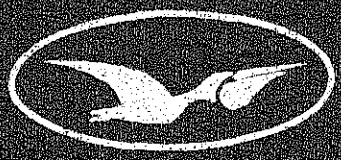
Centro de Investigación Científica y de Educación Superior de Ensenada



DISEÑO Y DESARROLLO DE UNA PLATAFORMA PARA APLICACIONES MOVILES COLABORATIVAS

TESIS
MAESTRIA EN CIENCIAS

JOSE MANUEL ALBA ALVAREZ

ENSENADA, B. C., AGOSTO DEL 2000.

TESIS DEFENDIDA POR

JOSE MANUEL ALBA ALVAREZ

Y APROBADA POR EL SIGUIENTE COMITE

Dr. Jesús Favela Vara
Director del Comité

M.C. José Luis Briseño Cervantes
Miembro del Comité

Dr. David Hilario Covarrubias Rosales
Miembro del Comité

Dr. Pedro Gilberto López Mariscal
Miembro del Comité

Dr. Robert Pozos Bonilla
Miembro del Comité

Dr. José Luis Briseño Cervantes
Jefe del Departamento de Ciencias de la Computación

Dr. Federico Graef Ziehl
Director de Estudios de Posgrado

29 de agosto de 2000

CENTRO DE INVESTIGACION CIENTIFICA Y DE EDUCACION SUPERIOR DE ENSENADA

DIVISION DE FISICA APLICADA

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACION

DISEÑO Y DESARROLLO DE UNA PLATAFORMA PARA APLICACIONES MOVILES COLABORATIVAS

TESIS

que para cubrir parcialmente los requisitos necesarios para obtener el grado de

MAESTRO EN CIENCIAS presenta:

JOSE MANUEL ALBA ALVAREZ

Ensenada, Baja California, México, Agosto de 2000

RESUMEN de la Tesis de **JOSE MANUEL ALBA ALVAREZ**, presentada como requisito parcial, para la obtención del grado de **MAESTRO EN CIENCIAS** en **CIENCIAS DE LA COMPUTACION**. Ensenada, Baja California, México. Agosto de 2000.

DISEÑO Y DESARROLLO DE UNA PLATAFORMA PARA APLICACIONES MOVILES COLABORATIVAS.

Resumen aprobado por:

Dr. Jesús Favela Vara
Director de Tesis

El mercado de las computadoras de bolsillo (*handheld computers*) es uno de los que actualmente crece más rápidamente. Conforme estos dispositivos se vuelven más populares, crece también la tendencia de volverlos en aparatos de comunicación móvil, particularmente, con el fin de accesar Internet. Esto los está convirtiendo en un instrumento natural para colaborar, ya que, además de almacenar mucha de la información personal del usuario (listas de contactos, horarios de reuniones, etc.), estos dispositivos están siempre disponibles, lo que no sucede con las computadoras de escritorio.

Sin embargo, todavía hay mucho que hacer para que estos aparatos se vuelvan la plataforma preferida en la colaboración por medio de computadoras. Primeramente, se debe determinar el rol que estos dispositivos juegan dentro de arquitecturas colaborativas que no solo incluyen computadoras de escritorio y aplicaciones servidor, sino también un número creciente de computadoras siempre presentes en los espacios en donde vivimos y trabajamos. Además, existe la necesidad de construir herramientas para desarrollar aplicaciones colaborativas que utilizen este tipo de dispositivos móviles.

Este trabajo de tesis aborda los aspectos antes mencionados: tomando como base un análisis de algunas aplicaciones y arquitecturas Handheld CSCW existentes, así como un conjunto de escenarios de utilización de colaboración con dispositivos handheld, se propone una plataforma de desarrollo para la construcción de aplicaciones colaborativas móviles. La plataforma propuesta es llamada COMAL y está compuesta por una arquitectura y un conjunto de bibliotecas que pueden ser usadas en el desarrollo de sistemas colaborativos móviles. Después de introducir COMAL, se explica la estructura de su arquitectura, así como sus bibliotecas y la forma en que éstas pueden ser utilizadas y extendidas. La arquitectura COMAL es una guía para las personas que utilizen la plataforma, ilustrando los componentes clave de una aplicación colaborativa móvil, y la relación entre tales componentes. También se ilustra el uso de COMAL con el desarrollo de Group Messenger, una aplicación colaborativa que integra computadoras de escritorio, dispositivos Palm y una aplicación servidor. Finalmente, se muestran las pruebas realizadas a las bibliotecas COMAL y a la aplicación desarrollada, los resultados de tales pruebas y las conclusiones de este trabajo.

Palabras clave: Plataforma de desarrollo, Handheld CSCW, PDA.

ABSTRACT of the Thesis of **JOSE MANUEL ALBA ALVAREZ**, presented as partial requirement to obtain the **MASTER IN SCIENCE** degree in **COMPUTER SCIENCE**. Ensenada, Baja California, México. August 2000.

DESIGN AND DEVELOPMENT OF A PLATFORM FOR MOBILE COLLABORATIVE APPLICATIONS.

ABSTRACT

Handheld computers represent one of the fastests growing segments of the computer industry today. As these devices become more popular, there is a strong tendency to convert them into mobile communication devices, and in particular, to enable them for Internet access. As this happens, handheld computers become a natural medium for collaboration. Not only do they store much of the user's personal information (contacts lists, meeting schedules, to-do lists, etc.), but they are always at hand, in sharp contrast with desktop computers.

However, much work remains to be done for handheld computers to become the platform of choice for computer-mediated collaboration. First, we must determine the role that these devices should play in a collaboration architecture that includes not only desktop computers and information servers, but also a growing number of ubiquitous computers placed around our working and living environments. Second, tools need to be built to facilitate the development of handheld collaborative applications.

The current thesis work addresses these two issues, based on an analysis of current Handheld CSCW applications, architectures, solutions and a set of use scenarios for typical handheld collaboration, it proposes a framework for the development of handheld collaborative applications. The proposed framework is called COMAL and is made of an architecture and a set of libraries that can be used for the construction of Handheld CSCW systems. After introducing COMAL, the structure of its proposed architecture is explained, as well as its libraries and the way they can be used and extended. The COMAL architecture is provided as a guide for people using the framework, ilustrating key components of Handheld CSCW applications and the relation between such components. Use of COMAL is also illustrated with the development of Group Messenger, a sample application for user collaboration that integrates desktop computers, Palm devices and an application server.

Finally, the tests performed to the libraries and the sample application are presented, along with the results obtained and the conlusions of this work.

Keywords: Development platform, Handheld CSCW, PDA.

Dedicatorias

A mi hermana Leslie

A mis padres, Anita y José Manuel y a mis abuelos Lolita y Cresencio

A todos los que mueren tratando de hacer de este planeta un mejor lugar para vivir

Agradecimientos

Un agradecimiento muy especial a mi director de tesis, Dr. Jesús Favela Vara, por su apoyo durante mi estancia en el posgrado y por su valiosa guía en la realización de este trabajo, gracias mil.

A los miembros de mi comité de tesis, M.C. José Luis Briseño, Dr. David Covarrubias, Dr. Gilberto López y Dr. Robert Pozos, por sus valiosos comentarios y sobre todo, por el interés mostrado en el trabajo que juntos realizamos.

Al Dr. Robert Pozos y a Biol. Marisela Avila por toda su ayuda durante mi estancia en SDSU, fue una muy buena experiencia, engrandecida por el hecho de haberlos conocido.

A mis compañeros de generación, con quienes me hubiera gustado haber convivido mucho más, y a mis compañeros de laboratorio, los *hardcore coolers*: Haydeé, Ricardo, Octavio, Juan, Rafilla, Adrián, César Arturo, Roberto y Mireles.

A mis compañeros del Neurokinetics Lab en SDSU, Iván y César, gracias por compartir conmigo largas noches de hacer tareas y tomar soda, por aguantar mi música estridente, y sobre todo, por convertirse en dos de mis mejores amigos.

Al personal del CICESE, a los maestros que contribuyeron a mi formación, a las secretarias del posgrado en Ciencias de la Computación, al personal del DSE y a los técnicos que siempre me apoyaron cuando fue necesario.

Al Consejo Nacional de Ciencia y Tecnología.

Contenido.

Lista de Figuras.

Capítulo I.

Introducción.

I.1 Introducción.

La tecnología de la información ha revolucionado la forma en que realizamos nuestras actividades diarias. Lectoras de códigos de barras, cajeros automáticos, computadoras, videocámaras, redes de datos y teléfonos celulares son solo algunos de los dispositivos electrónicos que podemos encontrar en cualquier mercado, casa u oficina. Estos aparatos son utilizados con el fin de reducir costos, mejorar procesos, seguridad, pero sobre todo, para asistirnos en nuestras tareas cotidianas. Uno de los dispositivos que mejor cumple con esta función es el Asistente Digital Personal.

Los Asistentes Digitales Personales, o PDAs (*Personal Digital Assistants*) son pequeños dispositivos que se crearon inicialmente como un remplazo electrónico a las agendas de bolsillo tradicionales. Estos dispositivos cuentan casi siempre con aplicaciones para manejar un calendario de citas o eventos, una libreta de direcciones, una lista de asuntos pendientes, una lista de gastos, un sencillo editor de texto y una calculadora. A través de estas aplicaciones el usuario se mantiene en contacto permanente con su información más importante independientemente del lugar en el que se encuentre, ya que, de la misma manera que las agendas de bolsillo tradicionales, los usuarios pueden llevar y utilizar estos dispositivos en la palma de su mano o en sus

portafolios. La gran mayoría de estos dispositivos personales cuentan con mecanismos para respaldar información en computadoras de escritorio y algunos permiten también intercambiar información con usuarios que utilicen dispositivos similares, favoreciendo escenarios de colaboración entre usuarios de PDAs.

La integración de estos dispositivos a redes de computadoras y con tecnologías de comunicación inalámbrica también permite ahora a sus usuarios accesar información actualizada y además, mantenerse en contacto con amigos y compañeros de trabajo. Esto ha provocado además la creación y el crecimiento acelerado del mercado del acceso a Internet a través de los PDAs.

I.2 Antecedentes.

En los años recientes, utilizar dispositivos de cómputo personales se ha convertido en un fenómeno muy común. De acuerdo a una estudio realizado por la compañía IDC [IDC, 1999], a finales de 1999 el 65% de los dispositivos electrónicos de bolsillo eran PDAs, en el análisis se estima también que al terminar 1999 había 5.4 millones de dispositivos alrededor del mundo y que para el año 2003 la cifra se elevaría a 18.9 millones. Ejemplos de estos dispositivos son las computadoras Palm de Palm Inc., las Jornada de Hewlett-Packard, las Aero de Compaq y las Cassiopeia de Cassio. Los dispositivos Palm, también conocidos como la Plataforma de Cómputo Palm, son actualmente los más populares, el 68% de los PDAs en E.U. y el 47% de los PDAs en todo el mundo son dispositivos Palm.

Debido a la naturaleza personal de los PDAs, la mayor parte de las aplicaciones que se ejecutan en ellos fueron diseñadas para ser utilizadas por un solo usuario. Sin embargo, quienes utilizan estos dispositivos, muchas veces forman parte de equipos de

trabajo y surge la necesidad de contar con aplicaciones que soporten actividades de trabajo colaborativo en los PDAs. Además, la habilidad que tienen algunos dispositivos de este tipo para interconectarse e intercambiar información puede sin duda ser aprovechada para la construcción de aplicaciones colaborativas.

Una motivación adicional para la utilización de PDAs en trabajo colaborativo es que la mayor parte de la información que se almacena en estos dispositivos, tal como fechas de reuniones y eventos, listas de asuntos pendientes y anotaciones o pequeños segmentos de texto, juegan un papel central en muchas aplicaciones utilizadas en situaciones de colaboración entre personas.

I.3 Planteamiento del problema.

Si bien actualmente existen pocas aplicaciones colaborativas que puedan utilizarse en PDAs, la necesidad de contar con herramientas para desarrollar este tipo de sistemas es cada vez mayor. La carencia de estas herramientas hace más difícil la construcción de aplicaciones colaborativas para PDAs, ya que, con cada nueva aplicación, el desarrollador debe comenzar su tarea casi con los mismos recursos con los que comenzóó a desarrollar su primer sistema.

Algunos de los aspectos del desarrollo de las aplicaciones colaborativas en PDAs están todavía en período de investigación y la evolución de estos aparatos ofrece cada día nuevos e importantes escenarios de utilización en el área del cómputo colaborativo que deben ser tomados en cuenta. Un ejemplo de ésto es el estudio de la problemática que se presenta al intercambiar información entre PDAs y pantallas públicas que se hace en [Greenberg *et al.*, 1999].

I.4 Objetivos.

Objetivo General. El objetivo de este trabajo de tesis fue diseñar y construir una herramienta para facilitar el desarrollo de aplicaciones colaborativas en computadoras móviles personales.

Además del objetivo general, también se plantearon los siguientes objetivos específicos:

- Caracterizar a las aplicaciones móviles, analizando las plataformas, aplicaciones y arquitecturas existentes y proponiendo varios casos de uso. El análisis y la elaboración de los escenarios de utilización permitió proponer además, una arquitectura de las aplicaciones móviles con soporte para colaboración, la cual puede ser utilizada como referencia para el desarrollo de aplicaciones que utilicen la herramienta construida.

- Se propuso también como objetivo que, una vez terminada la herramienta de desarrollo, se utilizara para construir una aplicación colaborativa móvil específica, con el propósito de probar y evaluar la herramienta desarrollada.

- Otro objetivo de esta tesis fue encontrar el lugar que ocupan las aplicaciones móviles colaborativas dentro del espacio de aplicaciones del Trabajo Cooperativo Asistido por Computadora. Esto debido a que las aplicaciones de este tipo no han sido bien definidas, a diferencia de las aplicaciones colaborativas tradicionales, aún cuando los PDAs están siendo ya utilizados en situaciones de colaboración.

I.5 Organización de la tesis.

En el siguiente capítulo se describe el área de la computación en la que se sitúa este trabajo y se dá una descripción de la situación actual de las aplicaciones colaborativas móviles y de algunos de los dispositivos PDAs en los que se utilizan. En el capítulo III se describen algunas arquitecturas que se han propuesto para el desarrollo de aplicaciones colaborativas móviles, y en ese mismo capítulo, se describen algunos escenarios típicos de utilización de dichas aplicaciones. Estos escenarios fueron utilizados como base para la arquitectura de aplicaciones colaborativas móviles que se propone en este trabajo y que se presenta al final del capítulo III.

En el capítulo IV se describe COMAL, la biblioteca para desarrollar aplicaciones colaborativas móviles que se construyó como parte de esta tesis, además, se muestra la forma en que la biblioteca puede ser utilizada y extendida. En el siguiente capítulo, el V, se presenta Group Messenger, la aplicación desarrollada con el objetivo de probar COMAL. Además de la descripción del sistema, el capítulo V contiene un análisis de cada uno de los componentes de la aplicación y una guía de utilización de la misma. En el capítulo VI se describen las pruebas realizadas a la biblioteca COMAL y a la aplicación Group Messenger, así como los resultados obtenidos. Finalmente, en el capítulo VII, se presentan las conclusiones de este trabajo de tesis.

