UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

SISTEMA DE PLANEACIÓN Y SEGUIMIENTO DE ACUERDOS

TESIS QUE PARA OBTENER EL TÍTULO DE
Ingeniero en Computación

PRESENTAN
Ana María Juárez Ariza
Fernando Hurtado Acuña

DIRIGIDA POR
M. EN I. YUKIHIRO MINAMI KOYAMA

Ciudad Universitaria, junio de 2006

Mi mas amplio agradecimiento a mis padres que siempre me han apoyado en todas mis decisiones, sueños y metas; sin ellos no seria posible lo que he logrado y lograré en mi vida, gracias a su amor y apoyo incondicional.

También quiero reconocer el apoyo y cariño de mis hermanos, he compartido tantos momentos, y sé que estarán ahí en el momento que los necesite.

Gracias a ti Fernando, por tu generosidad, cariño, comprensión y constate estimulo para crecer cada día en el aspecto profesional y personal, te agradezco darme la oportunidad de compartir este logro.

En general a toda mi familia y amigos que siempre están conmigo, de alguna forma son parte de lo que he logrado.

Finalmente, quisiera expresar mi agradecimiento a quienes estuvieron vinculados de alguna manera en este proyecto: Mateo, Luís, Miriam, Miguel, Karin y Elizabeth; gracias por todo el apoyo que nos han dado en estos años de experiencia laboral.

Ana Ma. Juárez Ariza

A mis padres, que con mucho cariño, esfuerzo y entusiasmo me impulsaron a concluir mis estudios profesionales, siempre estaré agradecido con ellos por el apoyo y amor que me han brindado.

A mi hermana con quien he compartido muchos momentos felices en mi vida.

A mi abuelita, la cual quiero y aprecio como una segunda mamá.

A Ana, que con su cariño y gran paciencia me ha ayudado en los momentos difíciles, y con la cual he pasado una serie de experiencias y vivencias que la hacen ser una persona muy especial para mí, a ella también debo este logro.

A toda mi familia y amigos en general que siempre me han apoyado y brindado su ayuda incondicional cuando lo he necesitado. Y un especial reconocimiento a Manuel Saldaña quien fue una fuerza motivadora importante para concluir esta tesis.

A Yukihiro, parte fundamental en el desarrollo de este trabajo de tesis y el cual estimo y aprecio como profesor y amigo.

A Luís Alva y Mateo Saito, por el apoyo y conocimiento que me han otorgado en el plano profesional y personal.

A la Universidad Nacional Autónoma de México, Facultad de Ingeniería y mis profesores a los cuales debo la formación y educación recibida.

Fernando Hurtado Acuña

CONTENIDO

CAPÍTULO 1 INTRODUCCIÓN 5

1.1 Objetivo 5

1.2 Problemática de la situación actual 6

1.3 Resumen 7

CAPÍTULO 2 REUNIONES DE TRABAJO 10

2.1 Importancia de la gestión de reuniones 11

2.2 Tipos de reuniones 12
2.3 Reuniones de trabajo efectivas 13
2.3.1 Etapa de planeación 13
2.3.2 Etapa de convocatoria 19
2.3.3 Etapa de desarrollo 20
2.3.4 Etapa de seguimiento 25
2.3.5 Etapa de cierre 27

CAPÍTULO 3 ANÁLISIS 28
3.1 Metodología de desarrollo 28
3.2 Lenguajes de programación 31
3.2.1 Programación estructurada y sus desventajas 33
3.2.2 Programación orientada a objetos (POO) 34
3.2.3 Las clases y los objetos 35
3.2.4 Privado, protegido y público 38
3.2.5 Herencia 39
3.2.6 Polimorfismo 40
3.2.7 Sobre escritura, redefinición y reintroducción de métodos 42
3.2.8 Métodos abstractos 43
3.3 Bases de datos 45
3.3.1 Definiciones 45
3.3.2 Ventajas a utilizar una base de datos 51
3.4 Etapas de diseño de una base de datos 52
3.4.1 Diseño conceptual 52
3.4.2 Diseño lógico 53
3.4.3 Normalización 53
3.4.4 Diseño físico 56
3.4.5 Diseño de transacciones 57
3.4.6 Diseño de interfaces de usuario 59

CAPÍTULO 4 DISEÑO 60

4.1 Diseño de la base de datos 61

4.2 Lenguaje de programación 75

CAPÍTULO 5 DESARROLLO 78

5.1 Estructura de la aplicación 79

5.1.1 Menú principal 80

5.1.2 Barra de botones 84

5.1.3 Estructura de árbol 85

5.1.4 Panel de dirección y contenido de un directorio 88

5.2 Catálogos del sistema 89

5.2.1 Agregar elementos 90

5.2.2 Eliminar elementos 91

5.2.3 Modificar elementos 92

5.2.4 Filtrar información 92

5.2.5 Buscar información 94

5.2.6 Imprimir un reporte 94

5.2.7 Listado y descripción de los catálogos del sistema 96

5.3 Configuración y cuentas de correo electrónico 99

5.4 Planeación de una reunión de trabajo 100

5.5 Convocatoria de una reunión 104

5.6 Desarrollo de una reunión 108

5.7 Seguimiento a los acuerdos derivados de una reunión 114

5.8 Cierre del registro de una reunión de trabajo 120

5.9 Exportación de información de una reunión 121

5.10 Importación de una reunión 122

5.11 Preferencias 123

CAPÍTULO 6 CONCLUSIONES Y RECOMENDACIONES 125

6.1 Conclusiones 125

6.2 Recomendaciones 127

REFERENCIAS BIBLIOGRÁFICAS 128

CAPÍTULO 1 INTRODUCCIÓN

1.1 Objetivo

El objetivo que se pretende con esta tesis es desarrollar una herramienta de software que reduzca el tiempo de registro y control de eventos, administre los acuerdos derivados de una reunión formal de trabajo, y otorgue a las organizaciones una infraestructura en el cual depositar y representar el conocimiento que se genera de reuniones de trabajo con la finalidad de llevar una adecuada gestión de proyectos.

1.2 Problemática de la situación actual

Las reuniones dominan el camino en el cual se hacen los negocios hoy en día. Aproximadamente 11 millones de reuniones se realizan diariamente en Estados Unidos. Aunque la mayoría nos quejamos acerca de ellas podemos esperar encontrarnos inmersos en toda nuestra carrera profesional. Muchos profesionistas atienden un total de 61.8 reuniones por mes y estudios recientes indican que el 50% del tiempo total invertido es mal utilizado.

La necesidad de mejorar las reuniones de trabajo es evidente. El reto es comunicar, aprender y utilizar las técnicas y tecnologías que mejoren nuestras reuniones. Como en otras áreas de negocio, la tecnología esta ayudando a realizar reuniones más fáciles y eficientes. En muchas organizaciones los correos electrónicos están siendo utilizados como un método fácil y rápido de comunicar información internamente sin la necesidad de reunirse. En realidad, el ochenta y dos por ciento de los ejecutivos comparten notas con colegas, 77% de ellos por correo electrónico. Sin embargo aún con la ayuda del correo electrónico para comunicarse, el 45% de los ejecutivos aún se sienten saturados por el número de reuniones que tienen que atender. Esto indica que cierta tecnología está contribuyendo al dilema de las reuniones, pero no resolviéndola completamente [1].

Es común observar durante el desarrollo de un proyecto, la omisión de actividades debido a la falta de registro de los acuerdos derivados de las reuniones de trabajo que el proyecto involucra, y que muchas de las actividades que se programan para que sean realizadas en un periodo de tiempo determinado se retrasan sin que nos enteremos al momento de su motivo, dejándonos incapacitados de realizar alguna acción o tomar una decisión para poderle dar continuidad. Entre más grande es el proyecto más son las actividades y personas que se ven involucradas en él, y resulta una tarea difícil para que el o los lideres del proyecto puedan seguir a detalle cada uno de los puntos que incluye el proyecto, y aún más difícil establecer el estado actual, de acuerdo a las actividades que se han realizado hasta el momento por cada uno de los participantes.

1.3 Resumen

La tesis que se presenta a continuación muestra el desarrollo de un sistema de software, el cual proporciona una guía para un proceso de reunión formal tomando aquellos aspectos considerados como eficaces para llevar a cabo reuniones de trabajo exitosas. El proceso incluye cinco etapas: planeación, convocatoria, desarrollo, seguimiento y cierre de una reunión de trabajo, por lo que el software se diseñó con la finalidad de manejar cada una de las etapas para una misma reunión y que tuviera una estructura de almacenamiento que nos permitiera ubicar con facilidad a una reunión de acuerdo al tipo y a sus principales atributos como el tema, el objetivo, la fecha, los participantes, los puntos que comprende, el estatus actual, acuerdos y acciones realizadas, entre otros.

Durante la reunión será posible acceder al espacio de trabajo del software diseñado especialmente para cada etapa, haciendo posible consultar agendas, puntos pendientes, objetivos, informes y archivos que se adjunten en cada una de las reuniones. Existe un área reservada dentro del ambiente de trabajo para almacenar todas las decisiones que han sido hechas, las tareas asignadas y sus fechas límites para cada tarea o acuerdo, así como las actividades realizadas por cada uno de los responsables.

Como se mencionó, la idea de la realización del sistema es la de llevar un proceso de reunión de manera electrónica con la finalidad de reducir el tiempo y hacer las reuniones productivas, considerando otros beneficios como:

- Todos los materiales y elementos de la reunión serán almacenados en una misma ubicación
- Las decisiones y las tareas son guardadas, asegurando el seguimiento
- Los registros anteriores de las reuniones son fácilmente accesibles
- Todos los elementos de software que se generan para y durante la reunión son guardados y asegurados.

La idea principal del diseño de la herramienta se basa en poder brindar al usuario un entorno de trabajo completo y a su vez fácil de usar, pensado principalmente en hacer del

software una herramienta útil que pueda ser utilizada por cualquier persona que desea dirigir una reunión de trabajo, llevar el control del seguimiento de un proyecto y coordinar a los responsables de las actividades inmersas en él, es decir que sirva como una herramienta de apoyo al liderazgo.

Actualmente existen en el mercado diferentes aplicaciones que llevan acabo un proceso electrónico de reunión, paquetes de software que apoyan la preparación y dirigen el curso de una reunión a través de sus diferentes etapas. Muchas de estas aplicaciones únicamente se centran en tareas específicas y no consideran el ciclo completo por el cual pasa una reunión de trabajo, por lo que es necesario utilizar varias aplicaciones para poder llevar un control pleno y a detalle de los aspectos que conlleva una reunión, y comunicar los cambios a los participantes. A continuación en la figura 1.1 se muestra una tabla comparativa con algunos de los principales paquetes de software y nuestro sistema de gestión y control de reuniones.

	Agenda	Ciclo de una reunión	Correo electrónico	Aplicación Web	Características principales
Meeting Planner Plus	●			●	• Multiusuario con acceso a un evento al mismo tiempo • Reportes profesionales para administración y clientes • Se muestra
Acuérdate	●	●	●	●	• Actualización de cambios vía correo electrónico. • Sincronización con Microsoft Outlook, Microsoft Exchange, Microsoft Mail, Lotus Notes. • Conjunto de reportes especializados para cada una de las etapas de la reunión. • Revisión de avances por proyecto. • Integración de utilerias dentro de la misma herramienta.
meeting maker incorporated	●			●	• Programación de reuniones • Multi usuario • Multidispositivo
Microsoft Outlook	●		●		• Planeación y envío de la información de una reunión • Calendarización de eventos • Agenda
MSExchange.org http://www.MSExchange.org	●		●		• Planeación de una reunión • Agenda • Manejo de correo electrónico

Figura 1.1 Tabla comparativa de “Acuérdate” con otros sistemas de su tipo

CAPÍTULO 2 REUNIONES DE TRABAJO

Las reuniones de trabajo o estudio son esenciales para tomar decisiones, retroalimentarse o planificar acciones en común. Pero pueden ser una pérdida de tiempo si no se aplican algunas reglas probadas para que sean eficientes, como lo son: establecer el propósito y los objetivos, planear con anticipación, realizar un orden del día, controlar su duración, entre otras. Todos hemos asistido a reuniones, algunas productivas y otras tremendamente inútiles. Cuando un equipo de trabajo suele tener reuniones con bajo nivel de eficacia, no sólo se pierde el valioso tiempo de los que participan de la misma, sino que además “todos” están aprendiendo el hábito de la inefectividad colectiva (EffectiveMettings.com [9]).

Las reuniones se llevan a cabo con diferentes propósitos dentro de las organizaciones como lo son: informar, convencer, obtener información, tomar decisiones, sensibilizar o motivar, resolver situaciones de trabajo.

La realización de reuniones efectivas permite recopilar conocimientos, sugerencias y experiencias de varias personas, para resolver un problema. Abre espacios de participación y cohesión para el logro de las metas. Establece un acercamiento entre los miembros y el buen manejo de las diferencias. Respecto a la participación, propicia tolerancia a las opiniones individuales y las decisiones en grupo.

2.1 Importancia de la gestión de reuniones

La reunión de trabajo es una herramienta muy importante para cualquier persona que desee hacer más eficientes sus esfuerzos para lograr un objetivo. El riesgo del fracaso siempre está latente, y sólo una acción efectiva de quienes se declaren responsables de la misma puede lograr que los resultados sean satisfactorios.

A nadie le gusta ser convocado a una reunión que requiere un aporte personal de tiempo y esfuerzo, sin saber el por qué o el para qué. La falta de claridad en los objetivos, la impertinencia, el desconocimiento de la agenda y un mensaje incompleto o ambiguo sobre la necesidad de una reunión, muy probablemente disminuyan la motivación en los participantes. Algunos ni siquiera se tomarán la molestia de asistir. Otros irán y disconformes, no participarán ni aprovecharán los contenidos del encuentro.

Es frecuente que, dada la funcionalidad de una reunión, se convoque excesivamente a ellas. A veces no hace falta reunirse, ya que los objetivos propuestos pueden conseguirse mediante el intercambio de información, ya sea escrita o telefónica. Solamente debe convocarse a una reunión si no hay otra forma más rápida y económica de comunicarse. Esta decisión se podrá tomar únicamente si se tienen claros los objetivos. Esto es fundamental y si se decide convocar a la reunión, todos deben tener presentes los propósitos de ella. En la reunión deben tomarse acuerdos, definir a los responsables y los plazos. Debe salir de ella un plan claro y concreto de acción. Debe planificarse un seguimiento de los acuerdos hasta que éstos se cumplan (MettingMaker.com [10]).

2.2 Tipos de reuniones

El organizador de la reunión deberá trabajar aspectos y estrategias para conseguir de forma rápida y eficaz los objetivos marcados, dependiendo del motivo por el que se convoque.

El término reunión es demasiado general, pero lo cierto es que dependiendo de su tipo se requieren actitudes específicas. Entre las reuniones más habituales se encuentran las siguientes (Ezequiel [1]):

- **Reuniones departamentales periódicas**

En este tipo de reuniones los responsables de cada área o departamento tienen el objetivo de dar a conocer las actividades que realizaron sus miembros, en un determinado periodo de tiempo. La característica principal de estas reuniones es que constituyen el único momento en el que los departamentos dejan de ser una abstracción y se establecen como un equipo.

- **Reuniones interdepartamentales**

Estas reuniones son realizadas por los integrantes de varios departamentos (contabilidad, informática, calidad, recursos humanos, logística, etc.). El problema más frecuente de estas reuniones son las batallas "tribales", ya que cada departamento siente el deber de defender su territorio. El convocante debe conseguir que sean lo más productivas y maduras posibles.

- **Reuniones de proyecto**

Como su propio nombre indica, son encuentros centrados en un tema concreto. Lo primordial es mantener vivo el interés por el proyecto y conseguir que todos los participantes cumplan los plazos. Dentro de la disciplina, es importante centrarse en el tema y el futuro del proyecto evitando las divagaciones.

- **Reuniones informativas**

Su propósito es transmitir una información que se considera necesaria o conveniente que se reciba, complementando con la sección de preguntas, aclaraciones y dudas (por parte de quienes reciben la información), y las respuestas pertinentes, aquéllas que tienen por finalidad transmitir información y las que se realizan para recibir información, ya sea para

conocer la opinión del grupo o para consultarlo acerca de una decisión que se piensa tomar.

Algunos las llaman reuniones de estrategia. También podrían denominarse reuniones de negociación. Se trata de reuniones en las que las partes presentes están en desacuerdo; el objetivo de este tipo de reuniones es alcanzar un acuerdo o consenso para superar una situación conflictiva.

2.3 Reuniones de trabajo efectivas

La realización de reuniones efectivas permite recopilar conocimientos, sugerencias y experiencias de varias personas, para resolver un problema. Abre espacios de participación y cohesión para el logro de metas. Establece un acercamiento entre los miembros y el buen manejo de las diferencias, así como respeto y tolerancia por las opiniones individuales y decisiones grupales (Ezequiel [1]).

La efectividad de las reuniones de trabajo depende de varios factores, por lo que se propone una metodología que considera las siguientes etapas entre las que podemos distinguir: *planeación, convocatoria, desarrollo, seguimiento y cierre.*

2.3.1 Etapa de planeación

El éxito de las reuniones depende fundamentalmente de su preparación cuidadosa por lo que deben ser planeadas con anticipación. Para que una reunión sea exitosa, es necesario planificarla y prepararla bien; establecer los objetivos y el temario general. Esto evitará que todos tengan que oír algo que no les interesa, como algún sermón fuera de lugar de algún jefe, o una discusión inútil con respecto a cualquier tema poco interesante.

Se requiere establecer claramente el objetivo; considere las veces que fue invitado a reuniones. ¿A cuáles de ellas asistió con las mismas ganas? A las que se sintió obligado a asistir o a aquéllas a las cuales sintió que podría realizar un aporte significativo. No podemos controlar el estado de ánimo de los demás; lo que sí podemos procurar es facilitar el proceso por el cual las personas sienten que están participando en algo de

provecho. Quien convoca a una reunión debe asegurarse de invitar a las personas que pueden contribuir al objetivo de la misma.

Por último, en esta etapa preparatoria está el establecimiento de roles; se requiere definir quién será el líder, el moderador, el secretario y los participantes, de acuerdo a lo siguiente:

- **Convocante**: inicia y cierra la reunión, obtiene conclusiones y establece los roles, además se debe preocupar porque se cumplan las pautas fijadas.
- **Moderador**: es responsable de coordinar la actividad y llevar el control del tiempo.
- **Secretario:** registra lo tratado durante la reunión, capta las ideas y toma nota de las mismas, sus conclusiones y demás.
- **Participantes:** son responsables de asistir y participar activamente con entusiasmo, creatividad y compromiso (aunque parezca que son los que menos hacen, son los más importantes de la reunión).

En la figura 2.1 se muestra el diagrama de flujo que sigue la etapa de planeación.

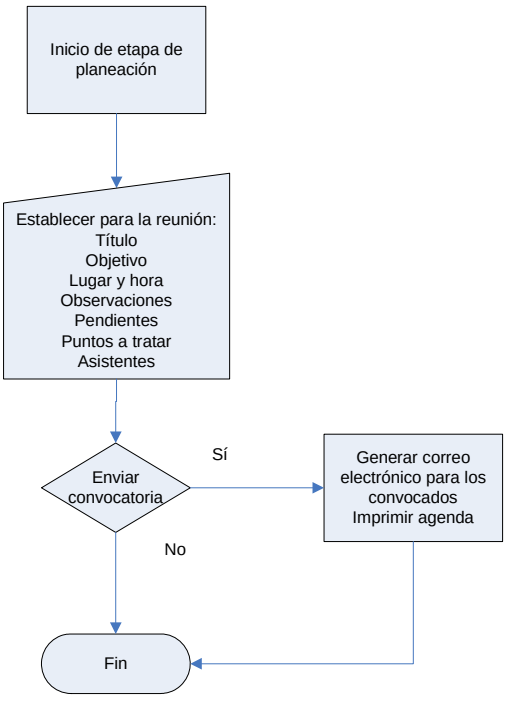


Figura 2.1 Diagrama de flujo de la etapa de planeación.

A continuación se describen las actividades inmersas en la planeación de una reunión.

- **Establecer el propósito y los objetivos**

El primer paso a la hora de preparar una reunión es determinar o establecer cuál es el propósito que tiene la reunión y cuáles son los objetivos que deben ser conseguidos por medio de la misma. Estos dos aspectos que hemos mencionado anteriormente son muy importantes. Las actividades no son fines en sí mismas. Las reuniones son, o han de ser, medios para conseguir fines últimos. De acuerdo a esto veamos la forma en cómo se deben fijar los propósitos y objetivos.

a) Propósito

Para establecer el propósito se deben hacer las preguntas clave *¿Qué quieres conseguir por medio de esta reunión? ¿Qué impacto deseas producir?* Toda reunión, para que pueda tener éxito, ha de tener un propósito claro. El propósito es una especie de declaración de intenciones. Es una idea general, no es preciso que sea muy específica, de aquello que deseas conseguir por medio de la reunión que estás preparando.

b) Objetivos

Los objetivos son los logros concretos y específicos que deseas conseguir por medio de la reunión. A continuación presento un cuadro comparativo entre propósito y objetivos

Propósito	Objetivos
General	Específico
Intenciones	Logros
Amplio	Restringido
Difícil de medir	Mensurable
Indefinido	Definido

Si un objetivo no reúne las características arriba mencionadas, no puede ser considerado como tal. La consecución de los diferentes objetivos traerá como consecuencia lógica el cumplimiento del propósito de la actividad. Dicho de otra manera, el propósito se consigue mediante el cumplimiento de los diversos objetivos que se establecen.

El objetivo es la razón por la que se convoca una reunión y es fundamental no perderlo de vista. Si se sabe a dónde se quiere llegar, se podrá determinar qué partes del orden del día pueden ayudar a conseguirlo y qué partes son innecesarias.

Por tanto, es fundamental elaborar el orden del día con antelación y hacerlo llegar a los participantes antes de la reunión. De esta forma, todos habrán tenido tiempo de leer la documentación y de preparar los temas, y así la reunión será más rápida y eficaz.

Si las actividades no contribuyen de manera clara al logro de objetivos, pueden convertirse en un factor que desmotive; esto sucede con mucha frecuencia cuando las actividades, por carecer de propósito y objetivos, se han convertido en fines en vez de ser medios. Cuando esto sucede, las actividades se enquistan y pierden todo su valor.

- **Selección previa de los participantes**

En relación con este aspecto, la práctica indica que debe citarse a la reunión sólo a aquellas personas que pueden hacer una contribución efectiva a ella, ya sea por sus conocimientos, experiencia o capacidad de análisis. Aunque este aspecto es delicado, pues puede crear sentimientos de frustración entre aquellas personas que no fueron convocadas a la reunión, el conductor de la reunión debe enfatizar a los asistentes el propósito de ésta y lo pertinente de la participación de cada uno de ellos en función de los objetivos que se persiguen.

- **Determinar el número de asistentes**

Naturalmente, el número de personas que se inviten a la reunión depende del propósito de la misma. Si es para informar o solicitar información, plantear problemas o solicitar ideas, se puede invitar sin restricciones a todas las personas que se estimen necesarias.

Por el contrario, si la finalidad de la reunión es convencer a los asistentes respecto de una decisión ya adoptada, tomar decisiones o analizar proposiciones, lo más conveniente es invitar un número restringido de personas. Los grupos formados por un número impar de asistentes, son más efectivos que los formados por un número par para la toma de decisiones, ya que se evitan empates y confrontaciones irresolubles, al mismo tiempo que se aprovecha mejor el tiempo disponible.

- **Orden del día**

Crear un orden del día efectivo es uno de los elementos más importantes para obtener una reunión productiva. A continuación se mencionan algunas razones por las cuales su valor es de gran importancia:

- Comunica información importante como:
  - Los temas a discutir
  - La presentación al líder para cada tema
  - Tiempo permitido para cada tema
- Provee un perfil para la reunión (cuánto tiempo se dedica a cada tema)
- Puede utilizarse como una lista de verificación, para asegurarnos que toda la información está cubierta
- Permite a los asistentes conocer cuáles serán los temas a discutir, si es distribuida antes de la reunión. Esto les da oportunidad para llegar a la reunión preparados para las discusiones
- Provee un enfoque claro para la reunión (el objetivo de la reunión deberá ser claramente establecido en el orden del día).

Un verdadero orden del día agiliza una reunión y consigue que sea productiva. Lo primero es decidir qué temas se van a tratar. Una buena fuente de información es el acta de la reunión pasada (suponiendo que no es la primera sobre un tema concreto).

También debe tratar los temas más urgentes, y estar ordenado de forma que los que estén relacionados o dependan de la conclusión de otro, estén juntos. De esta manera, se consigue que los asistentes no tengan que cambiar demasiado el enfoque al pasar de uno a otro.

El factor tiempo también es necesario tenerlo en cuenta: las personas suelen estar más atentas y mostrarse más creativas al inicio de la reunión. Cuando se produce una reducción del nivel de atención, es el momento de plantear temas que provoquen reacciones fuertes, con el fin de que los asistentes se interesen por el tema, a pesar del cansancio.

- **Distribuir una agenda previa**

Todas las personas consideradas para formar parte de la reunión deben recibir, con la máxima anticipación posible, la agenda de trabajo. Una buena agenda de trabajo debe incluir los siguientes puntos: el lugar donde se realizará la reunión, la fecha de realización, la hora de inicio y término, los temas a tratar en orden de prioridad e indicándose el nombre de las personas que los tratarán y el tiempo asignado a cada uno de ellos. Este último aspecto es de vital importancia para la dinámica de la reunión. En primer lugar,

señala a los participantes la importancia relativa de los diferentes asuntos. En segundo lugar, contribuye a evitar que en el tiempo dedicado a un asunto de la reunión esté en relación inversa a su importancia.

- **Establecer fecha, hora y lugar donde se celebrará la reunión**

La fecha, hora y lugar donde se celebrará la reunión son atributos propios de la reunión, que se deben conservar como información de la misma.

- **Acuerdos pendientes de reuniones anteriores**

Cuando se llevan a cabo reuniones de trabajo, es usual que asista de forma periódica el mismo grupo de asistentes, por lo que es importante tomar en cuenta todos aquellos acuerdos pendientes de reuniones anteriores, con la finalidad de no perder el seguimiento de ellos; así, antes de establecer el orden del día es necesario determinar qué pendientes formarán parte como puntos a tratar en la siguiente reunión.

- **Preparación y organización del material o documentación destinado a la reunión**

No basta con elaborar una agenda de trabajo y hacer una convocatoria adecuada en tiempo y forma, también hay que preparar el material y documentación que se ha de utilizar en la reunión.

Aunque la preparación de los documentos no siempre es absolutamente necesaria, en muchas reuniones, el disponer de documentación para presentar y entregar a los participantes, facilita y mejora notablemente su calidad. Hay dos tipos principales de documentos:

- Los documentos de trabajo destinados a los participantes (que conviene entregar antes de la reunión)
- Los documentos que el coordinador, o alguno de los participantes, ha de presentar durante la reunión (estadísticas, informes, gráficos o diapositivas).

2.3.2 Etapa de convocatoria

La convocatoria es el aviso previo, que hace saber el orden del día y agenda; se envían a los participantes algunos documentos para que puedan estudiarlos, y vayan a la reunión con pleno conocimiento de lo que se va a tratar y hayan podido formarse un juicio u opinión personal sobre los temas para considerar. Por último, es de gran importancia que la convocatoria se haga con suficiente anticipación para que los asistentes prevean el tiempo de duración. En general, cabe aconsejar que las reuniones no superen las dos horas y cuarto, de lo contrario debe preverse un descanso de diez o quince minutos. Esto puede parecer perder el tiempo, pero si no se hace, es bastante probable que se pierda más tiempo con divagaciones inoportunas o repeticiones, a causa del cansancio de la gente.

La figura 2.2 muestra el diagrama de flujo de la etapa de convocatoria que consiste básicamente en hacer una invitación formal, o enterar a todos los participantes de la reunión, ya sea por vía telefónica, fax, correo electrónico u otros medios, con el fin de:

- Confirmar fecha y hora de la reunión
- Especificar el nombre, orientación temática y lugar de localización
- Precisar las políticas y normas
- Enviar el orden del día a los posibles asistentes
- Precisar los asuntos que serán tratados durante el desarrollo de las reuniones
- Manejar las relaciones de posibles asistentes e invitados especiales, asuntos tratados, extensión de los mismos o cualquier otra circunstancia especial.

En el caso de que por alguna circunstancia no se pueda hacer reunir a la mayoría de los participantes, se puede disponer a cambiar la fecha de la reunión. Es importante enviar un aviso a todos los participantes en caso de que se haya convocado inicialmente a una fecha y hora, y posteriormente se actualice a una diferente.

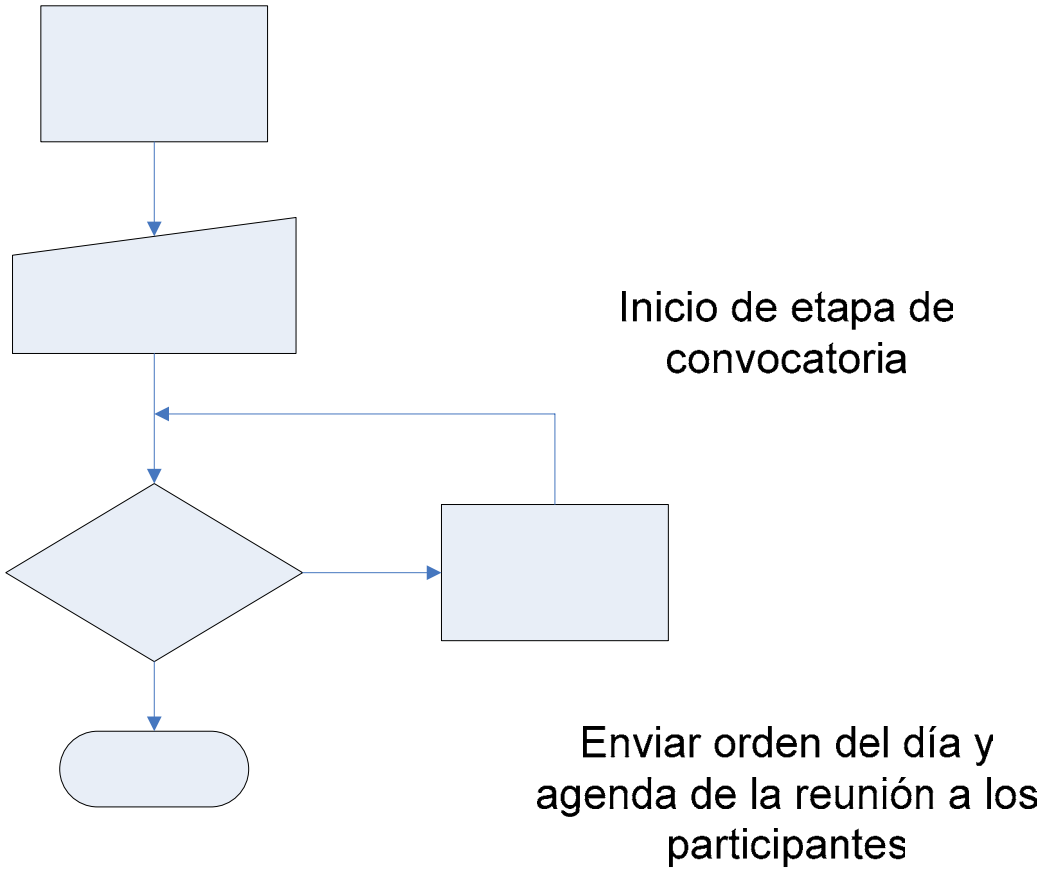


Figura 2.2 Diagrama de flujo de la etapa de convocatoria.

2.3.3 Etapa de desarrollo

Esta etapa comienza al iniciar la reunión; es recomendable dar una breve bienvenida, cordial, tendiente a eliminar tensiones y despertar el interés.

Antes del comienzo formal es importante considerar los siguientes puntos:

- Iniciar puntualmente la reunión. Desde el punto de vista motivacional no comenzar una reunión a la hora señalada es una forma de premiar la tardanza de los ausentes y de castigar la puntualidad de los presentes y, por consiguiente, estimular en éstos últimos una disposición negativa hacia la reunión, lo que empobrecerá sus participaciones personales y los resultados de la misma. Por otra parte, suponiendo que el personal de la organización está realmente ocupado en los asuntos que les son propios, asistir a la reunión será una actividad más de su recargada agenda diaria de trabajo. Si ésta

se extiende más allá del tiempo considerado, lo más probable es que se alteren los programas de actividades de todos los participantes, con las consiguientes e innecesarias tensiones y la inevitable distracción de la atención.

- La actitud y la disposición del conductor de la reunión determinan, en gran medida, la participación de los asistentes y, desde luego, el rendimiento de éstos. Por lo tanto, el conductor debe dar a conocer, de manera tranquila, pero firme, los resultados que se esperan de la reunión. Si bien el método para tomar decisiones o derivar conclusiones puede ser direccionable, la reunión debe ser conducida sin bromas por parte del conductor. Todos los puntos en la agenda deben ser presentados y desarrollados en los tiempos previstos para ellos.

- Fijar la duración de la reunión (el tiempo aconsejable es de 50 a 60 minutos como máximo); el respeto a esta duración contribuye al prestigio del director y crea confianza en los participantes para futuras reuniones, ya que pueden programar su tiempo sin prolongaciones.

- Mencionar los temas a tratar (para evitar que algunas personas anticipen) con señalamiento de su naturaleza, antecedentes, importancia y puntos sobresalientes; debe realizarse muy brevemente porque sólo es una introducción.

- Ser claro en las consignas de la misma. Reflexionemos un minuto sobre la forma en la que explican los invitados su presencia en la reunión: voy porque tenía el compromiso asumido con tal persona; voy porque es una exigencia laboral; voy porque el tema era interesante; voy sólo porque siempre voy; o, voy para lograr tales objetivos.

- Ritmo. Debemos asegurarnos que la reunión tenga un ritmo ágil, dicho de otro modo, que la sucesión de las diferentes partes de la misma se lleve a cabo de una manera rápida impidiendo que la reunión sea tediosa y aburrida. Lo bueno, si breve, dos veces bueno, afirma el viejo refrán castellano. Debe existir un buen equilibrio entre las diferentes partes que componen la reunión, a fin de evitar que unas monopolicen el tiempo, no dejando suficiente espacio para las otras.

