Universidad Tecnológica de la Mixteca

SISTEMA COLABORATIVO DE REUNIONES PARA PC'S Y DISPOSITIVOS MÓVILES

TESIS PARA OBTENER EL TÍTULO DE:
INGENIERO EN COMPUTACIÓN

PRESENTA:

VÍCTOR MOISES CANSECO SOTO

DIRECTOR DE TESIS:
M.C. GABRIEL GERÓNIMO CASTILLO

Huajuapan de León, Oax. Enero de 2004.

Contenido

Prólogo

La información se ha convertido en un recurso muy importante para las empresas, esto se puede apreciar en la gran cantidad de recursos que se invierten en tecnologías de información (TI) hoy en día. Un ejemplo de estas tecnologías son los sistemas colaborativos, este tipo de sistemas permiten que un grupo de personas trabajen de forma simultánea cada una en su lugar de trabajo pero compartiendo información. Existen en el mercado una gran cantidad de soluciones para este tipo de sistemas como son Lotus Domino de IBM y Documentum de Documentum Inc., este tipo de aplicaciones tienen el objetivo de reducir el tiempo y recursos utilizados por parte de la empresa para la realización de sus tareas cotidianas. Si bien estas aplicaciones reducen significativamente los costos y tiempos empleados para cada actividad, estas tienen un costo muy elevado.

Otra tecnología que esta teniendo una gran aceptación hoy en día son las tecnologías inalámbricas como son Bluetooth y WiFi (IEEE 802.11b), las cuales nos permiten acceder a la información disponible en paginas web o a través de una LAN (Local Area Network – Red de Area Local).

En el presente trabajo se propone un sistema colaborativo para PC's y dispositivos portátiles con el cual las personas puedan trabajar sin la necesidad de estar en su lugar de trabajo ya que si cuentan con algún dispositivo portátil como son los teléfonos celulares, Palm's o Pocket PC's también podrán hacer uso del sistema, empleando en los dispositivos móviles el lenguaje Java en su versión Micro Edition, el cual esta diseñado para dispositivos con menor poder de cómputo que una PC. Para poder realizar la conexión inalámbrica se utilizó Bluetooth ya que consume menos energía que WiFi, como se analizará mas adelante, y esto es uno de los problemas de los dispositivos móviles ya que estos no están conectados a una fuente continua de alimentación.

El objetivo principal de este trabajo es proveer un entorno de colaboración básico como son el soporte para reuniones grupales, conversaciones, leer y escribir e-mail; ya sea por medio de una PC o por medio de dispositivos móviles. Lamentablemente por el costo de las herramientas para el desarrollo de Bluetooth se utilizó un simulador de la compañía RococoSoftware, es por esto que éste trabajo sólo funciona como una simulación de un sistema real. Si se desea llevar la aplicación a algo real sólo se necesita comprar las APIs (Interfaz de Programación de Aplicaciones-Application Programming Interface) que produce esta compañía, compilar e instalar las aplicaciones en los dispositivos.

El capitulo 1 da una breve introducción al proyecto, los antecedentes y los objetivos del trabajo, y por último se habla un poco de los dispositivos PDA's (Asistente Personal Digital-Personal Digital Assistant). El capitulo 2 aborda el tema de los sistemas colaborativos, que son, en donde y para que se emplean. En el capitulo 3 se realiza el análisis y diseño del problema. El capitulo 4 describe brevemente como fue desarrollado el sistema y las pruebas realizadas al mismo. Finalmente se exponen los resultados obtenidos y el trabajo a futuro.

Capítulo 1 Panorama General

1

1.1 Introducción

Hoy en día la necesidad de información es muy grande y cuanto más extensa es una organización más compleja y confusa es la comunicación entre sus miembros. Es por esto que se crearon soluciones de información como son los sistemas colaborativos, este tipo de aplicaciones permiten a un grupo de trabajo realizar sus actividades en forma remota y en conjunto. Esto es posible gracias al uso de interfaces multiusuarios, videoconferencias, chats, correo electrónico, etc., este tipo de aplicaciones permiten a los usuarios expresar sus puntos de vista en tiempo real, además de permitir editar documentos de forma compartida.

Actualmente un gran número de organizaciones cuentan con una LAN para compartir su información, si estas empresas utilizarán un sistema colaborativo podrían maximizar el uso de sus recursos informáticos y mejorar el flujo de la información, en pocas palabras ser más productivas.

Una tecnología que está comenzando a ser empleada en vez de las LAN son las redes inalámbricas o WLAN (Red de Área Local Inalámbrica- Wireless Local Area Network) con este tipo de redes es posible realizar algún trabajo sin la necesidad de estar en el lugar acostumbrado de trabajo. Existen dos grandes soluciones de redes inalámbricas, la tecnología Bluetooth y WiFi (IEEE 802.11b). Bluetooth esta más orientado a pequeños dispositivos ya que consume menos voltaje que WiFi aunque esté tiene una menor tasa de transferencia de información (1Mbps contra 11Mbps (Mega bits por segundo)).

Este trabajo tiene como finalidad conjugar estas dos tecnologías (Sistemas colaborativos y redes inalámbricas) en un sistema de reuniones que permita a los usuario realizar sus actividades aunque no se encuentren en su sitio de trabajo.

A continuación hablaremos un poco de los antecedentes de este tipo de sistemas.

1.2 Antecedentes

Como ya se mencionó desde hace algunos años que se están desarrollando soluciones colaborativas para facilitar el flujo y manejo de la información de las organizaciones. Una solución propuesta es el sistema SISCO [URL 7], el cual es un sistema de reuniones electrónicas, que plantea diferentes escenarios de las reuniones como son: si la reunión se realiza diariamente, semanalmente o ocasionalmente; también se mencionan otros trabajos relacionados como son el sistema Prep [2] el cual es un editor de documentos asincrónico que puede ser usado por grupo y es más apropiado para los estados iniciales de los procesos de escritura tales como: tormenta de ideas, producción inicial de texto, comentario y revisión. El sistema SIBYL [12] que es un sistema para el soporte a las decisiones, para representar y administrar los aspectos cualitativos del proceso de construcción de decisiones.

Si bien los sistemas de reuniones son muy recientes y es por esto que no existen estándares para la implementación de los mismos sin embargo si estos son bien aplicados en una organización se pueden reducir costos y minimizar el tiempo para la realización de actividades.

1.3 Objetivos

El objetivo general de este proyecto es el realizar un sistema de reuniones para PC's y dispositivos móviles, proporcionando un ambiente de colaboración tanto para los usuarios de PC's como para los usuarios inalámbricos.

El sistema proporcionará los siguientes servicios:

- Chat
- Reuniones
    - Votaciones
- Correo Electrónico
    - Lectura
    - Escritura

Con estos servicios se proporciona un ambiente básico de colaboración permitiendo que los usuarios puedan realizar sus actividades básicas sin importar el lugar, o la distancia a la que se encuentren.

1.4 Justificación

Existen en el mercado diferentes estudios e implementaciones de sistemas de reuniones, cada uno con diferentes características como edición de documentos, lluvia de ideas, videoconferencias, etc., la funcionalidad de estos se encuentra limitada a una conexión LAN, es por esto que en el presente trabajo de tesis se desarrolla un sistema colaborativo que aparte de trabajar en redes LAN trabaja en dispositivos móviles.

Debido a que el sistema fue desarrollado en java es posible ejecutarlo casi en cualquier sistema operativo esto es porque la aplicación se ejecuta en una máquina virtual y no se compila código para una máquina en específico. Lo mismo sucede para los dispositivos Portátiles ya que se pueden ocupar diferentes marcas de teléfonos siempre y cuando tengan soporte para java, los principales fabricantes de teléfonos (Nokia, Motorola, Siemens, Sony-Ericsson) ya cuentan con dispositivos habilitados para java (Java Enable).

Para la realización de este trabajo se empleó un simulador de Bluetooth, esto fue a causa del elevado costo de las librerías de java para la programación de Bluetooth. Sin embargo para que el sistema sea una aplicación real sólo hay que comprar estas librerías compilar de nuevo el sistema e instalarlo en los dispositivos móviles que cuenten con la tecnología Java, estos pueden ser teléfonos celulares como son el Siemens S55 y M55, el Motorola i85 o en PDA's Palm como la Tungsten T/T2/T3/E.

A continuación daremos una breve introducción a los dispositivos móviles como son los PDA's, analizaremos y compararemos sus características básicas.

1.5 Introducción a los PDA's

Desde su aparición las computadoras personales revolucionaron la forma de trabajar facilitando varios aspectos que van desde la redacción de un simple documento hasta el almacenamiento e intercambio de información. El siguiente avance tecnológico que se dio fue crear computadoras portátiles casi igual de poderosas y funcionales que una computadora de escritorio lo cual facilito el trabajo de las personas que viajan constantemente ya que les permite realizar sus actividades sin estar necesariamente en su lugar de trabajo.

Debido a esta necesidad de desplazamiento se crearon dispositivo más pequeños que las computadoras portátiles y les llamaron PDA's, estos dispositivos son muy pequeños en comparación con las computadoras portátiles y pesan mucho menos, entre 150 y 300 gramos, y es por esto que son la mejor opción tecnológica disponible para personas que necesitan desplazarse constantemente y continuar trabajando en sus actividades cotidianas.

1.6 Dispositivos Móviles

Un PDA es un dispositivo móvil o computadora de bolsillo, cuentan con un microprocesador, memoria y un lápiz con el cual se ingresan los datos en forma escrita, algunos modelos poseen un puerto infrarrojo (IR port) que se emplea para comunicaciones entre dispositivos; es por esto que los PDA's guardan mayor similitud con las computadoras portátiles que con las Agendas Personales.

El primer PDA que salió al mercado fue el MessagePad 100 de Apple, este dispositivo salió a la venta en 1993 y contaba con el sistema operativo Newton, un sistema de reconocimiento de escritura manuscrita y en 1997 Apple dejo de producirla por la poca aceptación que tuvo en el mercado [13].

En marzo de 1996 salió a la venta la primer Palm Pilot con sus modelos 1000 y 5000 las cuales poseían 256Kb y 512Kb de memoria respectivamente, un año después lanzaron al mercado los modelos Pilot Personal y Pilot Professional los cuales contaban con 512Kb y 1Mb de memoria, además la Pilot Professional contaba con el protocolo de comunicación TCP/IP. Los PDA's de Palm cuentan con el sistema operativo PalmOS, Stylus (el Lápiz), puerto Infrarrojo y un Cradle para sincronizar el dispositivo con la PC [6].

A principios del año 2000 salieron a la venta los dispositivos Pocket PC y rápidamente empezaron a posicionarse en el mercado, como son los modelos iPaq de Compaq, Cassiopeia de Casio y los Jornada de HP, este tipo de dispositivos están basados en el sistema operativo de Microsoft, Windows CE [6].

Dado que existe una gran variedad de PDA's primero analizaremos los dispositivos Palm, luego los Pocket PC.

1.6.1 Dispositivos Palm

Estos PDA están basados en el sistema operativo de Palm Computing Inc., esta empresa es líder en el desarrollo y venta de PDA's. Comenzaron a fabricar y comercializar sus productos a partir del año de 1996, en este año sacaron al mercado su primer PDA el modelo Pilot 1000 y el Pilot 5000. Los dispositivos Palm además de contar con el sistema operativo PalmOS cuentan con el sistema de escritura Grafitti, el cual es un sistema de escritura muy parecido al alfabeto tradicional. Además cuenta con las siguientes aplicaciones: Agenda, Calculadora, Libreta de Direcciones y Memo Pad.

Estas aplicaciones son las más comunes ya que vienen instaladas de fábrica, los documentos creados en la Palm se pueden transferir fácilmente a la PC con sólo sincronizar la Palm y la PC. Existen además aplicaciones comerciales de todo tipo como hojas de cálculo, editores de texto, calculadoras científicas y juegos.

En el mercado existe una gran variedad de modelos de PDA's de Palm Computing Inc. en la tabla 1.1 se muestran algunos de los modelos disponibles en el mercado y algunas de sus características [URL 5]:

Tabla 1.1 Dispositivos Palm

MODELO	Características								
	S.O	Memoria	Puerto Infrarrojo	Medidas	Peso	Procesador	E-mail	Internet	Bluetooth
m125	Palm O.S 4.0	8 Mb	Si	4.82 in x 3.10 in x 0.87 in	5.31 oz.	33 MHz.	No	No	No
m500	Palm O.S 4.0	8 Mb	Si	4.5 in x 3.1 in x 0.4 in	4.0 oz.	33 MHz.	No	No	No
m515	Palm O.S 4.1	16 Mb	Si	4.5 in x 3.1 in x 0.5 in	4.9 oz.	33 MHz.	No	No	No
i705	Palm O.S 4.1	8 Mb	Si	3.06 in x 4.65 in x 0.61 in	5.9 oz.	33 MHz.	Si	Si	No
Tungsten T	Palm O.S 5.0	16Mb	Si	4.00 in x 3.00 in x 0.6 in	5.6 oz.	144 MHz.	No	No	Si
Tungsten T3	Palm O.S 5.2.1	64 Mb	Si	4.30 in x 3.0 in x 0.66 in	5.5 oz.	400 MHz.	No	No	Si

1.6.1.1 Dispositivos SONY

Estos dispositivos también están basados en el sistema operativo de Palm Computing Inc. por lo tanto también poseen el sistema de escritura Grafitti, y la compatibilidad de las aplicaciones que se desarrollan para la plataforma PalmOS. SONY ha lanzado al mercado sus PDA añadiéndole nuevas características como el sistema Jog Dial, el cual permite al usuario navegar mas fácilmente entre los menús de las aplicaciones además de manejar el scroll en algunas aplicaciones como por ejemplo en el Note Pad. En la tabla 1.2 se muestran algunos modelos disponibles en el mercado [URL 9].

Tabla 1.2 Dispositivos Sony

MODELO	Características								
	S.O	Memoria	Puerto Infrarrojo	Medidas	Peso	Procesador	E-mail	Internet	Bluetooth
PEG-SJ30	Palm O.S 4.1	16 Mb RAM 4 Mb ROM	Si	4.15 in x 2.9 in x 0.69 in	4.9 oz	33 MHz.	No	No	No
PEG-NR70V	Palm O.S 4.1	16 Mb RAM 6 Mb ROM	Si	5.5 in x 2.9 in x 0.69 in	7.0 oz	66 MHz.	No	No	No
PEG-TG50	Palm O.S 5.0	16 Mb RAM 6 Mb ROM	Si	2.79 in x 5.0 in x 0.50 in	6.2 oz	200 MHz.	No	No	Si
PEG-UX50	Palm O.S 5.0	104 Mb	Si	4.2 in x 3.5 in x 0.72 in	6.2 oz	230 MHz	Si	Si	Si

1.6.2 Dispositivos PocketPC

Además de los PDA basados en el sistema operativo PalmOS existen también PDA's que funcionan con el sistema operativo PocketPC desarrollado por Microsoft. Este tipo de PDA tienen un procesador mas veloz que los modelos Palm y también una cantidad mayor de memoria RAM sin embargo su sistema de reconocimiento de escritura no es tan bueno como el de Palm, a continuación veremos algunos modelos de Compaq y HP. Además incluyen diversos tipos de ranuras o slots de expansión, que permiten insertar tarjetas de diversos formatos (Multimedia, CompactFlash o PCMCIA) para aumentar memoria o incorporar módems, discos duros, tarjetas de red, etc., en la tabla 1.3 se muestran algunos modelos de Compaq y HP [URL 1][URL 3].

Tabla 1.3 Dispositivos Compaq y HP

MODELO	Características								
	S.O	Memoria	Puerto Infrarrojo	Medidas	Peso	Procesador	E-mail	Internet	Bluetooth
Compaq ipaq 3850	Windows Pocket PC 2002	64 Mb RAM 32 Mb ROM	Si	13.5 x 8.4 x 1.6 cm.	190 gr.	206 MHz	No	Si	No
Compaq Ipaq 3870	Windows Pocket PC 2002	64 Mb RAM 32 Mb ROM	Si	13.5 x 8.4 x 1.6 cm.	190 gr.	206 MHz	Si	Si	No
HP Jornada 525	Windows Pocket PC 2002	16 Mb RAM 16 Mb ROM	Si	13 x 7.8 x 1.7 cm.	230 gr.	200 MHz	Si	Si	No
HP Jornada 568	Windows Pocket PC 2002	64 Mb RAM 32 Mb ROM	Si	13.2 x 7.65 x 1.72 cm.	173 gr.	206 MHz	Si	Si	No
HP iPaq h1935	Windows Pocket PC 2003	64 Mb RAM 56 Mb ROM	Si	4.46 x 2.75 x 0.5 cm	124 gr	203 MHz	Si	Si	Si

Hasta aquí hemos analizado las características generales de un PDA y los diferentes modelos disponibles en el mercado, a continuación se dará un breve repaso a los protocolos de comunicación que existen para los PDA's como son el IEEE 802.11, Bluetooth y WAP, se comenzará analizando el protocolo IEEE 802.11.

1.7 Protocolos de Comunicación

Las redes de computadoras permiten compartir información y otros recursos como impresoras y medios de almacenamiento, pero se tenía que estar conectado por medio de un cable hacia una terminal, esto significaba un problema para las personas que tenían que desplazarse de un lado a otro, fue entonces que se empezó a investigar en otro tipo de conexiones como son las inalámbricas, este tipo de tecnología se empezó a utilizar en computadoras portátiles y de ahí dio el salto hacia los PDA's. Las tecnologías inalámbricas más comúnmente utilizadas en el mercado son el estándar IEEE 802.11 y Bluetooth, se iniciará analizando el estándar IEEE 802.11.

1.7.1 IEEE 802.11

En el año de 1997 se concluyo el estándar IEEE 802.11, el cual contiene las especificaciones físicas de hardware y de software para el desarrollo de aplicaciones. Este protocolo a sufrido diferentes modificaciones para mejorar su tasa de transferencia de información como se muestra en la tabla 1.4 [16].

Tabla 1.4 Características del protocolo IEEE. 802.11

Especificación	Estado	Máxima tasa de bits	Frecuencia de operación
IEEE 802.11	Utilizado por la mayoría de fabricantes de WLANs	2 Mbps	2.4 GHz
IEEE 802.11b	Especificación actual	11 Mbps	2.4 GHz
IEEE 802.11a	Especificación actual	24 – 54 Mbps	5.0 GHz
IEEE 802.11g	En desarrollo	54 Mbps	2.4 GHz

El IEEE 802.11 provee dos variaciones de la capa física. Estos incluyen dos tecnologías llamadas DSSS (Espectro Amplio de Secuencia Directa-Direct Sequence Spread Spectrum) y FHSS (Espectro Amplio de Saltos de Frecuencia-Frequency Hopped Spread Spectrum). Las opciones de DSSS y FHSS PHY fueron diseñadas específicamente para estar en conformidad con las reglas de las regulaciones del FCC (Comisión Federal de Comunicaciones-Federal Communications Commission) para las operación en la banda de 2.4 GHz [3].

Los autores del estándar 802.11 le dieron a la tecnología la posibilidad de utilizar la tecnología inalámbrica para video, voz, datos, sistema de distribución, y LANs que utilizan diferentes direcciones. El IEEE 802.11 solamente especifica la dirección para el medio inalámbrico, aunque fue específicamente hecho para facilitar la integración con otros estándares como es el IEEE 802.3 alámbrico. El IEEE 802.11 de 48-bit es un esquema compatible con los estándares del IEEE 802.

El IEEE 802.11 provee seguridad en dos formas: autenticación y criptografía. La Autenticación es la forma por la cual una estación es verificada para tener autorización para comunicarse con una segunda estación en un área de cobertura.

La autenticación puede ser un sistema abierto o de llave compartida. En un sistema abierto, cualquier estación puede solicitar la autenticación. La estación que recibe la solicitud puede otorgar la autenticación a cualquier solicitud, o solamente a estaciones que se encuentran en la lista de usuarios definidos. En un sistema de llave compartida, solamente las estaciones que posen una llave secreta criptográfica pueden ser autenticadas. La autenticación de llave compartida esta disponible

exclusivamente a los sistemas que tengan la capacidad de criptografía como una opción. La criptografía fue elaborada para proveer un nivel de seguridad comparable al que se provee en una LAN alámbrica. La WEP (Privacidad Equivalente al Alámbrico–Wired Equivalent Privacy) tiene características que utiliza el algoritmo RC4 de la empresa RSA Data Security, Inc. Este algoritmo de WEP fue elegido por tener el siguiente criterio: Razonablemente fuerte, Sincronizador propio, Exportable y Opcional.

El IEEE802.11g opera en la banda de los 2.4 GHz y cuenta con una tasa de transferencia de 54 Mbps. Por lo que los casos de uso, son los mismos que en el 802.11b, aunque también ofrece nuevas aplicaciones debido a la tasa binaria como son la mejora de las aplicaciones existentes.

Para la comunicación entre los dispositivos por medio del protocolo 802.11 existen 3 posibles configuraciones de conexión: la comunicación punto a punto, modo infraestructura y el enlace entre varias LAN o WLAN. La primera forma de conexión se establece directamente entre dos dispositivos que se encuentran dentro del rango de cobertura de sus respectivas antenas. Para la siguiente forma de conexión es necesario un punto de acceso (Access Point), el cual se puede conectar a otros puntos de acceso, o una red LAN, permitiendo al usuario acceder a una mayor cantidad de recursos. La ultima forma de conexión se logra haciendo uso de antenas direccionales para poder enlazar redes que se encuentran en diferentes sitios, como por ejemplo para comunicar dos edificios.

1.7.2 Bluetooth

En 1994 Ericsson Mobile Communications comenzó a examinar alternativas para la conexión de teléfonos móviles y sus accesorios, ellos utilizaron conexiones de radio porque las conexiones son unidireccionales y esto es una ventaja sobre las conexiones por puerto infrarrojo utilizadas por los PDA's y otros dispositivos. En 1999 salió la versión 1.0 de la especificación de Bluetooth [3].

En 1998 se fundó el Grupo de Interés Especial de Bluetooth el cual estaba conformado por las siguientes compañías:

- Ericsson Mobile Communications AB.
- Intel Corp.
- IBM Corp.
- Toshiba Corp.
- Nokia Mobile Phones.

Con la salida de la especificación de Bluetooth se unieron a este grupo las siguientes compañías:

- Microsoft.
- Lucent.
- 3Com.
- Motorola.

La tecnología Bluetooth no necesita un cable físico para comunicarse con otros dispositivos habilitados con Bluetooth. Bluetooth opera a 1Mbps. Aunque es más lento que el IEEE 802.11b, Bluetooth es más apropiado para dispositivos móviles; ya que minimiza el uso de la batería y el chip es mucho más pequeño, otra ventaja de Bluetooth sobre el IEEE 802.11b es que ya se está utilizando en teléfonos celulares. Ericsson, Nokia y Siemens ya venden al mercado teléfonos con esta tecnología.

Bluetooth provee dos niveles de acceso a los servicios: dispositivo confiable y no confiable. El dispositivo confiable tiene acceso sin restricción sobre los recursos, mientras que el no confiable no tiene una relación fija y por lo tanto puede utilizar los recursos limitadamente. Se provee de tres niveles de seguridad:

- Nivel 1. No existe seguridad y cualquier dispositivo que se conecte puede tener acceso sin alguna restricción.
- Nivel 2. Seguridad en las aplicaciones. Se establecen procedimientos de seguridad para las aplicaciones que se están comunicando.
- Nivel 3. Seguridad en la conexión. Se inician procedimientos de seguridad cuando otro dispositivo intenta conectarse a la red.

Para la comunicación entre los dispositivos se organizan en grupos de 2 a 8 dispositivos, estos grupos se llaman picoceldas o picoredes, al crearse la red una unidad actuará como maestra y el resto como esclavas mientras dure la conexión. Un dispositivo puede pertenecer a más de una picocelda y comportarse como un esclavo en ambas o un maestro en una picocelda y como esclavo en otra.

1.7.3 Bluetooth vs. IEEE 802.11

En la tabla 1.5 se muestra una comparativa entre estas dos tecnologías inalámbricas.

Tabla 1.5 Comparación entre Bluetooth e IEEE 802.11

Características y funciones	Bluetooth	IEEE 802.11		
		b	a	g
Tipo de conexión	Amplitud de espectro	Amplitud de espectro	Amplitud de espectro	Amplitud de espectro
Espectro	2.4 GHz	2.4 GHz	5.2 GHz	2.4 GHz
Velocidad de transferencia	1 Mbps	11 Mbps	54 Mbps	54 Mbps
Poder de transferencia	1 mW	100 mW	200 mW	100 mW
Rango	10 metros	100 metros	100 metros	100 metros
Dispositivos soportados	8			
Canales de voz	3	VOIP	VOIP	VOIP
Direccionamiento	48 bits MAC	48 bits MAC	48 bits MAC	48 bits MAC
Tipo de terminal	Teléfonos celulares, PC's notebooks, PDA's, Localizadores, Impresoras.	PC's, notebooks, PDA's.	PC's, notebooks, PDA's.	PC's, notebooks, PDA's.

Como podemos apreciar en la tabla 1.5 Bluetooth e IEEE 802.11 comparten algunas características como son el tipo de conexión, el espectro y el tipo de direccionamiento. Debido al bajo consumo de energía, Bluetooth se está empleando en más dispositivos de oficina que el IEEE 802.11. En la figura 1.1 se muestra una comparación de estas tecnologías de acuerdo al modelo OSI.

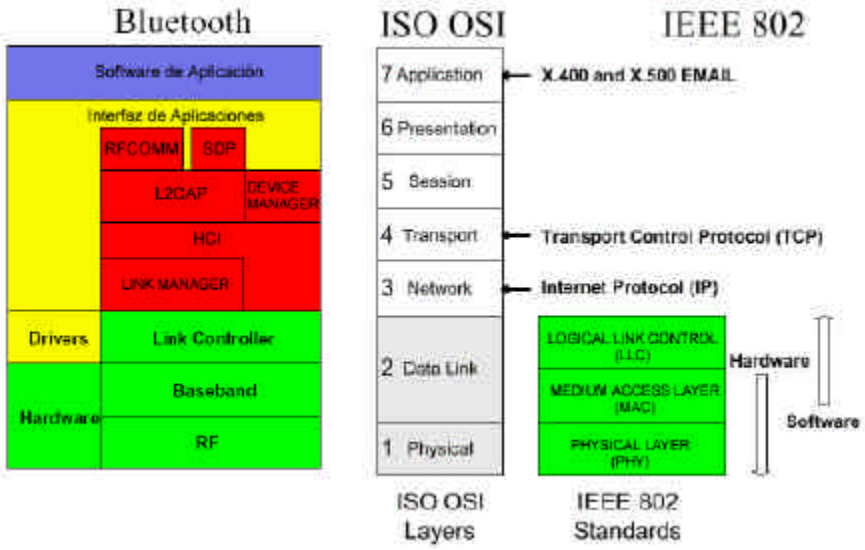

Figura 1.1 Comparación en el modelo OSI

HCI- Interfaz de Control del Anfitrión-Host Controller Interface.
L2CAP- Protocolo de Control de Enlace Lógico y Adaptación-Logical Link Control and Adaptation Protocol.
SDP- Protocolo de Servicio de Descubrimiento-Service Discovery Protocol.
RFCOMM- Puerto serial inalámbrico.
RF- Radio Frecuencia.

