manera sencilla y rápida. También es importante reducir terminologías médicas complejas para no causar confusión en el caso de poder expresar las cosas de una forma más sencilla.

- **Necesidades de información:** los usuarios del sitio también se dividirán entre quienes llegan a buscar contenidos determinados y quienes sólo llegan a ver si existe algo que les pueda servir en lo que estén realizando. En último caso, a la AMD le interesa dar énfasis a la localización de los médicos dermatólogos que son miembros de la institución para que la gente pueda ubicarlos en sus diferentes entidades regionales o estados de la República Mexicana.

- **Ubicación geográfica:** dentro de la audiencia existen personas que accedan desde diferentes lugares dentro de la República Mexicana e incluso de Latinoamérica y el resto del Mundo, por lo que los contenidos deben responder también a esta diversidad de audiencia; sin embargo, a corto plazo, el interés de la AMD es dar todos los contenidos únicamente en Español.

Una de las formas más concretas de establecer los tipos de audiencia que tiene el sitio, es comenzar por investigar en la propia institución, para determinar a que tipo de gente se atiende sus diferentes rubros, y aunque la AMD en su mayor parte

se concentra en la comunidad dermatológica, uno de los principales objetivos del sitio es precisamente cambiar eso y poder llevar el mensaje de la institución a todos los mexicanos e incluso a cualquier persona en el mundo.

Una vez que se ha hecho este trabajo, es interesante hablar con los usuarios que llegan al sitio y plantearles preguntas simples y directas pero que en sus respuestas se conlleva un profundo mensaje de si el contenido del sitio esta llevando su mensaje de manera correcta:

- ¿A qué vino a la institución?
- ¿Tiene acceso a Internet?
- ¿Propio o a través de Café Internet?
- ¿Qué tipo de información en el sitio esperaba encontrar?
- ¿Qué le gustaría ver en el sitio Internet de esta institución?

De las respuestas que se obtengan, se podría hacer un muy buen resumen de tres elementos:

- Tipos de usuario que se podrían atender a través de Internet
- Expectativas de los usuarios respecto del sitio
- Necesidad de información de la institución

Es importante conocer todos estos puntos, sin embargo, la mayoría de la gente no se detendría a responder un cuestionario, y mucho menos proporcionar sus datos si no consideran que la imagen que se presenta en el sitio es de una institución que carece de seriedad; por lo tanto es importante presentar una imagen que exponga de manera real el objetivo de la AMD.

Para evitarle al usuario contestar una encuesta la solución expedita para este dilema es poner en línea un libro de visitas. Esta es una forma simple de poder crear un antecedente de las visitas que se están registrando en el sitio, y al mismo tiempo, darle al usuario la certeza de que sus comentarios serán revisados individualmente y así asegurar que su opinión será revisada y tomada en cuenta.

3.7.2 Definición de Contenidos

Una vez que se han identificado los objetivos del sitio, los tipos de audiencia y el lenguaje de programación, es posible hacer las definiciones concretas que permitan decidir que contenidos son los que va a tener el sitio.

Para identificar los contenidos, se debe utilizar como insumo los materiales que se han obtenido durante la planeación, ya que esta etapa ha girado en torno a las necesidades que se deben crear para los usuarios del sitio, y de esta forma, incluir en el contenido de la página la información necesaria para cubrir estas necesidades, dividiendo los diferentes módulos en páginas independientes, pero a la vez, enlazadas por un sistema de navegación coherente a la información que se está publicando. La información más importante y destacada dentro del sitio debe ser:

- **Información de la Institución:** mostrar la información completa referida a la misión y objetivos de la AMD, su Mesa Directiva, ubicación de las oficinas, los requisitos de ingreso, la información para contactar la AMD como teléfonos, correo electrónico, etc.

- **Servicios:** destacar las principales actividades que el usuario puede hacer en la página, y por lo consiguiente, en la institución; este punto se refiere a lo que el usuario puede realizar en los servicios interactivos del sitio, en especial, la búsqueda de dermatólogos por ubicación geográfica y el libro de visitas; eso por el lado de los usuarios, para la Mesa Directiva, los servicios de actualizaciones de la base de datos del sistema de información y la administración de los comentarios almacenados por el libro de visitas.

- **Novedades de la Institución:** esto se refiere a los últimos eventos, actividades, noticias, etc.

- **Laboratorios:** Es importante destacar la información publicada por cada laboratorio tipo "A" de la AMD, como sus anuncios publicitarios, medicamentos y apartados individuales a los que cada uno tiene derecho por estar inscritos como tales, y por lo consiguiente, los ***hipervínculos*** o ***links*** a sus diferentes sitios en Internet.

Viendo esta lista mínima (que deberá crecer en la medida de las necesidades de entrega de información de la institución), hay que hacer énfasis en que el interés de los contenidos variará si se trata de un usuario interno o externo.

Por ejemplo, si mira el sitio desde el punto de vista del usuario externo a la institución, lo que más le interesará será la información de los dermatólogos miembros de la AMD, seguida de la información de como hacer contacto con la institución. Si se mira desde el punto de vista del usuario interno, es decir, de los miembros de la AMD, lo más importante será la información de los próximos eventos, seguida por el de organigrama de la Mesa Directiva, las últimas noticias y los últimos productos anunciados por los laboratorios.

Es muy relevante que tanto los objetivos como la audiencia del sitio se hayan definido muy bien en forma previa, porque de lo contrario no habrá posibilidad de atender a ambos usuarios de manera adecuada.

Para que la definición de contenidos se lleve a cabo es importante agrupar y etiquetar el contenido para crear una metodología que permita ordenar los contenidos, agruparlos en conjuntos coherentes y darles nombres que los identifiquen.

La técnica que se utilizó para hacerlo fue crear unas pequeñas tarjetas de papel, en las cuales se anotaron las principales áreas de contenido que se revisaron anteriormente. Una vez hecho, se han puesto las tarjetas sobre una mesa y se agruparon hasta formar conjuntos de elementos coherentes entre ellos. Luego, a cada conjunto se le ha puesto un nombre (utilizando idealmente una sola palabra) que identifique a todos sus contenidos.

Con esas agrupaciones hechas, ya es posible formar los elementos adecuados para generar posteriormente un árbol de contenidos que, a su vez, permitirá hacer el sistema de navegación. Una vez que el proceso de etiquetado ha concluido, se realizaron comprobaciones empíricas de la validez de los nombres escogidos. Para ello, se ha requerido que las etiquetas elegidas sean mostradas a personas de diversos orígenes, algunas que tengan conocimiento del funcionamiento de la institución y otras que desconozcan sus actividades por completo; para descubrir estos nombres se formularon básicamente dos preguntas:

1. ¿Para usted, qué significa este nombre?
2. ¿Qué tipos de contenidos esperaría encontrar en esta área?

Con las respuestas obtenidas se ha podido juzgar que los nombres que se han usado son los más adecuados, o bien, que efectivamente inducen a identificar su contenido.

3.7.3 Requerimientos de Funcionalidad

Junto con la búsqueda de áreas de contenido que deberá tener el sitio, se debe trabajar también en la definición de lo que se busca que el sitio "haga", es decir, los tipos de interacción que se busca incluir. Dentro de los servicios interactivos más importantes que se deben incluir en la página se cuentan los siguientes:

- Búsqueda de Miembros de la AMD por ubicación geográfica.
- Libro de visitas con almacenamiento en la base de datos.
- Contador de visitas.
- Sistema de autenticación para el ingreso exclusivo de la mesa directiva.
- Administrador de los registros del libro de visitas exclusivamente para la mesa directiva.
- Sistema de descarga de archivos, incluyendo las actualizaciones de la base de datos del sistema de información para los miembros de la mesa directiva, mapas de contacto de la AMD y FMD, formatos de inscripción, etc. Todos los documentos para impresiones deberán estar en formatos *.doc y *.pdf.
- Carga dinámica y aleatoria de banners publicitarios de los laboratorios durante la navegación dentro del sitio.

Es importante que el sitio cuente con todos los servicios interactivos descritos como mínimos, para ofrecer una mejor experiencia al usuario que lo visita.

3.8 Definición de la Estructura del Sitio

En esta etapa se define la forma que tendrá el sitio en de la AMD. Esto implica definir varias áreas en concreto, a través de las cuales se definirá la estructura del sitio, como el árbol de contenidos y los sistemas de navegación que se ofrecerán a los usuarios para que naveguen a través de sus contenidos.

3.8.1 Estructura del Sitio

Cuando se habla de la estructura de un sitio, se está refiriendo básicamente a cuál será el desplazamiento que tendrá un usuario a través sitio. En este momento es posible mostrar la distribución de los elementos de información e interacción que tendrá el usuario. Durante esta etapa no se han incluido elementos de diseño, ya que sólo es posible estudiar la estructura concreta del sitio, sin que intervengan aún consideraciones estéticas, las cuales se mantendrán como un estándar para todas las páginas. En la ***figura 3.1*** es posible ver el árbol funcional de navegación completo del sitio de Internet.

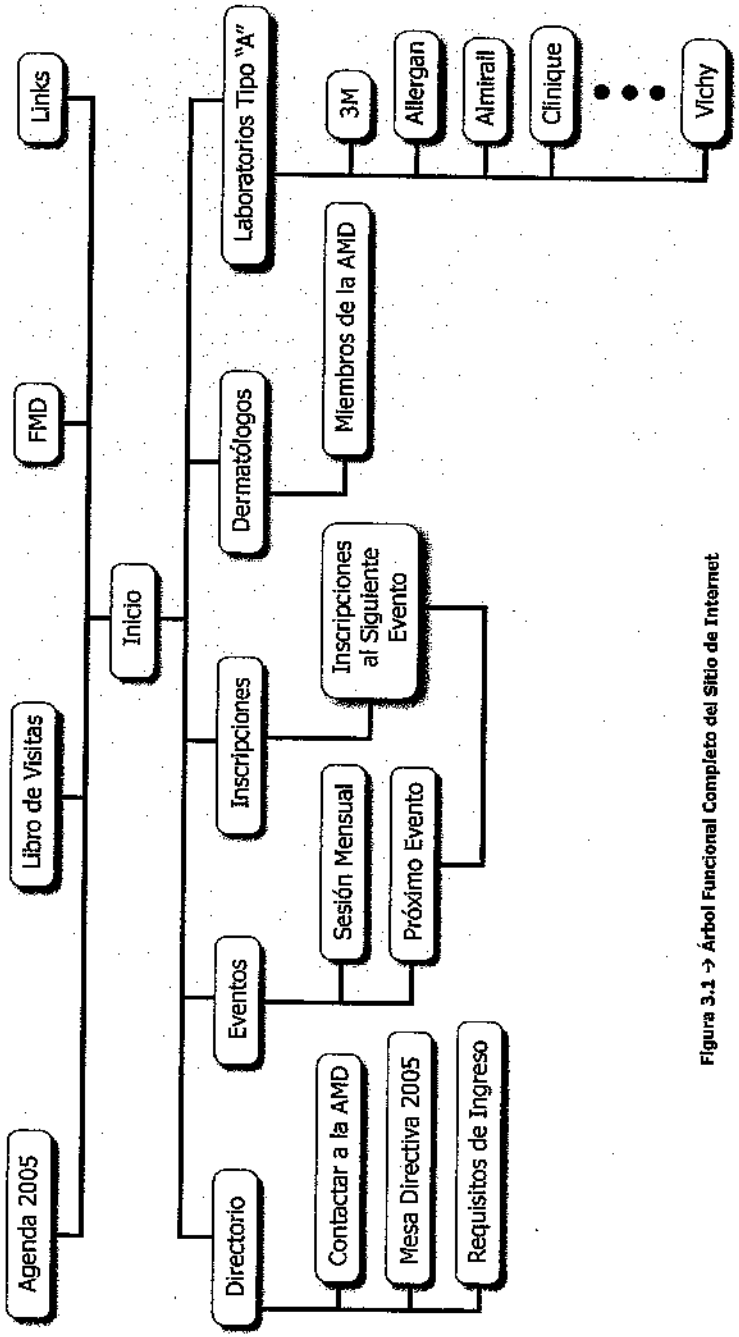


Figura 3.1 -> Árbol Funcional Completo del Sitio de Internet

3.8.2 Árbol Funcional

El árbol funcional de la ***figura 3.1*** agrupa los contenidos de acuerdo a las tareas que el usuario puede realizar dentro del sitio; se han creado la menor cantidad posible de secciones con el fin de concentrar las acciones del usuario en pocas áreas; hay que considerar que cada una de las áreas a integrar en el árbol funcional requerirá de mantenimiento posterior, lo que significa que el mantenimiento será menor y más rápido.

Es importante que el usuario esté siempre a menos de tres clicks del contenido que anda buscando. Por ello no se han creado más de tres niveles de acceso; esto significa que como máximo los contenidos deberán poseer una portada, una portadilla de sección y propiamente el contenido.

Los contenidos de cada una de las secciones deben estar relacionados entre si, y estas relaciones deberán estar presentes en todo el sitio. Por ello se debe crear un sistema de navegación que se incluya como parte de la plantilla para todas las secciones del sitio.

3.8.3 Elementos del Árbol Funcional

En este apartado se analizará el contenido en cada uno de las secciones distribuidas a lo largo de las ramas del árbol y la información principal que estarán publicadas en cada una de las secciones.

- **Inicio:** En esta sección estarán publicadas las últimas noticias acerca de los próximos eventos a realizar por la AMD, también se podrá encontrar a un costado del listado de la información antes mencionada, la lista completa de la actual Mesa Directiva. También deberán presentarse una pequeña introducción de cada una de las secciones más importantes del sitio y todos los hipervínculos necesarios para que el usuario acceda a las diferentes secciones descritas a lo largo de esta pantalla.

- **Contactar a la Academia:** En esta sección deberá estar publicada toda la información referente a los medios de contacto con la AMD como:

  - ✓ Dirección física
  - ✓ Dirección electrónica (e-mail)
  - ✓ Teléfono de la oficina
  - ✓ Fax
  - ✓ Mapa completo de la situación geográfica de la oficina

- **Mesa Directiva 2005:** En este apartado estará publicada la lista completa y detallada de los nombres y diferentes cargos de la Mesa Directiva de la AMD y el su periodo de gobierno.

- **Requisitos de Ingreso:** Esta sección publicará la información completa de los requisitos de inscripción a la AMD exclusivamente para médicos especialistas en todas las ramas dermatológicas.

- **Próximo Evento:** En esta sección estará publicada toda la información del siguiente magno evento de la AMD, esta información incluye:
  - ❖ Lugar y fechas del evento
  - ❖ Imagen o póster del evento
  - ❖ Mensaje a los asistentes del presidente en cargo de la AMD
  - ❖ Programa preliminar o programa final del Evento
  - ❖ Desglose completo del programa y los diferentes simposios, exposiciones, talleres, eventos especiales, etc. de las diferentes conferencias que se darán a lo largo del evento, incluyendo información como:
    - ✓ Fecha y hora
    - ✓ Título y tema
    - ✓ Nombre del médico expositor (En el caso de ser profesor extranjero, incluir currículum y fotografía)
  - ❖ Lista de los profesores extranjeros invitados
  - ❖ Lista de los profesores nacionales
  - ❖ Lista de eventos sociales
  - ❖ Lista de eventos especiales

- **Sesión Mensual:** En este apartado se publicará la información de la siguiente sesión mensual de la AMD, incluyendo la siguiente información:
  - ❖ Lugar y fecha del evento
  - ❖ Mes al que pertenece el evento
  - ❖ Título del evento
  - ❖ Bajo que persona u hospital esta a cargo el evento (si es el caso)
  - ❖ A que médico se dedica o se hace homenaje la sesión (si es el caso)
  - ❖ Lista de la Mesa Directiva
  - ❖ Programa de la sesión mensual y su respectivo desglose incluyendo:
    - ✓ Fecha y Hora
    - ✓ Título y tema
    - ✓ Nombre del médico expositor
  - ❖ Información y fecha del próximo evento
  - ❖ Información y fecha de la próxima sesión mensual

- **Inscripciones:** Toda la información referente a las inscripciones para el próximo evento como:
  - Nombre, fecha y lugar del evento en cuestión
  - Imagen o póster del evento
  - Notas y requisitos para obtener diploma o reconocimiento de asistencia
  - Costos de inscripción para miembros, residentes y asistentes en general
  - Costos de talleres
  - Formatos de inscripción al evento (En formato *doc y *pdf)
  - Información completa de la cuenta para realizar los depósitos correspondientes
  - Costos de hospedaje en los diferentes hoteles del evento

- **Dermatólogos:** Este apartado es una de las secciones interactivas del sitio, como ya se ha expuesto anteriormente, se podrá buscar a los diferentes miembros de la AMD por su ubicación geográfica; para ello se han creado dos opciones para las búsquedas:
  - Interior de la República
  - Distrito Federal

  Para esta sección se deberán incluir mapas gráficos interactivos con los cuales el usuario pueda ubicar su estado o delegación y simplemente al dar un click aparezca una lista con el nombre del dermatólogo y los teléfonos de sus respectivos consultorios. Adicionalmente se agregará una lista desplegable con los nombres de los diferentes estados o delegaciones en el caso de que el usuario no ubique su delegación con exactitud.

- **Agenda 2005:** En este apartado se publicarán las fechas y la información de todos los eventos confirmados que realizará la AMD a lo largo del año para que los diferentes médicos aparten dichas fechas en sus agendas.

- **Laboratorios Tipo "A":** Aquí se encontrará un listado de todos los laboratorios suscritos como tipo "A" de la AMD, este listado estará conformado por los diferentes logotipos de los laboratorios y estos mismos direccionarán a las publicaciones que cada uno tendrá dentro del sitio de la AMD. Esto quiere decir que cada uno de los laboratorios tiene derecho a publicar una pantalla con sus diferentes productos acoplados a la plantilla de la página de Internet. Estas publicaciones no las realiza la academia, sino cada uno de los laboratorios; para acceder a estos apartados el usuario podrá dar click en la imagen del laboratorio incrustada en esta sección o dando click en el *banner* publicitario del mismo laboratorio mostrado a lo largo de la navegación del sitio.

Es importante recordar que los banners publicitarios de los laboratorios deberán aparecer de forma aleatoria en un apartado especial al costado de todas las pantallas del sitio, de esta forma, el usuario visualizará los diferentes productos anunciados en la página. Si el laboratorio no cuenta con una publicación dentro del sitio de la AMD, el logotipo del laboratorio direccionará a su página de Internet y este hipervínculo se abrirá en una página independiente del navegador en el que se encuentre navegando en la página de la AMD.

- **Libro de Visitas:** El libro de visitas es un apartado en el cual el usuario podrá dejar sus comentarios y sugerencias en un formulario que guardará dicha información en la base de datos de la página. El libro de visitas pedirá al usuario como datos requeridos:
  - ✓ Nombre
  - ✓ Correo Electrónico
  - ✓ Comentarios

- **FMD:** En esta sección estará publicada la lista del comité ejecutivo de la Fundación Mexicana para la Dermatología, A.C., Así como toda la información para contactar a la FMD como:
  - ✓ Dirección física
  - ✓ Dirección electrónica (e-mail)
  - ✓ Teléfono de la oficina
  - ✓ Fax
  - ✓ Mapa completo de la situación geográfica de la oficina

- **Links:** En esta sección el usuario podrá encontrar un listado de vínculos en Internet de páginas relacionadas con la AMD como:
  - ✓ Academia Americana de Dermatología
  - ✓ Laboratorios tipo "A" de la AMD

También es importante destacar que existirá un sistema de autenticación para el acceso exclusivo de la mesa directiva para acceder a un administrador de los comentarios vaciados desde el libro de visitas donde los directivos podrán verlos, eliminarlos o enviar un correo electrónico a la persona que lo haya escrito, adicionalmente de poder bajar las actualizaciones de la base de datos para el sistema de información. Este archivo estará compreso en un programa ejecutable que requerirá un *password* que se le proporcionará al directivo de manera independiente de la página de Internet.

3.8.4 Sistema de Navegación

Una vez que se tiene la estructura completa del árbol funcional, la tarea siguiente consiste en generar el sistema de acceso a dichos contenidos en el sitio web. A través de este sistema, los usuarios pueden avanzar por sus diferentes áreas, sin perderse.

El sistema de navegación debe estar compuesto por elementos concretos, tales como menús, hipervínculos, botones y otros elementos que deben ser claramente distinguibles dentro de la interfaz. Los sistemas que se han seleccionado para la navegabilidad representan claramente la función para la que fueron designados y no dejan lugar a dudas sobre su función ni sobre la acción que desarrollarán al ser usados. Adicionalmente, es muy importante que las palabras que se han seleccionado para indicar tales acciones, deban ser claras y precisas. En este sentido, si un botón o hipervínculo necesita ser explicado, será mejor desecharlo y buscar otra solución.

En el sentido contextual es importante realizar una estructura de como se presentarán los elementos antes mencionados, utilizando para ellos objetos basados en texto, gráficos o bien de entorno. Los elementos relevantes en este caso, son todos aquellos que permiten mostrar la navegación en la pantalla; entre ellos, un menú gráfico que deberá presentarse en todas las secciones del sitio y que representa de forma clara todas las opciones que puede realizar el usuario durante su visita.

Al generar el sistema de navegación, se deben tener en cuenta las siguientes características:

- **Consistente:** el sistema debe ser idéntico o similar en todo el sitio, en lo referido a su ubicación y disposición en las páginas.
- **Uniforme:** el sistema debe utilizar similares términos con el fin de que el usuario que lo vea en las páginas, confíe en que sus opciones llevan siempre hacia los mismos lugares dentro del sitio.
- **Visible:** el sistema debe distinguirse claramente dentro del sitio, con el fin de que el usuario cuente con él, como si se tratara de una guía permanente en el área en que se encuentre del sitio.