Algunas otras recomendaciones que se sugieren para llevar acabo reuniones de trabajo efectivas son las siguientes:

- Acondicionamiento de la sala (retroproyector, material, pizarrón, entre otros)
- Evitar sitios oscuros o excesivamente iluminados
- La información recabada a través de la reunión, deberá ser almacenada y formar parte de los archivos de reuniones de la compañía
- Las decisiones hechas por el grupo deberán ser documentadas
- Para obtener una mejor efectividad en las reuniones de trabajo, es recomendable hacer una revisión al final de cada una y sugerir mejoras que deberán ser aplicadas en la siguiente reunión.

En la figura 2.3 muestra el diagrama de flujo de la etapa de desarrollo, así como la recopilación de información que se maneja en el desarrollo de la reunión. Para esto es importante considerar los siguientes puntos:

1) Verificar la asistencia, con el propósito de tener un registro de las personas que asistieron, para cualquier aclaración en cuanto a lo acordado en la reunión.

2) Dar lectura al orden del día.

3) Para cada uno de los puntos del orden del día se capturan los acuerdos, asignando responsables, fecha de entrega e instrucciones a seguir.

4) En caso de que se dé seguimiento, es necesario establecer la fecha, lugar y objetivo de la siguiente reunión.

5) Para finalizar esta etapa, sólo se debe terminar la reunión especificando la hora de término; esto significa que ha concluido el orden del día y que no queda algo pendiente por discutir.

6) Se genera la minuta y se entrega a los asistentes.

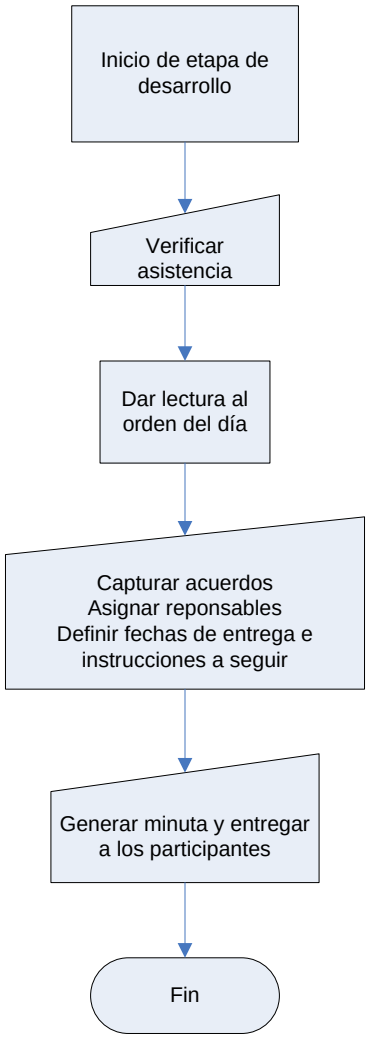


Figura 2.3 Esquema de la etapa de desarrollo

A continuación se presentan algunas directrices que permiten optimizar el desarrollo de las reuniones:

- Sólo una persona debe hablar a la vez. Cualquiera que desee hablar, debe indicarlo y ser reconocido antes de hacerlo
- No se deben aceptar conversaciones privadas durante la reunión
- Las personas deben hacer uso de la palabra sólo para referirse al tema que se discute
- Los comentarios y discusiones deben versar sobre asuntos e ideas, no sobre personas

Es importante mencionar que el orden del día establecido en la etapa de planeación no es el definitivo, porque en el momento en que se lleva a cabo la reunión, es muy probable que se agreguen, modifiquen o eliminen puntos; lo mismo sucede con los acuerdos, responsables e instrucciones.

- **Termino de la reunión**

Toda reunión hay que terminarla, no dejarla morir como ocurre con frecuencia. El ideal es que las reuniones terminen a la hora adecuada. Esto es oportuno, por varias razones: permite que cada uno organice su tiempo; otra razón importante es poder disponer de algunos minutos, después de terminada la reunión, para cambiar impresiones o para hablar de lo que surja espontáneamente.

¿Cómo terminar las reuniones para que sean realmente eficaces? Una reunión que se realiza de manera productiva y en un buen clima, puede quedar sólo en eso, si al final de la misma no se precisan algunas cuestiones bien concretas:

- ¿Cuáles son los acuerdos y decisiones que se han tomado?
- ¿Cuáles son los pasos, actividades y tareas que hay que realizar para cumplir con los acuerdos y llevar a cabo las decisiones tomadas?

Una vez hecho esto, hay que proceder a:

- Designar a las personas responsables de llevar a cabo lo acordado.
- Asignar los recursos que sean necesarios para concretar las actividades y tareas propuestas.
- Establecer un cronograma de actividades que se derivan de las resoluciones tomadas en la reunión.

No necesariamente hay que elaborar un acta. Basta un documento en el que se indique con claridad lo que se ha acordado, cómo hacerlo, cuándo y quiénes serán los responsables de llevar acabo, con indicación de medios y recursos que contarán para ello, si esto fuese necesario por la naturaleza de lo acordado.

¿Cómo documentar los resultados?

Cuando se considera oportuno, todo esto hay que considerarlo por escrito en un acta final. Este documento o acta de lo acordado, debe ser breve, escrito con un lenguaje claro, concreto y preciso. Como es obvio, este documento final es imposible de redactar, si desde el comienzo de la reunión no se ha designado un secretario de actas o, simplemente un responsable de tomar notas acerca de lo que se va tratando en la reunión.

Todos los asistentes deben estar de acuerdo con el término de la reunión, porque toda la información capturada no podrá eliminarse ni modificarse; es así como da inicio la etapa de seguimiento.

2.3.4 Etapa de seguimiento

Para que una reunión cumpla totalmente su objetivo es necesario que las decisiones o acuerdos alcanzados en ella se respeten. Para ello es fundamental que el convocante haga llegar, lo más pronto posible, a quienes asistieron a la reunión la minuta de la misma, en la que se incluyan todos los acuerdos adoptados, las responsabilidades asignadas y los plazos establecidos para el cumplimiento de ellas, (en la figura 2.4 se muestra el flujo de información del convocante y asistentes).

La función del convocante de una reunión no termina con la entrega (ya sea por escrito o por correo electrónico) de la minuta, sino con el seguimiento posterior de la forma en que cada participante cumple con las responsabilidades que contrajo. Lamentablemente, ésta es una práctica muy poco frecuente y es lamentable comprobar que luego de largas reuniones, en que se distribuyen diferentes tareas, éstas no se cumplen por falta de seguimiento posterior. Es por esto, que a continuación se establecieron las actividades a realizar durante esta etapa, tanto para el convocante como para el convocado, (aunque prácticamente ésta enfocada a los responsables de cada acuerdo).

Convocado.

- Realizar las actividades necesarias para cumplir en su totalidad con sus pendientes.

Convocante.

- Evaluar las tareas realizadas por cada uno de los responsables.
- Actualizar la información y enviar a cada uno de los asistentes, con la finalidad de que cada uno de los participantes se entere de los avances de sus actividades.

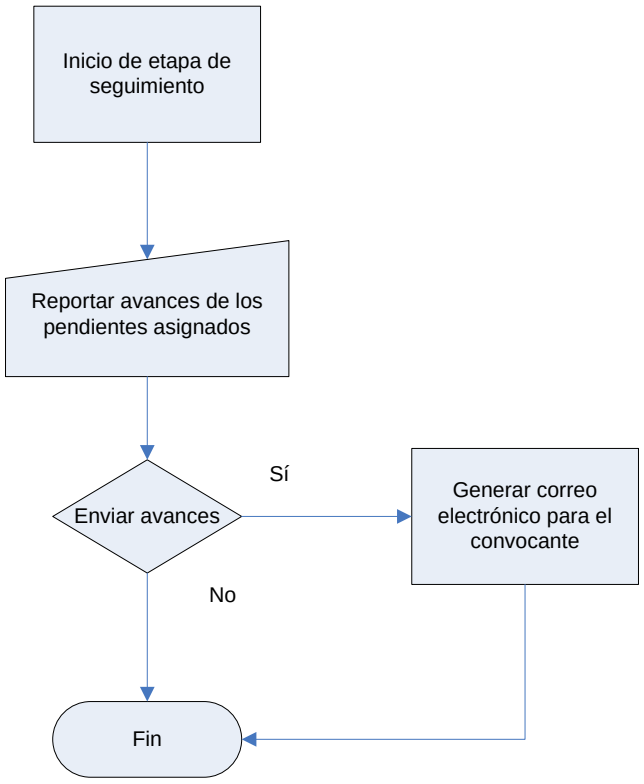


Figura 2.4 Esquema de la etapa de seguimiento

2.3.5 Etapa de Cierre

Una vez que todos los acuerdos son calificados como resueltos por parte del convocante, es necesario mandar la información a cada uno de los responsables, entonces la reunión pasa al último estatus que es cierre.

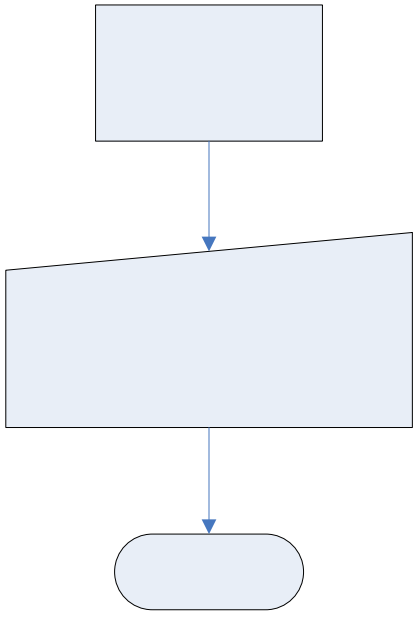


Figura. 2.5 Esquema de la etapa de cierre

Cuando la reunión llega a este estatus la información contenida solo deberá servir de consulta; algunas veces es indispensable tener un archivo de respaldo de cada una de las reuniones, sin embargo queda a criterio de cada persona el tiempo en que se debe conservar.

CAPÍTULO 3 ANÁLISIS

En este capítulo se presentan los elementos teóricos necesarios, que permitirán analizar al problema establecido e identificar las posibles soluciones; se comenzará con la teoría acerca de las metodologías de desarrollo de software.

3.1 Metodología de desarrollo

Para resolver problemas reales de una industria, un ingeniero de software o un equipo de ingenieros, deben incorporar una estrategia de desarrollo que acompañe al proceso, métodos, herramientas, modelos, etc. ingeniería de software. Se selecciona un modelo de proceso para la ingeniería de software según la naturaleza del proyecto y de la aplicación,

los métodos y las herramientas a utilizarse, y con los controles y entregas que se requieren.

Todo el desarrollo del software se puede caracterizar como un bucle de resolución de problemas (Figura 3.1) en el que se encuentran cuatro etapas distintas: Status quo, definición de problemas, desarrollo técnico e integración de soluciones. El primero representa el estado actual de sucesos; la definición de problemas identifica el problema específico a resolverse; el desarrollo técnico resuelve el problema a través de la aplicación de alguna tecnología, y la integración de soluciones ofrece resultados (p. ej.: documentos, programas, datos, nueva función comercial, producto nuevo) a los que solicitan la solución en primer lugar. Las fases y los pasos genéricos de ingeniería del software se dividen en estas etapas (Presman [2]).

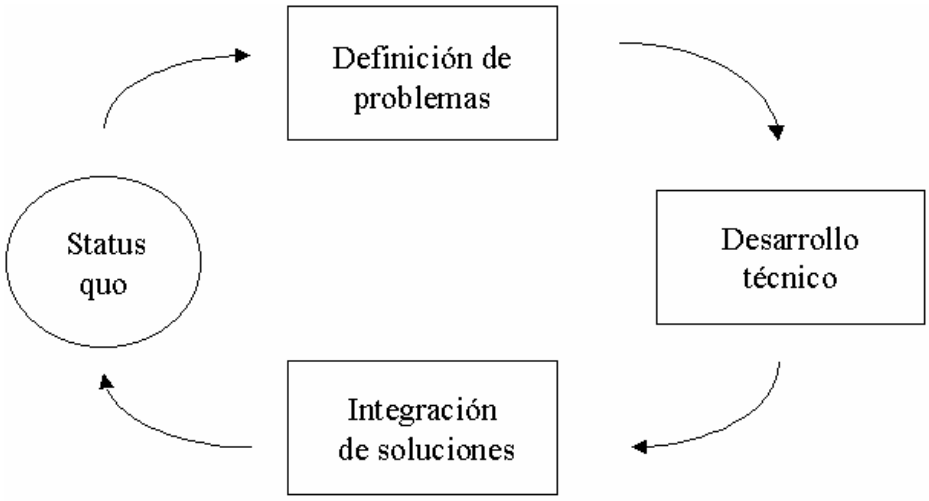


Figura 3.1 Fases de un bucle de resolución de problemas

3.1.1 Modelo de proceso

El modelo lineal secuencial para la ingeniería de software es llamado algunas veces ciclo de vida básico o modelo de cascada (Figura 3.1.1), este sugiere un enfoque sistemático y secuencial del desarrollo del software, que comienza en un nivel de sistemas y progresa con el análisis, diseño, codificación, pruebas y mantenimiento.

Modelado según el ciclo de ingeniería convencional, el modelo lineal secuencial acompaña a las actividades siguientes:

- **Ingeniería y modelado de Sistema/Información.** Como el software siempre forma parte de un sistema más grande (o empresa), el trabajo comienza estableciendo requisitos de todos los elementos del sistema y asignando al software algún subgrupo de estos requisitos. Esta visión del sistema es esencial cuando el software se debe interconectar con otros elementos como hardware, personas y bases de datos. La ingeniería y el análisis del sistema acompaña a los requisitos que se recogen en el nivel del sistema como una pequeña parte de análisis y de diseño. La ingeniería de información acompaña a los requisitos que se recogen en el nivel estratégico de empresas y en el nivel del área de negocio.

- **Análisis de los requisitos del software.** El proceso de reunión requisitos se intensifica y se centra especialmente en el software. Para comprender la naturaleza de el (los) programa (s) a construirse, el ingeniero (analista) del software debe comprender el dominio de información de software, así como la funcionalidad requerida, comportamiento, rendimiento e interconexión.

- **Diseño.** El diseño del software es realmente un proceso de muchos pasos que se centra en cuatro atributos distintos de un programa: estructura de datos, arquitectura del software, representación de interfaz y detalle procedimental.

- **Generación de código.** El diseño se debe traducir en una forma legible por la máquina. El paso de generación de código lleva a cabo esta tarea. Si se lleva a cabo el diseño de una forma detallada, la generación de código se realiza mecánicamente.

- **Pruebas.** Una vez que se ha generado un código, comienzan las pruebas del programa. El proceso de pruebas se centra en los procesos lógicos internos del software, asegurando que todas las sentencias se han comprobado, y en los procesos externos funcionales, es decir, la realización de las pruebas para la detección de errores y el sentirse seguro de que la entrada definida produzca resultados reales de acuerdo con los resultados requeridos.

- **Mantenimiento.** El software indudablemente sufrirá cambios después de ser entregado al cliente. Se producirán cambios porque se han encontrado errores, porque el software debe adaptarse para acoplarse a los cambios de su entorno

externo. El mantenimiento vuelve a aplicar cada una de las fases precedentes a un programa ya existente y no a uno nuevo.

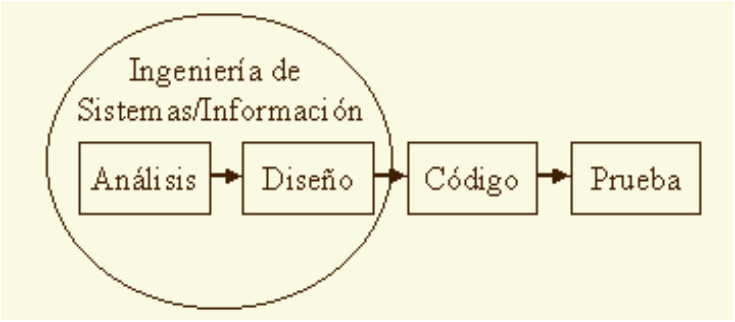


Figura 3.1.1 El modelo lineal secuencial

3.2 Lenguajes de Programación

Al desarrollarse las primeras computadoras electrónicas, se vio la necesidad de programarlas, es decir, de almacenar en memoria la información sobre la tarea que iban a ejecutar. Las primeras se usaban como calculadoras simples; se les indicaban los pasos de cálculo, uno por uno.

John Von Neumann desarrolló el modelo que lleva su nombre, para describir este concepto de "programa almacenado". En este modelo, se tiene una abstracción de la memoria como un conjunto de celdas, que almacenan simplemente números. Estos pueden representar dos cosas: los datos, sobre los que va a trabajar el programa; o bien, el programa en sí.

¿Cómo es que describimos un programa como números? Se tenía el problema de representar las acciones que iba a realizar la computadora, y que la memoria, al estar compuesta por switches correspondientes al concepto de bit, solamente nos permitía almacenar números binarios.

La solución que se tomó fue la siguiente: a cada acción que sea capaz de realizar nuestra computadora, asociarle un número, que será su código de operación (opcode) . Por ejemplo, una calculadora programable simple podría asignar los opcodes: 1 = SUMA, 2 = RESTA, 3 = MULTIPLICA, 4 = DIVIDE. Supongamos que queremos realizar la operación 5 * 3 + 2, en la calculadora descrita arriba. En memoria, podríamos "escribir" el programa de la siguiente forma:

Localidad Opcode Significado Comentario 0 5 5 En esta localidad, tenemos el primer número de la fórmula 1 3 * En esta localidad, tenemos el opcode que representa la multiplicación. 2 3 3 En esta localidad, tenemos el segundo número de la fórmula 3 1 + En esta localidad, tenemos el opcode que representa la suma. 4 2 2 En esta localidad, tenemos el último número de la fórmula

Podemos ver que con esta representación, es simple expresar las operaciones de las que es capaz el hardware (en este caso, nuestra calculadora imaginaria), en la memoria.

La descripción y uso de los opcodes es lo que llamamos lenguaje de máquina. Es decir, la lista de códigos que la máquina va a interpretar como instrucciones, describe las capacidades de programación que tenemos de ella; es el lenguaje más primitivo, depende directamente del hardware, y requiere del programador que conozca el funcionamiento de la máquina al más bajo nivel.

Los lenguajes más primitivos fueron los lenguajes de máquina. Esto, ya que el hardware se desarrolló antes del software, y además cualquier software finalmente tiene que expresarse en el lenguaje que maneja el hardware.

La programación en esos momentos era sumamente tediosa, pues el programador tenía que "bajarse" al nivel de la máquina y decirle, paso a pasito, cada punto de la tarea que tenía que realizar. Además, debía expresarlo en forma numérica; y por supuesto, este proceso era propenso a errores, con lo que la productividad del programador era muy limitada. Sin embargo, hay que recordar que en estos momentos, simplemente aún no existía alternativa.

El primer gran avance que se dio, como ya se comentó, fue la abstracción dada por el Lenguaje Ensamblador, y con él, el nacimiento de las primeras herramientas automáticas para generar el código máquina. Esto redujo los errores triviales, como podía ser el número que correspondía a una operación, que son sumamente engorrosos y difíciles de detectar, pero fáciles de cometer. Sin embargo, aún aquí es fácil para el programador perderse y cometer errores de lógica, pues debe bajar al nivel de la forma en que trabaja el CPU, y entender bien todo lo que sucede dentro de él.

Con el desarrollo en los 50s y 60s de algoritmos de más elevado nivel, y el aumento de poder del hardware, empezaron a entrar al uso de computadoras científicos de otras

ramas; ellos conocían mucho de Física, Química y otras ramas similares, pero no de Computación, y por supuesto, les era sumamente complicado trabajar con lenguaje Ensamblador en vez de fórmulas. Así, nació el concepto de Lenguaje de Alto Nivel, con el primer compilador de FORTRAN (formula translation), que, como su nombre indica, inició como un "simple" esfuerzo de traducir un lenguaje de fórmulas, al lenguaje ensamblador y por consiguiente al lenguaje de máquina. A partir de FORTRAN, se han desarrollado innumerables lenguajes, que siguen el mismo concepto: buscar la mayor abstracción posible, y facilitar la vida al programador, aumentando la productividad, encargándose los compiladores o intérpretes de traducir el lenguaje de alto nivel, al lenguaje de computadora.

Hay que notar la existencia de lenguajes que combinan características de los de alto nivel y los de bajo nivel (es decir, Ensamblador). Por ejemplo es C: contiene estructuras de programación de alto nivel, y la facilidad de usar librerías que también son características de alto nivel; sin embargo, fue diseñado con muy pocas instrucciones, las cuales son sumamente sencillas, fáciles de traducir al lenguaje de la máquina; y requiere de un entendimiento apropiado de cómo funciona la máquina, el uso de la memoria, etcétera. Por ello, muchas personas consideramos a lenguajes como C (que fue diseñado para hacer sistemas operativos), lenguajes de nivel medio.

3.2.1 Programación estructurada y sus desventajas

La programación estructurada emplea la *técnica descendente* o el *refinamiento sucesivo*, que comienza descomponiendo el programa en piezas manejables más pequeñas, conocidas como *funciones* (subrutinas, subprogramas o procedimientos), que realizan tareas menos complejas. Esta técnica introdujo el concepto de *abstracción* que se define como la capacidad para examinar algo sin preocuparse de sus datos internos.

En un programa estructurado, los *datos locales* se ocultan dentro de funciones y los *datos compartidos* se pasan como argumentos.

A medida que la complejidad de un programa crece, también lo hace su independencia de los tipos de datos fundamentales que procesa, provocando que el acceso a los mismos se convierta *crítico*.

Los programas basados en funciones son difíciles de diseñar. El problema es que sus componentes principales (funciones y estructuras de datos) no modelan bien el mundo real.

3.2.2 Programación orientada a objetos (POO)

Supongamos que se desea construir una *computadora*, ¿qué componentes se necesitarían para construirla?, entre los componentes tenemos: una tarjeta madre, un chip CPU, una tarjeta de video, un disco duro, un teclado, etc. Lo ideal sería que cuando se ensamblen todos los componentes, se tenga un sistema en donde todos estos componentes se unan para crear un sistema más grande.

En su estructura interna, cada uno de estos componentes puede ser muy complicado y fabricarse por diferentes compañías con distintos métodos de diseño. Pero no se necesita saber como funcionan los componentes o que hace cada chip en la tarjeta para poder ensamblarla y que todas las unidades interactúen entre sí. ¿Encajará la tarjeta de video en las ranuras de la tarjeta madre?, ¿Funcionará el monitor con esta tarjeta de video?, una vez que conozca las interacciones que hay entre los componentes y la forma en que coincide con aquellas, es sencillo juntar el sistema completo.

¿Qué tiene que ver esto con la programación? Todo. La POO funciona de esta manera. Al utilizarla el programa general estará formado por componentes individuales (objetos) numerosos y diferentes; cada uno de los cuales realizará su papel en el programa y todos se comunicarán en formas predefinidas.

La teoría de los objetos tiene un enfoque que se ha ido estructurando en este último tiempo de acuerdo a la creación de diseños de aplicaciones, usando una forma de pensar orientada a objetos, utilizando su propio lenguaje para tener una mejor compresión de lo que tratan estos conceptos sencillos, conocidos y aplicados en nuestro mundo continuamente.

La teoría o el análisis orientado a objetos tiene un papel preponderante en este último tiempo, apareciendo hace alrededor de dos décadas como un enfoque alternativo para el desarrollo de sistemas de software, en donde se a madurado la perspectiva clásica de los resultados, métodos estructurados para desarrollar un sistema mediante los objetos que

forman parte de él, sus estructuras, sus relaciones, sus atributos y sus operaciones, consiguiendo nombrar una abstracción del mundo real.

La programación orientada a objetos ofrece ventajas considerables en el desarrollo de aplicaciones de software, ejemplos de estas son:

- Ventajas de la abstracción de datos más disciplina de programación
- Reutilización y mantenimiento de código
- Potencia del lenguaje: herencia y polimorfismo
- Reflejar conceptos de problemas reales.

En las siguientes secciones se describen los conceptos fundamentales de POO que son: clases, objetos, atributos, métodos, herencia y polimorfismo.

3.2.3 Las clases y los objetos

Una **clase** es un tipo de dato definido por el usuario, que tiene un estado (su representación) y algunas operaciones (comportamiento). Una clase tiene datos y métodos internos en forma de procedimientos o funciones y suele describir las características genéricas y el comportamiento de diversos objetos similares. Ejemplos: Mobiliario, Persona, Vehículo, etc (Steve [3]).

Un ***objeto*** es entidad que posee atributos y acciones (métodos) que se asocian con un objeto del mundo real.

¿Qué clases de cosas pueden ser objetos en un POO? La respuesta está sólo limitada a su imaginación. Por ejemplos, objetos: aviones, automóviles, casas.

Los **atributos** son las características individuales que diferencian a un objeto y determina la apariencia, estado u otras cualidades de ese objeto.

Ejemplo:
Supongamos que poseemos el objeto *motocicleta*, con los siguientes atributos:
Cloro: rojo, verde, plateado, café.
Estilo: crucero, deportivo, estándar.
Fabricante: Honda, BMW, Harley-Davidson.

Características como la condición de la maquina (apagado o encendida).
Velocidad actual seleccionada.

Un POO consiste en un número de objetos que se comunican unos con otros, llamando a funciones miembro. Los procedimientos y funciones, denominados ***métodos o funciones*** miembros, residen en el objeto y determinan como actúan los objetos cuando reciben un mensaje. Un mensaje es la acción que hace un objeto. Un método especifica como se ejecuta un mensaje y como cambian el estado de un objeto.

Al conjunto de mensajes a los cuales puede responder un objeto se denomina *protocolo del objeto*. Ejemplo:
Tomando el caso del objeto *motocicleta*, se pueden tener los siguientes métodos:
Arrancar la motocicleta, Detener la motocicleta, Acelerar, Cambiar velocidades, Frenar.

Cuando se ejecuta un POO ocurren tres sucesos:

- Los objetos se crean a medida que se necesitan
- Los mensajes se mueven de un objeto a otro a medida que el programa procesa información internamente o responde a la entrada del usuario
- Cuando los objetos ya no son necesarios se borran y se libera la memoria.

Delphi dispone del Object Pascal que es un lenguaje orientado a objetos. Como su propio nombre indica, es una versión del lenguaje Pascal original pero orientado a objetos. Esto denota que en el Object Pascal es posible definir clases de objetos, que serían los moldes a partir de los cuales, posteriormente, se crearían copias de ese objeto. Existen multitud de clases predefinidas, disponibles para ser usadas por nosotros.

Clases y objetos son términos utilizados comúnmente en Object Pascal y otros lenguajes OOP (Programación orientada a objetos). A continuación se muestra un ejemplo de una clase en Object Pascal, con algunos campos de datos locales y métodos, la sintaxis es la siguiente:

```
type
  TDate = class
      Month, Day, Year: Integer;
      procedure SetValue (m, d, y: Integer);
```

```
      function LeapYear: Boolean;
  end;

procedure TDate.SetValue(m, d, a: Integer);
begin
   Month := m;
   Day := d;
   Year := y;
end;

function TDate.LeapYear: Boolean;
begin
  Result := IsLeapYear(Year);
end;
```

Una vez definida la clase, podemos crear un objeto y utilizarlo como sigue:

```
var
  ADay : TDate;
begin
   ADay := TDate.Create;
   ADay.SetValue(1, 1, 2000);
   if ADay.LeapYear then
      ShowMessage('Saltar año: ' + IntToStr(ADay.Year));
  ADay.Free;
end;
```

Para declarar una variable de un tipo de clase, hay lenguajes que crean una instancia de esa clase, pero Object Pascal se basa en un modelo de referencia a objetos. La idea es que cada variable de un tipo de clase, como ADay en el fragmento del código anterior, no aloje el valor del objeto, sino que contenga una referencia o un puntero para indicar la posición de memoria en que se almacena el objeto.

3.2.4 Privado, protegido y público

Una clase puede tener cualquier cantidad de datos y métodos. Sin embargo, para utilizar el método orientado a objetos adecuado, los datos han de estar ocultos o encapsulados dentro de la clase que los utiliza. Cuando se accede a una fecha, por ejemplo, no tiene sentido cambiar el valor del día por sí mismo. De hecho cambiar el valor del día puede resultar una fecha invalida, como 30 de Febrero. La utilización de métodos para acceder a la representación interna de un objeto limita el riesgo de generar situaciones erróneas, puesto que los métodos pueden comprobar si la fecha es valida y rechazar modificar el nuevo valor en caso de que no lo sea. El encapsulado es importante porque permite a la persona que escribe la clase modificar la representación interna en una versión posterior (Steve [3]).

El concepto de encapsulado es muy sencillo: sólo hay que pensar que una clase es una "caja negra" con una pequeña parte visible. La parte visible, llamada interfaz de la clase, permite a las otras partes del programa acceder y utilizar los objetos de esa clase. Sin embargo, cuando utilizamos los objetos, la mayor parte de su código se encuentra oculto. Normalmente no sabemos qué datos internos tienen el objeto y no tenemos manera de acceder directamente a los datos. Desde luego se supone que vamos a utilizar métodos para acceder a los datos, que están blindados contra accesos no autorizados. Éste es el método orientado a objetos para un concepto clásico de programación que se conoce como información oculta.

Object Pascal tiene tres especificadores de acceso: private, protected y public. Los tres básicos son los siguientes:

- La directiva private indica campos y métodos de una clase que no es accesible desde fuera de la unidad (el archivo de código fuente) que declara la clase

- La directiva public indica campos y métodos a los que se puede acceder libremente desde cualquier parte del programa, así como en la unidad en que están definidos

- La directiva protected se utiliza para indicar métodos y campos con visibilidad limitada. Solamente la clase actual y sus subclases pueden acceder a elementos que estén protegidos con esta directiva.

En general, los campos de una clase deben ser privados y los métodos públicos. No obstante, no siempre ocurre de esta manera. Los métodos pueden se privados o protegidos si es necesario realizar cálculos parciales internamente. Los campos pueden ser protegidos o públicos para tener acceso más fácil y directo y cuando existe la seguridad de no tener que cambiar la definición del tipo.

3.2.5 Herencia

Propiedad de los objetos mediante la cual los ejemplares de una clase pueden tener acceso a los datos y definiciones de métodos contenidos en una clase previamente definida, sin que tales definiciones se declaren de nuevo. Cuando una clase hereda de otra esta es una *subclase*. A la clase base de la cual se hereda se llama *superclase (Cantu [8]).*

A menudo es necesario utilizar una versión ligeramente diferente de una clase existente que hemos escrito o que alguien nos ha entregado. Por ejemplo; puede ser preciso agregar un nuevo método o un ligero cambio a uno existente. Esto se puede hacer con facilidad modificando el código original, a menos que seamos capaces de utilizar dos versiones distintas de la clase en diferentes circunstancias. Además, si la clase fue escrita originalmente por otra persona, conviene mantener nuestras modificaciones aparte.

Una alternativa típica es hacer una copia de la definición del tipo original, cambiar el código para que soporte las nuevas funciones y dar un nuevo nombre a la clase resultante. Esto puede funcionar, pero también puede dar problemas: al duplicar el código también se duplican los errores; y si queremos añadir una nueva función, necesitaremos agregarla dos veces más, según el número de copias que hayamos hecho del código original. Este método resulta en dos tipos de datos completamente diferentes, por lo que el compilador no puede facilitarnos obtener beneficios de las similitudes entre ambos tipos.

Para solucionar este tipo de problemas al expresar similitudes entre clases, Object Pascal permite definir una nueva clase directamente a partir de una existente. Para heredar de una clase existente, solamente necesitamos indicar esa clase al principio de la

declaración de la subclase. Por ejemplo, Delphi lo hace automáticamente cada vez que creamos una nueva forma.

3.2.6 Polimorfismo

Sinónimo de sobrecarga. Identificador de procedimiento o de método que denota más de un procedimiento.

Las funciones y procedimientos de Pascal se suelen basar en el enlace estático, que también se llama enlace anterior. Esto quiere decir que una llamada a un método la resuelve el compilador o el editor de enlaces, que reemplazan la demanda con una llamada a la posición específica de la memoria en que reside la función o el procedimiento. Los lenguajes de programación orientados a objetos permiten el empleo de otra forma de enlace, llamado enlace dinámico o enlace posterior. En este caso, la dirección real del método a llamar viene determinada en tiempo de ejecución, en base al tipo de la instancia utilizada para hacer la llamada (Steve [3]).

La ventaja de esta técnica se llama polimorfismo, que significa que podemos escribir una llamada a un método, aplicándolo a una variable, pero el método al que realmente llama Delphi depende del tipo de objeto con el que se relaciona la variable. Delphi no puede determinar la clase real del objeto al que se refiere la variable hasta hallarse en tiempo de ejecución, simplemente debido a la regla de compatibilidad de tipos.

Por ejemplo, supongamos que tanto una clase como su subclase (digamos TAnimal y TDog) definen un método, y que este método tiene enlace posterior. Entonces podemos aplicar este método a una variable genérica, como MyAnimal, que se puede referir, en tiempo de ejecución, tanto a un objeto de la clase TAnimal como a un objeto de la clase TDog. El método real a llamar viene determinado en tiempo de ejecución, según la clase del objeto actual.

```
type
  TAnimal  = class
   public
     function Voice: String; virtual;
```

```
 TDog = class (TAnimal)
 public
    function Voice: String; override;
```

la implementación de los métodos podría como sigue:

```
uses
  MMSystem;

function  TAnimal.Voice: String;
begin
  Voice := 'Voice of the animal';
  PlaySound ('Anim.wav', 0, snd_Async);
end;
function Tdog.Voice :String;
begin
 Voice := 'Arf Arf';
 PlaySound(''dog.wav', 0, snd_Async);
end;
```

El efecto de la llamada MyAnimal.Voice depende de si la variable MyAnimal activa se refiere a un objeto de la clase TAnimal, en cuyo caso, llamará al método TAnimal.Voice. Si se refiere a un objeto de la clase TDog, llamará al método TDog.Voice. Esto solamente sucede debido a que la función es virtual.

La llamada a MyAnimal.Voice funciona para un objeto que es una instancia de cualquier secundario de la clase TAnimal, incluso clases definidas después de la llamada a este método o fuera de su alcance. El compilador no necesita conocer todos los métodos secundarios para hacer la llamada compatible con ellos; solamente si la clase principal es necesaria. En otras palabras, esta llamada a MyAnimal.Voice es compatible con todas las futuras subclases TAnimal.

Ésta es la razón técnica fundamental por la que los lenguajes de programación orientados a objetos favorecen la reutilización. Podemos escribir código que utilice clases dentro de una jerarquía. En otras palabras, la jerarquía (y el programa) sigue siendo ampliable, a

pesar de que hayamos escrito miles de líneas de código utilizándola. Desde luego, existe una condición y es que las clases principales de la jerarquía tienen que designarse prestando mucha atención.