Capítulo II.

Aplicaciones colaborativas móviles.

II.1 Trabajo Cooperativo Asistido por Computadora.

El avance que han tenido las áreas de telecomunicaciones y computación en los últimos años ha facilitado enormemente el intercambio de información utilizando redes de computadoras. Para confirmar lo anterior basta con mencionar los millones de usuarios alrededor del mundo que se comunican diariamente utilizando Internet, la *madre de todas las redes* que sigue creciendo, año tras año, a un ritmo impresionante. Este fenómeno no es nuevo, desde principios de la década de los 90's, el núcleo de las nuevas tecnologías computacionales ha sido la computadora conectada a una red de datos. Esto ha provocado que el término *computadora* sea menos frecuentemente utilizado para describir un dispositivo que sirve para leer datos, hacer cálculos con ellos y presentar resultados al usuario. Este escenario computacional no describe la esencia del uso que se les da actualmente a la mayoría de las computadoras y la forma en que serán utilizadas cada vez más en el futuro. Muchas de las tareas que las computadoras de la actualidad realizan son las relacionadas con la comunicación y con la coordinación de actividades tales como dar seguimiento a órdenes de compra, inventarios y cuentas de clientes, enviar y recibir correos electrónicos y realizar teleconferencias. Mientras

más aumenten las interconexiones entre estas computadoras, encontraremos muchas más formas de coordinar las actividades que con ellas se realizan. Por lo anterior, es muy posible que las redes de computadoras sean recordadas no como una tecnología utilizada principalmente para hacer cálculos, sino como una tecnología de coordinación y comunicación de información [Malone y Rockart, 1991].

De entre los cambios provocados por la evolución en el uso de la tecnología de la información destaca el impacto que la utilización de computadoras ha tenido en la forma de trabajar de las personas. Son pocos los espacios de trabajo que actualmente no cuentan con una computadora conectada a alguna red de datos sobre la cual se distribuye la información de la organización para la cual se labora. Además, son cada vez más frecuentes los equipos de trabajo distribuidos, en los cuales, varios o todos los miembros del grupo viven en ciudades diferentes y se comunican y coordinan sus actividades por medio de las computadoras. Las reuniones de trabajo, tradicionalmente realizadas con la ayuda de lápices y hojas de papel también han cambiado, frecuentemente se llevan a cabo ahora en recintos equipados con equipos de cómputo, proyectores multimedia y cámaras de video. El objetivo de la utilización de estos recursos es el de enriquecer la calidad de la información que se presenta en las reuniones, mejorar la distribución y el acceso a información, proveer a los miembros del grupo medios alternativos para el intercambio de ideas y permitir la interacción con personas en lugares distantes.

El área multidisciplinaria que explora el potencial de las computadoras como soporte al trabajo en grupo es conocida como *Trabajo Cooperativo Asistido por Computadora* (Computer Supported Cooperative Work); ésta surgió a mitad de los 80's aunque sus orígenes se remontan a la década de los 60's. Algunos ejemplos de las actividades que incluye son la comunicación, solución de problemas, coautoría y soporte a reuniones cara a cara. El Trabajo Cooperativo Asistido por Computadora es la dis-

ciplina científica que regula el diseño y desarrollo apropiado de sistemas de software multiusuario que proporcionen soporte a tales actividades. Esta familia de productos de software es conocida como *groupware* [Soriano y Favela, 1997].

De acuerdo a [Ellis *et al.*, 1991], el objetivo del groupware es asistir a grupos de personas en la comunicación, la colaboración y la coordinación de sus actividades. Específicamente, definen al groupware como: *sistemas basados en computadoras que asisten a grupos de personas involucradas en una tarea u objetivo común, y que proveen una interfase común a un ambiente compartido* [Ellis *et al.*, 1991].

II.2 Trabajo colaborativo en Asistentes Digitales Personales.

Una de las áreas de la industria de la computación que también ha crecido notablemente en fechas recientes es la del *Handheld Computing*, compuesto por las computadoras que pueden ser operadas sobre la palma de la mano (*Handheld PCs*), pueden llevarse de un lado a otro o pueden "vestirse" la mayor parte del tiempo sin estorbar demasiado a quien las opera. Esta categoría no solo incluye a las Handheld PCs, sino también a los Asistentes Digitales Personales, los dispositivos para el envío de mensajes, tarjetas inteligentes y las *Wearable Computers* (computadoras integradas a la vestimenta del usuario). Mientras por un lado se ha reducido el tamaño de las computadoras de escritorio hasta poder ser operadas en la palma de la mano, los teléfonos portátiles y otros dispositivos para envío y recepción de mensajes se han transformado en verdaderas computadoras; algunos de los teléfonos móviles más modernos cuentan con organizadores personales y mecanismos para intercambiar información vía mensajes y enlaces de comunicación infrarroja [Gellersen, 1998].

Antes de continuar, es preciso establecer la diferencia entre el Handheld Computing y el Cómputo Móvil (*Mobile Computing*). De acuerdo a [Gellersen, 1998], mientras en el Cómputo Móvil el objetivo es la utilización de un recurso, independientemente del lugar en el que se encuentra el usuario, el Handheld Computing está orientado hacia nuevas formas de utilizar los recursos, en vez de trasladar su utilización. Es decir, el propósito del Handheld Computing es el de tomar ventaja de la movilidad, en vez de esconderla, propiciando la interacción con el entorno y la implementación de sistemas de información dependientes de la posición del usuario. El tema central de este trabajo es el Handheld Computing como herramienta de colaboración. En este mismo contexto, se hace referencia a las aplicaciones móviles, no en el sentido estricto de la definición del Cómputo Móvil, sino para describir las aplicaciones que son ejecutadas en los PDAs, debido a la libertad que tienen la personas que utilizan dichos sistemas en situaciones de movimiento, fuera de su espacio de trabajo.

Gran parte del éxito del mercado de los PDAs se debe a que algunos de los modelos más recientes permiten su integración a redes locales inalámbricas e Internet. Es decir, además de poder accesar su información personal almacenada en el PDA, el usuario puede utilizar el dispositivo para consultar otros datos que se encuentran disponibles en la red. Esto es particularmente útil cuando el usuario desea contar con información actualizada mientras se encuentra en movimiento.

Muchos de los usuarios de estos dispositivos forman parte de equipos de trabajo y tienen la necesidad de contar con aplicaciones que soporten actividades de trabajo colaborativo en los PDAs, y además la de integrar a los PDAs a sistemas *groupware* existentes. Debido a que los PDAs acompañan a sus usuarios donde sea que se encuentren, la mayor parte del tiempo, son candidatos ideales a utilizar en escenarios de interacción casual, como soporte a reuniones de trabajo y otros escenarios en los cuales

una persona necesita realizar una actividad de colaboración y se encuentra fuera de su espacio de trabajo.

II.3 Panorama actual.

Uno de los primeros proyectos en donde se exploró la idea de utilizar pequeños dispositivos de cómputo portátiles para la comunicación y la interacción entre personas se desarrolló durante 1993 en los laboratorios Xerox PARC [Dittrich y Thomas, 1998]. Esta área de investigación es hoy conocida como *Handheld CSCW*. Con la invención de PDAs más pequeños con mayor capacidad de procesamiento y mejores dispositivos de desplegado y de comunicación inalámbrica, el Handheld CSCW ha tenido un gran crecimiento en los últimos dos años. En 1998 se llevó a cabo la primera edición del *Workshop on Handheld CSCW* [Gellersen, 1998] como parte de la conferencia *The 1998 ACM Conference on CSCW*, este *Workshop* se convirtió después en el *International Symposium on Handheld and Ubiquitous Computing* [HUC, 1999].

A pesar de que el interés en el Handheld CSCW ha crecido últimamente, la mayoría de las aplicaciones que se han desarrollado para PDAs están orientadas hacia un solo usuario o son aplicaciones sencillas para el intercambio de mensajes. Con el aumento en la integración del Handheld Computing y las redes de computadoras se espera que la tendencia de las aplicaciones para estos dispositivos cambie, de un solo usuario a multi-usuario, de la misma forma que cambió en el mundo de las computadoras de escritorio [Gellersen, 1998].

II.3.1 Algunas aplicaciones existentes.

A continuación se describen algunas de las aplicaciones más representativas del Handheld CSCW. En el siguiente capítulo se describen de manera más detallada las aplicaciones AvantGo y NETMAN, así como la arquitectura que cada una utiliza.

AvantGo.

Este software provee mecanismos que permiten el acceso a información disponible en formato HTML desde PDAs que utilizan PalmOS y WindowsCE [AvantGo, 2000]. Los documentos HTML se obtienen del servidor HTTP donde se encuentran almacenados, pero antes de ser trasladados a el PDA son adecuados a la pantalla de éstos dispositivos, las imágenes se hacen más pequeñas y son removidos elementos que no pueden ser desplegados en un PDA como los Applets de Java. El software AvantGo es utilizado la mayoría de las veces para convertir aplicaciones WWW tradicionales en aplicaciones WWW móviles que puedan ser accesadas desde PDAs.

NETMAN.

Este sistema se desarrolló para permitir la colaboración entre personas que utilizan computadoras que pueden ser vestidas (Wearable Computers) y que trabajan como técnicos de mantenimiento de una red de computadoras que abarca todo un campus. El objetivo principal del sistema es aumentar la calidad de la colaboración entre técnicos que salen a hacer reparaciones a cualquier lugar dentro del campus y técnicos expertos que se encuentran en sus oficinas utilizando una computadora de escritorio [Kortuem *et al.*, 1999].

Pebbles.

El objetivo del proyecto Pebbles es el de crear aplicaciones para conectar varios PDAs a una computadora personal (PC). En [Myers *et al.*, 1998] se describen dos aplicaciones desarrolladas dentro de este proyecto: Remote Commander y PebblesDraw. Remote Commander permite utilizar los PDAs de los asistentes a una reunión como los dispositivos de entrada (teclado y ratón) de una PC. Después de conectar los PDAs a la PC, por medio de cables tipo serial y comenzar la ejecución de Remote Commander, los usuarios pueden colaborar manipulando las aplicaciones de la computadora de escritorio a través de sus Asistentes Digitales Personales. PebblesDraw es un programa de dibujo multiusuario que se ejecuta en una PC y puede ser controlado por varios usuarios utilizando PDAs conectados a la computadora, de la misma forma que cuando se utiliza Remote Commander.

SharedNotes.

Esta es una aplicación creada con el objetivo de conocer los aspectos relacionados con la publicación de notas privadas (generadas en un PDA) en pantallas públicas, la edición de tales notas y la posterior sincronización entre la pantalla pública y el PDA. Utilizando SharedNotes, el usuario puede crear y manipular notas personales y públicas en tres tipos de dispositivos: un PDA (Palm), una estación de trabajo y una pantalla pública. SharedNotes se implementó utilizando las herramientas GroupKit [Roseman y Greenberg, 1996] y permite también la elaboración de reuniones distribuidas, en las que los participantes pueden conectarse a la sesión utilizando sus estaciones de trabajo [Greenberg *et al.*, 1999].

En las aplicaciones anteriores vemos como son utilizados los PDAs para actividades de colaboración, enriqueciendo las interacciones entre sus usuarios y facilitándoles

el acceso a la información, cuando se encuentran fuera de sus espacios de trabajo. Si bien AvantGo es ya un producto altamente utilizado, el resto de las aplicaciones mencionadas son prototipos de investigación, creados para conocer de manera más precisa la problemática asociada al Handheld CSCW. Esto es una muestra de lo nueva que ésta área de la computación es y lo mucho que todavía queda por investigarse en esta dirección.

II.4 Ventajas y desventajas.

Aún cuando utilizar PDAs en situaciones de colaboración parece tener enormes ventajas y ofrecer una nueva gama de posibilidades de colaboración, existen también ciertas desventajas que deben tomarse en cuenta, desventajas inherentes a los recursos limitados con los que cuenta un Asistente Digital Personal.

Una de las características deseables de un PDA es que sea pequeño, ligero, que consuma poca energía y que sea fácilmente portable. Es por ésto que generalmente tienen poco espacio de almacenamiento, poca memoria RAM, pantallas reducidas y procesadores menos poderosos que los que se utilizan en computadoras de escritorio tradicionales. A pesar de que la miniaturización de los componentes antes mencionados ha reducido la diferencia entre los recursos utilizados en un PDA y los utilizados en una computadora convencional, la diferencia es todavía significativa.

Otra desventaja importante en la utilización de PDAs es la referente a sus dispositivos de entrada de datos. La mayoría de las personas que trabajan con computadoras están acostumbradas a controlarlas utilizando un ratón y un teclado. Estos dispositivos de entrada de datos no son muy comunes en los PDAs ya que tienden a aumentar el tamaño de los aparatos. Ratón y teclado son frecuentemente remplazados por 4 ó 5

botones y una pluma con la que el usuario puede manipular botones, ventanas y listas y escribir información en el PDA. Los trazos del usuario son leídos por el PDA utilizando un software para el reconocimiento de escritura, este software reconoce un conjunto predefinido de símbolos y no la escritura natural del usuario.

Si bien pareciera que las desventajas de la utilización de PDAs son demasiadas como para pensar en sustituir a las computadoras convencionales por estos pequeños dispositivos, hay que tomar en cuenta que tanto las Handheld PCs, como los Asistentes Digitales Personales, proporcionan una cosa que las computadoras de escritorio no pueden ofrecer: movilidad, y con eso, acceso a nuestra información y otros recursos independientemente del lugar donde nos encontremos, la mayor parte del tiempo. Se mencionan las desventajas de estos dispositivos móviles, por que son aspectos que se deben tener siempre presentes al trabajar con ellos, ya que, mientras mejor se conocen, más se pueden aprovechar.

II.5 Dispositivos y plataformas.

El mercado de los PDAs tiene una gran variedad de opciones, con precios que varían desde menos de 100 dólares por un organizador electrónico sencillo, hasta los más de 2000 dólares que se llega a pagar por una Wearable Computer. En todo el amplio espectro de opciones que existen, dos plataformas concentran una gran mayoría de los dispositivos, los PDAs que utilizan el PalmOS y las Handheld PCs y PDAs que ejecutan el sistema operativo WindowsCE (dispositivos ahora conocidos como PocketPCs). En la Figura 1 se pueden observar algunos dispositivos que utilizan los sistemas operativos antes mencionados.