3.8.5 Elementos del Sistema de Navegación

Entre los elementos más importantes que conforman el sistema de navegación del sitio se cuentan los siguientes:

Barra Corporativa: Ubicada en la parte superior, muestra la imagen corporativa de la AMD, el nombre completo y muestra, en forma de ***scroll*** horizontal, la meta y objetivo de la institución.

Menú General: siempre presente en todo el sitio, permite el acceso a cada una de las áreas del sitio.

Contenido: Muestra en la parte central la información de la sección en la que se encuentra actualmente.

Barra Publicitaria: Muestra los banners publicitarios de los laboratorios ubicados aleatoriamente en el costado derecho durante la navegación dentro del sitio.

Pié de Página: Ubicado en la parte inferior de cada página, indica el nombre de la institución, teléfonos, dirección física y correo electrónico.

Barra de Acceso: Muestra en un listado horizontal debajo del pié de página todas las secciones a las que puede acceder el usuario utilizando texto simple en forma de hipervínculos sencillos.

El siguiente paso es crear el diseño gráfico de las páginas como una estructura uniforme y acorde a los elementos de navegación anteriormente mencionados.

3.9 Diseño Gráfico

Una vez que se han aprobado todos los elementos para la navegación del sitio, se ha creado el boceto final del diseño de las páginas, implementando la navegabilidad concebida anteriormente, los elementos gráficos como logotipos e imágenes, los textos y el contenido en general. Todos estos elementos están distribuidos como se muestra en la ***figura 3.2:***

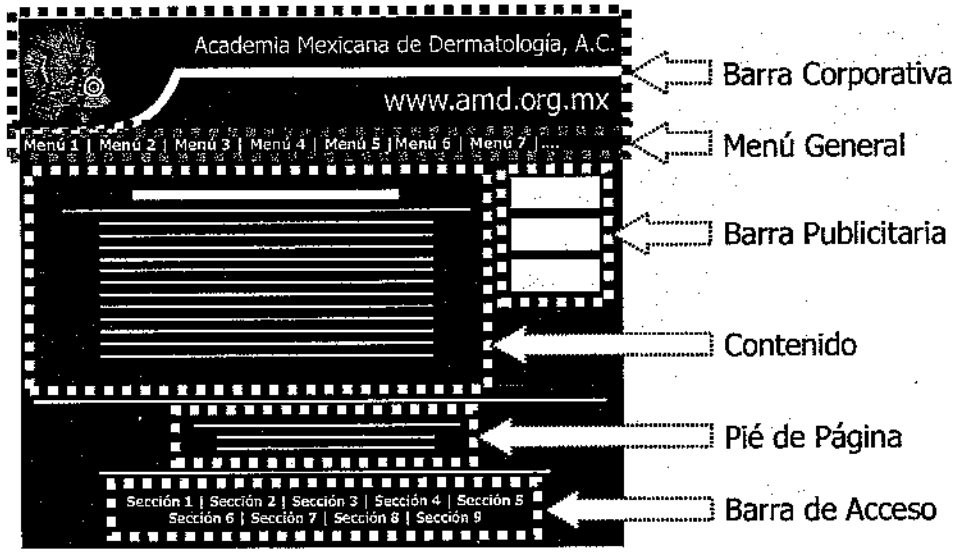


Figura 3.2 → Boceto Final del Diseño Gráfico

Esta plantilla se toma como un objeto lineal para cada una de las pantallas del sitio, ya que la ubicación de estos elementos estarán siempre distribuidos en el mismo orden. Los elementos de esta plantilla tienen esta distribución para que el usuario ubique cada una de las secciones de la página siempre en el mismo orden lógico.

Todas las secciones deberán tener un tamaño máximo de 800 ***pixeles*** de ancho, para que todas las secciones se ajusten automáticamente a cualquier resolución de la pantalla. Este pequeño truco es usado en muchas de las páginas comerciales en el mundo, y es sumamente útil si se desea ahorrar espacio en el alojamiento de las pantallas, es decir, es posible crear pantallas para diferentes resoluciones y diferentes navegadores comerciales y de esta forma direccionar al usuario a diferentes pantallas, dependiendo obviamente del sistema operativo que esté

usando, la resolución de su pantalla y el navegador con el que esté accesando a la página de Internet.

Sin embargo, de lo antes mencionado se deduce que se deben crear pantallas para cada una de estos posibles casos, y esto se deriva en espacio que para la AMD es innecesario, de tal forma que si es posible crear una página que pueda ser accedida desde cualquier navegador, sistema operativo y desde cualquier resolución que utilice el usuario, esto reducirá costos de mantenimiento y espacio en el servidor donde se aloja el sitio. En la ***figura 3.3*** se puede observar una maqueta del tamaño máximo que deben adoptar todas las pantallas de la página.

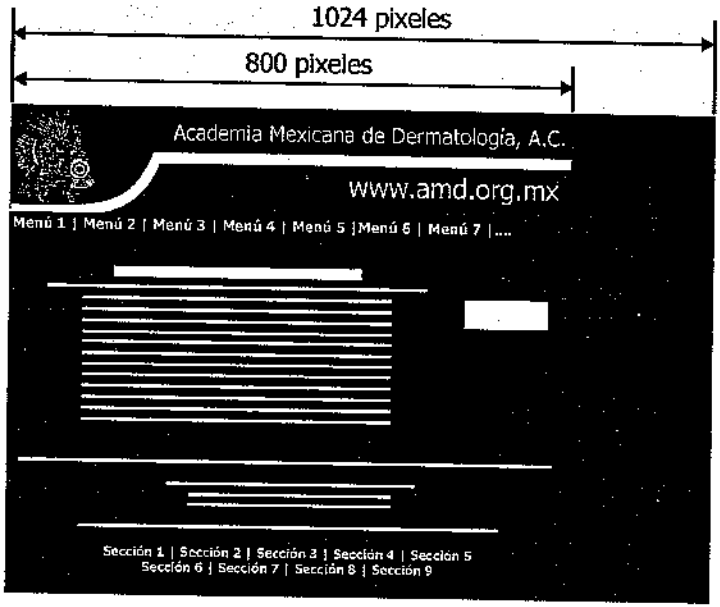
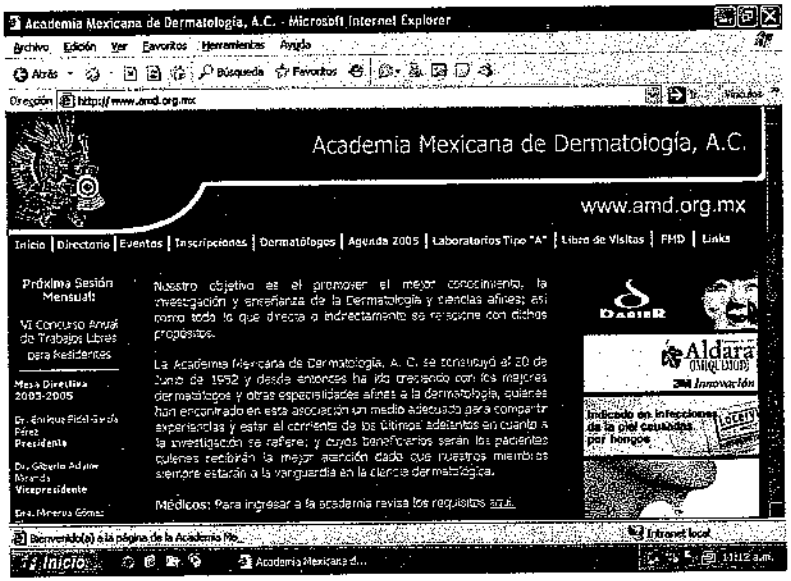


Figura 3.3 → Maqueta del tamaño máximo de las pantallas del sitio

En la ***figura 3.4*** se puede observar como es que la pantalla de la página de Internet se ajusta automáticamente a dos diferentes resoluciones.



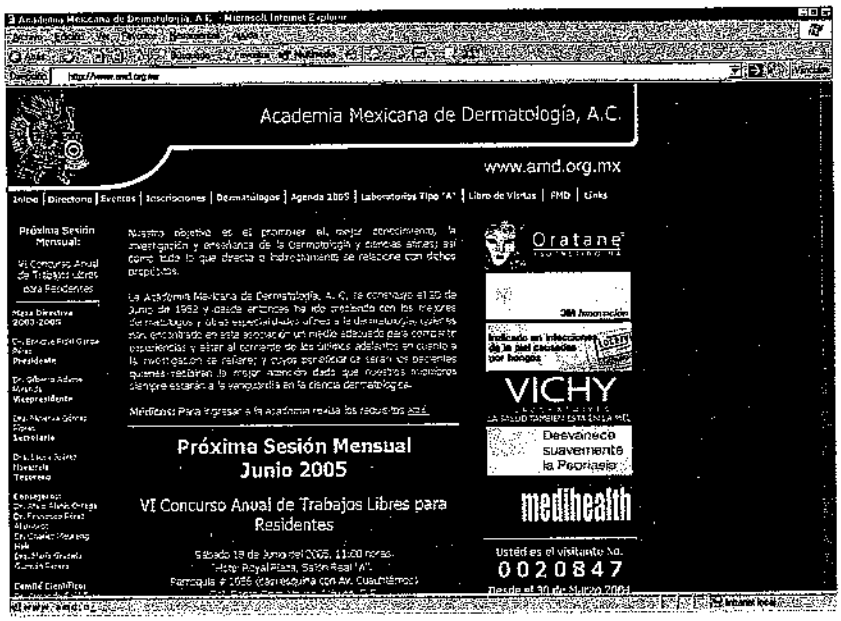


Figura 3.4 → *Arriba:* Pantalla utilizando una resolución de 800x600 pixeles.
***Abajo:* Pantalla utilizando una resolución de 1024x768 pixeles**

Como la página de la AMD es relativamente chica no es necesario cambiar prácticamente en nada la plantilla que se ha creado y, utilizando este diseño, es posible crear pantallas idénticas en su distribución y que se ajusten automáticamente a cualquier resolución. De esta forma el usuario ubicará todas las partes en un mismo lugar y podrá desplazarse a otras secciones del sitio de forma rápida y sencilla.

3.10 Interoperatividad y Compatibilidad en Diferentes Navegadores

Dado que el sitio web de la AMD puede ser accedido sin problemas desde computadoras que utilizan diferentes sistemas operativos, se debe cuidar el aspecto de la diversidad de navegadores de Internet.

Para ello se ha pensado en los mayores contratiempos que causaría al navegar desde los diferentes programas de acceso a Internet; para ello se han tomado en cuenta los siguientes puntos en la creación de los programas del sitio:

- ✓ Utilizar código ***HTML (Hipper Text Markup Language o Lenguaje de Marcado de Hipertexto)*** estándar, no mejorado para un navegador en especial.

- ✓ Probar el sitio con las versiones para diferentes sistemas operativos de diversos navegadores o browsers como Internet Explorer, Netscape Communicator, Mozilla, y Opera.

- ✓ Utilizar lo menos posibles ***scripts*** especiales como ***JavaScript,*** y si se utilizan estandarizar su uso para todas las versiones de navegadores.

- ✓ Erradicar el uso de ***frames*** o marcos en el sitio.

- ✓ Optimizar el peso de las imágenes cambiando el formato original a uno compreso como GIF o JPEG. Cuando esto no sea posible hacerlo por su tamaño o reduciendo el número de colores disponibles y la resolución de las mismas (72 dpi es la norma).

- ✓ Ofrecer los ***Plug-ins*** *(programas visualizadores especiales)* necesarios para el acceso y la presentación correcta de todos los elementos multimedia del sitio. Cuando el usuario accede al sitio de la AMD, esta cuenta con varios elementos que utilizan un *Plug-in* llamado ***"flash"*** para mostrar objetos como la barra corporativa, el menú y los mapas para la búsqueda de los miembros de la AMD.

Entonces pues, el acceso a la página de la AMD requieren el uso de *Pulg-ins* para navegar a través de ella; para ello se cuenta con un programa o módulo incrustado en cada una de las páginas del sitio que requiere y baja automáticamente la última versión del plug-in para visualizar elementos Flash, detectando automáticamente el navegador que este utilizando actualmente el usuario, incluyendo los diferentes sistemas operativos en el que se va a instalar.

Para verificar el funcionamiento correcto de los puntos anteriores, se han realizado las pruebas de usabilidad bajo los principales navegadores y sus diferentes versiones con el fin de establecer si cumplen o no con las necesidades de cada uno de ellos; y de acuerdo a este resultado se ha concluido que la página de Internet de la AMD puede ser accedida desde casi cualquiera de los más populares navegadores de Internet.

3.11 Posicionamiento en Motores de Búsqueda

Es importante tener presencia en los principales buscadores, ya que son estos en los que la mayoría de las personas realizan las búsquedas de un sitio cuando desconocen la dirección electrónica o que simplemente son usuarios inexpertos y no utilizan la barra de dirección para acceder directamente al sitio de su preferencia; por ello se ha posicionado el sitio de la AMD en los principales buscadores de México como son:

- Google
- Yahoo!
- Altavista
- T1msn
- Terra

Los buscadores son servidores Web que tienen acceso a una extensa base de datos sobre recursos disponibles en la propia Web. El usuario conecta con un buscador e indica unas pocas palabras representativas del tema sobre el que está buscando información y que se utilizan como clave de búsqueda. Como resultado de la búsqueda se muestra al usuario una lista con enlaces a páginas Web en cuya descripción o contenidos aparecen las palabras claves suministradas. Cuando un usuario busque en cualquiera de estos sitios palabras como "Dermatología" o especifique el nombre de "Academia Mexicana de Dermatología" podrá ver en los primeros resultados el sitio de la AMD. En la ***figura 3.5*** se puede observar el resultado de algunos de los buscadores antes mencionados.

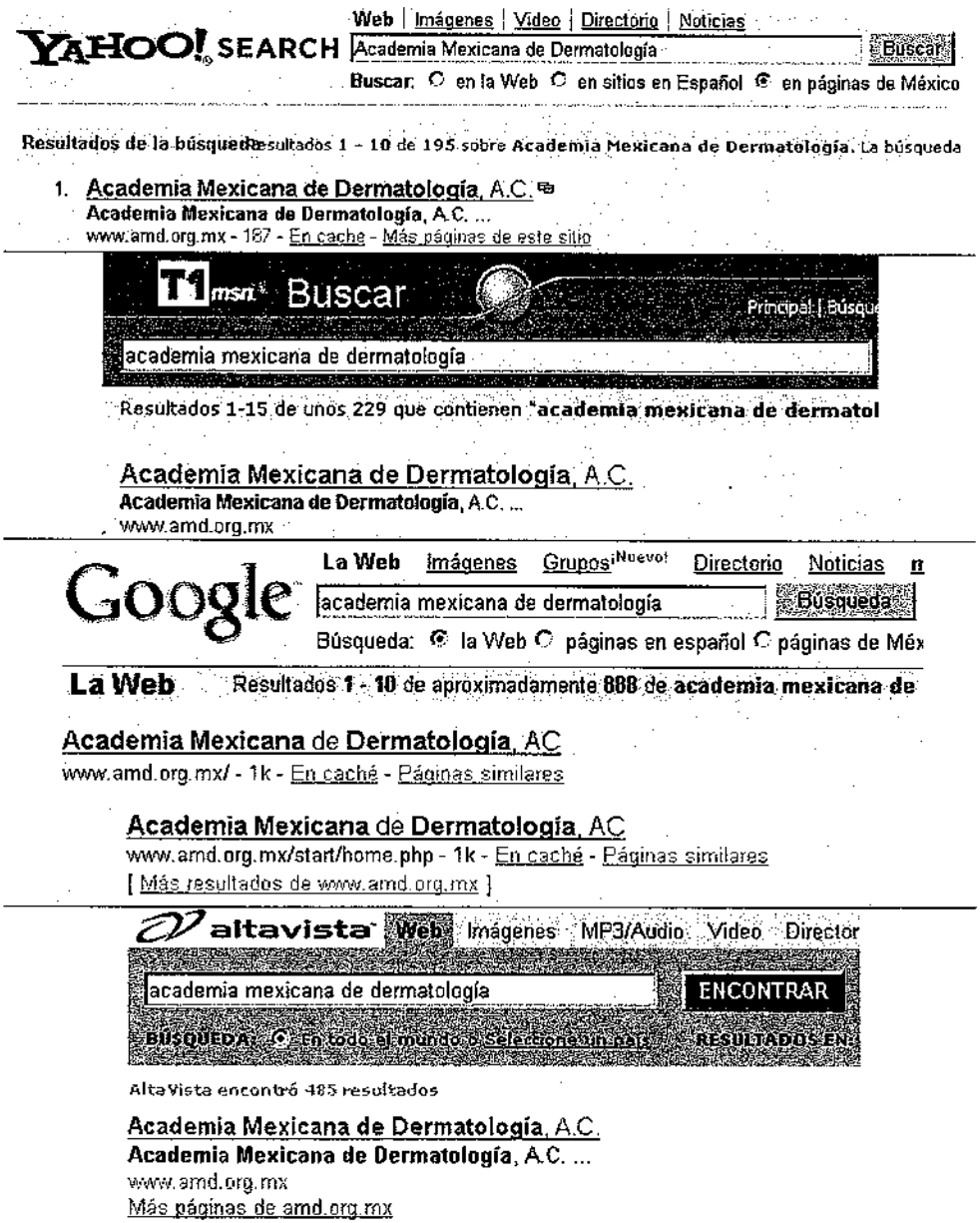


Figura 3.5 → Presencia en los principales buscadores

Otra métrica relevante para saber el grado de efectividad que está teniendo el Sitio Web, consiste en revisar periódicamente su presencia a través de los buscadores de Internet más populares, esto es porque estos sitios son los que concentran el mayor tráfico y, por lo tanto, que el Sitio Web aparezca en ellos garantizará que los usuarios que estén buscando la institución la podrán encontrar.

3.12 Mantenimiento y Actualización de Contenidos

Un tema que suele ser dejado de lado pero que es determinante, es la frecuencia con la que se actualizarán los contenidos de un Sitio Web. Lo más importante que se debe tener en cuenta, es que por tratarse de una herramienta de comunicación, el Sitio Web requiere de una actualización permanente, con el fin de dar cuenta a sus usuarios de que siempre hay información de interés en el sitio, gracias a lo cual gana en credibilidad.

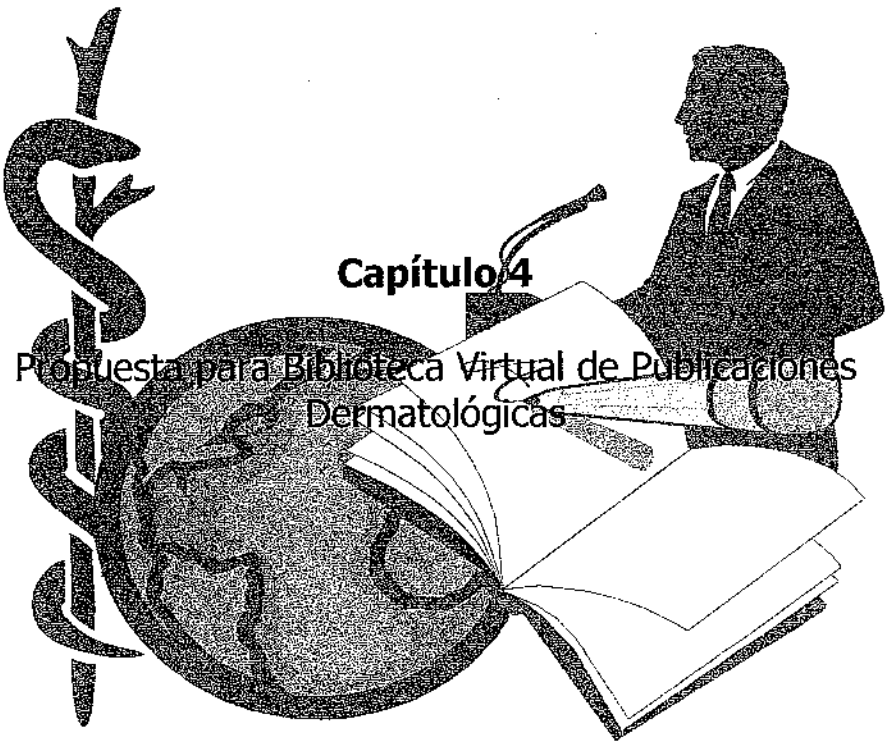
En el caso del sitio de la AMD las actualizaciones se llevan a cabo al menos una vez al mes, ya que es durante este periodo en que se realizan cambios en los programas como los próximos eventos y las diferentes sesiones mensuales; sin embargo, cuando se requiere de una publicación que así lo amerite, las actualizaciones suelen ser más frecuentes.

Adicionalmente, es importante considerar que el sitio debe ser una herramienta comunicativa de la institución a la que pertenece y por ello, debe seguir sus orientaciones estratégicas y programáticas. Por lo anterior, a través del sitio web se debe ir incorporando todo aquello que esté en sintonía con lo que el servicio esté realizando, es decir, cuando se realizan cambios en la Base de Datos del sistema de información, esta también es migrada al sistema de administración *MySQL* e inmediatamente actualizada en el servidor de Prodigy, ya que esto refleja la constante adición de médicos especialistas a la AMD.

3.13 Respaldo y Recuperación

Para respaldar la información del sitio web de la AMD se requiere copiar el contenido completo del sistema (bases de datos, programación, scripts, imágenes, etc.) a un medio externo al servidor de Prodigy, y es que aunque el servicio básico de hospedaje cuente con respaldos diarios y recuperación de la información publicada, nunca será suficiente ante un ataque inminente. De tal forma que para la AMD la misma oficina es el lugar más seguro y es el único lugar que permite la recuperación segura, rápida y eficaz.