3.2.7 Sobrescritura, redefinición y reintroducción de métodos.

Como hemos visto, para sobre escribir un método de enlace posterior en una clase secundaria, es necesario utilizar la palabra clave override. Obsérvese que esto puede tener lugar solamente si el método se ha definido como virtual en la clase principal. De otra manera, si era un método estático, la única forma de activar el enlace posterior es modificar el código de la clase principal.

Las reglas son sencillas: un método definido como estático permanece estático en todas las subclases, a menos que se oculte con un nuevo método virtual que tenga el mismo nombre. Un método definido como virtual permanece con enlace posterior en todas las subclases. No hay forma de cambiarlo, debido a la manera en que el compilador genera código diferente para los métodos de enlace posterior.

Para redefinir un método estático, simplemente hay que agregar un método a una subclase que tenga los mismos o diferentes parámetros que el original, sin necesidad de proporcionar más especificaciones. Para sobrescribir un método virtual hay que especificar los mismos parámetros y utilizar la palabra clave override:

```
MyClass = class
    procedure One; virtual;
    procedure Two; (static method)
 end;
```

Hay dos formas típicas de sobrescribir un método. Una es reemplazar el método de la clase principal con una nueva versión. La otra es añadir más código al método existente. Esto se puede llevar acabo utilizando la palabra clave inherited para llamar al mismo método de la clase principal. Por ejemplo, podemos escribir:

```
procedure MySubClassl.One;
begin
```

```
  //nuevo código
  ...
  //llamada al procedimiento heredado MyClass.One
  inherited One;
end;
```

Cabe preguntarse para qué es necesario utilizar la palabra clave override. En otros lenguajes, al redefinir un método de una subclase, automáticamente sobrescribimos el original. Sin embargo, tener una palabra clave específica permite al compilador comprobar la correspondencia entre los nombres de los métodos de la clase principal y la subclase (en otros lenguajes OOP, son muy comunes los errores mecanográficos al redefinir una función), verificar que el método era virtual en la clase principal, etc.

3.2.8 Métodos abstractos

La palabra clave abstract se utiliza para declarar métodos que se van a definir solamente en subclases de la clase actual. La directiva abstract define totalmente el método. Si tratamos de dar una definición para el método, el compilador se quejará. En Object Pascal, podemos crear instancias de clases que tengan métodos abstract.

Cabe preguntarse el motivo de utilizar métodos abstractos. La razón es el uso del polimorfismo. Si la clase TAnimal tiene el método abstracto Voice, todas las subclases pueden redefinirlo. La ventaja es que ahora podemos utilizar el objeto genérico MyAnimal para hacer referencia a cada animal definido por una subclase e invocar a este método. Si el método no está presente en la interfaz de la clase TAnimal, el compilador no permitirá la llamada, pues realiza una comprobación de tipos estáticos. El uso del objeto genérico MyAnimal permite llamar solo al método definido por su propia clase, TAnimal.

No podemos llamar a métodos dados por subclases, a menos que la clase principal tenga como mínimo la declaración de este método, en forma de un método abstracto. El siguiente ejemplo, Animals3, muestra el uso de los métodos abstractos y el error de llamada abstracto. Las interfaces de sus clases son:

Type

```
TAnimal = class
public
  Constructor Create;
  function GetKind: String;
  function Voice: String; virtual; abstract;
private
   Kind: String;
end;

TDog = class (TAnimal)
public
   Constructor Create;
   function Voice: String; override;
   function Eat: String; virtual;
end;

TCar = class (TAnimal)
public
   Constructor Create;
   function Voice: String; override;
   function Eat: String; virtual;
end;
```

La parte más interesante es la definición de la clase TAnimal, que incluye un método virtual abstracto: Voice. También es importante tener en cuenta que cada clase derivada sobrescribe esta definición y agrega un nuevo método virtual, Eat. Las implicaciones de estos dos diferentes aspectos consistentes en que para llamar a la función Voice, simplemente tenemos que escribir el mismo código que en la versión anterior del programa:

3.3 BASES DE DATOS

Hoy, la importancia e impacto de las bases de datos es incuestionable a medida que organizaciones gubernamentales, instituciones académicas y entidades comerciales crean y mantienen importantes bases de datos que contienen toda clase de información desde documentos de texto en lenguaje natural, tablas estadísticas, datos financieros y otros.

3.3.1 Definiciones

Es necesario comenzar con algunos conceptos básicos para el mejor entendimiento de este capítulo, las siguientes definiciones están involucradas en el tema de base de datos.

Base de datos es un conjunto de datos almacenados entre los que existen relaciones lógicas y que ha sido diseñada para satisfacer los requerimientos de información de una empresa u organización (Date [5]).

Dato es un conjunto de caracteres con algún significado, pueden ser numéricos, alfabéticos, o alfanuméricos.

Información es un conjunto ordenado de datos los cuales son manejados según la necesidad del usuario, para que un conjunto de datos pueda ser procesado eficientemente y pueda dar lugar a información, primero se debe guardar lógicamente en archivos.

Entidad.- Persona, lugar, objeto o evento de interés acerca del cual se recogen o procesan datos. Por ejemplo: pacientes, clientes, artículos son entidades de un hospital y de una tienda comercial respectivamente.

Campo.- Conjunto de datos de un mismo tipo. Por ejemplo: conjunto de nombres, conjunto de notas, conjunto de direcciones, etc.

Registro.- Conjunto de datos pertenecientes a una misma entidad. El registro consta de campos, cada campo tiene una longitud definida, por lo tanto los registros son de longitud fija.

Atributos de una entidad.- Cada entidad tiene características propias. Por ejemplo: la entidad de alumnos tiene las siguientes características: nombres, apellidos, edad, sexo, fecha de nacimiento, grado, dirección, teléfono, etc. a cada una de estas características o propiedades de la entidad se denomina atributo de la entidad.

Tabla.- Es un conjunto de datos dispuesto en una estructura de filas y columnas. En una tabla las filas se denominan registros y las columnas campos. En una tabla la primera fila contiene los nombres de campo. Cada campo contiene determinado tipo de datos y tiene una longitud expresada en el número de caracteres máximo del campo. Para crear una tabla será necesario definir su estructura:

a. El nombre de la tabla

b. Los tipos de dato de cada campo

c. Las propiedades o características de cada campo

d. El campo clave

Tipos De Datos

TIPO DE DATO	**ALMACENA**	**TAMAÑO**
Texto	Caracteres alfanuméricos	Hasta 255 bytes
Memo	Caracteres alfanuméricos	Hasta 64000 bytes
Numérico	Valores Enteros o fraccionarios	1,2,4 u 8 bytes
Fecha/Hora	Fechas y Horas	8 bytes
Moneda	Valores de moneda	8 bytes
Autonumérico	Valor numérico de incremento automático	4 bytes
Si/No	Valores Booleanos (Verdadero o Falso)	1 byte
Objeto OLE	Imágenes o Gráficos	Hasta 1 Gb.

Tipo De Dato Numérico

VALOR DE CAMPO	RANGO	LUGARES DECIMALES	TAMAÑO
Byte	0 – 255	Ninguno	1 byte
Entero	-32.768 a 32.767	Ninguno	2 byte
Entero Largo	-2.147.486.648 a 2.147486.647	Ninguno	4 byte
Simple	-3.4*10 a 3.4*10	7	4 byte
Doble	-1.797*10 a 1.797*10	15	8 byte

Propiedades Del Campo: Es la apariencia que tiene los datos, evita la introducción incorrecta de los mismos, especifica valores predeterminados, acelera la búsqueda y la ordenación de la tabla mediante índices. Las propiedades de los campos se visualizan y se modifican individualmente para cada campo.

PROPIEDADES DEL CAMPO	PARA
Tamaño de campo	Ajusta el tamaño de un campo de tipo texto o limita el rango de valores permitidos en un campo numérico.
Máscara de entrada	Presentación de un formato de un campo para no escribirlos y los datos se ajusten a la máscara.
Reglas de validación	Permite limitar los datos introducidos en un campo.
Texto de validación	Texto que se muestra al infringir la regla de validación

Llave primaria: Es aquel valor clave que hace único a un registro dentro de una tabla.

Llave foránea: Es aquel valor clave que esta contenido dentro una tabla como una columna más, sin formar parte de la llave primaria.

Sistema Manejador de base de datos (DBMS)

Un DBMS es una colección de numerosas rutinas de software interrelacionadas, cada una de las cuales es responsable de una tarea específica. El objetivo primordial de un sistema manejador de base de datos es proporcionar un contorno que sea a la vez conveniente y eficiente para ser utilizado al extraer, almacenar y manipular información de la base de datos. Todas las peticiones de acceso a la base, se manejan centralizadamente por medio del DBMS, por lo que este paquete funciona como interfase entre los usuarios y la base de datos (Korth [6]).

El DBMS permite a los usuarios definir, crear y mantener la base de datos, y proporciona acceso controlado a la misma.

El DBMS es la aplicación que interacciona con los usuarios de los programas de aplicación y la base de datos. En general, proporciona los siguientes servicios:

- Permite la definición de la base de datos mediante el *lenguaje de definición de datos*. Este lenguaje permite especificar la estructura y el tipo de los datos, así como las restricciones sobre los datos. Todo esto se almacenará en la base de datos

- Permite la inserción, actualización, eliminación y consulta de datos mediante el *lenguaje de manejo de datos*. El hecho de disponer de un lenguaje para realizar consultas reduce el problema de los sistemas de ficheros, en los que el usuario tiene que trabajar con un conjunto fijo de consultas, o bien, dispone de un gran número de programas de aplicación costosos de gestionar

- Hay dos tipos de lenguajes de manejo de datos: los *procedurales* y los *no procedurales*. Estos dos tipos se distinguen por el modo en que acceden a los datos. Los lenguajes procedurales manipulan la base de datos registro a registro, especifica qué operaciones se deben realizar para obtener los datos resultado, mientras que los no procedurales operan sobre conjuntos de registros y se especifica qué datos deben

obtenerse sin decir cómo hacerlo; el más utilizado es el SQL (Structured Query Language) que, de hecho, es un estándar y es el lenguaje de los DBMS relacionales.

- Proporciona un acceso controlado a la base de datos mediante:
  - un sistema de seguridad, de modo que los usuarios no autorizados no puedan acceder a la base de datos
  - un sistema de integridad que mantiene la integridad y la consistencia de los datos
  - un sistema de control de concurrencia que permite el acceso compartido a la base de datos
  - un sistema de control de recuperación que restablece la base de datos después de que se produzca un fallo del *hardware* o del *software*
  - un diccionario de datos o catálogo accesible por el usuario que contiene la descripción de los datos de la base de datos.

El DBMS gestiona la estructura física de los datos y su almacenamiento. Con esta funcionalidad, se convierte en una herramienta de gran utilidad. Sin embargo, desde el punto de vista del usuario, se podría discutir que los DBMS han hecho las cosas más complicadas, ya que ahora los usuarios ven más datos de los que realmente quieren o necesitan, puesto que ven la base de datos completa. Conscientes de este problema, los DBMS proporcionan un mecanismo de *vistas* que permite que cada usuario tenga su propia vista o visión de la base de datos. El lenguaje de definición de datos permite definir vistas como subconjuntos de la base de datos.

Las vistas, además de reducir la complejidad permitiendo que cada usuario vea sólo la parte de la base de datos que necesita, tienen otras ventajas:

- Las vistas proporcionan un nivel de seguridad, ya que permiten excluir datos para que ciertos usuarios no los vean
- Las vistas proporcionan un mecanismo para que los usuarios vean los datos en el formato que deseen
- Una vista representa una imagen consistente y permanente de la base de datos, incluso si la base de datos cambia su estructura.

Todos los DBMS no presentan la misma funcionalidad, depende de cada producto. En general, los grandes DBMS multiusuario ofrecen todas las funciones que se acaban de citar y muchas más. Los sistemas modernos son conjuntos de programas extremadamente complejos y sofisticados, con millones de líneas de código y con una documentación consistente en varios volúmenes. Lo que se pretende es proporcionar un sistema que permita gestionar cualquier tipo de requisitos y que tenga un 100% de fiabilidad ante cualquier fallo *hardware* o *software*. Los DBMS están en continua evolución, tratando de satisfacer los requerimientos de todo tipo de usuarios. Por ejemplo, muchas aplicaciones de hoy en día necesitan almacenar imágenes, vídeo, sonido, etc. Para satisfacer a este mercado, los DBMS deben cambiar. Conforme vaya pasando el tiempo irán surgiendo nuevos requisitos, por lo que los DBMS nunca permanecerán estáticos.

Las tecnologías de bases de datos, incluyendo los métodos de acceso, se están desarrollando rápidamente para mantenerse al día con esta demanda de mecanismos de administración de la información, ya que actualmente se manejan cantidades excesivamente muy grandes.

En este capítulo se presentan los argumentos suficientes del por qué empleamos una base de datos y los elementos necesarios para su diseño, describiendo el uso de cada una de las entidades que lo conforman.

Puesto que es indispensable conservar el registro de cada una de las reuniones a las cuales se asiste, es necesario emplear como herramienta de almacenamiento una base de datos en donde se conserve toda la información referente a las mismas, y la cual se pueda consultar en el momento en que se requiera.

Los sistemas que de base de datos se diseñan de tal forma que se puedan manejar grandes cantidades de información, ya que la manipulación de los datos involucra tanto la definición de estructuras para el almacenamiento como la provisión de mecanismos para el manejo de la información; además, un sistema de base de datos debe tener funciones de seguridad que garanticen la integridad de la información, a pesar de caídas que se presenten del sistema o intentos de accesos no autorizados.

El objetivo principal de un sistema de base de datos es proporcionar a los usuarios finales una visión abstracta de los datos, y esto se logra escondiendo ciertos detalles de como se almacenan y mantienen los datos.

La base de datos es un gran almacén de datos que se define una sola vez y que puede utilizarse al mismo tiempo por varios usuarios. En lugar de trabajar con información redundante, todos los datos se integran con una mínima cantidad de duplicidad. Además, la base de datos no sólo contiene los datos de la organización, sino también guarda una descripción de dichos datos, que se almacena en el diccionario de datos.

3.3.2 Ventajas de utilizar una base de datos

- **Disminución de redundancia e inconsistencia de datos**

Puesto que los archivos que mantienen almacenada la información son creados por diferentes tipos de programas de aplicación, existe la posibilidad de que si no se controla detalladamente el almacenamiento, se pueda originar un duplicado de información, es decir que la misma información se conserve más de una vez en un dispositivo de almacenamiento. Esto aumenta los costos de almacenamiento y acceso a los datos, además de que puede originar la inconsistencia de los datos es decir diversas copias de un mismo dato no concuerdan entre sí; por ejemplo: que se actualiza la dirección de un asistente en un registro y que en otros permanezca la anterior.

- **Facilidad en el acceso a los datos**

Un sistema de base de datos debe contemplar un entorno de datos que le facilite al usuario el manejo de los mismos. Suponiendo que se desea conocer todos los pendientes generados en cierto periodo de tiempo, si el diseño del sistema es eficiente entonces se tendrá la funcionalidad para este tipo de consultas.

La facilidad del acceso a los datos va muy ligada con el diseño de la base de datos, por lo que es importante que en la etapa de diseño se contemplen los requerimientos establecidos en un inicio.

- **Concentración de los datos**

Con una base de datos se puede concentrar toda la información en un solo archivo, evitando que los datos estén repartidos en varios archivos con diferentes formatos.

- **Acceso concurrente**

En algunos casos, se requiere que varios usuarios tengan acceso a la misma base de datos, sobre todo aquellos casos en los que el director y su asistente quieren consultar la misma información.

- **Seguridad**

La información personal y de toda empresa es importante, aunque unos datos lo son más que otros; por lo que se debe considerar el control de acceso a los mismos de tal forma que no todos los usuarios puedan visualizar la misma información, por tal motivo para que un sistema de base de datos sea confiable debe mantener un grado de seguridad que garantice la autentificación y protección de los datos.

- **Integridad**

Los valores de los datos almacenados en la base de datos deben satisfacer cierto tipo de restricciones de consistencia. Se puede lograr que estas restricciones se cumplan mediante líneas de programación o colocando reglas de validación en los campos de las tablas que las requieran.

3.4 Etapas del diseño de una base de datos

A continuación se describen a detalle los objetivos de cada una de las etapas del diseño de bases de datos: diseño conceptual, diseño lógico y diseño físico.

3.4.1 Diseño conceptual

En esta etapa se debe construir un esquema conceptual de la información que se usa en una reunión; al construir el esquema, se descubre el significado de los datos, se encuentran entidades, atributos y relaciones. El objetivo es comprender:

- la perspectiva que el usuario tiene de los datos
- la naturaleza de los datos, independientemente de su representación física
- el uso de los datos a lo largo de las etapas de la reunión.

El esquema conceptual se construye utilizando la información que se encuentra en la especificación de los requisitos de usuario. El diseño conceptual es completamente independiente de los aspectos de implementación, como pueden ser los programas de aplicación, los lenguajes de programación, el hardware disponible o cualquier otra consideración física. Durante todo el proceso de desarrollo del esquema conceptual éste se prueba y se valida con los requisitos de los usuarios. El esquema conceptual es una fuente de información para el diseño lógico de la base de datos.

3.4.2 Diseño lógico

El diseño lógico es el proceso de construir un esquema de la información que utiliza el registro de una reunión, basándose en un modelo de base de datos específico.

En esta etapa, se transforma el esquema conceptual en un esquema lógico que utilizará el modelo relacional de la base de datos en el que se basa el sistema. Conforme se va desarrollando el esquema lógico, éste se va probando y validando con los requisitos del usuario.

El esquema lógico es una fuente de información para el diseño físico. Además, juega un papel importante durante la etapa de mantenimiento del sistema, ya que permite que los futuros cambios que se realicen sobre los programas de aplicación o sobre los datos, se representen correctamente en la base de datos.

3.4.3 Normalización

La *normalización* es una técnica que se utiliza para comprobar la validez de los esquemas lógicos basados en el modelo relacional, ya que asegura que las relaciones (tablas) obtenidas no tengan datos redundantes.

La normalización se utiliza para mejorar el esquema lógico, de modo que satisfaga ciertas restricciones que eviten la duplicidad de datos. La normalización garantiza que el esquema resultante se encuentre lo más próximo al flujo de la información, que sea consistente y que tenga la mínima redundancia y la máxima estabilidad.

La normalización es un proceso que permite decidir a qué entidad pertenece cada atributo. Uno de los conceptos básicos del modelo relacional es que los atributos se agrupan en entidades (tablas) porque están relacionados a nivel lógico. En la mayoría de las ocasiones, una base de datos normalizada no proporciona la máxima eficiencia, sin embargo, el objetivo ahora es conseguir una base de datos normalizada por las siguientes razones:

- Un esquema normalizado organiza los datos de acuerdo a sus dependencias funcionales
- El esquema lógico no tiene porqué ser el esquema final. Debe representar el significado de los datos, de hecho, la normalización obliga a entender completamente a cada uno de los atributos que se han de representar en la base de datos
- Un esquema normalizado es robusto y carece de redundancias, por lo que está libre de ciertas anomalías que éstas pueden provocar cuando se actualiza la base de datos
- Los equipos informáticos de hoy en día son mucho más potentes, por lo que en ocasiones es más razonable implementar bases de datos fáciles de manejar
- La normalización produce bases de datos con esquemas flexibles que pueden extenderse con facilidad.

La normalización se lleva a cabo en una serie pasos. Cada paso corresponde a una forma normal que tiene unas propiedades. Conforme se va avanzando en la normalización, las relaciones tienen un formato más estricto y, por lo tanto, son menos vulnerables a las anomalías de actualización. El modelo relacional sólo requiere un conjunto de relaciones en primera forma normal. Las restantes formas normales son opcionales. Sin embargo, para evitar las anomalías de actualización, es recomendable llegar al menos a la tercera forma normal.

a) Primera forma normal (1FN)

Una relación está en primera forma normal si, y sólo si, todos los dominios de la misma contienen valores atómicos, es decir, no hay grupos repetitivos. Si se ve la relación gráficamente como una tabla, estará en 1FN si tiene un solo valor en la intersección de cada fila con cada columna.

Si una relación no está en 1FN, hay que eliminar de ella los grupos repetitivos. Un grupo repetitivo será el atributo o grupo de atributos que tiene múltiples valores para cada tupla de la relación. Hay dos formas de eliminar los grupos repetitivos. En la primera, se repiten los atributos con un solo valor para cada valor del grupo repetitivo. De este modo, se introducen redundancias ya que se duplican valores, pero estas redundancias se eliminarán después mediante las restantes formas normales. La segunda forma de eliminar los grupos repetitivos consiste en poner cada uno de ellos en una relación aparte, heredando la clave primaria de la relación en la que se encontraban.

b) Segunda forma normal (2FN)

Una relación está en segunda forma normal si, y sólo si, está en 1FN y, además, cada atributo que no está en la clave primaria es completamente dependiente de la clave primaria.

La 2FN se aplica a las relaciones que tienen claves primarias compuestas por dos o más atributos. Si una relación está en 1FN y su clave primaria es simple (tiene un solo atributo), entonces también está en 2FN. Las relaciones que no están en 2FN pueden sufrir anomalías cuando se realizan actualizaciones.

Para pasar una relación en 1FN a 2FN hay que eliminar las dependencias parciales de la clave primaria. Para ello, se eliminan los atributos que son funcionalmente dependientes y se ponen en una nueva relación con una copia de su determinante (los atributos de la clave primaria de los que dependen).

c) Tercera forma normal (3FN)

Una relación está en tercera forma normal si, y sólo si, está en 2FN y, además, cada atributo que no está en la clave primaria no depende transitivamente de la clave primaria. La dependencia es transitiva si existen las dependencias, siendo atributos o conjuntos de atributos de una misma relación.

Aunque las relaciones en 2FN tienen menos redundancias que las relaciones en 1FN, todavía pueden sufrir anomalías frente a las actualizaciones. Para pasar una relación de 2FN a 3FN hay que eliminar las dependencias transitivas. Para ello, se eliminan los atributos que dependen transitivamente y se ponen en una nueva relación con una copia de su determinante (el atributo o atributos no clave de los que dependen).

d) Forma normal de Boyce-Codd (BCFN)

Una relación está en la forma normal de Boyce-Codd si, y sólo si, todo determinante es una clave candidata.

La 2FN y la 3FN eliminan las dependencias parciales y las dependencias transitivas de la clave primaria. Pero este tipo de dependencias todavía pueden existir sobre otras claves candidatas, si éstas existen. La BCFN es más fuerte que la 3FN, por lo tanto, toda relación en BCFN está en 3FN.

La violación de la BCFN es poco frecuente ya que se da bajo ciertas condiciones que raramente se presentan. Se debe comprobar si una relación viola la BCFN si tiene dos o más claves candidatas compuestas que tienen al menos un atributo en común.

Tanto el diseño conceptual, como el diseño lógico, son procesos iterativos, tienen un punto de inicio y se van refinando continuamente. El diseño conceptual y el diseño lógico son etapas clave para conseguir un sistema que funcione correctamente. Si el esquema no es una representación fiel del flujo de información, será difícil, definir todas las vistas de usuario (esquemas externos), o mantener la integridad de la base de datos. Además, hay que tener en cuenta que la capacidad de ajustarse a futuros cambios es un sello que identifica a los buenos diseños de bases de datos. Por todo esto, es fundamental producir el mejor esquema que sea posible.

3.4.4 Diseño físico

El diseño físico es el proceso de producir la descripción de la implementación de la base de datos en memoria secundaria: estructuras de almacenamiento y métodos de acceso que garanticen un acceso eficiente a los datos.

Para llevar a cabo esta etapa, se debe haber decidido cuál es el DBMS que se va a utilizar, ya que el esquema físico se adapta a él. Entre el diseño físico y el diseño lógico

hay una realimentación, ya que algunas de las decisiones que se toman durante el diseño físico, pueden afectar a la estructura del esquema lógico.

En general, el propósito del diseño físico es describir cómo se va a implementar físicamente el esquema lógico obtenido en la fase anterior, en el modelo relacional, esto consiste en:

- Obtener un conjunto de relaciones (tablas) y las restricciones que se deben cumplir sobre ellas
- Determinar las estructuras de almacenamiento y los métodos de acceso que se van a utilizar para conseguir unas prestaciones óptimas
- Diseñar el modelo de seguridad del sistema.

El modelo seguido con los sistemas de bases de datos, en donde se separa la definición de los datos de los programas de aplicación, es muy similar al modelo que se sigue en la actualidad para el desarrollo de programas, en donde se da una definición interna de un objeto y una definición externa separada. Los usuarios del objeto sólo ven la definición externa y no se deben preocupar de cómo se define internamente el objeto y cómo funciona. Una ventaja de este modelo, conocido como abstracción de datos, es que se puede cambiar la definición interna de un objeto sin afectar a sus usuarios ya que la definición externa no se ve alterada. Del mismo modo, los sistemas de bases de datos separan la definición de la estructura de los datos, de los programas de aplicación y almacenan esta definición en la base de datos. Si se añaden nuevas estructuras de datos o se modifican las ya existentes, los programas de aplicación no se ven afectados ya que no dependen directamente de aquello que se ha modificado.

3.4.5 Diseño de transacciones

Una transacción es un conjunto de acciones llevadas a cabo por un usuario o un programa de aplicación, que acceden o cambian el contenido de la base de datos. Las transacciones representan eventos del mundo real, como registrar un inmueble para ponerlo en alquiler, concertar una visita con un cliente a un inmueble, dar de alta un nuevo empleado o registrar un nuevo cliente. Estas transacciones se deben realizar sobre la base de datos para que ésta siga siendo un fiel reflejo de la realidad.

Una transacción puede estar compuesta por varias operaciones, como la transferencia de dinero de una cuenta bancaria a otra. Sin embargo, desde el punto de vista del usuario, estas operaciones conforman una sola tarea.

El DBMS garantiza la consistencia de la base de datos incluso si se produce algún fallo, y también garantiza que una vez se ha finalizado una transacción, los cambios realizados por ésta quedan permanentemente en la base de datos, no se pueden perder ni deshacer (a menos que se realice otra transacción que compense el efecto de la primera). Si la transacción no se puede finalizar por cualquier motivo, el DBMS garantiza que los cambios realizados por esta transacción son deshechos. En el ejemplo de la transferencia de fondos entre dos cuentas bancarias, si el dinero se extrae de una cuenta y la transacción falla antes de que el dinero se ingrese en la otra cuenta, el DBMS deshará la extracción de fondos.

El objetivo del diseño de las transacciones es definir y documentar las características de alto nivel de las transacciones que requiere el sistema. Esta tarea se debe llevar a cabo al principio del proceso de diseño para garantizar que el esquema lógico es capaz de soportar todas las transacciones necesarias. Las características que se deben recoger de cada transacción son las siguientes:

- Datos que utiliza la transacción
- Características funcionales de la transacción
- Salida de la transacción
- Importancia para los usuarios
- Frecuencia de utilización.

Hay tres tipos de transacciones:

- En las *transacciones de recuperación* se accede a los datos para visualizarlos en la pantalla a modo de informe
- En las *transacciones de actualización* se insertan, borran o actualizan datos de la base de datos
- En las *transacciones mixtas* se mezclan operaciones de recuperación de datos y de actualización.

El diseño de las transacciones utiliza la información dada en las especificaciones de requisitos de usuario

3.4.6 Diseño de interfaces de usuario

Antes de programar las funciones de las pantallas de captura y de los informes, hay que diseñar su aspecto. Es conveniente tener en cuenta las siguientes recomendaciones:

- Utilizar títulos que sean significativos, que identifiquen sin ambigüedad el propósito del informe o pantalla de captura
- Dar instrucciones breves y fáciles de comprender
- Agrupar y secuenciar los campos de forma lógica
- Hacer que el aspecto del informe o las pantallas sea atractivo a la vista
- Utilizar nombres familiares para etiquetar los campos
- Utilizar terminología y abreviaturas consistentes
- Hacer un uso razonable y consistente de los colores
- Dejar un espacio visible para los datos de entrada y delimitarlos
- Permitir un uso sencillo y adecuado del cursor
- Permitir la corrección carácter a carácter y de campos completos
- Dar mensajes de error para los valores “ilegales'”
- Marcar los campos que sean opcionales
- Dar mensajes a nivel de campo para explicar su significado
- Dar una señal que indique cuándo la información está completa.

CAPÍTULO 4 DISEÑO

En este capítulo se presenta el diseño de la arquitectura del sistema, que incluye la interfaz gráfica de usuario, el acceso a una forma de almacenamiento persistente (base de datos) y el uso de ciertos archivos de los cuales hace uso la aplicación, para facilitar la construcción y modificación de los reportes que emite.

Para ilustrar el diseño la Fig. 4.1 indica las interacciones del sistema con entidades externas. Como se puede observar estas son tres: base de datos, Internet y los archivos propios de la aplicación.

La parte más importante del almacenamiento se centra en la base de datos la cual contendrá toda aquella información recopilada de las reuniones de trabajo, ésta es

ingresada por el usuario a través de la interfase gráfica de usuario la cual muestra una serie componentes gráficos de captura comunes en un ambiente Windows. El sistema tendrá la capacidad de transmitir y recopilar información por medio de los protocolos SMTP y POP3 respectivamente haciendo uso de Internet o Intranet. Finalmente el sistema dispondrá de un conjunto de plantillas construidas sobre documentos en Microsoft Word, que servirán para mostrar la información en reportes de cada una de las etapas por las cuales atraviesa una reunión de trabajo.

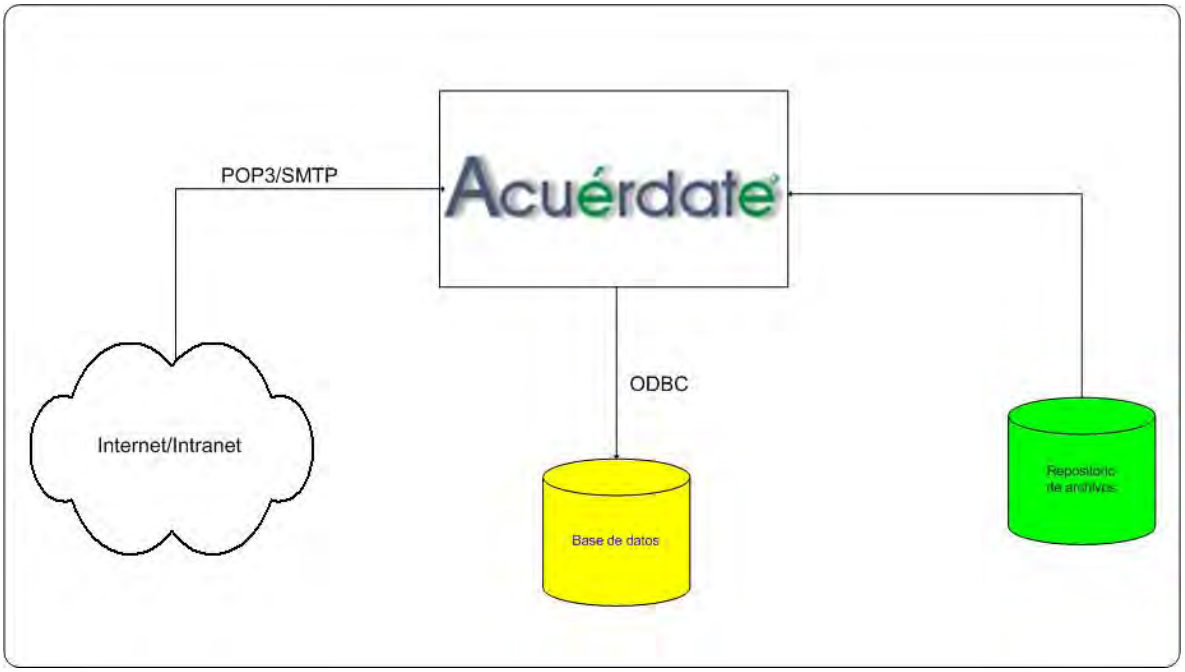


Figura 4.1 Diagrama de interacción del sistema con entidades externas

4.1 Diseño de la base de datos

Uno de los retos en el diseño de la base de datos es el de obtener una estructura estable y lógica tal que:

- El sistema de base de datos no sufra de anomalías de almacenamiento.
- El modelo lógico pueda modificarse fácilmente para admitir nuevos requerimientos.

Una base de datos implantada sobre un modelo bien diseñado tiene mayor esperanza de vida aun en un ambiente dinámico, que una base de datos con un diseño pobre. En promedio, una base de datos experimenta una reorganización general cada seis años, dependiendo de lo dinámico de los requerimientos de los usuarios. Una base de datos bien diseñada tendrá un buen desempeño aunque aumente su tamaño, y será lo suficientemente flexible para incorporar nuevos requerimientos o características adicionales.

4.1.1 Aplicación de las formas normales

Existen diversos riesgos en el diseño de las bases de datos relacionales que afecten la funcionalidad de la misma, los riesgos generalmente son la redundancia de información y la inconsistencia de datos.

Como se citó en el capitulo de análisis, la normalización es el proceso de simplificar la relación entre los campos de un registro, así un conjunto de datos en un registro se reemplaza por varios registros que son más simples y predecibles y, por lo tanto, más manejables. Se lleva a cabo por cuatro razones:

- Estructurar los datos de forma que se puedan representar las relaciones pertinentes entre los datos.
- Permitir la recuperación sencilla de los datos en respuesta a las solicitudes de consultas y reportes.
- Simplificar el mantenimiento de los datos actualizándolos, insertándolos y borrándolos.
- Reducir la necesidad de reestructurar o reorganizar los datos cuando surjan nuevas aplicaciones.

La teoría de normalización tiene como fundamento el concepto de formas normales que son las técnicas para prevenir las anomalías en las tablas. Es importante mencionar que una base de datos normalizada hasta la **3FN,** significa que debe estar normalizada en **2FN** y **1FN**; la fig. 4.2 muestra la relación entre las formas normales.

Figura 4.2 Relación entre las formas normales

Todas las formas normales fueron descritas en el capitulo de análisis, sin embargo, es necesario recordar lo más importante de cada una de ellas.

a) Primera forma normal (1FN)

Se considera que una relación se encuentra en la primera forma normal cuando cumple lo siguiente:

1. Las celdas de las tablas posean valores simples y no se permiten grupos ni arreglos repetidos como valores, es decir, contienen un solo valor por cada celda.
2. Todos los ingresos en cualquier columna (atributo) deben ser del mismo tipo.
3. Cada columna debe tener un nombre único, el orden de las columnas en la tabla no es importante.
4. Dos filas o renglones de una misma tabla no deben ser idénticas, aunque el orden de las filas no es importante.
5. Las columnas repetidas deben eliminarse y colocarse en tablas separadas.