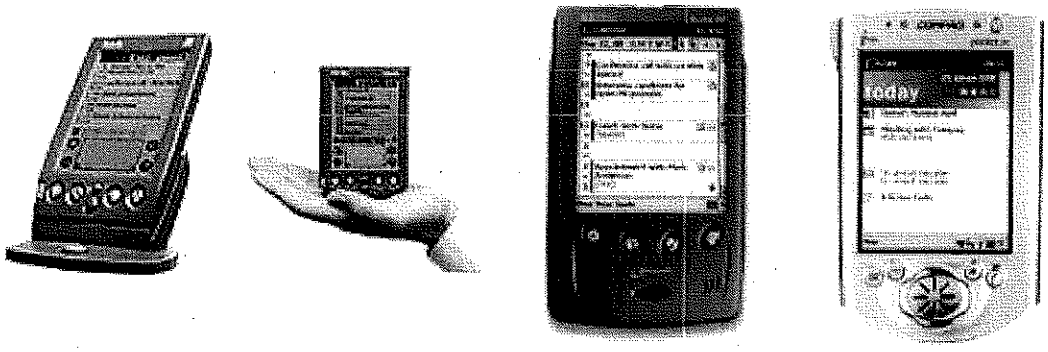


Figura 1: Algunos PDAs, de izquierda a derecha se muestran el Palm IIIx, que usa el PalmOS, otro PDA Palm en la mano de una persona, una PocketPC Jornada de Hewlett-Packard y una Aero de Compaq, éstos dos últimos con WindowsCE.

PalmOS y WindowsCE deben su popularidad al gran número de aplicaciones existentes para ambas plataformas. En el caso del PalmOS el desarrollo de aplicaciones se fomentó con una política abierta, permitiendo el acceso libre a herramientas de desarrollo y al código de fuente de la mayoría de las aplicaciones, incluso del mismo sistema operativo PalmOS, construídas por la compañía 3Com (creadores del dispositivo Palm). El desarrollo de aplicaciones para WindowsCE fue un poco más controlado, pero también tuvo éxito y recibió un poco de ayuda extra de la popularidad de los sistemas operativos Windows de la compañía Microsoft.

Entre los otros PDAs existentes están los REX, del tamaño de una tarjeta de crédito, pero con métodos de entrada de datos muy limitados y los Wizard, de la compañía Sharp, que cuentan con un mini-teclado. Estos y otros dispositivos no han tenido mucho éxito debido a algunas de sus limitaciones y a una política de desarrollo de aplicaciones más restringida que en los casos mencionados en el párrafo anterior. Las Wearable Computers tampoco han sido aceptadas de manera masiva, en parte por su precio, por que no son todavía socialmente bien aceptadas y por que algunas veces requieren demasiadas atenciones y cuidados de los usuarios.

II.5.1 Plataforma de Cómputo Palm y PalmOS.

PalmOS es el sistema operativo creado por la compañía 3Com para los dispositivos Palm, antes conocidos como PalmPilot (actualmente Palm Inc. es la compañía que fabrica los dispositivos Palm y da mantenimiento al PalmOS, esta empresa fue creada por 3Com). El Dispositivo y el sistema operativo forman lo que se conoce como la Plataforma de Cómputo Palm (*Palm Computing Platform*). Los 5 componentes principales de esta plataforma son los siguientes [Bachmann, 1999]:

- Un diseño estándar de hardware.
- Una interfase para extender el dispositivo añadiendo componentes de hardware.
- El sistema operativo PalmOS.
- Sincronización de datos desde y hasta una computadora de escritorio.
- Un kit y herramientas de desarrollo para construir aplicaciones.

Además de fabricar dispositivos Palm y dar mantenimiento al sistema operativo, Palm Inc. también licencia el PalmOS a otras compañías para que construyan dispositivos similares, entre estas compañías están: IBM, Handspring, Symbol Technologies y Qualcomm. Debido a que comparten el mismo sistema operativo, los dispositivos con PalmOS son totalmente compatibles aunque sean construidos por diferentes compañías.

Este sistema operativo incluye soporte para manejar varios controles de interfaz de usuario y construir formas. Las formas son similares a las ventanas que se utilizan en algunos sistemas operativos de escritorio. Debido a la simplicidad de la interfaz de usuario de los dispositivos Palm, sólo una forma es desplegada a la vez. La interfaz de programación de aplicaciones para manejar formas incluye muchos elementos para construir interfaces de usuario tales como: botones de radio, cajas de selección

(checkboxes), botones, listas, tablas, barras de recorrido, menús, etiquetas y campos de edición de texto [Rhodes y Mckeehan, 1999].

El PalmOS tiene soporte para varios métodos de comunicación. Gran parte del éxito de los dispositivos Palm se debe a la manera en que el aspecto del intercambio de datos es manejado en éstos dispositivos. Los protocolos de comunicación que actualmente forman parte del PalmOS son: un protocolo para la comunicación serial, una implementación del protocolo TCP/IP (con una interfaz de sockets) y un protocolo de comunicación infrarroja que utiliza el estándar de transmisión IrDA [Rhodes y Mckeehan, 1999].

II.5.2 WindowsCE.

De acuerdo a la descripción que la compañía Microsoft hace de WindowsCE [Microsoft, 2000], éste es un sistema operativo que sirve como plataforma para una amplia variedad de dispositivos de comunicaciones, entretenimiento y cómputo móvil. WindowsCE fue diseñado con la idea de permitir el intercambio de información entre estos dispositivos, compartir datos con computadoras que cuentan con sistemas operativos de la familia Windows y tener acceso a Internet. Entre los aparatos en los cuales se consideró utilizar este sistema operativo están las Handheld PCs (computadoras que pueden ser operadas sobre la palma de las manos), teléfonos celulares "inteligentes", reproductores DVD y dispositivos inalámbricos de comunicación.

WindowsCE es un sistema operativo de 32 bits, es multi-tareas, tiene soporte para múltiples hilos de ejecución y ha sido diseñado bajo una arquitectura abierta, lo que permite su implementación en una amplia variedad de dispositivos. Además, es un sistema compacto, que ofrece alto rendimiento en configuraciones de memoria

limitada [Microsoft, 2000].

II.5.3 Otras plataformas.

Con la miniaturización de los componentes electrónicos, es posible hacer computadoras de escritorio cada vez más pequeñas y con esto, crear pequeños y poderosos dispositivos capaces de ejecutar sistemas operativos como alguna versión de Microsoft Windows, UNIX BSD ó Linux. Un ejemplo de esto es la *WearComp* de Steve Mann [Mann, 2000]. Sin embargo, utilizar un sistema gráfico de ventanas como XWindow parece no ser muy buena idea cuando se tiene una pantalla con una resolución de 160x160 pixeles o con una superficie de menos de 25 cm.^{2}.

II.6 Métodos de intercomunicación.

La mayoría de los PDAs en el mercado actual pueden comunicarse con otros dispositivos de alguna manera. La forma de intercomunicación más popular es la que se dá entre un PDA y una computadora personal. Este tipo de comunicación se realiza conectando el PDA y la computadora de escritorio por medio de un cable tipo serial o tipo USB, esta conexión permite al usuario utilizar su PDA realmente como una extensión del espacio de trabajo disponible en su computadora de escritorio. De esta forma, se puede pasar información desde la computadora personal hacia el PDA y viceversa, consultar y editar los datos necesarios en uno de los dos dispositivos y después, actualizar la información almacenada en ambos aparatos mediante un proceso de sincronización.

Uno de los métodos de comunicación que ha crecido recientemente es el inalámbrico. Este fenómeno ha sido impulsado por la aparición de nuevos estándares de comuni-

cación inalámbrica como Bluetooth (`http://www.bluetooth.com`), el WAP (`http://www.wapforum.org`) y el IrDA (`http://www.irda.org`), así como por la inclusión de transmisores y receptores de comunicación inalámbrica en muchos de los PDAs que han aparecido en los últimos dos años. Algunos PDAs pueden también conectarse a modems y utilizar la red telefónica para accesar Internet y otras redes de datos.

II.6.1 Sincronización.

La sincronización de información es el método de transmisión de datos más utilizado para la comunicación entre PDAs y computadoras personales. El sincronizado de información puede clasificarse en 3 categorías:

- *Sincronización escritorio a PDA.* En este caso la información fluye desde la computadora personal del usuario hacia el PDA. De esta manera, el usuario puede accesar una copia de la información que está almacenada en su escritorio, a través de su PDA. Este mecanismo se utiliza también para actualizar información en el PDA que fue modificada en la computadora de escritorio del usuario y para instalar aplicaciones en el PDA.

- *Sincronización PDA a escritorio.* Este tipo de sincronización es frecuentemente realizado con la intención de respaldar la información que contiene el PDA en la computadora personal del usuario, para poder restaurarla en caso de que algún inconveniente suceda. También es muy útil esta sincronización en los casos en los que el usuario ha modificado información en su PDA y desea que el cambio se refleje en su computadora de escritorio.

- *Sincronización entre escritorio y PDA.* Este es el tipo de sincronización en el cual la información de ambos dispositivos, la computadora de escritorio y el

PDA, es actualizada. La aplicación encargada de realizar este procedimiento es responsable de decidir que información debe actualizarse en cada dispositivo. La aplicación de sincronización también se encarga de establecer una política para resolver conflictos de sincronización (e.g. cuando existe información que fue modificada tanto en la computadora de escritorio, como en el PDA). Soluciones típicas a este problema son: dejar que la información del PDA sobreescriba la información en la computadora de escritorio del usuario, que la información de la computadora personal del usuario sobreescriba la del PDA y la tercera solución es dejar que el usuario resuelva el conflicto de forma manual.

II.6.2 Protocolos de intercambio de datos.

La mayoría de los protocolos que se utilizan para la sincronización de información son propietarios. Aunque las compañías que crearon los protocolos no permiten el acceso directo a sus especificaciones, si proporcionan herramientas de desarrollo con las que se pueden construir aplicaciones que utilicen los mecanismos de sincronización que los protocolos proporcionan. Ejemplos de estos protocolos de sincronización son HotSync de Palm Inc., ActiveSync de Microsoft e IntelliSync de Puma Technologies.

Los protocolos de sincronización se utilizan solamente para comunicar una computadora personal y un PDA de la forma que se describe en el punto . Cuando se requiere comunicar dos PDAs o comunicar un PDA con una aplicación que se ejecuta en una computadora que se encuentra en otro lugar dentro de una red de datos se utilizan otros protocolos, también disponibles en una gran cantidad de PDAs. Algunos de los protocolos más populares son el PPP, el SLIP, la familia de protocolos TCP/IP y los protocolos de comunicación inalámbrica Bluetooth, WAP e IrDA.

II.6.3 IrDA.

La *Infrared Data Association* (IrDA) ha producido un conjunto de especificaciones de protocolos que utilizan tecnología infrarroja, conocidos como protocolos IrDA. Estos protocolos de comunicación infrarroja son de los más utilizados actualmente, más de 150 millones de dispositivos alrededor del mundo los utilizan para transmitir y recibir datos. Entre las características de esta familia de protocolos, diseñados como remplazo a los cables en conexiones punto a punto, están: compatibilidad con una amplia variedad de plataformas de hardware y software, no producen interferencia con otros dispositivos electrónicos y altas velocidades de transmisión. En el Apéndice VII.1 se describe la problemática asociada a las comunicaciones infrarrojas y se presenta de manera más detallada la familia de protocolos IrDA.

En este Capítulo se ha descrito el estado actual del campo de las aplicaciones colaborativas móviles. Se presentaron algunas aplicaciones existentes y las características, ventajas, desventajas y diferencias de las plataformas de cómputo más comúnmente utilizadas en el Handheld CSCW. En el Capítulo siguiente se describen algunas arquitecturas de aplicaciones colaborativas móviles existentes y se presenta un conjunto de escenarios de utilización de aplicaciones del mismo tipo. El análisis de tales arquitecturas y casos de uso son la base de la arquitectura y herramienta de desarrollo que se proponen hacia el final del mismo capítulo.

Capítulo III.

Arquitectura de las aplicaciones colaborativas móviles.

III.1 Importancia de las arquitecturas de software.

Las especificaciones y el diseño de los sistemas de software se vuelven más importantes que la selección de algoritmos y estructuras de datos, cuando el tamaño y la complejidad de la estructura general de los sistemas aumenta notablemente. Los aspectos estructurales a tomar en cuenta incluyen la organización del sistema como un conjunto de componentes, estructuras globales de control, protocolos de comunicación, sincronización y acceso de datos, composición y funcionalidad de los elementos del sistema, distribución física de los elementos, rendimiento y la selección de las alternativas de diseño apropiadas [Shaw y Garlan, 1996].

La arquitectura de software consiste en la descripción de los elementos a partir de los cuales se construyen los sistemas de software, la interacción entre estos elementos, los patrones que guían su composición y las limitantes de estos patrones. Un sistema en particular es generalmente definido en términos de una colección de componentes y las interacciones entre ellos, dicho sistema puede a su vez ser descrito como un componente en la descripción de otro sistema más grande [Shaw y Garlan, 1996].

Dadas las características de algunas de las arquitecturas que se han propuesto para las aplicaciones colaborativas móviles y con el objetivo de conocer de manera más profunda los aspectos involucrados en el desarrollo de estos sistemas se propuso una arquitectura que describe a este tipo de aplicaciones. Antes de definir la arquitectura se realizó un análisis de las arquitecturas que a continuación se presentan.

III.2 Algunas arquitecturas existentes.

Como resultado del interés que ha surgido en el Handheld CSCW se han desarrollado algunas plataformas y arquitecturas como base para la construcción de aplicaciones colaborativas móviles, algunos de estas herramientas se encuentran incluso disponibles de manera comercial. A continuación se describen algunos de los trabajos más representativos realizados en el área.

III.2.1 AvantGo.

Este software provee mecanismos que permiten el acceso a información disponible en formato HTML desde PDAs que utilizan PalmOS y WindowsCE [AvantGo, 2000]. Los documentos HTML se distribuyen desde el servidor HTTP donde se encuentran almacenados y posteriormente son modificados para que puedan ser desplegados en las pequeñas pantallas de los PDAs. El software AvantGo es utilizado la mayoría de las veces para convertir aplicaciones WWW tradicionales en aplicaciones WWW móviles que puedan ser accesadas desde PDAs. AvantGo está compuesto por 3 componentes principales: un servidor AvantGo, un cliente AvantGo y un Enlace Móvil (ver Figura 2).