En este sentido, hay que preocuparse no sólo de probar la confiabilidad del sistema al momento de respaldar, sino también para la acción de recuperar y volver a instalar lo respaldado. Por ello se ha creado un archivo de todas las versiones publicadas en Internet y se han respaldado en diferentes equipos así como en diferentes discos ópticos para tener un segundo respaldo y contar con la seguridad y la certeza de que el funcionamiento del sitio permanecerá por siempre.



Capítulo 4

Propuesta para Biblioteca Virtual de Publicaciones Dermatológicas

Existen sistemas en Internet dedicados a publicar documentación de prácticamente cualquier tema del conocimiento humano, sin embargo, existen muchos sitios que lucran con este conocimiento, por así decirlo; ya no es ajeno a la gente común que existen documentos en Internet, en formatos digitales con contenidos exclusivos para la red mundial. Estos documentos solo pueden ser accedidos si se cubren cuotas que van desde un simple mensaje ***SMS*** desde un celular hasta precios considerablemente altos por su exclusividad.

Por otra parte existe una gran comunidad en Internet que están conformando una sociedad virtual que intenta aumentar sus conocimientos accediendo a sitios que están dedicados a gente común que realiza desde simples pasatiempos, hasta los más altos ejecutivos que deciden el destino del mundo con un simple click del ratón de sus computadoras. El conocimiento esta en la red, y está al alcance de cualquier persona que desee aprehender más acerca de cualquier rama de las ciencias desde sus hogares u oficinas.

Así se han creado foros de discusión y sistemas para comunidades en la red mundial desde sus inicios, y estas comunidades han permanecido y crecido en proporciones que nadie habría podido imaginar hace unos cuantos años, cuando era inaudito pensar que hasta las tiendas y organizaciones más pequeñas contarían con presencia en Internet para promocionar sus productos. También en estas comunidades se hallan los más especializados profesionistas intentando compartir al mundo sus conocimientos y experiencias para crear de nuestro mundo un lugar mejor para vivir.

Parte del juramento hipocrático, atribuido a Hipócrates en el año 350 A.C. cita:

"Trataré al que me haya enseñado este arte como a mis progenitores, y compartiré mi vida con él, y le haré partícipe, si me lo pide, y de todo cuanto le fuere necesario, y consideraré a sus descendientes como a hermanos varones, y les enseñaré este arte, si desean aprenderlo, sin remuneración ni contrato.

Y haré partícipes de los preceptos y de las lecciones orales y de todo otro medio de aprendizaje no sólo a mis hijos, sino también a los de quien me haya enseñado y a los discípulos inscritos y ligados por juramento según la norma médica, pero a nadie más."

Es interesante considerar el espacio que menciona la importancia de compartir el conocimiento con todos los practicantes de cualquier rama de la medicina; el conocimiento en pos de la ciencia no debe ser objeto de lucro ni de beneficio

personal, la sabiduría de la humanidad esta basada en el conocimiento empírico y científico que la humanidad ha dado a conocer a través de libros y de generación en generación hasta nuestra actualidad.

Las tecnologías médicas y en general, de todas las ciencias, han alcanzado una madurez que ha beneficiado a miles de personas en todo el mundo, por lo tanto, los nuevos conocimientos no deben quedar en una sola persona ni en un grupo de gente que únicamente busque su beneficio personal. Desgraciadamente no existe un juramento hipocrático para todas las carreras, y ojala este no fuera olvidado a lo largo de una vida.

El sistema podrá ser utilizado por todos los médicos que se especializan en la rama de la Dermatología y que desean compartir sus conocimientos, experiencias y su sabiduría con toda la comunidad dermatológica del país e inclusive, a todo el planeta Tierra.

La finalidad esencial de este capítulo es presentar una propuesta para un sistema independiente para publicaciones dermatológicas que hayan sido evaluadas por otros médicos de la misma especialidad y puestas a juicio ante un juzgado de dermatólogos para que estas publicaciones sean corregidas y aumentadas para poder ser divulgadas a todos los médicos que deseen ser coparticipes de este proyecto, aumentar sus conocimientos o simplemente aprehender algo nuevo, pero al mismo tiempo, compartir sus ideas y crear una comunidad dermatológica unida por la página de Internet de la AMD.

La creación de este soporte documental da hincapié a una aplicación web basada en la idea de un foro abierto y funcional que se integra como una biblioteca virtual de publicaciones dermatológicas; el análisis y evaluación preliminar del sistema permitirá a la mesa directiva de la AMD tomar la decisión de crear dicho programa e implementar lo expuesto a lo largo de este capítulo como un componente funcional dentro de su sitio de internet.

4.1 Antecedentes Históricos

Un aspecto clave del rápido crecimiento de Internet ha sido el acceso libre y abierto a los documentos básicos. En los comienzos de ***ARPANET*** en la comunidad de investigación universitaria se estimuló la tradición académica de la publicación abierta de ideas y resultados. Sin embargo, el ciclo normal de la publicación académica tradicional era demasiado formal y lento para el intercambio dinámico de ideas, aspecto esencial para crear redes informáticas.

En 1969 *Steve Crockett's* dio un paso clave al establecer la serie de notas ***RFC (Request For Comments o Petición de Comentarios)***. Estos memorándums pretendieron ser una vía informal y de distribución rápida para compartir ideas con otros investigadores en redes. Al principio, las *RFC* fueron impresas en papel y distribuidas vía correo postal. Pero cuando el ***FTP (File Transfer Protocol o Protocolo de Transferencia de Archivos)*** empezó a usarse, las *RFC* se convirtieron en ficheros difundidos online a los que se accedía vía FTP. Hoy en día, desde luego, están disponibles en el World Wide Web en decenas de sitios en todo el mundo y en muchos idiomas diferentes.

El efecto de las *RFC* era crear un ciclo positivo de realimentación, con ideas o propuestas presentadas a base de que una *RFC* impulsara a otra con ideas adicionales y así sucesivamente. Una vez que se hubiera obtenido un consenso, se prepararía un documento de especificación. Tal documento sería entonces usado como la base para las implementaciones por parte de los equipos de investigación.

Con el paso del tiempo, las *RFC* se han enfocado a estándares de protocolos informáticos y sus especificaciones oficiales, aunque hoy todavía existen *RFC* informativas que describen enfoques alternativos o proporcionan información de soporte en temas de protocolos e ingeniería. Las *RFC* son vistas ahora como los documentos de registro dentro de la comunidad de estándares y de ingeniería en Internet. El acceso abierto a las *RFC* promueve el crecimiento de Internet porque permite que las especificaciones sean usadas a modo de ejemplo en las aulas universitarias o por emprendedores al desarrollar nuevos sistemas.

Otro medio que crecía en paralelo a la distribución de los *RFC* y cuyos temas comenzaron a carecer de contenidos informáticos fueron los ***foros de discusión***. Los foros de discusión son aplicaciones que le dan soporte a debates en línea. Estos programas son los descendientes modernos de los sistemas de noticias ***BBS (Bulletin Board System o Sistema de Tablero de Boletines)*** y los ***foros de noticias*** o ***USENET***.

La aparición de los primeros ordenadores personales, a finales de los 70's, trajo consigo la aparición de las primeras *BBS*, sistemas de información basados en un ordenador que algún particular de manera voluntaria hacía accesible vía módem a través de líneas telefónicas convencionales. Durante la década de los 70's y 80's tuvieron una cierta popularidad entre muchos aficionados a la informática que las utilizaban para intercambiar mensajes y software.

La primera de estas ***BBS*** apareció en 1978 en *Chicago*, fruto de la iniciativa de *Ward Christensen* y *Randy Suess* que crearon el software y el hardware necesario para que dos ordenadores personales pudieran intercambiar información a través de la línea telefónica. A menudo las *BBS*, que han llegado a contarse por decenas de miles, no tenían otro interés que el intercambio de software, pero ya a finales de los 70's ejemplos como *CommuniTree*, en *Santa Cruz (California)* u *Old Colorado City*, en *Colorado Springs (Colorado)* fueron ejemplos del uso de esta nueva tecnología para tratar temas de interés común a una ciudad o a una comunidad. Paralelamente, desde 1983 con la introducción del software *FidoBBS*, las diferentes *BBS* podían comunicarse fácilmente entre sí, creando una primera aproximación, aunque tecnológicamente mucho más limitada, de lo que diez años más tarde sería la World Wide Web.

A partir de las *BBS* nacerían las primeras redes libres, la base del concepto de lo que se entiende por una red ciudadana y que en Estados Unidos se expresa como *freenet*, *civic network* o *community network*. Fue en 1984 cuando el *Dr. Tom Grundner*, del *Departamento de Medicina Familiar* de la *Case Western Reserve University* en *Cleveland (Ohio)* estableció una *BBS*, llamada *St. Silicon's Hospital and Information Dispensary*, basada en un ordenador personal ***Apple*** y una línea telefónica. A través de ella los ciudadanos podían enviar consultas específicas para que un médico contestara la petición durante las siguientes 24 horas. El éxito impulsó al *Dr. Grundner* a dar un paso más y con el apoyo de la Universidad estableció en 1986 la *Cleveland Free-Net*, la primera red ciudadana que hoy sigue activa con más de cien mil usuarios. Es importante mencionar que los vínculos de la *free-net* de *Cleveland* con la Universidad fueron decisivos para que la tecnología utilizada en ella encajara perfectamente con el boom que experimentó Internet en años posteriores. A menudo las *free-nets* originales, tecnológicamente ligadas a las *BBS*, tuvieron ciertos reparos en adoptar la nueva tecnología de Internet que en los últimos años se ha convertido en un estándar y que ha integrado muchas de estas soluciones tecnológicas creadas para formar foros y centros de distribución de conocimiento. Otro tipo de foros abiertos de discusión que ha existido desde los inicios del Internet son los llamados *grupos de discusión* o *newsgroups (grupos de noticias)*.

La idea de los grupos de noticias nació en 1979, cuando dos estudiantes, *Tom Truscott* y *Jim Ellis*, pensaron en conectar ordenadores con el propósito de intercambiar información entre usuarios, e instalaron una pequeña red de tres ordenadores en *Carolina del Norte*.

Inicialmente el tráfico de información era manejado por cierto número de shell scripts (más tarde reescritos en C), pero que nunca fueron hechos públicos. Fueron rápidamente reemplazados por una nueva versión del sistema denominada *Anews*, que es la primera edición pública de programas para estos foros. *Anews* no estaba diseñado para manejar más que unos pocos artículos por grupo y día. Cuando el volumen de información continuó creciendo, fue reescrito por *Mark Horton* y *Matt Glickman*, quienes lo denominaron la versión *B* o *Bnews*. *Geoff Collyer* y *Henry Spencer* reescribieron y lanzaron en 1987 otra nueva versión, conocida como *Cnews*.

Todas las versiones hasta la *C* están principalmente diseñadas para utilizarse en redes ***UUCP***, aunque igualmente se han utilizado en otros entornos. La transferencia eficiente de noticias sobre redes tipo *TCP/IP* requirieron de otro planteamiento. Ésta es la razón por la que en 1986 se introdujo la ***Network News Transfer Protocol (NNTP)***. Este protocolo se basó en conexiones de red, y especifica cierto número de comandos para transferir los artículos de forma interactiva.

Otra aplicación *NNTP* diferente es ***INN*** o ***Internet News***, que es un sistema de noticias por derecho propio. Consta de un sofisticado sistema de noticias que es capaz de mantener varias conexiones *NNTP* simultáneas, y es por lo tanto, el software elegido por muchos servidores en Internet para implementar dichos foros. Estos foros fueron creciendo y siendo administrados por muchos programas incluidos en casi todos los sistemas operativos y son conocidos como ***USENET***.

USENET puede definirse como la colaboración de servidores separados que intercambian noticias de todo tipo. La unidad fundamental de las noticias de *USENET* es el artículo, que es un mensaje que un usuario publica en la red. Para posibilitar que los sistemas de noticias lo manejen, está precedido de información administrativa, conocida como cabecera del artículo; que es muy similar a la cabecera utilizada para el correo electrónico. Los artículos son enviados a uno o más grupos de noticias. De esta forma podría considerarse a los grupos como foros para artículos relativos a una misma temática. Todos los grupos están organizados en una jerarquía, en la cual el nombre de cada grupo indica su lugar en la misma. Esto a menudo hace más fácil ver sobre que versa un grupo de noticias.

Estos artículos son intercambiados entre todos los servidores de *USENET* a los que les interese tener noticias de este grupo. Cuando dos servidores acuerdan intercambiar noticias, son libres de intercambiar cualquier grupo que deseen, y pueden incluso añadir sus propias jerarquías locales.

Hoy en día, estos foros han crecido hasta alcanzar enormes dimensiones. Los servidores que llevan la totalidad de los grupos suelen tener que transferir grandes cantidades de información en rangos de Giga Bytes. Las noticias se distribuyen por la red de varias formas, hoy en día el caudal principal es llevado por servidores permanentemente conectados a Internet.

Actualmente los foros en Internet existen como un complemento a un sitio web invitando a los usuarios a discutir y compartir información relevante a la temática del sitio. Cada sitio de Internet aloja todos los comentarios y cualquiera puede acceder a ellos. Adicionalmente a los foros de discusión se han creado escenarios virtuales para depositar documentos con contenidos y temáticas referentes al sitio o a las empresas donde se alojan; estos programas son conocidos como servidores de documentos.

4.2 Servidor de Documentos

Un servidor de documentos puede definirse como un repositorio de documentos que son compartidos y administrados por usuarios organizados por diferentes jerarquías que les proveen diferentes privilegios de acceso al sistema; los limita y les permite tener acceso a ciertas funciones que van desde escribir, publicar, revisar, administrar y buscar los contenidos del servidor usualmente usando un sistema basado en páginas de Internet para su gestión.

Existen en el mercado varios servidores de documentos con reglas ya establecidas acerca de su administración y gestión; el más común en ambientes Windows es ***Microsoft Share Point***. Este servidor crea automáticamente un sitio administrable desde una interfaz Web una serie de carpetas que pueden ser accedidas desde un explorador o directamente desde un editor de texto, si lo que se desea es realizar una publicación.

Ciertamente este tipo de programas realizan una labor extraordinaria en cuanto a la facilidad para realizar publicaciones ya sea en una intranet dentro de una empresa o directamente en Internet como un sitio corporativo robusto y con una seguridad prácticamente invulnerable. Sin embargo, no significa que de respuesta a cualquier tipo de necesidades empresariales.

Cualquier sistema ya diseñado, ya sean servidores de documentos o una simple aplicación de facturación puede ser adaptado al tipo de administración y objetivo de una compañía, sin embargo, la idea no es adaptar a la empresa a un cierto sistema, sino todo lo contrario, lo correcto es adaptar el sistema a la misión de la empresa; ya que si no fuese así, existirían al fin y al cabo necesidades que no podrían ser cubiertas por los programas implantados por empresas que no pueden contemplar todas las necesidades de los clientes.

Es trabajo de los programadores confeccionar trajes a la medida exacta del cliente, cubrir todas sus necesidades y adaptar el sistema a la forma de trabajar de la empresa, ya que esta forma de trabajar afecta en menor grado la adaptación a una nueva forma de hacer las cosas, si es que nunca se ha trabajado con una administración computarizada; de otra forma, la adaptación a un nuevo sistema informático debería ser transparente y el personal a cargo podrá familiarizarse con los programas y al final, trabajar mejor en poco tiempo.

Cuando nunca se han adoptado sistemas de cómputo para llevar una gestión de alguna entidad funcional, se puede presentar el problema de tener que elegir entre adquirir un sistema ya hecho o crear uno propio. Tal vez lo primero que se debe de tomar en cuenta es que si la empresa cuenta con un departamento de sistemas que cuente con personal para desarrollo; si es así no habrá problema. De otra forma la solución inmediata sería adquirir una aplicación establecida y adaptarse a ella; tomando en cuenta que, para el caso del servidor de documentos de Microsoft los costos de las licencias son elevados, ya que el tipo de gestión que realiza el programa es, sin duda alguna, para una robusta cantidad de publicaciones y para redes muy grandes y por lo tanto, pensadas para tráficos pesados y congestionados.

En el caso de la AMD si existe la posibilidad de desarrollar los sistemas de cómputo; y es por esto que se presenta esta propuesta para desarrollar un sistema que adopte lo mejor de otros servidores de documentos, adapte sus funciones y adicione nuevas ideas exclusivas para publicaciones dermatológicas y que la finalidad sea crear un foro para dermatólogos que deseen publicar sus ideas y descubrimientos con toda la comunidad médica con el fin de contribuir con la ciencia en pos del bienestar de la gente.

4.3 Definición del Sistema

El sistema de la biblioteca virtual para publicaciones dermatológicas esta basado en una aplicación que administre documentos de forma automática, el gestor de estos documentos serán las mismas personas que accedan al sitio para fungir con varios roles que especifican sus funciones y privilegios dentro del sistema. Los roles dentro del sistema son autores, coordinadores, jueces y por último, los lectores. La finalidad del sistema es que un autor someta a juicio un documento para su discusión y posterior aprobación para poder ser liberado a una sección en donde no hay restricciones de acceso y poder ser leído por cualquier persona.

En principio la idea es que la biblioteca funja como un contenedor de documentos exclusivo para los miembros de la AMD, sin embargo, la idea final es que los documentos sean liberados a toda la comunidad médica que desee estudiarlos; sin embargo, hay que tener en cuenta que este tipo de documentos son estudios médicos que pueden ser utilizados de forma indebida o simplemente plajeados por otras personas, y es por esto que esta propuesta se remite a que las personas que acceden al sistema tengan la certeza de que sus documentos van a ser accedidos por personas completamente identificadas y certificadas como miembros de la AMD y, por lo tanto, la AMD debe hacerse responsable de la exclusividad de estos documentos, si es que el autor no desea que sean mal utilizados. Por esto el sistema debe de contar con un sistema de autentificación que permita el acceso exclusivo a miembros de la AMD.

La biblioteca debe contar con un sistema de almacenaje de documentos que se adapte al tipo de hospedaje con que cuenta la AMD. Esto es muy interesante, ya que cuando se cuenta con un servicio de hospedaje dedicado, el uso del servidor y los permisos con que se cuenta no existe restricción de acceso alguno, y por lo tanto, se pueden dar de alta servicios como FTP o adoptar cualquier servidor de Internet. Pero actualmente la AMD cuenta con un servicio de hospedaje dedicado en el que no se pueden contar con estos servicios, simplemente el permiso de escritura de archivos en el servidor esta restringida, y por ello la única forma de guardar un archivo es a través de Bases de Datos.

Uno de los propósitos primordiales de esta propuesta es reducir los costos al mínimo, por ello se adoptarán los recursos con que se cuenta como los únicos medios para poder realizar dicho proyecto.

4.4 PHP y MySQL como Plataformas de Desarrollo

Como se mencionó en el capítulo anterior, la plataforma de desarrollo elegida por la AMD para sus desarrollos en Internet es PHP. Este lenguaje de programación proporciona todas las herramientas para poder desarrollar un sitio con las características necesarias para llevar una administración autónoma e independiente de la base de datos que se haya elegido ***(ver tabla 3.1)***. Dentro de los servicios que proporciona *Prodigy* en el plan de hospedaje de la AMD ***(ver tabla 3.3)*** se incluye el administrador de base de datos *MySQL* sin costo adicional.

A través de PHP es posible almacenar documentos o cualquier tipo de archivos dentro de registros especiales en tablas de *MySQL*. En la ***figura 4.1*** se muestra el esquema básico de almacenaje de documentos dentro de la base de datos de *MySQL*.

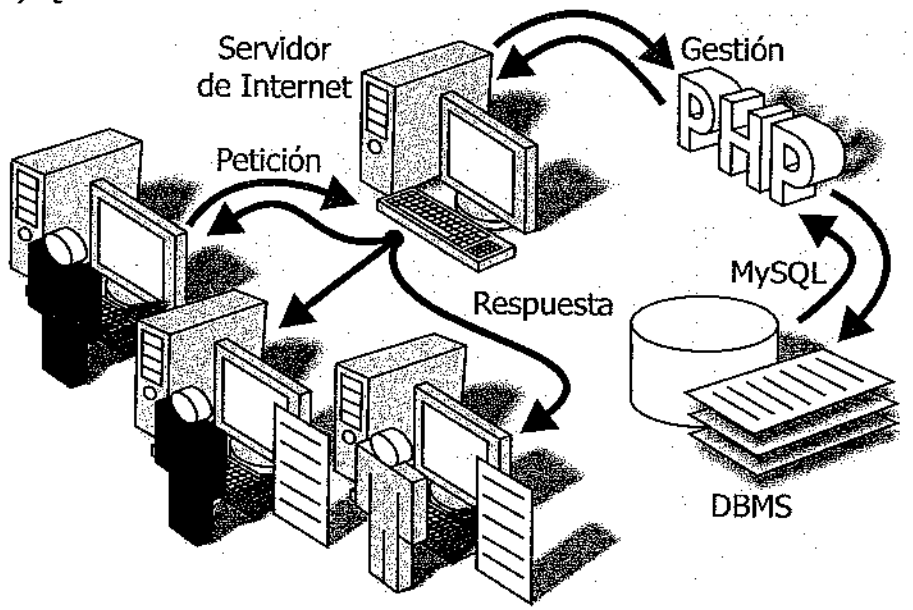


Figura 4.1 → Esquema básico de almacenaje de documentos

Cuando un usuario realiza una petición al servidor de Internet, el sistema debe realizar la gestión necesaria para obtener o guardar un documento dentro de la base de datos y realizar la respuesta a la petición de forma integral. Este tipo de almacenamiento tiene varias ventajas, la más evidente es que no se requiere de un servidor dedicado para almacenar los documentos, y esto implica que cualquier servidor de Internet que cuente con los servicios de *PHP* y *MySQL* podrá almacenar el programa.