El evento de asistir a una reunión implica que debe haberse establecido la hora, lugar y fecha; así como las personas que asistirán; la Fig. 4.3 muestra la relación entre la tabla **persona** y **reunión** que ejemplifica lo citado anteriormente.

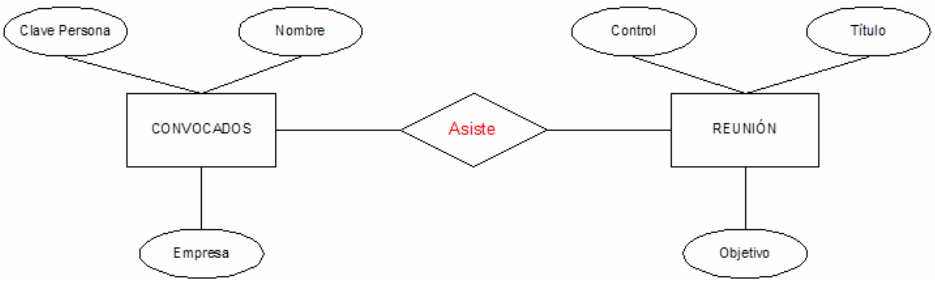


Figura 4.3 Representación gráfica de 1FN.

Tablas de catálogo: Son todas aquellas tablas en donde se almacenan los diferentes datos que puede tener una entidad.

Tablas de sistema: Son aquellas entidades que relacionan a tablas catálogo, principalmente son las tablas centrales en la base de datos.

El primer paso para el diseño de la base de datos fue identificar las entidades requeridas para la recolección y procesamiento de lo datos; para esta definición fue necesario analizar la problemática a resolver. Con este análisis las tablas definidas son:

- **Reunión:** Esta es la tabla principal de la base de datos, porque en ella se almacena los datos generales de una reunión por ejemplo, el título, objetivo, hora inicial y final, duración, convocante, su clasificación y otros.

Reunion

Reunion: Text(14)
TituloReunion: Text(200) NombreCompleto: Text(50) NOT NULL Objetivo: Text(200) Caracter: Text(20) NOT NULL ResultadosDeseados: Text(200) FReunion: Date/Time NOT NULL HoraInicio: Date/Time NOT NULL DiaReunion: Text(18) HoraFinal: Date/Time NOT NULL TiempoEstimado: Double NOT NULL HoraInicioReal: Date/Time HoraFinalReal: Date/Time FReal: Date/Time TiempoReal: Double Observacion: Memo FAlta: Date/Time NOT NULL Lugar: Text(150) NOT NULL Status: Text(22) NOT NULL FStatus: Date/Time NOT NULL Licencia: Text(30) Serie: Text(10) Exportar: Yes/No NOT NULL Reenviar: Yes/No Enviado: Yes/No ClaveAgenda: Integer FEnvio: Date/Time ReunionProxima: Text(14)

- **OrdenDia:** En ella se almacenan los puntos y subpuntos a tratar en una reunión.

OrdenDia

Punto: Long Integer NOT NULL
Descripcion: Text(120) NOT NULL ReunionOrigen: Text(14) Subpunto: Integer PuntoOrigen: Long Integer FechaAlta: Date/Time NOT NULL Status: Text(15) NOT NULL FechaStatus: Date/Time NOT NULL Secuencia: Single Expandido: Yes/No NOT NULL

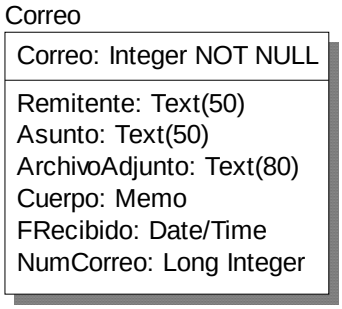
- **Minuta:** Tabla para almacenar todos los acuerdos generados de una reunión.

Minuta

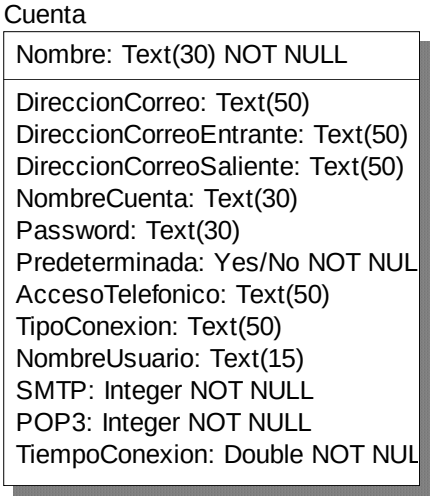
Acuerdo: Integer NOT NULL
Status: Text(18) NOT NULL Secuencia: Single NOT NULL FStatus: Date/Time NOT NULL Descripcion: Memo Prioridad: Text(18) NOT NULL ReunionOrigen: Text(14) PuntoOrigen: Long Integer AcuerdoOrigen: Integer ReunionDestino: Text(14) PuntoDestino: Long Integer AcuerdoDestino: Integer Transferir: Yes/No NOT NULL FInicialAcuerdo: Date/Time

- **ParametrosGenerales:** Es una tabla de parámetros generales, que permite guardar valores que serán de utilidad a lo largo de toda la aplicación.

- **Correo:** Se utilizará para almacenar todos los correos recibidos de otros usuarios enviados por medio del sistema.



- **Cuenta**: Tabla para almacenar todas las cuentas de correo que utilice el propietario.



- **Empresa:** Todos los datos referentes a las empresas de donde provienen nuestros asistentes, se registraran en esta tabla.

Empresa

Empresa: Integer NOT NULL
Descripcion: Text(120) NOT NULL Direccion: Text(120) Persona: Text(12) Ciudad: Text(50) EstadoProvincia: Text(50) CodigoPostal: Text(10) Pais: Text(25) Telefono1: Text(18) Telefono2: Text(18) Fax: Text(18) DireccionPagina: Text(100) CorreoElectronico: Text(100) FAlta: Date/Time NOT NULL FModificacion: Date/Time RutaImagen: Text(150) Status: Text(18) NOT NULL FStatus: Date/Time

- **Persona:** En esta tabla almacenamos los datos de los contactos.

Persona

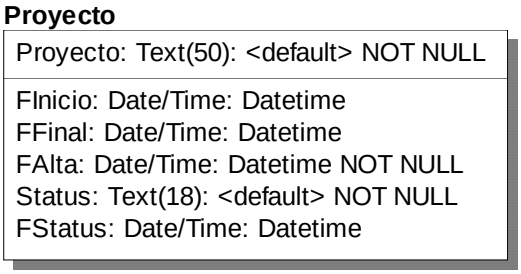
Persona: Text(12) NOT NULL
Nombre: Text(30) NOT NULL (IE1.2,IE2.4) ApellidoPaterno: Text(30) NOT NULL (IE1.1,IE2.2) ApellidoMaterno: Text(30) (IE2.3) Titulo: Text(5) Sobrenombre: Text(18) NombreMostrar: Text(40) Departamento: Text(60) Puesto: Text(60) Direccion: Text(120) Ciudad: Text(50) EstadoProvincia: Text(50) CodigoPostal: Text(10) Pais: Text(25) Telefono1: Text(18) Telefono2: Text(18) Fax: Text(18) Movil: Text(18) Radiolocalizador: Text(18) DireccionPagina: Text(100) CorreoElectronico1: Text(100) CorreoElectronico2: Text(100) FAlta: Date/Time NOT NULL FModificacion: Date/Time Status: Text(15) NOT NULL FStatus: Date/Time NOT NULL UserID: Text(10) Password: Text(10) Licencia: Text(8) NOT NULL Iniciales: Text(3) Sonido: Text(10) Sincronizado: Yes/No NOT NULL Enviar: Yes/No NOT NULL

- **MotivoCancelacion:** Este es un catálogo de motivos de cancelación de acuerdos, la definición esta a cargo del propietario.

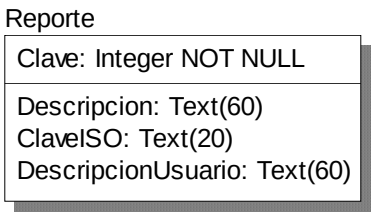
MotivoCancelacion

MotivoCancelacion: Integer NOT NUL
Descripcion: Text(100)

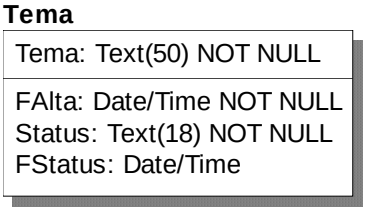
- **Proyecto:** Los acuerdos generados de una reunión se pueden clasificar por proyecto, en esta tabla almacenamos todos los datos de un proyecto.



- **Reporte:** Esta tabla es un catálogo de reportes en la que almacenamos el título y clave ISO para cada uno de los reportes que genera el sistema, la finalidad de utilizar esta tabla es la de personalizar la presentación final del documento a generar.



- **Tema:** Los acuerdos generados en una reunión se pueden clasificar con base a un tema en especifico, en esta tabla se guardan todos los datos de un tema.



- **TipoReunion:** Todas las reuniones debemos clasificarlos de acuerdo al área, departamento, proyecto, empresa, etc.

TipoReunion

TipoReunion: Long Integer NOT NULL
Descripcion: Text(60) NOT NULL FAlta: Date/Time NOT NULL TipoReunionOrigen: Long Integer Expandido: Yes/No NOT NULL Tipo: Text(18) NOT NULL

Las tablas descritas anteriormente se encuentran en 1FN, el paso siguiente es normalizarlas en segunda forma normal.

b) Segunda forma normal (2FN)

Para definir formalmente la segunda forma normal requerimos saber que es una **dependencia funcional:** Consiste en edificar que atributos dependen de otro(s) atributo(s). *Ver fig. 4.4*

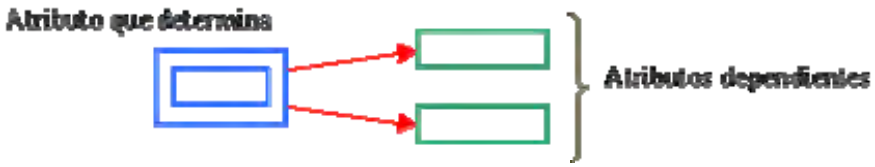


Fig. 4.4 Representación gráfica 2FN.

Definición formal: Una relación R está en 2FN si y solo si está en 1FN y los atributos dependen funcionalmente de la llave primaria.

Una relación se encuentra en segunda forma normal, cuando cumple con las reglas de la primera forma normal y todos sus atributos que no son llaves, dependen por completo de está. De acuerdo con esta definición, cada tabla que tiene un atributo único como llave primaria, está en segunda forma normal.

La segunda forma normal se representa por dependencias funcionales como lo muestra la fig. 4.5, la tabla **Persona** almacena a todos los contactos, cada uno de ellos tiene una clave única que los identifica.

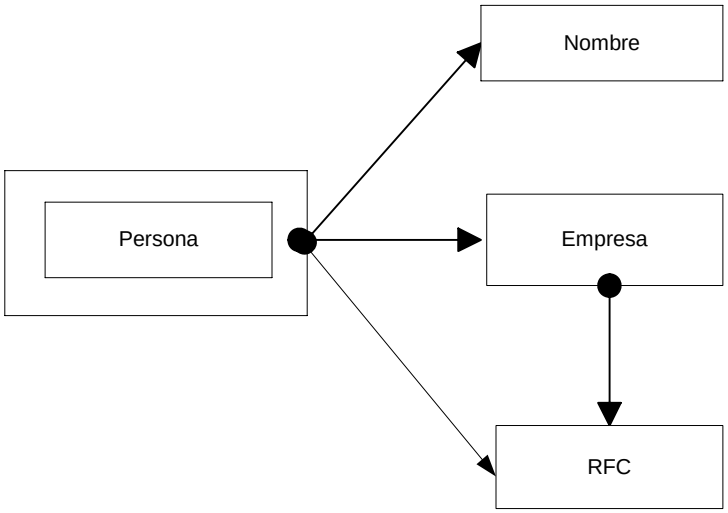


Fig. 4.5 Dependencias funcionales

Nótese que las llaves primarias están representadas con doble cuadro, las flechas nos indican que de estos atributos se puede referenciar a los otros atributos que son funcionalmente la llave primaria.

c) Tercera forma normal (3FN)

Para definir formalmente la 3FN necesitamos definir **dependencia transitiva:** En una afinidad (tabla bidimensional) que tiene por lo menos 3 atributos (A,B,C) en donde A determina a B**,** B determina a C pero no determina a A.

Dependencia formal: Una relación R está en 3FN si y sólo si está en 2FN y todos sus atributos no primos dependen no transitivamente de la llave primaria.

Consiste en eliminar la dependencia transitiva que queda en una segunda forma normal, en pocas palabras una relación está en tercera forma normal si está en segunda forma normal y no existen dependencias transitivas entre los atributos; nos referimos a dependencias transitivas cuando existe más de una forma de llegar a referencias a un atributo de una relación.

Por ejemplo, consideremos el siguiente caso:

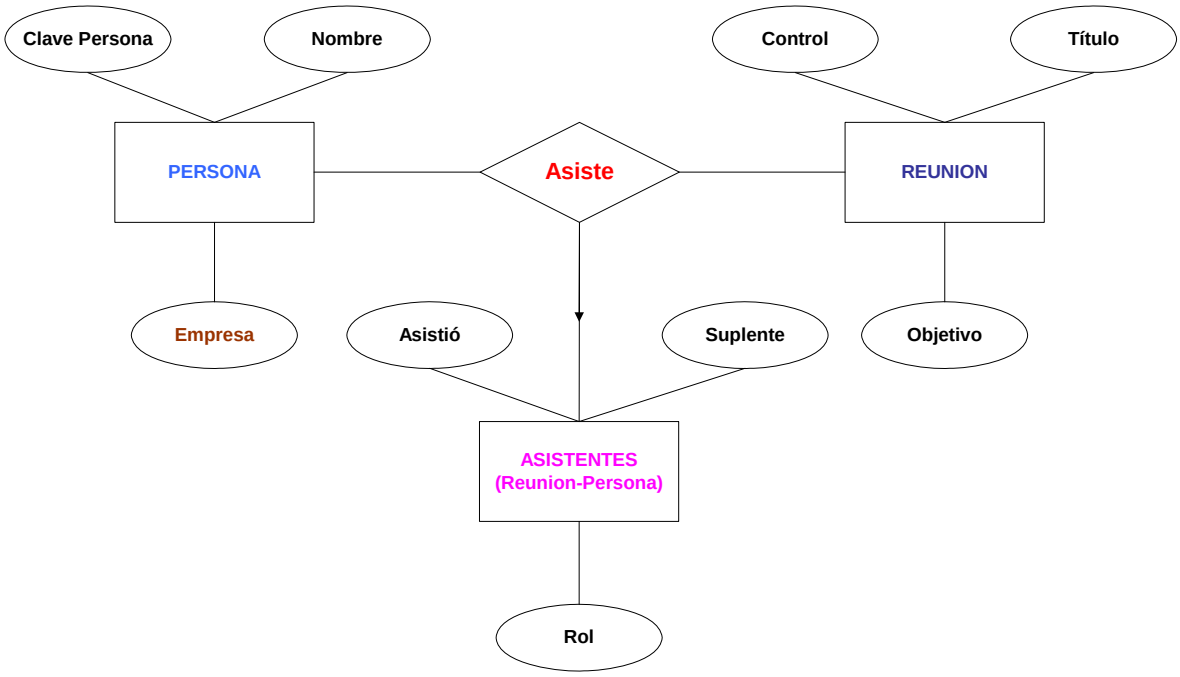


Fig. 4.6 Diagrama persona-asiste-reunión

La Fig. 4.6 muestra la relación persona-asiste-reunión, pero en especial consideremos al elemento persona, gráficamente la podemos representar de la siguiente manera.

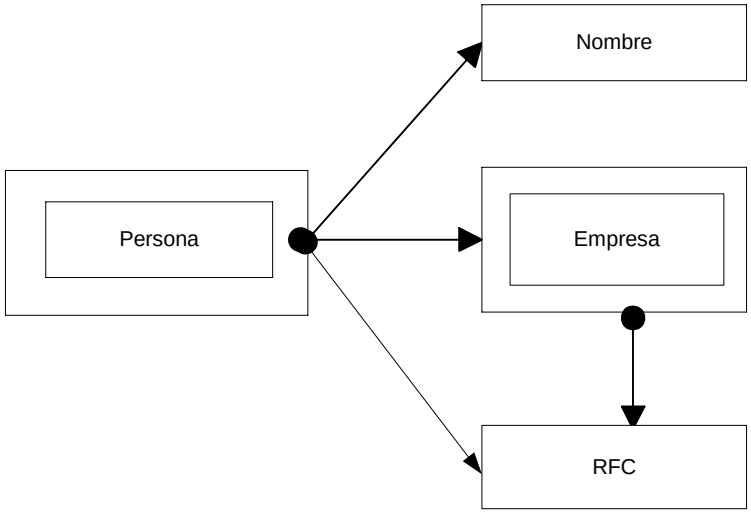


Fig. 4.7 Relación en segunda forma normal

La fig.4.7 muestra a la relación en segunda forma normal, los atributos llave están indicados en doble cuadro indicando los atributos que dependen de dichas llaves, sin

embargo, en la llave **Persona** tiene como dependientes a 3 atributos en el cual el RFC puede ser referenciado por dos atributos: **Persona y Empresa,** (existe dependencia transitiva), esto se soluciona al aplicar la tercer forma normal que consiste en eliminar estas dependencias separando los atributos, por lo tanto tenemos:

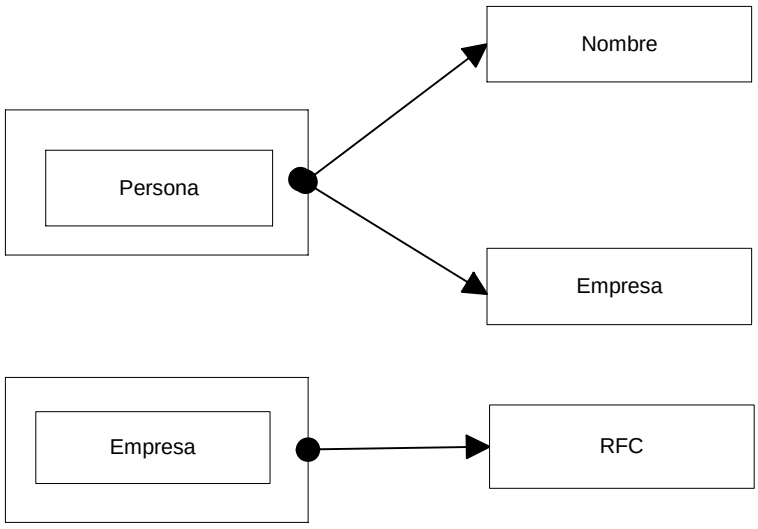
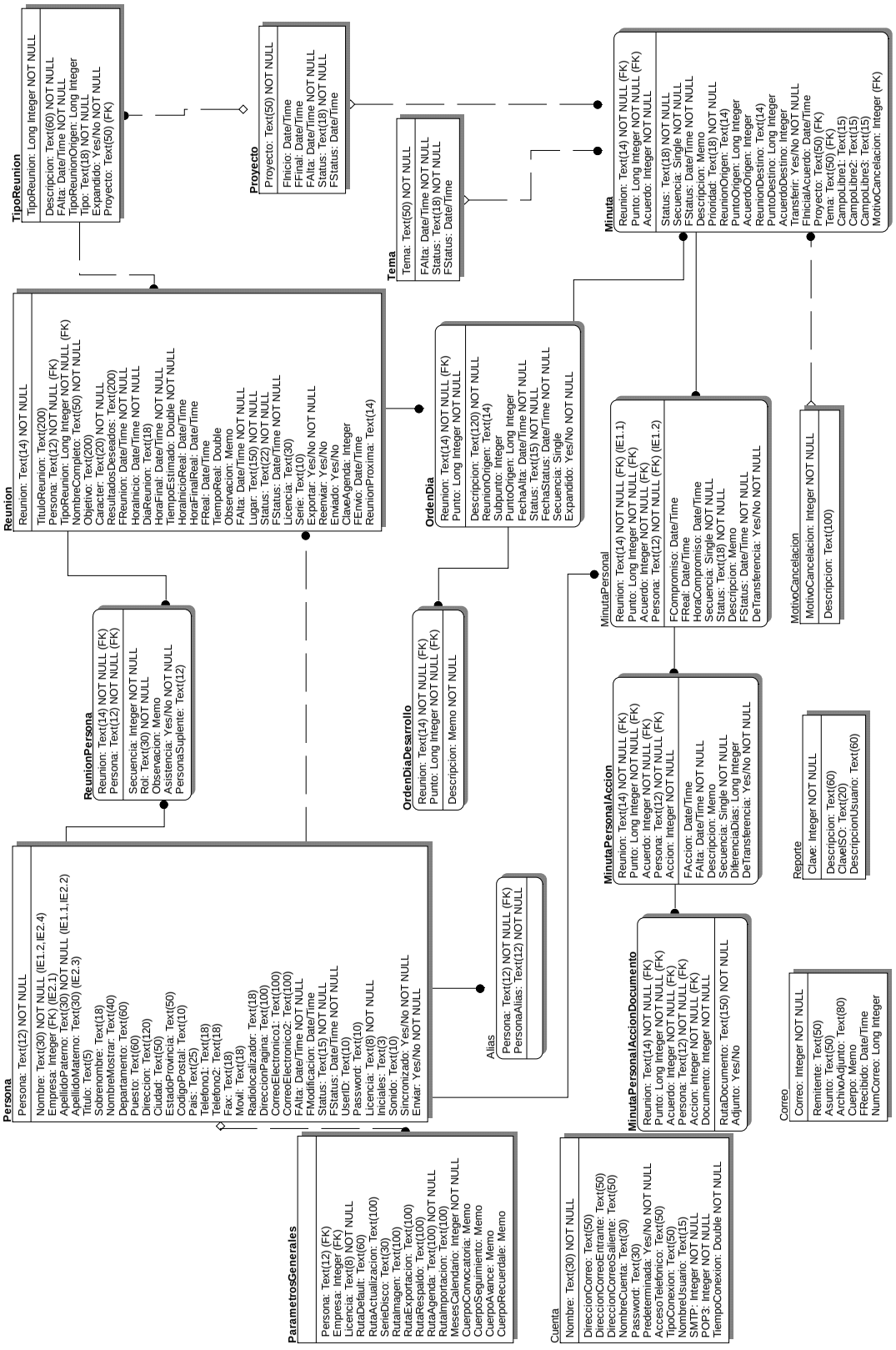


Fig. 4.8 Diagramas en tercera forma normal

La Fig.4.8 muestra la eliminación de las dependencias, finalmente la dependencia transitiva se rompió al separar los atributos que dependen directamente de la llave primaria, es decir, en este diagrama en tercera forma normal tenemos a dos dependencias por separado, por un lado tenemos al elemento **Persona** y sus atributos **Nombre y Empresa**; y por otro **Empresa** con el atributo **RFC.**

Una vez definidas las tablas en **1FN**, se aplico la **2FN** en donde aparecieron dependencias transitivas como en el ejemplo anterior, por lo que fue necesario aplicar **3FN,** es así como el diseño final de la base de datos quedo de la siguiente manera:

El modelo entidad-relación



4.2 Lenguaje de programación

Para el diseño de la interfase gráfica de usuario se decidió por hacer uso del ambiente de programación Delphi en su versión 5. Delphi es una potente herramienta de desarrollo de programas que permite la creación de aplicaciones. Borland Delphi es considerado un ambiente de Desarrollo Rápido de Aplicaciones (RAD: Rapid Application Development), esto significa que ciertas tareas, como el diseño de las interfaces de usuario o el modelo de acceso a datos, se efectúan de manera visual, arrastrando y soltando elementos conocidos como componentes. Las herramientas RAD disponen de componentes prefabricados, que efectúan una determinada labor, y asistentes que generan código automáticamente. Delphi es un ambiente RAD para el lenguaje Object Pascal en el sistema operativo Windows 95/98/NT/2000/XP e inclusive para el sistema operativo Linux, siendo Kylix la versión de Delphi para éste sistema. Los sistemas RAD están orientados a facilitar la productividad en el desarrollo de software. Además del beneficio de la productividad, un sistema RAD debe también promover la excelencia en la calidad del software desarrollado, facilitando el uso eficiente de hardware y software.

Algunas características que hacen de Delphi un sistema de alta productividad y excelencia son:

- Ambiente visual de desarrollo (similar al de Visual Basic de Microsoft) para aplicaciones controladas por intervenciones o eventos de usuario sobre interfaces gráficas.
- Lenguaje de programación de excelencia en estilo y expresividad. Object Pascal de Delphi es un Pascal orientado a objetos (OO), completamente comprometido con el modelo de objetos, comparable al de Java. Es heredero del original en Macintosh al igual que Turbo Pascal. Además, la implementación por Borland es eficiente en compilación y en ejecución.

- Proporciona una jerarquía muy extensa de Clases de Objetos reusables y extensibles, con recursos de computación visuales y de procesamiento.

- Permite desarrollar rápidamente aplicaciones soportadas por Bases de Datos, mediante la inclusión de Clases para acceso a diversos sistemas de BD, con una consulta SQL, visualización y navegación en tablas originales o resultantes de

consulta, incorporación de datos para procesamiento dentro de la aplicación y posteriormente actualización a la BD.

- En las versiones Professional y Enterprise, Delphi ofrece clases para el desarrollo de aplicaciones en Internet soportadas por TCP/IP, para aplicaciones distribuidas. También con soporte a los modelos de distribución de objetos COM, DCOM Y CORBA.

- Abundante documentación y ayuda. Delphi proporciona además una variedad de ejemplos en su directorio demos los cuales ilustran muchas de las clases ofrecidas por la versión.

Definitivamente Delphi tiene muchas ventajas técnicas sobre otros lenguajes de programación para el desarrollo de aplicaciones, estas ventajas redundan en costos de desarrollo y mantenimiento más bajos, así como, su soporte hacia la programación orientada a objetos que las consideramos de gran importancia, a diferencia de Visual Basic 6 el cual se anunció como un lenguaje orientado a objetos, la realidad es que solo soporta la encapsulación (y a medias) y no soporta la herencia, el polimorfismo, la creación de interfaces ni la sobrecarga.

Delphi es una Two way Tool, es decir, una herramienta de dos direcciones, porque permite crear el desarrollo de programas de dos formas: una de forma visual en la pantalla, por medio de las funciones de arrastrar y colocar componentes y la otra a través de la programación convencional, escribiendo el código. Ambas técnicas pueden utilizarse de forma alternativa o simultánea.

También mediante la programación orientada a objetos es posible crear nuevos programas a partir de estos. El polimorfismo permite que se mezcle correctamente el código desde un programa ya existente con el de otro nuevo lográndose un comportamiento distinto en cada situación. Por otro lado las interfaces son una pieza clave en la computación distribuida e irónicamente son requeridas para programar objetos COM y DCOM entre otros.

La programación orientada a objetos llega a su máximo nivel de reutilización mediante la creación de componentes y controles, programarlos en Delphi es muy sencillo y claro. El código se escribe en el mismo Object Pascal y con las mismas reglas que se utilizan en Delphi para cualquier programa. Una vez que se tiene un componente en Delphi éste se encadena normalmente con el ejecutable, sin embargo, también es posible convertirlos a ActiveX para ser usados mediante archivos OCX en cualquier herramienta que use esta tecnología, como Visual Basic. Debido a esto existen mucho más componentes de terceros de Delphi que de Visual Basic. En Visual Basic es raro comprar un componente que incluya su código fuente ya que a los mismos programadores de Visual Basic no les interesa. En Delphi es raro no comprar un componente con código fuente.

Creemos que como estudiantes y profesionistas debemos de luchar por tener lo más claro posibles los conceptos y sobre todo tener lograr bases firmes especialmente en aspectos universalmente aceptados como lo es la orientación a objetos.

CAPÍTULO 5 FUNCIONALIDADES DEL SISTEMA

Este capítulo tiene como objetivo describir la estructura y las principales funcionalidades con las que cuenta el sistema. Acuérdate es un software completo de gestión y control de reuniones de trabajo, su interfase lógica e intuitiva nos permite dar seguimiento y una mejor administración a los acuerdos derivados de estas, así como el mantener notificado al equipo de trabajo de las acciones realizadas por los demás miembros, acerca de estos acuerdos mediante su mecanismo de actualización vía correo electrónico.

La idea principal en el diseño de la aplicación es poder ofrecer al usuario una herramienta fácil de utilizar, de forma tal que al encontrarse registrando la información de una reunión en cualquiera de sus etapas, no pierda detalle de todos aquellos aspectos relevantes que conlleva como lo son principalmente los acuerdos y los responsables de los mismos.

5.1 Estructura de la aplicación

Al iniciar una sesión en el sistema Acuérdate se desplegará la pantalla principal la cual se muestra en la figura 5.1

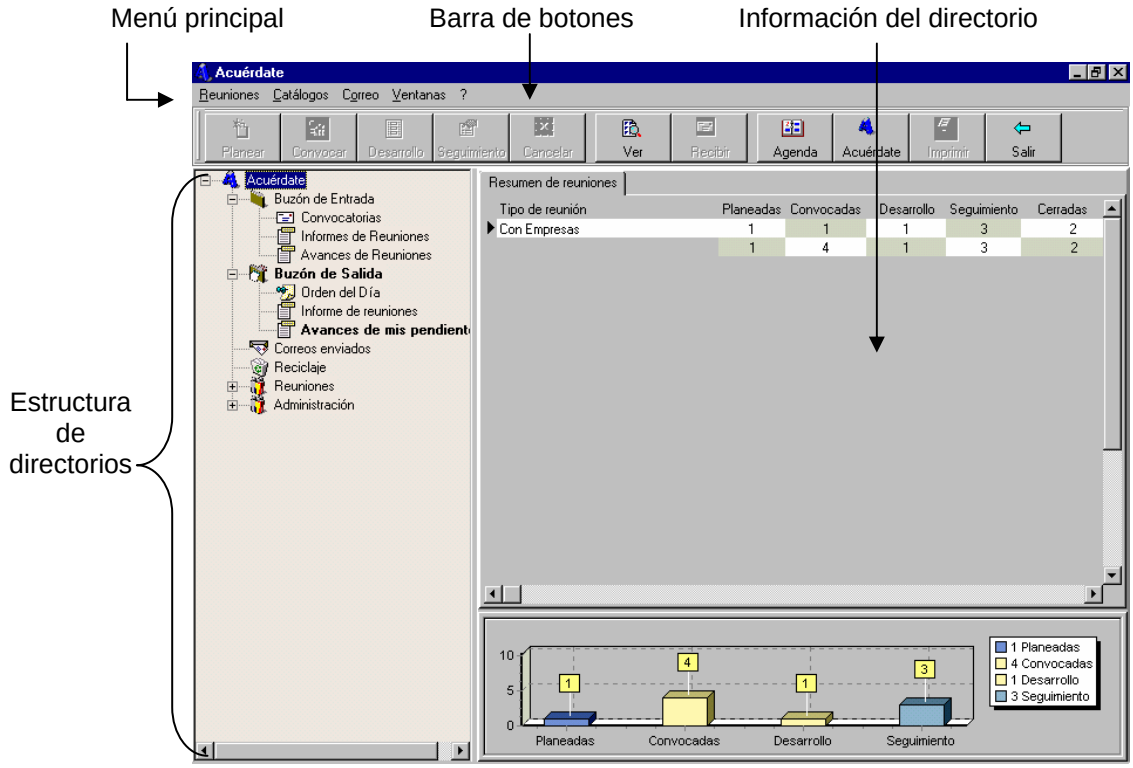


Figura 5.1 Pantalla principal del sistema

En esta pantalla podemos identificar cuatro elementos importantes para el manejo de información, estos son:

Menú principal. Este menú nos permite acceder a las diferentes funcionalidades del sistema, como lo son las pantallas para el manejo de la información de la reunión en cada etapa, así como los catálogos, y la configuración de las cuentas de correo electrónico.

Estructura de directorios. La información es organizada mediante el uso de una estructura de árbol que nos permite conjuntar información de manera tipificada, como por

ejemplo, por tipo de proyectos, por actividades personales, actividades grupales o cualquier otro que se desee definir.

Barra de botones. La barra de botones nos permite acceder a funcionalidades específicas de acuerdo a directorio en donde nos encontremos.

Panel de información y contenido de un directorio. Este panel nos muestra la información contenida en el directorio donde estemos ubicados en la estructura del árbol.

5.1.1 Menú principal

El primer elemento que encontramos dentro del menú principal, es el de **Reuniones**, el cual nos ofrece las herramientas necesarias para manejar la información de una reunión en sus diferentes etapas, como se muestra en la figura 5.2.

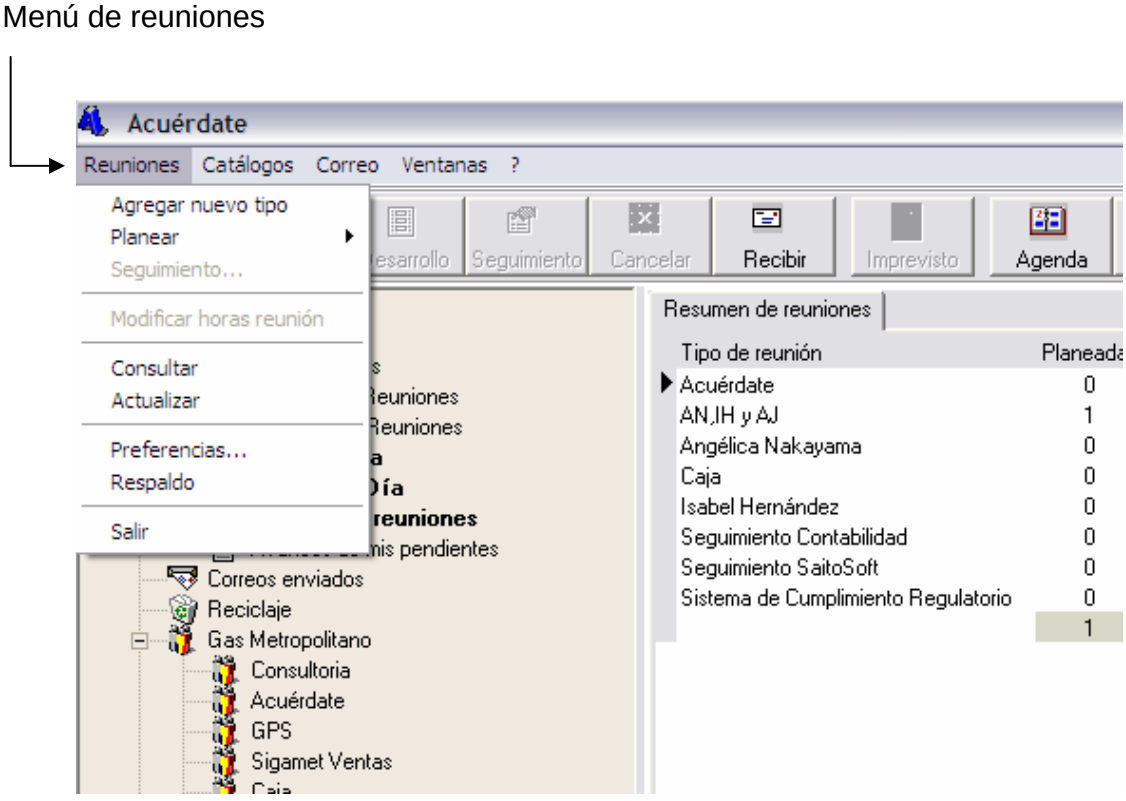


Figura 5.2 Menú de reuniones.