Para el desarrollo del sistema de reuniones se trabajo únicamente en la capa de aplicación del protocolo Bluetooth, en la figura 1.2 se muestra la arquitectura de java con Bluetooth.

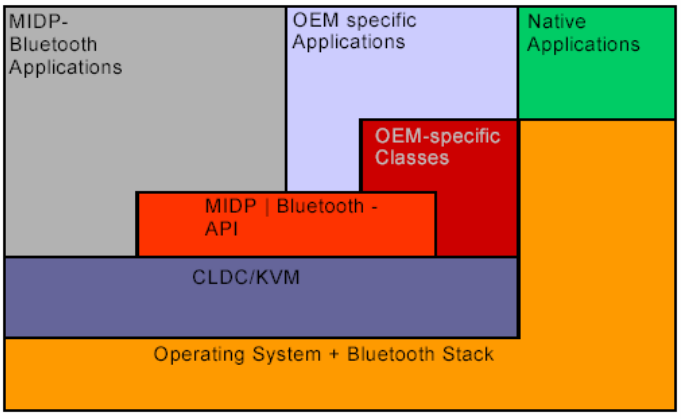

Figura 1.2 Arquitectura de Java para Bluetooth

El sistema colaborativo de reuniones desarrollado se encuentra la capa de MIDP (Perfil de Dispositivo Móvil de Información-Mobile Information Device Profile), el cual es un ambiente de ejecución de java para dispositivos móviles como son los teléfonos celulares, PDA, etc., las clases empleadas para la comunicación con otros dispositivos trabajan en la capa de MIDP | Bluetooth API, en esta capa las clases se comunican a través de la CLDC/KVM (Configuración de Dispositivo Conectado Limitado-Connected Limited Device Configuration, Maquina Virtual de 1 Kb- K Virtual Maquine). La

KVM es la encargada de ejecutar las aplicaciones en diferentes dispositivos además de comunicarse directamente con el sistema operativo. En la figura 1.2 se muestra como se comunican las aplicaciones OEM (Manufacturador de Equipo Original-Original Equipment Manufacturer), este tipo de aplicaciones son desarrolladas por algún fabricante de software y por último las aplicaciones nativas, las cuales se comunican directamente con el sistema operativo ya que fueron desarrolladas por el fabricante del dispositivo.

1.7.4 Protocolo de Aplicaciones Inalámbricas

El WAP (Protocolo de Aplicaciones Inalámbricas-Wireless Application Protocol), permite la comunicación inalámbrica de un dispositivo móvil equipado con un micronavegador y un gateway conectado a Internet. Es un protocolo creado para acceder a Internet desde los teléfonos celulares. El protocolo incluye especificaciones para las capas de sesión y de transporte del modelo OSI, así como funcionalidades de seguridad. WAP también define un entorno de aplicaciones [15]. Es escalable, permitiendo así a las aplicaciones disponer de las capacidades de pantalla y recursos de red según su necesidad y en una gran variedad de tipos de terminales. Esta tecnología fue desarrollada y promovida por diversos fabricantes de dispositivos móviles como son: Ericsson, Nokia y Motorola, los cuales fundaron en 1997 el WAPForum, una organización en la cual hoy en día participan más de doscientas empresas de todo el mundo.

WAP se basa en una arquitectura definida para el World Wide Web (WWW), pero adaptada a los nuevos requisitos del sistema. Dado que un servidor Web de Internet convencional no es capaz de dialogar con un dispositivo móvil, se necesita la presencia de un gateway WAP para que el teléfono celular pueda recuperar la información almacenada en el servidor.

Con este protocolo se accede a los contenidos WAP que se depositan en servidores WEB convencionales, aprovechando la infraestructura de Internet que ya existe. Es importante aclarar que los contenidos a los que se accede deben estar diseñados y creados para poder ser interpretados por los dispositivos WAP, la información debe suministrarse por los servidores WEB en formato WML y no en HTML. Para conseguir consistencia en la comunicación entre el dispositivo móvil y los servidores de red que proporcionan la información, WAP define un conjunto de componentes estándar:

- Un modelo de nombres estándar. Se utilizan las URLs definidas en WWW para identificar los recursos locales del dispositivo (tales como funciones de control de llamada) y las URLs (también definidas en el WWW) para identificar el contenido WAP en los servidores de información.
- Un formato de contenido estándar, basado en la tecnología WWW.
- Protocolos de comunicación estándares, que permitan la comunicación del micronavegador del dispositivo móvil con el servidor Web en red.

Capítulo 2 Introducción a los Sistemas Colaborativos

2.1 Introducción

A partir de la llegada de las computadoras personales al ambiente empresarial se inició una nueva revolución. Uno de los cambios más drásticos fue el incremento en la velocidad de procesamiento de los datos.

Desde los años 60s se tenía la necesidad de compartir recursos de cómputo, como la memoria, las unidades de almacenamiento y principalmente el procesador; pero no se compartía la información, ya que existían diferentes barreras que lo impedían, como las distancias entre oficinas o los diferentes sistemas operativos. Con la llegada de Internet algunas barreras se rompieron, con este avance tecnológico se logró compartir información pero no se podían realizar las actividades que necesitan colaboración, a partir de esta necesidad se comenzó a trabajar en un nuevo tipo de tecnología: el software colaborativo.

A los sistemas que permiten colaboración se les llama Groupware, un subconjunto de estos sistemas que ayudan a la administración y automatización de procesos de negocios se les conoce como Workflow. El sistema desarrollado en este trabajo de tesis cae en la clasificación de los sistemas colaborativos, a continuación hablaremos de los Groupware y analizaremos los sistemas Workflow.

2.2 Groupware

El Groupware es un tipo de software colaborativo que ayuda a grupos de trabajo a realizar sus actividades a través de una red. Formalmente se puede definir al groupware de la siguiente manera:

> "Sistemas basados en computadoras que apoyan a grupos de personas que trabajan en una tarea común y que proveen una interfaz para un ambiente compartido"
> ~Dave Chaffney[4]

Las características más importantes de los groupware son:

- Proveer de un ambiente de colaboración, en el que realmente se perciba que el trabajo en grupo se lleva a cabo.
- Mantener la información en un sitio común para todos los miembros.
- Interactuar con otros usuarios, de forma escrita, voz o video.

Los groupware se pueden clasificar en base a: tiempo y espacio. En base al tiempo se clasifican en sincrónicos y asincrónicos; y en base al espacio, pueden estar en el mismo lugar o en forma distribuida. Las aplicaciones típicas de los groupware sincrónicos (los cuales soportan aplicaciones en tiempo real) son: pizarrones compartidos, teleconferencia, chat y sistemas de toma de decisiones. Algunos ejemplos de

aplicaciones típicas de los groupware asincrónicos son: e-mail, newsgroups, calendarios y sistemas de escritura colaboracionales [4].

Los groupware se están volviendo más populares dentro de las empresas, ya que resulta más barato instalar una Intranet y comprar o implementar un sistema de colaboración a estar transportando personal de un lugar a otro. Además si se necesita tomar una decisión urgente y las personas se encuentran en diferentes partes del mundo, para cuando se reúnan la decisión posiblemente ya no funcione, o peor aún que la empresa quiebre; con los groupware esto no pasaría, ya que se pueden tomar decisiones sin importar la distancia entre cada miembro del equipo.

Es por esto que los groupware deben proporcionar tres funciones esenciales dentro de un grupo, llamadas las tres C's:

- La Comunicación, es la función más importante del groupware, ya que es el medio en que la información es compartida.
- La Colaboración, utilizada para unir la cooperación y resolver problemas de negocios o alguna actividad empresarial. Proporciona la ventaja de resolver problemas de las asambleas tradicionales como: lugar y tiempo para la realización de la misma o la disponibilidad de información. Además de mejorar la eficiencia en la toma de decisiones con la contribución de todos los miembros del grupo.
- La Coordinación, es la acción de asegurar que el equipo esta trabajando eficientemente y en conjunto para alcanzar una meta. Esto incluye la distribución de tareas y revisión de su ejecución.

Al unir estas tres características dentro del groupware la información fluye mas rápidamente, y con precisión, existen menos barreras entre cada departamento, se mejora la toma de decisiones y sobre todo se mejora el servicio al cliente.

Como se puede ver un groupware tiene características que lo hacen una gran inversión para los negocios, se pueden crear grupos de discusión, compartir documentos, realizar videoconferencias, etc. Para realizar todo esto es necesario contar con una Intranet o una conexión a Internet y poder comunicar las máquinas clientes con el servidor de las aplicaciones groupware.

Es muy confuso distinguir entre groupware y workflow, esto surge desde que los workflow´s son considerados como una función o un subconjunto de los groupware. Una definición estricta dice que todos los tipos de groupware deben incluir un elemento de colaboración, pero esto no es necesario en los sistemas workflow, algunas veces son utilizados para tareas individuales que no están directamente en colaboración.

2.3 Workflow

Los Workflows son sistemas que ayudan a administrar y automatizar procesos de negocios. Un workflow puede ser descrito como el flujo y control en un proceso de negocio.

La WfMC (Coalición de Administración de Workflow-Workflow Management Coalition) define a los workflows como [URL 11]:

"La automatización de un proceso de negocio, total o parcial, en la cual documentos, información o tareas son pasadas de un participante a otro para efectos de su procesamiento, de acuerdo a un conjunto de reglas establecidas."

También definen lo que es un proceso de negocio:

"Es un conjunto de uno o mas procedimientos o actividades directamente ligadas, que colectivamente realizan un objetivo del negocio, normalmente dentro del contexto de una estructura organizacional que define roles funcionales y relaciones entre los mismos."

Entre los ejemplos de proceso de negocios tenemos: procesamiento de órdenes, reportes de gastos, procedimientos de producción, etc.

Cabe mencionar que los workflows son sólo un camino para la información, que reducen tiempo, dinero y esfuerzo en la ejecución de un proceso de negocio. Las funciones más comunes que proporcionan los workflows son:

- Asignación de tareas al personal.
- Aviso al personal de tareas pendientes.
- Permitir la colaboración en las tareas comunes.
- Optimización de recursos humanos y técnicos, alineándolos a la estrategia de la empresa.
- Automatización de las secuencias de los procesos de negocio y optimización de las mismas.
- Agilización de los procesos de negocio y como resultado un mejor servicio al cliente.
- Control y seguimiento de dichos procesos.

Como se mencionó anteriormente, un workflow es el control del flujo de información en un proceso de negocio. Para poder identificar cada elemento dentro de cada workflow se puede utilizar el modelo de componentes de proceso de negocio. En la figura 2.1, se puede observar los elementos que forman a un proceso [URL 2].

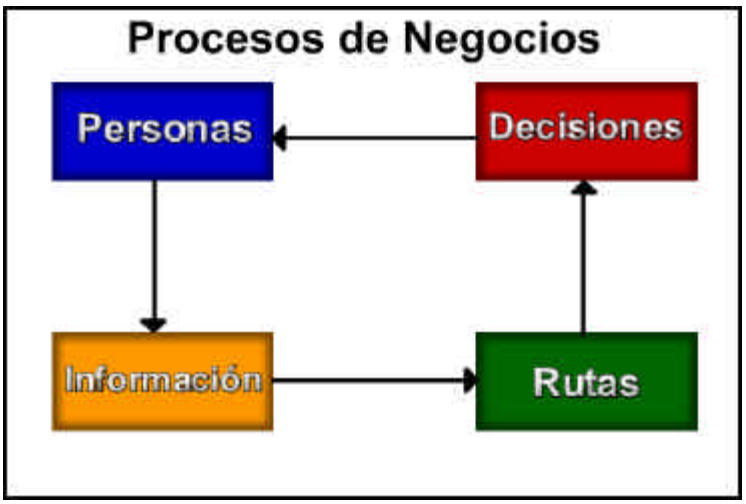

Figura 2.1 Elementos clave de un proceso de negocio.

Estos cuatro elementos clave forman parte de los componentes de un proceso de negocios y por lo tanto de un workflow. Para identificar estos componentes clave dentro de un proceso es necesario formularse las siguientes preguntas: ¿Qué rutas se siguen?, ¿Qué gente participa?, ¿Cuál es el rol que juega cada participante?, ¿Qué decisiones son tomadas?, ¿Cómo se llevan a cabo estas decisiones?, ¿Qué información es requerida por cada participante?. Estas preguntas son indispensables para poder identificar correctamente los procesos de negocio que pueden ser mejorados e implementados a través de un workflow [14].

A continuaciones hablará brevemente del modelo de referencia de workflow. El modelo de referencia de workflow [URL 11] mostrado en la figura 2.2 fue desarrollado por la WfMC para tener una estructura genérica en el desarrollo de aplicaciones de workflows, es decir, un estándar.

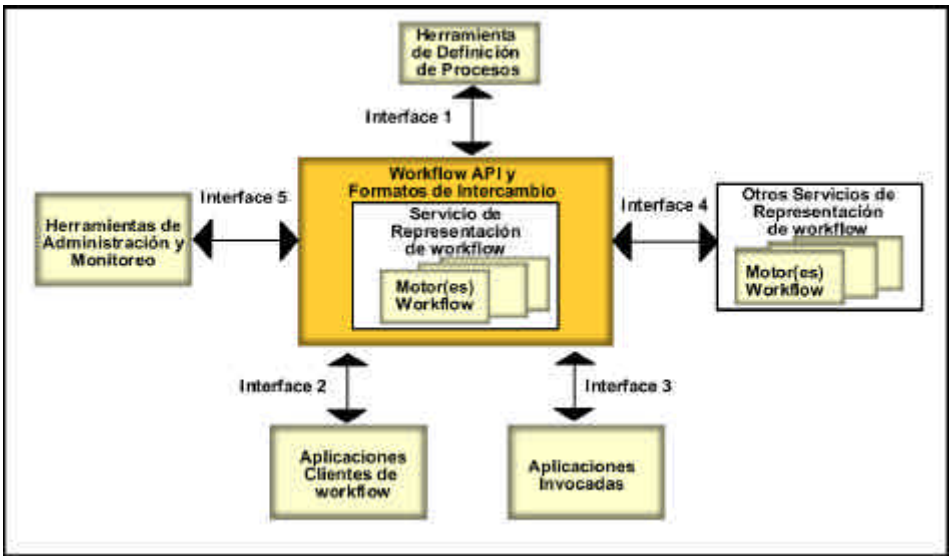

Figura 2.2 Modelo de referencia de Workflow– componentes e interfaces

El modelo de referencia de Workflow fue desarrollado a partir de estructuras genéricas de aplicaciones de Workflow, identificando las interfaces con estas estructuras, de forma que permita a los productos comunicarse a distintos niveles. Todos los sistemas de Workflow contienen componentes genéricos que interactúan de forma definida. Para poder tener cierto nivel de interoperabilidad entre los diversos productos de Workflow, es necesario definir un conjunto de interfaces y formatos para el intercambio de datos entre dichos componentes.

2.4 Ventajas del uso de workflows

La automatización de los procesos de negocio de una empresa trae grandes beneficios como la reducción del tiempo de búsqueda de papeles o el menor gasto en papelería, estos problemas son los primeros que se atacaron con la tecnología de workflows. A continuación conoceremos algunas razones por las cuales las organizaciones podrían considerar adoptar una solución de workflow.

- Eficiencia en los procesos y estandarización de los mismos. Esto conduce a:
  - Una reducción de costos dentro de una empresa.
  - La estandarización de los procesos lleva a tener un mayor conocimiento de los mismos, lo que a su vez conduce a obtener una mejor calidad de estos.

    - Administración de los Procesos. Utilizando la tecnología de Workflow es posible monitorear el estado actual de las tareas así como también observar como evolucionan los planes de trabajo realizados. Permite ver cuales son los embotellamientos dentro del sistema, es decir aquellas tareas o decisiones que están requiriendo de tiempo no planificado y se tornan en tareas o decisiones críticas.
- Asignación de tareas a la gente. La asignación de tareas se realiza mediante la definición de roles dentro de la empresa, eliminando la tediosa tarea de asignar los trabajos caso por caso.
- Recursos disponibles. Se asegura que los recursos de información (aplicaciones y datos) van a estar disponibles para los trabajadores cuando ellos los requieran.
- Diseño de procesos. Se fomenta a pensar los procesos de una manera distinta a la tradicional forma jerárquica que se utiliza para diseñarlos en la actualidad.

Hay además muchos aspectos operacionales por los cuales es deseable contar con una tecnología de Workflow ya que aspectos como la secuencia de tareas, quienes realizan dicha secuencia, los mecanismos de control y monitoreo, son implementadas en el software de Workflow.

El Workflow permite automatizar diferentes aspectos del flujo de la información: rutear los trabajos en la secuencia correcta, proveer acceso a datos y documentos, y manejar ciertos aspectos de la ejecución de un proceso.

La diversidad de procesos que puede haber en una organización lleva a pensar en la existencia de diferentes tipos de software de Workflow. El Workflow entonces, ofrece a una empresa la posibilidad de automatizar sus procesos, reducir costos, y mejorar servicios. Parece ser obvio que son grandes beneficios. Organizaciones que no hayan evaluado esta tecnología podrían encontrarse con desventajas en un futuro.

2.5 Workflow como herramienta de Reingeniería

¿Qué potencialidad tiene la reingeniería del negocio si además se utiliza Workflow?. La respuesta a esta interrogante es inmediata si se conocen algunos principios que la reingeniería propone:

- Combinación de tareas desarrollándose en el momento adecuado y donde tienen más sentido.
- Reducción de tiempos, verificaciones y controles.
- Disminución de niveles jerárquicos. Esto lleva a la ejecución de los procesos en el orden natural.
- Las tareas se conviertan en procesos.

Por su parte, el Workflow ofrece:

- Integración entre personas, actividades, programas y datos.
- Optimización de recursos humanos y técnicos, alineándolos con la estrategia del negocio.
- Eliminación de partes innecesarias en la secuencia de los procesos y la automatización de dicha secuencia.

Se podrían seguir enumerando elementos, pero la idea es simplemente mostrar que el Workflow es estratégico en cualquier proceso de reingeniería.

Capítulo 3 Análisis y Diseño del Sistema

3.1 Introducción

Para la realización de cualquier proyecto de software es muy importante seguir una metodología de desarrollo, en la actualidad existen diferentes métodos de desarrollo de software como son: el método de cascada, el método en espiral, etc. Para el desarrollo de este sistema se emplea el método Iterativo propuesto por Rational Software, los pasos de esta metodología se muestran en la figura 3.1, cabe mencionar que el método consiste de un leguaje y un proceso de modelado. El lenguaje de modelado y el proceso que se emplea es el UML (Lenguaje de Modelado Unificado-Unified Modeling Language) y el RUP (Proceso Unificado Racional-Rational Unified Process).

Figura 3.1 Pasos del Método Iterativo

UML es un lenguaje de modelado visual propuesto en 1996 por Grady Booch, Jim Rumbaugh e Ivar Jacobson y se utiliza para analizar y diseñar aplicaciones, ayudando a reducir la complejidad y captura los procesos desde la perspectiva del usuario. UML define una notación que es empleada por los métodos para representar los diseños. UML es un estándar del OMG (Grupo de Administración de Objetos-Object Management Group) a partir de 1997 y es empleado para la especificación, construcción, visualización y documentación de los sistemas de software [7].

"Un modelo es una descripción completa de un sistema desde una perspectiva particular"
-Rational Software [URL 6].

En la figura 3.2 se muestran los elementos que componen un modelo dentro de UML.

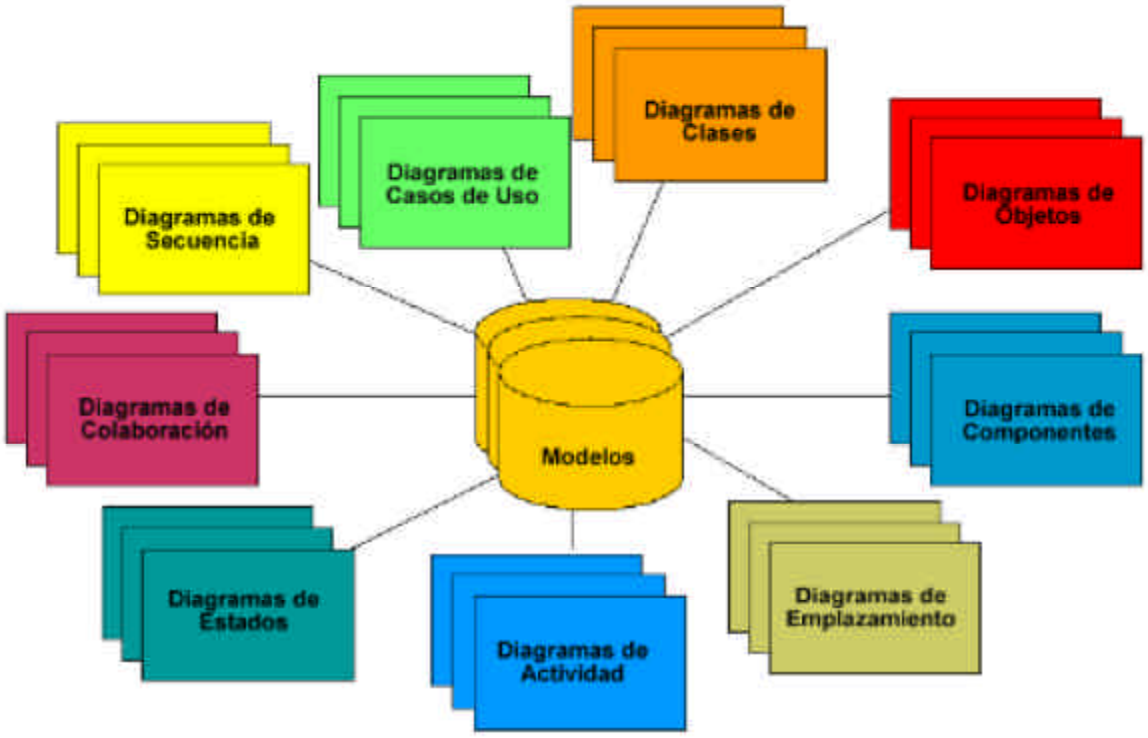

Figura 3.2 Elementos de un modelo UML.

Los diagramas de casos de uso se emplean para optimizar el entendimiento de los requerimientos, los casos de uso describen la funcionalidad del sistema desde la perspectiva del usuario, además de mostrar la interacción entre un usuario y el sistema, ayudando a capturar los requerimientos de un sistema de cómputo.

"Un caso de uso es una secuencia de acciones que dan un resultado observable para un actor particular"
Ivar Jacobson.

Un actor es una entidad externa que realiza un papel dentro del sistema, pueden ser personas que desempeñen alguna función dentro del sistema, otros sistemas de cómputo, máquinas o aparatos eléctricos [8]. En la figura 3.3 se muestra la notación en UML para un actor y un caso de uso.

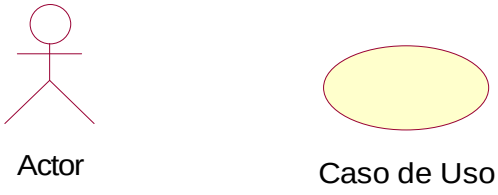

Figura 3.3 Notación de UML para Actores y Casos de Uso

Los diagramas de clases describen los tipos de objetos y las relaciones que existen entre ellos dentro del sistema. En la figura 3.4 se muestra la notación de UML para una Clase.

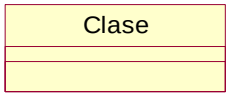

Figura 3.4 Notación de UML para Clases

Los diagramas de secuencia resaltan el orden en que los eventos suceden dentro del sistema entre un objeto y otro. Generalmente se construye un diagrama de secuencia para cada caso de uso, este diagrama se compone de objetos y mensajes. En la figura 3.5 se muestra la notación de UML para los diagramas de secuencia.

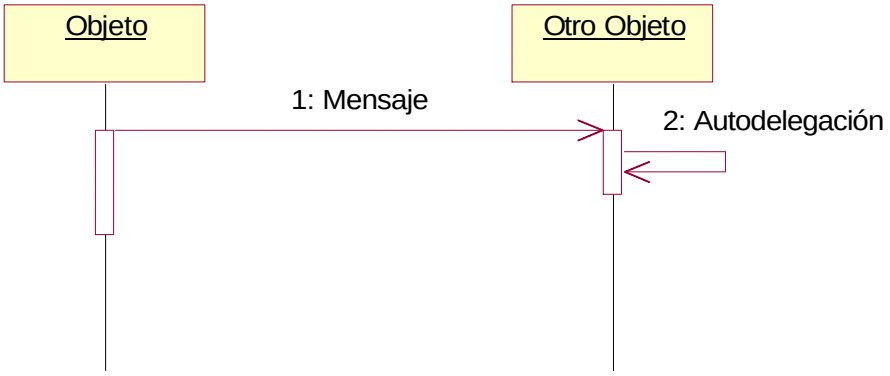

Figura 3.5 Notación de UML para los Diagramas de Secuencia

UML puede ser utilizado por cualquier proceso de desarrollo de software ya que es independiente de la tecnología de implementación.

3.2 Descripción general del sistema

Hoy en día la forma de trabajo de las organizaciones es grupal, se crean grupos para cada una de las áreas de la empresa y comúnmente se realizan reuniones de trabajo. La realización de reuniones trae consigo algunos inconvenientes como son: el lugar de la reunión y los acuerdos a los que se lleguen.

El lugar de la reunión se convierte en un problema si alguno de los participantes de la misma se encuentra en otro lugar, lo cual aplazaría la reunión hasta que él se encontrara en el lugar acordado.

Los acuerdos que se realizan dentro de una reunión también representan algunos problemas ya que casi nadie que asiste a la reunión toma la precaución de tomar notas de lo que se esta discutiendo y esto causa malas interpretaciones de los acuerdos.

Para evitar estos problemas se creó este sistema colaborativo el cual cuenta con un canal de comunicación entre los miembros de un grupo de trabajo y la creación de salas virtuales para las reuniones, además de contar en las reuniones con un sistema de votación y graficación de los mismos y al finalizar la reunión se redacta de forma automática un documento con los temas tratados dentro de la reunión.

El sistema desarrollado es un sistema colaborativo ya que cumple con las características de estos sistemas como son: proveer un ambiente de colaboración, manteniendo la información en un sitio común y proporcionando la interacción entre los usuarios. Además que cuenta con características síncronas (Reuniones y Chat) y asíncronas (E-mail).

Cabe mencionar que el sistema se puede ejecutar en cualquier sistema operativo, esto debido que se utilizo el lenguaje de programación Java, incluyendo el Palm OS. Al ser ejecutado el sistema en una Palm tiene la ventaja de brindar movilidad a las personas, además de brindar el envío y recepción de correo electrónico a través de dicho dispositivo.