Dadas las restricciones que existen para la escritura de archivos en un servidor compartido, la única forma de poder almacenar archivos es a través de una base de datos, sin embargo, también existen desventajas a este tipo de administración; la más importante es que no es posible crear una estructura de directorios para almacenar los documentos, y es por este motivo que debe crearse un sistema para poder crear categorías a los archivos, pero este tipo de tarea debe ser administrada por el sistema de la biblioteca; es decir, al guardar los documentos en tablas, sólo es posible jerarquizarlos a través de un sistema de gestión de categorías que estén incluidos en el sistema de las tablas y al mismo tiempo, cada registro deberá guardar su posición dentro de estas categorías.

Como desventaja obvia a este sistema es que sólo se pueden guardar un número finito de subcategorías, pero al mismo tiempo es posible tener el control completo de éstas y poder crear un sistema de búsquedas basado en estas categorías y poder buscar documentos de forma rápida y eficaz.

El espacio en disco duro para el hospedaje compartido es limitado; por lo tanto, es importante llevar un control del espacio con que se cuenta. *PHP* permite obtener el tamaño de cada uno de sus registros de la base de datos y sus tablas, por lo tanto, es sencillo limitar el número de documentos y sus tamaños dentro del sistema. Al tener conocimiento del espacio que se ocupa, es posible anticiparle al usuario que es lo que puede publicar y cuánto espacio le queda. Esta es una ventaja que otros sistemas de archivos no poseen, ya que muchos de ellos la única forma de avisarle al usuario que ya no existe espacio es enviarle mensajes de error cuando una petición no se pudo llevar a cabo.

PHP no presenta limitantes de desarrollo tanto para el sistema de almacenamiento de archivos dentro de la base de datos, como para los sistemas de gestión, administración y autenticación de usuarios. Por ello y por los beneficios que atrae al ser *Open Source*, como lenguaje de programación, es la mejor opción para este sistema.

4.5 Roles y Tareas para los Usuarios del Sistema

Uno de los aspectos más importantes dentro de este sistema son los diferentes roles de usuarios, ya que son estos los que generan la estructura del funcionamiento de la biblioteca de documentos. Los roles para los diferentes usuarios crean estructuras bien definidas del flujo de los datos dentro del sistema. Es interesante comenzar a definir primero los roles, ya que son los usuarios los que llevarán por completo la gestión y administración del sitio; serán ellos quienes decidan qué y que no será publicado en la biblioteca, también ellos serán los que den los permisos dentro de la estructura de usuarios y de esta forma limitar el

acceso de los médicos a las diferentes secciones de la aplicación. Existen básicamente cuatro tipos de usuarios dentro del sistema, en rango de jerarquía son:

1. **Lectores**
2. **Autores**
3. **Jueces**
4. **Coordinadores**

El ***Lector*** es el usuario final de todo el sistema; es la persona que navega dentro del sitio para revisar los documentos, ubicándolos por categorías o simplemente buscándolos por título o por la descripción de su contenido. El lector accede a los documentos con la certeza de que han sido aprobados y revisados minuciosamente por médicos especialistas en el tema y él mismo deberá estar avalado por la AMD. En la ***figura 4.2*** es posible ver la interacción que realiza el lector dentro del sistema.

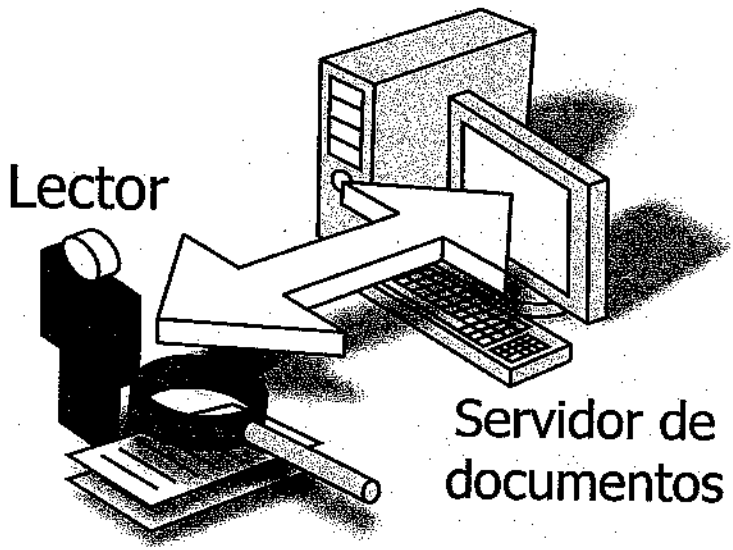


Figura 4.2 → Lector

Los ***Autores*** son el pilar del programa, ya que son ellos los que redactan los documentos que serán publicados dentro del sitio. Cualquier miembro registrado dentro de la base de datos puede ser un autor. Para que los dermatólogos puedan ser miembros de la AMD deben recaudar puntos, los cuales, son obtenidos asistiendo a los diferentes eventos que organiza la AMD. También es posible obtener puntos participando en simposios o presentando trabajos libres en las sesiones mensuales, jornadas en provincia o congresos bienales. Los puntos por

presentar estos trabajos, obviamente, superan en cantidad a los que se podrían obtener por simplemente asistir a los eventos. Una de las propuestas para este sistema es que los interesados en ingresar a la AMD puedan publicar un trabajo en la biblioteca. Posteriormente el ***Coordinador*** le podrá asignar tres ***Jueces*** para que sean los responsables de las revisiones, las correcciones y la posterior aprobación del documento. En la ***figura 4.3*** se puede observar el proceso de asignación de un jurado por parte del coordinador.

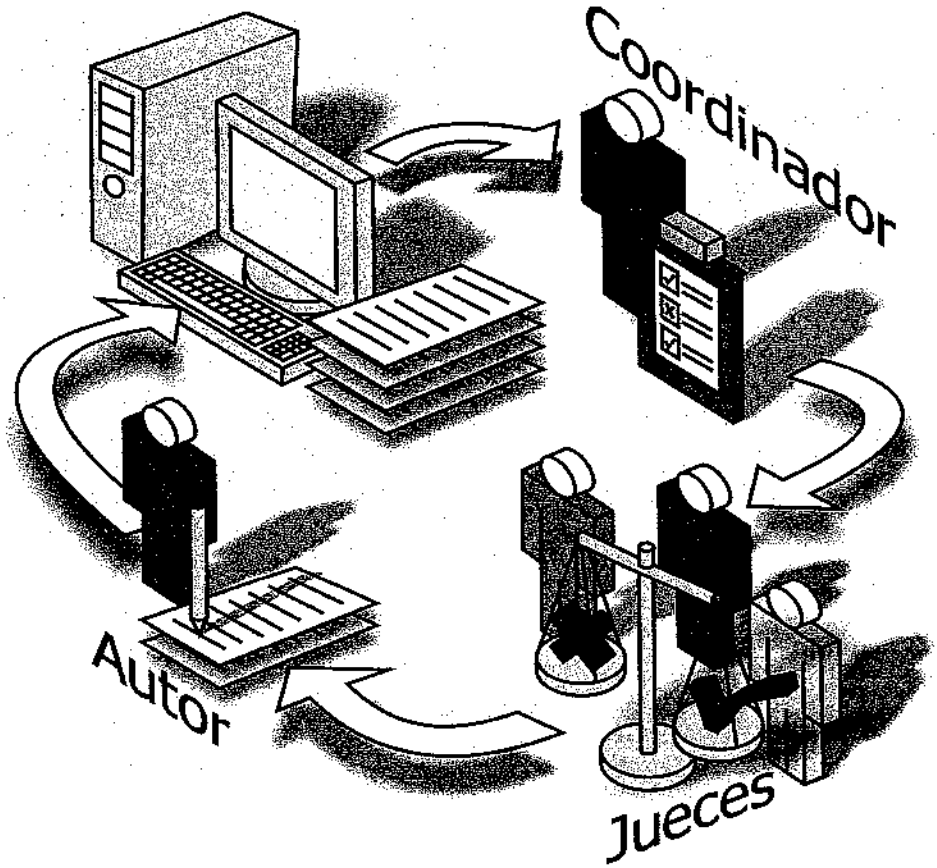


Figura 4.3 → Asignación del jurado para un autor

Cuando un autor desea publicar un documento, lo primero que deberá hacer es publicar una primera propuesta, especificando el título del tema y su descripción. El autor debe poner el tema de acuerdo a su especialidad (del médico) y considerando la temática de su proyecto. Posteriormente el documento estará

publicado en un apartado de proyectos que temporalmente se presentarán como propuestas. Mientras permanece como propuesta, el autor se encontrará en espera de que existan por lo menos dos dermatólogos de su misma especialidad interesados en participar en el proyecto como jueces. Cuando haya una lista de por lo menos dos médicos interesados en ser jueces de este proyecto, un coordinador asignará a dos médicos de esta lista como jueces finales; adicionalmente asignará un tercer juez que deberá ser parte de la mesa directiva de la AMD.

Básicamente el trabajo de un juez es fungir como sinodal del documento, es decir, el autor presenta una primera versión del documento, cada uno de los jueces calificará el manuscrito ingresando sus comentarios dentro de un foro especialmente diseñado para requerir respuesta del autor, de esta forma si el juez desaprueba la primera versión, el autor deberá responder a la petición del juez en el foro y deberá lanzar una segunda versión, y así sucesivamente; como se muestra en la ***figura 4.4.***

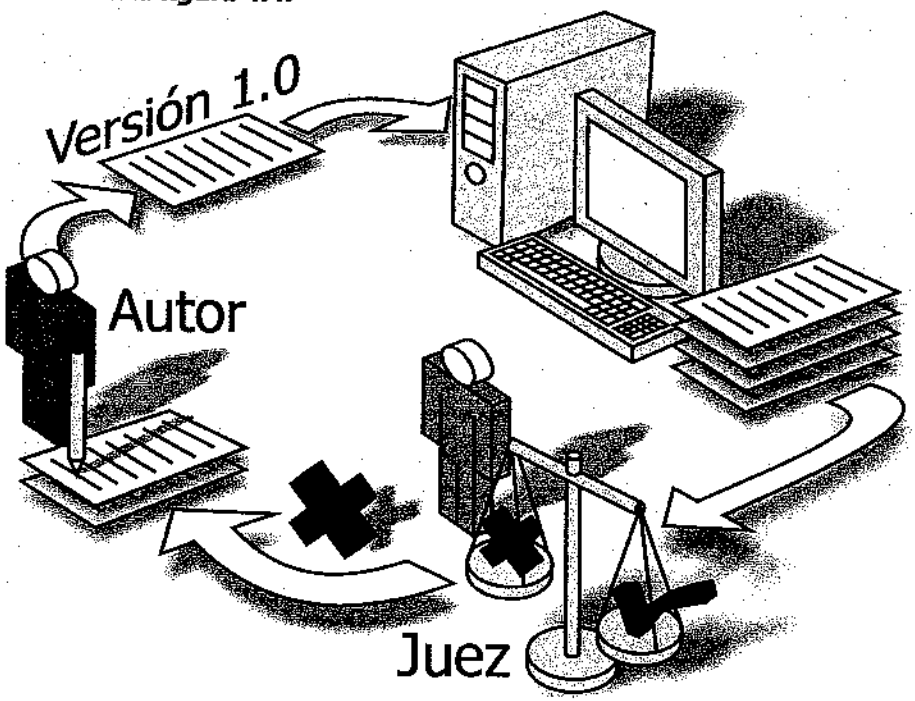


Figura 4.4 → Desaprobación de la primera versión del documento

El foro de discusión de los documentos es una parte importante del sistema, ya que es en esta sección donde el juez expresa sus inquietudes acerca del documento al autor, y es a través de este medio que el autor le da respuesta al jurado y posteriormente podrá liberar una nueva versión del documento. El control de las versiones será gestionado automáticamente por el sistema, aumentando una décima de número al documento. El control de las versiones es importante, ya que es a través de ésta que el autor y el juzgado lleven el control de cuáles son los cambios que se están adicionando al documento.

Cuando el juez haya dado el visto bueno a la versión final del documento, automáticamente podrá dar su voto de aprobación para que posteriormente el coordinador pueda aprobarlo y automáticamente pasará al apartado de documentos liberados y poder ser visto por cualquier persona; como se muestra en la ***figura 4.5.***

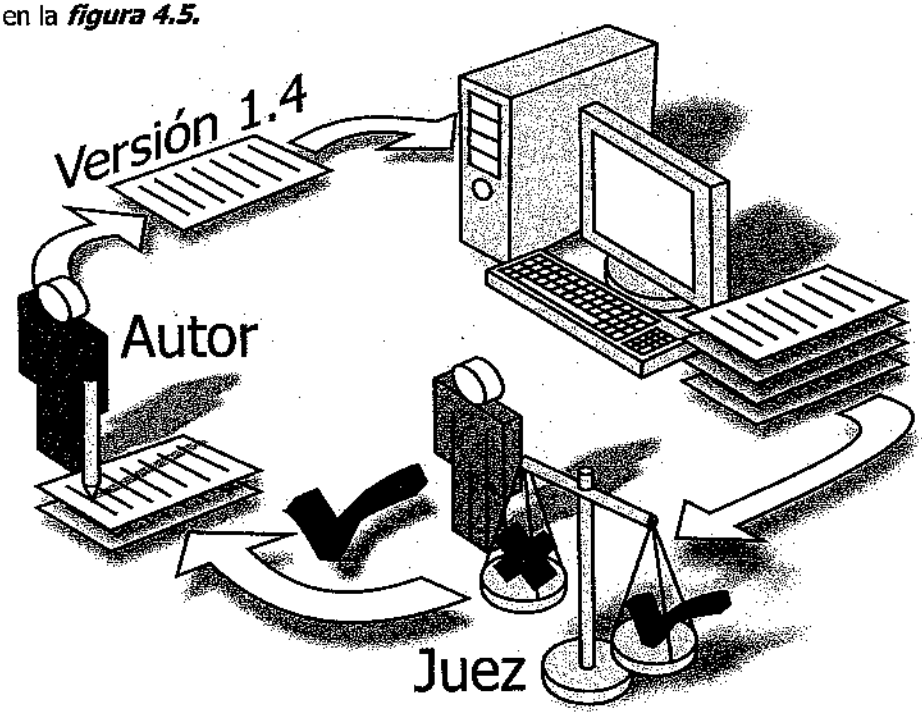


Figura 4.5 → Aprobación de la versión definitiva de un documento

Para que un documento sea liberado, el primer requisito es que el documento deberá contar con la aprobación de los tres jueces; posteriormente un coordinador podrá dar un último vistazo al documento, y si no existe objeción, el coordinador

será el único con permisos para poder filtrarlo al apartado de documentos liberados y poder ser visto por todos los usuarios, incluyendo obviamente a los lectores. En la ***figura 4.6*** se puede observar una ilustración de lo antes mencionado.

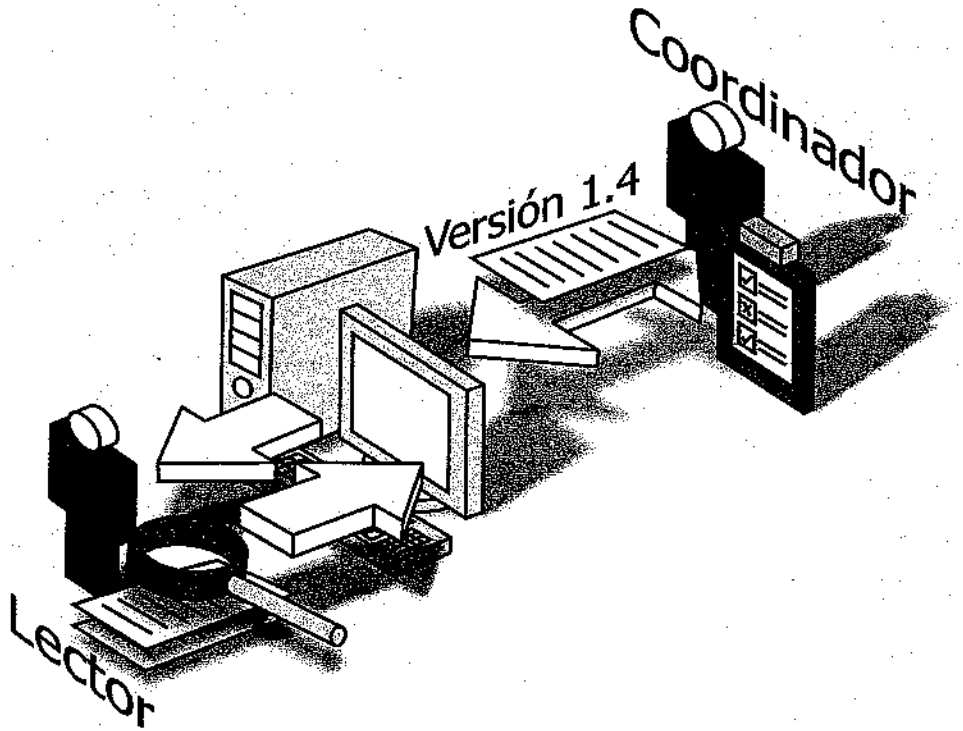


Figura 4.6 → Liberación de la última versión del documento

La autoridad principal dentro del sistema es el coordinador. Todos los miembros de la mesa directiva de la AMD son coordinadores, son sólo ellos los que tienen los permisos para:

- Asignar jurado a un documento
- Crear nuevas categorías para los documentos (se debe recordar que como el programa no tiene un sistema de almacenamiento basado en carpetas, la aplicación organiza los documentos por categorías y subcategorías)
- Liberar un documento después de haber obtenido las tres aprobaciones del jurado

El coordinador juega un papel crucial en la administración del sistema, ya que es él el que realiza la mayoría de las tareas administrativas y de organización de los registros, documentos y usuarios dentro del sistema. Este tipo de perspectiva es sumamente útil cuando se desea delegar las responsabilidades de gestión por completo a los usuarios finales, ya que de otra forma, la carga de trabajo cae sobre el informático, y es él la única persona que puede realizar el trabajo, así que su presencia e intervención es indispensable.

Dentro del sistema, todos las encargos los realizan los usuarios; son ellos los responsables de todo lo que aparece en el sitio, ellos son los que realizan las publicaciones aún sin saber nada de programación en Internet, y por lo tanto, el personal informático puede ocuparse de otros proyectos sin ser interrumpido por una simple publicación, aprobación de algún documento o simplemente para dar de alta a un usuario; todo este tipo de labores se podrán realizar a través del sitio con una interfaz simple basada en páginas de Internet, que actualmente, casi cualquier persona ha usado e interactuado, ya sea por diversión o porque se ha vuelto una necesidad para cualquier tipo de profesión u oficio; de esta forma el Internet se ha vuelto no sólo un medio de comunicación, sino una herramienta indispensable.

La publicación de documentos dentro del sitio de la AMD debe resultar una forma atractiva de compartir información; la primera propuesta para los médicos que desean formar parte como miembros de la Academia es que se les otorguen puntos por cada publicación dentro del sitio; sin embargo, como una propuesta adicional, se podría hacer la recomendación de dar descuentos para posteriores asistencias a los diferentes eventos organizados por la AMD, esto resultaría sutilmente atractivo para los miembros que deseen integrarse como parte del cuerpo de jueces del sistema; ya que es importante recordar que el juez juega un rol de suma importancia en la jerarquía de usuarios, ya que sin ellos no existiría un control de calidad de las publicaciones en el sitio.

Como último recurso, si no existiera la forma de sustentar los gastos de dichos descuentos, podría proponerse como una obligación para seguir siendo miembros de la AMD, es decir, no sólo cubrir las cuotas anuales para los miembros sería suficiente, ya que el sitio promueve la investigación y la divulgación de los conocimientos médicos, sería muy interesante cultivar en los médicos la cultura de una continuidad en su formación y el interés por crear una verdadera comunidad científica, especialista en dermatología y unida a través de la Academia Mexicana de Dermatología.

Es importante resaltar que la idea de la biblioteca es precisamente que los médicos que realizan las publicaciones no formen una imagen de lucro ni de beneficio por la participación en el mismo, sino completamente lo contrario, es importante en los tiempos actuales que la humanidad tenga una mentalidad altruista y de cooperación, y al final, poder beneficiar a toda la gente que acude a ellos en busca no solo de alivio a sus padecimientos, sino apoyo físico y moral, el cual es, al fin y al cabo, obligación de todos los médicos, sin importar su especialidad o rango.

4.6 Análisis Estructurado

El análisis estructurado permite comprender la forma en que funcionará un sistema informático, creando la división del sistema en componentes y construyendo un modelo organizado del sistema. Este método incorpora elementos tanto de análisis como de diseño, se concentra en especificar lo que se requiere que haga el sistema o la aplicación.

El análisis estructurado permite que los desarrolladores observen los elementos lógicos, es decir, lo que debe hacer cada uno de los componentes, separados de los elementos físicos con que funcionan y el lenguaje de programación que se utilizará para realizar el desarrollo. Éste análisis permite tanto al analista como al programador conocer un sistema o proceso en una forma lógica y manejable que proporciona la base para asegurar que no se omita ningún detalle durante el desarrollo.

4.6.1 Diccionario de Datos

En el diccionario de datos del sitio de Internet contiene las características lógicas de los registros del sistema que se almacenarán en la base de datos, incluyendo nombres, descripciones, tipos, organización y características especiales. A través del diccionario se podrán identificar los datos que se emplean en los procesos del sitio donde se requiera el acceso o el ingreso a la información del sistema.