Dentro de las opciones se encuentran en el menú de Reuniones tenemos las siguientes:

Agregar nuevo tipo. Esta función nos permite incorporar un nuevo grupo de reuniones dentro del árbol de directorios.

Planear. Esta opción contiene un submenú con las opciones de Agregar, Eliminar y modificar las cuales permiten respectivamente planear una nueva reunión, eliminar y modificar un registro de una reunión con estatus de planeada.

Seguimiento. Muestra la ventana con las opciones correspondientes para dar seguimiento a reuniones de trabajo que se encuentren en dicha etapa.

Modificar horas reunión. Funcionalidad que permite cambiar la hora de inicio y la duración de una reunión.

Consultar. Permite establecer filtros de las reuniones que deseamos que se muestren cuando nos encontramos en un directorio en específico del árbol.

Actualizar. Refresca la información en la base de datos y pantallas del sistema.

Preferencias. Muestra la ventana de configuración de preferencias personales, la cual permite establecer colores de fondo de ventanas y componentes, así como definir rutas de directorios utilizados por el sistema.

Respaldo. Funcionalidad del sistema la cual permite generar una copia de seguridad de la base de datos.

El siguiente elemento dentro del menú principal son los **Catálogos** del sistema, donde se captura la información de contactos, empresas, temas de proyectos y los títulos para los reportes de información con los que cuenta el sistema con la finalidad de personalizarlos. El menú de catálogos se muestra en la siguiente figura:

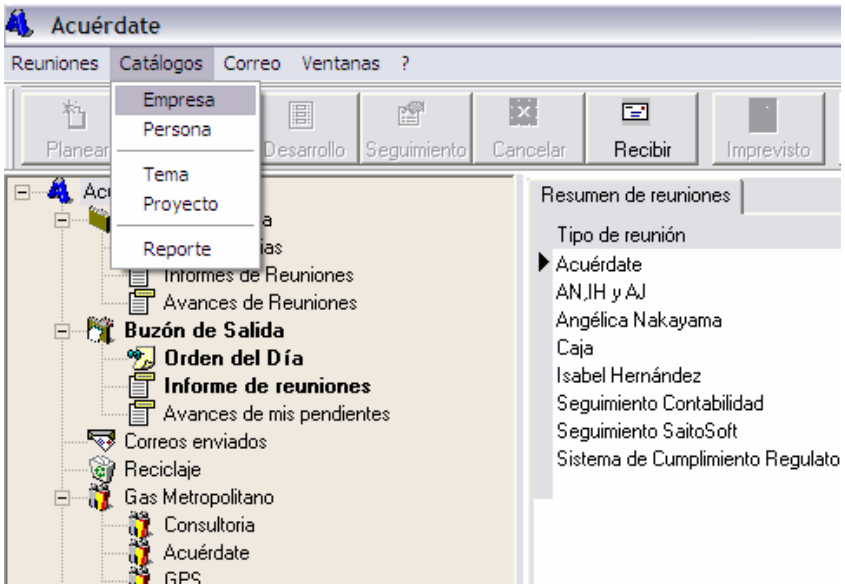


Figura 5.4 Menú de catálogos.

La siguiente opción dentro del menú principal es la de **Correo**, la cual cuenta con dos elementos, el primero es para descargar los correos electrónicos de una o más cuentas dadas de alta en el sistema, y el segundo es para configurar las cuentas de correo, este menú se muestra en la figura 5.5.

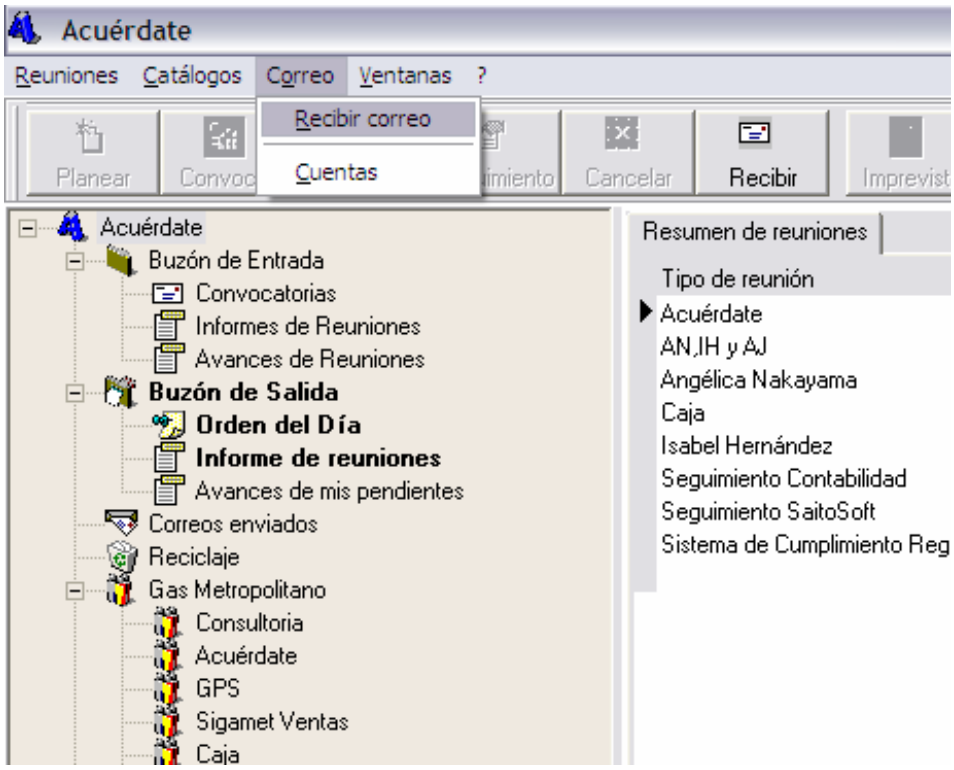


Figura 5.5 Menú de correo electrónico.

En la figura 5.6 se muestra la ventana de configuración de cuentas de correo electrónico, la cual aparece después de haber hecho clic en la opción Cuentas, ésta nos permite ingresar los parámetros e información de conexión para poder descargar correo electrónico enviado por otros sistemas Acuérdate. Entre la información que se registra se encuentra: el nombre de la cuenta de correo electrónico, nombre de usuario, contraseña, dirección de servidor de correo entrante POP3, dirección de servidor de correo saliente SMTP, así como la información para un servidor que requiere autenticación. Cabe mencionar que el sistema trabaja con los servidores de correo electrónico compatibles con los protocolos POP3 y SMPT, así como Lotus Notes y Microsoft Exchange.

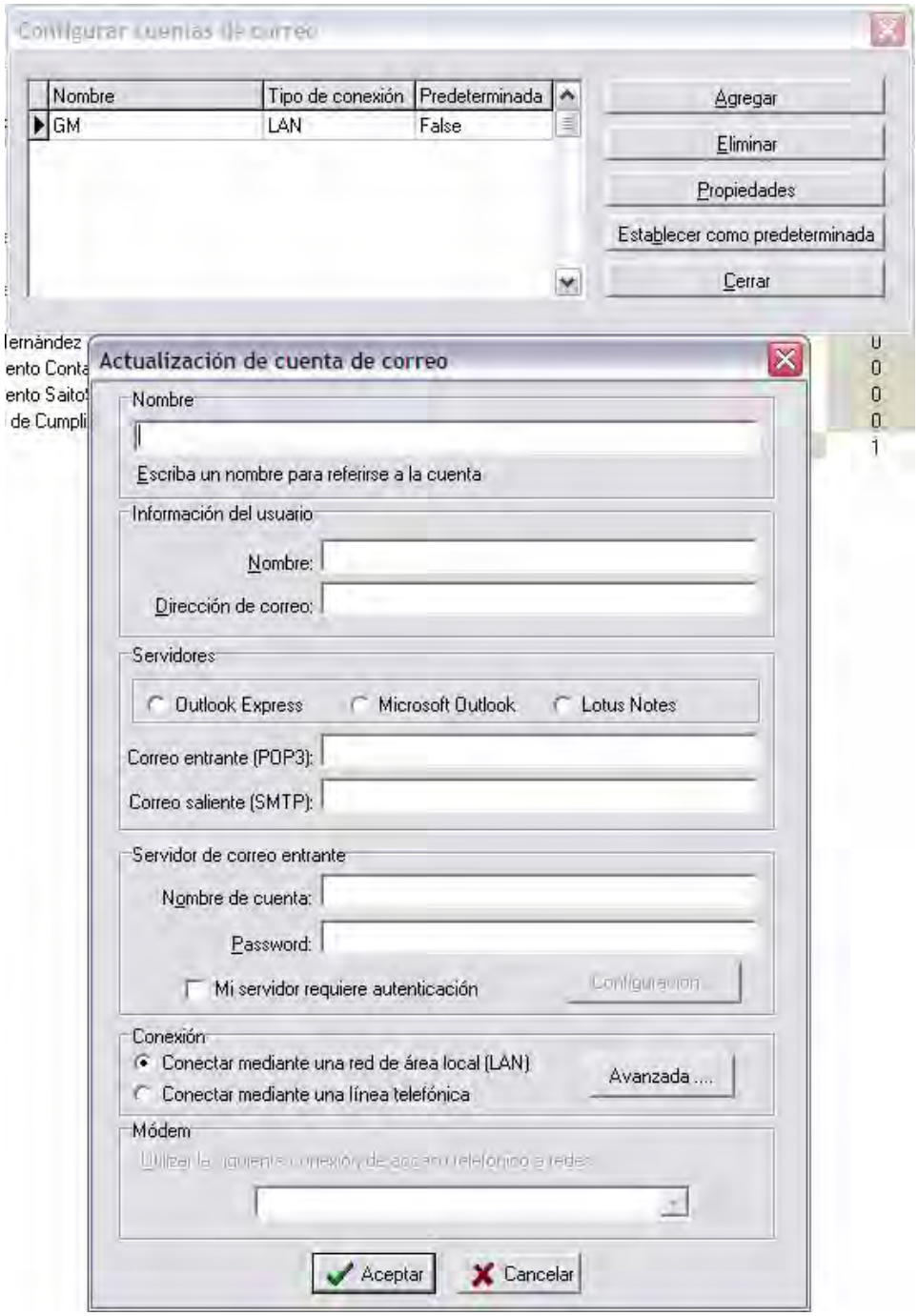


Figura 5.6 Configuración de cuentas de correcto electrónico.

Finalmente encontraremos dos opciones más dentro del menú principal, la primera es la opción de ventanas la cual tiene las funcionalidades de Cascada, Organizar y Mosaico y la opción “acerca de” representada por un signo de interrogación (?) la cual muestra información acerca de la aplicación como lo es su versión.

5.1.2 Barra de botones

De la misma manera la pantalla principal cuenta también con una barra de botones para acceder a la diferentes funcionalidades del sistema Acuérdate. A continuación se describe la funcionalidad de cada uno de los botones contenidos en esta barra.

Botón	Función
	Muestra la pantalla de captura de información para la planeación de una reunión.
	Envía la información de una reunión en estatus de planeada por correo electrónico a los convocados.
	Ingresa a la pantalla principal del desarrollo de una reunión.
	Botón que me lleva a la pantalla de seguimiento de la información de una reunión.
	Cancela una reunión.
	Descarga la información por correo electrónico que envía otro sistema acuérdate.
	Enviar por correo electrónico información de una reunión en cualquiera de sus etapas.

Botón	Función
	Despliega la agenda del sistema.
	Acceso a la ventana de visualización de acuerdos pendientes.
	Imprime los reportes del sistema como: La agenda del día, reporte de avances, minutas, etc.
	Cierra la aplicación.

5.1.3 Estructura de árbol

Ubicado en el panel izquierdo de la pantalla principal se encuentra la estructura de árbol, que representa las distintas carpetas en la que se almacena la información de reuniones de trabajo.

La estructura del árbol esta organizada de una forma predeterminada y en ella encontramos algunos nodos o elementos predefinidos como la muestra la figura 5.7.

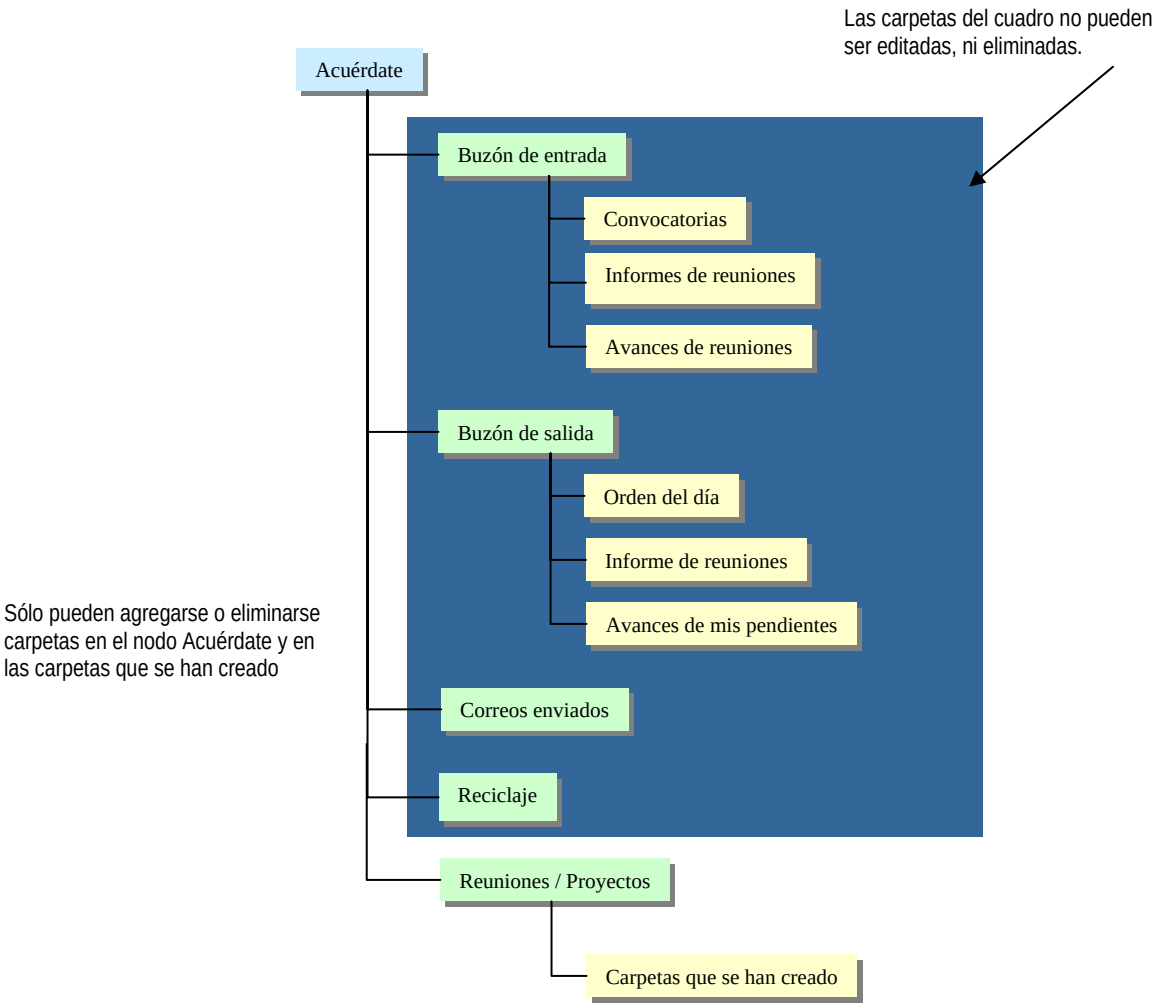


Figura 5.7 Estructura de árbol predefinida.

Por lo tanto los directorios mostrados son fijos y los nodos contenidos en ellos serán ubicados de acuerdo a la etapa o función en especifico que se este utilizando, por ejemplo una vez terminado el registro de la planeación de una reunión se colocará en el directorio de orden del día, y a su vez este directorio servirá como buzón de salida para que los elementos contenidos aquí, puedan ser enviados por correo electrónico a los convocados.

Otro ejemplo de esto es cuando recibimos por medio de correo electrónico una invitación a una reunión, automáticamente se guardará un elemento en el directorio de convocatorias que forma parte del buzón de entrada.

Cuando deseemos establecer nuestros propios directorios y necesitemos organizar nuestra información de una manera especial, estos directorios se crearan un nivel inferior

que el nodo principal o raíz Acuérdate. La forma para adjuntar directorios se describe a continuación.

1. Ubicados en el área del árbol seleccione con clic derecho el nodo principal Acuérdate, al hacer esto se desplegará un menú emergente para agregar un nuevo tipo de reunión como se muestra en la figura 5.8.

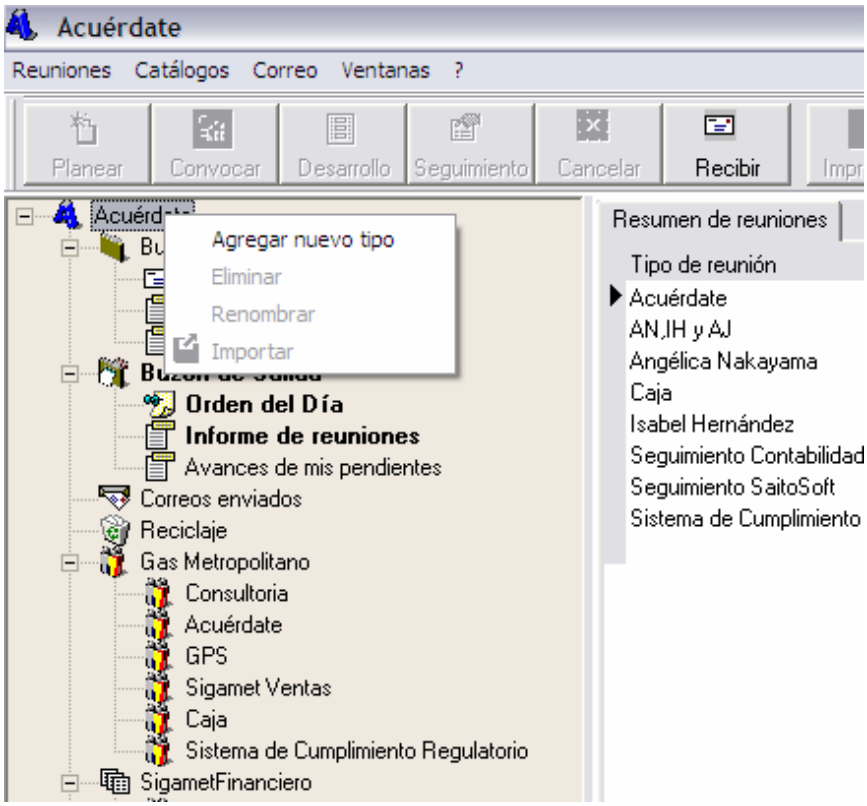


Figura 5.8 Menú emergente para agregar un directorio al árbol.

2. Posteriormente se mostrará la ventana de la figura 5.9 para especificar el tipo de directorio que se desea crear, en esta aparece siempre el tipo grupal por defecto.

Figura 5.9 Tipos de reuniones

5.1.4 Panel de información y contenido de un directorio

Este panel nos proporciona información detallada del directorio en donde nos encontremos ubicados dentro del árbol. El despliegue de la información puede variar dependiendo del tipo de nodo en donde nos encontremos, por ejemplo si nos encontramos en el nodo principal Acuérdate se mostrará un cuadro de estadísticas de todas las reuniones como aparece en la figura 5.10.

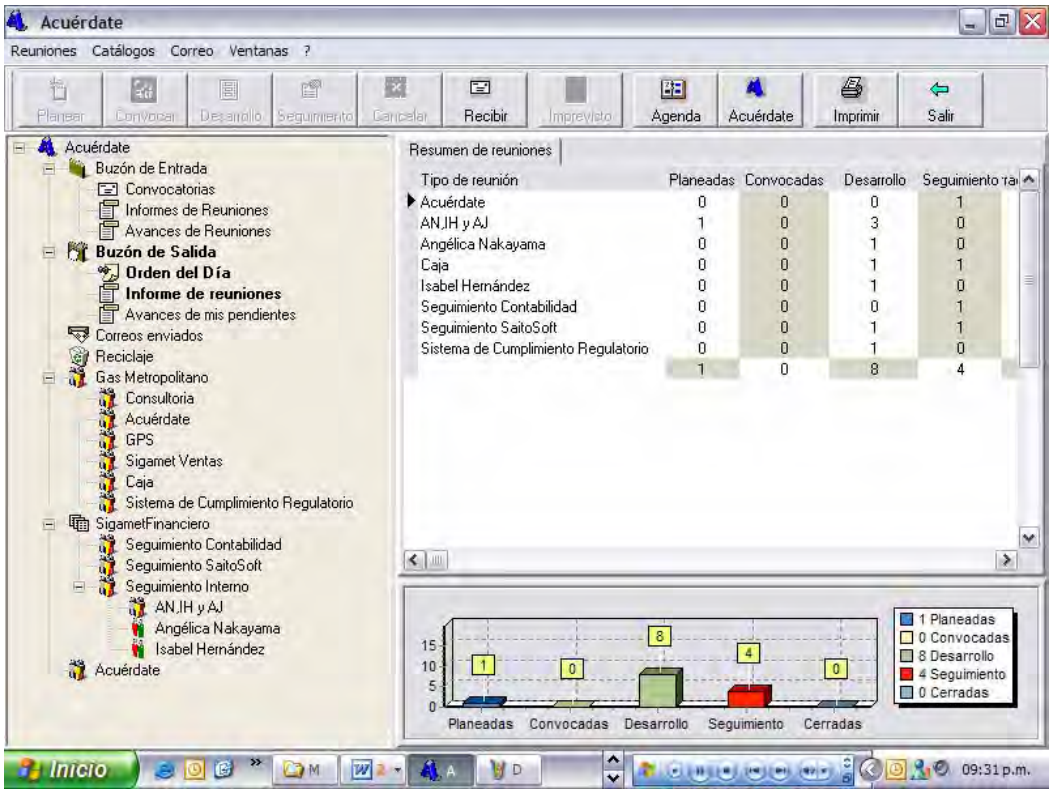


Figura 5.10 Estadísticas de las reuniones.

Al ubicarnos en cualquier otro directorio, el panel de información cambiará para mostrarnos la información relevante de cada una de las reuniones que tenemos ubicadas en dicho directorio como se muestra en la figura 5.11.

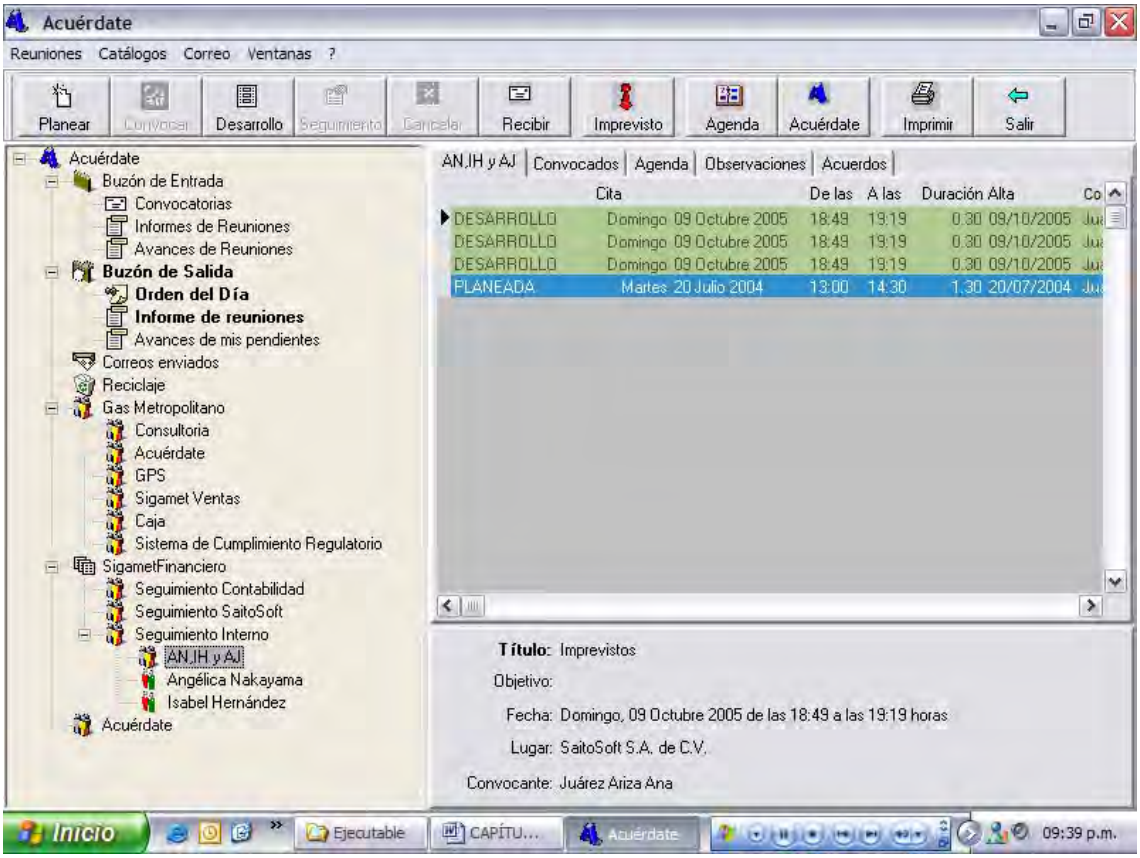


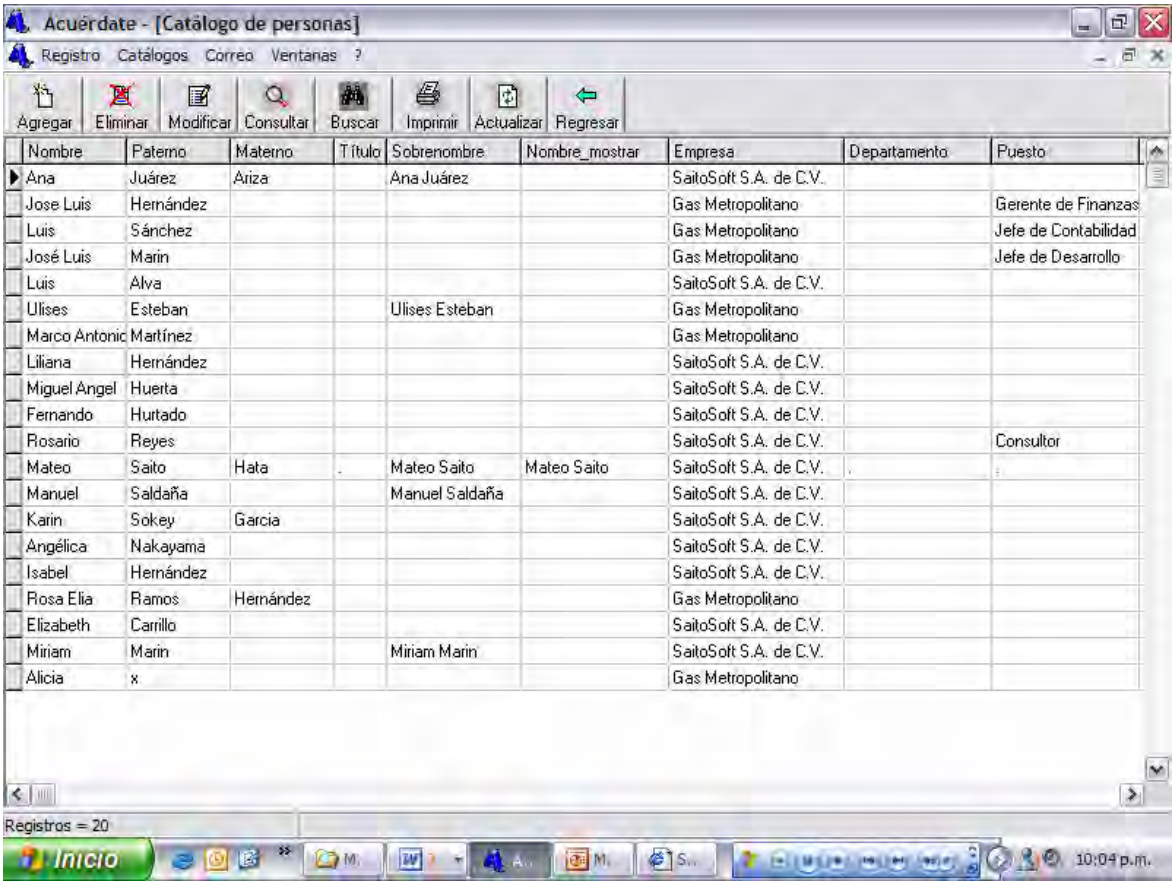
Figura 5.11 Panel de información para un directorio específico.

5.2 Catálogos del sistema

Los catálogos del sistema ofrecen la posibilidad de clasificar la información para facilitar la captura y mantenimiento de la misma.

Las funcionalidades principales de un catálogo son dar altas, bajas y modificaciones de registros de información que contenidos en él. De esta manera un cambio realizado en el catálogo se verá reflejado en todas las pantallas y vistas que hacen referencia éste.

La venta de un catálogo se ve en el sistema como lo muestra la figura 5.12.



Nombre	Paterno	Materno	Título	Sobrenombre	Nombre_mostrar	Empresa	Departamento	Puesto
Ana	Juárez	Ariza		Ana Juárez		SaitoSoft S.A. de C.V.		
Jose Luis	Hernández					Gas Metropolitano		Gerente de Finanzas
Luis	Sánchez					Gas Metropolitano		Jefe de Contabilidad
José Luis	Marin					Gas Metropolitano		Jefe de Desarrollo
Luis	Alva					SaitoSoft S.A. de C.V.		
Ulises	Esteban			Ulises Esteban		Gas Metropolitano		
Marco Antonio	Martínez					Gas Metropolitano		
Liliana	Hernández					SaitoSoft S.A. de C.V.		
Miguel Angel	Huerta					SaitoSoft S.A. de C.V.		
Fernando	Hurtado					SaitoSoft S.A. de C.V.		
Rosario	Reyes					SaitoSoft S.A. de C.V.		Consultor
Mateo	Saito	Hata		Mateo Saito	Mateo Saito	SaitoSoft S.A. de C.V.		
Manuel	Saldaña			Manuel Saldaña		SaitoSoft S.A. de C.V.		
Karin	Sokey	Garcia				SaitoSoft S.A. de C.V.		
Angélica	Nakayama					SaitoSoft S.A. de C.V.		
Isabel	Hernández					SaitoSoft S.A. de C.V.		
Rosa Elia	Ramos	Hernández				Gas Metropolitano		
Elizabeth	Carrillo					SaitoSoft S.A. de C.V.		
Miriam	Marin			Miriam Marin		SaitoSoft S.A. de C.V.		
Alicia	x					Gas Metropolitano		

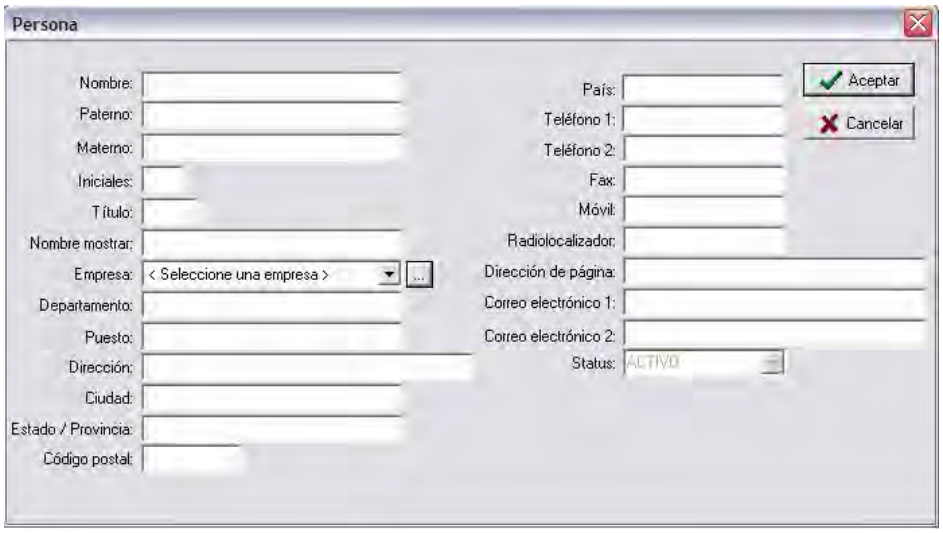
5.12 Pantalla de un catálogo de sistema (Personas).

La estructura de la interfaz gráfica de usuario se compone de dos partes principales, una barra de herramientas para ejecutar las funcionalidades propias del catálogo y una tabla o grid para desplegar los registros almacenados en él.

A continuación se describe la forma de operar las diferentes funcionalidades de un catálogo.

5.2.1 Agregar elementos

Para agregar un elemento a un catálogo deberemos realizar un clic en el botón de Agregar de la barra de botones del catálogo. A continuación se mostrará una ventaja dependiendo del catálogo en el cual se este trabajando para capturar los datos de un registro nuevo en el catálogo, como se muestra en la figura 5.13.



5.13 Ventana de captura de datos para el catálogo de personas.

5.2.2 Eliminar elementos

Para eliminar elementos del catálogo, deberemos seleccionar un elemento en particular que deseemos eliminar, realizando un clic en el grid sobre él y posteriormente dar clic en el botón de eliminar, al hacer esto aparecerá una ventana de confirmación de que queremos eliminar el registro seleccionado, como aparece en la figura 5.14.