- *Servidor AvantGo.* A través de este componente se provee acceso a la información almacenada en formato HTML. Desde este servidor es posible actualizar el con-

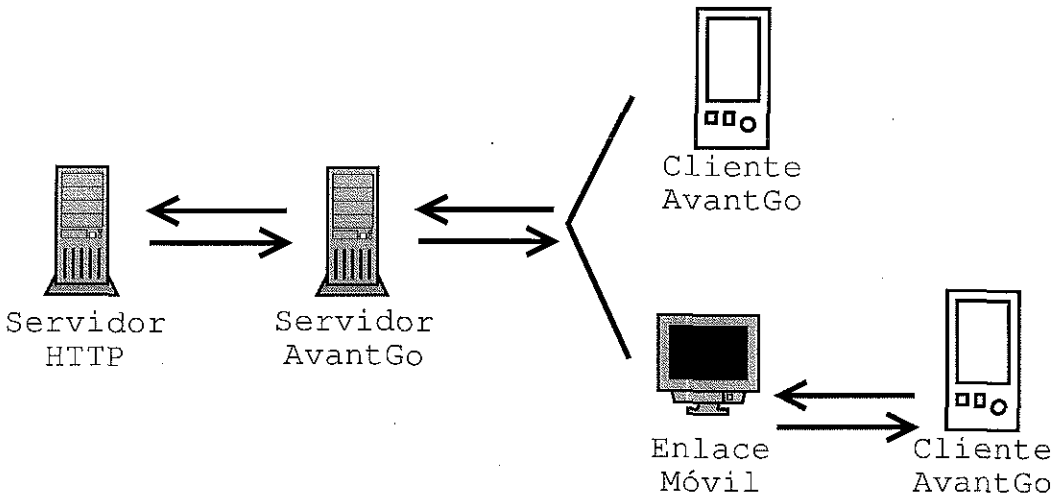


Figura 2: Arquitectura AvantGo para aplicaciones móviles.

tenido y las aplicaciones que serán accesadas desde los clientes móviles sin realizar actualizaciones en los dispositivos cliente. Este componente es el encargado de adecuar los archivos HTML a las pantallas de los PDA, por ejemplo, reduciendo el tamaño de las imágenes y eliminando Applets de Java. La versión 3.1 del Servidor AvantGo está basado en la tecnología utilizada en el servidor WWW Apache y puede utilizarse en las siguientes plataformas: Windows NT, Linux, Solaris y FreeBSD.

- *Cliente AvantGo.* Esta es una aplicación que se ejecuta en computadoras móviles que cuentan con el sistema operativo PalmOS o con WindowsCE, consiste en un eficiente navegador WWW con capacidad para manejar entrada dinámica de datos, como en el caso de formas, generando respuestas de acuerdo a los datos registrados por el usuario y al tipo de dispositivo que se utiliza en un momento dado.

- *Enlace Móvil.* Este componente permite el intercambio de información entre el cliente y el servidor AvantGo utilizando el protocolo MAL (*Mobile Application*

Link). El MAL, desarrollado específicamente para el sistema AvantGo, está compuesto por un estándar de comunicación y un software a través del cual los dispositivos móviles intercambian datos y pequeños programas directamente con aplicaciones servidor centralizadas, utilizando las tecnologías HotSync [Palm, 2000] y ActiveSync [McPherson, 2000] para intercambiar información entre dispositivos móviles y computadoras de escritorio.

La arquitectura utilizada por AvantGo es ampliamente utilizada en la actualidad y el software para desarrollar aplicaciones de este tipo, así como una serie de servicios de acceso a información del WWW para dispositivos móviles, se encuentra disponible en el sitio `http://www.avantgo.com/`.

III.2.2 NETMAN.

Este sistema se desarrolló para permitir la colaboración entre personas que utilizan Wearable Computers y que trabajan como técnicos de mantenimiento de una red de computadoras que abarca todo un campus. El objetivo principal del sistema es aumentar la calidad de la colaboración entre técnicos que salen a hacer reparaciones a cualquier lugar dentro del campus y técnicos expertos que se encuentran en sus oficinas utilizando una computadora de escritorio [Kortuem *et al.*, 1999]. Este sistema se implementó en la Universidad de Oregon en cooperación con el personal del Centro de Cómputo de la universidad, quienes son los encargados de dar mantenimiento a la red de datos del campus. El escenario de utilización de NETMAN se muestra en la Figura 3.

NETMAN es un sistema de groupware distribuido que consiste en los componentes de hardware y de software que a continuación se describen:

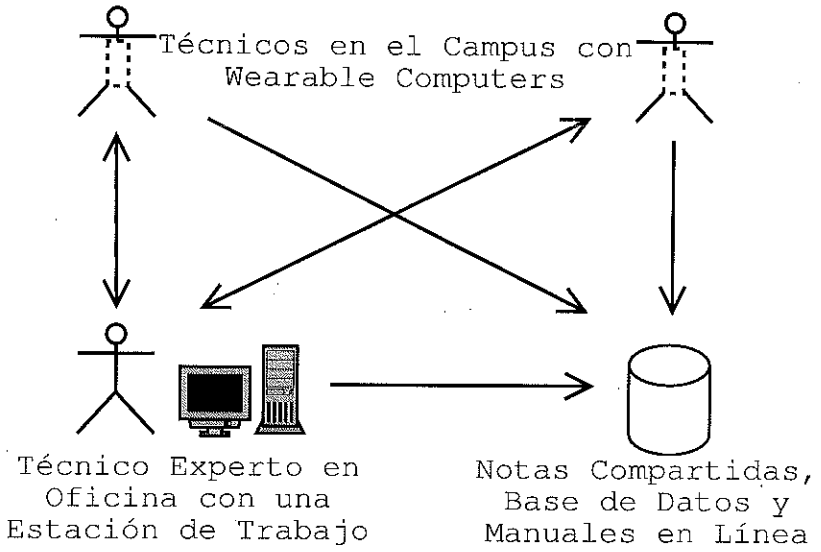


Figura 3: Escenario de utilización del sistema NETMAN.

- *Una Wearable Computer.* Es vestida por el personal técnico durante tareas de reparación y mantenimiento.
- *Varias computadoras de escritorio.* O estaciones de trabajo, utilizadas por técnicos expertos en el Centro de Cómputo.
- *Aplicaciones de software.* Estas aplicaciones se ejecutan tanto en las Wearable Computers como en las estaciones de trabajo.
- *Un servidor de bases de datos centralizado.*

Tanto las Wearable Computers como las estaciones de trabajo tienen acceso a Internet. En el caso de las Wearable Computers, el acceso se realiza a través de una red inalámbrica disponible en todo el campus. El software que se ejecuta en Las Wearables Computers es idéntico al de las estaciones de trabajo de los técnicos expertos y está compuesto por tres módulos: el Manejador de Aplicaciones, Los Módulos de Aplicación y el Manejador de Sesión. La forma en que éstos elementos se encuentran organizados

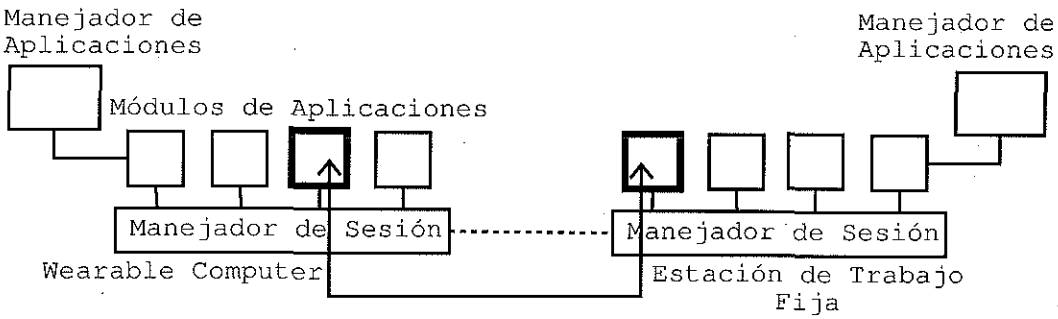


Figura 4: Arquitectura de software del sistema NETMAN.

se ilustra en la arquitectura de software de NETMAN (Figura 4).

El Manejador de Aplicaciones es una interfaz gráfica sencilla para cambiar de una aplicación a otra, consiste en un área donde se despliega la aplicación actualmente activa y un par de botones (*Previous* y *Next*) para cambiar la aplicación actual. Aún cuando existan varias aplicaciones ejecutándose de manera concurrente, el usuario verá solamente una aplicación a través del Manejador de Aplicaciones. El objetivo de éste módulo es el de simplificar la interfaz de las aplicaciones que se ejecutan, evitándose situaciones como la de ocultar una aplicación con una ventana de otro programa.

Los Módulos de Aplicaciones son los programas que se ejecutan en las Wearable Computers y las estaciones de trabajo que se utilizan en NETMAN. Estas aplicaciones son:

- *Camera Viewer.* Despliega el video de lo que el usuario de una Wearable Computer se encuentra viendo y permite que otros usuarios puedan ver lo mismo aunque se encuentren en otro lugar.
- *Web Browser.* Este módulo permite a los técnicos accesar manuales en línea, archivos de ayuda, archivos de configuración, etc.

- *Emailer.* Esta aplicación permite a los técnicos mandar y recibir mensajes de correo electrónico mientras se encuentran haciendo una reparación. El módulo cuenta también con una alarma que emite una señal acústica o visual para notificar cuando un nuevo mensaje ha sido recibido.

El Manejador de Sesión es responsable de establecer las conexiones de audio y video entre dos computadoras y también implementa un mecanismo para compartir la pantalla del espacio de trabajo de los usuarios.

III.2.3 MEX.

MEX (MessageEXchanger) es una arquitectura de software, cuyo objetivo es servir como plataforma para la construcción de aplicaciones para incrementar y mejorar la comunicación y la cooperación de personas que utilizan Wearable Computers por espacios de tiempo prolongado [Lehikoinen *et al.*, 1999]. En la estructura de esta arquitectura también se contempla la interacción entre quien porta la Wearable Computer y otros recursos disponibles en el lugar donde se encuentra.

Un ambiente MEX consiste en un servidor MEX y un conjunto de servicios asociados al servidor (ver Figura 5). El servidor se encarga de llevar un registro de los servicios disponibles y de transferir eventos hacia los servicios. Cada servicio se registra en el ambiente MEX especificando el nombre del servicio y los eventos en los cuales está interesado. Un ambiente MEX local se puede describir como una burbuja personal alrededor del usuario, esta burbuja puede, de ser necesario, absorber servicios próximos que son ofrecidos por dispositivos al alcance del ambiente MEX local.

La mayoría de las veces los servicios se conectan solamente a un ambiente pero en algunos casos es necesario que el servicio se registre cuando menos en dos ambientes

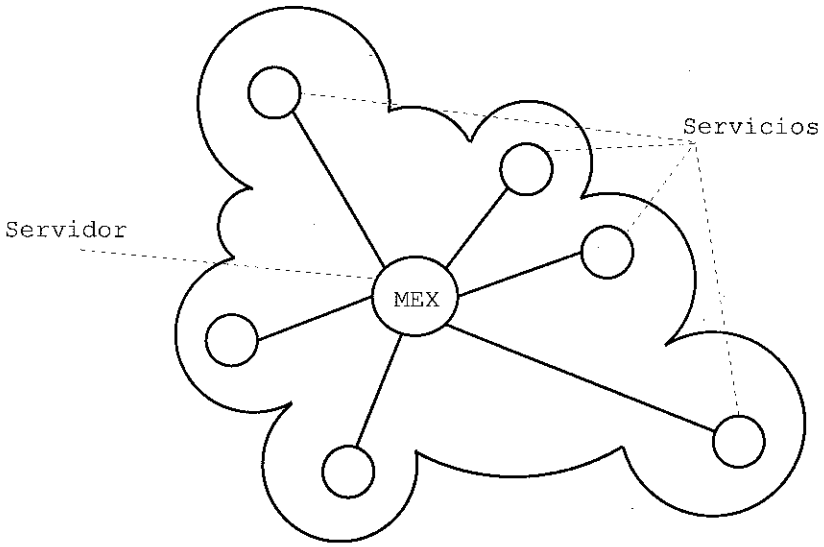


Figura 5: Estructura de un ambiente MEX.

distintos, como en el caso en que se desea que se establezca una colaboración entre dos usuarios, es decir, entre dos ambientes MEX. Este tipo de servicios son conocidos como *proxies* o enlaces ya que proveen un punto de intercambio de información entre dos ambientes MEX.

MEX es una arquitectura en estado experimental con mucha similitud con la tecnología Jini [Sun, 2000], la implementación actual utiliza la familia de protocolos TCP/IP para realizar la conexión entre los servicios y el servidor, pero según los autores, se podría utilizar cualquier otro protocolo punto-a-punto sin realizar muchas modificaciones.

III.2.4 Aplicaciones WAP.

El WAP surge de la convergencia de dos tecnologías de redes de datos: las comunicaciones inalámbricas e Internet. El *Wireless Application Protocol* (WAP) es resultado del esfuerzo que realiza la organización *WAP Forum* (`http://www.wapforum.org/`)

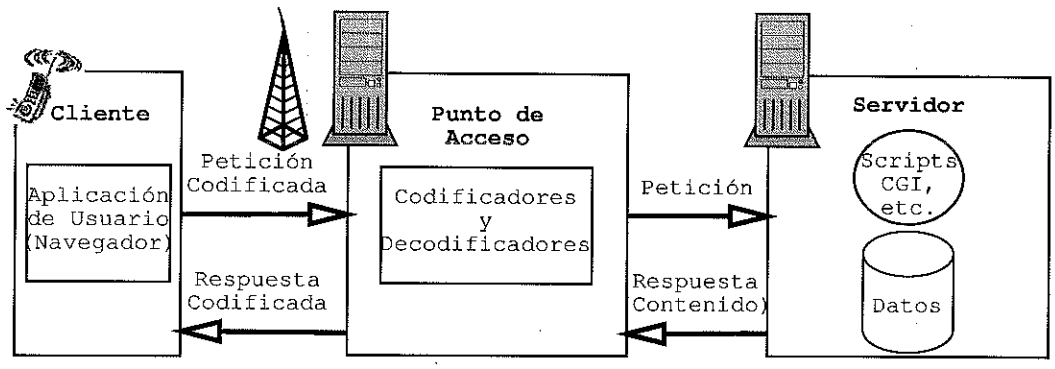


Figura 6: Modelo de programación WAP.

en la promoción de especificaciones de tecnología para el desarrollo de aplicaciones y servicios que operan sobre redes de datos inalámbricas. Entre los objetivos del WAP Forum está llevar a teléfonos celulares, y a otras terminales inalámbricas, contenido Internet y otros avanzados servicios de datos, así como crear una especificación de un protocolo de comunicaciones inalámbricas global que pueda operar a través de diversas redes inalámbricas. En este protocolo se especifica un entorno de desarrollo y varios protocolos de red que pueden ser utilizados en dispositivos inalámbricos como teléfonos celulares, radiolocalizadores y PDAs.

El modelo de programación definido en el WAP (ver Figura 6) es similar al que se utiliza en las aplicaciones WWW de Internet. Esto genera beneficios para la comunidad de desarrolladores de aplicaciones, tales como un modelo de programación familiar, una arquitectura ampliamente utilizada y probada y la posibilidad de seguir utilizando algunas de las herramientas existentes (e.g. servidores WWW, herramientas para el procesamiento de documentos XML, etc.) [WAP Forum, 1998]. Tanto el contenido, como las aplicaciones y los protocolos que forman parte de la especificación WAP están basadas en los formatos que se utilizan en el WWW. Aunque algunos de estos formatos y protocolos fueron modificados para operar en redes inalámbricas, se trató de

incorporarlos al modelo WAP sin cambio alguno, y solo adaptarlos cuando era realmente necesario.

La aplicación de usuario que se muestra en la Figura 6 es la mayoría de las veces un *micro-navegador*, una versión reducida de los desplegadores de documentos HTML de escritorio. Estos micro-navegadores se ejecutan en los dispositivos inalámbricos (el cliente) y se encargan de enviar peticiones de documentos a los puntos de acceso, recibir los documentos y presentarlos al usuario. El intercambio de información entre los dispositivos cliente y los puntos de acceso está comúnmente codificado en WML (*Wireless Markup Language*), un lenguaje similar al HTML, optimizado para utilizarse en terminales móviles de bolsillo [WAP Forum, 1998].