La tabla raíz de la que se derivarán todas las subsecuentes será sin duda la tabla *"miembros"*, dicha tabla es importada a la base de datos *MySQL* a través de un controlador *ODBC* desde la base de datos del sistema de información ***(ver Capítulo 2)*** y desde su tabla con el mismo nombre. Esta tabla no contendrá todos los datos de la tabla origen; por cuestiones de seguridad no son exportados todos los campos de la original, lo datos listados en la ***tabla 4.1,*** dicha tabla contiene los registros estrictamente necesarios para el sistema de la biblioteca virtual y para los contenidos dinámicos de la página de Internet ***(Capítulo 3).***

Nombre del Campo	Tipo	Tamaño	Características	Descripción
código	Texto	12	**Clave Principal,** Requerido, NO Duplicado	Código alfa-numérico único e indicador de los miembros.
nombre	Texto	20	Requerido	Nombre(s) del miembro.
app	Texto	20	Requerido	Apellido Paterno
apm	Texto	20	Requerido	Apellido Materno
genero	Texto	6	Requerido	Genero del Miembro; únicamente podrá ser uno de los siguientes: • DR. • LIC. • DRA. • EST. • SR. • ING. • SRA.
delegacioncon	Texto	25	Requerido	Domicilio (Delegación Política del Consultorio Principal)
estado	Texto	23	Requerido	Estado de la República Mexicana del Extranjero, podrá ser uno de los 32 Estados de la República Mexicana o Cualquiera del Extranjero.
pais	Texto	6	Requerido	País en el que radica, únicamente podrá ser uno de los siguientes: • MEXICO • EXTRAN
especialidad_1	Texto	20	Requerido	Especialidad Médica Uno
especialidad_2	Texto	20	No Requerido	Especialidad Médica Dos
tel_consu_1	Texto	19	Requerido	Teléfono Uno de Consultorio
tel_consu_2	Texto	19	No Requerido	Teléfono Dos de Consultorio
fax	Texto	19	No Requerido	
miembro	Texto	8	Requerido	Indica si es *miembro* de la AMD, de la FMD* o ninguno de los dos, únicamente podrá ser uno de los siguientes: • AMD • AMD-FMD • OTRO

Nombre del Campo	Tipo	Tamaño	Características	Descripción
cedula	Texto	12	No Requerido	Tipo de cédula profesional, podrá tomar únicamente los siguientes valores: • NO TIENE • PROFESIONAL • ESPECIALISTA
nocedula	Texto	7	No Requerido	Número de Cédula
cp	Texto	10	Requerido	Código Postal
email	Texto	60	No Requerido	Correo Electrónico Particular

Tabla 4.1 → Tabla "miembros" de la base de datos

** Al ser miembro de la Fundación Mexicana para la Dermatología, automáticamente forma parte de la Academia Mexicana de Dermatología, y es por esto que se considera una única clave unificada dentro del campo Miembro para AMD-FMD.*

Es importante recordar que la adición, edición y la eliminación de registros de la tabla *"miembros"* se lleva a cabo en el sistema de información, la tabla alojada en la base de datos *MySQL* simplemente es una copia que se transfiere cada vez que se realiza un cambio en la página de Internet. Esta tabla servirá para identificar a cada uno de los miembros cuando accedan al sistema y realicen cualquier transacción, ya que los únicos usuarios autorizados para entrar al sitio de la biblioteca deberán estar registrados en la tabla *"miembros"*.

Los primeros elementos para el almacenamiento de los documentos en la biblioteca son las categorías; como se ha mencionado anteriormente, para poder clasificar los documentos se deben crear categorías para establecer una estructura finita de clasificaciones de los documentos; básicamente las categorías deben crearse a partir de las diferentes especialidades médicas existentes para dermatólogos, ya que primordialmente la documentación publicada se basa en la dermatología como tronco principal de donde se ramifican dichas especialidades. Para ello se deberán crear dos tablas descritas en las ***tablas 4.2 y 4.3.***

Nombre del Campo	Tipo	Tamaño	Características	Descripción
key_categoria	Entero	4	**Clave Principal**, Requerido, NO Duplicado	Llave numérica única e indicadora de las categorías.
categoria	Texto	30	Requerido	Nombre de la categoría.

Nombre del Campo	Tipo	Tamaño	Características	Descripción
descripcion	Objeto Binario Mediano (MediumBlob)	---	Requerido	Descripción textual de la categoría.
habilitado	Booleano	1	Requerido, Valor inicial: Verdadero	Registro que indica si la categoría actual se encuentra habilitada para poder ser procesada.

Tabla 4.2 → Tabla "categoría" de la base de datos

Nombre del Campo	Tipo	Tamaño	Características	Descripción
key_subcategoría	Entero	4	**Clave Principal,** Requerido, NO Duplicado	Llave numérica única e indicadora de las subcategorías.
key_categoria	Entero	4	Requerido	Llave numérica única e indicadora de la categoría raíz original.
subcategoria	Texto	30	Requerido	Nombre de la subcategoría.
descripcion	Objeto Binario Mediano (MediumBlob)	---	Requerido	Descripción textual de la subcategoría.
habilitado	Booleano	1	Requerido, Valor inicial: Verdadero	Registro que indica si la subcategoría actual se encuentra habilitada para poder ser procesada.

Tabla 4.3 → Tabla "subcategoria" de la base de datos

El nombre de la base de datos y los permisos de acceso son configurados automáticamente por el servidor de Prodigy al dar de alta las cuenta para el hospedaje compartido; este servicio provee un nombre especial para la base de datos y no se puede cambiar, por lo tanto, el nombre de la base de datos y los permisos para acceder a las tablas deberá crearse como un *socket* especial de acceso, es decir, ubicarlo en una capa independiente de los programas que accedan a las tablas dentro del sistema; de esta forma, si se llegara a cambiar la forma de hospedar la aplicación, no debería existir diferencia al migrar el contenido de las tablas a otro ambiente, incluso a otro sistema operativo.

El tipo de dato especial llamado ***BLOB*** es un registro dentro de las bases de datos *MySQL* llamado también *Objeto Binario Grande (Binary Large Object),* este objeto puede almacenar un volumen variable de datos de cualquier tipo, texto, imágenes o archivos en forma binaria. Los cuatro tipos *blob, TINYBLOB, BLOB,*

MEDIUMBLOB, y *LONGBLOB* difieren sencillamente en el tamaño máximo de los valores que pueden almacenar, que van desde 255 (2^8 - 1) hasta 4,294,967,295 (2^{32} -1) caracteres.

Las tablas *"categoría"* y *"subcategoría"* representan dos catálogos simples conectados entre si a través de la relación uno a varios representada en la ***figura 4.7.***

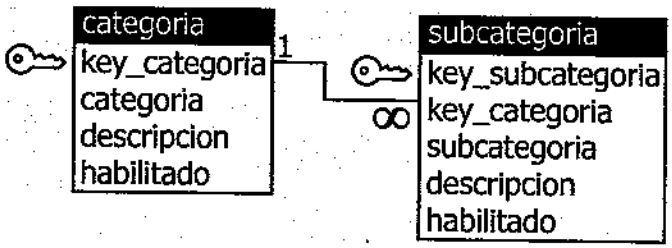


Figura 4.7 → Relación entre las tablas "categoría" y "subcategoría"

Dentro del sistema deben de existir elementos adicionales para identificar a los miembros; de ello se encargará la tabla *"extras_miembros"*, en ella se alojarán campos especiales de identificación de los miembros. Uno de los principales campos de esta tabla es el poder alojar una fotografía del médico, ya que esta será una forma adicional de identificar a los usuarios y sus roles dentro del sistema, además de poder crear lazos de familiaridad entre los usuarios. La lista completa de los campos de esta tabla se encuentra en la ***tabla 4.4.***

Nombre del Campo	Tipo	Tamaño	Características	Descripción
código	Texto	12	**Clave Principal**, Requerido, NO Duplicado	Código alfa-numérico único e Indicador del código original en la tabla *"miembros"*.
foto	Objeto Binario Mediano (MediumBlob)	---	No Requerido	Fotografía del miembro.
tipo_foto	Texto	15	No Requerido	Campo textual necesario para mostrar la imagen, contiene el tipo específico del archivo original de la fotografía del miembro.
curriculum	Objeto Binario Mediano (MediumBlob)	---	No Requerido	Currículum del miembro.

Nombre del Campo	Tipo	Tamaño	Características	Descripción
password	Texto	32	Requerido	Password del miembro *encriptado* con ***MD5***
habilitado	Booleano	1	Requerido, Valor inicial: Verdadero	Registro que indica si la fila actual se encuentra habilitada para poder ser procesada.

Tabla 4.3 → Tabla "extras_miembros" de la base de datos

Otro elemento adicional y opcional de la tabla anterior es el currículum. Este campo podrá contener un pequeño compendio en formato textual del currículum del miembro, lo cual podrá ser mostrado como un dato adicional de identificación del miembro, ya sea dentro de sus publicaciones o como identificación como juez dentro del sistema.

Para que existan definiciones específicas de los roles de los usuarios dentro del sistema existen diferentes tablas que los identifican y sirven para dividir las diferentes opciones y programas a los que podrán acceder y utilizar. También es importante jerarquizar los permisos de acceso a los programas, es decir, el rol que posee todos los permisos de acceso y funge como superusuario dentro del sistema es el coordinador. La tabla que lleva su mismo nombre especifica los campos básicos para su identificación. Es posible ver los campos de la tabla *"coordinador"* en la ***tabla 4.4.***

Nombre del Campo	Tipo	Tamaño	Características	Descripción
código	Texto	12	**Clave Principal,** Requerido, NO Duplicado	Código alfa-numérico único e indicador del código original en la tabla *"miembros".*
fecha_alta	Fecha	---	Requerido	Fecha en que se dio de alta el coordinador dentro del sistema.
hora_alta	Hora	---	Requerido	Hora en que se dio de alta el coordinador dentro del sistema
habilitado	Booleano	1	Requerido, Valor inicial: Verdadero	Registro que indica si la fila actual se encuentra habilitada para poder ser procesada.

Tabla 4.4 → Tabla "coordinador" de la base de datos

Antes de poder registrar un documento, los autores deberán presentar una propuesta para que sea revisada por los coordinadores, dicho coordinador podrá revisar esta propuesta y posteriormente ponerla a juicio de diferentes jueces que de la misma forma se podrán proponer como jueces de dicho documento. Los registros de las propuestas de los posibles autores serán registrados en la tabla *"propuesta"* desglosada en la ***tabla 4.5.***

Nombre del Campo	Tipo	Tamaño	Características	Descripción
no_propuesta	Entero	7	**Clave Primaria,** Requerido, NO Duplicado, Auto-numérico	Llave principal auto-numérica e indicadora en conjunto con el campo *"código"* del número de propuesta.
código	Texto	12	**Clave Secundaria,** Requerido	Código alfa-numérico único e indicador del código original en la tabla *"miembros"* del autor de la propuesta.
titulo	Texto	100	Requerido	Título del tema propuesto.
descripcion	Objeto Binario Mediano (MediumBlob)	---	Requerido	Descripción textual del tema propuesto.
key_categoria	Entero	4	Requerido	Llave numérica indicadora de la categoría raíz original.
key_subcategoria	Entero	4	Requerido	Llave numérica indicadora de la subcategoría raíz original.
fecha_propuesta	Fecha	---	Requerido	Fecha en que se dio de alta el coordinador dentro del sistema.
hora_propuesta	Hora	---	Requerido	Hora en que se dio de alta el coordinador dentro del sistema
no_tema	Entero	5	Requerido	Llave numérica identificadota del número de tema asignado cuando la propuesta sea aceptada.

Nombre del Campo	Tipo	Tamaño	Características	Descripción
propuesta_aceptada	Booleano	1	Requerido, Valor inicial: Falso	Registro que indica si la propuesta ya ha sido aceptada.
habilitado	Booleano	1	Requerido, Valor inicial: Verdadero	Registro que indica si la fila actual se encuentra habilitada para poder ser procesada.

Tabla 4.5 → Tabla "propuesta" de la base de datos.

Cuando un posible autor desea crear una nueva propuesta en el sistema, deberá indicar primero la categoría y subcategoría de la especialidad médica a la que pertenece su investigación; en el caso de que no existiese la categoría o su subclasificación, la propuesta sería que presentara por correo electrónico una moción a cualquier coordinador para que diese de alta la categoría o subcategoría, esto es porque en primera instancia la única persona con permisos para poder crear las categorías sería un coordinador, y es por esto que es tan importante que un miembro de la mesa directiva realice un estudio para poder clasificar todas las posibles especialidades en que se divide la dermatología, y de esta forma, contemplar las más importantes.

Cuando la propuesta sea aceptada por cualquier coordinador, tanto el nombre del documento como su descripción permanecerán sin cambios hasta su publicación final. Una vez aceptada, la propuesta pasará como un documento a juicio de ser evaluado por los jueces y estará vinculada con una nueva tabla llamada *"autor"* por el campo *"no_tema"*. Este campo permitirá poder clasificarla posteriormente si se requiriera un reporte de las propuestas aceptadas y las que permanecen en espera de poder ser juzgadas para su posterior publicación.

Como se ha mencionado, cuando las propuestas sean aceptadas para poder ser juzgadas, deberá crearse un nuevo registro en una nueva tabla llamada *"autor"*; esta tabla registra todos los documentos que han sido aceptados y se encuentran en espera de que se les asigne un jurado o que el jurado designado para su evaluación de su aprobación para que el coordinador pueda darle el último visto bueno y pueda ser publicado para el público en general. En la ***tabla 4.6*** se puede ver el desglose de los datos que se registrarán en la tabla *"autor"*.

Nombre del Campo	Tipo	Tamaño	Características	Descripción
no_tema	Entero	7	**Clave Primaria,** Requerido, NO Duplicado, Auto-numérico	Llave principal auto-numérica e indicadora en conjunto con el campo *"código"* del numero de tema.
código	Texto	12	**Clave Secundaria,** Requerido	Código alfa-numérico único e Indicador del código original en la tabla *"miembros"* del autor del tema.
titulo	Texto	100	Requerido	Título del tema.
descripcion	Objeto Binario Mediano (MediumBlob)	---	Requerido	Descripción textual del tema.
key_categoria	Entero	4	Requerido	Llave numérica indicadora de la categoría raíz original.
key_subcategoria	Entero	4	Requerido	Llave numérica indicadora de la subcategoría raíz original.
version	Número Flotante	4,2	Requerido	Número de versión del documento; cada vez que se realice un cambio en el documento, se aumentará una décima a la versión.
juez_1	Texto	12	Requerido	Primer juez asignado para la evaluación y posterior aprobación del documento.
juez_2	Texto	12	Requerido	Segundo juez asignado para la evaluación y posterior aprobación del documento.
juez_3	Texto	12	Requerido	Tercer juez asignado para la evaluación y posterior aprobación del documento. El tercer juez deberá ser un miembro de la mesa directiva.

Nombre del Campo	Tipo	Tamaño	Características	Descripción
fecha_alta	Fecha	---	Requerido	Fecha en que se aceptó la propuesta.
hora_alta	Hora	---	Requerido	Hora en que se aceptó la propuesta.
fecha_cambios	Fecha	---	Requerido	Fecha en que se realizó el último cambio al documento.
hora_cambios	Hora	---	Requerido	Hora en que se realizó el último cambio al documento.
documento	Objeto Binario Mediano (MediumBlob)	---	Requerido	Documento en formato binario.
tipo_documento	Texto	15	Requerido	Campo textual necesario para mostrar el contenido del documento, contiene el tipo específico del archivo original del documento.
fecha_publicado	Fecha	---	Requerido	Fecha en que autorizó la publicación del documento.
hora_publicado	Hora	---	Requerido	Hora en que autorizó la publicación del documento.
publicado	Booleano	1	Requerido, Valor Inicial: Falso	Registro que indica si el documento ya ha sido publicado.
habilitado	Booleano	1	Requerido, Valor inicial: Verdadero	Registro que indica si la fila actual se encuentra habilitada para poder ser procesada.

Tabla 4.6 → Tabla "autor" de la base de datos

Cuando el autor obtenga los tres votos aprobatorios de sus jueces y la aceptación de algún coordinador, entonces se utilizará el mismo registro de la tabla *"autor"* para que el documento sea publicado por el público en general; para ello

simplemente se consultarán los registros que tengan un valor *"verdadero"* en el campo *"publicado"* de la misma tabla.

Cuando la propuesta del autor forme parte de los registros en la tabla *"autor"*, los miembros de su misma especialidad podrán postularse como posibles jueces de dicho documento; para ello deberán registrarse en el sistema como jueces potenciales de dicho tema, para ello se ha creado la tabla *"juzgado"*, que está descrita en la ***tabla 4.7.***

Nombre del Campo	Tipo	Tamaño	Características	Descripción
código	Texto	12	**Clave Primaria,** Requerido	Código alfa-numérico único e indicador del código original en la tabla *"miembros"* del posible juez del tema.
no_tema	Entero	7	**Clave Secundaria,** Requerido	Llave principal auto-numérica e indicadora del número de tema al que se hace referencia.
peticion	Objeto Binario Mediano (MediumBlob)	---	Requerido	Petición del juez para poder ser juez del documento al que se hace referencia. El coordinador será el único usuario que tendrá acceso a esta información.
fecha_peticion	Fecha	---	Requerido	Fecha en que realizó la petición.
hora_peticion	Hora	---	Requerido	Hora en que realizó la petición.
habilitado	Booleano	1	Requerido, Valor inicial: Verdadero	Registro que indica si la fila actual se encuentra habilitada para poder ser procesada.

Tabla 4.7 → Tabla "juzgado" de la base de datos

Cuando el coordinador ha revisado la petición del posible juez, éste deberá asignar dos jueces para que puedan revisar el documento, posteriormente, cuando el documento ya tenga asignados a los dos primeros jueces, el coordinador procederá a asignar como tercer juez a un coordinador, es decir, asignará a un miembro de la mesa directiva para que pueda coordinar la evolución del documento hasta que pueda ser aprobado y liberado para su publicación.

Cuando el coordinador admita la petición de un posible juez para formar parte del jurado de un documento, se creará un nuevo registro en una nueva tabla llamada *"juez"* a partir de los datos de la petición de la tabla *"juzgado"*, dentro de la nueva tabla *"juez"* se almacenará la información del juez junto con sus comentarios finales para que puedan ser vistos cuando la publicación sea aprobada y liberada para ser accedida por el público en general. En la ***tabla 4.8*** se muestran los registros de dicha tabla.

Nombre del Campo	Tipo	Tamaño	Características	Descripción
código	Texto	12	**Clave Primaria,** Requerido	Código alfa-numérico único e indicador del código original en la tabla *"miembros"* del juez.
no_tema	Entero	7	**Clave Secundaria,** Requerido	Llave principal auto-numérica e indicadora del número de tema al que se hace referencia.
fecha_alta	Fecha	---	Requerido	Fecha en que el juez fue asignado al tema.
hora_alta	Hora	---	Requerido	Hora en que el juez fue asignado al tema.
fecha_aprobacion	Fecha	---	Requerido	Fecha en que el juez aprobó el tema para su publicación.
hora_aprobacion	Hora	---	Requerido	Hora en que el juez aprobó el tema para su publicación.
aprobacion	Booleano	1	Requerido, Valor inicial: Falso	Registro que indica si el documento al que se hace referencia ha sido aprobado por el juez.
comentarios_finales	Objeto Binario Mediano (MediumBlob)	---	Requerido	Comentarios finales del juez que serán requeridos cuando el documento haya sido aprobado y aparecerán cuando el mismo sea publicado.

Nombre del Campo	Tipo	Tamaño	Características	Descripción
habilitado	Booleano	1	Requerido, Valor inicial: Verdadero	Registro que indica si la fila actual se encuentra habilitada para poder ser procesada.

Tabla 4.8 → Tabla "juez" de la base de datos

Mientras un documento se encuentra en discusión, el autor deberá recibir los comentarios, sugerencias y correcciones de los jueces para que pueda realizar los cambios pertinentes sobre el documento e intercambiarlo por la versión anterior; como se ha mencionado anteriormente, el documento cambiará de versión automáticamente aumentando una décima de número a la versión actual y de esta forma existirá un ciclo finito de comentarios y correcciones entre el autor y los jueces.

Para que esta retroalimentación de información exista, se ha creado la tabla *"foro"*, la cual funge, como su nombre lo indica, como un foro de discusión interrelacionado con los documentos y sus respectivos temas. En el foro, los jueces ingresaran sus comentarios o correcciones esperando una respuesta del autor, así como sus respectivos cambios sobre el documento. En la ***tabla 4.9*** se puede apreciar la estructura de la tabla *"foro"*.

Nombre del Campo	Tipo	Tamaño	Características	Descripción
key_foro	Entero	7	**Clave Primaria,** Requerido, No Duplicado, Auto-numérico	Llave principal auto-numérica e indicadora del número del actual foro.
juez	Texto	12	**Clave Secundaria,** Requerido	Código alfa-numérico único e indicador del código original en la tabla *"miembros"* del juez.
autor	Texto	12	**Clave Secundaria,** Requerido	Código alfa-numérico único e indicador del código original en la tabla *"miembros"* del autor.

Nombre del Campo	Tipo	Tamaño	Características	Descripción
no_tema	Entero	7	**Clave Secundaria,** Requerido	Llave secundaria indicadora del número de tema al que se hace referencia.
fecha_mensaje	Fecha	---	Requerido	Fecha en que el juez ingreso el mensaje al autor.
hora_mensaje	Hora	---	Requerido	Hora en que el juez ingreso el mensaje al autor.
mensaje	Objeto Binario Mediano (MediumBlob)	---	Requerido	Mensaje del juez al autor; podrá incluir en este campo cualquier tipo de sugerencia o corrección.
fecha_respuesta	Fecha	---	Requerido	Fecha en que el autor dio respuesta al juez.
hora_arespuesta	Hora	---	Requerido	Hora en que el autor dio respuesta al juez.
respuesta	Objeto Binario Mediano (MediumBlob)	---	Requerido	Respuesta del autor al juez; para la respuesta no es requerido que el usuario realice algún cambio sobre el documento.
respondido	Booleano	1	Requerido, Valor inicial: Falso	Registro el mensaje del juez ha tenido respuesta del autor.