3.3 Descripción funcional

Las funciones que provee el sistema son: correo electrónico, conversaciones y reuniones.

- Correo Electrónico. Se brinda el servicio de lectura, redacción y envío de correos electrónicos, para enviar información a los miembros de la organización, además de contar con machotes de documentos para agilizar el trabajo de envío de notificaciones, por ejemplo para reuniones, asignación de trabajos, calendario de actividades, etc., el sistema almacena la información en la base de datos. Si el usuario desea enviar la información a todos los miembros del grupo, los campos se llenaran automáticamente.
- Conversaciones. Las conversaciones sólo se podrán realizar con miembros de su grupo de trabajo.
- Reuniones. Dentro de las reuniones sólo entran los miembros a los que les fue enviada una notificación, dentro de esta notificación se les informara la hora, un login y password para dicha reunión. La hora de entrada se restringirá a más menos minutos. Se contará con un chat, para conversar con todos los miembros de la reunión. Si se necesita realizar votaciones dentro de la reunión se cuenta con una aplicación para almacenar, contabilizar y graficar los votos de los miembros. Antes de comenzar la votación se solicita el motivo por el cual se realizara dicha votación. Cuando se levante la sesión se redacta de forma automática un documento donde se recopila quién asistió a la reunión, la hora de entrada, la fecha de la reunión, las conversaciones realizadas y sí se llevaron a cabo votaciones, los resultados de las mismas.

El sistema cuenta con una base de datos con la información de los miembros de una organización o empresa (nombre, dirección, teléfono, etc.), sus respectivos login, password y el grupo al que pertenece, cada grupo se clasificará por niveles de prioridad.

En la figura 3.6 se muestra un diagrama de cómo funciona el sistema una vez integrado.

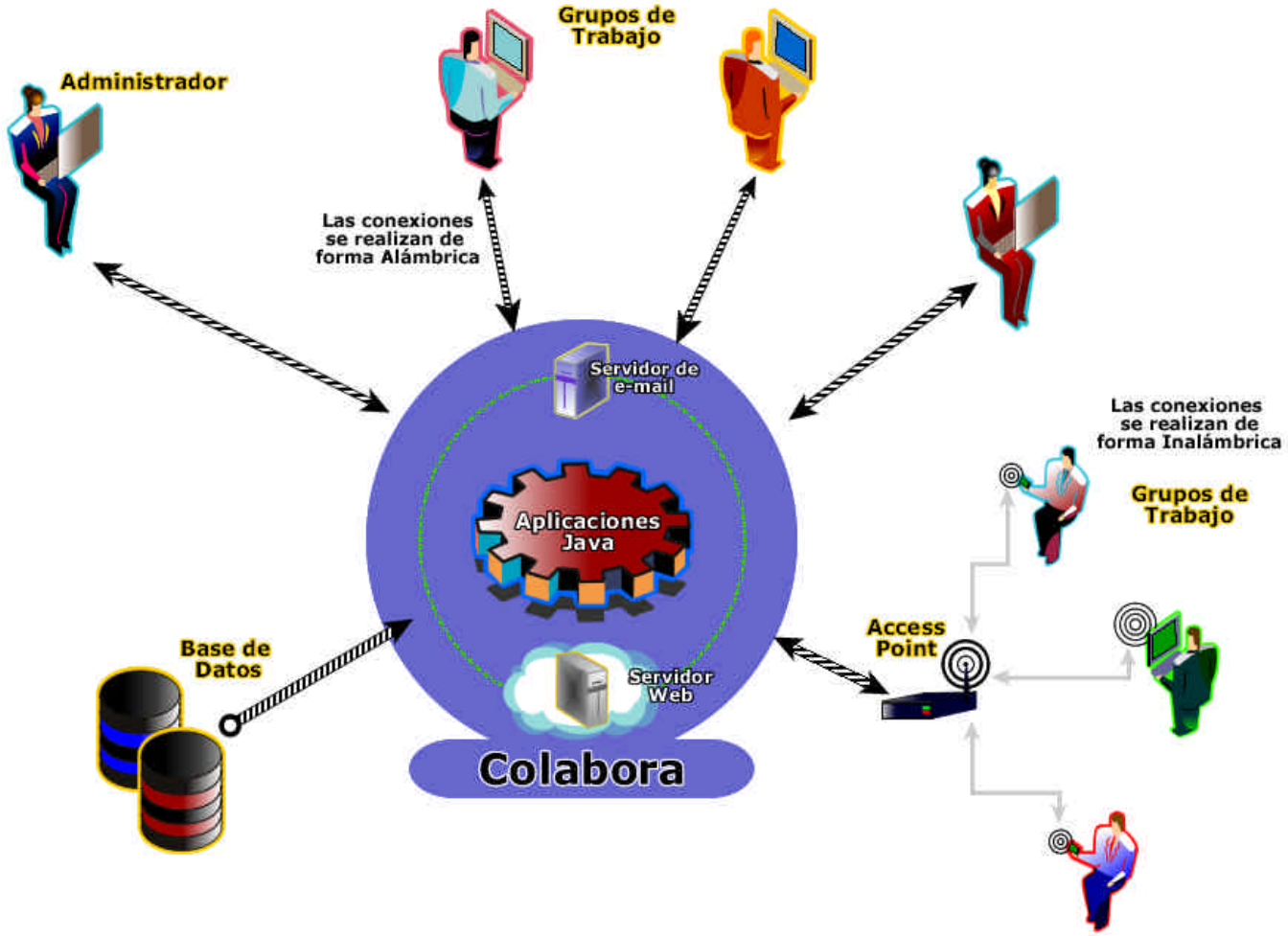

Figura 3.6 Diagrama del Sistema Colabora

3.4 Descripción del Problema

El problema consiste en desarrollar un sistema que automatiza las reuniones del personal sin importar el lugar en que cada miembro se encuentre, además de establecer un canal de comunicación entre los mismos.

En la actualidad las reuniones de trabajo de un grupo de personas que se encuentran en lugares distintos, se tienen que planear con unos días de anticipación, además del pago de viáticos al personal que necesita transportarse al sitio de la reunión. Esto afecta a todo el personal que asiste a reuniones en lugares remotos. El impacto de este sistema es realizar un canal de comunicación para las reuniones de personal sin que importen el lugar en que se encuentren ni el sistema operativo con el que cuenten. Esta solución exitosa ofrece las siguientes características:

- Un canal de comunicación seguro y confiable para las reuniones en línea.
- Un sistema de votaciones para la toma de decisiones dentro de la reunión.
- Redacción automática de un documento con todo lo que se trato dentro de la reunión.
- Lectura de correo electrónico.
- Conversación con otros miembros de su grupo de trabajo.
- Soporte multiplataforma tanto en PC's y Dispositivos Móviles.

3.5 Descripciones de Cliente y Usuario

Dentro del sistema existen diferentes usuarios con los cuales interactúa el sistema, a continuación se describen los tipos de usuario, su descripción y el rol que desempeña dentro del sistema.

3.5.1 Usuario

El usuario es la persona que sólo maneja exteriormente el sistema, es decir con respecto a reuniones y conversaciones en grupo. La responsabilidad que tiene dentro del sistema es el buen manejo del sistema. Este tipo de usuario podrá acceder al sistema desde cualquier lugar en el que se encuentre, ya sea desde una PC o un dispositivo móvil.

3.5.2 Administrador

El administrador es la persona responsable de la configuración, manutención y actualización del sistema. Las responsabilidades que tiene dentro del sistema son la configuración, la manutención y la administración del sistema. Este tipo de usuario será el encargado de ingresar los datos del personal, establecer los permisos de los grupos y mantener la funcionalidad del sistema.

3.6 Necesidades clave del usuario / cliente

En la tabla 3.1 se muestra una lista de necesidades así como las soluciones propuestas.

Tabla 3.1 Necesidades y Soluciones

Necesidad	Soluciones propuestas
Reuniones	Salas virtuales para llevar a cabo las reuniones sin importar el lugar en que se encuentren los miembros de la organización.
Votaciones	Sistema de contabilización y graficación de los votos en línea.
Comunicación	Establecimiento de un canal de comunicación tipo chat para los grupos de trabajo.
Movilidad	El acceso inalámbrico por medio de dispositivos móviles al sistema colaborativo cubrirá esta necesidad.

3.7 Perspectiva del producto

El producto mejora la comunicación entre los miembros que conforman un grupo de trabajo por medio de la creación de salas de reunión virtual, las cuales cuentan con un sistema de votación y la creación de un documento que contenga los puntos tratados en cada reunión.

3.8 Sumario de Capacidades

Los beneficios para el consumidor son los siguientes:

- Reducción de costos de transportación de las personas que se reúnen frecuentemente.
- Comunicación más rápida y eficiente entre los miembros de un grupo de trabajo.
- Movilidad de los empleados ya que pueden desplazarse y trabajar al mismo tiempo.

Las Características soportadas por el sistema son:

- Lectura de correo electrónico.
- Chat.

- Reuniones virtuales.
- Votaciones.
- Soporte para plataforma PC y dispositivos móviles.

3.9 Características del Producto

- Interfaz Amigable. Cuenta con una interfaz gráfica amigable para los usuarios del sistema, con esto los usuarios tienen mayor y mejor interacción con el sistema.
- Manejo Funcional. El sistema desarrollara todas sus tareas a cargo, a fin que el usuario obtenga los resultados deseados.
- Facilidad de Uso. El sistema será de fácil utilización para cualquier miembro de la organización, sea un usuario experimentado o no.
- Multiplataforma. El sistema se podrá utilizar en cualquier sistema operativo para PC's además de dispositivos móviles con tecnología Bluetooth.

3.10 Restricciones

El sistema sólo simula el acceso inalámbrico de los dispositivos móviles con tecnología Bluetooth. Una vez implantado el sistema en forma real la restricción sería el alcance de los dispositivos móviles.

3.11 Requerimientos del sistema

En la tabla 3.2 se muestran los requerimientos en los equipos para la funcionalidad del sistema.

Tabla 3.2 Requerimientos del sistema

PC	Dispositivo Móvil	Servidor
Java2 1.4 Procesador Pentium II. 64 Mb. RAM. Procesador a 333 MHz. Capacidad de disco duro 2Gb.	Palm m125 o superior, teléfono celular, dispositivo RIM. Java KVM. Tecnología Bluetooth. 1 Mb. disponible en memoria.	Java2 1.4 MySql. Driver JDBC para MySql 64 Mb. RAM Procesador 333MHz Capacidad de disco duro 2Gb.

3.12 Especificación de Casos de Usos

3.12.1 Caso de uso: Chat

Este caso de uso es el encargado de comunicar a los miembros de un grupo de trabajo. Los actores que interactúan con este caso de uso son los Usuarios, a continuación se muestra en la tabla 3.3 el flujo básico de eventos y en la figura 3.7 se muestra el diagrama de secuencia de este caso de uso.

Tabla 3.3 Flujo de eventos del caso de uso Chat

Actor	Sistema
1. Este caso de uso inicia cuando el usuario desea conversar con los miembros de su grupo de trabajo. 2. El usuario escoge si desea platicar con alguno miembro en específico o con todos los miembros.	3. Si el usuario desea conversar con todos los miembros el sistema establece un canal de comunicación general. 4. Si el usuario desea conversar con un sólo miembro el sistema establecerá un canal privado.

- **Precondiciones**
  - Identificación: Para que este caso de uso se ejecute, los usuarios deben de estar identificados.
- **Poscondiciones**
  - Fin del Chat: Al terminar este caso de uso se finaliza el sistema por parte del usuario.

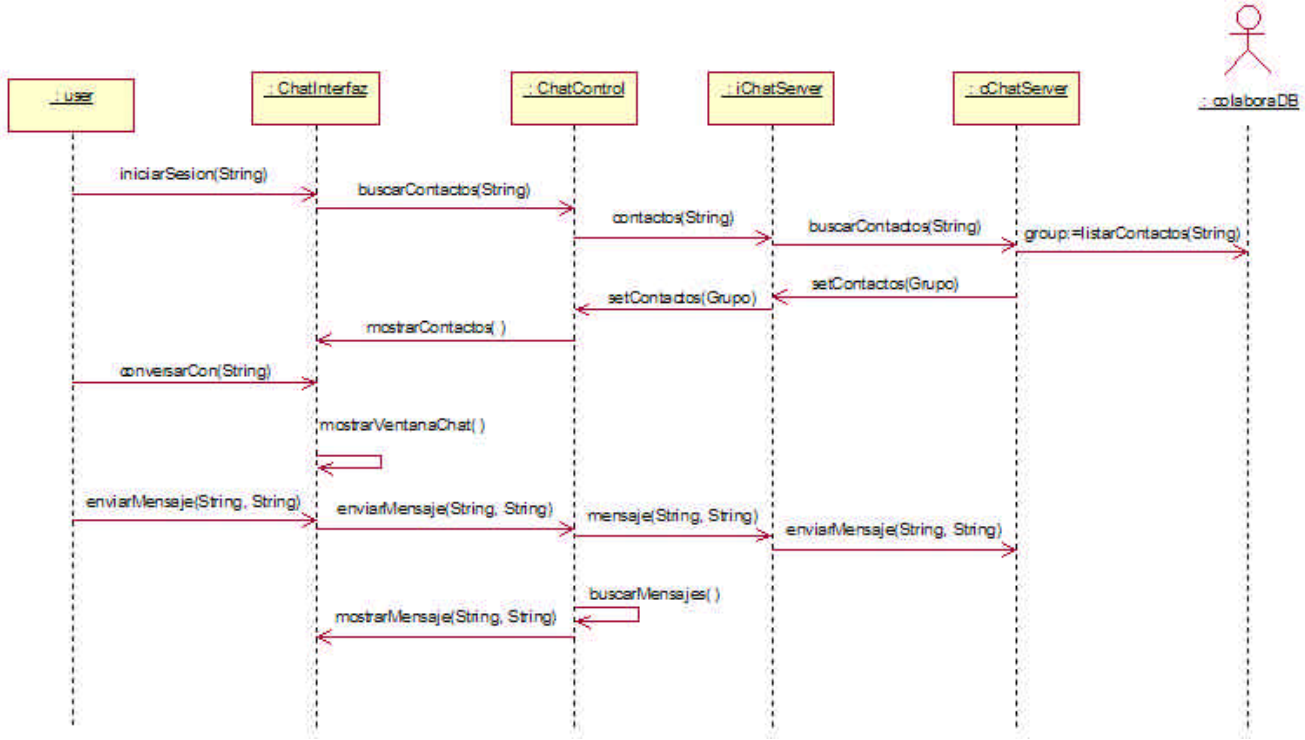

Figura 3.7 Diagrama de secuencia del caso de uso Chat

3.12.2 Caso de uso: e-mail

Este caso de uso es el encargado de leer y enviar correos electrónicos, en el caso de los dispositivos móviles solo mensajes de texto, entre los usuarios del sistema. Al igual que el caso de uso anterior sólo se interactúa con los Usuarios, a continuación se muestra en la tabla 3.4 el flujo básico de eventos y en la figura 3.8 se muestra el diagrama de secuencia de este caso de uso.

Tabla 3.4 Flujo de eventos para el caso de uso E-mail

Actor	Sistema
1. Este caso de uso inicia cuando el usuario decide leer su e-mail. 3. El usuario le indica al sistema que correo electrónico desea ver. 5. Si el usuario desea enviar un correo electrónico le hace la petición al sistema.	2. El sistema le muestra al usuario una lista con los correos electrónicos del usuario. 4. El sistema muestra al usuario el texto del correo electrónico que selecciono. 6. El sistema le muestra al usuario el formulario para envío de e-mail. 7. Si el usuario desea ocupar alguna plantilla para el envío de e-mail se ejecuta el flujo alterno plantillas.

- **Precondiciones**
    - Identificación: Para que este caso de uso se pueda iniciar el usuario deberá de estar identificado dentro del sistema.
- **Poscondiciones**
    - Chat: El usuario entra al caso de uso de Chat.

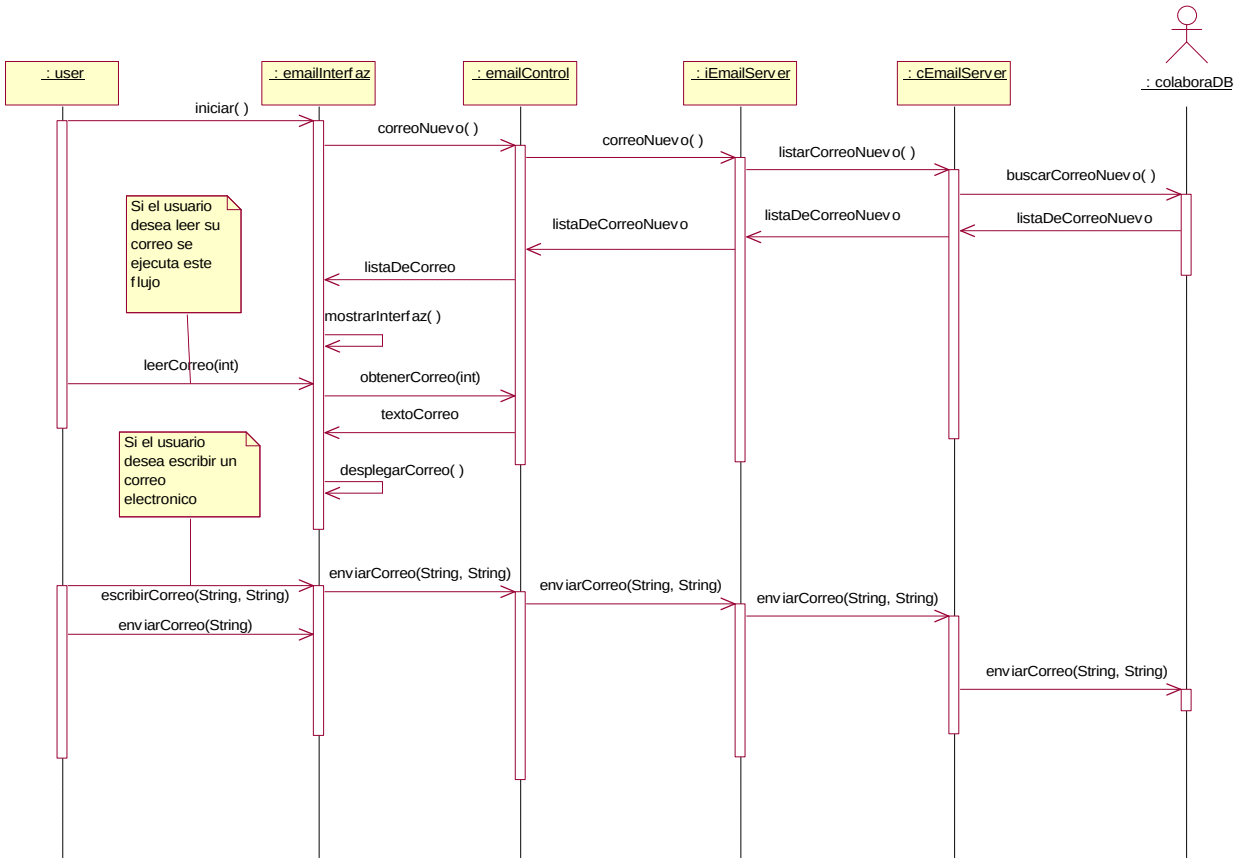

Figura 3.8 Diagrama de secuencia del caso de uso E-mail

3.12.3 Caso de uso: Reunión

Este caso de uso tiene el propósito de crear salas de reunión para el trabajo en grupo, los actores que interactúan con este caso de uso son los Usuarios, a continuación se muestra en la tabla 3.5 el flujo básico de eventos y en la figura 3.9 se muestra el diagrama de secuencia de este caso de uso.

Tabla 3.5 Flujo de eventos para el caso de uso Reunión

Actor	Sistema
1. Este caso de uso comienza cuando un usuario entra a la reunión a la que fue notificado por medio de correo electrónico.	2. Si el usuario llega a la reunión con un límite de +/- cinco minutos el sistema lo registra como asistente a la reunión, si no se va al flujo alterno el usuario llego tarde. 3. El sistema comunica al usuario con los demás miembros de la reunión por medio del caso de uso chat. 4. Si los usuarios deciden realizar una votación se lleva a cabo el flujo alterno Votación. 5. Al finalizar la reunión el sistema ejecuta el flujo alterno Fin de Reunión y después se regresa al sistema principal.

- **Flujos Alternativos**
    - El usuario llego tarde: Si el usuario llega a la reunión cinco minutos tarde, el sistema no le permite el acceso a la reunión y crea un registro de que no asistió a la reunión.
    - Votación: Para realizar las votaciones se ejecuta el caso de uso de votación.
    - Fin de Reunión: Al finalizar la reunión se creará un documento con todos los puntos tratados en la reunión.
- **Precondiciones**
    - Identificar: Para que este caso de uso se inicie el usuario deberá de estar identificado.
- **Poscondiciones**
    - Fin de la Reunión: Al finalizar este caso de uso se retorna al caso de uso chat con los miembros de su grupo de trabajo.

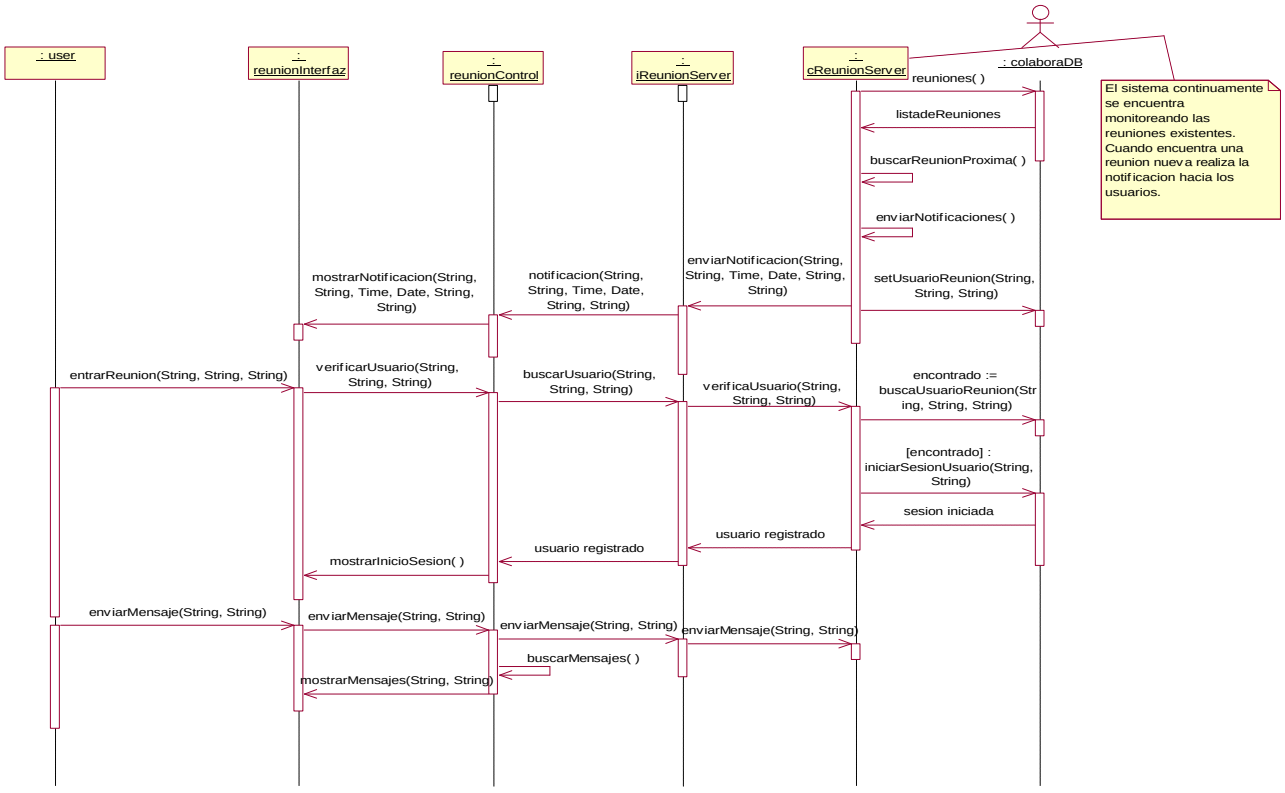

Figura 3.9 Diagrama de secuencia del caso de uso Reunión

3.12.4 Caso de uso: Identificar

Este caso de uso es el encargado de identificar a los actores dentro del sistema. Los actores que interactúan con este caso de uso son los Usuarios, a continuación se muestra en la tabla 3.6 el flujo básico de eventos y en la figura 3.10 se muestra el diagrama de secuencia de este caso de uso.

Tabla 3.6 Flujo de eventos para el caso de uso Identificar.

Actor	Sistema
1. El usuario introduce su login, password y grupo de trabajo al que pertenece.	2. El sistema busca al usuario en la base de datos. 3. En caso de encontrarlo se inicia el caso de uso Chat. En caso contrario se ejecuta el flujo alterno Usuario no encontrado.

- **Flujos Alternativos**
  - Usuario no encontrado: Si el usuario no se encontró dentro de la base de datos, en este caso se vuelven a pedir los datos (login, password y grupo).

- **Precondiciones**
  - Iniciar la aplicación: Para que este caso de uso se ejecute el usuario debió de haber iniciado la aplicación.