Los puntos de acceso, dentro de la especificación WAP, se encargan de traducir las peticiones en formato WML que reciben de los clientes inalámbricos, a peticiones en protocolos como TCP/IP y HTTP, que pueden ser reconocidos en Internet. Los codificadores y decodificadores realizan labores de traducción entre el contenido que se maneja en el WAP y tipos de datos compactos, con el fin de reducir el consumo de recursos de red. El resto del modelo WAP es el mismo que se utiliza en la mayoría de las aplicaciones de Internet.

Una vez descritas algunas de las arquitecturas utilizadas en aplicaciones colaborativas móviles se presenta una serie de escenarios de utilización de otras aplicaciones, para ilustrar algunos aspectos particulares de este tipo de sistemas y complementar la representación más general que hacen las arquitecturas introducidas anteriormente.

III.3 Escenarios típicos de utilización.

Un caso de uso es esencialmente una interacción típica entre un usuario y un sistema de cómputo, es un escenario de utilización en el que se pretende capturar la funcionalidad de la aplicación desde el punto de vista del usuario [Fowler y Scott, 1997]. Estos escenarios son representados frecuentemente por medio de diagramas basados en la notación gráfica descrita en [UML-RTF, 1999]. En esta notación se representa a los usuarios del sistema (llamados actores) como figuras de palo y a la funcionalidad del sistema como óvalos con etiquetas que explican las metas que el usuario alcanza cuando utiliza el sistema.

A continuación se presentan algunos escenarios de utilización típicos de aplicaciones colaborativas móviles, junto con sus respectivos diagramas de uso. Además de ilustrar los requerimientos frecuentemente encontrados en las aplicaciones colaborativas móviles, los escenarios de utilización auxiliaron en la identificación de los componentes principales de tales aplicaciones, así como en encontrar el papel que cada uno de éstos componentes juega dentro de las aplicaciones.

III.3.1 Calendario de actividades y lista de contactos.

En este caso, (ilustrado en la Figura 7) la aplicación móvil es un calendario de actividades, una lista de contactos, una aplicación para mantener estos datos actualizados y un sistema para el envío de mensajes. Una copia de la información del calendario y la lista de contactos se mantiene en una aplicación o servicio externo (como *ICQ* `http://www.icq.com` ó `http://www.anyday.com`). Entre esta aplicación externa y la aplicación móvil puede haber una comunicación directa o una computadora que cumpla con la función de punto de acceso.

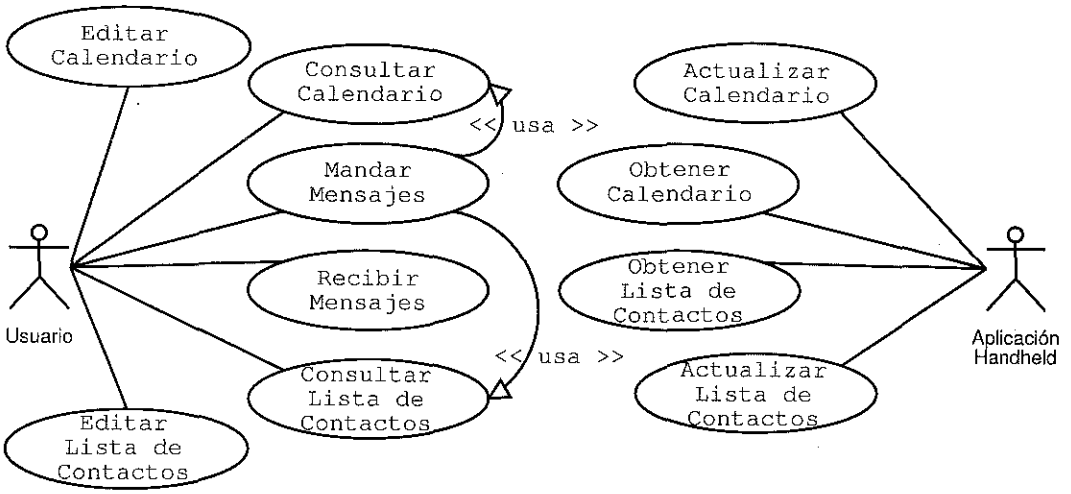


Figura 7: Caso de uso de aplicación de calendario de actividades.

De esta forma, el usuario puede actualizar su calendario y utilizar su lista de contactos mientras se encuentra fuera de su lugar de trabajo. Puede incluso mantenerse en contacto con las personas con las que realiza alguna actividad colaborativa enviándoles mensajes. Junto con la información del calendario de actividades, estos mensajes pueden informar acerca del trabajo relacionado con un proyecto de trabajo o utilizarse como medio para establecer horarios de reunión. Los mensajes generados mientras el usuario no tiene conexión a Internet son almacenados en la computadora móvil y enviados cuando la aplicación o servicio externo se encuentran disponibles.

A través de esta aplicación el usuario puede también dar a conocer a las personas con las que trabaja si se encuentra disponible, ocupado, en tránsito, utilizando una computadora de escritorio o en tránsito y utilizando una computadora móvil.

III.3.2 Sistema de información de un campus.

En este escenario una persona equipada con una computadora móvil personal visita un campus. El campus cuenta con un sistema de información y en sus accesos

existen terminales en donde el visitante puede cargar en su computadora móvil una aplicación cliente con la cual se puede accesar al sistema de información. El objetivo del sistema es el de proveer al usuario con información acerca de la organización del campus, nombres de edificios, direcciones de oficinas, información acerca de eventos a realizarse y acerca de las personas que trabajan en el campus. A lo largo y ancho del campus existen otras terminales o puntos de acceso desde donde se puede accesar y actualizar la información del sistema y consultar datos particulares al lugar en el que se encuentra el usuario. En el diagrama de utilización de este sistema (Figura 8), se puede observar que aún cuando los visitantes y el personal que labora en el campus pueden realizar algunas tareas comunes, existen algunas actividades reservadas solamente a los visitantes (e.g. registrarse) y otras tareas que solo puede realizar el personal del campus (e.g. recibir mensajes).

A través de este sistema de información, el visitante puede:

- Registrar su entrada al campus (esto se puede hacer al mismo tiempo que la aplicación cliente es cargada).
- Obtener información acerca de un edificio, oficina, pasillo o cuarto en particular.
- Obtener lista y horarios de eventos a realizarse en el campus o en un lugar en particular.
- Obtener una lista de las personas que se encuentran en el campus.
- Buscar y localizar a una de las personas que se encuentra en el campus o en algún edificio.
- Dejar un mensaje a alguna de las personas que laboran en el campus.

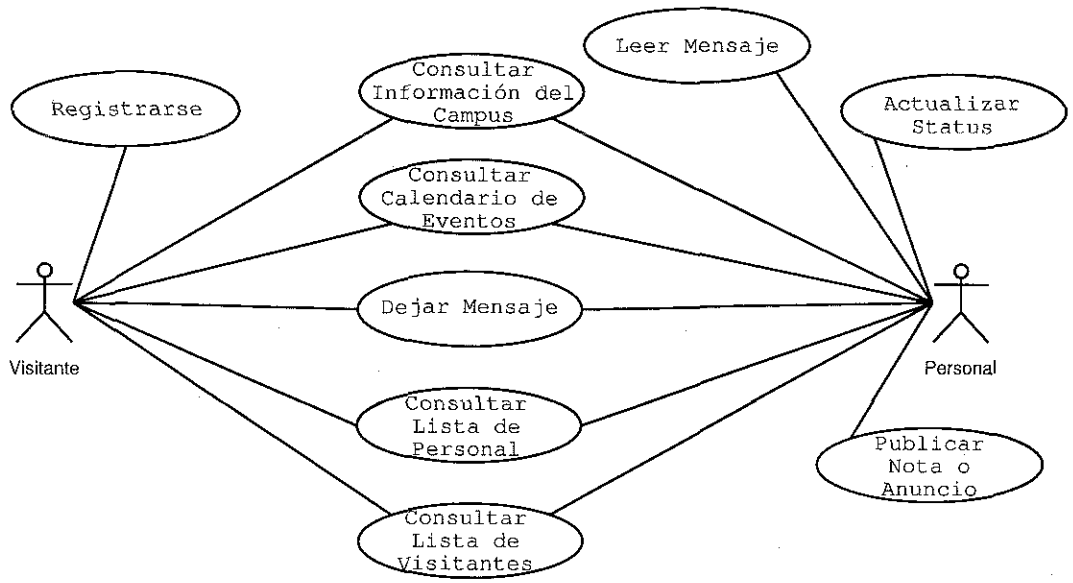


Figura 8: Caso de uso de aplicación de un sistema de información de un campus.

Las personas que laboran en el edificio también pueden utilizar este sistema de información, ya que además de poder realizar las actividades propias de un visitante, pueden:

- Publicar si se encuentran disponibles, ausentes, ocupados, en tránsito dentro del campus, etc.
- Recibir mensajes de visitantes o colegas.
- Dejar una nota o anuncio (el cual pudiera ser público o privado).

III.3.3 Puntos de acceso a Internet.

En este caso se plantea la utilización de puntos de acceso a Internet que pueden ser usados desde computadoras móviles personales.

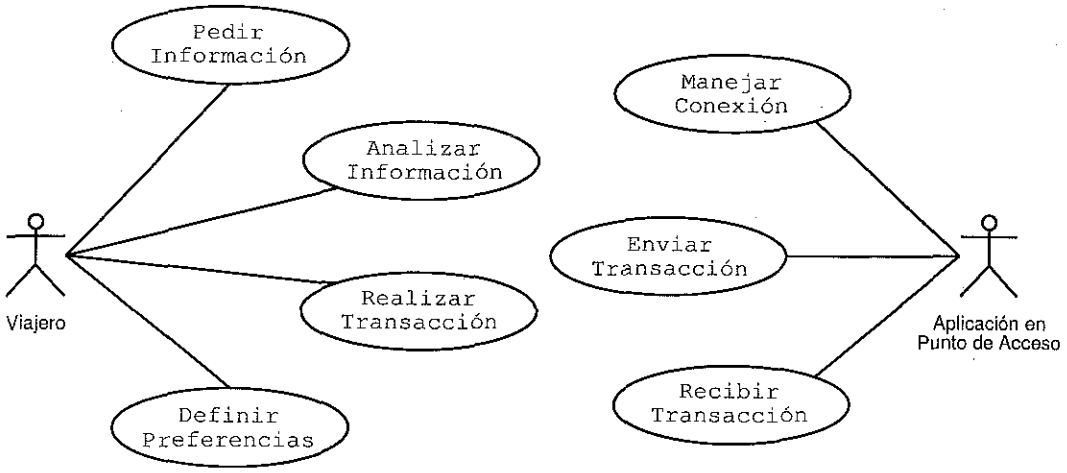


Figura 9: Diagrama de caso de uso de un punto de acceso a Internet.

La computadora móvil cuenta con una aplicación a través de la cual el usuario puede solicitar información disponible en Internet. La solicitud es transmitida al punto de acceso o terminal, la terminal obtiene la información y la envía la respuesta a la aplicación móvil, donde el usuario la puede analizar o procesar, incluso después de que la conexión entre la computadora móvil y el punto de acceso termina. En el diagrama de caso de uso de esta aplicación (Figura 9) los actores son el usuario y la aplicación que reside en el punto de acceso, la cual se encarga de intercambiar datos entre el dispositivo móvil e Internet.

La aplicación móvil puede también recibir peticiones del usuario mientras no existe conexión con algún punto de acceso. Estas peticiones son almacenadas en la computadora móvil y enviadas al establecer una conexión con un punto de acceso. La utilización del punto de acceso permite proporcionar al usuario información relativa al lugar en el que se encuentra y realizar intercambio de datos aún cuando existe alguna falla en la red de datos que conecta al punto de acceso con el resto de Internet, cuando la conexión se restablece, la transacción del usuario es enviada al lugar correspondiente.

La aplicación cliente puede también instruir al punto de acceso para recibir mensajes o información del usuario, (e.g. correo electrónico) disponibles en computadoras conectadas a Internet, de acuerdo a las preferencias que el usuario define.

Utilizando un esquema de este tipo un usuario pudiera pedir información acerca de los hoteles disponibles en el lugar a donde se dirige, transfiriendo la lista desde el punto de acceso a su computadora móvil. El viajero entonces podría revisar la lista, seleccionar uno de los hoteles y hacer la reservación mientras se encuentra en el avión o autobús que lo lleva a su destino. La próxima vez que el usuario conecta su computadora móvil personal a uno de los puntos de acceso (podría ser en la terminal de arribo) la reservación es enviada.

III.3.4 Reunión de trabajo.

Este es el escenario en el cual los asistentes a una reunión de trabajo cuentan con computadoras personales móviles, el lugar de reunión tiene una o varias computadoras como apoyo a las sesiones de trabajo y cuando menos una de éstas puede servir como punto de acceso para distribuir y recibir información desde y hacia las computadoras móviles.

Cada participante en la reunión puede entonces distribuir información personal, mensajes, datos y presentaciones a los demás (la distribución puede hacerse usuario a usuario ó usuario a punto de acceso a usuario). En la Figura 10 se ilustra el escenario de utilización de esta aplicación. La Aplicación de Sala de Reuniones que se presenta en el diagrama es la que sirve como punto de distribución de información cuando se lleva a cabo una reunión.

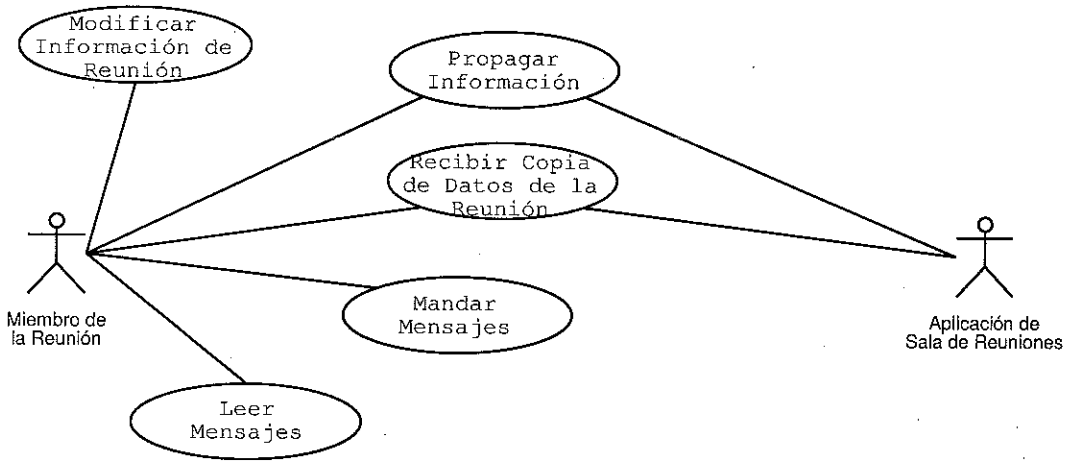


Figura 10: Caso de uso de aplicación de Sala de Reuniones.