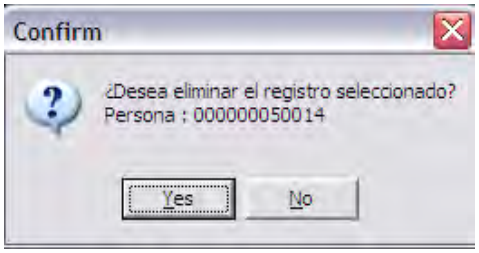


Figura 5.14 Ventana de confirmación para eliminar un registro.

Si confirmamos la operación se eliminará el registro y desparecerá del grid de la pantalla principal del catálogo.

5.2.3 Modificar elementos

Para modificar los datos de un elemento en particular que se haya dado de alta previamente, deberemos seleccionar primero dicho registro y al realizar clic en el botón de modificar de la barra de herramientas, al realizar esto aparecerá la ventana que se mostró cuando dimos de alta el registro pero con la información del registro a modificar, como se muestra en la siguiente figura.

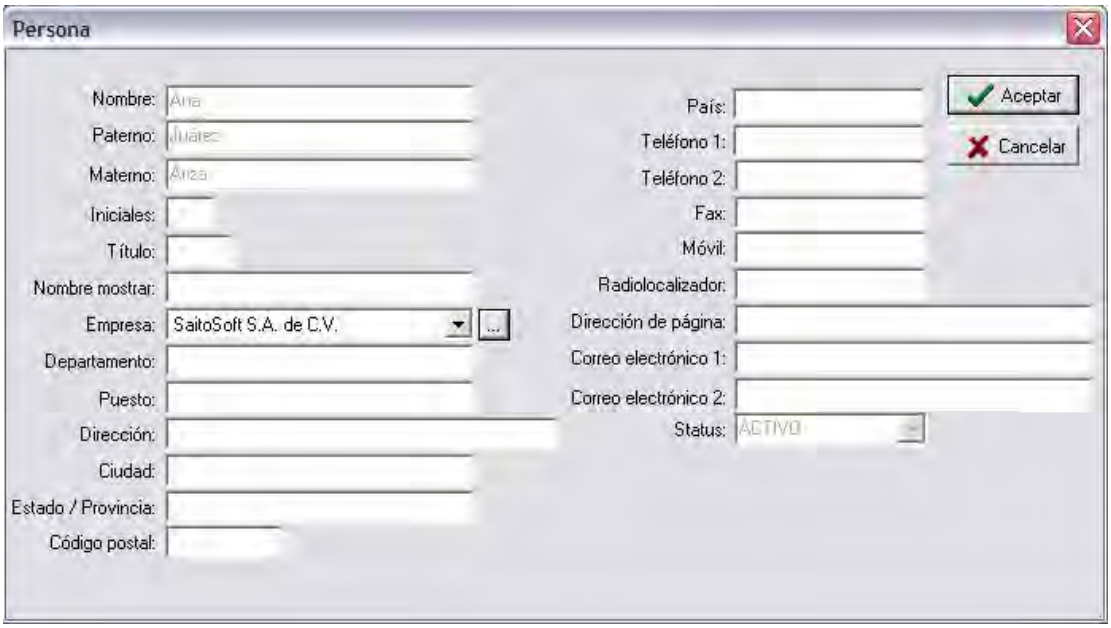


Figura 5.15 Ventana de modificación de información.

5.2.4 Filtrar información

El filtrar la información consiste en especificar un valor para todos los registros en un campo que es común para todos ellos, es decir los campos que pueden clasificar a la información como un estatus, una empresa, un tipo de reunión, etc.

Al realizar clic sobre el botón de consultar se mostrará la ventana de la figura 5.16, la cual contiene campos del registro de información del catálogo que son comunes para poder filtrar u obtener un conjunto de registros más reducido. Para cada uno de los campos

mostrados en esta ventana se pueden aplicar ciertas reglas de filtrado, los cuales son operadores lógicos que aplicarán sobre el valor que seleccionemos para dicho campo.

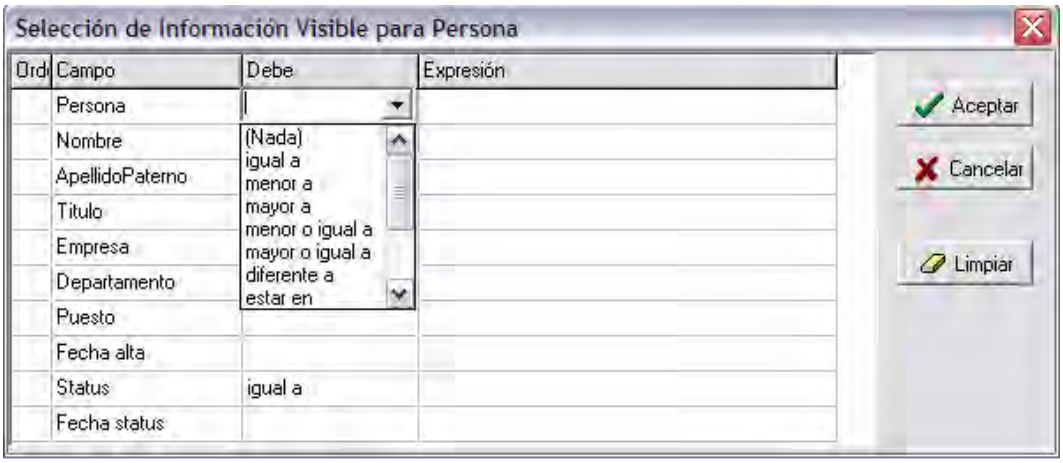


Figura 5.16 Ventana de filtrado de información en un catálogo.

Los operadores de lógicos que están disponibles son los siguientes:

Operadores
(Nada)
igual a
menor a
mayor a
menor o igual a
mayor o igual a
diferente a
estar en
estar entre
vacío

Una vez establecido el criterio deseado, al oprimir el botón aceptar desaparecerá la ventana de consulta y los registros que contendrá el catálogo serán aquellos que cumplan con el criterio establecido en la ventana de filtro.

5.2.5 Buscar información

La opción de búsqueda invocada por el botón buscar, permite mover el apuntador de registro seleccionado a aquel que coincida con lo especificado en el cuadro de texto de búsqueda, el cuál se muestra en la figura 5.17. La búsqueda se podrá realizar para encontrar información de todos los campos del registro o si especificamos la opción de buscar sólo el campo activo realizará la búsqueda únicamente para la columna seleccionada.

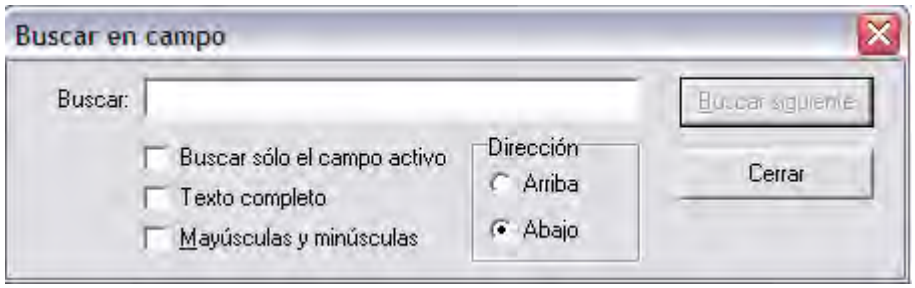


Figura 5.2.5 Ventana de búsqueda de información.

5.2.6 Imprimir un reporte

El botón de imprimir despliega un reporte con la información de todos los registros contenidos en el catálogo. Este reporte ofrece la posibilidad de mandarse imprimir, así como generar un documento pdf de él o exportarlo a un archivo en Microsoft Excel.

Un ejemplo de cómo se visualiza un reporte de un catálogo se muestra en la siguiente figura.

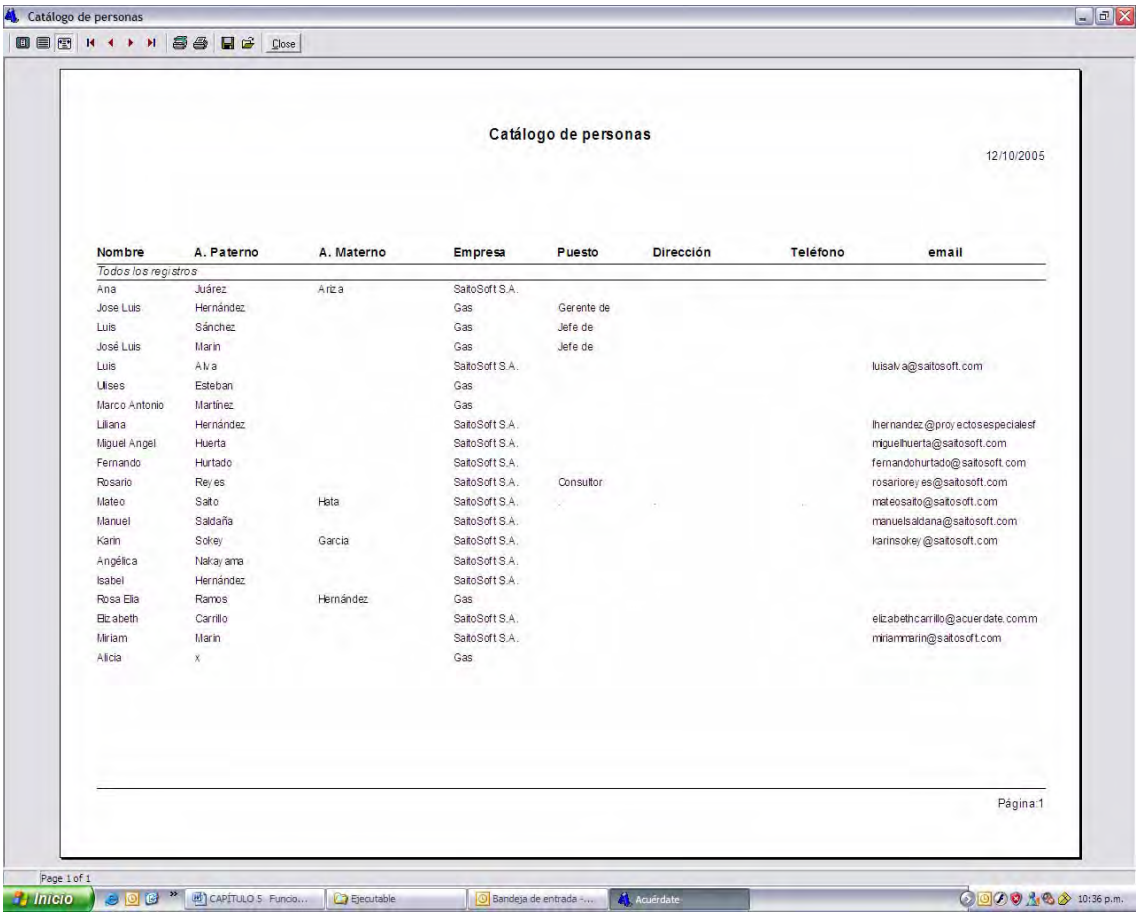


Figura 5.17 Reporte de un catálogo.

En la siguiente tabla se muestra un resumen de las funcionalidades de la barra de botones que aparecen en la ventana de cualquier catálogo de sistema.

Botón		Acción
		Agrega un nuevo registro. Activa la ventana de captura/edición para registrar los datos.
		Elimina el registro seleccionado, previa confirmación del usuario.
		Modifica la información del registro seleccionado. Activa la ventana de captura/edición para modificar los datos.
		Permite seleccionar un conjunto de registros en función de condiciones especificadas por el usuario.
		Hace una búsqueda específica.
		Genera un listado de los registros que se visualizan en pantalla manteniendo el orden de los mismos.

Botón		Acción
		Actualiza la información.
		Cierra la ventana actual y regresa a la ventana anterior (desde donde fue llamada).

5.2.7 Listado y descripción de los catálogos del sistema

Se lista a continuación los catálogos del sistema y se describe brevemente la funcionalidad de cada uno de ellos.

1. Catálogo de empresas

En este catálogo se registran las empresas con la finalidad de poder asociar un contacto o persona a una empresa preestablecida.

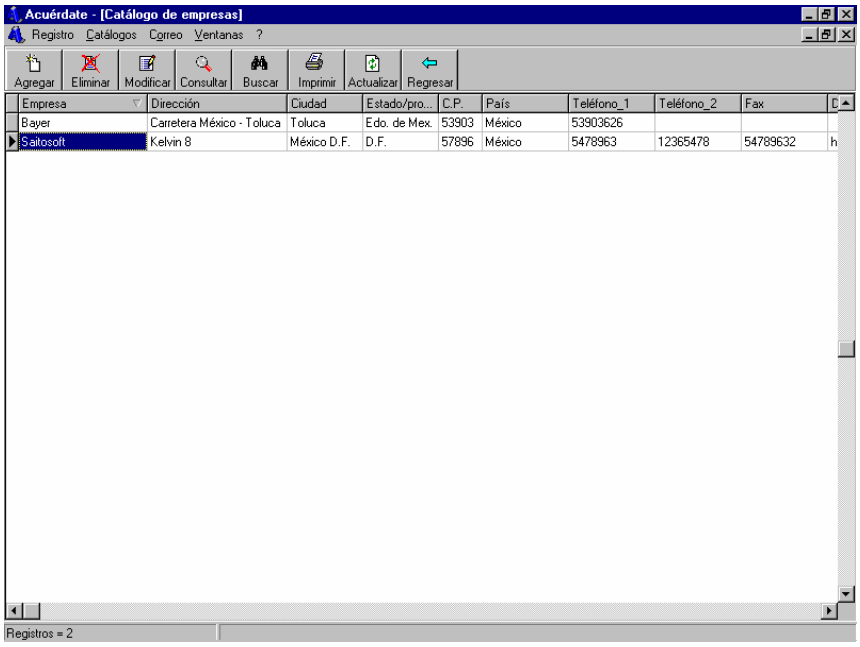
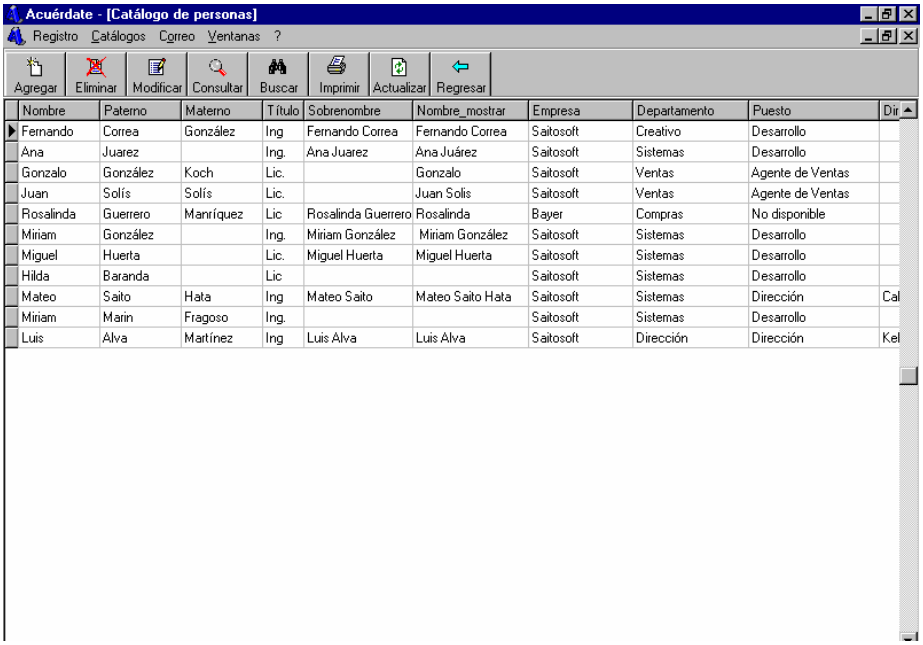


Figura 5.18 Catálogo de empresas.

2. Catálogo de persona

La finalidad de este catálogo es dar de alta las personas y contactos que tendrán participación dentro de nuestras reuniones de trabajo.

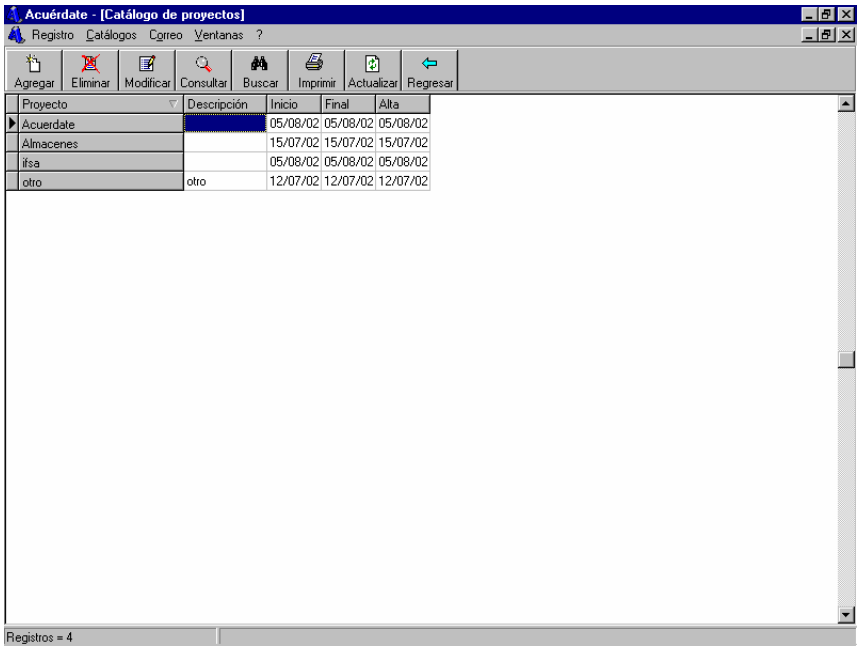


Nombre	Paterno	Materno	Título	Sobrenombre	Nombre_mostrar	Empresa	Departamento	Puesto	Dir
Fernando	Correa	González	Ing	Fernando Correa	Fernando Correa	Saitosoft	Creativo	Desarrollo	
Ana	Juarez		Ing.	Ana Juarez	Ana Juárez	Saitosoft	Sistemas	Desarrollo	
Gonzalo	González	Koch	Lic.		Gonzalo	Saitosoft	Ventas	Agente de Ventas	
Juan	Solís	Solís	Lic.		Juan Solis	Saitosoft	Ventas	Agente de Ventas	
Rosalinda	Guerrero	Manríquez	Lic	Rosalinda Guerrero	Rosalinda	Bayer	Compras	No disponible	
Miriam	González		Ing.	Miriam González	Miriam González	Saitosoft	Sistemas	Desarrollo	
Miguel	Huerta		Lic.	Miguel Huerta	Miguel Huerta	Saitosoft	Sistemas	Desarrollo	
Hilda	Baranda		Lic			Saitosoft	Sistemas	Desarrollo	
Mateo	Saito	Hata	Ing	Mateo Saito	Mateo Saito Hata	Saitosoft	Sistemas	Dirección	Cal
Miriam	Marin	Fragoso	Ing.			Saitosoft	Sistemas	Desarrollo	
Luis	Alva	Martínez	Ing	Luis Alva	Luis Alva	Saitosoft	Dirección	Dirección	Kel

5.19 Catálogo de persona.

3. Catálogo de proyecto

Este catálogo tiene la finalidad de registrar los proyectos para los cuales comúnmente se establecen reuniones de trabajo.



5.20 Catálogo de proyecto.

4. Catálogo de tema

El catálogo de tema nos permite establecer temas de interés o de alta prioridad que necesitemos tener clasificados o aquellos que sean repetibles, por ejemplo: finanzas,

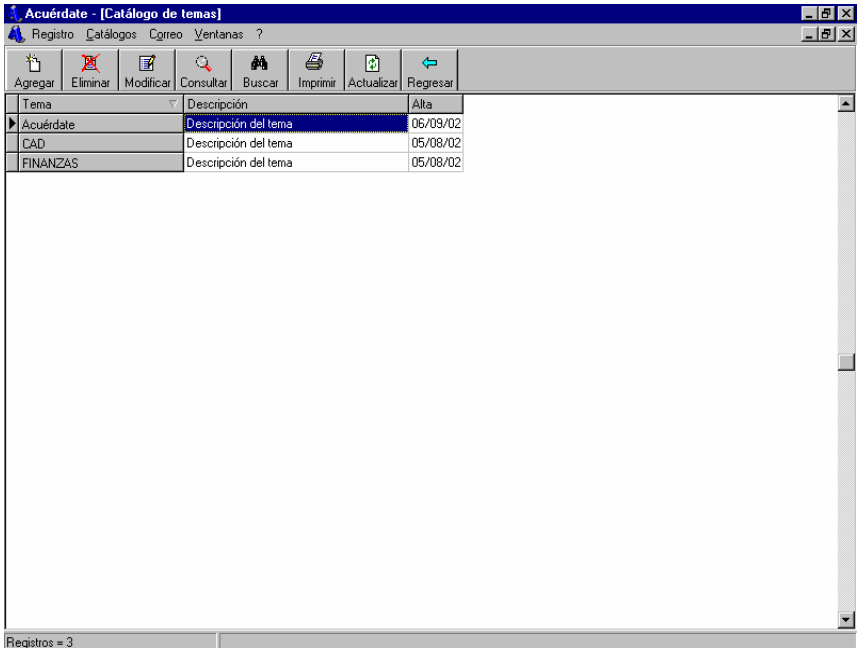


Figura 5.21 Catálogo de temas.

5. Catálogo de reportes

La finalidad de este catálogo es poder parametrizar los títulos de los reportes predefinidos de la aplicación, para poder incluir claves ISO o identificadores que permitan utilizar dichos reportes oficialmente dentro de un proceso de calidad preestablecido.

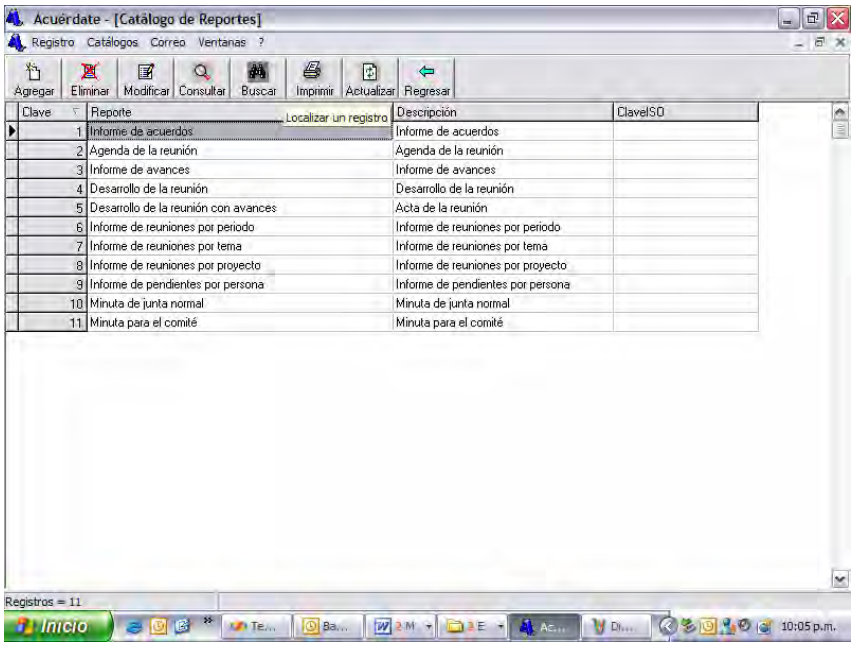


Figura 5.22 Catálogo de títulos de reportes.

5.3 Configuración de cuentas de correo electrónico

La configuración de cuentas de correo electrónico permite establecer los parámetros de conexión necesarios para permitir al Acuérdate leer mensajes de correo, identificar y descargar a su base de datos aquellos que hayan sido enviados por otros sistemas Acuérdate.

Para configurar una cuenta de correo electrónico deberemos seleccionar el submenú cuentas de la opción correo en el menú principal, a continuación aparecerá la pantalla que se muestra en la siguiente figura:

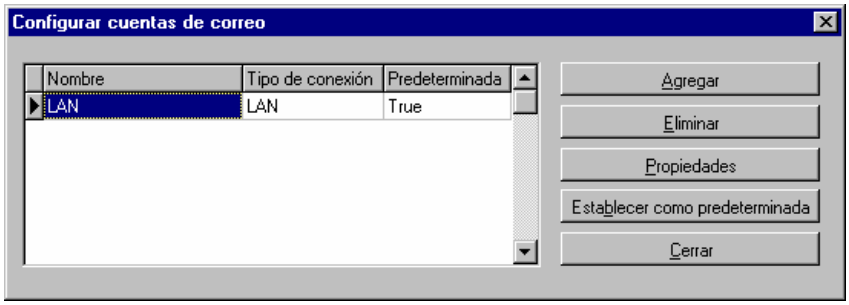


Figura 5.23 Configuración de cuentas de correo electrónico.

Posteriormente deberemos de hacer clic en el botón agregar de la pantalla para ingresar en la ventana de configuración de una cuenta de correo.

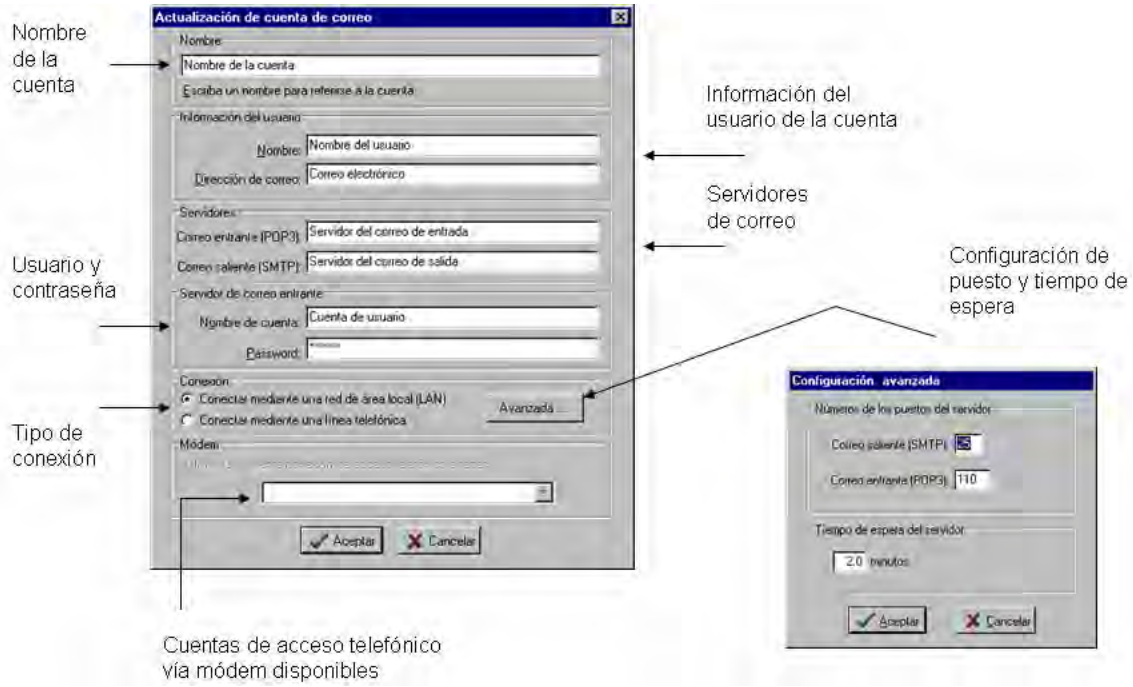


Figura 5.24 Parámetros de configuración de una cuenta de correo.

5.4 Planeación de una reunión de trabajo

La etapa inicial de una reunión de trabajo es la planeación de la misma, como se mencionó en Capítulo 2 “Reuniones de trabajo”, el objetivo principal es registrar el título y objetivo de la reunión, así como la fecha y los participantes que deseemos convocar.

De esta manera al hacer clic en botón de planeación estando posicionados en un directorio dentro del árbol de un grupo de reuniones predefinidas, el sistema mostrará el calendario

para especificar la fecha, hora y duración de la reunión como se muestra en la siguiente figura:

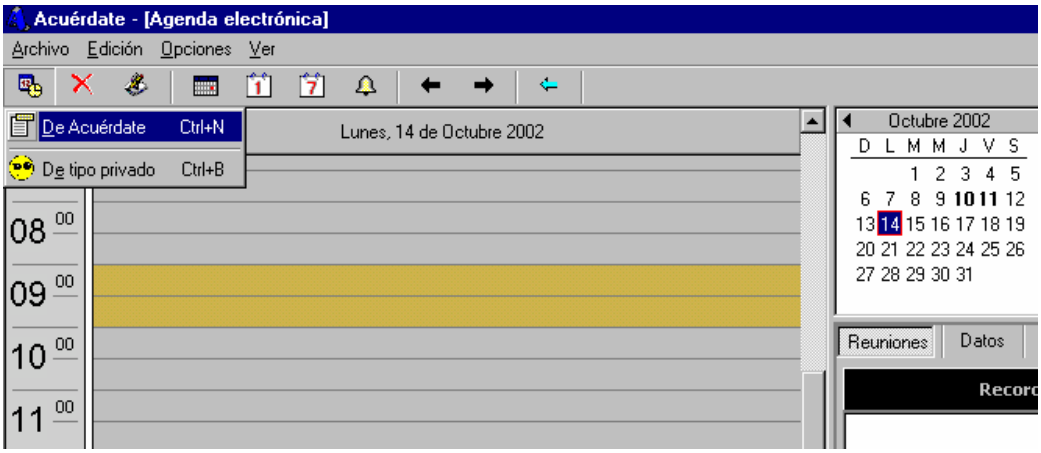


Figura 5.25 Agenda para planeación de reuniones.

Una vez registrada la fecha y hora de la reunión aparecerá una ventana para registrar los demás atributos de la reunión en su etapa de planeación esta ventana se muestra en la figura 5.26.

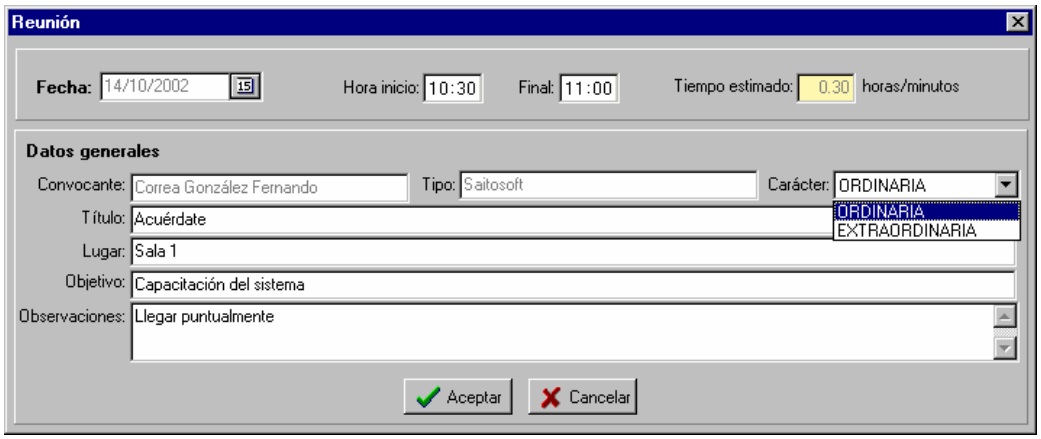


Figura 5.26 Registro de los datos generales de una reunión en su etapa de planeación.

Como se puede apreciar en la figura es posible especificar también el carácter de la reunión para notificar a los convocados el tipo de reunión de la cual se trata.

Una vez terminada la captura de los datos generales de la reunión, se procederá a establecer una agenda con los diferentes puntos a tratar, por lo tanto la siguiente ventana

que se mostrará es la de la figura 5.27 para capturar un primer punto que queramos establecer para la agenda, al dar clic en aceptar aparecer nuevamente esta ventana para ingresar un segundo punto y a así sucesivamente todos los puntos que deseemos agregar.

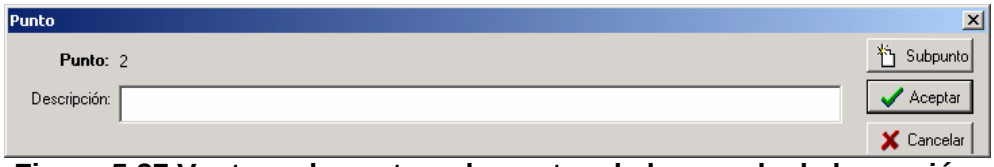


Figura 5.27 Ventana de captura de puntos de la agenda de la reunión.

Sí es necesario agregar un subpunto a la agenda la ventana de captura de puntos de la agenda cuenta con un botón de subpunto para adicionar subpuntos a puntos preestablecidos, al hacer clic sobre el botón de subpuntos aparecerá la ventana de la siguiente figura:

Figura 5.28 Ventana de captura de subpuntos.

Finalmente después de haber definido la agenda en cuanto a puntos y subpuntos, la ventana que aparecerá será la de asignación de participantes a la reunión (Figura 5.29). Esta ventana contiene una doble lista, una contiene todos los contactos establecidos en el catálogo de personas y la otra contendrá aquellos que se asignen de la primera lista los cuales serán aquellos que queremos que participen, es decir los convocados.

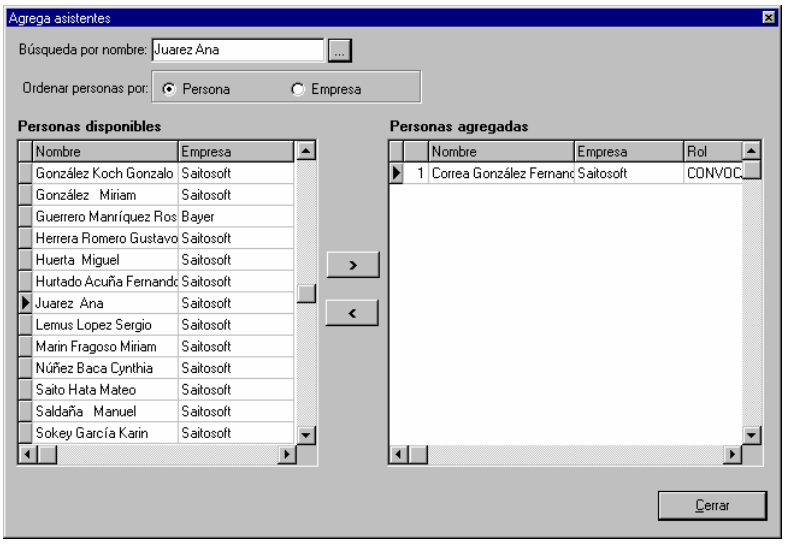


Figura 5.29 Ventana para asignar participantes a la reunión.