Tabla 4.9 → Tabla "foro" de la base de datos

Los primeros campos de la tabla *"foro"* existen como índices para poder ubicarlos rápidamente cuando el autor o el juez de algún documento accedan al sitio para realizar alguna transacción. Tanto la tabla "foro", como las demás tablas, guardan relaciones muy estrechas entre sí, y es a través de ellas que se podrán realizar las transacciones necesarias para el correcto funcionamiento del sistema; en la ***figura 4.8*** se puede estudiar las relaciones que existen entre las tablas y el tipo de concordancia que existe entre una y otra a través de sus campos.

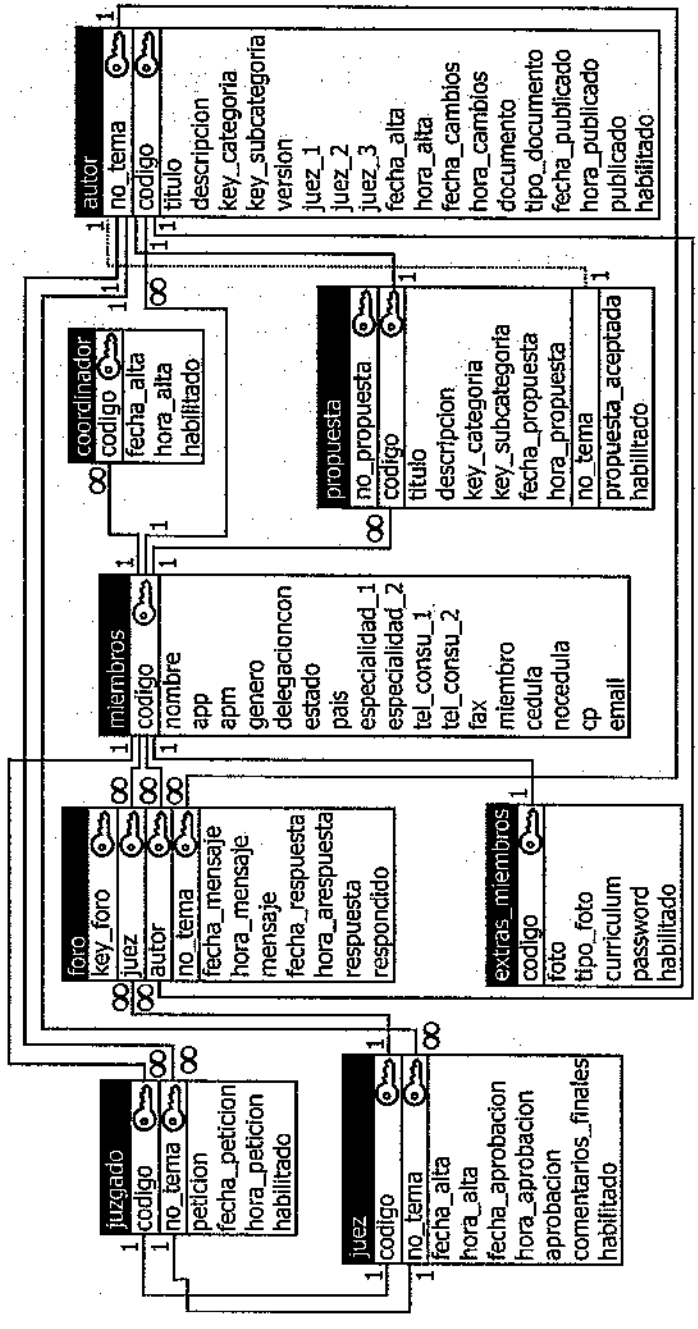


Figura 4.8 → Relaciones entre las tablas de la base de datos

La ***figura 4.8*** muestra como es que se deben relacionar las tablas para crear el campo de trabajo en que la biblioteca podrá realizar consultas y guardar toda la información necesaria para que el sistema pueda funcionar como es requerido.

4.6.2 Bitácora de Operaciones

Una parte importante dentro de los sistemas informáticos es el llevar la bitácora de las transacciones que se han realizado a lo largo de la vida de un sistema o simplemente guardar la información de las actividades que se han realizado durante una sesión del mismo. La primera propuesta para crear una bitácora para la biblioteca es crear una serie de tablas que sean actualizadas cada vez que el usuario realice una operación dentro del sitio; para ello se ha creado la tabla *"log"*, la cual llevará el control de todo lo que pasa en el sitio. La ***tabla 4.10*** muestra la estructura básica de dicha tabla.

Nombre del Campo	Tipo	Tamaño	Características	Descripción
key_log	Entero	7	**Clave Primaria**, Requerido, No Duplicado, Auto-numérico	Llave principal auto-numérica e indicadora del número de log.
key_rol	Entero	1	Requerido	Llave numérica indicadora del rol del usuario que ha generado la acción.
código	Texto	12	Requerido	Código alfa-numérico único e indicador del código original en la tabla *"miembros"* del usuario que generó la acción.
fecha	fecha	---	Requerido	Fecha en que se realizó la acción.
hora	hora	---	Requerido	Hora en que se realizó la acción.
key_accion	Entero	3	Requerido	Llave numérica indicadora de la acción que ha realizado el usuario.
key_foro	Entero	7	No Requerido	Llave numérica indicadora del número de foro al que se hace referencia si la acción realizada por el usuario corresponde al foro.

Nombre del Campo	Tipo	Tamaño	Características	Descripción
no_tema	Entero	5	No Requerido	Llave numérica indicadora del número de tema al que se hace referencia si se ha realizado alguna acción referente al documento.

Tabla 4.10 → Tabla "log" de la base de datos

Cada uno de los scripts o programas tendrán un segmento de código o una instancia a una clase que estará destinada para conocer que usuario es el que está accediendo a dicho procedimiento, su rol dentro del sitio y con estos datos podrá actualizar la tabla *"log"* ingresando un número en el campo *"key_accion"*, el cual hace referencia a una segunda tabla llamada *"accion"*, la cual tendrá registrada cada una de las posibles operaciones que realiza cada uno de los scripts o programas del sistema. Si se actualizara o se agregara una nueva función, programa o script, será obligatorio registrar en esta tabla la función que realiza y agregar en el segmento de la cabecera que genera el *log* el número de dicha acción. En la ***tabla 4.11*** se pueden revisar los campos de la tabla *"accion"*.

Nombre del Campo	Tipo	Tamaño	Características	Descripción
key_accion	Entero	3	**Clave Primaria**, Requerido, No Duplicado, Auto-numérico	Llave principal auto-numérica e indicadora del número de la acción.
accion	Texto	100	Requerido	Descripción de la acción.
habilitado	Booleano	1	Requerido, Valor inicial: Verdadero	Registro que indica si la fila actual se encuentra habilitada para poder ser procesada.

Tabla 4.11 → Tabla "accion" de la base de datos

Una tabla muy similar a la anterior deberá servir para conocer el rol que juega el usuario cuando accede a un script. Es importante resaltar que un mismo usuario podrá ser autor y juez al mismo tiempo, sin embargo, como lo que interesa dentro de este segmento es saber que acción esta realizando en un momento preciso, no basta con guardar que usuario es el que realizó la acción, ya que si fuera de esta forma lo único que se debiera registrar es el código del miembro; sin embargo, es

importante conocer el rol en el cual está generando una acción, y como depende del script o función al que está accediendo es importante que la cabecera del script pueda registrar el rol que está utilizando el usuario en ese momento, ya que de esta forma no es posible realizar dos diferentes acciones al mismo tiempo, pero no impide que pueda tomar un rol diferente a lo largo de una sesión. Para dicho propósito de ha creado la tabla *"rol"*, la cual hara referencia al rol actual que está utilizando el usuario, y de esta forma el script tendrá como bandera de rol un número y no una descripción, además de poder agregar más roles y permisos a lo largo de la vida y evolución de la aplicación. La ***tabla 4.12*** tiene los registros de la tabla *"rol"*.

Nombre del Campo	Tipo	Tamaño	Características	Descripción
key_rol	Entero	2	**Clave Primaria,** Requerido, No Duplicado	Llave principal indicadora del número de rol.
rol	Texto	20	Requerido	Nombre del rol.
habilitado	Booleano	1	Requerido, Valor inicial: Verdadero	Registro que indica si la fila actual se encuentra habilitada para poder ser procesada.

Tabla 4.12 → Tabla "rol" de la base de datos

Las relaciones que se generan a través de la tabla *"log"* y sus respectivos tipos son ilustradas en la ***figura 4.9:***

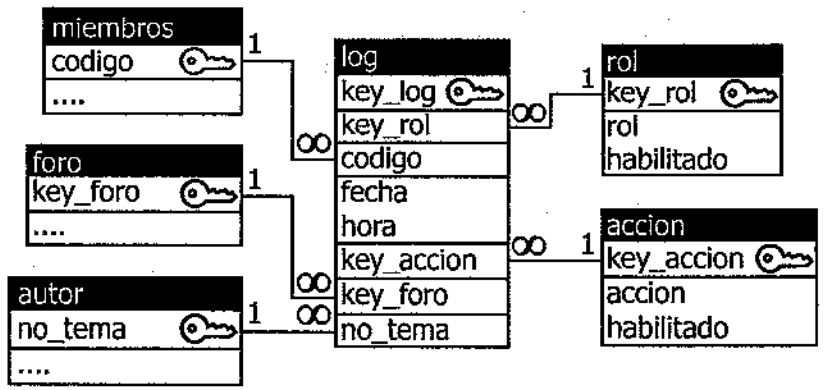


Figura 4.9 → Relaciones generadas a través de la tabla "log"

La bitácora de operaciones registra todas las acciones de los diferentes usuarios agregando el rol que ha tomado para utilizar los diferentes programas a los que tiene acceso dependiendo del mismo. La bitácora enlaza las diferentes tablas para tener acceso a mucha información que podría ayudar a la toma de decisiones acerca del funcionamiento del sistema, los plazos de tiempo tomados para realizar una acción, la forma en que el usuario utiliza los programas del sistema, y un sin fin de cosas que pueden mejorar y saber que es lo que está pasando en la biblioteca, así como tener un historial completo de las operaciones de todos y cada uno de los usuarios dentro del sistema.

4.6.3 Sistema de Autenticación y Seguridad

El sistema de seguridad se basa en un esquema de autenticación de usuarios y en sesiones de Internet. El sistema de autenticación requiere que el usuario proporcione, a través de un formulario estándar, su nombre de usuario y su password; posteriormente esta información debe ser ***encriptada*** para poder ser procesada para prevenir el riesgo se ser vista sin autorización por algún atacante en la red, Se puede observar el esquema básico del sistema de seguridad en la ***figura 4.10.***

Cuando el usuario ya se ha identificado como un usuario válido, se le creará una sesión. Las sesiones en PHP son una forma de conservar ciertos datos a lo largo de los subsiguientes accesos al sitio, es decir, conservar cierta información durante la navegación dentro del sitio, una vez que el usuario cierre la sesión desde el sitio, cierre el explorador o termine el tiempo de duración de la sesión, la sesión terminará y todos los datos contenidos dentro de la misma serán borrados.

La información que se encuentra guardada dentro de la sesión, se conserva a través de la navegación por ***cookies*** o guardada en la cabecera del navegador si este no soporta las *cookies* o las tiene deshabilitadas para mayor protección, esto es porque de cierto modo, las *cookies* son una forma insegura de guardar y conservar información confidencial del usuario, ya que podrán ser accesadas por algún ***hacker*** de la red. Para evitar esto, la información guardada dentro de la sesión deberá estar *encriptada* para evitar que la persona que acceda a esta, descifre su contenido y tome provecho de la misma.

El método de encriptación usado por el sistema se llama ***MD5.*** Este método utiliza un algoritmo de encriptación llamado ***hash.*** *Hash* es un algoritmo que genera un valor *hash* de algún dato, como una clave de mensaje o de sesión en este caso.

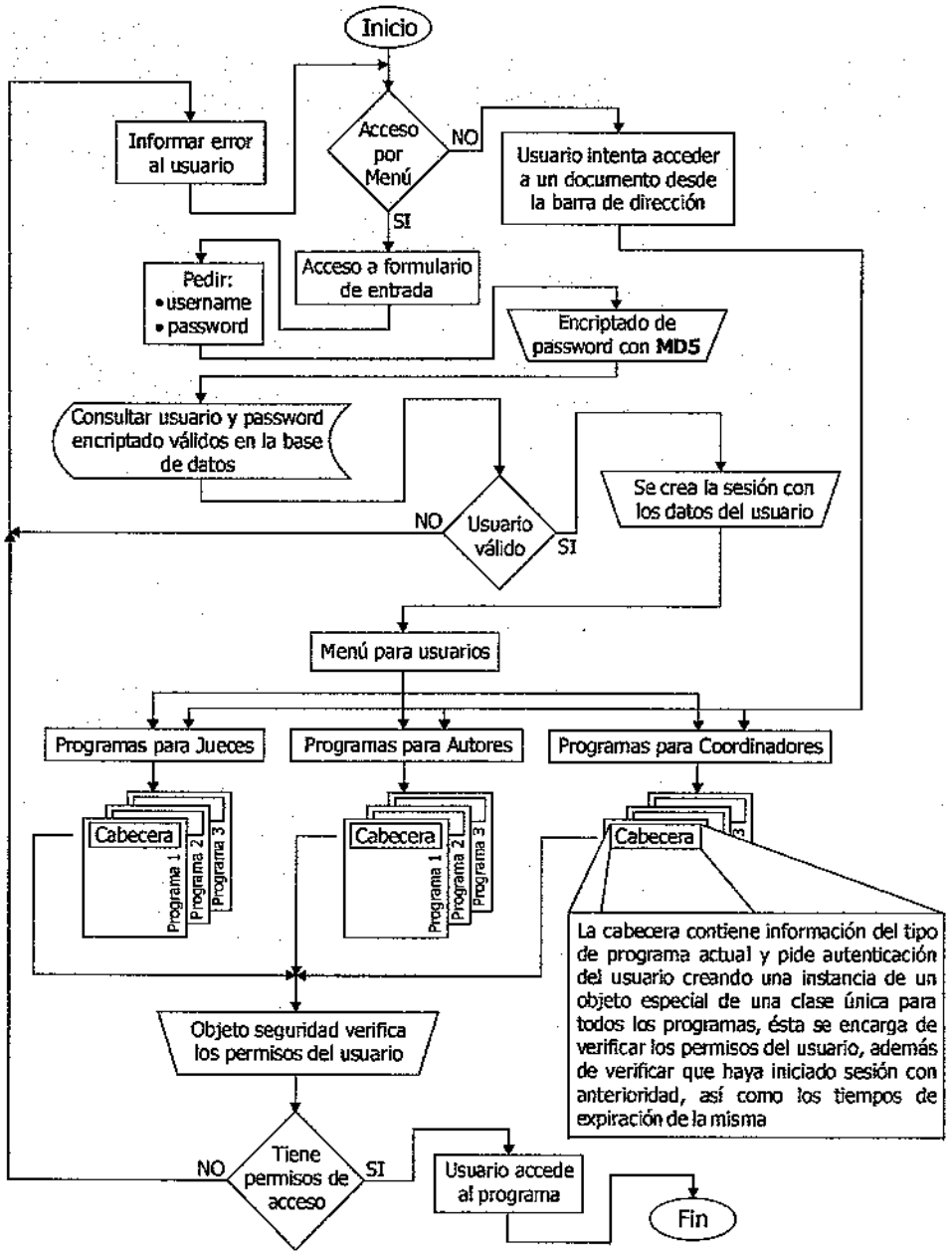


Figura 4.10 → Esquema de seguridad para la biblioteca

Con un algoritmo de *hash*, los cambios que se produzcan en los datos de entrada pueden cambiar todos los bits del valor *hash* resultante, por lo que estos valores son útiles para detectar cualquier modificación en un objeto de datos, como un mensaje o una cadena de datos. Además, un algoritmo de *hash* hace que sea computacionalmente imposible crear dos entradas independientes que tengan el mismo valor *hash*. Los algoritmos de *hash* más comunes son *MD2, MD4, MD5* y *SHA-1*. Estos algoritmos también se llaman funciones de *hash;* básicamente un esquema de *hash* es un método de transformación de datos (como una cadena que incluya una contraseña) en el que el resultado es único y no se puede devolver a su forma original.

El algoritmo ***MD5*** es un esquema de *hash* normalizado unívoco de 128 bits desarrollado por ***RSA Data Security, Inc.*** y utilizado por varios proveedores de *Protocolo Punto a Punto (PPP)* para autenticación cifrada, incluyendo servidores de Internet basados en PHP. De esta forma, se puede enviar al servidor o al cliente información cifrada de forma segura a través de la red.

Dentro de todos los programas incluidos en el sistema, se agregará como parte de la plantilla, una cabecera corta especial que hará referencia a través de una instancia a una clase especial denominada "seguridad" que se encargará de autenticar al usuario cada vez que se desee acceder al contenido de una página antes de ser mostrada. La cabecera podrá pues, requerir de la clase "seguridad" la aprobación de acceso del usuario que intenta acceder al programa, si el usuario no tiene los permisos de acceso, entonces será la clase la que lo redireccione del programa al inicio y se pueda avisar que efectivamente no posee los permisos de acceso. En el caso de que el usuario no haya iniciado sesión con anterioridad, será la misma instancia a la clase la que se encargue de sacarlo y no permitirle el acceso a los programas incluidos dentro del sitio.

Al ser la cabecera parte de la plantilla de diseño de las páginas, no habrá un solo *script* que no contenga el esquema de seguridad, y como estándar la cabecera estará contenida por un máximo de cuatro líneas, no presentará mayor problema para el programador su implementación, ya que los únicos cambios que presentará dicha cabecera entre uno y otro *script* será los permisos de acceso para los diferentes roles de los usuarios, es decir, el programador deberá incluir en la cabecera de cada documento del sitio que tipo de usuarios tienen acceso al mismo, ya sea un autor, un juez, todos los usuarios o únicamente el coordinador.

4.6.4 Módulo de Autor

Aunque los usuarios acceden a un único sistema de autenticación a través de un formulario de entrada, los usuarios deben ser dirigidos automáticamente al menú del rol del tipo de usuario asignado; este menú consiste básicamente en hipervínculos creados dinámicamente en forma de listado que conforman todas las opciones que puede realizar el usuario. Esto quiere decir que depende de las acciones que haya realizado con anterioridad dentro del sistema las opciones que se mostrarán en dicho listado.

Para conocer las opciones y los programas a los que el usuario puede acceder se deberán dividir las opciones que pueden utilizar los diferentes usuarios dependiendo del rol del mismo. Básicamente se pueden dividir los roles en tres módulos, el del autor, el del juez y por último el del coordinador.

El autor forma parte fundamental dentro del esquema de usuarios del programa, ya que es éste el que realiza las publicaciones de los diferentes documentos que serán juzgados y aprobados por un jurado que será asignado por el coordinador. El autor realiza, en primera instancia, una propuesta de publicación, que será revisada y autorizada por el coordinador y será entonces que el autor podrá publicar la primera versión de su documento. Posteriormente podrán postularse los diferentes jueces que deberán cumplir básicamente con un único requerimiento, y este se refiere a que los jueces deberán tener la misma especialidad médica que el autor, ya que ésta será una forma de generar una publicación profesional y que será revisada y autorizada únicamente por médicos especializados en el tema y no por cualquier persona. Para ello el autor deberá situar su documento en el esquema de categorías que el coordinador podrá crear a partir de las peticiones de los mismos usuarios del sistema.

Las categorías serán administradas automáticamente a través de registros y tablas especiales dentro de la base de datos y será a través de ellas que se podrán crear las diferentes clases de documentos y las categorías para las diferentes especialidades médicas y dermatológicas.

Para poder conocer el proceso completo de la adición de los documentos y los programas que deberán existir para que el autor pueda utilizar el sistema de publicación y que dichos documentos sean juzgados y autorizados para su liberación y escrutinio de los lectores finales, se ha creado el diagrama de flujo que se muestra en la ***figura 4.11,*** que indica de manera detallada los procesos y programas que el autor deberá utilizar para realizar sus publicaciones dentro del sistema de la biblioteca.