La información de la computadora móvil puede transferirse desde una pantalla pública hacia una computadora móvil para hacer modificaciones y desde la computadora móvil hacia la pantalla pública para distribuir los cambios, de una manera similar a la que se describe en [Greenberg *et al.*, 1999].

Los participantes pueden también recibir una copia exacta de la información generada durante la reunión (notas, diagramas, modificaciones, etc.) para su modificación o consulta posterior o para distribuirla a otras personas.

Se puede observar que en las aplicaciones de los escenarios de utilización presentados anteriormente existen componentes y funcionalidad comunes. Por ejemplo, el componente de la aplicación del punto de acceso a Internet que se encarga de enviar y recibir transacciones que el usuario realiza desde su PDA es similar a la aplicación de la Sala de Reuniones del último caso de uso. Una característica común de las aplicaciones de los escenarios es que son similares a algunas aplicaciones colaborativas de escritorio existentes, con la diferencia de que en los escenarios descritos los usuarios pueden utilizar los sistemas aún cuando se encuentran en situaciones de movimiento.

III.4 Requerimientos de una herramienta de soporte al desarrollo de aplicaciones colaborativas móviles.

El estudio de los escenarios de utilización de las aplicaciones presentadas anteriormente y un análisis de las arquitecturas mencionadas al principio de este capítulo fueron las bases de la herramienta de desarrollo que se propone en este trabajo de tesis. El entorno de desarrollo que se propone es llamado COMAL, un término que agrupa las siglas que significan *Collaborative Mobile Application Library.* El entorno COMAL está compuesto por una arquitectura y un par de bibliotecas para desarrollar aplicaciones colaborativas móviles.

El entorno para aplicaciones colaborativas móviles que se propone se construyó bajo los siguientes requerimientos:

- *Compatible con las aplicaciones colaborativas actuales.* La mayoría de las aplicaciones colaborativas han sido diseñadas para computadoras de escritorio. Utilizando el entorno de desarrollo que se propone, estas aplicaciones colaborativas de escritorio pueden integrarse con dispositivos móviles, utilizando a las computadoras de escritorio como puntos de contacto entre las aplicaciones colaborativas existentes y los sistemas que se ejecutan en las computadoras móviles personales.

- *Habilidad de operar sin acceso a una red de datos.* En los escenarios de las aplicaciones presentadas anteriormente, el acceso a una red de datos no era continuo. De hecho, la utilización de éste recurso es mínimo, por eso es que el entorno propuesto debe proveer mecanismos para almacenar temporalmente transacciones cuando no hay acceso a una red de datos y enviarlas cuando el acceso a la red es establecido.

- *Funcionalidad consistente.* Las bibliotecas utilizadas en este entorno proveen un conjunto consistente de llamadas de funciones para ser usadas tanto en aplicaciones colaborativas móviles como en sus contrapartes de escritorio (las aplicaciones que comunican a los dispositivos móviles con otros recursos). Estas llamadas pueden ser utilizadas como una base sobre la que se desarrollen aplicaciones colaborativas móviles, siguiendo una estructura predefinida.

- *Diferentes modos de comunicación.* Este entorno tiene soporte para intercambiar información a través de sincronización de datos (entre PDAs y computadoras de escritorio) y a través de conexiones punto a punto (para accesar recursos de una red de datos, cuando ésto es posible).

De esta manera, las personas que utilicen este entorno de desarrollo pueden enfocar su atención en construir el sistema colaborativo móvil, utilizando el esquema de comunicación que COMAL provee, y no en hacer programación de *bajo nivel.*

Encapsular llamadas de funciones en una biblioteca compartida permite que varias aplicaciones colaborativas móviles compartan el mismo código, ahorrando espacio de memoria en el dispositivo móvil donde las aplicaciones se encuentran instaladas. Este también es considerado como un requerimiento en las bibliotecas que se construyeron, debido a los limitados recursos de memoria de los PDA actuales.

III.5 Arquitectura de las aplicaciones colaborativas móviles de COMAL.

De las características de los escenarios de utilización descritos en la Sección se identificaron los siguientes componentes principales, dentro de las aplicaciones colaborativas móviles:

- *Servidores de aplicaciones.* Estos servidores almacenan información acerca de aplicaciones colaborativas, información que puede ser accesada a través de una red de datos. Estos datos pueden incluso estar distribuidos entre varios servidores y bases de datos en diferentes lugares.

- *Una terminal o computadora de escritorio.* Esta computadora es utilizada para accesar información disponible en la red o en el dispositivo móvil.

- *Una aplicación móvil.* Esta es la porción móvil de la aplicación colaborativa móvil que puede ser utilizada mientras el usuario se encuentra alejado de alguna computadora de escritorio. Este tipo de aplicaciones también son frecuentemente utilizadas para colaborar con otras personas que utilizan dispositivos móviles similares o con otros dispositivos instalados en paredes o kioscos, de la forma que se describe en la Sección .

Otro componente importante, siempre presente cuando se lleva a cabo un proceso de colaboración, son los canales de comunicación a través de los cuales intercambian datos las entidades enlistadas anteriormente.

La relación entre estos componentes es ilustrada en la Figura 11. Esta es la organización de una aplicación colaborativa móvil que se propone en este trabajo, en ella se puede observar que las bibliotecas COMAL (denotadas por un *), solamente existen en las computadoras de escritorio y en los dispositivos móviles, el resto de la aplicación colaborativa móvil es muy similar a las aplicaciones colaborativas tradicionales.

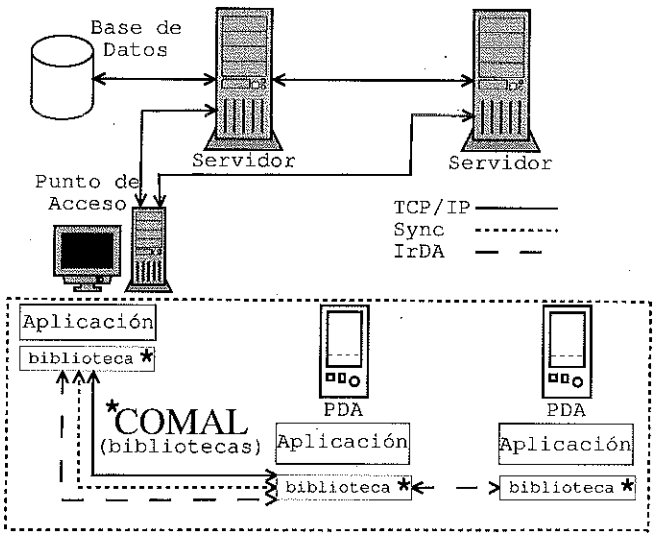


Figura 11: Arquitectura de una aplicación colaborativa móvil utilizando COMAL.

III.5.1 Dispositivo móvil.

Al planear el desarrollo del entorno COMAL surgió la necesidad de utilizar un PDA para estudiarlo, hacer pruebas con él y posteriormente usarlo para desarrollar, implementar y probar las bibliotecas que forman parte de COMAL. De las opciones que existen en el mercado se decidió escoger la Plataforma de Cómputo Palm.

Los dispositivos Palm tienen varias ventajas sobre otras plataformas: es, por mucho, el PDA más utilizado alrededor del mundo, es más aceptado socialmente que otras alternativas [Greenberg *et al.*, 1999], tiene un bajo nivel de consumo de energía y existe una gran variedad de herramientas de desarrollo para su sistema operativo (PalmOS), algunas de ellas gratuitas. Además, esta plataforma de cómputo ya ha sido utilizada en Handheld CSCW, como se documenta en [Dittrich y Thomas, 1998], [Myers *et al.*, 1998] y [Greenberg *et al.*, 1999].

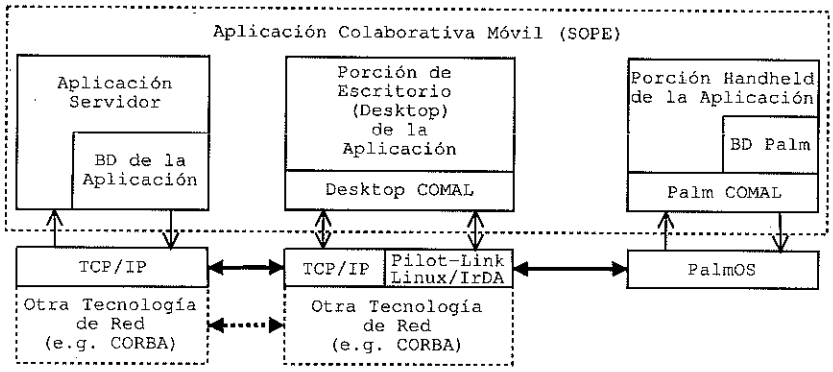


Figura 12: Arquitectura de software utilizada en el entorno COMAL.

III.5.2 Arquitectura de software COMAL.

La arquitectura de software utilizada en el entorno de desarrollo COMAL se ilustra en la Figura 12. En esta figura, el rectángulo punteado que contiene a las 3 aplicaciones representa a la aplicación colaborativa móvil, que dentro del entorno de COMAL es conocida como SOPE (*Shared Objects for Portable Environments*). En la implementación actual de las bibliotecas COMAL se utiliza el protocolo TCP/IP para intercambiar información entre las computadoras de escritorio cliente y los servidores de aplicación, pero como se muestra en la arquitectura de software, otros paradigmas de comunicación pueden ser usados, por ejemplo, CORBA. Para intercambiar datos entre computadoras de escritorio y los PDAs COMAL utiliza *pilot-link*, un conjunto de herramientas desarrolladas para accesar las bases de datos de los dispositivos Palm desde computadoras UNIX. COMAL también permite el intercambio de datos a través de dispositivos de comunicación infrarroja conectados al punto de acceso, utilizando la implementación del protocolo IrDA para el sistema operativo Linux, Linux/IrDA.

La capa PalmOS, que se ubica abajo de la aplicación móvil, contiene varios mecanismo para intercambiar información con computadoras de escritorio, incluso permite establecer conexiones TCP/IP directamente con aplicaciones servidor, utilizando la

implementación que el PalmOS contiene de este protocolo, base fundamental de Internet.

En el capítulo siguiente se describen más detalladamente las bibliotecas COMAL, se presentan sus características, sus componentes y la forma en que pueden ser utilizadas y extendidas. En ese Capítulo también se explica la estructura de los SOPEs y su relación con las bibliotecas COMAL.

Capítulo IV.

Bibliotecas COMAL para el desarrollo de aplicaciones colaborativas móviles.

IV.1 Descripción y objetivos de las bibliotecas.

El encapsulamiento de rutinas de código en bibliotecas compartidas (*shared libraries*) provee un mecanismo para la reutilización de funcionalidad entre dos o más aplicaciones, provocando una reducción en el tiempo de desarrollo de nuevos sistemas. Debido a que estas bibliotecas pueden ser utilizadas por varias aplicaciones al mismo tiempo, también se reduce el espacio de memoria que estos sistemas ocupan cuando se encuentran ejecutándose y cuando están almacenados. El tiempo de vida de este tipo de bibliotecas no esta restringido a la duración de las aplicaciones que las utilizan y además, el mantenimiento del código contenido en las bibliotecas puede realizarse independientemente de las aplicaciones. Arquitectónicamente, el uso de bibliotecas compartidas reduce la dependencia entre los módulos de una aplicación, dando como resultado un sistema mas estable y fácil de mantener [Bachmann, 1999].

Por otro lado, existen dos casos en los que no es conveniente utilizar bibliotecas compartidas. El primer caso se dá cuando el tiempo en que se ejecuta una aplicación

es un factor crítico, ya que el lapso que toma realizar el proceso de enlace entre la aplicación y la biblioteca compartida requerida se añade al tiempo de ejecución de la aplicación. Incorporando de manera estática las rutinas de la biblioteca dentro de la aplicación mejora los tiempos de ejecución, sin embargo, los casos en que la mejora es substancial son muy raros. El otro caso en el que no es conveniente utilizar bibliotecas compartidas es cuando la aplicación se desarrolla para un sistema operativo que no tiene soporte para este tipo de bibliotecas [Valley *et al.*, 1992].

Debido a las ventajas de usar este tipo de bibliotecas y a que éstas pueden utilizarse en una gran variedad de sistemas, incluso en los dispositivos que utilizan PalmOS, se decidió implementar las bibliotecas COMAL como un conjunto de bibliotecas compartidas.

El objetivo principal de las bibliotecas COMAL es el de proveer una base sobre la cual se puedan construir aplicaciones colaborativas móviles o integrar dispositivos móviles a aplicaciones colaborativas de escritorio existentes. Estas bibliotecas proveen llamadas de funciones para transmitir, recibir y sincronizar datos con, y desde, dispositivos Palm. Por medio de las bibliotecas también es posible recibir peticiones de información creadas en el dispositivo Palm, procesar la transacción en una computadora de escritorio o punto de acceso y almacenar el resultado hasta la próxima vez que la computadora Palm pueda ser accesada. Además de poder intercambiar información entre computadoras de escritorio y Palms utilizando el cable serial con el que regularmente se lleva a cabo la sincronización de información, COMAL tiene funciones para enviar y recibir unidades de información a través del dispositivo de comunicación infrarroja con que cuentan algunos modelos de computadoras Palm. A continuación se describe la estructura de las bibliotecas COMAL y su relación con otras bibliotecas que proveen acceso a métodos de intercambio de información de bajo nivel.

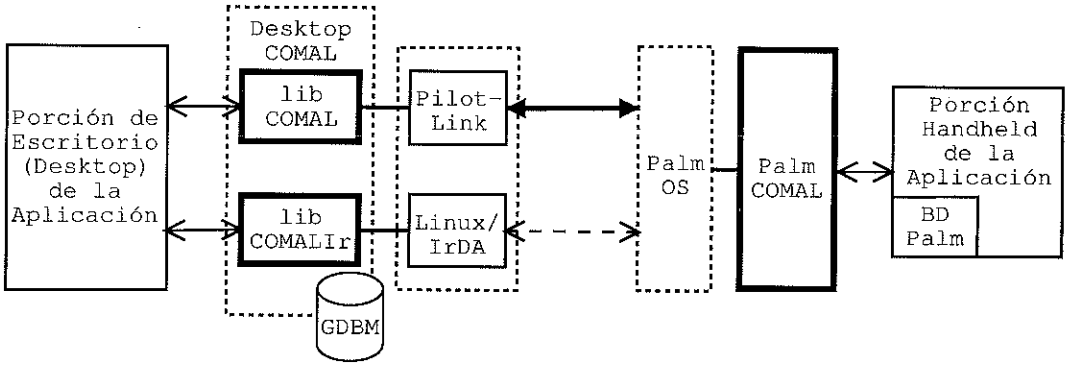


Figura 13: Las 3 bibliotecas que forman parte de COMAL y el lugar que ocupan dentro de una aplicación colaborativa móvil.

IV.2 Estructura de las bibliotecas COMAL.