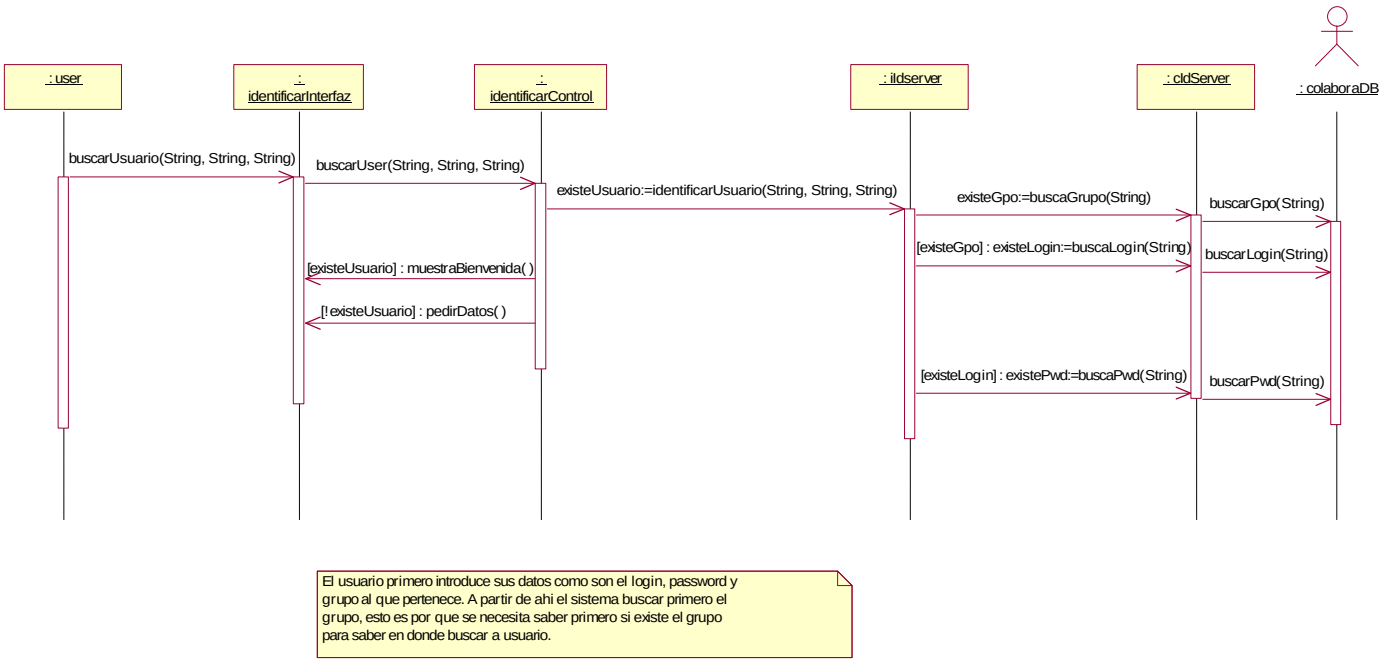

Figura 3.10 Diagrama de secuencia del caso de uso Identificar

3.12.5 Caso de uso: Votar

Este caso de uso es el encargado de llevar el conteo de los votos que se realicen dentro de una reunión, además de graficar y mostrar los resultados de las votaciones. Los actores que interactúan con este caso de uso son los Usuarios, a continuación se muestra en la tabla 3.7 el flujo básico de eventos y en la figura 3.11 se muestra el diagrama de secuencia de este caso de uso.

Tabla 3.7 Flujo de eventos para el caso de uso Votar

Actor	Sistema
1. Este caso de uso comienza cuando los asistentes a la reunión desean realizar una votación. 3. Los usuarios realizan su voto.	2. El sistema pide a los usuarios el motivo de la votación. 4. El sistema muestra gráficamente el resultado de la votación. 5. El sistema almacena el resultado de la votación.

- **Precondiciones**
  - Reunión: Para que este caso de uso se ejecute los usuarios deben de estar dentro de una Reunión.
- **Poscondiciones**
  - Fin de las Votaciones: Al finalizar este caso de uso se regresará al caso de uso Reunión.

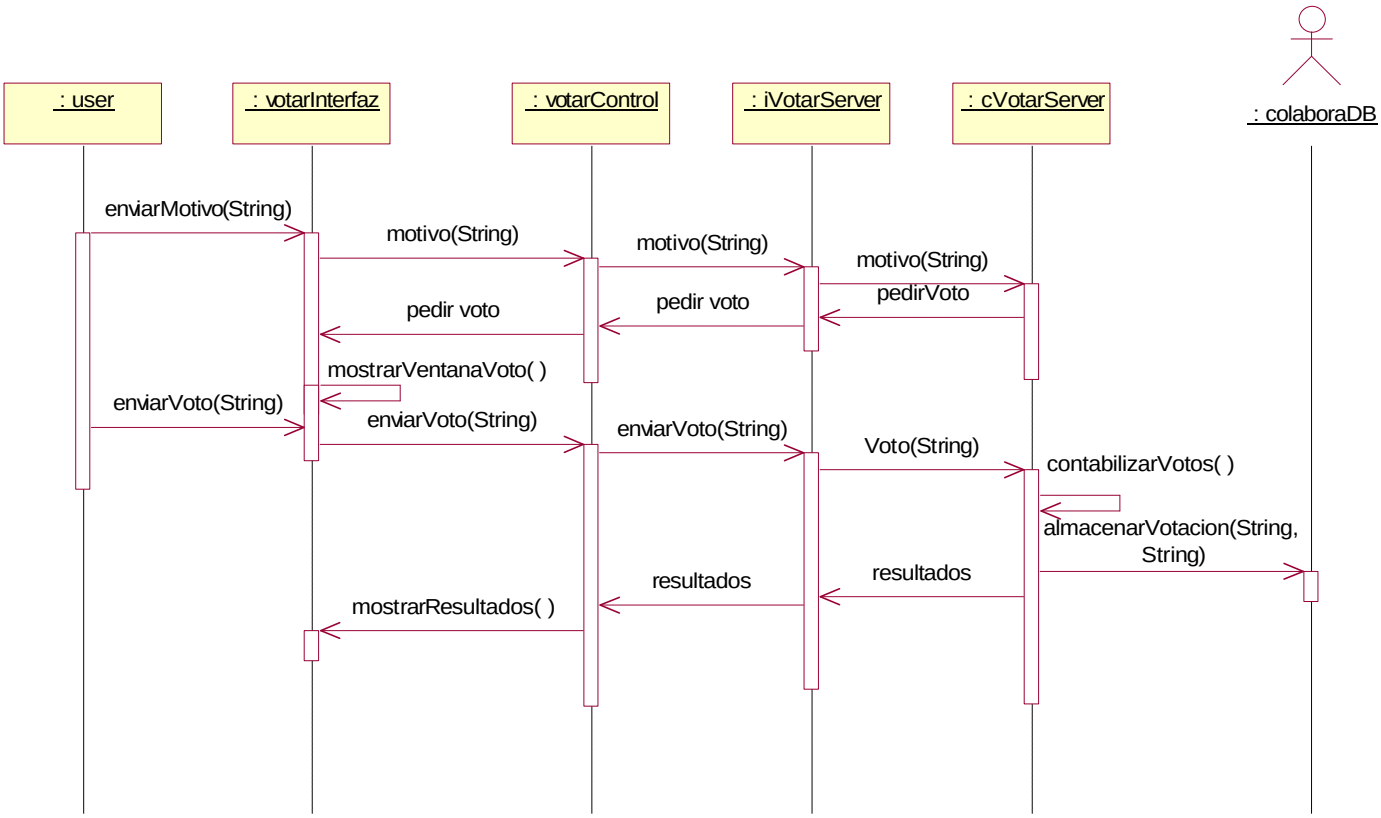

Figura 3.11 Diagrama de secuencia del caso de uso Votar

En la figura 3.12 se muestra el diagrama de casos de uso, en el cual se muestran las relaciones que existen entre los casos de uso encontrados en el sistema.

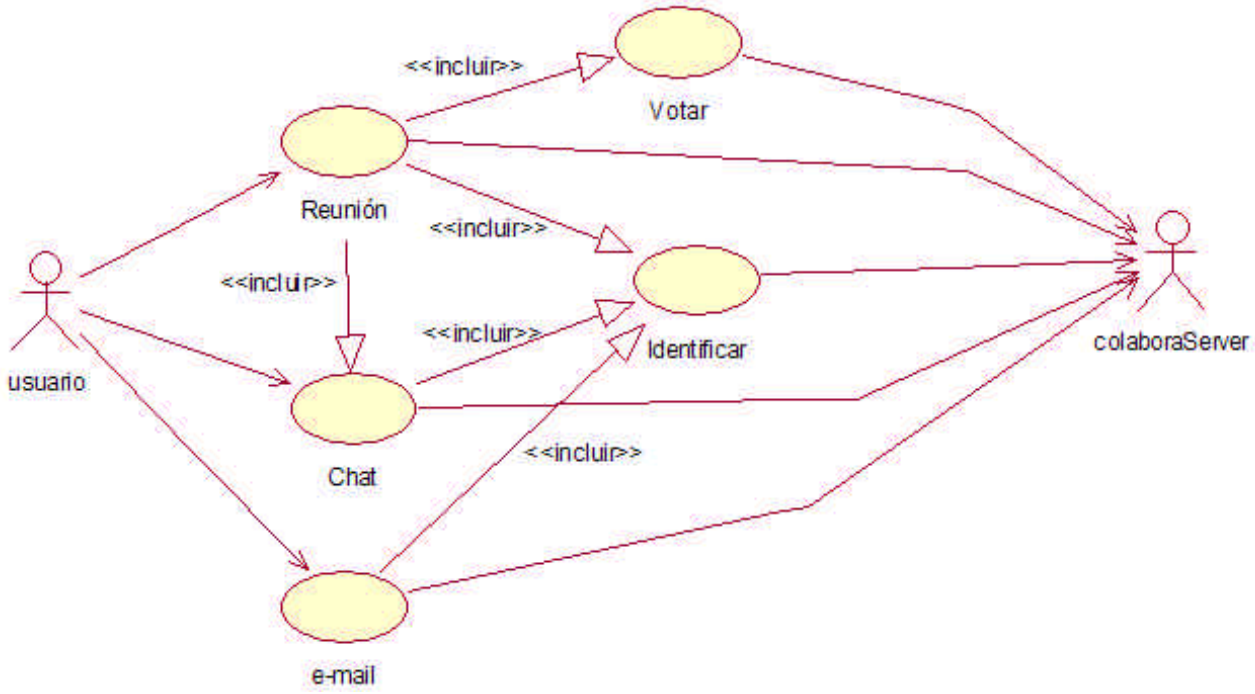

Figura 3.12 Diagrama de Casos de Uso

A continuación se muestran en las figuras 3.13 y 3.14 los diagramas de las clases que se encontraron para el sistema.

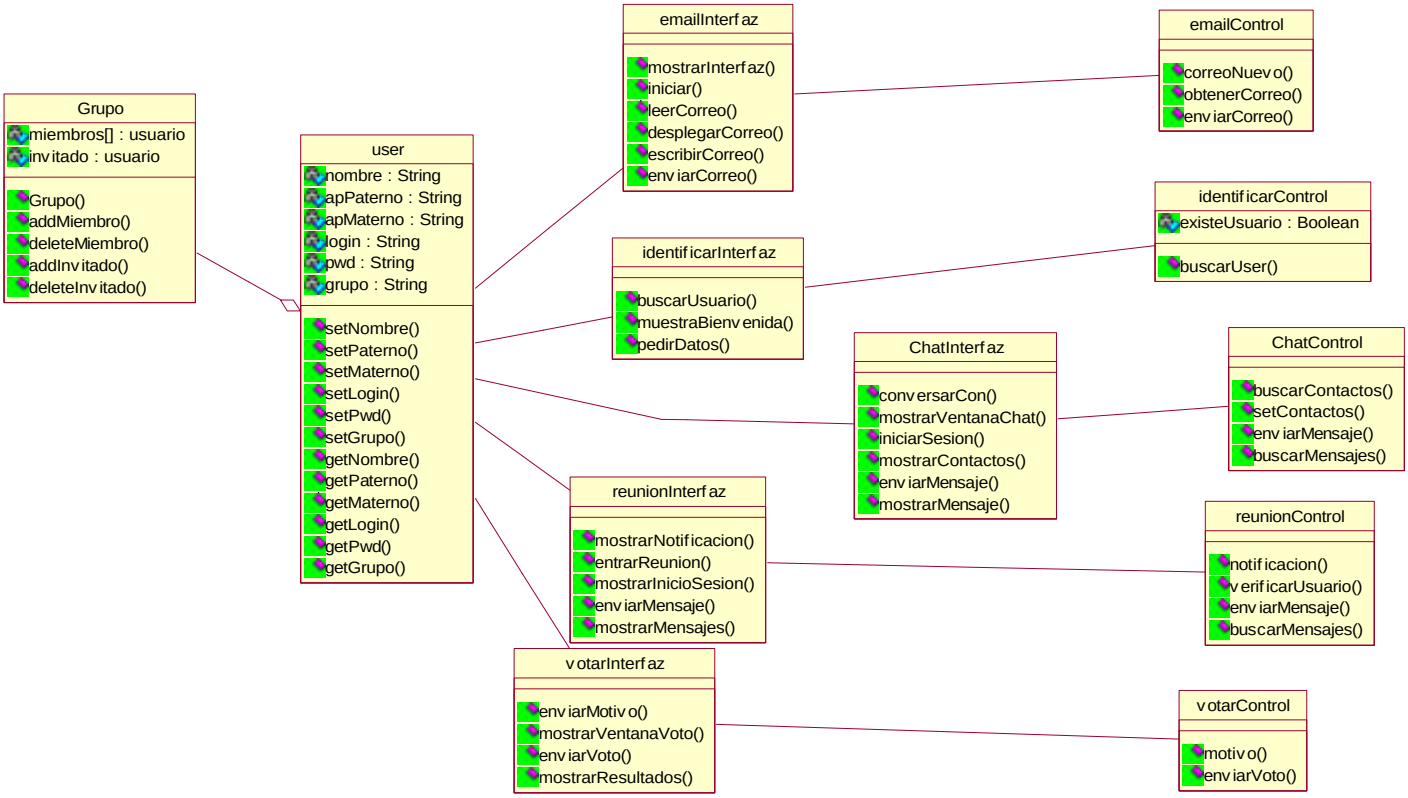

Figura 3.13 Diagrama de clases para la aplicación cliente

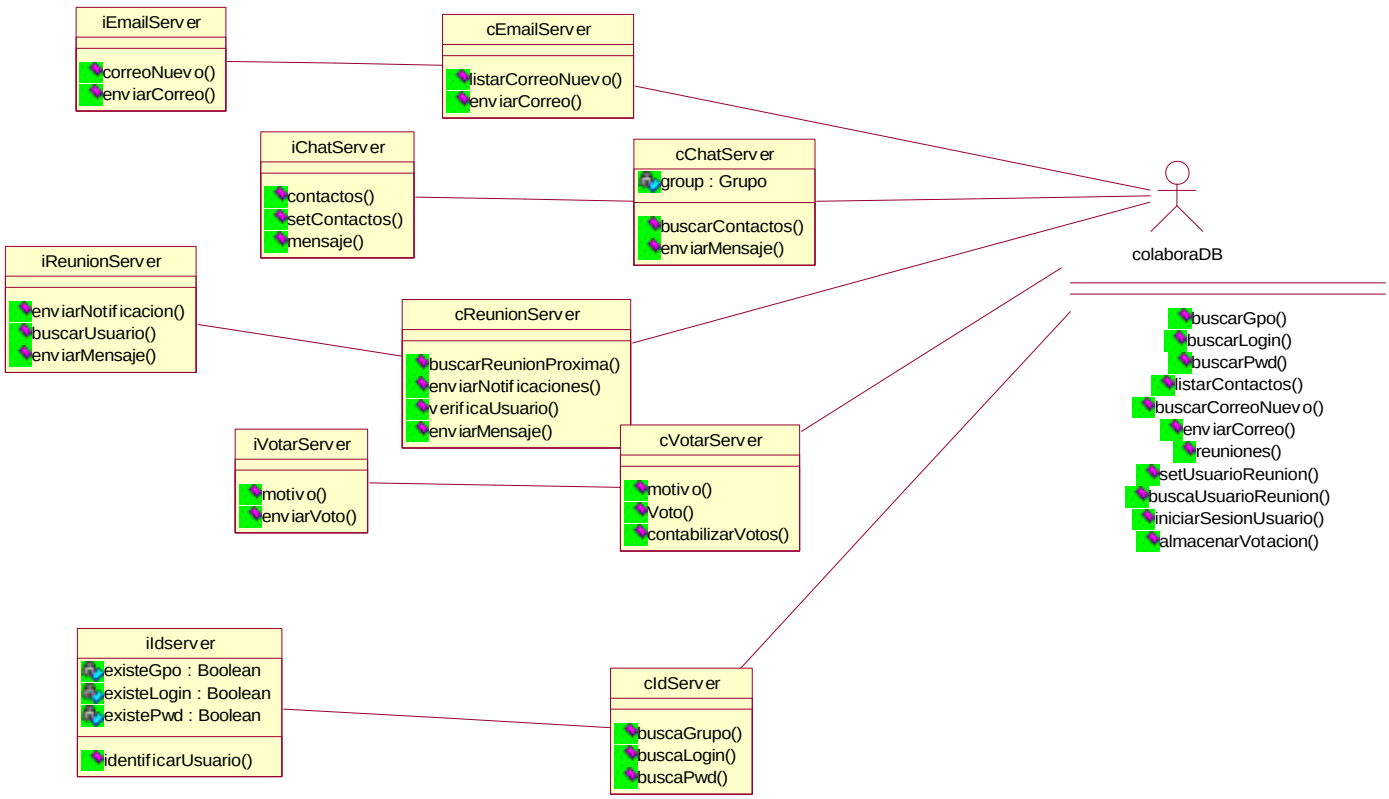

Figura 3.14 Diagrama de clases para el servidor de aplicaciones

En la figura 3.15 se muestra la base de datos, con sus respectivas tablas y relaciones.

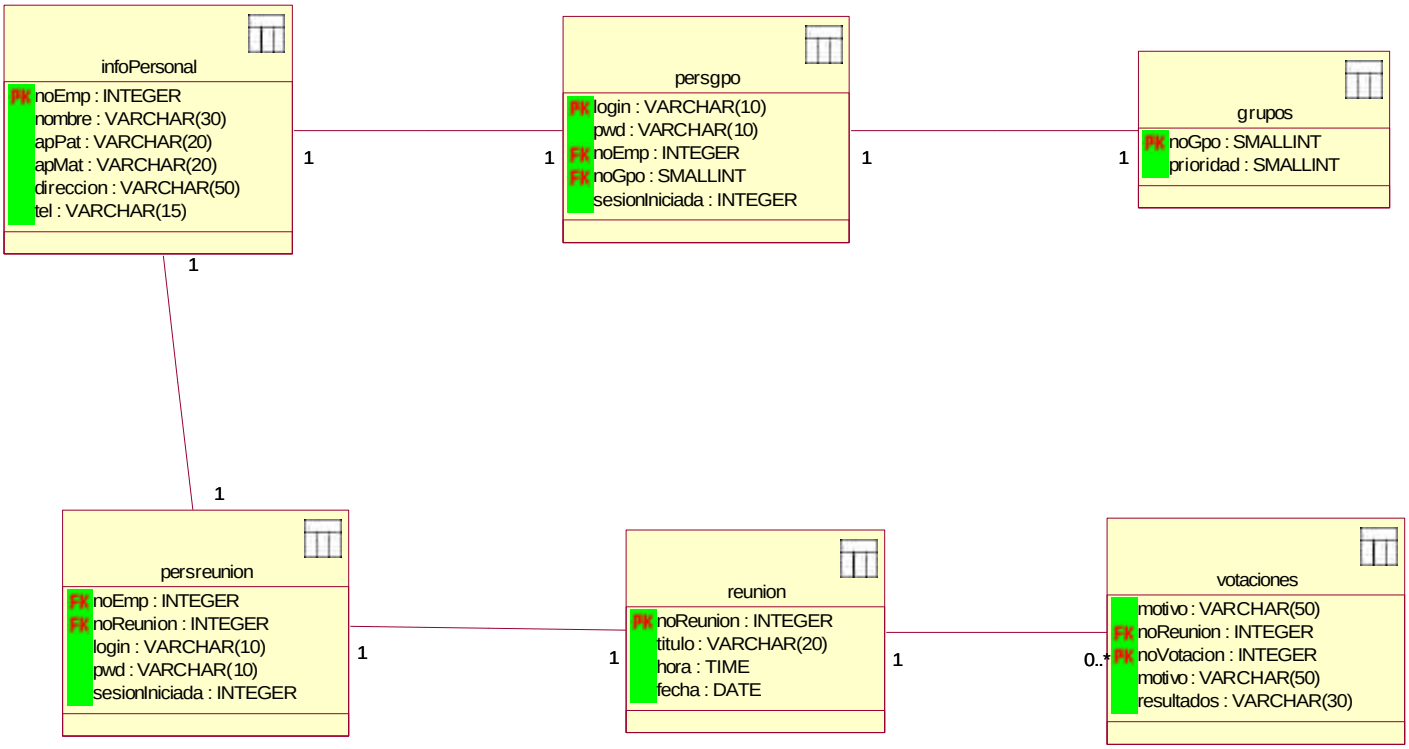

Figura 3.15 Diagrama de la base de datos

Capítulo 4 Implementación del Sistema

4.1 Introducción

Para realizar el desarrollo del sistema se empleo un simulador de Bluetooth de la empresa RococoSoftware el cual se puede descargar desde [URL 8]. Este simulador proporcionó la comunicación entre el dispositivo móvil, el servidor y el cliente de la PC, una aplicación real no fue posible debido al costo de las herramientas para el desarrollo de Bluetooth.

La utilización del lenguaje Java para el desarrollo de este proyecto permitió la portabilidad del mismo hacia cualquier sistema operativo (Windows, Linux, MacOS, etc.) además su robustez proporciona estabilidad a la comunicación y protección a fallas del sistema. La aplicación cliente de PC y el servidor de las aplicaciones se realizaron en Java Standard Edition J2SE [URL 10]y la aplicación cliente del PDA se realizó en Java Micro Edition J2ME [11].

El sistema cuenta con una base de datos la cual se programo en MySQL, este servidor de base de datos se encuentra en Linux Red Hat 8.0 al igual que el servidor de correo electrónico SendMail y el sistema administrador de la base de datos el cual se ejecuta en el servidor web Jakarta Tomcat con soporte de JSP y Servlets [9].

Para la simulación de los dispositivos móviles se empleó el simulador de PalmOS y el J2ME Wireless Toolkit 1.0.4.01 de la empresa Sun MicroSystems. Este simulador permite ejecutar la aplicación en teléfonos celulares.

A continuación se hablará de cómo se realizó el sistema así como la descripción del sistema.

4.2 Desarrollo del Sistema

El Sistema de Reuniones utiliza el protocolo de comunicación Bluetooth el cual ya se describió brevemente en capítulos anteriores, este protocolo se eligió por la gran variedad de dispositivos que lo emplean como teléfonos celulares, PDA's y PC's, además de consumir menos energía lo cual permitiría a los dispositivos estar conectados más tiempo.

La comunicación con los dispositivos móviles con la PC se llevo a cabo empleando las librerías del API de Bluetooth [5] [10] que se proporcionan con el simulador de Bluetooth. A continuación se describirá como se realizó la conexión por medio de este protocolo utilizando Java.

Primero que nada se necesitan incluir las librerías que contienen los objetos necesarios para la conexión de Bluetooth las cuales son:

```
import javax.Bluetooth.*;
import com.rococosoft.impronto.configuration.*;
import com.rococosoft.impronto.util.BluetoothConstants;
```

La librería javax.Bluetooth contiene las clases LocalDevice, RemoteDevice, UUID, así como las clases para el manejo de excepciones de Bluetooth. Las otras dos librerías com.rococosoft.impronto.configuration y com.rococosoft.impronto.util.BluetoothConstants contienen las clases para la conexión al simulador.

Para establecer la conexión en modo de Servidor, se implementó el siguiente código:

```
...
try{
      final StreamConnectionNotifier notifier = ( StreamConnectionNotifier )Connector.open( "btspp://localhost:" + uuid.toString() );
      final LocalDevice dev = device;
      StreamConnection c = null;
      for(;;){
        try{
          c = notifier.acceptAndOpen();//se mantiene aceptando las peticiones
          ServiceRecord r = dev.getRecord( notifier );
          r.setDeviceServiceClasses( BluetoothConstants.SERVICE_AUDIO);
          dev.updateRecord( r );
          //hasta aquí se tienen los parámetros para la conexión
          salida = c.openOutputStream();
          //se salva esta stream para usarlo después.
          outputStreams.put( c, salida);
          //se crea el nuevo hilo de ejecución para esta conexión
          new btServerThread(this, chatPC, c, output,c.openInputStream(),salida);
        }catch(IOException _){
              //se ignora la excepción
        }finally{
          if( c != null ){
            try{
              c.close();
            }catch( IOException _){
                  //se ignora la excepción
            }
          }
        }
    }catch(Exception e){
      // se ignora la excepción
    }
```

La clase StreamConnectionNotifier se emplea para recibir las peticiones de los dispositivos los cuales se conectan por medio de Streams, estos son equivalentes a los sockets en una conexión alámbrica.

La conexión de Bluetooth de la aplicación se hace en las clases btServer y btServerThread, estas clases son las encargadas de realizar la comunicación con los clientes móviles. A continuación se describirán brevemente los métodos.

- La clase btServer es un hilo de ejecución que se mantienen ejecutando en segundo plano y no afectan al desempeño del sistema. En el apéndice A en la tabla A.1 se muestran los métodos y objetos de esta clase.
- La clase btServerThread es la encargada de atender las peticiones de un sólo dispositivo, esta clase también es un hilo de ejecución. En el apéndice A en la tabla A.2 se muestran los métodos y objetos de esta clase.

La comunicación con los clientes alámbricos se realiza con las clases chatServer y serverThread. A continuación mencionaremos los métodos que las conforman:

- La clase chatServer es la encargada de establecer la comunicación entre los clientes de PC. En el apéndice A en la tabla A.3 se muestran sus métodos y objetos.
- La clase serverThread es la encargada de atender a un sólo cliente de PC. En el apéndice A en la tabla A.4 se muestran los métodos y objetos que conforman esta clase.

Estas cuatro clases son el núcleo del sistema ya que ellas son las encargadas de manipular la información para los usuarios. En la figura 4.1 se muestra el servidor totalmente integrado.

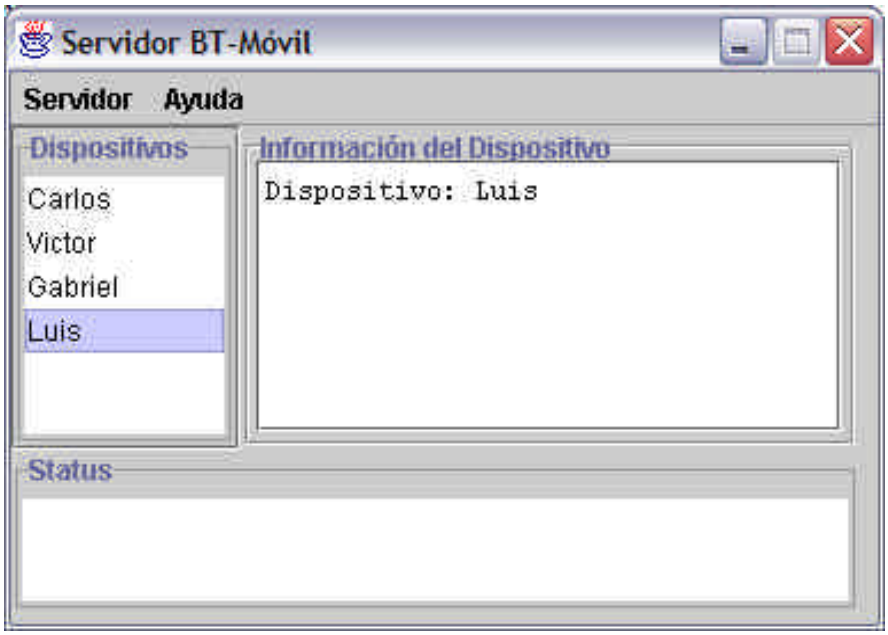

Figura 4.1 Servidor BT-Móvil

Las aplicaciones clientes cuentan con los módulos de lectura y escritura de correo electrónico, conversaciones y reuniones, básicamente se implementaron de la misma manera, sólo que la aplicación para la PC tiene una interfaz grafica más rica que la del dispositivo móvil, dado que la diferencia de poder de cómputo es muy grande entre un dispositivo y otro. El la figura 4.2 se muestra el sistema de reuniones para la PC.