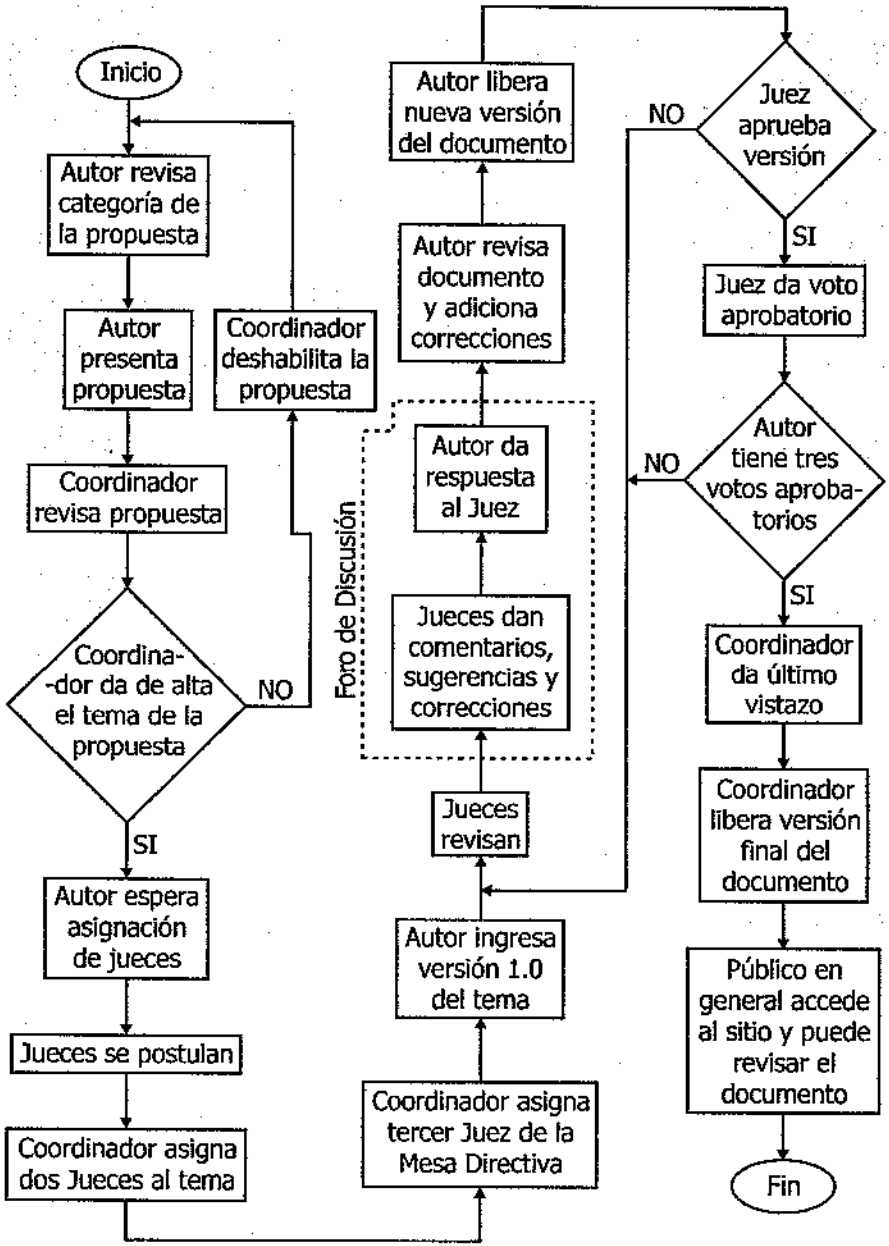


Figura 4.11 → Diagrama de flujo del módulo de Autor

El autor entra en un ciclo de correcciones y adiciones antes de que su documento pueda ser liberado al público en general; para ello deberá contar con los tres votos aprobatorios de todos los jueces que se encuentren revisando el documento. Como se muestra en la parte superior derecha del diagrama de la ***figura 4.11***, este ciclo permite al sistema llevar también el control de las versiones de los diferentes documentos; para conocer el número de cambios registrados de un documento el sistema aumenta una décima de número a la versión del documento para conocer este dato.

Para que el autor pueda conocer los comentarios, sugerencias y cambios requeridos por los diferentes jueces, entrará a un programa especial que consiste básicamente en un foro de discusión en el que sólo podrán entrar los médicos especialistas involucrados con el documento en cuestión. La sección específica del foro de discusión se encuentra enmarcada por un recuadro de una línea de corte al centro del diagrama de flujo de la ***figura 4.11***. Cuando los usuarios involucrados en una publicación accedan al sistema, podrán entrar al foro de sus documentos a juicio o, en el caso de los autores, sus publicaciones y agregar sus comentarios o dar respuesta a los mismos, estando el autor obligado a dar la contestación y realizar también los cambios pertinentes a su publicación.

Es el autor el usuario encargado de crear las publicaciones que permanecerán a lo largo de la vida del sistema y será los jueces los que llevarán la responsabilidad de que lo que se esté publicando contenga la información más actual, efectiva y auténtica; actuando con ética y profesionalismo cuando requieran los cambios que el autor deberá agregar al documento para que esto sea posible.

4.6.5 Módulo de Juez

El juez es el usuario del sistema que se encarga de revisar y corregir los documentos que serán publicados en el sitio por los autores. El juez se integra como parte de un juzgado que será asignado por un coordinador y podrán serlo siempre y cuando sean de la misma especialidad del autor que publica el documento; este punto será el requerimiento más importante para poder postularse como juez de una publicación, ya que los médicos especialistas en el genero del trabajo deben ser los únicos capaces de avalar lo que el documento contenga.

Además de llevar la responsabilidad de corregir el contenido de la investigación del autor, el juez podrá hacer sugerencias de la adición en algún contenido del documento, siempre y cuando, el autor lo autorice.

En la ***figura 4.12*** se puede observar los programas y procesos que el juez deberá seguir para poder juzgar una publicación dentro del sistema a través del diagrama de flujo del apartado del juez.

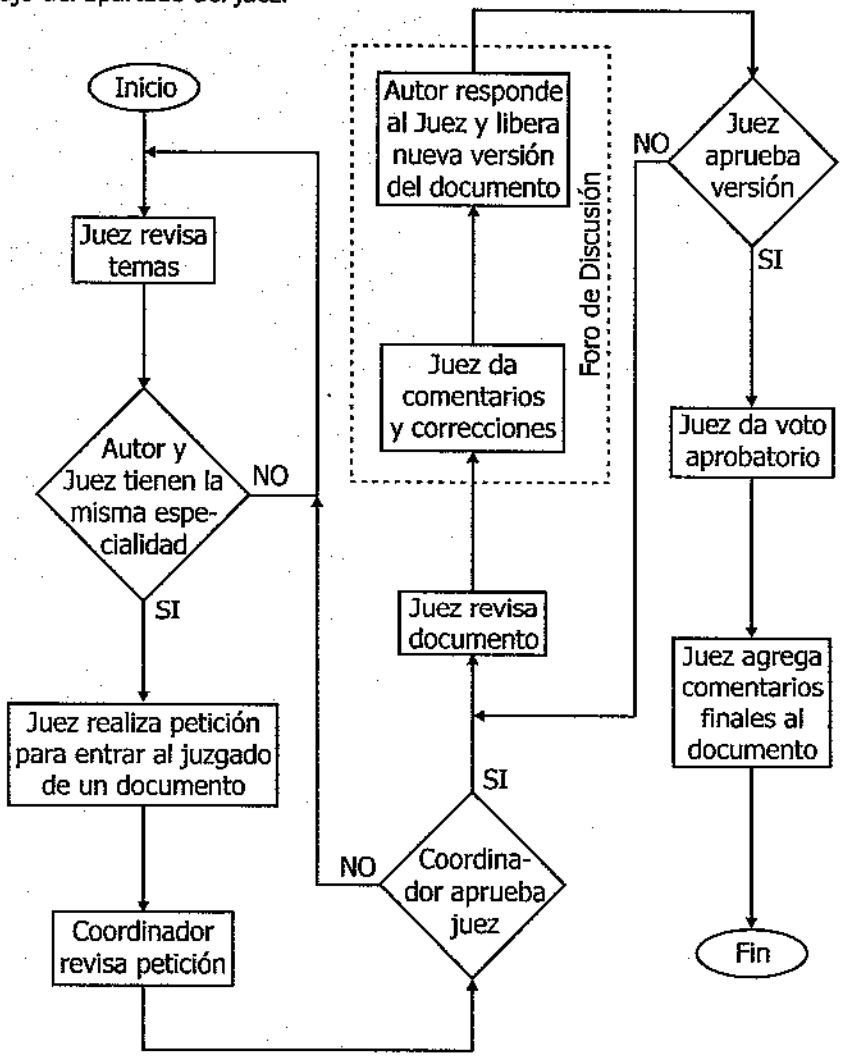


Figura 4.12 → Diagrama de flujo del módulo de Juez

Cuando el coordinador ha aceptado la postulación del juez para formar parte del juzgado de una publicación, el juez podrá entrar al sistema y dar sus comentarios, sugerencias y correcciones al autor a través de un foro de discusión o simplemente

enviando un correo electrónico al autor para que él pueda realizar las correcciones y adiciones pertinentes. Tanto el autor y los jueces involucrados en el proyecto serán los únicos usuarios que podrán entrar al foro de discusión destinado exclusivamente para dicha publicación. Dentro del foro serán accesibles todos los comentarios de los jueces y las respuestas emitidas por el autor para que puedan formar parte del foro y cada uno pueda ser accedido no solo por el usuario que lo creo, sino por todos los involucrados. De esta forma es posible generar un apartado que contenga la información necesaria para que el proyecto crezca y genere un estudio profundo y completo en beneficio final de los pacientes de todos los médicos que lo puedan leer y puedan revisar los últimos avances en lo que al tema se refiere.

Es trascendental que dentro del juzgado exista un miembro que pueda fungir como mediador del proyecto, y es por esto que es de suma importancia que uno de los jueces forme parte de la mesa directiva de la AMD, ya que él será el que represente los ideales de la AMD y será la autoridad representativa pertinente de la mesa directiva.

Cuando cada uno de los jueces considere que el documento se encuentra completamente terminado y corregido podrá dar su voto aprobatorio para que pueda ser liberado, no sin antes agregar sus comentarios finales del mismo. Estos comentarios podrán ser accedidos por los lectores cuando la publicación sea liberada al público en general por un coordinador cuando el documento obtenga los tres votos aprobatorios de los jueces.

4.6.6 Módulo de Coordinador

El coordinador funge como el súper usuario del sistema; tiene permisos de acceso a todos los módulos y es el único que puede dar autorización para habilitar o deshabilitar los diferentes proyectos de cualquier tipo de publicaciones. El único requisito de los coordinadores es que deben ser de la mesa directiva de la AMD; este requisito garantiza que el coordinador cuente con la preparación necesaria para poder llevar a cabo cualquier labor dentro del sistema con profesionalismo, ética y que sus decisiones se llevarán a cabo bajo los ideales de la institución.

Para poder ver con amplitud todo lo que el coordinador puede y debe realizar se ha creado el diagrama de flujo de la ***figura 4.13,*** en la cual es posible estudiar a detalle todas las funciones que el coordinador deberá regular cuando acceda por alguno de los caminos que el sistema tiene previstos como labores que el coordinador deberá realizar dentro del sistema.

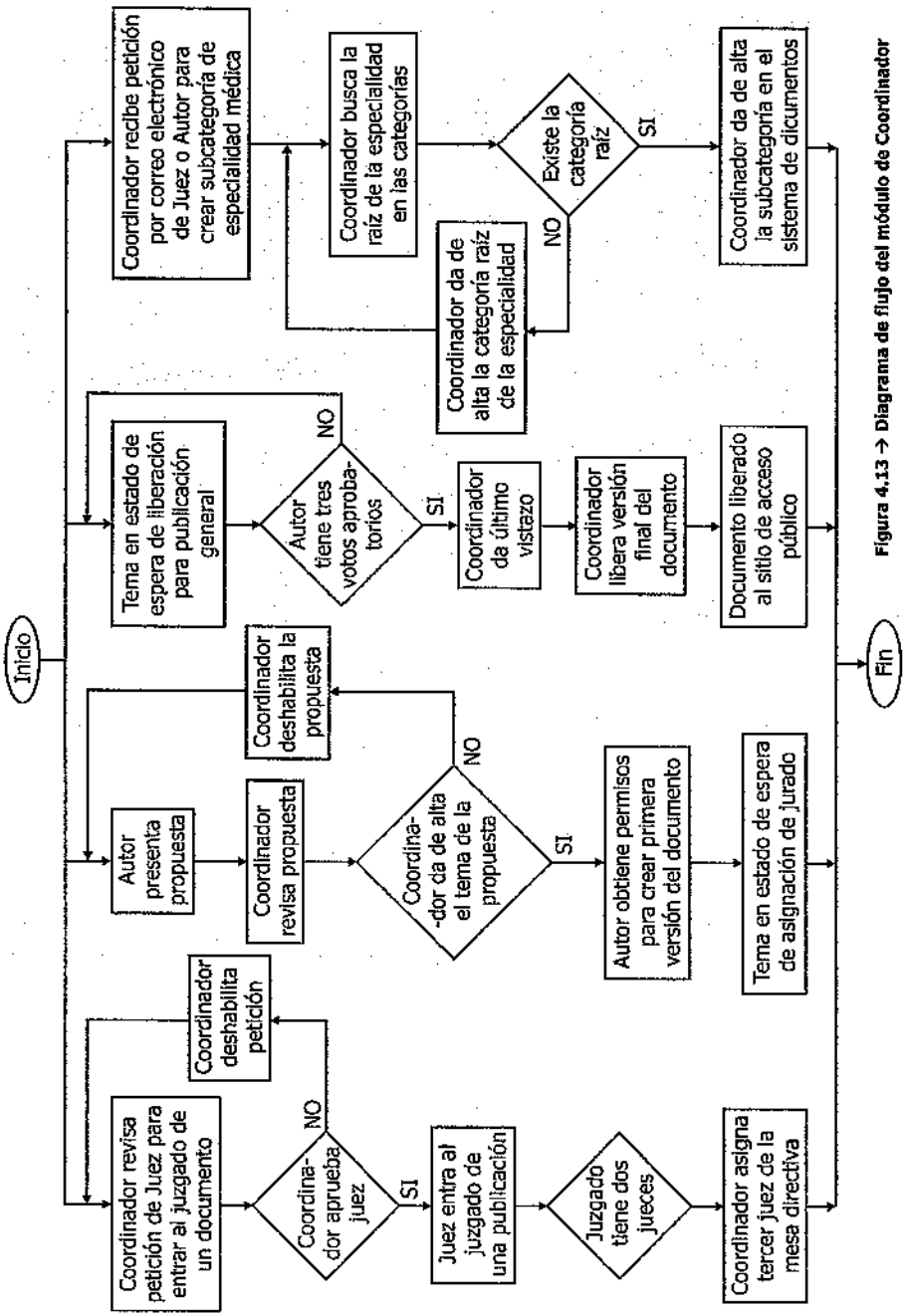


Figura 4.13 → Diagrama de flujo del módulo de Coordinador

El coordinador realiza las autorizaciones finales para que un Juez o un Autor puedan realizar sus funciones dentro del sistema; El coordinador tiene la última palabra en cuanto a que o quién puede realizar alguna publicación o formar parte de un juzgado. El coordinador es el usuario que da el último vistazo a una documento antes de que sea liberado al público en general, y es por esto que el coordinador juega uno de los papeles más importantes dentro del sitio, ya que será la persona que se encargue de delegar las responsabilidades de quienes son los médicos especialistas que podrán juzgar de la mejor manera dicho documento después de haberlo aprobado de entre todas las propuesta dejadas por los posibles autores.

Cuando un autor pretende realizar una publicación, uno de los primeros pasos a seguir es encontrar la subcategoría adecuada para su documento; dicha categoría, como se mencionó anteriormente, será registrada dentro del sistema de Bases de Datos, sin embargo, el Autor no posee los permisos necesarios para realizar dicha acción. El único usuario con los permisos para crear una subcategoría o una categoría raíz es el Coordinador.

Cuando el Autor o un Juez requieren que exista alguna subcategoría o una categoría raíz de una especialidad médica o dermatológica deberá hacer una petición por correo electrónico dirigida a cualquier Coordinador para que él pueda estudiar dicha petición y evaluar dar de alta la categoría para que el documento pueda ubicarse en la especialidad requerida y será en ésta misma donde el usuario final acceda para poder leerlo.

Es así que el Coordinador provee el servicio de regulación dentro del sitio con lo cual concluiría el estudio del análisis estructurado para el sistema de la biblioteca virtual de publicaciones dermatológicas.

El propósito principal del programa será automatizar la administración de un sitio que incluye un sistema de publicaciones de documentos previamente reconocidos por un juzgado asignado por un coordinador que controlará lo que se publica en el sitio desde la comodidad de su hogar u oficina a través de un sitio de Internet hecho con el único propósito de compartir la información y experiencias de un grupo conformado por los mejores especialistas dermatólogos del país, los cuales, integran y conforman de la AMD.

Para concluir esta sección es importante destacar que lo expuesto a lo largo de este capítulo es una propuesta para un sistema de publicaciones dermatológicas que gracias a dicha investigación será posible juzgar su viabilidad por la mesa directiva de la AMD dadas las bases para considerar su futuro desarrollo e implementación.

Conclusión

La Academia Mexicana de Dermatología, A.C. es una institución dedicada a promover el mejor conocimiento de la dermatología entre la comunidad médica y especialista del país; la inclusión de un sistema informático para mejorar el control de sus archivos ha significado un notable cambio en la forma en que se administra la información de sus miembros y sus respectivos historiales de asistencia a los diferentes eventos organizados a lo largo y ancho de nuestro país.

La mejoría en los contenidos de su página de internet ha significado tener un mayor acercamiento con sus miembros y las personas que desean acercarse a la institución para poder conocerla, contactar a sus miembros y saber cuáles son sus próximos eventos.

Sin embargo el propósito de este trabajo no se limita a dar a conocer los proyectos desarrollados dentro de esta institución, sino dar una perspectiva de como es que los sistemas de cómputo pueden influir en el modo de administración de cualquier tipo de empresa sin importar el rubro al que se dedique. Sin duda alguna, una planeación dedicada influye mucho en el resultado final de un proyecto informático, y el análisis previo de los proyectos desarrollados a lo largo de este trabajo demuestra que un sistema informático en el cual todas sus partes interactúan entre sí pueden mejorar la forma en que se trabaja y crear un marco que puede influir de forma dramática la toma de decisiones de una empresa o institución a través de la adaptación de metodologías específicas de sistemas informáticos y a la experiencia obtenida en el desarrollo de las versiones anteriores.

La creación de un sistema automático de documentación de publicaciones dermatológicas no se debe limitar únicamente para dar a conocer informes médicos, esta es una forma de esparcir cualquier tipo de conocimiento humano a través del medio informático más comúnmente usado por la gente, la internet.

Existen actualmente programas que controlan publicaciones a través de páginas de internet, sin embargo, este desarrollo demuestra que no se debe invertir mucho para poder crear un sistema personalizado para que las gente común de a conocer sus experiencias y conocimientos a sus diversas comunidades de forma sencilla y desde la comodidad de sus hogares u oficinas.

La propuesta de la biblioteca virtual es un proyecto que existe con el único propósito de que la AMD pueda deliberar su viabilidad y considerar su futuro desarrollo a través del estudio y análisis que ha sido expuesto por completo a lo largo del último capítulo de este trabajo.

En síntesis de los párrafos anteriores, todos los programas expuestos a lo largo de este libro han sido concluidos satisfactoriamente y a través de ellos se han cubierto las necesidades y requisitos del cliente.

Tanto el sistema de información como la página de internet se encuentran en completo funcionamiento, estos sistemas interactúan de manera independiente y se complementan a través de la misma base de datos, por lo tanto, el objetivo de análisis, desarrollo y creación de dichas aplicaciones ha sido cubierto completamente.

Una de las partes más importantes de este trabajo es el apartado para el programador avanzado de Visual Basic .Net, ya que los lector con conocimientos intermedios en programación podrán acceder a esta sección y conocer la mejor forma de poder realizar aplicaciones profesionales para esta plataforma .Net y al mismo tiempo, saber como se hizo el sistema de información ***AcadMexDerm .Net.***

A través de este apartado se ha alcanzado uno de los objetivos principales del libro, el cual consiste en aportar un extra a un trabajo que no sólo se enfoca en las necesidades de la AMD, sino que es posible recurrir a él como un compendio de programas desarrollados a través de ejemplos sencillos en los cuales el lector encontrará la mejor forma de realizar una migración exitosa de sus sistemas programados en Visual Basic 6 a la plataforma .Net Framework de manera óptima y profesional.

De la misma forma el programador que no conozca la plataforma .Net podrá encontrar en este libro un estudio completo de lo que ofrece .Net Framework a los programadores para sus desarrollos y como puede emplear ADO .Net en sus programas de Visual Basic .Net.

Glosario

.Net Framework: Infraestructura de programación diseñada para simplificar el desarrollo de sistemas creada por Miscrosoft para construir, implementar y crear aplicaciones y servicios web.

A

Active X: Conjunto de tecnologías que permite a los componentes de software interactuar entre sí en un entorno de red, con independencia del lenguaje en que se hayan creado.

ADO: Acrónimo de ActiveX Data Objects, interfaz para proporcionar servicios de acceso a datos.

ADO .Net: Interfaz para proporcionar servicios de acceso a datos en la plataforma .Net.

API: Acrónimo de Application Programming Interface o Interfaz de Programación de Aplicaciones, Conjunto de rutinas que utiliza una aplicación para solicitar y realizar servicios de bajo nivel efectuados por un sistema operativo del equipo. Estas rutinas generalmente llevan a cabo tareas de mantenimiento como administrar archivos y mostrar información.

Archivo: Conjunto de información que agrupa instrucciones, números, palabras o imágenes en unidades coherentes de bits que el ordenador puede recuperar, modificar, eliminar, guardar, interpretar o enviar a un dispositivo de salida y que es identificado con un nombre y extensión.

Archivo Secuencial: Archivo cuyos registros están organizados en el orden en que se colocaron en el mismo.

ARPANET: Acrónimo de Advanced Research Projects Agency Network o Red de la Agencia de Proyectos de Investigación Avanzada, red de computadoras creada en la década de 1960, desarrollada por el departamento de defensa de Estados Unidos.

ASCII: Acrónimo de American Standard Code for Information Interchange (Código Normalizado Americano para el Intercambio de Información). En computación, es un esquema de codificación que asigna valores numéricos a las letras, números, signos de puntuación y algunos otros caracteres. ASCII incluye 256 códigos que representan todas las combinaciones posibles en 8 bits o un byte.

ASP: Acrónimo de Active Server Page o Página del Servidor Activa, Lenguaje de scripting de ejecución en el servidor para crear páginas Web.

ASP .Net: Conjunto de tecnologías de Microsoft .NET Framework para crear aplicaciones Web y servicios Web XML. Las páginas ASP.NET se ejecutan en el servidor y generan lenguaje de marcado (como HTML, WML o XML) que se envía a un explorador móvil o de escritorio.

Avery 5160: Nombre dado a un formato estándar de estampas distribuidas en una hoja tamaño carta.

Aztecas: Pueblo que dominó el centro y sur de México desde el siglo XIV hasta el siglo XVI y que es famoso por haber establecido un vasto imperio altamente organizado y destruido por los conquistadores españoles y sus aliados tlaxcaltecas.