COMAL esta compuesto por las 3 bibliotecas que se muestran en la Figura 13, libCOMAL, libCOMALIr y Palm COMAL, esta figura es una vista detallada de una de las secciones de la arquitectura de software de COMAL (Figura 12). LibCOMAL y libCOMALIr residen en las computadoras de escritorio que funcionan como puntos de acceso para los dispositivos Palm, dispositivos en los que reside la biblioteca Palm COMAL.

Con las bibliotecas COMAL es posible construir aplicaciones en las que se manejan los siguientes tipos de datos:

- *Eventos.* Esta es información referente a una actividad que se lleva a cabo durante un período de tiempo. En este tipo de datos se almacena la fecha del evento, hora de inicio y terminación, una descripción de la actividad y otros atributos.

- *Usuarios.* En este tipo de datos se almacena la información acerca de una persona: nombre, apellido, dirección, números telefónicos y otros datos personales.

- *Mensajes.* Un mensaje en COMAL es un segmento de texto que una persona envía a otra. Además del contenido del mensaje, este tipo de datos registra el identificador de usuario del remitente.

- *Segmentos de texto.* Estos son pequeños archivos de texto que se pueden utilizar como notas o recados.

La elección de estos tipos de datos se hizo tomando como base el análisis de los escenarios de utilización del capítulo anterior. Además, estos tipos de datos son comúnmente utilizados en aplicaciones colaborativas de escritorio tradicionales. Otros tipos de datos pueden ser agregados a esta lista, añadiendo las rutinas correspondientes a las bibliotecas COMAL (ver Sección).

Las dos bibliotecas COMAL que residen en los puntos de acceso almacenan la información de los tipos de datos anteriormente mencionados en archivos con el formato GDBM (*The GNU database manager*). El acceso y almacenamiento se realiza utilizando las bibliotecas GNU dbm, un conjunto de rutinas para manejar archivos que contienen pares de datos del tipo *llave/valor.* Las rutinas proveen mecanismos para ordenar, consultar y borrar datos, por medio de las llaves de los pares almacenados y para recorrer todos los datos de archivos con el formato GDBM. Los datos que se envían desde el dispositivo Palm hacia una aplicación que utiliza COMAL son almacenados en archivos GDBM para que sean posteriormente procesados por la aplicación. En las subsecciones siguientes se describe cada una de las bibliotecas COMAL.

IV.2.1 LibCOMAL.

A través de esta biblioteca se intercambia información entre una aplicación de escritorio y una aplicación en el dispositivo Palm utilizando el cable serial con el que

se realiza frecuentemente la sincronización de datos. La comunicación y la lectura y escritura de los datos del PDA Palm se realiza haciendo llamadas a las rutinas de bajo nivel de las bibliotecas pilot-link. Utilizando pilot-link se pueden establecer conexiones punto a punto, entre computadoras de escritorio y Palms, similares a las que se obtienen cuando se utilizan *sockets* tipo BSD. Una vez establecida una de estas conexiones, es posible accesar la información de las bases de datos que utiliza el PalmOS para almacenar información en los PDA Palm.

En libCOMAL la rutina para establecer una conexión con PDAs Palm es `comal_open_palm_connection`. Después de haber establecido la conexión se pueden emplear rutinas como `comal_send_events_to_palm`, con la cual se envían a las base de datos del PDA Palm todos los eventos que están almacenados en un archivo GDBM, o `comal_get_events_from_palm`, con la que se lee la información de los eventos que están registrados en el dispositivo Palm y se almacenan en un archivo GDBM con el fin de que puedan ser accesados posteriormente, desde una aplicación de escritorio, en caso de ser necesario. Existen llamadas de funciones similares para el resto de los tipos de datos que pueden ser utilizados en el entorno COMAL, en el Apéndice A.3.2 se describen de manera detallada las rutinas de libCOMAL, libCOMALIr y Palm COMAL. Cuando la comunicación entre la aplicación de escritorio y el dispositivo Palm ya no es necesaria, la conexión es terminada llamando a la rutina `comal_close_palm_connection`.

La biblioteca libCOMAL también contiene las rutinas con las que se consulta y registra la información de los archivos GDBM que COMAL utiliza para almacenar la información de las aplicaciones colaborativas móviles. El formato de los pares llave/valor que se almacenan en los archivos GDBM que utiliza COMAL es especial, sin embargo, utilizando las rutinas de libCOMAL es innecesario conocerlo, ya que existen llamadas a funciones como `comal_save_event_info_to_db` a la que se le pasa como parámetro una

estructura con la información de un evento y libCOMAL se encarga de almacenarlo en el formato correspondiente.

IV.2.2 LibCOMALIr.

Esta biblioteca es la que se encarga de enviar y recibir unidades de información de la aplicación colaborativa móvil a través del protocolo de comunicación infrarroja IrOBEX (*Infra red Object Exchange Protocol*). A diferencia de libCOMAL, la implementación actual de libCOMALIr sólo puede enviar o recibir una estructura de datos u objeto, de los definidos en la Sección , por cada sesión de intercambio de información.

La forma en que se realiza la transferencia de datos en libCOMALIr también es diferente a como se hace en la biblioteca libCOMAL. Para transmitir información, primeramente se inicializa el protocolo OBEX llamando a la rutina `comalir_start_obex`, después se crea el objecto a enviar llamando una rutina como `comalir_create_event_object`, dependiendo del tipo de dato que se trate. Una vez creado el objeto a enviar se hace una petición de conexión, llamando la rutina `comalir_connect_request` y cuando se recibe la notificación de que la conexión ha sido establecida libCOMALIr se encarga de transmitir el objeto deseado.

Cuando en vez de enviar datos se requiere recibirlos simplemente se inicializa el protocolo OBEX y se manda llamar la rutina `comalir_install_listening_server`. Una vez que se ejecuta la rutina anterior solo resta esperar que libCOMALIr reciba el objeto y lo almacene en el lugar correspondiente.

LibCOMALIr utiliza la implementación del protocolo IrDA para el sistema operativo Linux, Linux/IrDA, por lo que, para poder utilizar esta biblioteca de COMAL es necesario instalar Linux/IrDA de la forma que se describe en [Heuser, 1999].

IV.2.3 Palm COMAL.

La función de la biblioteca Palm COMAL, que se instala en los dispositivos Palm, es la de almacenar y accesar información relacionada con los tipos de datos de usuarios y mensajes que se utilizan en el entorno COMAL. Para los tipos de datos eventos y texto, el PalmOS cuenta con aplicaciones y bases de datos para su manejo, por lo que no es necesario agregar rutinas para estos tipos de datos. De hecho, los dispositivos Palm también manejan información acerca de usuarios a través de la aplicación *Address Book*, sin embargo, la información de usuarios que maneja Palm COMAL permite crear listas de contactos que asocian a cada usuario un identificador único, algo difícil de lograr utilizando simplemente el soporte del PalmOS para registrar información acerca de personas.

Esta biblioteca también contiene rutinas para enviar y recibir datos de usuarios y mensajes a través del puerto infrarrojo de los dispositivos Palm y OBEX. Algunas de estas funciones son `BeamMessage`, `BeamContact` y `ReceiveBeam`.

Una vez presentadas las bibliotecas COMAL, a continuación se describen las aplicaciones que son construidas con ellas, los SOPEs.

IV.3 SOPEs.

En aplicaciones de Handheld Computing, los objetos que contienen datos viajan desde servidores y computadoras de escritorio hacia los dispositivos móviles y viceversa. Cuando este tipo de aplicaciones son también colaborativas, estos objetos en movimiento son además compartidos con un conjunto de personas (en este escenario no solamente son compartidos los datos, sino también las aplicaciones, como en el

escenario de utilización descrito en la Sección). Debido a esto las aplicaciones colaborativas móviles construidas utilizando el entorno de desarrollo COMAL son llamadas SOPEs (*Shared Objects for Portable Environments*, Objetos Compartidos para Ambientes Portátiles).

De acuerdo a la estructura de la arquitectura COMAL, el SOPE utiliza las bibliotecas COMAL como una estructura de comunicación para intercambiar datos entre los componentes de la aplicaciones externas, permitiendo con esto su integración con aplicaciones colaborativas ya existentes.

Aparentemente un SOPE es solamente el componente móvil de una aplicación colaborativa móvil, debido a que el resto de la aplicación es igual a muchos de los sistemas groupware de escritorio y a que el entorno de desarrollo COMAL involucra principalmente aspectos de comunicación entre computadoras de escritorio y PDAs y entre PDAs. Sin embargo, el hecho de que utilizando las bibliotecas COMAL se pueden accesar recursos disponibles en computadoras servidor remotas, extiende la extensión del SOPE más allá del punto de acceso. Es por eso que en la figura 12 el SOPE ocupa todo el espacio de la aplicación colaborativa móvil.

Las bibliotecas COMAL también pueden ser mezcladas con otras tecnologías para desarrollar SOPEs, por ejemplo, se puede construir una aplicación colaborativa móvil utilizando CORBA en la porción de escritorio de la aplicación y COMAL para manejar los escenarios de movilidad, tomando ventaja de las mejores características de las dos herramientas.

IV.4 Utilización.

La forma en que se utilizan las dos bibliotecas COMAL que se instalan en las computadoras de escritorio es la misma. LibCOMAL y libCOMALIr (libCOMAL.so y libCOMALIr.so) deben encontrarse en un directorio donde puedan ser localizadas por el compilador cuando se ejecuta la sección de enlazado en la compilación de la aplicación que se construye (e. g. /lib o /usr/lib). Además, el programa que utiliza las bibliotecas debe contener los archivos de definiciones libCOMAL.h o libCOMALIr.h dependiendo de la biblioteca respectiva. Por ejemplo, si en el programa messenger.c se hacen llamadas a rutinas de las dos bibliotecas, se deben agregar las siguientes dos líneas de código en la parte inicial del programa:

```
#include "libCOMAL.h"
#include "libCOMALIr.h"
```

suponiendo que los archivos de definiciones de las bibliotecas comal se encuentran en el mismo directorio que messenger.c.

Una vez añadidos los archivos de definiciones y teniendo las bibliotecas en un directorio donde puedan ser localizados por el compilador el programa messenger.c puede compilarse de la siguiente manera:

```
gcc -lCOMAL -lCOMALIr -o messenger messenger.c
```

suponiendo que gcc es el compilador de lenguaje C que se está utilizando.

En el caso de la biblioteca Palm COMAL, la biblioteca se carga dentro del código de la aplicación como cualquier otra biblioteca compartida dentro del PalmOS, llamando la rutina:

```
Err err = SysLibFind( "COMAL Library", &comal_ref_num );
```

una vez que se obtiene la referencia a la biblioteca (en `comal_ref_num`), se carga llamando la función:

```
CmlOpen( comal_ref_num );
```

Cuando ya no es necesario utilizar la biblioteca (como cuando la aplicación se termina), Palm COMAL se da de baja con la rutina:

```
CmlClose( comal_ref_num );
```

IV.5 Extensiones a las bibliotecas.

Hacer extensiones a las bibliotecas COMAL involucra añadir las rutinas necesarias al código de las bibliotecas, a continuación se muestra una lista de pasos a seguir para hacer las adiciones correspondientes.

1. Definir la estructura del nuevo tipo de objeto al que se quiere dar soporte en las bibliotecas COMAL.

2. Agregar las rutinas para salvar, accesar y borrar el nuevo tipo de objeto en archivos GDBM, dentro de la biblioteca libCOMAL.

3. Agregar rutinas en libCOMAL para intercambiar el nuevo tipo de objeto entre los archivos GDBM y las bases de datos del dispositivo Palm.

4. Agregar rutinas a la biblioteca libCOMALIr para enviar y recibir el nuevo tipo de objeto por medio del protocolo IrOBEX.

5. Agregar a Palm COMAL el soporte necesario para el nuevo tipo de dato, esto podría involucrar la creación de una nueva base de datos en el dispositivo Palm.

Al realizar estas adiciones es recomendable que se consulte la documentación de las bibliotecas COMAL, así como el código fuente de las mismas ya que el nuevo código debe ser muy similar al que se utiliza con los tipos de datos que actualmente soporta COMAL.

En este capítulo se han descrito cada una de las bibliotecas COMAL, las aplicaciones colaborativas móviles que con ellas se construyen, es decir, los SOPEs, la forma en que las bibliotecas son utilizadas y de manera general, se presentó un bosquejo de los pasos a seguir para extender la funcionalidad de las bibliotecas para nuevos tipos de datos. En el capítulo siguiente se muestra Group Messenger, un SOPE construido utilizando el entorno de desarrollo COMAL. Además de la descripción de la aplicación, a lo largo del capítulo se muestran cada uno de sus componentes y algunas imágenes que ilustran el funcionamiento del sistema.

Capítulo V.

Group Messenger: aplicación construida con la biblioteca de desarrollo de aplicaciones colaborativas móviles.

V.1 Descripción de la aplicación.

Con el objetivo de probar el entorno de desarrollo que se propone en este trabajo, su arquitectura y sus bibliotecas se desarrolló un SOPE (una aplicación colaborativa móvil) utilizando COMAL. La aplicación desarrollada tiene la funcionalidad de la aplicación descrita en el primero de los escenarios de utilización presentados anteriormente, es un sistema para el manejo de una lista de contactos y un calendario de actividades con soporte para el envío y recepción de mensajes de texto. Este SOPE fue llamado Group Messenger, debido a que su objetivo principal es mantener comunicado a un grupo de personas trabajando en un ambiente colaborativo.

Group Messenger tiene 3 componentes principales: un programa servidor escrito en lenguaje Perl, una aplicación de escritorio escrita en lenguaje C y una aplicación para dispositivos Palm, también escrita en C.

La información personal de los contactos o usuarios del sistema se almacena, de manera centralizada, en el servidor. Los datos de los usuarios incluyen: un identificador de usuario único dentro del sistema, nombre del usuario, alias, dirección, números telefónicos, contraseña y lista de contactos, entre otros datos. La información de los eventos consiste en: fecha del evento, hora de inicio, hora de terminación, descripción, información extra y banderas que indican si los datos son privados o si se debe activar una alarma cuando la fecha y hora del evento se encuentran próximos. La información acerca de las actividades del usuario también se almacena en el servidor.

V.2 Arquitectura de Group Messenger.

Tener toda la información almacenada en un solo lugar (el servidor) permite que los usuarios puedan accesarla desde diferentes computadoras de escritorio, incluso desde diferentes PDAs, utilizando otros puntos de acceso disponibles, sin tener que depender de información de datos que se encuentran únicamente en el escritorio del usuario. La arquitectura de Group Messenger se muestra en la Figura 14. Como se puede observar, esta aplicación cuenta con los 3 componentes frecuentemente encontrados en las aplicaciones colaborativas móviles: un servidor, computadoras de escritorio que funcionan como puntos de acceso y dispositivos móviles.

Las aplicaciones de escritorio son las encargadas de desplegar y editar la información que se encuentra en el servidor, pero al crear nuevos datos o modificar los existentes, deben ser almacenados en el servidor para que posteriormente dichos cambios puedan ser accesados desde otros puntos de acceso o PDAs, manteniendo la información actualizada. A continuación se describen cada uno de los componentes de Group Messenger, de manera individual.