Figura 4.2 Cliente para la PC

En la figura 4.3 se muestra el sistema para dispositivos móviles.

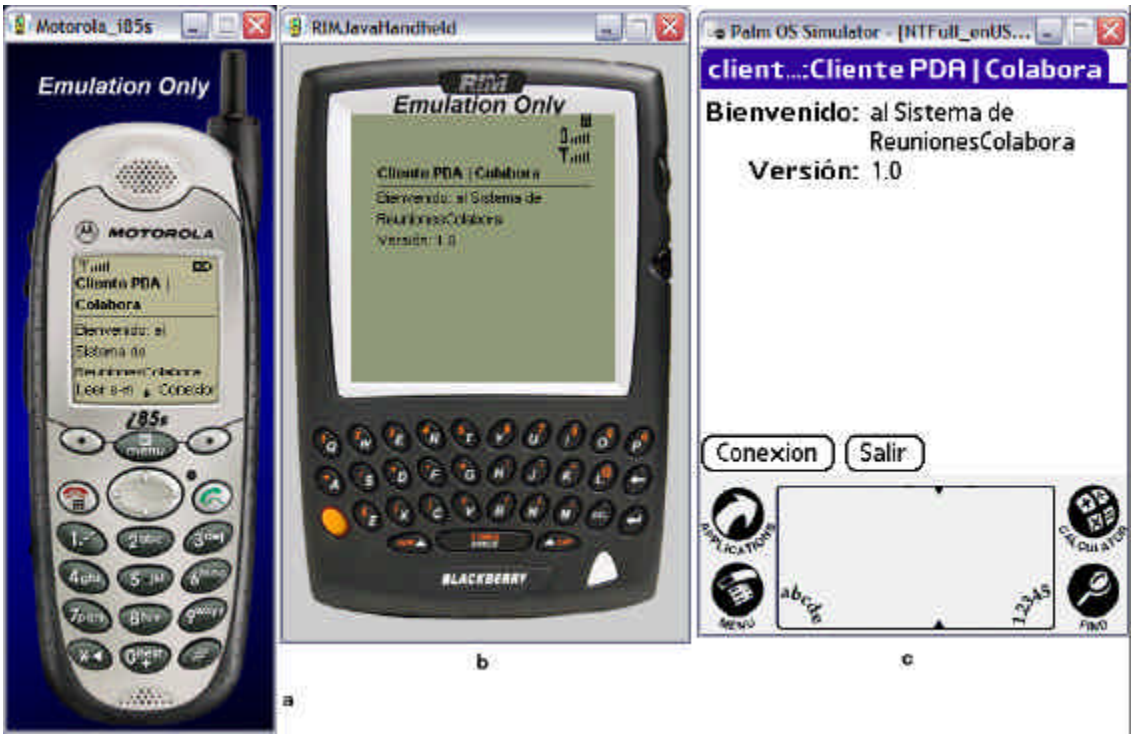

Figura 4.3 Aplicación cliente en: a) Motorola i85, b) RIM Blackberry y c) dispositivo PalmOs

Cabe mencionar que la aplicación móvil es la misma para todos los dispositivos mostrados en la figura 3, la razón por la que cambia la interfaz es por adaptarse al dispositivo en el cual se ejecuta, ésta operación de adaptación a la pantalla es realizada por la máquina virtual de java de dicho dispositivo.

La implementación de cada uno de los módulos del sistema se describirá a continuación.

4.3 Descripción Funcional

Las funciones que provee el sistema son: correo electrónico, conversaciones y reuniones, y se encuentran relacionadas como se muestra en la figura 4.4.

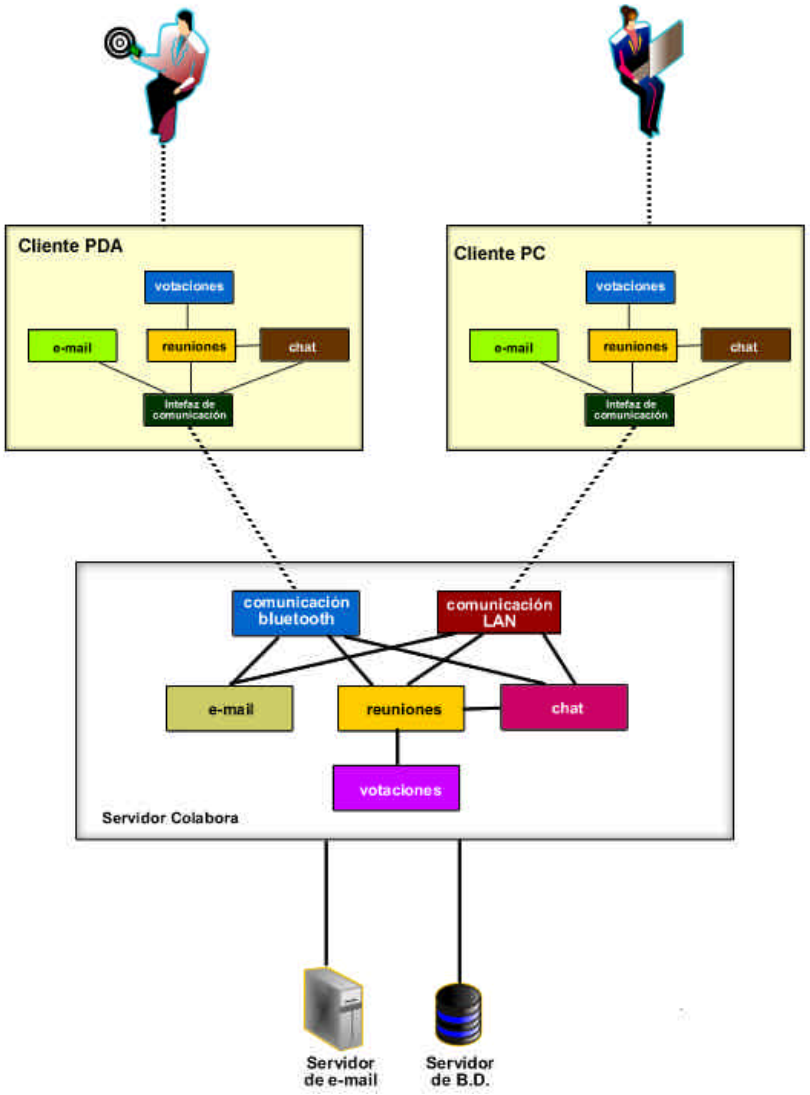

Figura 4.4 Módulos del sistema de reuniones

4.3.1 Correo Electrónico

La función de correo electrónico brinda el servicio de lectura, redacción y envío a los miembros de la organización. En la aplicación para los dispositivos móviles se necesita mandar ciertos mensajes para acceder a las funciones de correo electrónico, estos mensajes se muestran en la tabla 4.1.

Tabla 4.1 Mensajes del módulo e-mail al servidor

Mensaje	Acción
initsesion login@password@grupo	Cuando el sistema reciba estos datos, se obtendrá el login, el password y el grupo separados por @, para realizar la búsqueda en la base de datos.
Email	Cuando el sistema reciba esta instrucción, enviará al PDA una lista con los siguientes datos: Remitente y Asunto.
leermail NOMAIL	Esta instrucción hará que el sistema envié al PDA el e-mail que solicitó.
enviarmail destinatario;asunto;mensaje	Esta instrucción es la encargada de que el sistema envié el mail al destinatario.

4.3.2 Conversaciones

Las conversaciones sólo se podrán realizar con miembros de su grupo de trabajo, al igual que en módulo de e-mail, la aplicación cliente para el dispositivo móvil necesita enviar los mensajes dentro de un sólo mensaje, el formato se muestra en la tabla 4.2.

Tabla 4.2 Mensajes del módulo chat al servidor

Mensaje	Acción
mensaje destino:recado	Cuando el sistema recibe esta instrucción le envía al cliente destino el recado recibido.

4.3.3 Reuniones

Dentro de las reuniones sólo entran los miembros a los que les fue enviada una notificación por medio de correo electrónico, dentro de esta notificación se les informará la hora, así como un login y password diferentes para cada uno de los asistentes a la reunión. La hora de entrada se restringirá a más menos 5 minutos. Se contará con un chat, para conversar con todos los miembros de la reunión o en conversaciones individuales, sólo se podrán realizar dos conversaciones individuales por cada miembro de la reunión. Si se necesita realizar votaciones dentro de la reunión se cuenta con una aplicación para almacenar, contabilizar y graficar los votos de los miembros. Antes de comenzar la votación se solicita el motivo por el cual se procederá a votar. Cuando se levante la sesión se redacta de forma automática un documento donde se recopilan los nombres de los participantes que asistieron a la reunión, la hora de entrada, la fecha de la reunión, si se realizaron votaciones y sus resultados. Como se puede apreciar en la figura 4.4 el módulo de reuniones interactúa con el módulo de chat, esto es por que se emplea este módulo para realizar las conversaciones dentro de una reunión. En la tabla 4.3 se muestran las instrucciones que se tienen que enviar desde la aplicación del PDA al servidor para realizar la comunicación.

Tabla 4.3 Mensajes del módulo de reuniones al servidor

Mensaje	Acción
initreunion login@password	Cuando el sistema recibe este mensaje registra al usuario que acceso a la reunión.
votar si/no	Cuando el sistema reciba este mensaje se contabilizan los votos.

4.4 Módulo E-Mail

El módulo E-Mail esta compuesto por dos submódulos que son: lectura y escritura de correos electrónicos. El módulo de leer correo electrónico es el primero que analizaremos.

4.4.1 Leer e-mail

Este módulo se encarga de obtener los mensajes de correo electrónicos del servidor de e-mail y emplea el protocolo pop3 para obtener los correos. Este protocolo es ampliamente utilizado hoy en día por los proveedores de cuentas de correo electrónico.

Para obtener un correo de un servidor se empleo la librería Java Mail, esta librería es de código libre y es proporcionada por Sun para su libre uso, las clases proporcionadas nos permiten elegir el protocolo de lectura de correo electrónico como son pop3 e imap. El proceso de obtención de e-mail se describe brevemente en los siguientes pasos.

Paso 1: Incluir las clases que se emplean

```
import javax.mail.*;
import javax.mail.internet.*;
import javax.mail.internet.MimeMultipart ;
import javax.activation.* ;
import java.util.Properties ;
```

Paso 2: Crear las propiedades de la sesión

```
Properties props = new Properties();
```

Paso 3: Obtener la sesión

```
session2 = Session.getDefaultInstance( props, null );
```

Paso 4: Obtener el store

```
store = session2.getStore( "pop3" );
store.connect( "colabora", login, pwd );
```

Paso 5: Obtener el folder

```
folder = store.getFolder( "INBOX" );
folder.open( Folder.READ_ONLY );
```

Paso 6: Obtener el directorio de los mensajes

```
messages = folder.getMessages();
final String[] valores = new String[messages.length];
```

Paso 7: Añadir en la lista de mensajes

```
for(int i = 0; i<messages.length; i++){
        valores[i] = "" + (i+1) + ": "+  messages[i].getFrom()[0] + "-> "+messages[i].getSubject();
    sendMessage("email: "+valores[i]+"±");
}
```

Paso 8: Mostrar el mensaje seleccionado

```
msgEmailTxt.setText(messages[noMsg].getContent().toString());
```

Con este pequeño código se obtienen los mensajes del servidor a través del protocolo POP3. En la figura 4.5 se muestra la implementación de este módulo en los diferentes dispositivos.

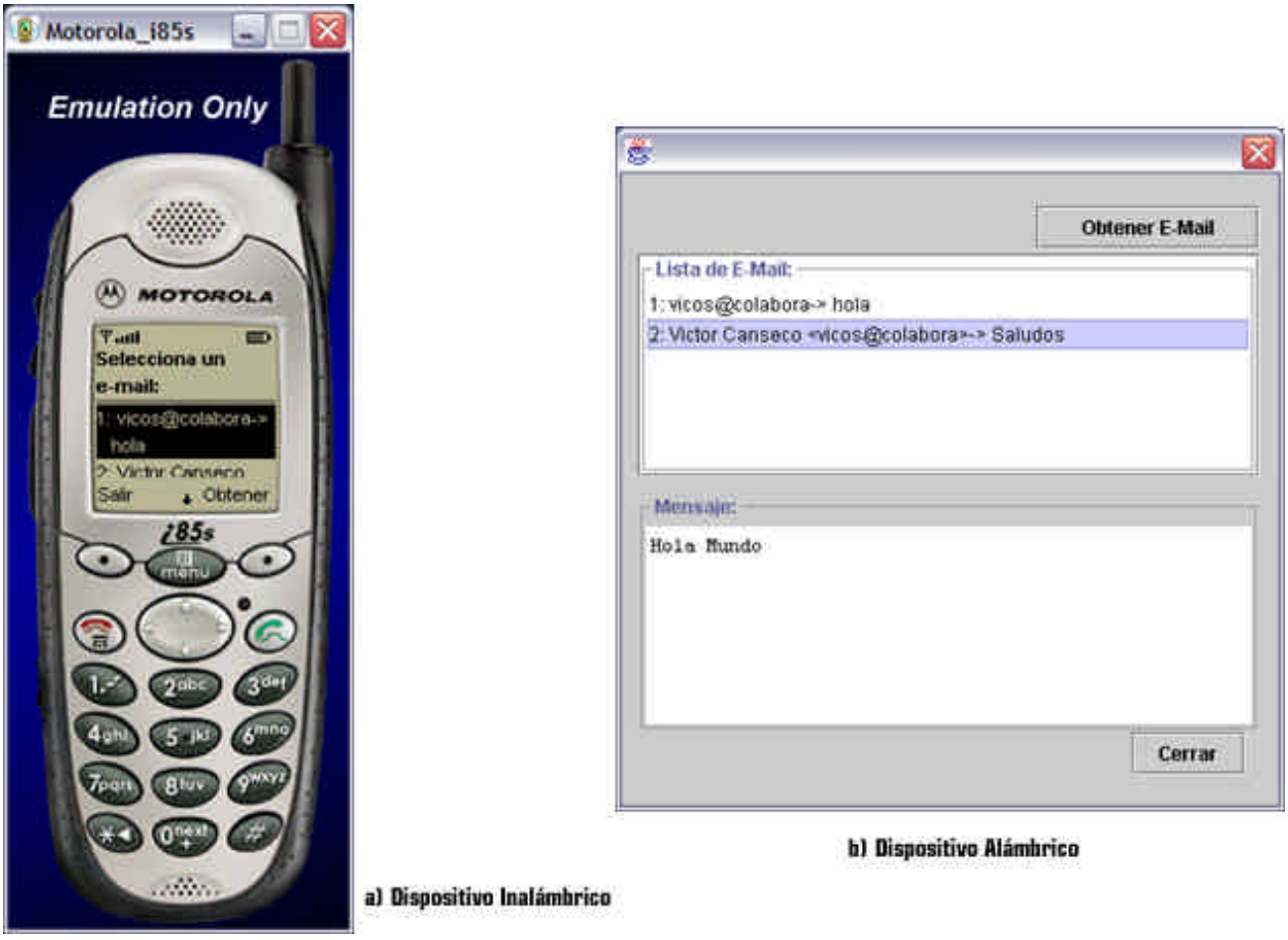

Figura 4.5 Módulo de leer e-mail

4.4.2 Enviar e-mail

Este módulo se encarga del envío de correos electrónicos a través del protocolo SMTP (Simple Mail Transfer Protocol), este es el protocolo mas utilizado hoy en día para las transferencias de correo electrónico, ya que es sencillo de utilizar. Para su implementación se emplean las mismas clases del módulo de lectura de e-mail. A continuación se enumeran los pasos para la implementación de este módulo.

Paso 1: Establecer las propiedades de la sesión

```
Properties props = new Properties();
```

Paso 2: Indicar el protocolo smtp

```
props.setProperty("mail.transport.protocol","smtp");
```

Paso 3: Configurar el servidor de correo

```
props.setProperty("mail.host","colabora");
```

Paso 4: Identificar al usuario que envía el correo

```
props.setProperty("mail.user",login);
```

Paso 5: Colocar el password de la cuenta

```
props.setProperty("mail.password",pwd);
```

Paso 6: Crear la sesión de email

```
Session mailSession = Session.getDefaultInstance(props,null);
```

Paso 7: Abrir el canal de comunicación

```
Transport transport = mailSession.getTransport("smtp");
MimeMessage message = new MimeMessage(mailSession);
String strTipoMsg = "text/html"
```

Paso 8: Añadir el asunto

```
message.setSubject(asunto);
```

Paso 9: Colocar la fecha de envío

```
message.setSentDate(new java.util.Date());
```

Paso 10: Colocar quien lo envía

```
InternetAddress addressFrom = new InternetAddress(login+"@colabora");
```

Paso 11: Colocar para quien va dirigido

```
message.setFrom(addressFrom);
message.addRecipient(Message.RecipientType.TO, new InternetAddress(destino));
```

Paso 12: Enviar una copia si esta se selecciono

```
if( !cc.equals( "" ) )
  message.addRecipient( Message.RecipientType.CC, new InternetAddress( cc ) );
```

Paso 13: Agregar el contenido

```
message.setContent(contenido,strTipoMsg);
```

Paso 14: Establecer la conexión

```
transport.connect("colabora",login,pwd);
```

Paso 15: Enviar el mensaje

```
transport.send(message);
```

Paso 16: Cerrar todas las sesiones de correo

```
transport.close();
```

Este código se implementó de la misma forma para las dos aplicaciones clientes y básicamente funcionan igual en las dos plataformas esto se puede apreciar en la figura 4.6 en donde se muestra la implementación de este módulo.

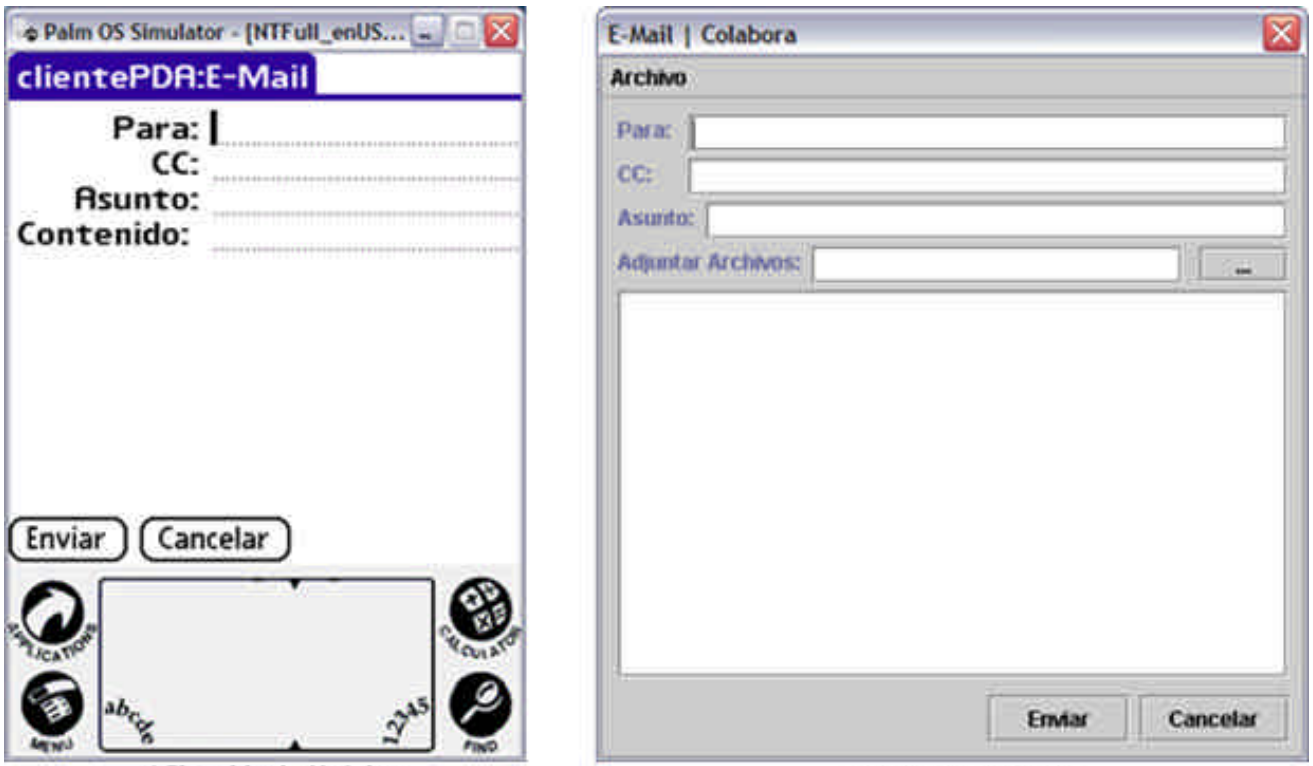

Figura 4.6 Módulo de enviar e-mail

4.5 Módulo Conversaciones

Este módulo es el encargado de comunicar a los usuarios móviles con los de oficina y viceversa. El principio básico de este módulo es un chat que se modificó para soportar usuarios móviles a través de Bluetooth, esto se puede apreciar mejor al analizar los métodos para el envío de mensajes.

Primero tenemos el método sendToAll del btServer.

```
public void sendToAll(String message){
    synchronized( outputStreams ){
      //para cada cliente
      for(Enumeration e = getOutputStreams(); e.hasMoreElements();){
        OutputStream out =  (OutputStream) e.nextElement();
        try{
          byte[] b = message.getBytes();
          out.write(b);
        }catch(IOException ioe){
          output.setText(output.getText()+"\n"+ioe.toString());
        }
      }
    }
  }
```

Método sendToAll del chatServer

```
public void sendToAll( String message ){
    //sincronizamos los hilos de ejecucion
    synchronized( outputStreams ){
      //para cada cliente.....
      for( Enumeration e = getOutputStreams(); e.hasMoreElements(); ){
```

```
        //... se obtiene el output stream
        DataOutputStream dout = ( DataOutputStream )e.nextElement();
        //se envia el mensaje
        try{
          dout.writeUTF( message );

        }catch( IOException ie ){
          status.setText( status.getText() + "Error: " + ie.getMessage() );
        }
      }
    }
  }
```

Se puede apreciar claramente la diferencia, en el btServer se emplea un OutputStream y en el chatServer se emplea un DataOutputStream, los dos objetos derivan de la clase Stream sólo que uno es para conexiones inalámbricas y el otro para alámbricas. Este módulo ya implementado y en funcionamiento se muestra en la figura 4.7.

a) Dispositivo Inalámbrico b) Dispositivo Inalámbrico c) Dispositivo Alámbrico

Figura 4.7 Módulo de Conversaciones

4.6 Módulo de Reuniones

Este módulo es el encargado de realizar las reuniones de los usuarios, para poder realizar una reunión se necesita saber la fecha y la hora en la que esta se llevará a cabo, además de los participantes de la misma. Una vez definido esto el sistema envía un correo electrónico a cada una de las personas involucradas con su login y password de la reunión. Ya que este módulo hace uso del módulo de conversaciones la interfaz es prácticamente la misma nada más que este agrega el uso del módulo de votaciones. La implementación de este módulo se aprecia en la figura 4.8.

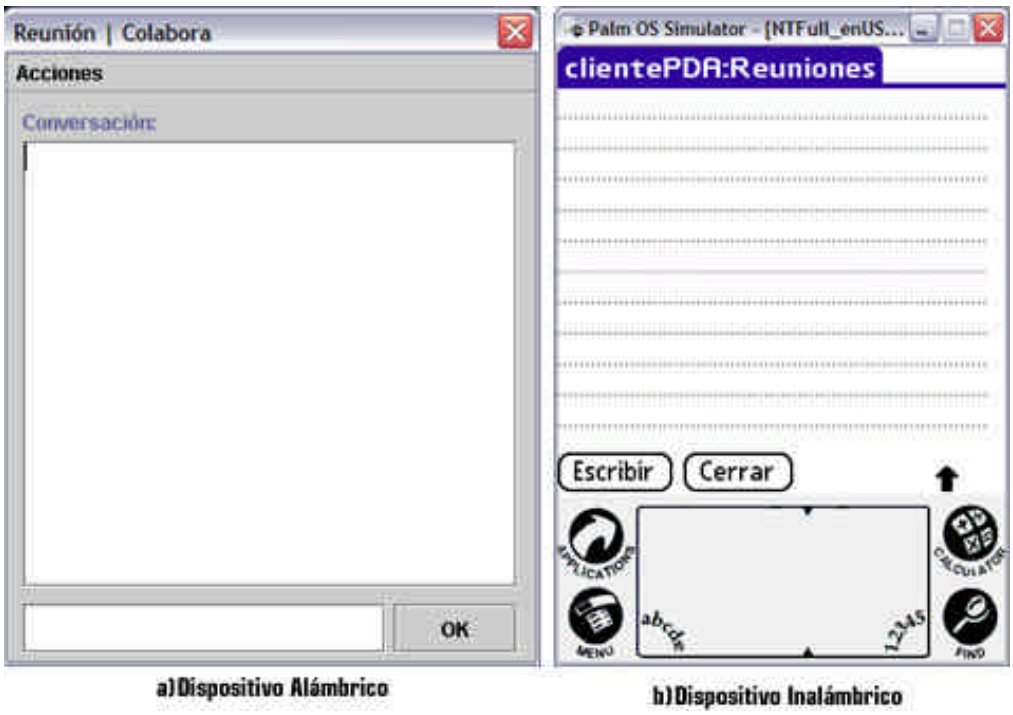

Figura 4.8 Módulo de Reuniones

4.7 Módulos para el control de la base de datos

Estos módulos son complementos del sistema, se encargan de realizar las operaciones en la base de datos. Su programación en JSP/Servlets permite que se utilice cualquier navegador de Internet para acceder a la aplicación. Los módulos son: alta, baja y modificación de usuarios, alta, baja y modificación de grupos y por último el módulo de reuniones. En la figura 4.9 podemos apreciar la pantalla de inicio del sistema.