Esta misma ventana ofrece también la posibilidad de agregar contactos, en caso de que no se encuentren disponibles en nuestro catálogo, con tan solo hacer clic en el botón etiquetado con tres puntos aparecerá la ventana de captura de contactos tal como la implementa el catálogo.

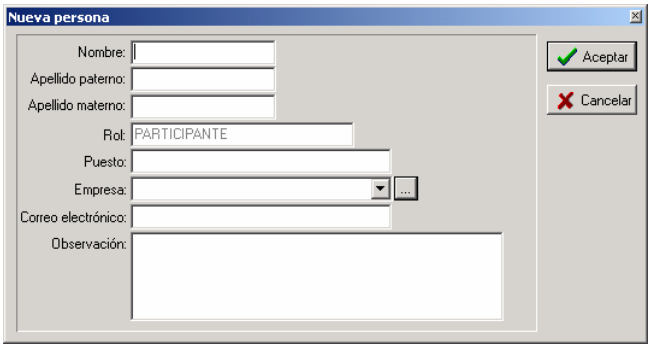


Figura 5.30 Ventana de captura de contactos.

Una vez que se terminan de agregar asistentes a la reunión se completa el proceso de registro de la etapa de planeación. A continuación aparecerá un registro en el panel de información mostrando como columnas los principales atributos de la reunión planeada como lo muestra la figura 5.31.

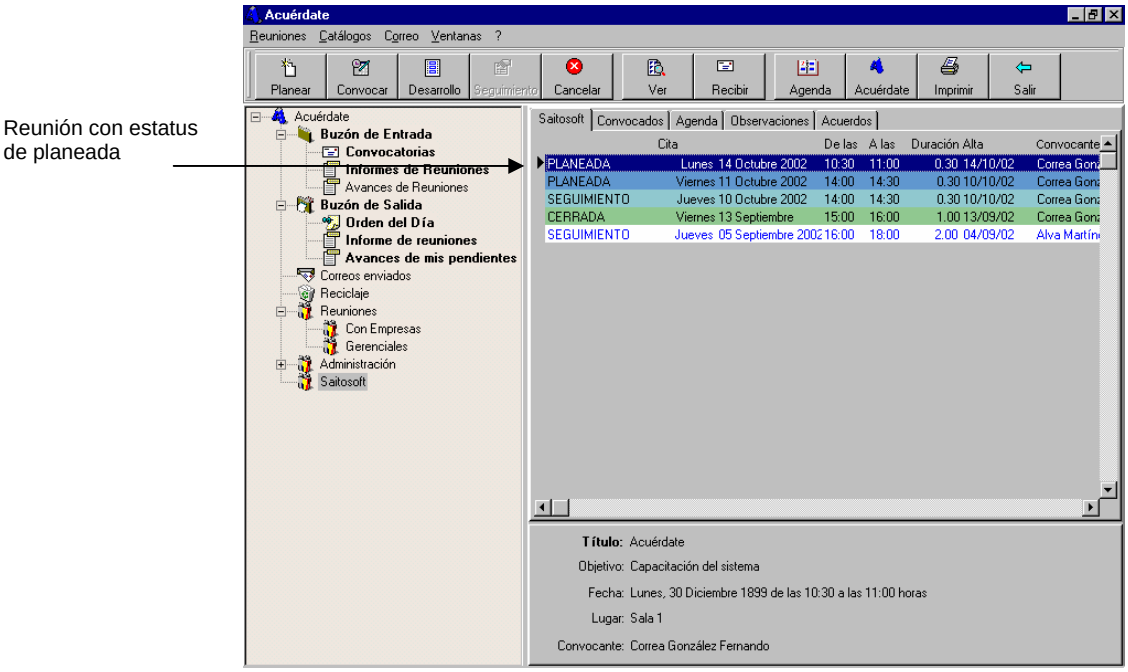


Figura 5.31 Registro de una reunión en el panel de información.

5.5 Convocatoria de una reunión.

Una vez terminada la planeación de una reunión se continuará con la convocatoria de la misma. Esta etapa consiste en notificar a los convocados que se llevará acabo una reunión en donde ellos están invitados, por lo tanto es importante hacerles llegar la información que registramos en la etapa de planeación como, el objetivo de la reunión y la agenda, de esta manera los convocados podrán prepararse para la reunión y ofrecer una mejor aportación a la misma.

Una vez planeada la reunión aparecerá un registro en el directorio donde se haya estado ubicado en el árbol cuando se inicio la planeación, sin embargo también aparecerá un registro en el subdirectorio Orden del día, dentro del directorio Buzón de salida como nos muestra la siguiente figura:

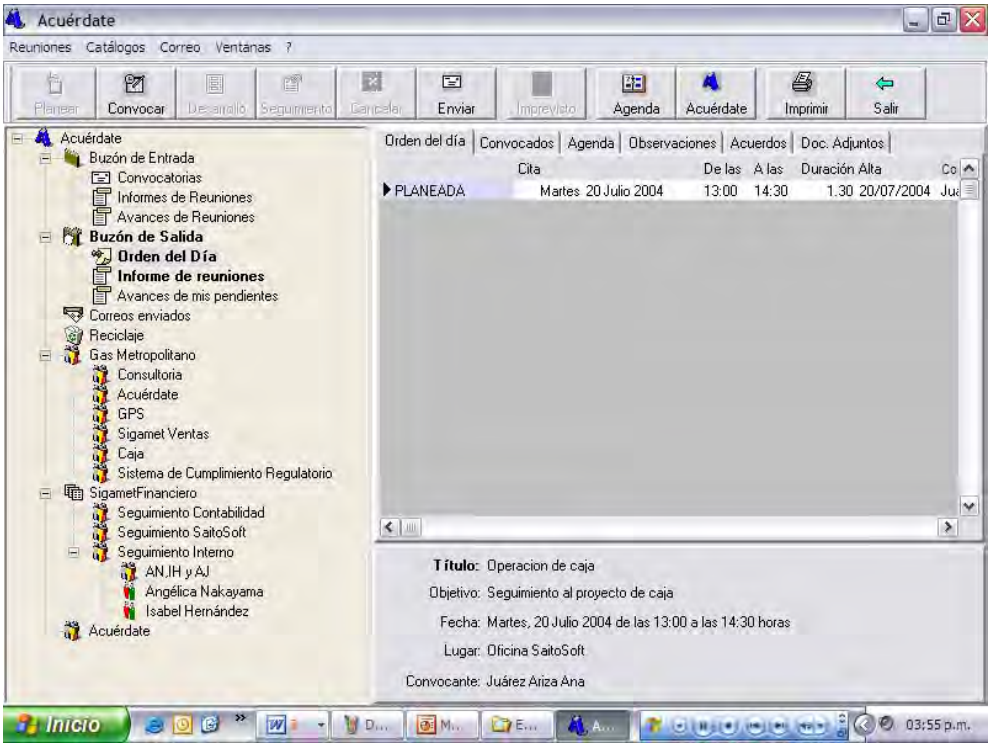


Figura 5.32 Ubicación automática de una reunión planeada en el buzón de salida.

Esta asignación automática a la bandeja de salida de la reunión planeada, es debido a que el sistema prepara la información para poder ser enviada por correo electrónico a los convocados. La forma de realizar el envío es haciendo clic derecho sobre el registro de la reunión en el panel de información, al hacer esto aparecerá un menú con la opción de enviar como lo muestra la siguiente figura:

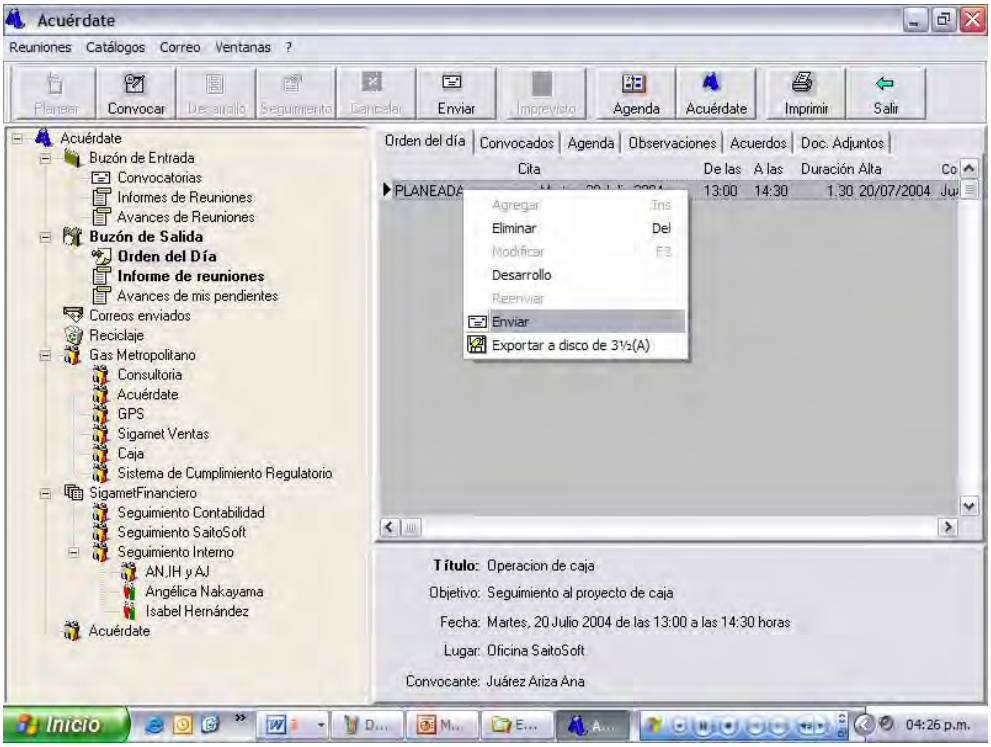


Figura 5.33 Envío de la información de una reunión por correo electrónico.

Posteriormente al hacer clic en botón de enviar se mostrará la ventana de envío de correo electrónico, entre la información y funcionalidades que aparecen en esta ventana podemos encontrar:

a) Los convocados a la reunión, que serán los destinatarios a los que se les enviara el correo.

b) El asunto, que para este caso será “Convocatoria de una reunión”

c) CC – Con copia, el cual nos permite enviar una copia del correo a otros destinatarios.

d) Archivos adjuntos. De manera predeterminada se envía la orden del día cuando se convoca a una reunión, ésta se envía en forma de reporte que contiene también los datos generales (objetivo, título, fecha, etc). Otro archivo que también se adjunta, es un archivo comprimido que al recibirlo un convocado que cuente con el sistema Acuérdate, actualizará su base de datos registrando la reunión dentro del sistema.

e) El contenido del mensaje de correo, el cual contiene la descripción de la reunión.

f) Un botón para enviar la información una vez que ha sido confirmada.

g) Botón para modificar el correo electrónico o datos del destinatario.

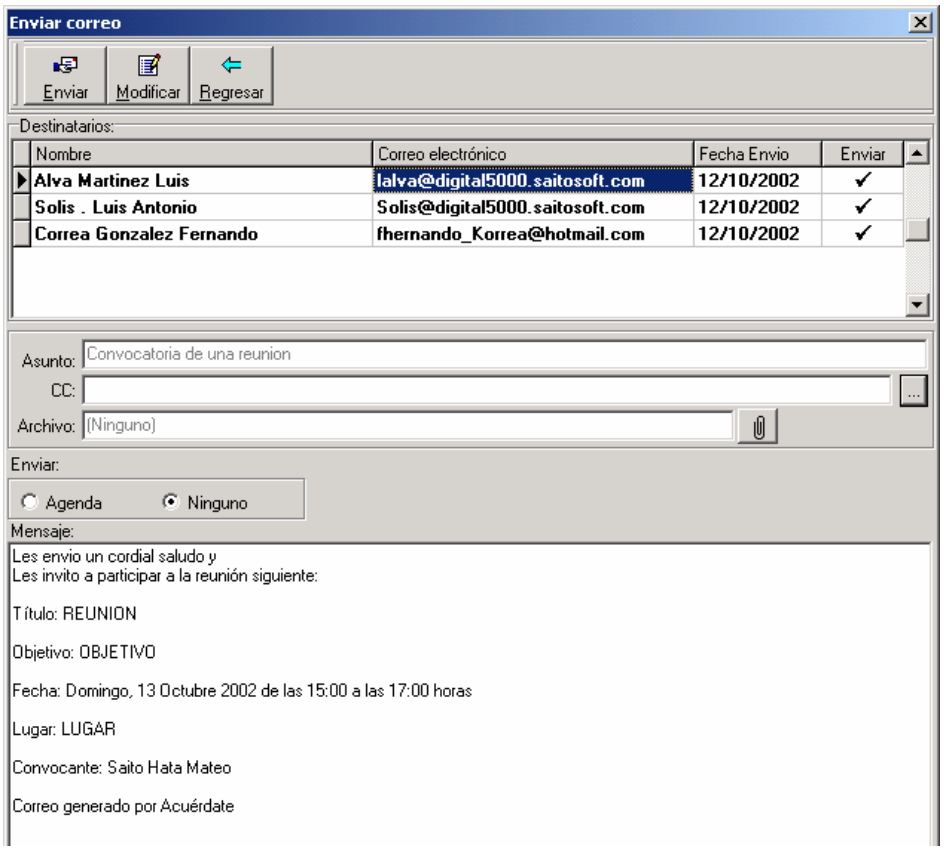


Figura 5.34 Ventana de envío de correo electrónico.

Cuando damos clic en el botón de enviar de la figura 5.34 comenzará el proceso de envío de correo el cual deberá tardar tan solo unos segundos cuando se tiene una conexión de mediana velocidad (36 kilo bits por segundo). La ventana que mostrará el proceso de envío es la que se muestra en la siguiente figura:

Figura 5.35 Ventana del estatus del envío de correo electrónico.

5.6 Desarrollo de una reunión

La etapa de desarrollo comienza cuando da inicio la reunión, para llevar el registro de información de esta etapa, deberemos seleccionar el registro que identifica a la reunión en el panel de información (Figura 5.36) y a continuación deberemos de dar clic sobre el botón de desarrollo de la barra de botones principal.

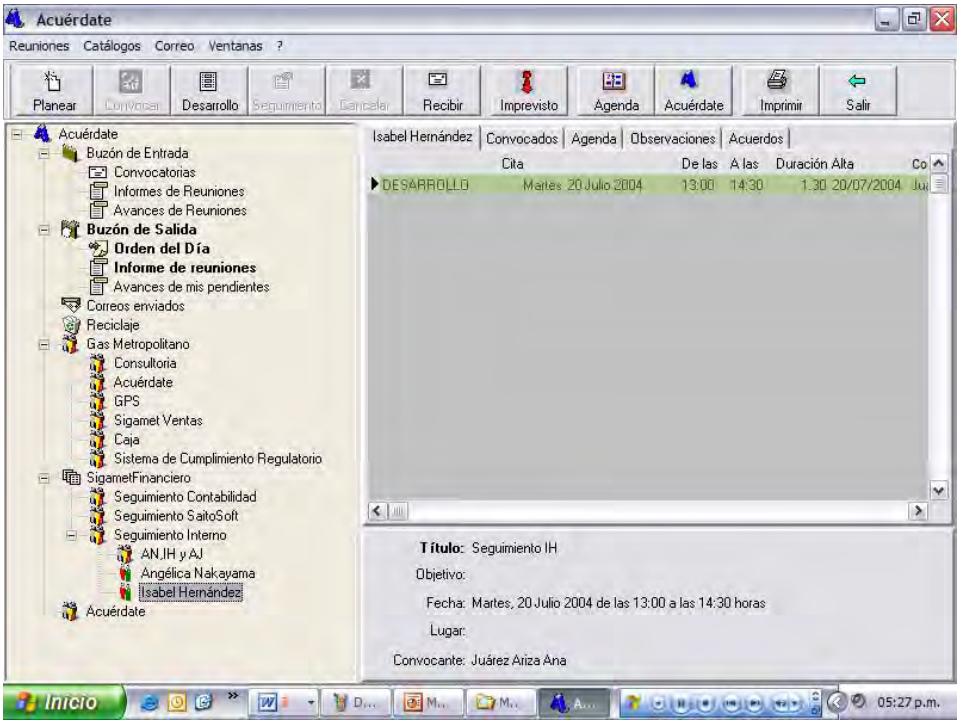


Figura 5.36 Ubicación de una reunión en etapa de desarrollo.

Al hacer clic sobre el botón de desarrollo se mostrará la ventana de la figura 5.37.

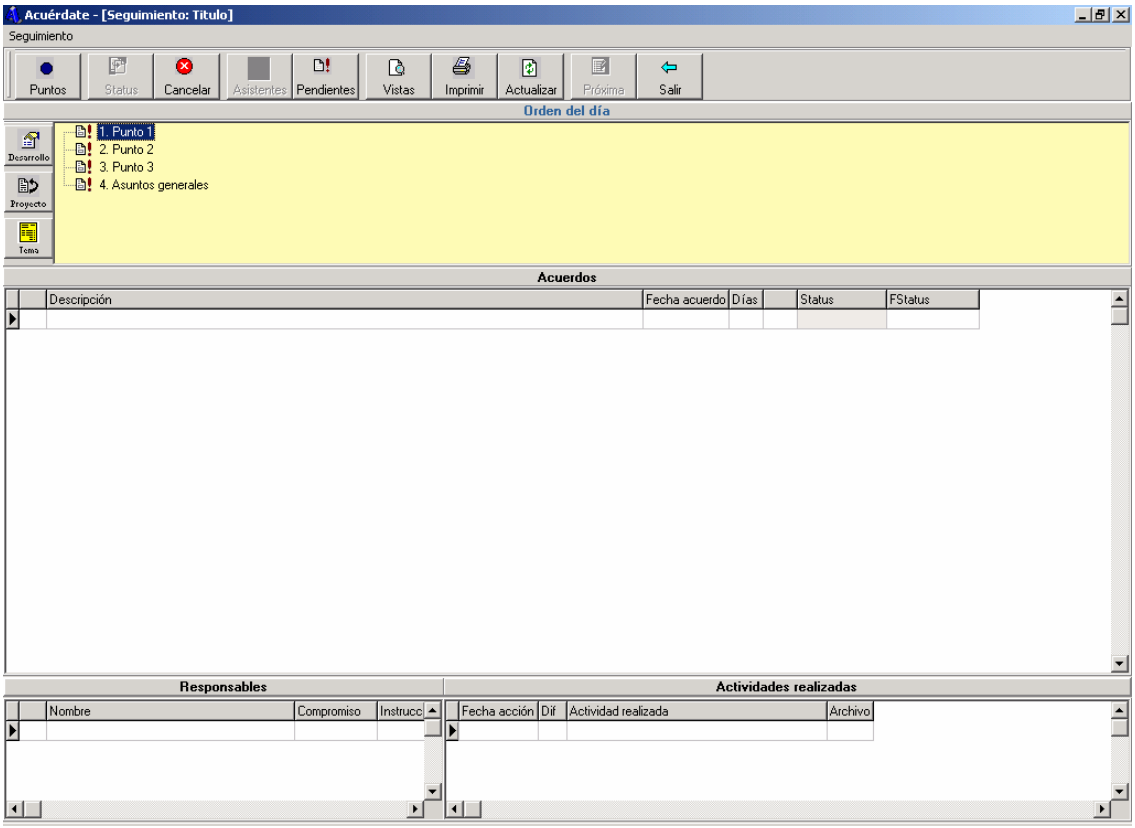


Figura 5.37 Ventana para el desarrollo de una reunión.

La ventana de desarrollo tiene como finalidad registrar los acuerdos obtenidos durante el transcurso de la reunión, estos acuerdos se relacionan a los puntos que se tienen de la agenda de la reunión, la cual fue desarrollada en la planeación de la reunión, como se muestra en la figura 5.38.

En cualquier momento durante el desarrollo de la reunión podremos cambiar puntos y subpuntos de la agenda, permitiéndonos incorporar cualquier otro tema a tomar durante el transcurso de la misma.

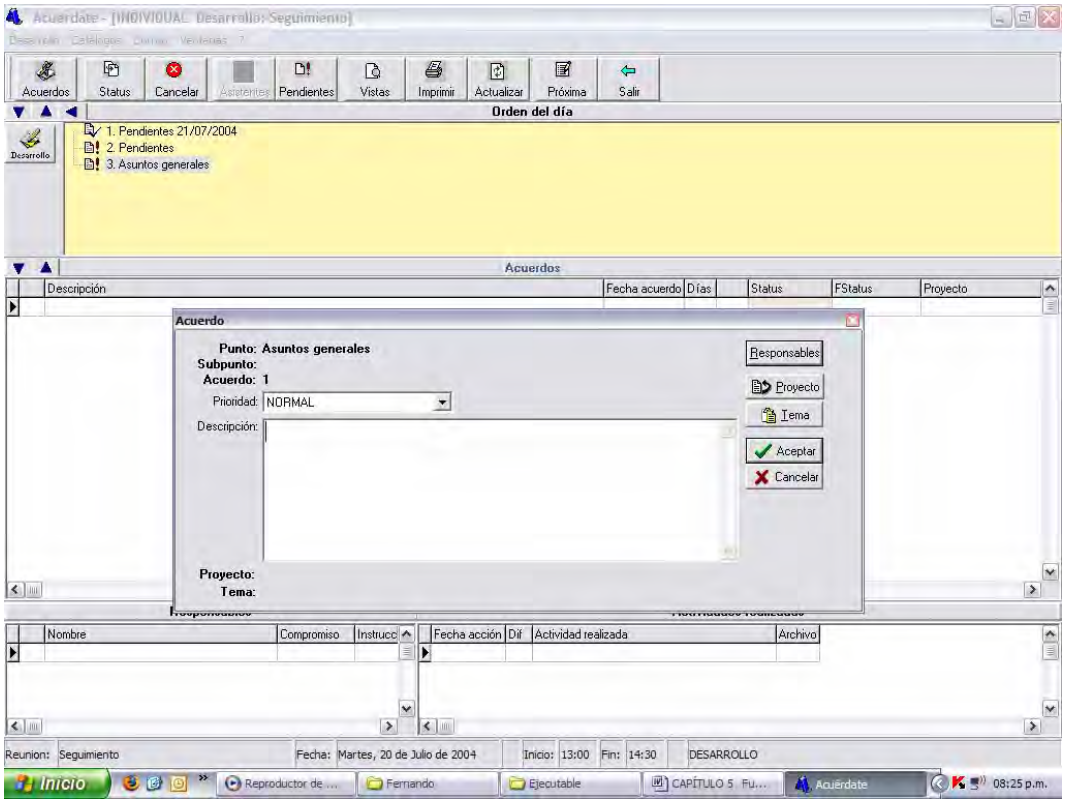


Figura 5.38 Registro de acuerdos en la etapa de desarrollo de una reunión.

De la misma forma a cada acuerdo relacionamos responsables de los mismos estableciendo una fecha compromiso para llevarlo acabo. En la figura 5.39 se muestra la forma en como asociamos responsables a un acuerdo.

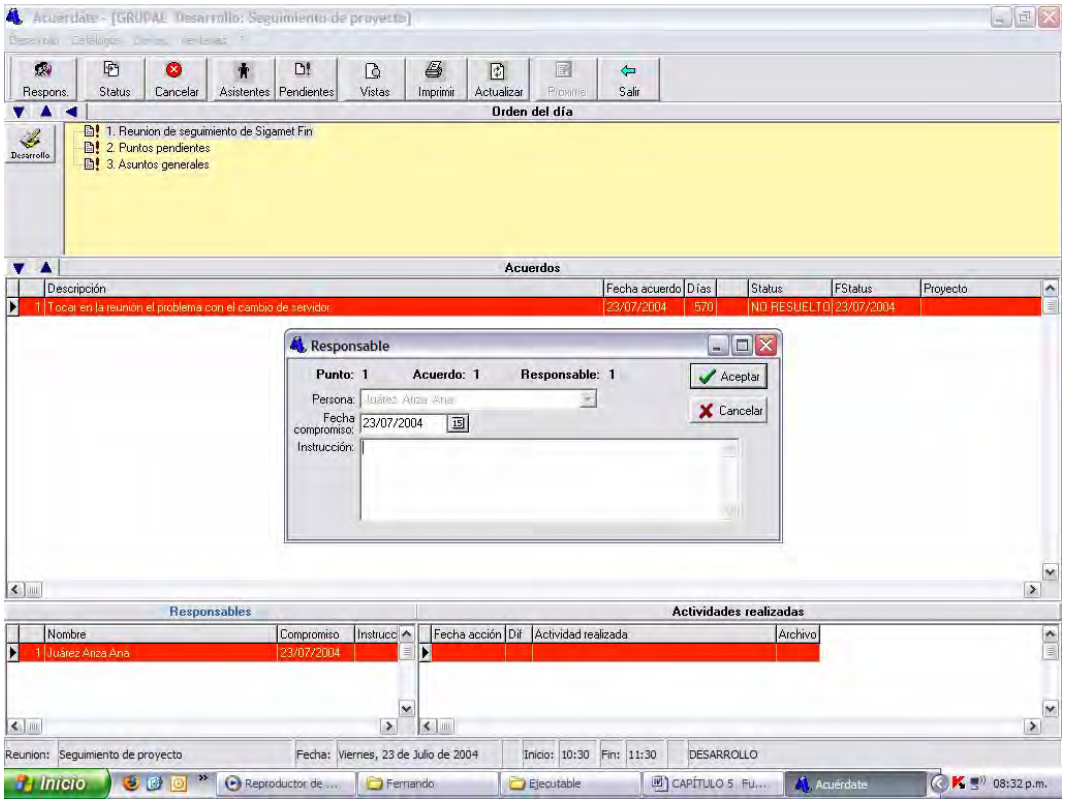


Figura 5.39 Registro de responsables y fecha compromiso a un acuerdo.

Una vez discutidos todos los puntos a tratar durante el transcurso de la reunión y registrados los acuerdos derivados de los mismos, con sus responsables y fechas compromisos termina la reunión de trabajo. El siguiente paso en el uso del sistema es salir de la ventana de desarrollo de una reunión haciendo clic en el botón Salir.

Al hacer clic sobre el botón de salir un cuadro de dialogo nos preguntará si se ha llegado al termino de la reunión, esto para confirmar el fin de la misma debido a el sistema cambiará automáticamente el estatus a seguimiento.

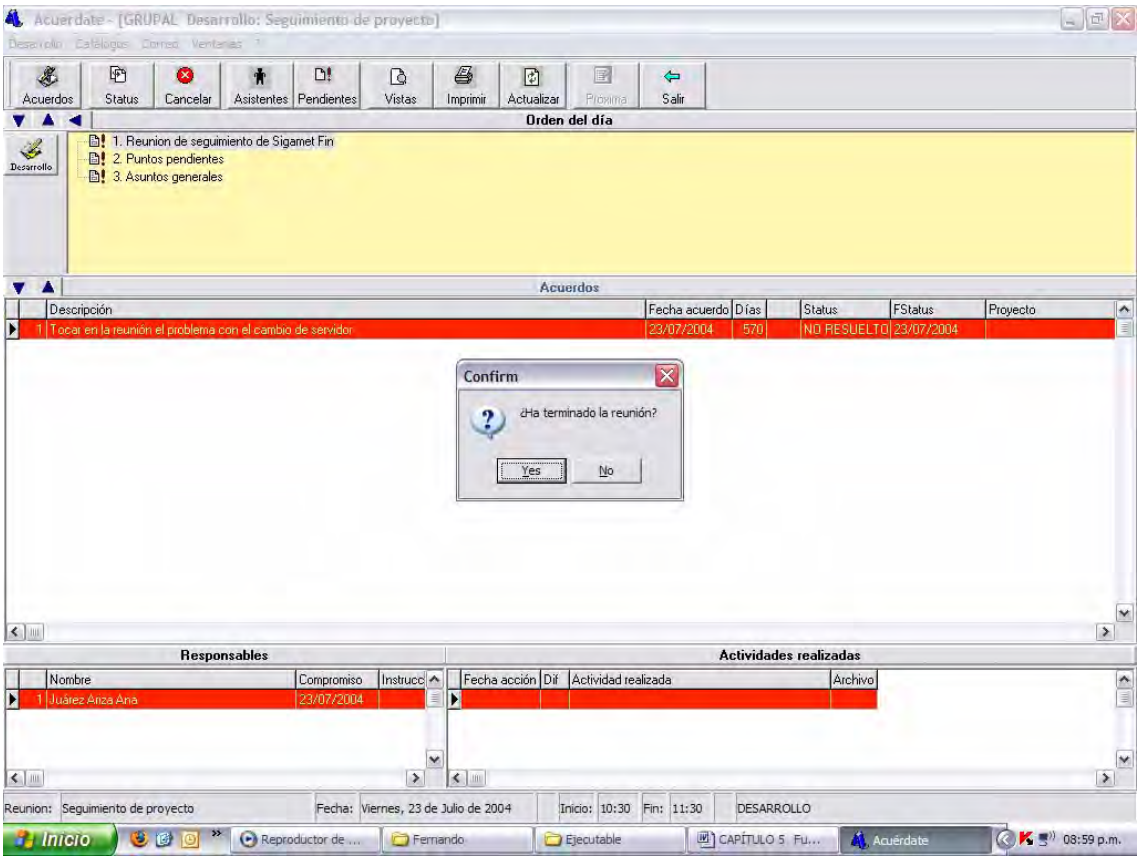


Figura 5.40 Confirmación de termino de una reunión.

Al aceptar el término de la reunión nos aparecerá otra ventana de confirmación preguntándonos acerca de si queremos enviar la información registrada en el desarrollo de la reunión a los participantes por correo electrónico, como se muestra en la siguiente figura:

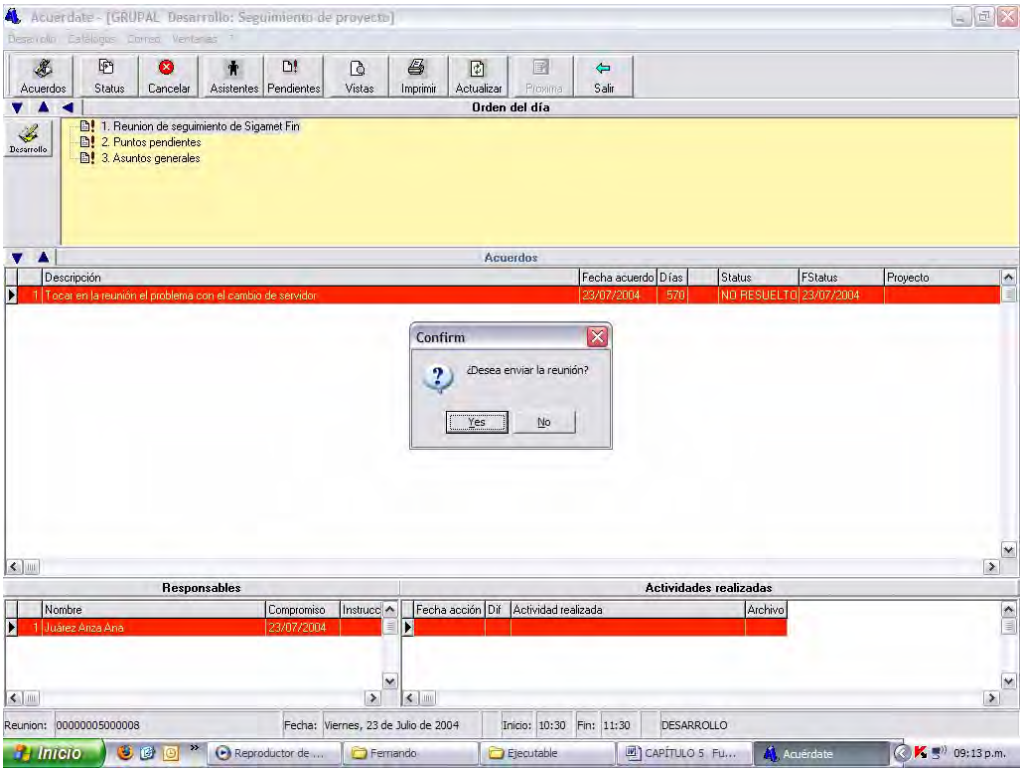


Figura 5.41 Confirmación de envío de la información registrada en el desarrollo de la reunión.

Finalmente se cerrará la ventana de desarrollo de una reunión mostrándonos nuevamente la ventana principal de la aplicación y ubicándonos en el registro de la reunión con estatus ahora de seguimiento.

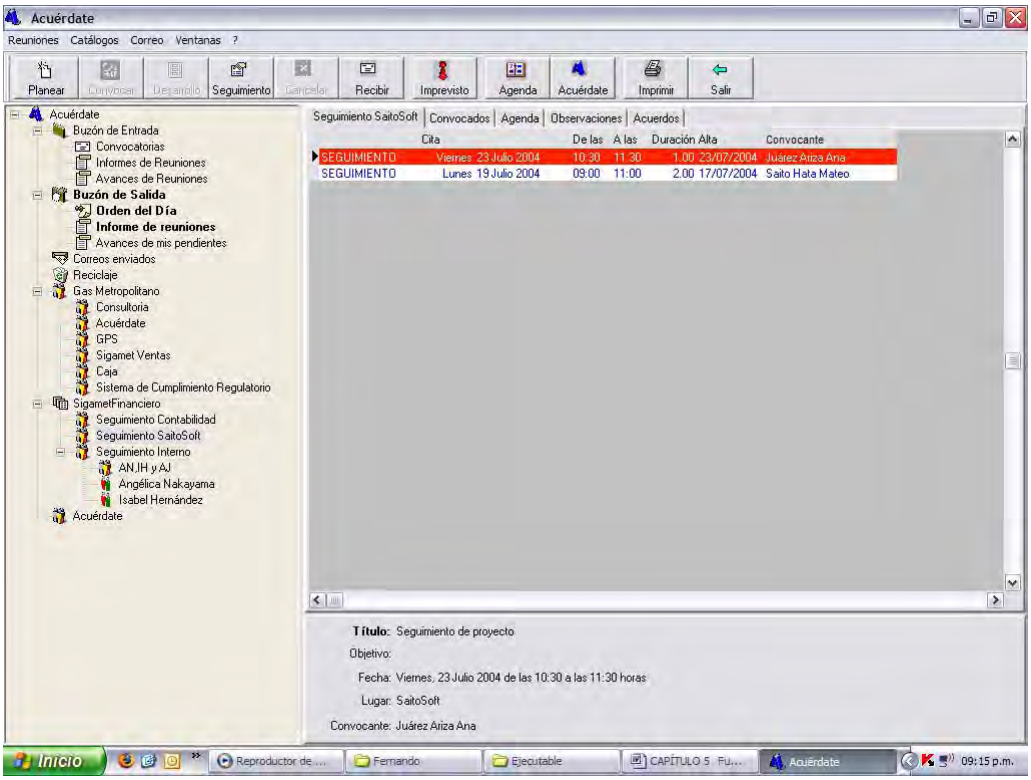


Figura 5.42 Registro de la reunión después de la etapa de desarrollo.