B

Banner: Imagen animada o gráfico multimedia incrustado en páginas web que usualmente tiene fines publicitarios.

Base de Datos: Cualquier conjunto de datos organizados para su almacenamiento en la memoria de un ordenador o computadora, dichos datos están distribuidos en tablas y otros objetos organizados para facilitar su búsqueda y clasificación.

BCL: Acrónimo de Base Class Library o Biblioteca de Clases Base. Esta biblioteca brinda acceso a la funcionalidad del sistema y es la base sobre la que se crean las aplicaciones, los componentes y los controles de .NET Framework.

Binario: Sistema numérico basado en combinaciones de unos y ceros.

Bit: Unidad básica de almacenamiento en un ordenador o computadora.

BLOB: Acrónimo de Binary Large Object u Objeto Binario Grande, en bases de datos tipo de registro que contiene datos binarios como, por ejemplo, gráficos, sonido o código compilado.

Bool: Dato que devuelve un valor verdadero o falso.

Booleano: Expresión que se refiere a un valor verdadero o falso.

Botox: Toxina Botulínica Tipo A.

Buffer: Memoria intermedia, depósito de datos intermedio, es decir, una parte reservada de la memoria en la que los datos son mantenidos temporalmente hasta tener una oportunidad de completar su transferencia hacia o desde un dispositivo de almacenamiento u otra ubicación en la memoria.

Button: Control de los formularios Windows Forms que permite al usuario hacer clic en él para ejecutar una acción.

Byte: Conjunto de ocho bits.

C

C#: Lenguaje de programación orientado a objetos de Microsoft creado como un híbrido del lenguaje C y C++ basado en la tecnología .Net Framework.

C++: Versión orientada a objetos del lenguaje de programación denominado C, desarrollada por Bjarne Stroustrup a comienzos de la década de 1980 en los Laboratorios Bell.

ccTLD: Siglas utilizadas para designar a los nombres de dominio de primer nivel, conocidos como códigos territoriales (Country Code Top Level Domain). Dichas siglas se refieren a la clasificación en el sistema de nombres de dominio (DNS), representado por un sufijo de dos letras, asignado conforme a los códigos estándar de ISO3166-1 para la representación de nombres de países o territorios.

Check Box: Casilla de Verificación.

Check List Box: Control Windows Forms que muestra una lista de elementos, como el control ListBox y también puede mostrar una marca de verificación junto a los elementos de la lista.

Clase: Tipo de referencia que encapsula datos (constantes y campos) y programas incluyendo su comportamiento (métodos, propiedades, indicadores, eventos, operadores, constructores de instancia, constructores estáticos y destructores), y puede contener tipos anidados. Los tipos de clase admiten la herencia, un mecanismo mediante el cual una clase derivada puede extender y especializar una clase base.

Clave Principal: Columna o combinación de columnas que identifican exclusivamente a una fila de cualquier otra fila de una tabla en una base de datos.

CLI: Acrónimo de Command Line Interface o Interfase de Línea de Comandos

CLR: Acrónimo de Common Language Runtime o Entorno Común de Ejecución. Motor que es el núcleo de la ejecución de código administrado. El motor de tiempo de ejecución proporciona al código administrado servicios como integración entre varios lenguajes, seguridad de acceso a código, administración de la duración de los objetos, y compatibilidad con la depuración y la generación de perfiles.

CLS: Acrónimo de Common Language Specification o Especificación de Lenguaje Común. Subconjunto de funciones del lenguaje admitidas por el Common Language Runtime, incluyendo funciones comunes de varios lenguajes de programación orientados a objetos.

COM: Acrónimo de Model Object Component o Modelo de Objetos Componentes, modelo de programación basado en objetos y diseñado para aumentar la interoperabilidad del software. Permite a dos o más aplicaciones o componentes funcionar de manera conjunta e incluso cuando han sido creados por fabricantes diferentes, en momentos diferentes, con lenguajes de programación diferentes o se estén siendo ejecutados en equipos diferentes con sistemas operativos diferentes. Las tecnologías OLE y Active X están basadas en COM.

Combo Box: Objeto que muestra un campo de edición combinado con un ListBox y permite al usuario seleccionar una opción de la lista o escribir texto nuevo.

Consulta: Petición específica de recuperación, modificación o eliminación de registros en una base de datos.

Correo Electrónico: Medio digital que permite transmitir datos y mensajes de una computadora a otra a través de Internet.

CPU: Acrónimo de Unidad Central de Proceso, Unidad aritmético-lógica que realiza cálculos, comparaciones, y toma decisiones lógicas que interpreta y ejecuta en forma de instrucciones.

D

DataAdapter: Representa un conjunto de comandos de datos y una conexión de base de datos que se utilizan para rellenar DataSet y actualizar el origen de datos.

DataGrid: Muestra datos de ADO.NET en una cuadrícula desplazable.

DataReader: Objeto que sirve para recuperar una secuencia de datos de sólo avance y de sólo lectura desde una base de datos en memoria.

DataRelation: Representa una relación primaria-secundaria entre dos objetos DataTable.

DataRow: Representa una fila de datos en un DataTable.

DataSet: Representa una caché de memoria interna de datos recuperados de un origen de datos, representa un componente fundamental de la arquitectura de ADO.NET.

DataTable: Representa una tabla de datos en memoria.

DataView: Representa una vista personalizada que puede enlazar datos de un DataTable para ordenación, filtrado, búsqueda, edición y exploración.

DBMS: Acrónimo de Database Management System o Sistema de Administración de Bases de Datos, depósito de la colección de archivos de datos que permite a los usuarios realizar diversas operaciones sobre dichos archivos como, ejemplo, recuperar, anexar, modificar, actualizar y generar informes.

Decimal: Sistema numérico basado en el 10 y sus potencias.

Denervación: Intervención quirúrgica consistente en seccionar los filetes nerviosos que conducen la sensibilidad de una articulación.

DNS: Siglas utilizadas para referirse al Domain Name System (Sistema de Nombres de Dominio) y/o al Domain Name Server (Servidor de Nombres de Dominio) de manera indistinta. El primero es el sistema de cómputo utilizado para traducir nombres de dominio en direcciones de IP y viceversa, y el segundo corresponde a los equipos de cómputo utilizados para desempeñar la función de resolución y traducción de nombres de dominio.

Dominio: Forma simple de dirección de internet que está formado por un conjunto de caracteres (letras, números y guiones).

E

Excepción: Cualquier situación de error o comportamiento inesperado que encuentra un programa de cómputo en ejecución.

F

FTP: Acrónimo de File Transfer Protocol o Protocolo de Transferencia de Archivos, miembro del conjunto de protocolos TCP/IP que se utiliza para copiar archivos entre dos equipos en Internet.

G

GDI+: Acrónimo de Graphic Design Interface o Interfaz de Diseño de Gráficos, interfaz que permite crear gráficos, dibujar texto y manipular imágenes gráficas como si fueran objetos.

Group Box: Control de formularios Windows Forms que se utilizan para proporcionar un agrupamiento identificable para otros controles.

H

Hacker: Programador que se conecta a una red para invadir en secreto computadoras, y consultar o alterar los programas o los datos almacenados en las mismas.

Hardware: Equipo físico utilizado para el funcionamiento de una computadora.

Hash: Resultado de tamaño fijo obtenido al aplicar una función matemática unívoca (a veces llamada algoritmo de hash) a una cantidad de datos arbitraria. Si se produce un cambio en los datos de entrada, el valor de hash cambia. Se puede utilizar hash en muchas operaciones, como la autenticación y la firma digital. También se denomina síntesis del mensaje.

Hipervínculo: Imagen o texto destacado mediante un subrayado o color que direcciona al usuario a otro sector del documento o a otra página de internet.

Hosting: Servicio ofrecido por algunos proveedores que brindan a sus clientes un espacio en su servidor de internet para alojar un sitio web.

HTML: Acrónimo de (Hipper Text Markup Language o Lenguaje de Marcado de Hipertexto, Lenguaje de marcado sencillo utilizado para crear documentos de hipertexto compatibles con varias plataformas. Los archivos HTML son archivos de texto ASCII sencillos con códigos incrustados (indicados por etiquetas de marcado) que indican formato y vínculos de hipertexto.

I

IDE: Acrónimo de Interactive Development Environment o Ambiente de Desarrollo Interactivo.

IIS: Acrónimo de Internet Information Server o Servidor de Información de Internet, servicios de software que admiten la creación, configuración y administración de sitios Web para sistemas Windows.

Índice: En una base de datos relacional, objeto de base de datos que proporciona acceso rápido a los datos de las filas de una tabla, en función de valores de claves. Los índices proporcionan acceso rápido a los datos y pueden exigir la unicidad de las filas de una tabla.

Instancia: En programación, elemento que hace referencia a un objeto y accede a sus funciones o datos desde un procedimiento externo a él.

Integridad Referencial: Mecanismo de integridad que asegura que los datos vitales de una base de datos relacional no sean eliminados.

Internet: Interconexión de redes informáticas que permite a los ordenadores o computadoras conectadas comunicarse directamente a nivel mundial.

Intranet: Red privada dentro de un inmueble que utiliza el mismo tipo de software usado en el Internet público para uso interno.

IP: Acrónimo de Internet Protocol o Protocolo de Internet, nomenclatura utilizada para asignar a los equipos de cómputo una dirección numérica.

J

JAVA: Lenguaje de programación orientado a objetos desarrollado por la empresa Sun Microsystems.

Jscript: Lenguaje de secuencias de comandos orientado a objetos.

Just-In-Time (JIT compilation): Compilación que convierte el Lenguaje intermedio de Microsoft (MSIL) en código máquina cuando se requiere el código en tiempo de ejecución.

K

Kilo Byte (Kbyte): Equivalente a 1,024 bytes.

L

Label: Control de los formularios Windows Forms que se utiliza para mostrar texto o imágenes que el usuario no puede editar.

LAN: Acrónimo de Local Area Network o Red de Área Local.

List Box: El control ListBox permite mostrar una lista de elementos para que el usuario los seleccione haciendo clic en ellos.

LOAD: Evento que se produce antes de que se muestre un formulario Windows Forms por primera vez.

M

MD5: Esquema de hash normalizado unívoco de 128 bits desarrollado por RSA Data Security, Inc. y utilizado por varios proveedores de Protocolo Punto a Punto (PPP) para autenticación cifrada, incluyendo servidores de Internet.

MDAC: Acrónimo de Microsoft Data Access Components o Componentes de Acceso a Datos de Microsoft, colección de componentes que tienen como función proporcionar conectividad a las bases de datos en plataformas Windows.

MDI: Acrónimo de Multiple Document Interfaz o Interfaz de Múltiples Documentos, interfaz de ventanas de Windows que permite mostrar varios documentos al mismo tiempo, cada uno de ellos en su propia ventana. Las aplicaciones MDI suelen tener un elemento de menú Ventana con submenús que permiten cambiar entre ventanas o documentos.

Mega Byte (Mbyte): Un millón de bytes o 1,048,576 bytes (2^{20} bytes)

MessageBox: Muestra un cuadro de mensaje que puede contener texto, botones y símbolos que informan e instruyen al usuario.

MSDN: Acrónimo de Microsoft Developer Network o Red de Desarrollo Microsoft, recurso para programadores que incluyen descargas, instrucciones, ejemplos y guías prácticas para desarrollar aplicaciones para los sistemas operativos Windows.

MSIL: Acrónimo de Microsoft Intermediate Language o Lenguaje Intermedio de Microsoft, lenguaje utilizado como salida de una serie de compiladores y como entrada para un compilador Just-In-Time (JIT). Common Language Runtime incluye un compilador Just-In-Time para convertir MSIL a código nativo.

Mucosa: Membrana que tapiza las cavidades interiores del cuerpo y segrega una especie de moco.

O

Objeto: En programación, entidad consistente de datos y procedimientos públicos o privados que pueden ser accedidos por una instancia.

ODBC: Acrónimo de Open DataBase Connectivity o Conectividad Abierta a Base de Datos, Interfaz de Programación de Aplicaciones (API) de bases de datos acorde con las normas de la interfaz de nivel de llamadas (CLI) de bases de datos del American National Standards Institute (ANSI) y la Organización Internacional de Estandarización (ISO). ODBC acepta el acceso a cualquier base de datos para la que haya disponible un controlador ODBC.

OLE: Acrónimo de Object Linking and Embedding o Vinculación de Objeto e Incrustado, es el modo de transferir y compartir información entre diferentes aplicaciones.

OLE DB: Interfaz de programación de aplicaciones (API) basada en COM para tener acceso a datos. OLE DB acepta el acceso a cualquier formato de almacenamiento de datos (bases de datos, hojas de cálculo, archivos de texto, etc.) para los que haya un proveedor de OLE DB disponible.

Open Source: Denominación alternativa del software libre enfocada más en los aspectos pragmáticos adscrito a la licencia GPL (General Public License o Licencia Pública General) que permite ver, modificar y distribuir el código fuente.

P

PEAR: Acrónimo de PHP Extension and Application Repository.

Pentium: Microprocesador lanzado al mercado por Intel en 1993, sucesor del 486. Según la sucesión lógica, debería haberse llamado 586 o 80586, pero Intel lo denominó Pentium por razones de copyright.

PHP: Acrónimo de Hypertext Preprocessor, lenguaje de programación orientado a objetos basado en el modelo de preprocesamiento de páginas HTML.

Postema: Tejido orgánico que acumula un líquido espeso, blanco amarillento, que secretan accidentalmente los tejidos inflamados o llagas, comúnmente llamado pus.

Proxy: Sistema que reenvía los paquetes TCP/IP recibidos a otra red, sin permitir el tráfico directo entre ellas.

R

RAM: Acrónimo de Random Access Memory o Memoria de Acceso Aleatorio, en informática, memoria basada en semiconductores que puede ser leída y escrita por el microprocesador u otros dispositivos de hardware.

RecordSet: Cursor especial que ADO proporciona para recuperar y presentar un conjunto de filas o registros de una base de datos para recorrerlos y actualizarlos.

S

SELECT: Instrucción de SQL utilizada para pedir una selección, proyección, combinación, consulta, etc., de una base de datos.

Sistema Operativo: Software básico que controla una computadora. El sistema operativo tiene tres grandes funciones: coordina y manipula el hardware del ordenador o computadora, organiza los archivos en diversos dispositivos de almacenamiento y gestiona los errores de hardware y la pérdida de datos.

SMTP: Acrónimo de Simple Mail Transfer Protocol o Protocolo Simple de Transferencia de Correo, miembro del conjunto de protocolos TCP/IP que controla el intercambio de correo electrónico entre agentes de transferencia de mensajes.

Software: Programas de computadoras. Las dos categorías primarias de software son los sistemas operativos (software del sistema), que controlan los dispositivos físicos del ordenador o computadora, y el software de aplicación, que dirige las distintas tareas para las que se utilizan las computadoras.

SQL: Acrónimo de Structured Query Language o Lenguaje de Consultas Estructurado, lenguaje de consultas de bases de datos que desarrolló originalmente IBM; se utiliza ampliamente para tener acceso a datos, consultar, actualizar y administrar sistemas de bases de datos relacionales.

Switch: Dispositivo de red capaz de realizar una serie de tareas de administración, incluyendo el redireccionamiento de los datos, que sirve para conectar varios equipos de computo a una red local.

T

TabControl: Control de Windows Forms que muestra múltiples fichas, similares a los divisores de un cuaderno o a las etiquetas de las carpetas de un archivador. Las fichas pueden contener imágenes y otros controles.

TCP/IP: Acrónimo de Protocolo de Control de Transporte/Protocolo Internet, conjunto de protocolos de red muy utilizados en Internet que permiten la comunicación entre redes interconectadas formadas por equipos con distintas arquitecturas de hardware y sistemas operativos. TCP/IP incluye estándares para la comunicación entre equipos y convenciones para conectar redes y enrutar las transmisiones.

TDS: Acrónimo de Tabular Data Stream o Flujo Tabular de Datos, protocolo nativo de comunicación para SQL Server.

Text Box: Control de los formularios Windows Forms que se utiliza para obtener entradas del usuario o para mostrar texto.

Toxina Botulínicia: Sustancia que paraliza las fibras musculares en las que se inyecta por denervación química de la placa motora, provocando la parálisis temporal de las mismas durante unos 6 a 8 meses.

Tzapotlán: Palabra náhuatl que significa: lugar donde abundan los zapotes. Se conforma de los vocablos zapotl, que quiere decir zapotes, y tlán, que significa lugar de abundancia.

V

Visual Basic: Lenguaje visual de programación desarrollado por Microsoft basado en el lenguaje BASIC (Beginners All-Purpose Symbolic Instruction Code).

Visual Basic .Net: Lenguaje de programación de Microsoft orientado a objetos y basado en el lenguaje Visual Basic y en la tecnología .Net Framework.

Visual Basic Script: Lenguaje de programación que lleva la ejecución de secuencias de comandos a una variedad de entornos, incluida la ejecución de secuencias de clientes Web en Microsoft Internet Explorer y la ejecución de secuencias de servidores Web en Servicios de Microsoft Internet Information Server.

Visual C++: Entorno de desarrollo dinámico para crear aplicaciones basadas en Microsoft Windows y Microsoft .NET, aplicaciones Web dinámicas y servicios Web XML mediante el lenguaje de programación C++.

Visual J#: Herramienta de desarrollo con sintaxis similar a la del lenguaje Java para generar aplicaciones y servicios en .Net Framework.

Visual Studio .Net: IDE compartido de Visual Studio, que utilizan Visual Basic, Visual C# y Visual C++ para crear aplicaciones.

W

Web Forms: Los formularios Web Forms son una tecnología ASP.NET que se utiliza para crear páginas Web programables. Los formularios Web Forms se representan como código HTML y secuencias de comandos compatibles con exploradores, lo que permite ver las páginas en cualquier explorador y plataforma.

Windows: Nombre común o coloquial del sistema operativo Microsoft Windows, un entorno multitarea dotado de una interfaz gráfica de usuario basada en ventanas.

Windows Forms: Los formularios Windows Forms son la nueva plataforma de desarrollo de aplicaciones para Microsoft Windows, basados en .NET Framework. Este marco de trabajo proporciona un conjunto de clases claro, orientado a objetos y ampliable, que permite desarrollar complejas aplicaciones para Windows.

Windows Installer: Componente del sistema operativo Windows que simplifica el proceso de instalación de aplicaciones.

World Wide Web: Sistema para explorar Internet mediante hipervínculos. Cuando se utiliza un explorador Web, el Web aparece como una colección de texto, imágenes, sonidos y películas digitales.

X

XML: Acrónimo de eXtensible Markup Language o Lenguaje de Marcado Extensible, Metalenguaje de marcado que proporciona un formato para describir datos estructurados. Permite declarar el contenido de modo más preciso y obtener resultados de búsqueda más significativos en múltiples plataformas.

XSD: Lenguaje de definición de esquemas XML, representa el esquema de un conjunto de datos, es decir, las tablas, columnas, tipos de datos y restricciones que se encuentran en el conjunto de datos, se define mediante un esquema XML.

Bibliografía

Libros

Applied Microsoft .NET Framework Programming
Jeffrey Richter
Mc Graw Hill – Microsoft Press
Primera Edición 2002

Ingeniería de Software, Un enfoque práctico
Roger S. Presuman
Mc Graw Hill
Tercera Edición

Professional PHP Programming
Jesus Castagnetto - Harish Rawat - Sascha Schumann - Chris Scollo - Deepak Veliath
Wrox Press
Primera Edición 1999

Programación Avanzada con Microsoft Visual Basic .Net
Francesco Balena
Mc Graw Hill – Microsoft Press
Primera Edición 2003

Programming Microsoft Windows with C#
Charles Petzold
Mc Graw Hill - Microsoft Press
Primera Edición 2002

Visual Basic .Net en 21 Lecciones Avanzadas
Duncan Mackenzie – Kent Sharkey
Prentice Hall
Primera Edición 2003

Programas y Libros Electrónicos

Introducción a SQL Server 7.0 (CHM)
Traducción al Español 1998
http://www.microsoft.com/sql

Microsoft SharePoint Portal Server 2001 (CHM)
Traducción al Español 2001
http://www.microsoft.com

MSDN Microsoft Developer Network Library de Visual Studio .Net 2003

PHP Manual Ver. 5.0.4
Traducción en español
http://www.php.net/docs.php

Referencias en Internet

http://alarcos.inf-cr.uclm.es/per/fgarcia/isoftware/
Universidad de Castilla - La Mancha
Feliz García
Ingeniero en Informática por la UCLM

http://es.gotdotnet.com/quickstart/aspplus/
Tutorial de ASP.NET
Microsoft

http://eupt2.unizar.es/asignaturas/itig/ingenieria_de_software_II/teoria.html
Escuela Universitaria Politécnica de Teruel
Ingeniería de Software II

http://pear.php.net/index.php
PEAR - PHP Extension and Application Repository

http://programacion.com/
Programación en castellano

http://www.elguille.info/default.aspx
La Web del Visual Basic, C#, .NET y más...
Guillermo 'guille'

http://www.ezwp.com/category/php/
Ez Web Programming

http://www.guiaweb.gob.cl/guia/archivos/
Guia para el desarrollo de sitios web - Gobierno de Chile

http://www.lania.mx/biblioteca/newsletters/1996-otono-invierno/articulo1.html
Laboratorio Nacional de Informática Avanzada

http://www.marcelopedra.com.ar/glosario_A.htm
Marcelo Pedra Network - Glosario

http://www.microsoft.com/spanish/msdn/comunidad/dce/default.asp
Sitio para Desarrolladores Cinco Estrellas de Microdoft

http://www.mysql-hispano.org/
MySQL Hispano

http://www.phpclasses.org/
PHP Classes Repository

http://www.programar.net/
Recursos y aprendizaje de .Net
Ing. Rubén Darío Sánchez - Ing. Millán Andrés Sánchez

http://www.rae.es/
Real Academia Española