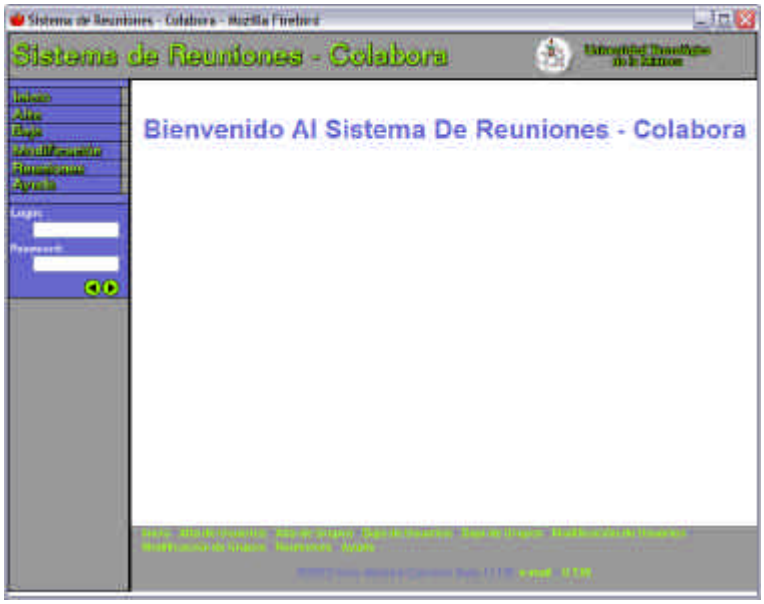

Figura 4.9 Pantalla principal del administrador de la B.D

4.7.1 Alta de Usuarios

Módulo encargado de capturar los datos de los empleados o personas que vayan a utilizar el sistema. En la figura 4.10 se aprecia el módulo.

Figura 4.10 Módulo de alta de usuarios

4.7.2 Módulo de Alta de Grupos

Módulo encargado de capturar los grupos existentes en la organización. En la figura 4.11 se muestra este módulo.

Figura 4.11 Módulo de alta de grupos

4.7.3 Módulo de Baja de Usuarios

Módulo encargado de eliminar a un usuario de la base de datos. Este módulo se aprecia en la figura 4.12.

Figura 4.12 Módulo de Baja de Usuarios.

4.7.4 Módulo de Baja de Grupos

Módulo encargado de eliminar a un grupo de la base de datos. Este módulo se aprecia en la figura 4.13.

Figura 4.13 Módulo de Baja de Grupos

4.7.5 Módulo de Modificación de Usuarios

Módulo encargado de modificar los datos de un usuario en la base de datos. Este módulo se aprecia en la figura 4.14.

Figura 4.14 Módulo de Modificación de Usuarios

4.7.6 Módulo de Reuniones

Módulo encargado de crear una reunión en la base de datos. Este módulo se aprecia en la figura 4.15.

Figura 4.15 Módulo de Reuniones

4.8 Ejecutando el sistema

Para poder realizar las pruebas necesarias al sistema necesitamos ejecutar el simulador de Bluetooth de RococoSoftware, el cual se puede apreciar en la figura 4.16. Una vez iniciado el simulador se puede proceder a ejecutar las aplicaciones.

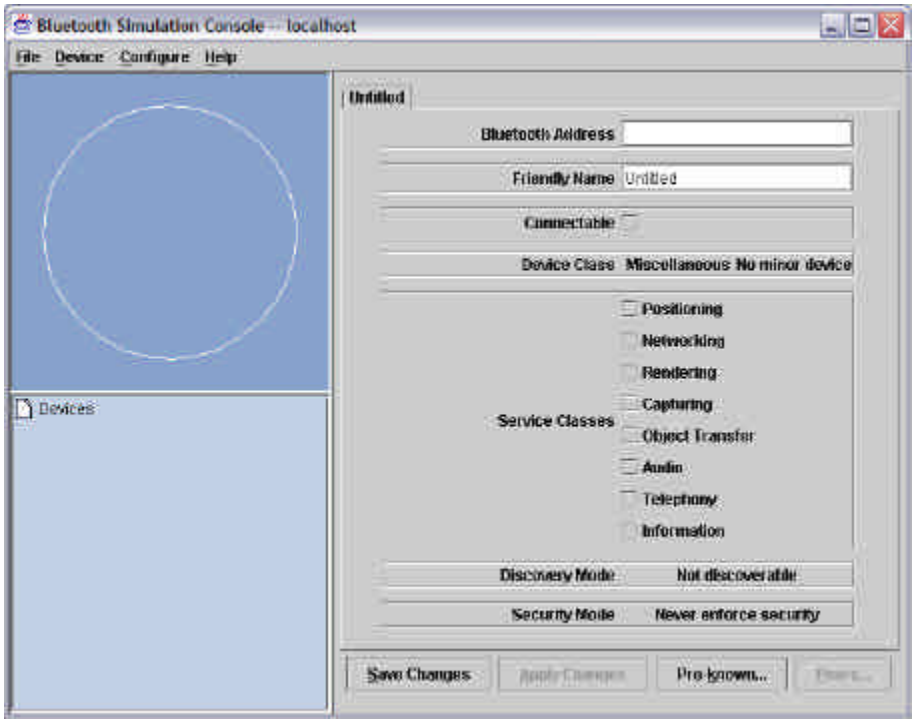

Figura 4.16 Simulador de Bluetooth

4.8.1 Iniciando

Comenzaremos con el servidor, para ello ejecutamos el archivo ServidorBT.bat después de esto aparecerá una ventana de bienvenida y comenzaremos a utilizar el servidor. Damos click en el menú Servidor y de ahí iniciar, o más fácil Ctrl+S, esto se puede apreciar en la figura 4.17. A continuación mostrará unos mensajes en la parte de status, para saber si todos los servicios fueron aceptados correctamente.

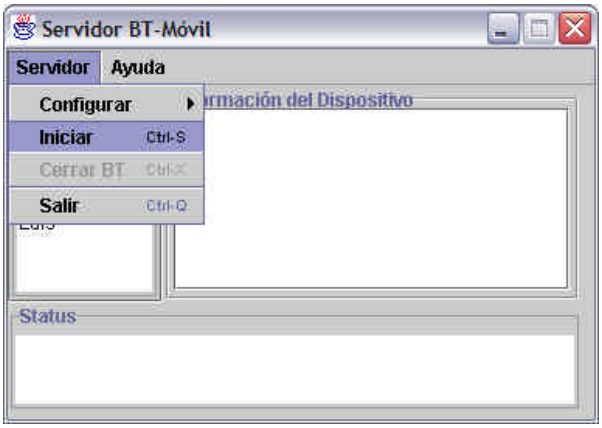

Figura 4.17 Servidor BT

Terminada la carga del sistema se puede ver el resultado de inicializar el servicio de Bluetooth en el simulador tal y como se muestra en la figura 4.18.

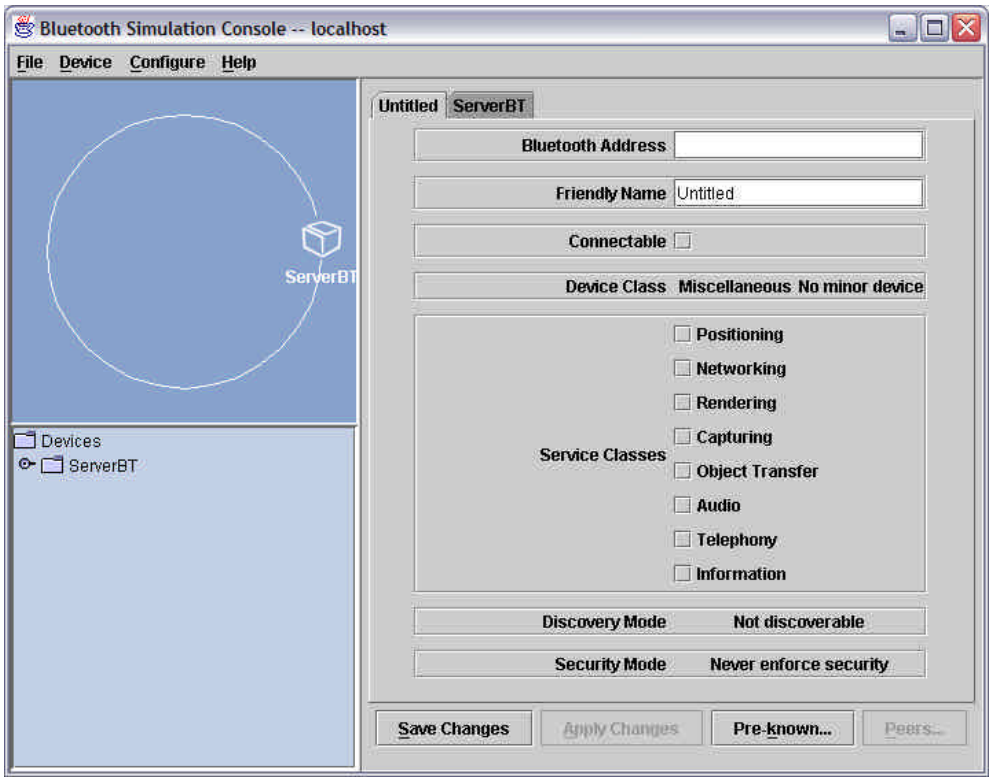

Figura 4.18 Servidor BT iniciado

Una vez que el servidor aparezca en el simulador podemos ya iniciar cualquiera de las aplicaciones clientes, en este caso iniciaremos ambas. Comenzaremos con la aplicación para PDA, damos click en launch y aparecerá la pantalla de bienvenida del sistema, una vez ahí damos click en conexión, se muestra una ventana para capturar el login, password y grupo de trabajo y damos click en Ok, si todo salió satisfactoriamente nos mostrará un mensaje de envió de datos y volveremos a la pantalla inicial, esto se puede ver en la figura 4.19.

Figura 4.19 Conexión al servidor

Si la aplicación se logró enlazar correctamente con el servidor entonces el simulador mostrará una ventana como la figura 4.20.

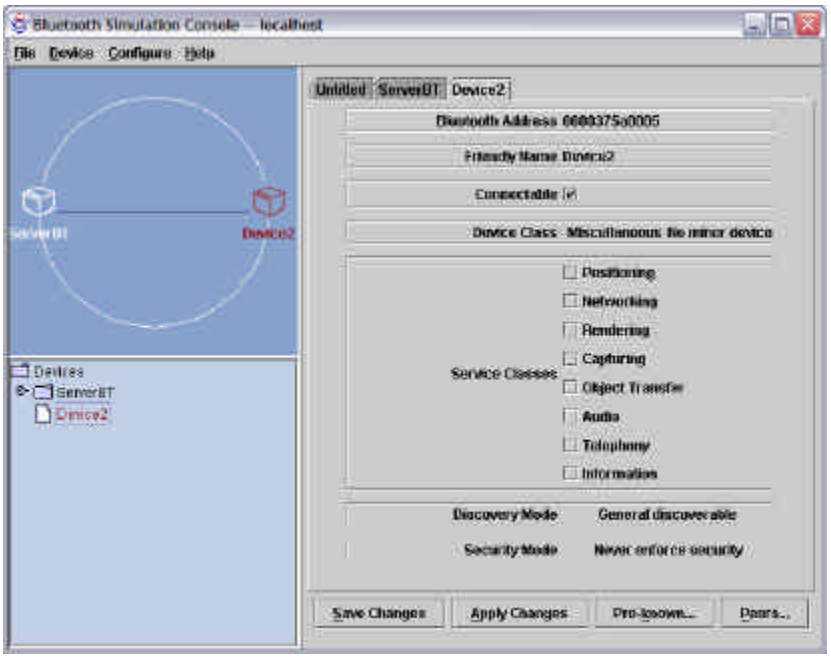

Figura 4.20 Dispositivo enlazado al servidor BT

A continuación podemos iniciar la aplicación cliente. Dirigiéndose a archivo->iniciar sesión o Ctrl+O, mostrará una ventana de captura de datos, introducimos el login, el password y el grupo por último le damos OK. En la figura 4.21 se observa el módulo del cliente.

Figura 4.21 Aplicación cliente para PC

Cabe mencionar que una vez que el cliente fue aceptado el servidor sólo muestra un mensaje en la ventana de status que acepto la conexión, esto no se refleja en el simulador ya que esta conexión es alámbrica por medio de sockets no por medio de Bluetooth.

4.8.2 Módulo chat

Seleccionando la opción conversaciones en el PDA se mostrará una ventana igual a la que se tiene en la figura 4.22 (a), al dar click en Escribir, se muestra una ventana en donde se puede escribir el mensaje que queremos enviar a todos los usuarios, figura 4.22 (b).

Figura 4.22 a) Módulo de conversaciones, b) Escritura de mensajes.

En la aplicación de PC damos click en Acciones->chat o Ctrl+C, se muestra una ventana de envío de mensajes, escribimos el mensaje y seleccionamos Enviar o Enter para el envío a todos los usuarios dicho mensaje, como en la figura 4.23. En la figura 4.24 se puede observar el mensaje enviado.

Figura 4.23 Módulo de conversaciones

Figura 4.24 Mensaje

Respondemos a través de escribir y le damos OK como se muestra en la figura 4.25, la respuesta se da en el cliente de PC como se muestra en la figura 4.26.

Figura 4.25 Respuesta

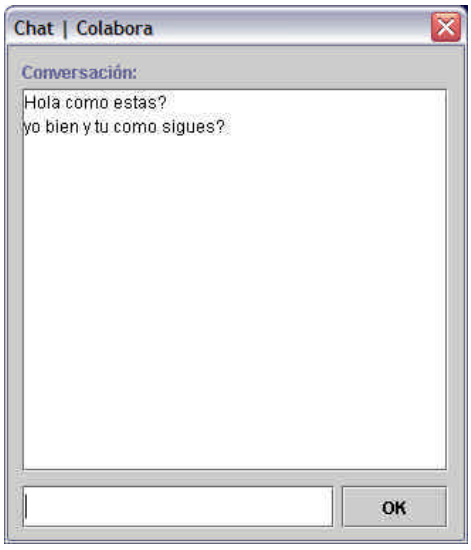

Figura 4.26 Respuesta del cliente

4.8.3 Módulo de E-mail

El módulo de e-mail se encuentra dividido en dos submódulos, el módulo de escritura y el módulo de lectura. El módulo de escritura es el encargado de editar y enviar un correo electrónico. En la aplicación de PC, dirigiéndose al menú Acciones->E-mail se muestra una ventana como en la figura 4.27.

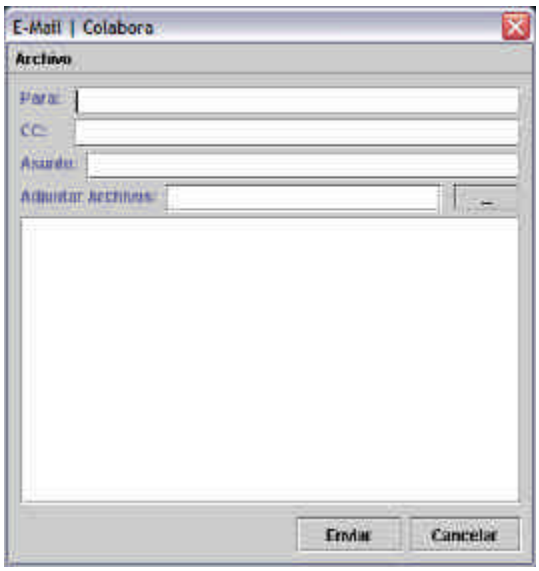

Figura 4.27 Módulo de e-mail

Podemos apreciar en esta ventana los siguientes campos:

- Para. Que es el destinatario.
- CC. Con copia para.
- Asunto. El asunto del correo.
- Adjuntar Archivos. En este campo indicamos que archivo vamos a enviar.
- Y un área de texto para el contenido del correo.

En la figura 4.28 vemos el módulo de escritura y envío de correo electrónico de los dispositivos móviles. Se puede apreciar que la aplicación móvil no tiene el campo de adjuntar archivo, debido a que las clases de java para Bluetooth disponibles no tienen el soporte para el intercambio de objetos.

Figura 4.28 Módulo e-mail del dispositivo móvil

En el módulo de lectura los usuarios pueden revisar sus correos electrónicos sin importar el lugar en el que se encuentren. En la figura 4.29 se aprecia la aplicación de escritorio, con el botón Obtener E-Mail

se accede a los correos que se encuentran almacenados en el servidor. En la figura 4.30 se muestra el mismo módulo pero en su versión para dispositivo portátil.

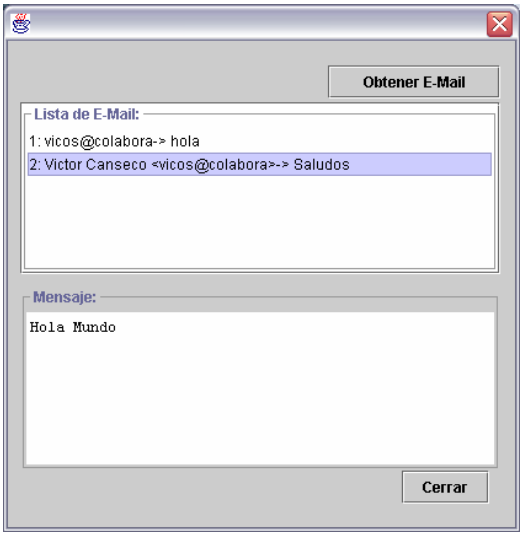

Figura 4.29 Módulo de lectura de e-mail para la PC

Figura 4.30 Módulo de lectura de e-mail

4.8.4 Módulo de Reuniones

Este módulo se utiliza para que los participantes de las reuniones se puedan comunicar, en la figura 4.31 se aprecia este módulo en la versión móvil. Si deseamos enviar un mensaje a los asistentes le damos click al botón de escribir y aparece una ventana similar a la figura 4.32. Al dar click en OK el mensaje se envía a todos los participantes de la reunión como se observa en la figura 4.33.

Figura 4.31 Módulo de Reuniones

Figura 4.32 Ventana de envío de mensajes en la reunión

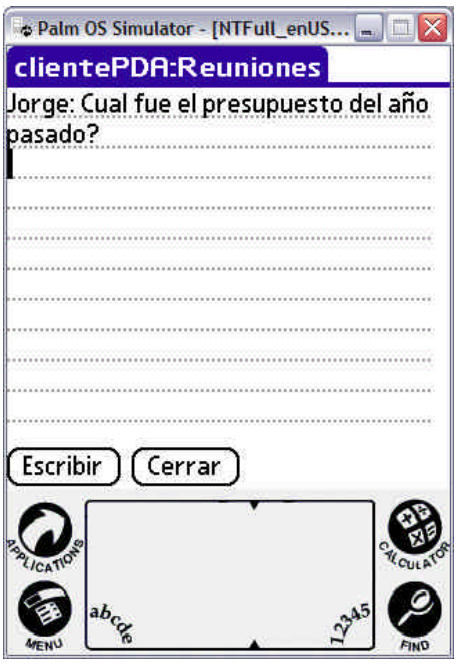

Figura 4.33 Resultado del envío del mensaje

En la aplicación de escritorio se debe de seleccionar Acciones->Reunión o Ctrl+R para entablar comunicación con los participantes de la reunión. La ventana de reuniones es como se observa en la figura 4.34, igual que en la aplicación móvil se puede enviar un mensaje a los asistentes a la reunión, esto se aprecia en la figura 4.35.

Figura 4.34 Ventana del módulo de Reunión para la PC

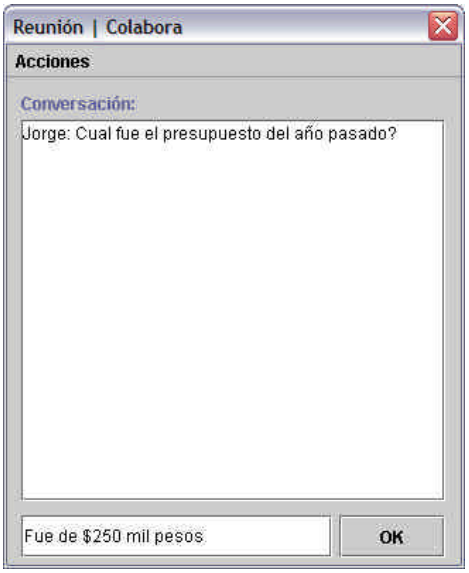

Figura 4.35 Mensaje de las reuniones

Por último se analizan los resultados obtenidos y el desarrollo futuro de este trabajo.

Capítulo 5 Conclusiones y Trabajo futuro

5

5.1 Conclusiones

El objetivo principal de este proyecto, que fue el desarrollo de un sistema de reuniones para PC's y dispositivos móviles se cumplió en su totalidad. El sistema permite comunicar a varios clientes de PC's conectados a través de una red LAN y acceder a los distintos servicios del sistema. La simulación de la aplicación funciona en su totalidad y permite comunicar a los clientes móviles con los diferentes usuarios de PC.

Este sistema puede ser implementado en cualquier organización que proporcione un ambiente grupal, con el cual se pueden realizar tareas simples como son la lectura y escritura de correo electrónico, conversaciones con compañeros y se pueden realizar reuniones de trabajo. La ventaja de utilizar la tecnología java es su portabilidad y con esta característica se pueden hacer uso de teléfonos celulares que tengan la tecnología Bluetooth. Esto permite utilizar los teléfonos de los usuarios que cumplan con esta característica reduciendo el costo de equipamiento. Por otro lado si se desea se pueden emplear dispositivos con mayores y mejores características que los teléfonos como son los PDA's dado que existen máquinas virtuales para ejecutar aplicaciones java en Palm's, Pocket PC's u otros dispositivos como los RIM de Blackberry. Con estos dispositivos se tiene un mejor poder de cómputo permitiendo a los usuarios almacenar información en los mismos.

Este sistema puede ser implantado en la Universidad Tecnológica de la Mixteca en cualquiera de sus áreas, debido a que se pueden definir grupos de trabajo y permite la comunicación de los empleados sin importar el lugar en el que se encuentren.

Dentro de las restricciones que tendría el sistema una vez implementado en el mundo real, se encuentra el alcance de la conexión inalámbrica, el mayor problema seria en el módulo de reuniones ya que el administrador coloca la restricción de la hora de entrada, si un usuario móvil perdiera su conexión e intentará volver a entrar a la reunión, el sistema lo rechazaría y por lo tanto la reunión se vería afectada ya que si se llevara a cabo una reunión la persona que perdió su conexión no podría dar su voto.

Otra restricción del sistema es la capacidad de almacenamiento del dispositivo móvil esto debido a la gran variedad de modelos existentes en el mercado, lo cual restringe la escalabilidad del sistema, limita el numero máximo de correos almacenados y además de no permitir el envió de archivos adjuntos en los correos electrónicos.

El sistema es susceptible a fallas ocasionadas por la conexión LAN entre los clientes y la BD, si la conexión no es muy estable el sistema colapsa totalmente con lo cual es necesario reiniciar el sistema.

5.2 Trabajo Futuro

El presente trabajo sólo quedo como una simulación, como ya se mencionó esta aplicación se puede llevar al mundo real, comprando una licencia para el desarrollo de Bluetooth, otra alternativa sería desarrollar las librerías (ya sea en java o en cualquier otro lenguaje que pueda se portado a dispositivos móviles) y no pagar por dichas licencias, tomando como base la documentación de Bluetooth que se encuentra libre.

Debido a la forma modular en la que fue programado se pueden añadir nuevas características al sistema como son los módulos para la edición de documentos en forma compartida, sistemas de lluvia de ideas, etc.

Para solucionar uno de los problemas ocasionados por la pérdida de la señal en el módulo de reuniones se puede modificar el sistema para que una vez que el usuario ya se registró en la reunión, se elimine el límite de tiempo y con esto permitir al usuario regresar a la reunión sin que se afecte su trabajo.

Debido a que el sistema de reuniones desarrollado es un sistema distribuido, se puede llevar a cabo un análisis y modelado de concurrencia, al utilizar UML analizamos y modelamos la funcionalidad del sistema pero no se adentra en detalle al análisis de cada uno de los procesos que se ejecutan dentro del sistema y por lo tanto no es posible saber si dos o más procesos pudieran tener algún conflicto entre ellos, como por ejemplo el intentar acceder a un recurso al mismo tiempo, o que un proceso nunca se ejecute, etc. UML se puede emplear para el análisis de los requerimientos y descripción de la funcionalidad del sistema y el modelo de la arquitectura dinámica del sistema se puede realizar empleando algún leguaje de especificación formal como CCS (Cálculo de Sistemas de Comunicación-Calculus of Communicating Systems) y CSP (Comunicación de Procesos Secuenciales-Communicating of Sequence Process). CCS permite especificar la relación de la distribución temporal entre las interacciones que constituyen el comportamiento externo del sistema y CSP es un modelo para programar, en el cual cada proceso tiene su propio sitio de control, y los procesos secuenciales se comunican intercambiando mensajes.

Para proveer de mayor seguridad al sistema se puede encriptar la información que se intercambia por medio de correo electrónico, esto se puede lograr empleando la herramienta PGP que permite encriptar y desencriptar correos electrónicos utilizando el algoritmo RSA.

Para evitar fallas ocasionadas por la conexión a la base de datos se puede crear una BD distribuida, con lo cual es sistema se haría más estable y funcional.

Bibliografía

[1] Allen Rob.Workflow: An Introduction. Open Image Systems Inc.

[2] Beacker, Ronald y Posner, Ilona. "How People Write Together",Readings in Groupware and Computer-Suport Cooperative Work. Morgan Kaufmann Publishers, 1993, pp.239-250.

[3] Bray, Jennifer y Sturman, Charles. Bluetooth connect without cables. Ed. Prentice Hall.

[4] Chaffey Dave. Groupware, Workflow and Intranets. Reengineering the Enterprise with Collaborative Software. Ed. Digital Press, 1998.

[5] Deitel, Harvey y Deitel, Paul. Java, Prentice Hall, 2001.

[6] Evans. Dean. Agenda Digitales de Bolsillo. Revista T3 México, No. 21. Ed. Mina Editores. Pag. 30-38.

[7] Fernández y Fernández, Carlos Alberto. Programación Orientada a Objetos II. Notas de Clase. Universidad Tecnológica de la Mixteca, 2000.

[8] Flower, Martín. UML, gota a gota. Addison Wesley. Mexico, 1999.

[9] Hall, Marty. Servlets y Java Server Pages. Prentice Hall. 2002.

[10] Hopkins, Bruce y Antony, Ranjith. Bluetooth for Java. A! Press, 2003.

[11] Kroll, Michael y Haustein, Stefan. Java 2 Micro Edition Application Development. Sams Publishing, 2002.

[12] Lee, J. SIBYL: A tool for managing group decision rationale. CSCW'90 Proceedings, 1990, pp. 79-92.

[13] Revista Emprendedores, No.57. Para que Sirven los PDA's. Pag. 148-152. Ed. Hachette Filipacchi.

[14] WfMC. Workflow and Internet: Catalysts for radical change, white paper, 1998.

[15] Yi-Bing Lin, Imrich Chlamtac. Wireless and Mobile Network Architectures. Ed. Wiley.

URL'S

[URL 1] Compaq Mexico. http://www.compaq.com.mx, Último acceso: 19 de Febrero de 2003.

[URL 2] GFi Fax & Voice Ltd. Workflow Technology, http://www.gficomms.com. Último acceso: Julio de 2002

[URL 3] HP Mexico. http://www.hp.com.mx. Último acesso: 19 de Febrero de 2003.