5.7 Seguimiento a los acuerdos derivados de una reunión

Una vez terminado el desarrollo de una reunión y de haber establecido acuerdos durante su transcurso, el siguiente paso es dar seguimiento al cumplimiento de los mismos.

Para dar seguimiento a los acuerdos derivados de una reunión de trabajo deberemos de seleccionar una reunión en estatus de seguimiento como se muestra en la siguiente figura:

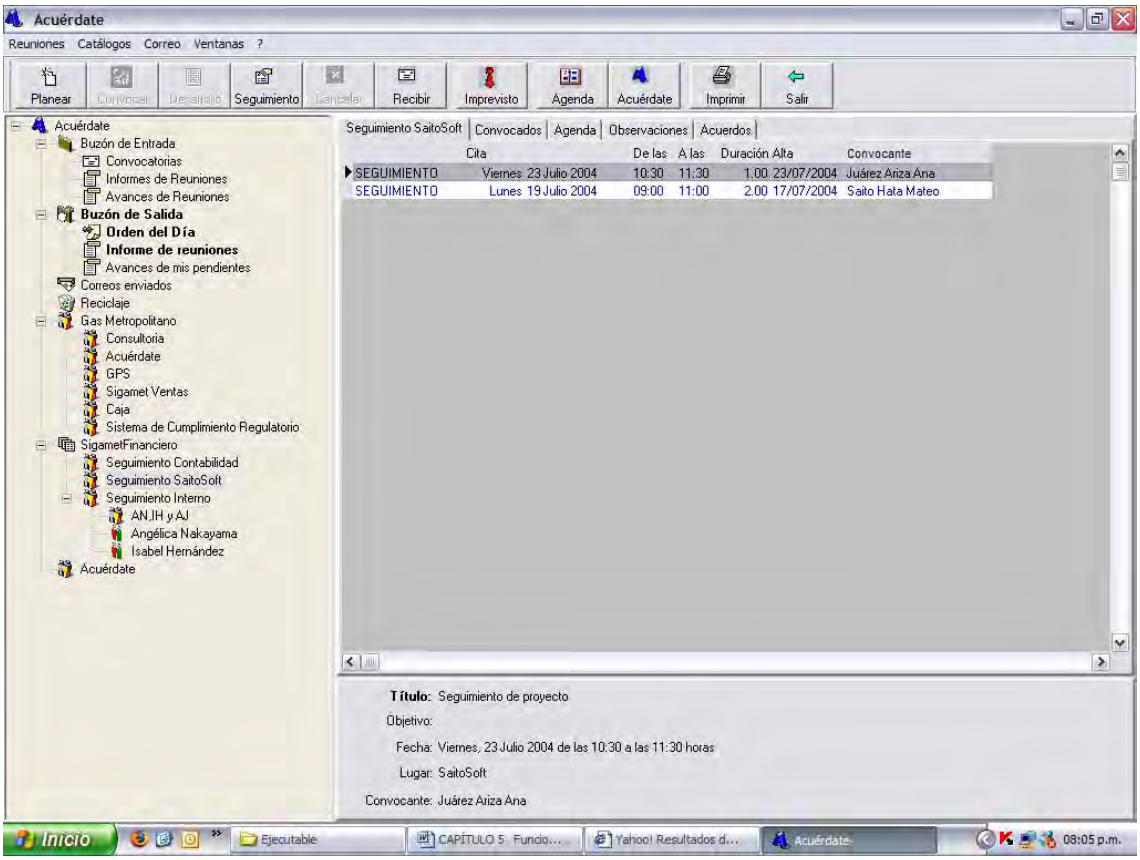


Figura 5.43 Registro de una reunión en estatus de seguimiento.

Posteriormente al hacer clic sobre el botón de Seguimiento de la barra de botones se abrirá la misma ventana que se utilizó en el desarrollo de la reunión, como se muestra en la figura 5.44. Esta ventana nos permitirá visualizar lo registrado como avance en cada uno de los acuerdos que se ingresaron en la reunión en la etapa de desarrollo, en el área denominada Actividades realizadas.

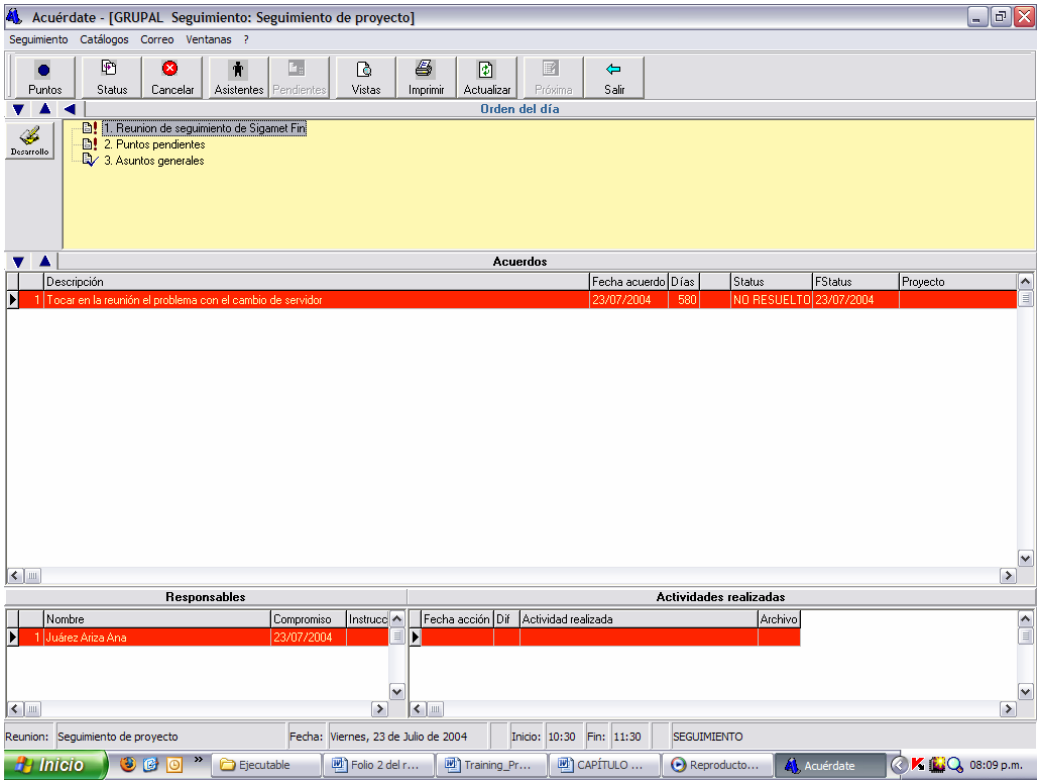


Figura 5.44 Seguimiento de acuerdos derivados de una reunión de trabajo.

La forma para registrar los avances es haciendo doble clic en el área de Actividades realizadas para un acuerdo en específico con lo cual aparecerá una siguiente ventana que nos permitirá escribir lo realizado a la fecha para dicho acuerdo.

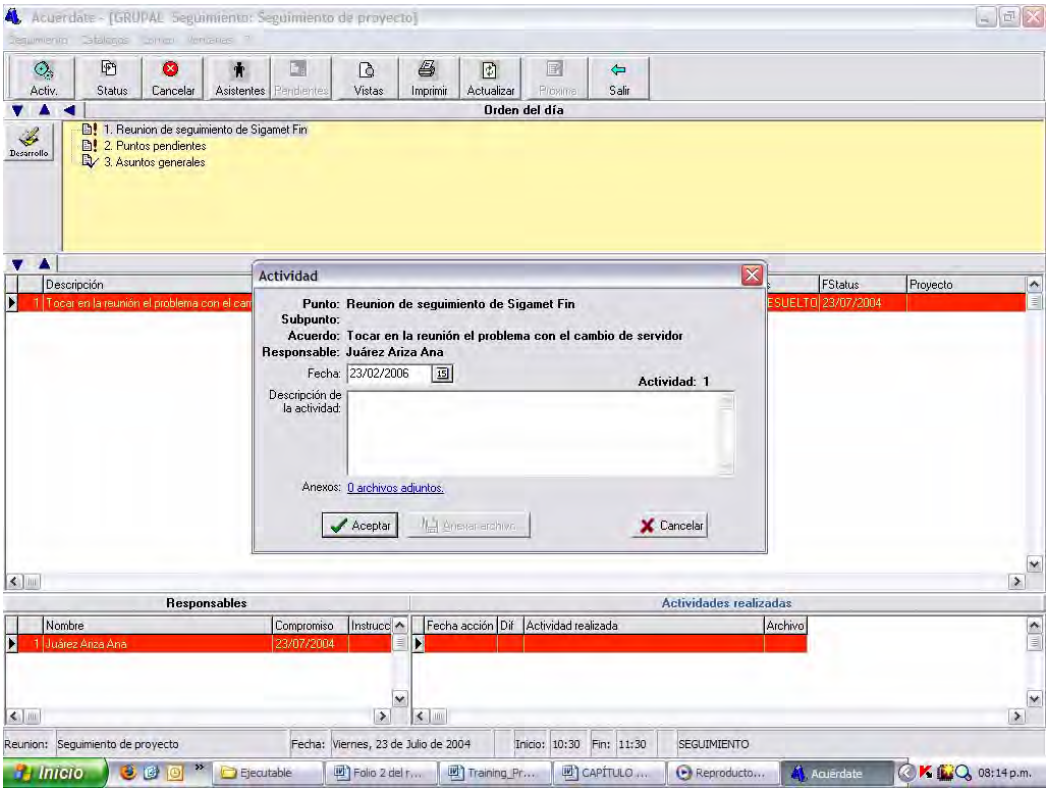
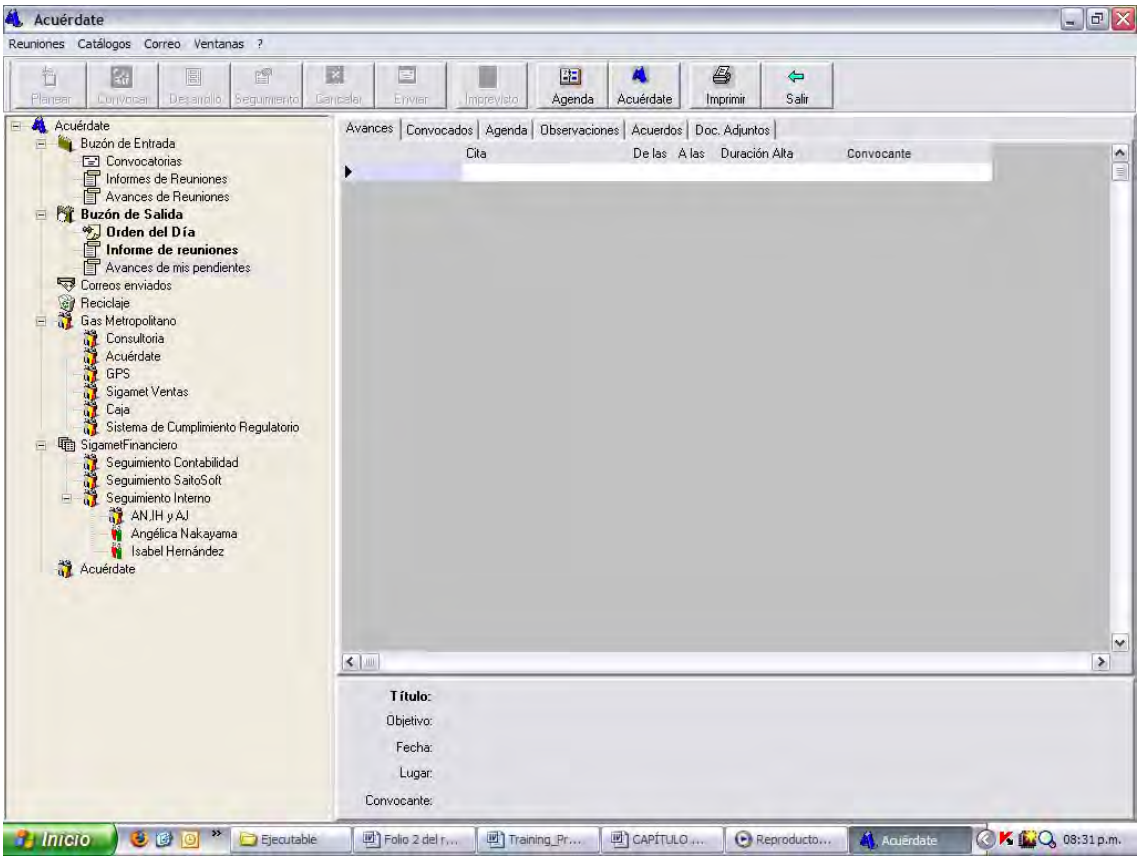


Figura 5.45 Registro de avances para un acuerdo derivado de una reunión.

Cabe mencionar que el registro de actividades para un acuerdo solo lo puede realizar el responsable del mismo, por lo que el sistema bloquea la inserción o actualización de registro de actividades si estamos en algún acuerdo que no nos corresponde.

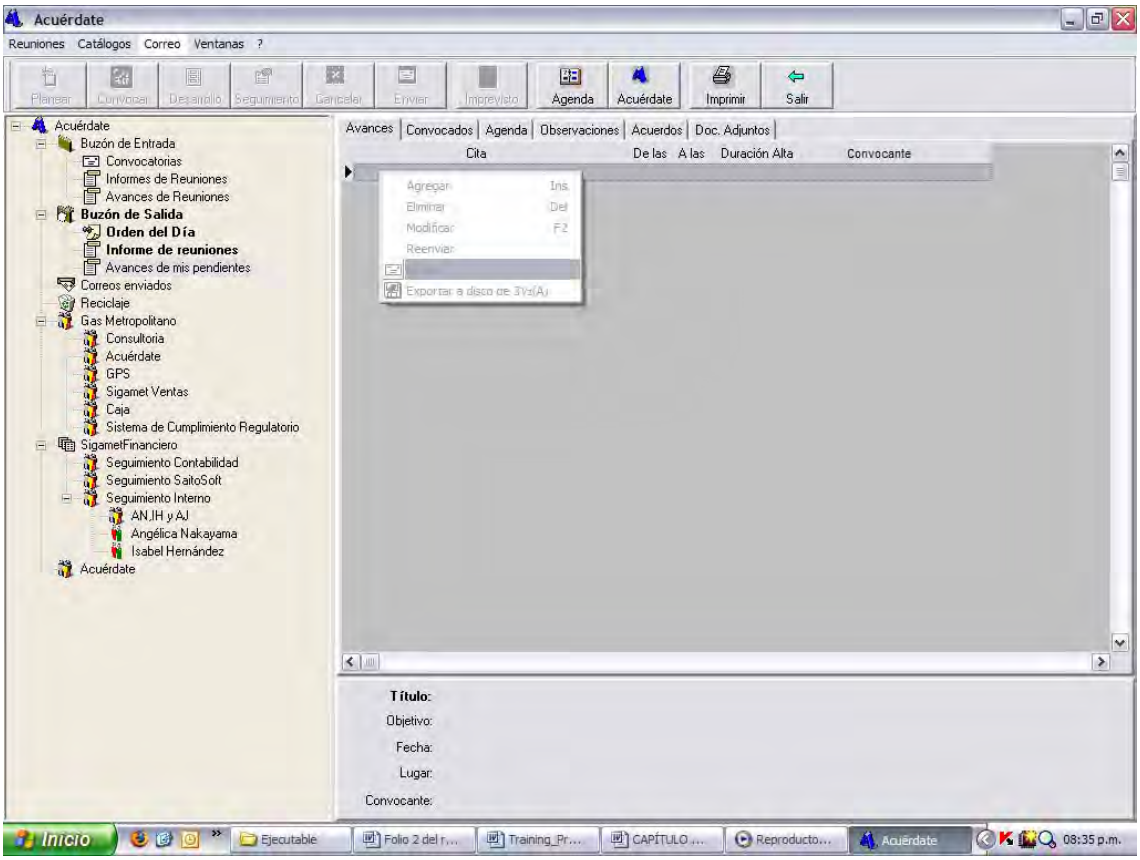
El registro de usuario y de todos nuestros datos se manejaran como información dentro del uso de Acuérdate se realiza desde el proceso de instalación del sistema.

Una vez que hayamos terminado de registrar todas nuestras actividades que forman parte del avance o termino de un acuerdo se sale de esta ventana con el botón salir. Al salir de esta ventana se remarcado en letras negritas el buzón de salida en Avance de mis pendientes.



5.46. Buzón de Avance de mis pendientes activo para enviar información.

A continuación lo que corresponde es enviar por correo electrónico el avance de los pendientes haciendo clic derecho sobre el registro de la reunión, y dar clic en la opción enviar.



5.47 Envío del avance de las actividades realizadas.

La información será enviada a las cuentas de correo de los participantes a la reunión de trabajo y principalmente al convocante o la persona que registra nuestros acuerdos, con la finalidad que pueda ver en su sistema Acuérdate una vez que descargue su correo con el mismo el avance realizado.

Finalmente una vez que la persona que realizó el registro de acuerdos revisa los avances, podrá si lo considera así pertinente cambiar el estatus del acuerdo de pendiente a terminado, si así lo considera permitente de acuerdo a lo descrito en el avance que ve registrado.

5.8 Cierre del registro de una reunión de trabajo

El cierre del registro de una reunión de trabajo se lleva acabo una vez que todos los acuerdos registrados en la misma se cumplen, o lo que es lo mismo el convocante da por terminados cada uno de ellos.

Los acuerdos de una reunión se pueden relacionar o incluir en una siguiente reunión en la etapa de planeación de la reunión para revisar en juntas de seguimiento el avance de los mismos, por lo tanto si se relacionan todos los acuerdos de una reunión a una nueva esta automáticamente se actualizará con el estatus de cerrada para dar lugar a una nueva reunión con estatus de seguimiento.

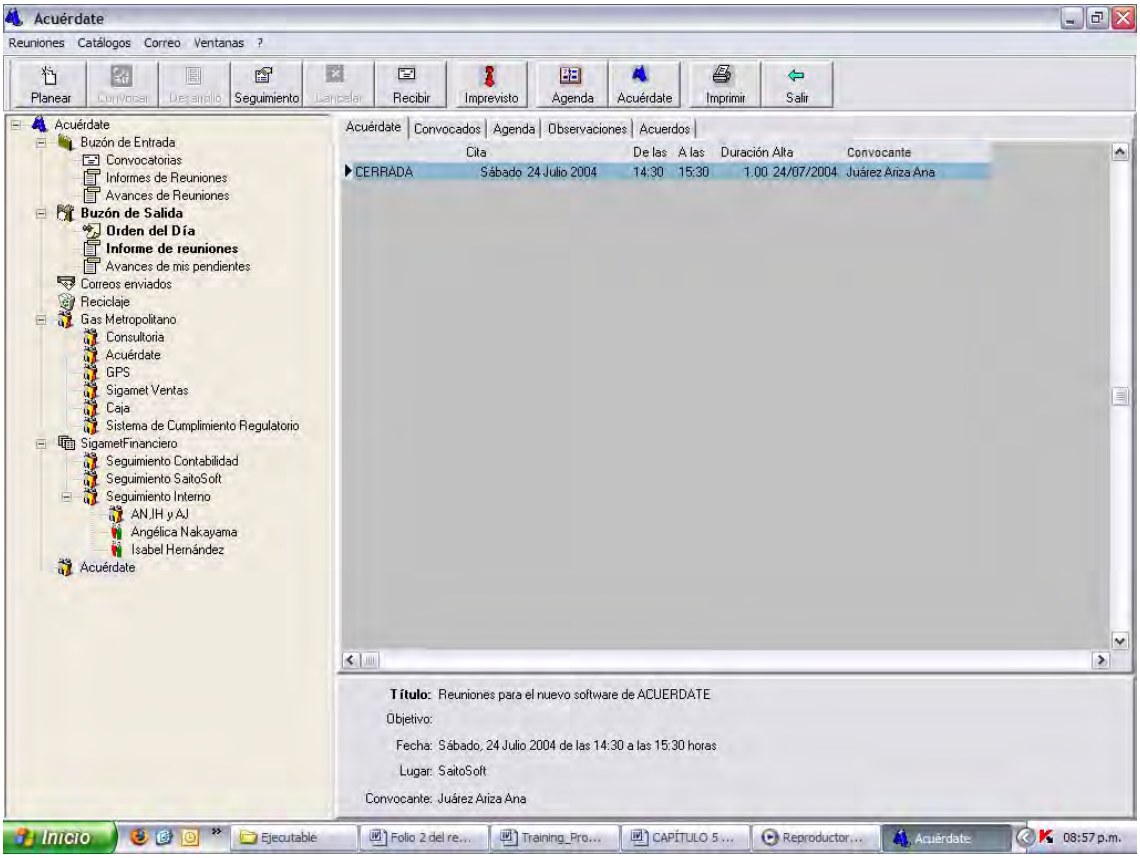


Figura 5.48 Cierre de una reunión de trabajo.

5.9 Exportación de información de una reunión

Si no se dispone de una cuenta de correo para realizar el envío de información de una reunión, el sistema Acuérdate ofrece la posibilidad de hacer la exportación a un medio magnético, como un disco floppy de 3 ½ pulgadas o una unidad de almacenamiento con conexión de bus serial universal (USB Universal Serial Bus). Es decir cualquier dispositivo de almacenamiento detectado por la computadora donde se encuentre instalado el sistema Acuérdate, podrá servir para realizar la exportación de información, para que posteriormente se realice el proceso de importación en la computadora destino.

La exportación de información de una reunión se realiza haciendo clic derecho sobre el registro de la reunión en el panel de información, al hacer esto aparecerá una ventana en donde en la parte inferior nos mostrará la opción de exportar a disco de 3 ½ (A), como lo muestra la figura 5.49

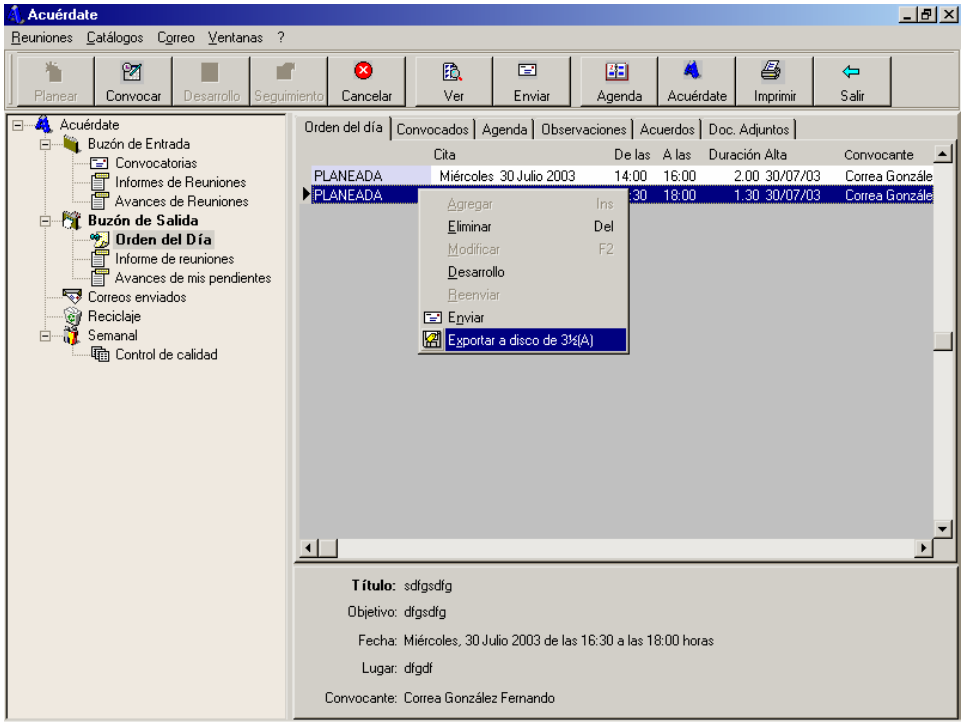


Figura 5.49 Exportación de la información de una reunión.

5.10 Importación de información

Posterior al proceso de exportación se realiza la importación del archivo generado por dicho proceso, para importar un archivo nos deberemos de ubicar en el árbol en el directorio de buzón de entrada, en el cual al hacer clic derecho nos mostrará una ventana con la opción de importar, como lo muestra la siguiente figura:

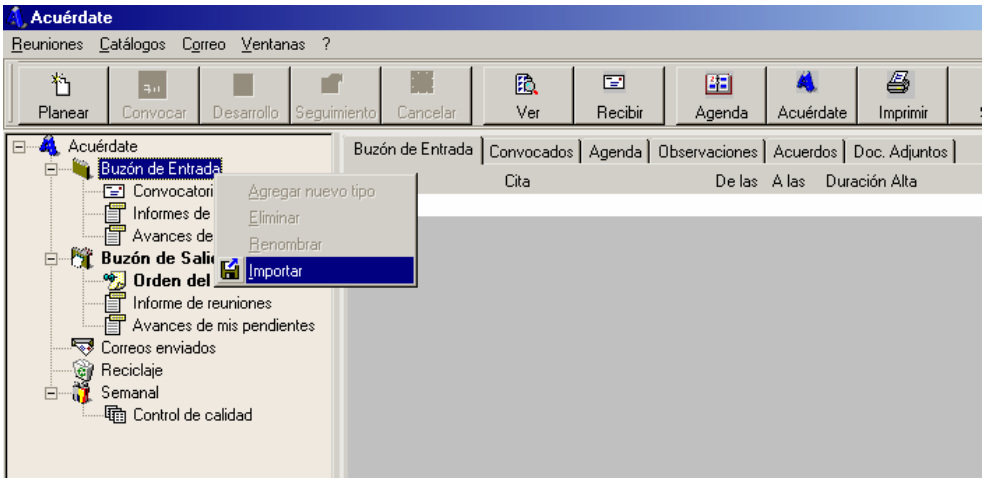
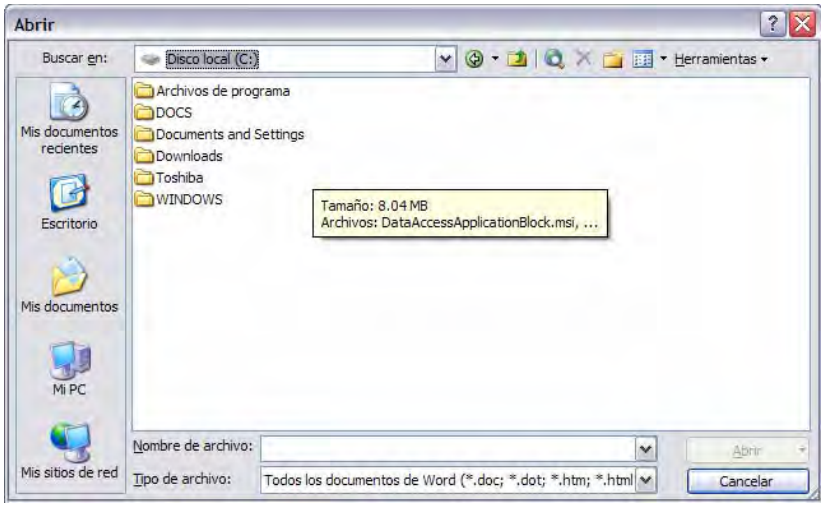


Figura 5.50 Importación de la información de una reunión.

Al hacer clic en el botón de importar aparecerá una ventana para seleccionar el archivo que deseamos importar.



5.51 Ventana de selección de archivo a importar.

5.11 Preferencias

Es posible configurar ciertas preferencias del sistema, con la finalidad de personalizar pantallas, rutas, mensajes de correo preestablecidos, entre otros. Para acceder a la parte de preferencias deberemos hacer clic en el menú de archivo del menú principal, en este aparece la opción preferencias como lo muestra la siguiente figura.

Incluir menú de preferencias.

Al dar clic en la opción de preferencias se mostrará la ventana de la figura 5.52

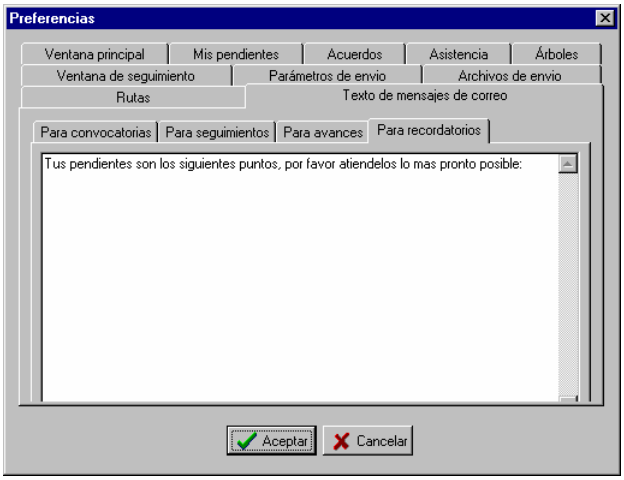


Figura 5.52 Ventana de configuración de preferencias.

Esta ventana nos permite configurar las siguientes opciones de ventanas y parámetros del sistema:

a) Configuración de color de fondo y tipo de letra en las siguientes ventanas:

- Ventana principal.
- Área de acuerdos en la ventana de desarrollo.
- Ventana de asistencia.
- Árbol de directorios.
- Ventana de mis pendientes.
- Ventana de seguimiento.

b) Parámetros de envío.

c) Archivos de envío.

d) Rutas de depósito y envío de información (archivos adjuntos, archivos de importación y exportación).

e) Plantillas de texto de mensajes de correo electrónico para convocatorias, seguimiento, avances y recordatorios.

CAPÍTULO 6 CONCLUSIONES Y RECOMENDACIONES

En este capítulo se presentan las conclusiones obtenidas con base en el objetivo plasmado en el capítulo 1, que es ***desarrollar una herramienta de software que reduzca el tiempo de registro y control de eventos, administre los acuerdos derivados de una reunión, y otorgue a las organizaciones una infraestructura en el cual depositar y representar el conocimiento que se genera de reuniones de trabajo con la finalidad de llevar una adecuada gestión de proyectos.***

6.1 Conclusiones

En el marco de esta tesis se ha propuesto realizar una metodología, que nos sirva como guía para identificar las etapas por las cuales atraviesa una reunión formal de trabajo y la información relevante que involucra cada una de estas.

Con toda la información obtenida de la investigación y documentación de los conceptos para lograr reuniones de trabajo efectivas y nuestra experiencia anterior, se desarrollo una herramienta de software que permitiera recolectar, almacenar y utilizar la información basándonos en la metodología propuesta.

Al concluir los trabajos de desarrollo de la herramienta y utilizarla en la práctica en proyectos con grupos de trabajo de tamaño considerable, nos pudimos dar cuenta de los beneficios que representó en comparación de los métodos tradicionales para guiar reuniones de trabajo y dar seguimiento a los acuerdos derivados de las mismas. Dentro de estos beneficios podemos destacar los siguientes:

- Reduce el tiempo de registro y consulta de información.
- Permite una mejor gestión de las actividades y por consecuencia de los proyectos.
- Incrementa la comunicación y colaboración entre los equipos de trabajo.
- Otorga a las organizaciones la infraestructura para administrar el conocimiento y experiencias que se generan.
- Define claramente los responsables y fechas compromisos de los acuerdos establecidos, eliminando las situaciones de duda o mala interpretación de los mismos.

Por consecuencia consideramos también que:

- Elimina los retrasos en los proyectos.
- Hace más eficiente la toma de decisiones por los líderes de proyectos.
- Mejora en la planeación.
- Mejora en coordinación.
- Reduce gastos en la reasignación de tareas.
- Permite obtener una memoria de comunicación formal.
- Facilita una celeridad en el flujo de información.
- Incremento en la productividad.
- Evita malos entendidos que se generan en el proceso de comunicación.

- Se tiene un mejor control debido a que la información se tiene organizada y clasificada sobre el cumplimiento de las actividades que lleva a desempeñar el papel de **Liderazgo**.

Por lo tanto podemos señalar que la herramienta cumple en buena medida con el objetivo del proyecto, además de ofrecer ventajas competitivas con software de su mismo tipo como se mostró en la tabla comparativa en el Capítulo 1 de Introducción, pudiendo competir como una mejores en el ramo.

6.2 Recomendaciones

Como trabajo a futuro nos proponemos desarrollar el sistema en una aplicación Web; con esto los beneficios y ventajas se incrementan, puesto que la arquitectura garantiza compartir recursos e información en toda la organización; el esquema cambiaría a centralizado, es decir, una base de datos central para todos los usuarios; en este tipo de arquitecturas el nivel de seguridad es primordial sobre todo porque se comparte información muy importante de la organización.

Una funcionalidad deseable a incluir es aquella que permita registrar el tiempo invertido estimado de la tarea que se ha realizado. El tiempo podría ser medido en días, horas o minutos dependiendo del caso en particular, de esta manera podríamos obtener estimadores aproximados para una futura planeación para tareas del mismo tipo.

Dentro del proceso operativo de una empresa, es importante medir la productividad del personal; el sistema puede contar con un esquema de incentivos o reconocimiento de desempeño, se puede lograr a partir de las tareas atendidas por cada uno de los miembros de la organización.

Otra posibilidad interesante que puede proponerse pensando en dispositivos móviles, es un sistema portátil que contenga las funcionalidades básicas de acceso y presentación de información, sería de gran apoyo en toma de decisiones en cualquier momento.

REFERENCIAS BIBLIOGRÁFICAS

[1] Ezequiel Ander-Egg, ¿Cómo hacer reuniones eficaces?, Lumen Humanitas Buenos Aires, 1998.

[2] Presman S. Roger, Ingeniería de software: Un enfoque práctico, McGraw-Hill / Interamericana de España, 1998.

[3] Steve McConnel, Code Complete Second Edition, Microsoft Press, Redmond Washington, United States, 2004.

[4] Mike Gunderloy, Coder to Developer, Sybex Publishers, California, United States, 2004.

[5] C.J. Date, Introducción a los sistemas de bases de datos, Addison-Wesley Iberoamericana, Wilmintong, Delaware, E.U.A., 1986

[6] Henry F. Korth, Abraham Silberschate, Fundamentos de bases de datos, MacGraw-Hill, Austin Texas, E.U.A, 1987.

[7] Gio Wiederhold , Diseño de base de datos, MacGraw-Hill, Standford, E.U.A., 1985.

[8] Marco Cantu, Mastering Delphi 5, Sybex, E.U.A., 1999.

[9] www.effectivemeetings.com, SMART Technologies Inc.

[10] www.meetingmaker.com, PeopleCube.