[URL 4] Lopez Ortiz., Francisco.Wireless LAN. White Paper. http://greco.dit.upm.es/~david/TAR/trabajos2002/ 08-802.11-Francisco-Lopez-Ortiz.pdf. Último acceso: Noviembre de 2003.

[URL 5] Palm Computing Inc. Company History. http://www.palm.com. Último acceso: Marzo de 2003.

[URL 6] Rational Software. http://www.rational.com. Último acceso: Mayo del 2002.

[URL 7] Romero Perez, Flavia. Sistema Colaborativo Para el Apoyo Electrónico a Reuniones. 1997. http://www2.ing.puc.cl/~group/sisco/indice.html. Último acceso: Noviembre de 2003

[URL 8] RococoSoftware. http://www.rococosoft.com, Ultimo acceso: Junio de 2003

[URL 9] Sony Corp. http://www.sony.com. Último acceso: 19 de Febrero de 2003.

[URL 10] Sun Microsystems. http://www.sun.com. Último acceso: Octubre de 2003.

[URL 11] WfMC.Workflow Reference Model, white paper, 1995. http://www.wfmc.org. Último acceso: Julio del 2002

Apéndice A. Métodos y Objetos

Tabla A.1 Métodos y Objetos de la clase btServer

Métodos	Descripción
public btServer(JTextArea output, chatServer chatpc)	Constructor de la clase. Se encarga de inicializar los objetos.
Enumeration getOutputStreams()	Método que devuelve los Streams que están conectados.
public void listen()	Método para escuchar a los clientes.
void removeConnection(StreamConnection conn)	Método para eliminar una conexión.
public void run()	Método principal de la clase. Es el encargado de ejecutar la conexión.
public void sendMessage(String message)	Método para enviar un mensaje a un cliente.
public void sendToAll(String message)	Método para enviar un mensaje a todos los clientes.
Objetos	**Descripción**
private chatServer chatPC;	Objeto para establecer comunicación con los clientes móviles.
private LocalDevice device;	Objeto para obtener el dispositivo local.
private JTextArea output	Objeto para mostrar los mensajes de error y alertas.
private Hashtable outputStreams	Objeto para almacenar las conexiones con los clientes.
private OutputStream salida;	Objeto para el Stream de salida.
public static final UUID uuid	Objeto para almacenar el UUID (Universal Unique IDentifier) del dispositivo.

Tabla A.2 Métodos y Objetos de la clase btServerThread

Métodos	Descripción
public btServerThread(btServer btserver, chatServer chatPC,StreamConnection conn, JTextArea output,InputStream input,OutputStream salida)	Constructor de la clase es el encargado de inicializar los objetos.
private void readFrom(InputStream input)	Método para leer la información de un Stream.
public void run()	Método principal de la clase. Es el encargado de manipular la información solicitada por los dispositivos.
private void sendMail(String destino, String cc, String asunto, String contenido)	Método para que los dispositivos móviles envíen correos electrónicos.
public void sendMessage(String message)	Método para enviar un mensaje a un dispositivo.
Objetos	**Descripción**
private btServer btserver	Objeto para la comunicación con los demás dispositivos.
private StreamConnection conn	Objeto para la conexión por medio de Streams.

private JTextArea output	Objeto para la mostrar los mensajes de error y las alertas del sistema.
private chatServer chatPC	Objeto para la comunicación con los clientes de PC.
private InputStream input	Objeto stream de entrada.

Tabla A.3 Métodos y Objetos de la clase chatServer

Métodos	Descripción
public chatServer(int port, JTextArea status)	Constructor de la clase chatServer. Es el encargado de inicializar los objetos para su utilización.
Enumeration getOutputStreams()	Método que devuelve los Streams que están conectados.
public void listen()	Método para escuchar a los clientes.
void removeConnection(StreamConnection conn)	Método para eliminar una conexión.
public void run()	Método principal de la clase, es el encargado de ejecutar la conexión.
public void sendBT(String message)	Método para enviar un mensaje a un cliente móvil.
public void sendToAll(String message)	Método para enviar un mensaje a todos los clientes.
Objetos	**Descripción**
private btServer bts	Objeto para la comunicación con los móviles.
private ServerSocket ss	Objeto para los Sockets tipo servidor.
private JTextArea status	Objeto para mostrar los mensaje del servidor.
DataOutputStream private Hashtable outputStreams	Objeto para almacenar las conexiones existentes.
private int port	Objeto del Puerto para la comunicación.

Tabla A.4 Métodos y Objetos de la clase ServerThread

Métodos	Descripción
public ServerThread(chatServer server,btServer bts, Socket socket, JTextArea status)	Constructor de la clase btServerThread. Es el encargado de inicializar los objetos para su utilización.
public void run()	Método principal de la clase. Es el encargado de inicializar y establecer la conexión con los clientes.
Objetos	**Descripción**
private chatServer server	Este objeto es el encargado de realizar la comunicación con los clientes de PC.
private btServer btserver	Objeto del servidor para la comunicación entre los clientes de PC y móviles.
private Socket socket	Objeto Socket para la conexión.
private JTextArea status	Objeto para desplegar los mensajes de error y confirmación.

Apéndice B. Código Fuente de las clases básicas

Clase Principal serverBT1.

```
package ServerBluetooth;

import javax.swing.*;
import java.awt.Toolkit;
import java.sql.*;
import java.io.*;
import java.awt.Dimension;
import javax.Bluetooth.*;
import com.rococosoft.impronto.configuration.*;
import com.rococosoft.impronto.util.BluetoothConstants;
import org.gjt.mm.mysql.*;

/** Clase principal del servidor de Bluetooth para el sistema <B>Colabora</B>
 * @author Victor M. Canseco Soto
 * @version 1.0
 * @Copyright Universidad Tecnológica de la Mixteca.
 * @Date Junio/2003
 */
public class serverBT1 extends javax.swing.JFrame {

    chatServer chatserver;
  reunionServer reunionserver;
  /** Constructor predeterminado, para serverBT1
   */
  public serverBT1() {
    createSplashScreen();
    showSplashScreen();
    initComponents();
    hideSplash();
  }

  /** Este método es el encargado de inicializar los componentes de la aplicación.
   */
  private void initComponents() {//GEN-BEGIN:initComponents
    acercaDlg = new javax.swing.JDialog();
    jLabel1 = new javax.swing.JLabel();
    jLabel2 = new javax.swing.JLabel();
    jLabel3 = new javax.swing.JLabel();
    jLabel4 = new javax.swing.JLabel();
    okBtn = new javax.swing.JButton();
    jLabel5 = new javax.swing.JLabel();
    jScrollPane1 = new javax.swing.JScrollPane();
    listaDispositivos = new javax.swing.JList();
    infoPanel = new javax.swing.JPanel();
    infoScroll = new javax.swing.JScrollPane();
    infoTexto = new javax.swing.JTextPane();
    jScrollPane2 = new javax.swing.JScrollPane();
    status = new javax.swing.JTextArea();
    menu = new javax.swing.JMenuBar();
```

```
menuServidor = new javax.swing.JMenu();
configMenu = new javax.swing.JMenu();
confNomb = new javax.swing.JMenuItem();
noPuertoChat = new javax.swing.JMenuItem();
noPuertoReunion = new javax.swing.JMenuItem();
iniciar = new javax.swing.JMenuItem();
jSeparator1 = new javax.swing.JSeparator();
salir = new javax.swing.JMenuItem();
menuAyuda = new javax.swing.JMenu();
ayuda = new javax.swing.JMenuItem();
acercade = new javax.swing.JMenuItem();

acercaDlg.getContentPane().setLayout(null);

acercaDlg.setTitle("Acerca de ...");
jLabel1.setText("Acerca de ServerBT-Mov\u00edl");
jLabel1.setFont(new java.awt.Font("Arial", 1, 18));
acercaDlg.getContentPane().add(jLabel1);
jLabel1.setBounds(20, 10, 250, 22);

jLabel2.setText("Proyecto de T\u00e9sis  Realizado por:");
acercaDlg.getContentPane().add(jLabel2);
jLabel2.setBounds(40, 40, 200, 17);

jLabel3.setText("V\u00edctor Mois\u00e9s Canseco Soto.");
acercaDlg.getContentPane().add(jLabel3);
jLabel3.setBounds(50, 70, 170, 17);

jLabel4.setText("Universidad Tecnol\u00f3gica de la Mixteca.");
acercaDlg.getContentPane().add(jLabel4);
jLabel4.setBounds(30, 90, 230, 20);

okBtn.setText("OK");
okBtn.addActionListener(new java.awt.event.ActionListener() {
  public void actionPerformed(java.awt.event.ActionEvent evt) {
    okBtnActionPerformed(evt);
  }
});

acercaDlg.getContentPane().add(okBtn);
okBtn.setBounds(100, 140, 81, 27);

jLabel5.setText("2003.");
acercaDlg.getContentPane().add(jLabel5);
jLabel5.setBounds(120, 110, 41, 17);

getContentPane().setLayout(null);

setTitle("Servidor BT-M\u00f3vil");
setName("servidorFrame");
setResizable(false);
addWindowListener(new java.awt.event.WindowAdapter() {
  public void windowClosing(java.awt.event.WindowEvent evt) {
    exitForm(evt);
  }
});

jScrollPane1.setViewportBorder(new javax.swing.border.TitledBorder("Dispositivos"));
listaDispositivos.setModel(new javax.swing.AbstractListModel() {
```

```
        String[] strings = { "Carlos", "Victor", "Gabriel", "Luis" };
        public int getSize() { return strings.length; }
        public Object getElementAt(int i) { return strings[i]; }
    });
    listaDispositivos.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
    listaDispositivos.addListSelectionListener(new javax.swing.event.ListSelectionListener() {
        public void valueChanged(javax.swing.event.ListSelectionEvent evt) {
            listaDispositivosValueChanged(evt);
        }
    });

    jScrollPane1.setViewportView(listaDispositivos);

    getContentPane().add(jScrollPane1);
    jScrollPane1.setBounds(0, 0, 100, 140);

    infoPanel.setLayout(null);

    infoPanel.setBorder(new javax.swing.border.TitledBorder(new javax.swing.border.EtchedBorder(), "Informaci\u00f3n del
Dispositivo"));
    infoScroll.setViewportView(infoTexto);

    infoPanel.add(infoScroll);
    infoScroll.setBounds(7, 15, 255, 117);

    getContentPane().add(infoPanel);
    infoPanel.setBounds(100, 0, 270, 140);

    jScrollPane2.setBorder(new javax.swing.border.TitledBorder("Status"));
    status.setLineWrap(true);
    status.setEditable(false);
    status.setForeground(new java.awt.Color(0, 0, 0));
    status.setFont(new java.awt.Font("Dialog", 0, 10));
    status.setBackground(new java.awt.Color(255, 255, 255));
    jScrollPane2.setViewportView(status);

    getContentPane().add(jScrollPane2);
    jScrollPane2.setBounds(0, 140, 370, 70);

    menuServidor.setText("Servidor");
    configMenu.setText("Configurar");
    confNomb.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_N,
java.awt.event.InputEvent.CTRL_MASK));
    confNomb.setText("Nombre del Servidor");
    confNomb.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            confNombActionPerformed(evt);
        }
    });

    configMenu.add(confNomb);
    noPuertoChat.setToolTipText("No. Puerto Chat");
    noPuertoChat.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_P,
java.awt.event.InputEvent.CTRL_MASK));
    noPuertoChat.setText("No. Puerto Chat");
    noPuertoChat.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            noPuertoChatActionPerformed(evt);
        }
```

```
    });

    configMenu.add(noPuertoChat);
    noPuertoReunion.setToolTipText("No. Puerto Reunion");
    noPuertoReunion.setText("No. Puerto Reunion");
    noPuertoReunion.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
        noPuertoReunionActionPerformed(evt);
      }
    });

    configMenu.add(noPuertoReunion);
    menuServidor.add(configMenu);
    iniciar.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_S,
java.awt.event.InputEvent.CTRL_MASK));
    iniciar.setText("Iniciar");
    iniciar.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
        iniciarActionPerformed(evt);
      }
    });

    menuServidor.add(iniciar);
    menuServidor.add(jSeparator1);
    salir.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_Q,
java.awt.event.InputEvent.CTRL_MASK));
    salir.setText("Salir");
    salir.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
        salirActionPerformed(evt);
      }
    });

    menuServidor.add(salir);
    menu.add(menuServidor);
    menuAyuda.setText("Ayuda");
    ayuda.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_F1, 0));
    ayuda.setText("Ayuda");
    menuAyuda.add(ayuda);
    acercade.setText("Acerca de ...");
    acercade.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
        acercadeActionPerformed(evt);
      }
    });

    menuAyuda.add(acercade);
    menu.add(menuAyuda);
    setJMenuBar(menu);

    pack();
    java.awt.Dimension screenSize = java.awt.Toolkit.getDefaultToolkit().getScreenSize();
    setSize(new java.awt.Dimension(384, 270));
    setLocation((screenSize.width-384)/2,(screenSize.height-270)/2);
  }//GEN-END:initComponents

  private void noPuertoReunionActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_noPuertoReunionActionPerformed
    String num = ( String )JOptionPane.showInputDialog(this,"Número de Puerto para Chat:","Configuración del Servidor",
```

```
            JOptionPane.PLAIN_MESSAGE, null, null, String.valueOf( noPuerto ));
          if (num != null) {
      noReunion = Integer.parseInt( num );
      noPuertoReunion.setEnabled(false);
    }
  }//GEN-LAST:event_noPuertoReunionActionPerformed

  private void noPuertoChatActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_noPuertoChatActionPerformed
    String num = ( String )JOptionPane.showInputDialog(this,"Número de Puerto para Chat:","Configuración del Servidor",
            JOptionPane.PLAIN_MESSAGE, null, null, String.valueOf( noReunion ));
          if (num != null) {
      noReunion = Integer.parseInt( num );
      noPuertoChat.setEnabled(false);
    }
  }//GEN-LAST:event_noPuertoChatActionPerformed

  private void cerrarBTActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_cerrarBTActionPerformed
    iniciar.setEnabled(true);
  }//GEN-LAST:event_cerrarBTActionPerformed
  /**Esta función es para configurar el nombre del servidor Bluetooth*/
  private void confNombActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_confNombActionPerformed
    String env = System.getProperty(nameProperty);
          String nom =  (String)JOptionPane.showInputDialog(this,"Nombre del Servidor:","Configuración del Servidor",
            JOptionPane.PLAIN_MESSAGE, null, null, env);
          if (nom != null) {
      System.setProperty(nameProperty, nom);
      confNomb.setEnabled(false);
    }

  }//GEN-LAST:event_confNombActionPerformed

  private void iniciarActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_iniciarActionPerformed
    initServidor();
  }//GEN-LAST:event_iniciarActionPerformed

  private void listaDispositivosValueChanged(javax.swing.event.ListSelectionEvent evt) {//GEN-
FIRST:event_listaDispositivosValueChanged
    if(evt.getValueIsAdjusting() == false){
      if(listaDispositivos.getSelectedIndex() == -1){
        infoTexto.setText("Seleccione un dispositivo.");
      }else{
        infoTexto.setText("Dispositivo: "+listaDispositivos.getSelectedValue().toString());
      }
    }
  }//GEN-LAST:event_listaDispositivosValueChanged

  private void salirActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_salirActionPerformed
    System.exit(0);
  }//GEN-LAST:event_salirActionPerformed

  private void okBtnActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_okBtnActionPerformed
    acercaDlg.hide();
  }//GEN-LAST:event_okBtnActionPerformed

  private void acercadeActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_acercadeActionPerformed
    // Se Muestra la ventana de Acerca de
    java.awt.Dimension screenSize = java.awt.Toolkit.getDefaultToolkit().getScreenSize();
    acercaDlg.resize(300,210);
    acercaDlg.setLocation((screenSize.width-300)/2,(screenSize.height-210)/2);
```

```
    acercaDlg.show();

}//GEN-LAST:event_acercadeActionPerformed

/** Método para salir de la aplicacion
 * @param evt Este evento solo ocurre cuando se modifica listaDispositivos
 */
private void exitForm(java.awt.event.WindowEvent evt) {//GEN-FIRST:event_exitForm
    reunionserver.cierraArchivo();
    System.exit(0);
}//GEN-LAST:event_exitForm

/** Método principal del servidor
 * @param args the command line arguments
 */
public static void main(String args[]) {
    new serverBT1().show();
    System.setProperty(nameProperty, "ServerBT");
}

/** Este método es el encargado de inicializar el servidor,
 * tanto la conexión a la base de datos como la inicialización de
 * Bluetooth.
 */
public void initServidor() {
    try {
        Class.forName("org.gjt.mm.mysql.Driver");
    }catch(ClassNotFoundException e){
        status.setText( status.getText() + "\nNo se puede encontrar el Driver JDBC: "+
            e.getMessage());
    }
    /**se realiza la conexion a la DB*/
    try{
        con = DriverManager.getConnection(url,user,pwd);
        status.setText( status.getText() + "\nConexión a la Base de Datos: OK.");

        /** hasta aqui se realizo la conexión a la base de datos y se comienza a
        * inicializar Bluetooth. */
            try{
                //se inicia el servidor de chat
                chatserver = new chatServer( noPuerto, status,con );
                reunionserver = new reunionServer(noReunion,status,con);
                btserver = new btServer(status, chatserver,reunionserver,con);
                btserver.start();
                chatserver.setBTServer(btserver);

                reunionserver.setBTServer(btserver);

            }catch( Exception ce ){
                status.setText( status.getText() + "\n" +ce.getMessage() );
            }
            iniciar.setEnabled(false);


    }catch(SQLException e){
        status.setText( status.getText() + "\nError-> Al crear la conexión: " + e.getMessage());
    }

}
```

```
  private String getServerString() {
        return "btl2cap://localhost:" + SERVICE_CLASS_ID.toString() + ";ReceiveMTU=" + L2CAPConnection.DEFAULT_MTU +
";TransmitMTU=" + L2CAPConnection.DEFAULT_MTU + ";name=Echo";
  }
  /**
   * Muestra la ventana Splash
   */
  public void createSplashScreen() {
        splashLabel = new JLabel(createImageIcon("Splash.png", "Splash.accessible_description"));
        splashScreen = new JWindow();
        splashScreen.getContentPane().add(splashLabel);
        splashScreen.pack();
        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
        splashScreen.setLocation(screenSize.width/2 - splashScreen.getSize().width/2,
                                 screenSize.height/2 - splashScreen.getSize().height/2);
  }

  public void showSplashScreen() {
        splashScreen.show();
  }

  /**
   * Cierra la ventana Splash
   */
  public void hideSplash() {
  splashScreen.setVisible(false);
  splashScreen = null;
  splashLabel = null;
  }
  public ImageIcon createImageIcon(String filename, String description) {
        String path = filename;
        return new ImageIcon(getClass().getResource(path));
  }

   // Variables declaration - do not modify//GEN-BEGIN:variables
  private javax.swing.JMenuItem salir;
  private javax.swing.JTextArea status;
  private javax.swing.JMenu configMenu;
  private javax.swing.JMenuItem iniciar;
  private javax.swing.JDialog acercaDlg;
  private javax.swing.JMenuItem ayuda;
  private javax.swing.JList listaDispositivos;
  private javax.swing.JMenu menuAyuda;
  private javax.swing.JTextPane infoTexto;
  private javax.swing.JMenuItem acercade;
  private javax.swing.JMenu menuServidor;
  private javax.swing.JMenuItem noPuertoChat;
  private javax.swing.JScrollPane infoScroll;
  private javax.swing.JLabel jLabel5;
  private javax.swing.JLabel jLabel4;
  private javax.swing.JScrollPane jScrollPane2;
  private javax.swing.JLabel jLabel3;
  private javax.swing.JScrollPane jScrollPane1;
  private javax.swing.JLabel jLabel2;
  private javax.swing.JLabel jLabel1;
  private javax.swing.JSeparator jSeparator1;
  private javax.swing.JMenuItem confNomb;
  private javax.swing.JMenuBar menu;
```

```
    private javax.swing.JPanel infoPanel;
    private javax.swing.JMenuItem noPuertoReunion;
    private javax.swing.JButton okBtn;
    // End of variables declaration//GEN-END:variables

    /** Conexión a la base de datos*/
    private java.sql.Connection con;
    /** URL de la base de datos*/
    private String url = "jdbc:mysql://localhost:3306/colaboraDB";
    /** Nombre de usuario para la base de datos*/
    private String user = "colaboraAdmin";
    /** Password para el usuario de la base de datos*/
    private String pwd = "cdb1977";
    /** Statement para la consulta en la DB*/
    private java.sql.Statement st = null;
    /** El UUID del servicio*/
    final static UUID SERVICE_CLASS_ID = new UUID(0x7C3434CC);
    /** Servidor Bluetooth*/
    //private BluetoothServer BTserver;
    private btServer btserver;
    /** Nombre del Servidor Bluetooth*/
    private final static String nameProperty = "impronto.localdevice.friendlyname";
    //no del puerto de chat
    private int noPuerto = 9090;
    private int noReunion = 9092;
    private JWindow splashScreen = null;
    private JLabel splashLabel = null;


}
```

Clase para la conexión por Bluetooth btServer.

```
package ServerBluetooth;

import java.io.*;
import javax.Bluetooth.*;
import java.util.*;
import javax.swing.JTextArea;
import com.rococosoft.io.*;
import com.rococosoft.impronto.configuration.*;
import com.rococosoft.impronto.util.BluetoothConstants;
import java.sql.*;
/**
*
* @author  Victor
*/
public class btServer  extends Thread{
    public static final UUID uuid = new UUID( 0x7C3434C );
    private LocalDevice device;
    private OutputStream salida;
    private chatServer chatPC;
    private reunionServer reunionPC;
    private Hashtable outputStreams = new Hashtable();
    private java.sql.Connection con;
    private JTextArea output = new JTextArea(12, 48);
    /** Creates a new instance of btServer */
    public btServer(JTextArea output, chatServer chatpc,reunionServer reunionPC, java.sql.Connection con) {
        this.output = output;
```

```
  this.reunionPC = reunionPC;
  try{
    device = LocalDevice.getLocalDevice();
    chatPC = chatpc;
    this.con = con;
  }catch(BluetoothStateException bte){
    output.setText( output.getText() + bte.toString() );
  }
}
public void run(){
 listen();
}
public void listen(){
  output.setText( output.getText() + "\nBluetooth OK->Iniciando..." );
  try{
    final StreamConnectionNotifier notifier = ( StreamConnectionNotifier )Connector.open( "btspp://localhost:" + uuid.toString() );
    final LocalDevice dev = device;
    StreamConnection c = null;
    for(;;){
      try{
        c = notifier.acceptAndOpen();//se mantiene aceptando las las peticiones
        ServiceRecord r = dev.getRecord( notifier );
        r.setDeviceServiceClasses( BluetoothConstants.SERVICE_AUDIO);
        dev.updateRecord( r );
        //hasta aqui se tienen los parametros
        salida = c.openOutputStream();
        //se salva esta stream para usarlo despues.
        outputStreams.put( c, salida);
        //se crea el nuevo hilo de ejecucion para esta conexion
        new btServerThread(this, chatPC, reunionPC,c, output,c.openInputStream(),salida,con);
        //readFrom(c.openInputStream() );
      }catch(IOException _){
            //se ignora
      }finally{
        if( c != null ){
          try{
            c.close();
          }catch( IOException _){
                //se ignora
          }
        }
      }
    }
  }catch(Exception e){

  }
}

Enumeration getOutputStreams(){
  return outputStreams.elements();
}
public void sendToAll(String message){
  synchronized( outputStreams ){
    //para cada cliente
    for(Enumeration e = getOutputStreams(); e.hasMoreElements();){
      OutputStream out =  (OutputStream) e.nextElement();
      try{
        byte[] b = message.getBytes();
        out.write(b);
```

```
        }catch(IOException ioe){
          output.setText(output.getText()+"\n"+ioe.toString());
        }
      }
    }
  }


  public void sendMessage( String message ){
    byte[] b= message.getBytes();
    System.out.println(message);
    try{
      salida.write(b);
      System.out.println("!!!");
    }catch(IOException _){
      try{
        salida.close();
      }catch(IOException __){
      }
    }
  }
  void removeConnection( StreamConnection conn ){
    // se sincroniza
    synchronized( outputStreams ){
      output.setText( output.getText() + "Eliminando la conexion desde: " + conn );
      //se remueve del hashtable
      outputStreams.remove( conn );
      //nos aseguramos que se cierre la conexion
      try{
        conn.close();
      }catch( IOException e ){
        output.setText( output.getText() + "Error cerrando: " + conn );
        output.setText( output.getText() + e.getMessage() );
      }
    }
  }
}
```

Clase btServerThread.

```
package ServerBluetooth;

import java.io.*;
import javax.Bluetooth.*;
import java.util.*;
import javax.mail.*;
import javax.swing.JTextArea;
import com.rococosoft.io.*;
import com.rococosoft.impronto.configuration.*;
import com.rococosoft.impronto.util.BluetoothConstants;
import javax.mail.internet.*;
import javax.activation.*;
import javax.mail.internet.MimeMultipart.*;
import java.sql.*;
import java.util.Properties;

/**
*
* @author Victor
```

```
*/
public class btServerThread extends Thread {

  private btServer btserver;
  private StreamConnection conn;
  private java.sql.Connection con;
  private JTextArea output;
  private chatServer chatPC;
  private reunionServer reunionPC;
  private InputStream input;
  private String login;
  private String pwd;
  private String grupo;
  private OutputStream salida;
  //se crea un arreglo de objetos del tipo message
  Message messages[];

  Folder folder;

  Store store;
  Session session2;
  Properties props;

  /** Creates a new instance of btServerThread */
  public btServerThread(btServer btserver, chatServer chatPC,reunionServer reunionPC,StreamConnection conn, JTextArea
output,InputStream input,OutputStream salida, java.sql.Connection con) {
    this.btserver = btserver;
    this.conn = conn;
    this.output = output;
    this.chatPC =chatPC;
    this.input = input;
    this.salida = salida;
    this.con=con;
    this.reunionPC =reunionPC;
    props = new Properties();
    start();
  }

  public void run(){
    try{
      //obtengo la salida de bt
      InputStream in = input;
      System.out.println("1");
      while(true){
        readFrom(in);
        System.out.println("2");
      }
    }finally{
      //La conexion se cierra por alguna razon
      // dejamos que el servidor se encarge de ello
      System.out.println("3");
      btserver.removeConnection( conn );
    }
  }
  private void readFrom( InputStream input ){
    byte[] b = new byte[1];
    String comando ="";
    String letra;
    String funcion;
```

```
for(;;){
  try{
    int n = input.read();
    if( n == -1)
      break;
    b[0] = ( byte )n;
    letra = new String(b);
    if( !letra.equals( "±") ){

      comando = comando + new String( b );
    }else{
      //hasta aqui se tiene ya el comando capturado.
      output.setText( output.getText() + "\n"+ comando );
      //se comienza a procesar
      int primerEspacio = comando.indexOf(" ");
      int dosPuntos;
      if(primerEspacio > 0){
        funcion = comando.substring(0,primerEspacio);
        output.setText( output.getText() + "\n"+ funcion );
      }else
        funcion ="";
      //cuando se inicia la sesion
      if(funcion.equals("initsesion") ){
        int esp = comando.indexOf(" ");
        String todo = comando.substring(esp+1);
        int arroba = todo.indexOf("@");
        login = todo.substring(0,arroba);
        //se lee el pwd
        todo = todo.substring(arroba+1);
        arroba = todo.indexOf("@");
        pwd = todo.substring(0,arroba);
        //se lee el grupo
        todo = todo.substring(arroba+1);
        grupo = todo;
        try{
          Statement st = con.createStatement();
          ResultSet rs = st.executeQuery("select login,pwd,descripcion from persgpo,grupos where grupos.nogpo=persgpo.nogpo "
                +" and persgpo.login = \""+login+"\" and persgpo.pwd= \""+pwd+"\" and grupos.descripcion=\""+grupo+"\";" );
          rs.next();
          if(login.equals(rs.getString("login")) && pwd.equals(rs.getString("pwd"))){
            //aqui se envia la confirmacion del login y password
            System.out.println("Usuario: "+rs.getString("login")+" Password: "+rs.getString("pwd"));
          }
        }catch(SQLException e){
          e.printStackTrace();
        }
        System.out.println("inisesion");
        System.out.println("login: "+login);
        System.out.println("pwd: "+pwd);
        System.out.println("grupo: "+grupo);
      }
      //cuando se solicita la lista de correos
      if(funcion.equals("email")){
        //aqui se manda la lista de los mensajes al movil
        try{
          //se obtiene la sesion
          session2 = Session.getDefaultInstance( props, null );

          //se obtienen el store
```

```
      store = session2.getStore( "pop3" );
      store.connect( "colabora", login, pwd );

      // se obtiene el folder
      folder = store.getFolder( "INBOX" );
      folder.open( Folder.READ_ONLY );

      // se obtiene el directorio de los mensajes
      messages = folder.getMessages();
      final String[] valores = new String[messages.length];
      // se añaden en la lista de mensajes
      for(int i = 0; i<messages.length; i++){
        valores[i] = "" + (i+1) + ": "+  messages[i].getFrom()[0] + "~> "+messages[i].getSubject();
        sendMessage("email: "+valores[i]+"±");
      }
      //se cierra la conexion
    }catch( Exception e){
      e.printStackTrace();
    }
    System.out.println("email");
  }
  //cuando se solicita leer un correo
  if(funcion.equals("leermail")){
    String nomail = comando.substring(primerEspacio+1);
    try{
      int num = Integer.parseInt(nomail);
      String de = messages[num].getFrom()[0].toString();
      String asunto = messages[num].getSubject();
      String contenido = messages[num].getContent().toString();
      String mes = "leermail: "+de+"¶"+asunto+"¶"+contenido+"±";
      sendMessage(mes);
      System.out.println(mes);
    }catch(Exception enum){
      System.out.println("Error: "+enum.toString());
    }

    System.out.println(comando);
    System.out.println("leermail");
  }
  //cuando se solicita envia un email
  if(funcion.equals("enviaremail")){
    int esp = comando.indexOf(" ");
    String todoemail = comando.substring(esp+1);
    //hasta aqui ya tengo todo el email
    //se toma el destinatario
    int pycoma = todoemail.indexOf(";");
    String destino = todoemail.substring(0,pycoma);
    //se obtiene el cc
    todoemail = todoemail.substring(pycoma+1);
    pycoma = todoemail.indexOf(";");
    String cc = todoemail.substring(0,pycoma);
    //se obtiene el asunto
    todoemail = todoemail.substring(pycoma+1);
    pycoma = todoemail.indexOf(";");
    String asunto = todoemail.substring(0,pycoma);
    //se obtiene el contenido del email
    todoemail = todoemail.substring(pycoma+1);
    pycoma = todoemail.indexOf(";");
    String contenido = todoemail.substring(0);
```

```
            System.out.println(todoemail);
            System.out.println("enviaremail");
            System.out.println("destino "+destino);
            System.out.println("cc "+cc);
            System.out.println("asunto "+asunto);
            System.out.println("contenido "+contenido);
            //se envia el email.
            sendMail(destino,cc,asunto,contenido);
          }
          //cuando se solicita enviar un mensaje
          if(funcion.equals("mensaje")){
            dosPuntos = comando.indexOf(":");
            String mensaje = comando.substring( dosPuntos+1, comando.length());
            String mensajemovil = "mensaje: "+mensaje+"±";
            //aqui se envia a los usuario del cliente de PC
            chatPC.sendToAll(mensaje);
            //aqui se envia a los usuario del cliente Movil
            btserver.sendToAll(mensajemovil);
            System.out.println("mensaje: "+mensaje);
            System.out.println("mensajemovil: "+mensajemovil);
          }
          //cuando se inicia sesion en reunion
          if(funcion.equals("initreunion"))System.out.println("initreunion");
          //cuando se envia un voto
          if(funcion.equals("votar"))System.out.println("votar");
          //cuando se solicita enviar un mensaje para una reunion
          if(funcion.equals("mensreunion:")){
            dosPuntos = comando.indexOf(":");
            String mensaje = comando.substring( dosPuntos+1, comando.length());
            String mensajemovil = "mensreunion: "+mensaje+"±";
            //aqui se envia a los usuario del cliente de PC
            reunionPC.sendToAll(mensaje);
            //aqui se envia a los usuario del cliente Movil
            btserver.sendToAll(mensajemovil);
            System.out.println("mensaje: "+mensaje);
            System.out.println("mensajemovil: "+mensajemovil);
          }
          comando="";
        }
      }catch(IOException _){
        break;
      }
    }
    try{
      input.close();
      output.setText( output.getText() + "\nCerrado" );
    }catch(IOException _){
      //
    }
    output.setText( output.getText() + "\nCerrado" );
  }
  private void sendMail(String destino, String cc, String asunto, String contenido){
    try{
      Properties props = new Properties();
      //protocolo smtp
      props.setProperty("mail.transport.protocol","smtp");
      //servidor web suele ser nombre_servidor.dominio.com
      props.setProperty("mail.host","colabora");
      //usuario que envia el correo
```

```
        props.setProperty("mail.user",login);
        //password de la cuenta
        props.setProperty("mail.password",pwd);
        Session mailSession = Session.getDefaultInstance(props,null);
        Transport transport = mailSession.getTransport("smtp");
        MimeMessage message = new MimeMessage(mailSession);
        String strTipoMsg = "text/html";
        message.setSubject(asunto);
        message.setSentDate(new java.util.Date());
        //quien lo envia
        InternetAddress addressFrom = new InternetAddress(login+"@colabora");
        message.setFrom(addressFrom);
        message.addRecipient(Message.RecipientType.TO, new InternetAddress(destino));
        if( !cc.equals( "" ) )
          message.addRecipient( Message.RecipientType.CC, new InternetAddress( cc ) );

        message.setContent(contenido,strTipoMsg);
        //conexion
        transport.connect("colabora",login,pwd);
        //envio
        transport.send(message);
        transport.close();
      }catch(Exception e){
        output.setText(output.getText()+"\nError al enviar el email: "+e.toString());
      }
    }
    public void sendMessage( String message ){
      byte[] b= message.getBytes();
      System.out.println(message);
      try{
        salida.write(b);
      }catch(IOException _){
        try{
          salida.close();
        }catch(IOException __){
        }
      }
    }
}
```

Clase BluetoothServer.

```
package ServerBluetooth;

import java.io.*;
import javax.swing.*;
import javax.Bluetooth.*;
import com.rococosoft.io.*;
import com.rococosoft.impronto.discovery.*;
import com.rococosoft.impronto.util.UUIDGenerator;


/** Este es el servidor de Bluetooth para la conexión
 * con los dispositivos
 * @author Victor M. Canseco Soto
 * @version 1.0
 * @Copyright Universidad Tecnológica de la Mixteca
 * @Date Junio/2003
 */
public class BluetoothServer extends Thread {
```

```
  /** Constructor predeterminado para BluetoothServer
   * @param texto Parametro para la salida de los mensajes.
   */
  public BluetoothServer(JTextArea texto){
    output = texto;
    output.setText(output.getText());
  }

  /** Función que se ejecuta en el Thread
   * @param url Para la dirección en que correra el servidor.
   * @throws IOException Si el servidor no arranca se lanza la excepción.
   */
  public void start(String url)
                        throws IOException
          {
                        this.url = url;
                        start();
          }
  /** Esta función es la que se ejecuta cuando se inicia el Thread, aqui se
   * inicializa la conexión por Bluetooth y se c
   */
  public void run(){
                        try {
                                L2CAPConnectionNotifier n = (L2CAPConnectionNotifier)Connector.open(url);
                                ServiceRecord record = LocalDevice.getLocalDevice().getRecord(n);
                                UUID uuid = UUIDGenerator.generateUUID();
                                record.setAttributeValue(ImprontoServiceRecord.SERVICE_ID,
                    new DataElement(DataElement.UUID, uuid));
             output.setText(output.getText()+"\nServidor Bluetooth: OK");
                                output.setText(output.getText()+"\nCorriendo en la dirección: ["+uuid+"].");
                                for (;;){
                                        L2CAPConnection conn = n.acceptAndOpen();
                                        ServerConnection server =  new L2CAPServerConnection(output, conn);
                                        server.start();
                                }
                        } catch(IOException e){
           output.setText(output.getText()+"\nError-> Connector.open:"+e);
                        }catch(com.rococosoft.util.rmi.RuntimeRemoteException rmie){
           output.setText(output.getText()+"\nError-> Excepción RMI: "+rmie);
         }
            }

 /** JTextArea para mostrar los mensajes del servidor
  */
            private JTextArea output = new JTextArea(12, 48);
     /** Url para la direccion del servidor
      */
            private String url;
}

abstract class ServerConnection
            extends Thread
{
            ServerConnection(JTextArea output)
            {
                        this.output = output;
            }
```

```
	public void run() {
		try {
			for (int i=0; i < 100; i++) {
				int c = receiveAndSend();
				if (c == 0)
					break;
				count += c;
				output.append(new Integer(count).toString()+"/");
			}
			close();
		} catch(IOException _) {
			output.setText(output.getText()+"\nError->ServerConnection.run:"+_);
		}
		output.append("\nDone (transferred "+count+" bytes)\n");
	}

	int count = 0;
	JTextArea output;

	abstract int receiveAndSend() throws IOException;
	abstract void close() throws IOException;
}

class L2CAPServerConnection
	extends ServerConnection
{
	L2CAPServerConnection(JTextArea output, L2CAPConnection conn)
	{
		super(output);
		this.conn = conn;
	}

	int receiveAndSend()
		throws IOException
	{
		int sz = conn.receive(buf);
		byte[] pkt = new byte[Math.min(conn.getTransmitMTU(), sz)];
		System.arraycopy(buf, 0, pkt, 0, pkt.length);
		conn.send(pkt);
		return sz + pkt.length;
	}

	void close()
		throws IOException
	{
		conn.close();
	}

	L2CAPConnection conn;
	byte[] buf = new byte[10 * L2CAPConnection.DEFAULT_MTU];
}
```

Clase reunionServer.

```
package ServerBluetooth;

import java.io.*;
import java.net.*;
import java.util.*;
```

```
import javax.swing.*;
import java.sql.*;
/**
 *
 * @author  Victor
 */
public class reunionServer extends Thread{

  //Server socket es utilizado para aceptar nuevas conexiones
  private btServer bts;
  private ServerSocket ss;
  //se crea un objeto JTextArea
  private JTextArea status = new JTextArea();
  private DataOutputStream archivo;
  //se crea un mapa de los sockets en DataOutputStream
  private Hashtable outputStreams = new Hashtable();
  private int port;
  private String motivoVotacion;
  private int novotos = 0;
  private int si = 0;
  private int no = 0;
   private Connection con;
  /** Creates a new instance of Server */
  public reunionServer( int port, JTextArea status,Connection con ) throws IOException{
    this.port = port;
    this.status = status;
    this.con=con;
    java.util.Date fecha = new java.util.Date();
    String nomArch = "c:\\reuniones\\reunion_"+fecha.getDay()+"_"+fecha.getMonth()+"_"+(fecha.getYear()+1900)+".rtf";
    archivo = new DataOutputStream(new FileOutputStream(nomArch));
    escribeArchivo("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n");
    escribeArchivo("Fecha: "+fecha.toString()+"\n");
    escribeArchivo("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n");
    start();
  }
  public void run(){
    try{
      listen( port );
    }catch( Exception e){
      status.setText( status.getText() + e.getMessage() );
    }

  }

  private void listen( int port ) throws IOException{
    //se crea el SocketServer
    ss = new ServerSocket( port );
    //Se muestra que ya esta conectado
    status.setText( status.getText() + "\nEscuchando en: " + ss );

    //se mantiene aceptando las conexiones
    while(true){
      //se aceptan las conexiones
      Socket s = ss.accept();

      //se muestra la conexion que se establecio
      status.setText( status.getText() + "Conexion desde: " + s );

      //Crea el DataOutput para escribir datos hacia el otro lado
```

```
    DataOutputStream dout = new DataOutputStream( s.getOutputStream() );
    //se salva este stream para posterior uso
    outputStreams.put( s, dout );
    //se crea el nuevo hilo para esta conexcion
    new rServerThread( this,bts, s, status,con);
  }
}

//se obtiene una enumeracion de todos los outputStreams, uno por cada
//uno de los clientes conectados a nosotros.
Enumeration getOutputStreams(){
  return outputStreams.elements();
}

public void sendToAll( String message ){
  //sincronizamos los hilos de ejecucion
  synchronized( outputStreams ){
    //para cada cliente.....
    for( Enumeration e = getOutputStreams(); e.hasMoreElements(); ){
      //... se obtiene el output stream
      DataOutputStream dout = ( DataOutputStream )e.nextElement();
      //se envia el mensaje
      try{
        dout.writeUTF( message );
        escribeArchivo(message+"\n");
      }catch( IOException ie ){
        status.setText( status.getText() +"Error: " + ie.getMessage() );
      }
    }
  }
}
//se remueve un socket y su correspondiente outputstream de la lista
void removeConnection( Socket s ){
  // se sincroniza
  synchronized( outputStreams ){
    status.setText( status.getText() + "Eliminando la conexión desde: " + s );
    //se remueve del hashtable
    outputStreams.remove( s );
    //nos aseguramos que se cierre la conexion
    try{
      s.close();
    }catch( IOException e ){
      status.setText( status.getText() + "Error cerrando: " + s );
      status.setText( status.getText() + e.getMessage() );
    }
  }
}
public void setBTServer(btServer bts){
  this.bts = bts;
}
public void sendBT(String mensaje){
  bts.sendMessage( mensaje );
}
public void escribeArchivo(String cadena){
  try{
    archivo.writeUTF(cadena);
    archivo.flush();
  }catch(IOException e){
    e.printStackTrace();
```

```
    }
  }
  public void cierraArchivo(){
    try{
      archivo.flush();
      archivo.close();
    }catch(IOException e){
      e.printStackTrace();
    }
  }
  public void votaciones(String motivo, String voto){
    if(novotos == 0){
      motivoVotacion = motivo;
      sendToAll("~~~Votacion~~~ Motivo: "+motivo);
      sendBT("mensreunion: ~~~Votacion~~~ Motivo: "+motivo+"±");
    }
    if(motivoVotacion.equals(motivo)){
      if(voto.equals("SI"))
        si++;
      if(voto.equals("NO"))
        no++;
      novotos++;
      sendToAll("Votos: SI:"+si+" NO:"+no);
      sendBT("mensreunion: Votos: SI:"+si+" NO:"+no+"±");
    }

  }
}
```

Clase rServerThread.

```
package ServerBluetooth;

import java.io.*;
import java.net.*;
import javax.swing.*;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;
import java.sql.SQLException;
/**
 *
 * @author  Victor
 */
public class rServerThread extends Thread {
  //el servidor que emplearemos
  private reunionServer server;
  private btServer btserver;
  //el socket conectado a nuestro cliente
  private Socket socket;
  private Connection con;
  JTextArea status = new JTextArea();
  private DataOutputStream dout;

  /** Creates a new instance of ServerThread */
  public rServerThread( reunionServer server,btServer bts, Socket socket, JTextArea status, Connection con) {
    //se salvan los parametros
    this.server = server;
    this.socket = socket;
    this.status = status;
```

```
    this.btserver = bts;
    this.con =con;
    try{
      this.dout=new DataOutputStream(socket.getOutputStream());
    }catch(Exception e){
      System.out.println("Error al crear el socket");
    }
    //se inicia el hilo
    start();
  }

  public void run() {
    String funcion;
    try{
      //se Crea un DataInputStream para la comunicacion
      //el cliente usa un DataOutputStream para escribirnos
      DataInputStream din = new DataInputStream( socket.getInputStream() );
      // se hace siempre
      while( true ){
        //...leer el siguiente mensaje
        String message = din.readUTF();
        int primerEspacio = message.indexOf(" ");
        int dosPuntos;
       if(primerEspacio >0){
        funcion = message.substring(0,primerEspacio);
        status.setText( status.getText() + "\n"+ funcion );
        }else
          funcion="";
        //cuando se inicia la sesion
        if(funcion.equals("initreunion") ){
            int esp = message.indexOf(" ");
            String todo = message.substring(esp+1);
            int arroba = todo.indexOf("@");
            String login = todo.substring(0,arroba);
            //se lee el pwd
            todo = todo.substring(arroba+1);
            String pwd = todo;
            //hasta aqui ya tenemos los datos para la sesion
            //faltan de modificar
            try{
              Statement st = con.createStatement();
              ResultSet rs = st.executeQuery("select login,pwd,fecha,hora from persreunion,reunion where
reunion.noreunion=persreunion.noreunion "
                    +" and persreunion.login = \""+login+"\" and persreunion.pwd= \""+pwd+"\";" );
              rs.next();
              if(login.equals(rs.getString("login")) && pwd.equals(rs.getString("pwd"))){
                try{
                  java.util.Date fecha = rs.getDate("fecha");
                  java.sql.Time hora = rs.getTime("hora");
                  fecha.setTime(hora.getTime());

                  java.util.Date fechaHoy = new java.util.Date(); //fecha de hoy

                  //se tienen la hora y fecha de la reunion
                  //se comparan la actual y la de la DB
                  if(fecha.compareTo(fechaHoy) == 0)
                    System.out.println("Misma fecha");
                  else
                    System.out.println("Otra fecha");
```

```
                    dout.writeUTF("initreunion si");
                    System.out.println("Usuario: "+rs.getString("login")+" Password: "+rs.getString("pwd"));
                  }catch(Exception doute){
                    status.setText( status.getText() +doute.toString());
                  }
                }else{
                  dout.writeUTF("initreunion no");
                  System.out.println("Rechazado");
                }
              }catch(SQLException e){
                dout.writeUTF("initreunion no");
                  System.out.println("Rechazado");
                e.printStackTrace();
              }
              System.out.println("initreunion");
              System.out.println("login: "+login);
              System.out.println("pwd: "+pwd);
              //aqui se registra al usuario en el archivo
              server.escribeArchivo("Login: "+login+" Hora de entrada: "+new java.util.Date().toString()+"\n");
          }else{
            if(funcion.equals("votar")){
              int esp = message.indexOf(" ");
              String todo = message.substring(esp+1);
              int arroba = todo.indexOf("@");
              String motivo = todo.substring(0,arroba);
              String sino = todo.substring(arroba+1);
              System.out.println(motivo);
              System.out.println(sino);
              server.votaciones(motivo,sino);
            }else{
              //se envia a todos
              server.sendToAll( message );
              btserver.sendToAll("mensreunion: "+message+"±");
            }
          }
        }
      }catch( EOFException ie ){
        //no se envia nada
      }catch( IOException ie ){
        status.setText( status.getText() + ie.getMessage() );
      }finally{
        //La conexion se cierra por alguna razon
        // dejamos que el servidor se encarge de ello
        server.removeConnection( socket );
      }
    }
  }
```

Clase chatServer.

```
/*
 * chatServer.java
 *
 * Created on 10 de junio de 2002, 11:45 AM
 */

package ServerBluetooth;

import java.io.*;
```

```
import java.net.*;
import java.util.*;
import javax.swing.*;
import java.sql.*;


/**
*
* @author Victor
*/
public class chatServer extends Thread{

  //Server socket es utilizado para aceptar nuevas conexiones
  private btServer bts;
  private Connection con;
  private ServerSocket ss;
  //se crea un objeto JTextArea
  private JTextArea status = new JTextArea();

  //se crea un mapa de los sockets en DataOutputStream
  private Hashtable outputStreams = new Hashtable();
  private int port;

  /** Creates a new instance of Server */
  public chatServer( int port, JTextArea status,Connection con ) throws IOException{
    this.con = con;
    this.port = port;
    this.status = status;
    start();
  }
  public void run(){
    try{
      listen( port );
    }catch( Exception e){
      status.setText( status.getText() + e.getMessage() );
    }

  }

  private void listen( int port ) throws IOException{
    //se crea el SocketServer
    ss = new ServerSocket( port );
    //Se muestra que ya esta conectado
    status.setText( status.getText() + "\nEscuchando en: " + ss );

    //se mantiene aceptando las conexiones
    while(true){
      //se aceptan las conexiones
      Socket s = ss.accept();

      //se muestra la conexion que se establecio
      status.setText( status.getText() + "Conexion desde: " + s );

      //Crea el DataOutput para escribir datos hacia el otro lado
      DataOutputStream dout = new DataOutputStream( s.getOutputStream() );
      //se salva este stream para posterior uso
      outputStreams.put( s, dout );
      //se crea el nuevo hilo para esta conexcion
      new ServerThread( this,bts, s, status,this.con);
```

```
    }
  }

  //se obtiene una enumeracion de todos los outputStreams, uno por cada
  //uno de los clientes conectados a nosotros.
  Enumeration getOutputStreams(){
    return outputStreams.elements();
  }

  public void sendToAll( String message ){
    //sincronizamos los hilos de ejecucion
    synchronized( outputStreams ){
      //para cada cliente.....
      for( Enumeration e = getOutputStreams(); e.hasMoreElements(); ){
        //... se obtiene el output stream
        DataOutputStream dout = ( DataOutputStream )e.nextElement();
        //se envia el mensaje
        try{
          dout.writeUTF( message );

        }catch( IOException ie ){
          status.setText( status.getText() + "Error: " + ie.getMessage() );
        }
      }
    }
  }
  //se remueve un socket y su correspondiente outputstream de la lista
  void removeConnection( Socket s ){
    // se sincroniza
    synchronized( outputStreams ){
      status.setText( status.getText() + "Eliminando la conexcion desde: " + s );
      //se remueve del hashtable
      outputStreams.remove( s );
      //nos aseguramos que se cierre la conexion
      try{
        s.close();
      }catch( IOException e ){
        status.setText( status.getText() + "Error cerrando: " + s );
        status.setText( status.getText() + e.getMessage() );
      }
    }
  }
  public void setBTServer(btServer bts){
    this.bts = bts;
  }
  public void sendBT(String mensaje){
    bts.sendMessage( mensaje );
  }
}
```

Clase ServerThread.

```
/*
 * ServerThread.java
 *
 * Created on 11 de junio de 2002, 01:18 AM
 */

package ServerBluetooth;
```

```
import java.io.*;
import java.net.*;
import javax.swing.*;
import java.sql.*;
/**
*
* @author  Victor
*/
public class ServerThread extends Thread {
  //el servidor que emplearemos
  private chatServer server;
  private Connection con;
  private btServer btserver;
  //el socket conectado a nuestro cliente
  private Socket socket;
  private DataOutputStream dout;
  JTextArea status = new JTextArea();

  /** Creates a new instance of ServerThread */
  public ServerThread( chatServer server,btServer bts, Socket socket, JTextArea status,Connection con) {
    //se salvan los parametros
    this.con = con;
    this.server = server;
    this.socket = socket;
    this.status = status;
    this.btserver = bts;
    try{
      this.dout=new DataOutputStream(socket.getOutputStream());
    }catch(Exception e){
      System.out.println("Error al crear el socket");
    }
    //se inicia el hilo
    start();
  }

  public void run() {
    String funcion;

    try{
      //se Crea un DataInputStream para la comunicacion
      //el cliente usa un DataOutputStream para escribirnos
      DataInputStream din = new DataInputStream( socket.getInputStream() );
      // se hace siempre
      while( true ){
        //...leer el siguiente mensaje
        String message = din.readUTF();
        int primerEspacio = message.indexOf(" ");
        int dosPuntos;
        if(primerEspacio > 0 )
          funcion = message.substring(0,primerEspacio);
        else
          funcion = "";
        status.setText( status.getText() + "\n"+ funcion );
        //cuando se inicia la sesion
        if(funcion.equals("initsesion") ){
            int esp = message.indexOf(" ");
            String todo = message.substring(esp+1);
            int arroba = todo.indexOf("@");
```

```
            String login = todo.substring(0,arroba);
            //se lee el pwd
            todo = todo.substring(arroba+1);
            arroba = todo.indexOf("@");
            String pwd = todo.substring(0,arroba);
            //se lee el grupo
            todo = todo.substring(arroba+1);
            String grupo = todo;
            //hasta aqui ya tenemos los datos para la sesion
            try{
              Statement st = con.createStatement();
              ResultSet rs = st.executeQuery("select login,pwd,descripcion from persgpo,grupos where grupos.nogpo=persgpo.nogpo "
                    +" and persgpo.login = \""+login+"\" and persgpo.pwd= \""+pwd+"\" and grupos.descripcion=\""+grupo+"\";" );
              rs.next();
              if(login.equals(rs.getString("login")) && pwd.equals(rs.getString("pwd"))){
                //aqui se envia la confirmacion del login y password
                try{
                  dout.writeUTF("initsesion si");
                  System.out.println("Usuario: "+rs.getString("login")+" Password: "+rs.getString("pwd"));
                }catch(Exception doute){
                  status.setText( status.getText() +doute.toString());
                }
              }else{
                dout.writeUTF("initsesion no");
                System.out.println("Rechazado");
              }
            }catch(SQLException e){
              dout.writeUTF("initsesion no");
                System.out.println("Rechazado");
              e.printStackTrace();
            }
            System.out.println("inisesion");
            System.out.println("login: "+login);
            System.out.println("pwd: "+pwd);
            System.out.println("grupo: "+grupo);
          }else{
            //se avisa
            //System.out.println( "Enviando: " + message );
            //se envia a todos
             server.sendToAll( message );
             btserver.sendToAll("mensaje: "+message+"±");
          }
        }
      }catch( EOFException ie ){
        //no se envia nada
      }catch( IOException ie ){
        status.setText( status.getText() + ie.getMessage() );
      }finally{
        //La conexion se cierra por alguna razon
        // dejamos que el servidor se encargue de ello
        server.removeConnection( socket );
      }
    }
  }
```

Lista de figuras

Lista de Tablas