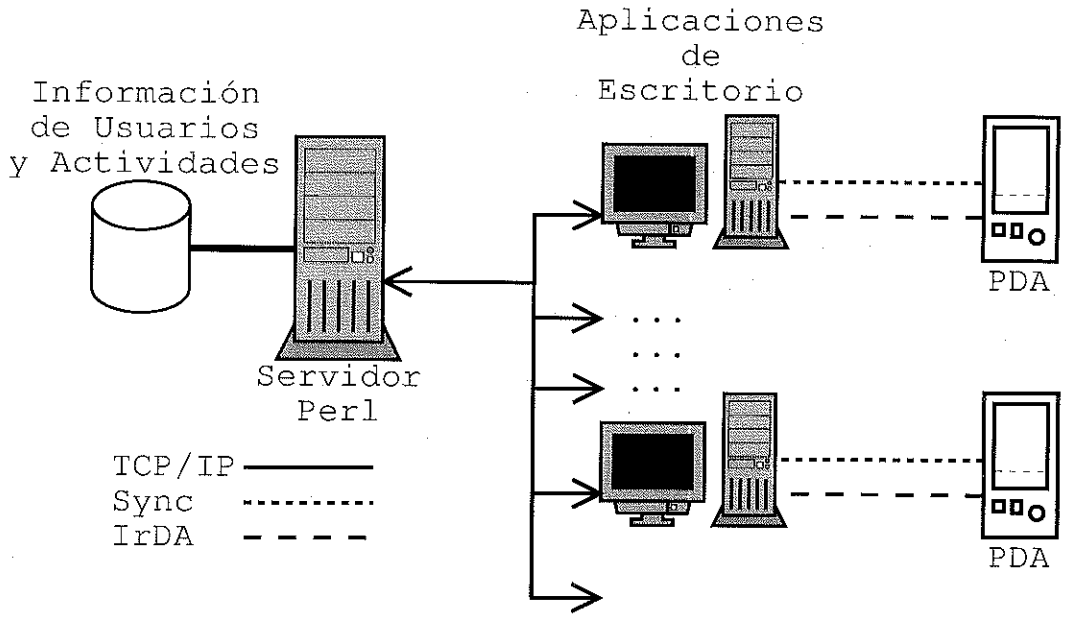


Figura 14: Arquitectura de Group Messenger.

V.2.1 Servidor.

El programa servidor de Group Messenger se encuentra esperando peticiones de conexión de manera permanente en un puerto de red preestablecido, de la computadora en la que se encuentra en ejecución. Al recibir una petición de un programa cliente se establece una conexión punto a punto entre el nuevo cliente y el servidor, a través de este enlace el cliente envía comandos al programa servidor, quien a su vez responde con el resultado del comando que el cliente pidió ejecutar. El servidor responde a comandos para consultar y modificar sus datos personales, enviar mensajes a otros usuarios, recibir mensajes, agregar y remover usuarios a su lista de contactos, etc. Los comandos que el servidor puede procesar se describen en el Apéndice A.3.2. Para que los usuarios puedan ejecutar algún comando en el servidor deben primero accesar al sistema con su identificador de usuario y contraseña correspondiente, con el fin de evitar interferencias entre usuarios del sistema, protegiendo la información personal de los mismos.

Debido al tipo de conexiones que se establecen entre el servidor y los clientes, se puede incluso interactuar con el servidor utilizando el programa de comunicación entre computadoras remotas llamado *telnet*, dándole como parámetros la dirección de la computadora donde se ejecuta el servidor y el número de puerto en el que se encuentra escuchando por conexiones, esto resultó muy útil para realizar las pruebas al servidor.

La información de los datos de los usuarios y sus agendas de actividades son almacenadas en el servidor en archivos de texto simple, con líneas de la forma *atributo = "valor"*. Por ejemplo, si el archivo de información de uno de los usuarios del sistema tiene una línea de texto `first_name = "Manuel"`, significa que el primer nombre del usuario es Manuel. Utilizar el lenguaje de programación Perl para el desarrollo del servidor facilitó el procesamiento de estos archivos, debido a la capacidad del lenguaje para el procesamiento de cadenas de texto.

V.2.2 Cliente.

Esta es la porción de Group Messenger que sirve como puente entre la aplicación servidor y el programa que se ejecuta en el dispositivo Palm. Además de proveer una interfaz gráfica para el envío de comandos al servidor de la aplicación, permite la edición de la información personal del usuario, su calendario de eventos y su lista de contactos, así como la recepción de información que el servidor genera.

Como se puede observar en la Figura 15, la interfaz de la aplicación de escritorio representa a los usuarios del sistema que forman parte de la lista de contactos del usuario actual como elementos de una lista, en el panel izquierdo de la ventana principal. A la derecha de la representación de los usuarios se despliega una etiqueta que informa cual es el estado de la conexión de los usuarios dentro del sistema. El estado de la

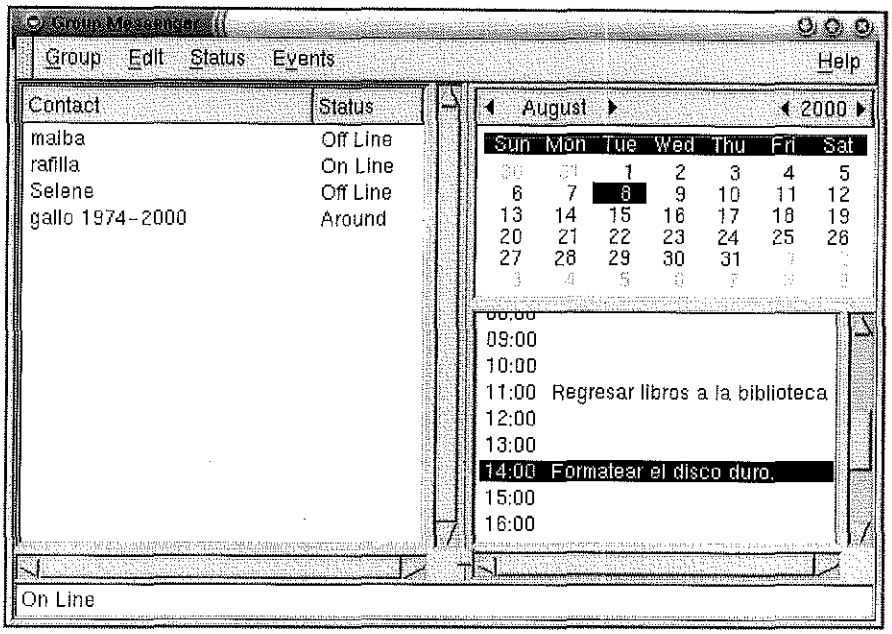


Figura 15: Ventana principal de la aplicación de escritorio de Group Messenger.

conexión del usuario actual se muestra en la parte inferior de la aplicación. El estado de conexión de los usuarios dentro de Group Messenger puede ser uno de los siguientes:

- On Line. Significa que el usuario se encuentra en su espacio de trabajo o utilizando su PDA con una conexión directa a la red de datos, y puede recibir y enviar mensajes cuando lo desee.
- Off Line. El usuario no puede ser contactado por el momento.
- Away (Away for a while). Significa que la persona no esta disponible actualmente pero que lo estará en cualquier momento.
- Busy. El usuario tiene una conexión directa a la red pero se encuentra ocupado y no desea ser molestado.
- Around. Este usuario se encuentra en tránsito de un lugar a otro utilizando su dispositivo móvil y puede establecer una conexión con la red de datos o sincronizar

su información en cualquier momento.

El usuario puede seleccionar alguno de los estados anteriormente descritos por medio de uno de los menús de la aplicación, sin embargo, cuando se detecta alguna falla en la conexión con el servidor, el estado es automáticamente cambiado a Off Line.

Con el fin de establecer la conexión con el servidor, el usuario puede cambiar su identificador de usuario, su contraseña e incluso cambiar la dirección y el puerto del servidor Group Messenger al que desea conectarse. Esto se realiza a través de la caja de diálogo de preferencias de la aplicación.

Al seleccionar alguno de los usuarios de la lista de contactos aparece una caja de diálogo con la cual se le puede enviar un mensaje de texto. A través de esta lista de contactos también se pueden desplegar los datos personales de los usuarios o remover un usuario de la lista personal. En la Figura 16 se muestra la vista que se obtiene cuando la aplicación recibe un mensaje de otro usuario, desde el servidor, y lo muestra en la pantalla.

El lado derecho de la aplicación se encuentra dividido en dos partes, en la parte superior se despliega un calendario y en la parte inferior se muestran las actividades programadas para el día seleccionado en el calendario, al cambiar la fecha seleccionada, la lista de actividades es actualizada. Los eventos pueden ser creados a través del menú de la aplicación o simplemente seleccionando la hora en la que se desea crear la actividad dentro del horario que aparece en la interfase. En la Figura 17 se muestra la caja de diálogo en la cual se editan las propiedades de un evento.

La edición de las propiedades de las actividades se realiza al seleccionarlas de entre la lista de actividades programadas que se encuentra bajo el calendario. En la

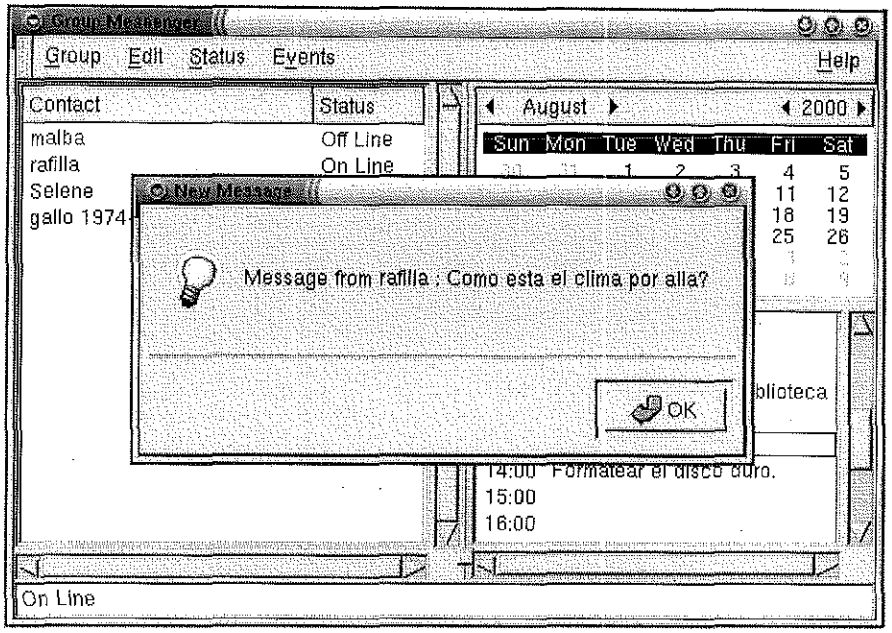


Figura 16: Recibiendo un mensaje de uno de los usuarios en la lista de contactos.

versión actual de Group Messenger, crear dos actividades a la misma hora provoca que la primer actividad se eliminada y que sólo quede la que se agregó al final.

Por medio de la aplicación del punto de acceso se pueden intercambiar datos acerca de los contactos en la lista del usuario y datos de eventos en el calendario, con el cliente móvil de Group Messenger. Por medio de las rutinas de libCOMAL puede sincronizarse la lista de usuarios completa, así como todo el calendario, por medio de la conexión serial entre la computadora de escritorio y el dispositivo Palm. Este intercambio de datos se puede realizar seleccionando la opción respectiva en el menú de la aplicación de escritorio.

También es posible intercambiar datos entre la aplicación de escritorio y dispositivos Palm utilizando comunicación infrarroja, sin embargo, en este caso solamente puede realizarse una transacción por cada conexión.

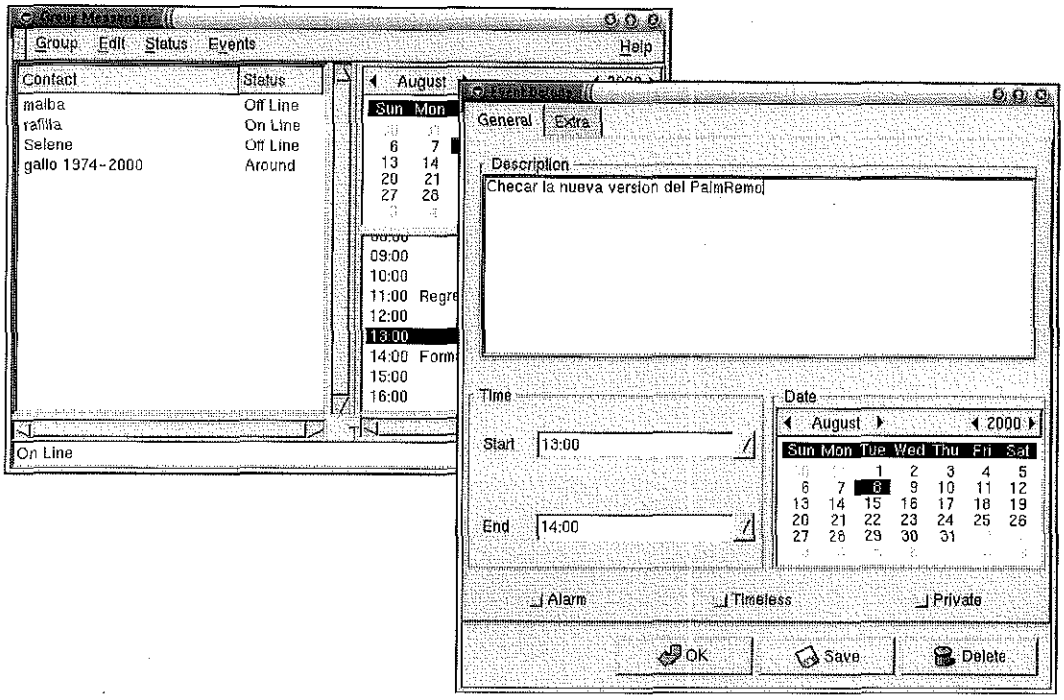


Figura 17: Editando un evento en el calendario de Group Messenger.

V.2.3 Cliente móvil.

La aplicación móvil que forma parte de Group Messenger se muestra en la Figura 18, su interfaz es similar a la de la aplicación de escritorio, con la diferencia que en el programa del PDA Palm los eventos son manejados por medio del programa *Date Book* que forma parte del PalmOS. La información personal de los contactos se almacena en los dispositivos Palm de dos maneras, la información completa es almacenada en la base de datos de la aplicación *Address Book* y el identificador de usuario, alias y estado de conexión del usuario es almacenado como parte de la aplicación móvil.

Los mensajes que el usuario envía y recibe son almacenados en dos bases de datos diferentes, con el objetivo de mandar hacia la aplicación de escritorio los mensajes que se envían y desplegar al usuario los mensaje que llegan desde la aplicación de escritorio después de realizar la sincronización de información.

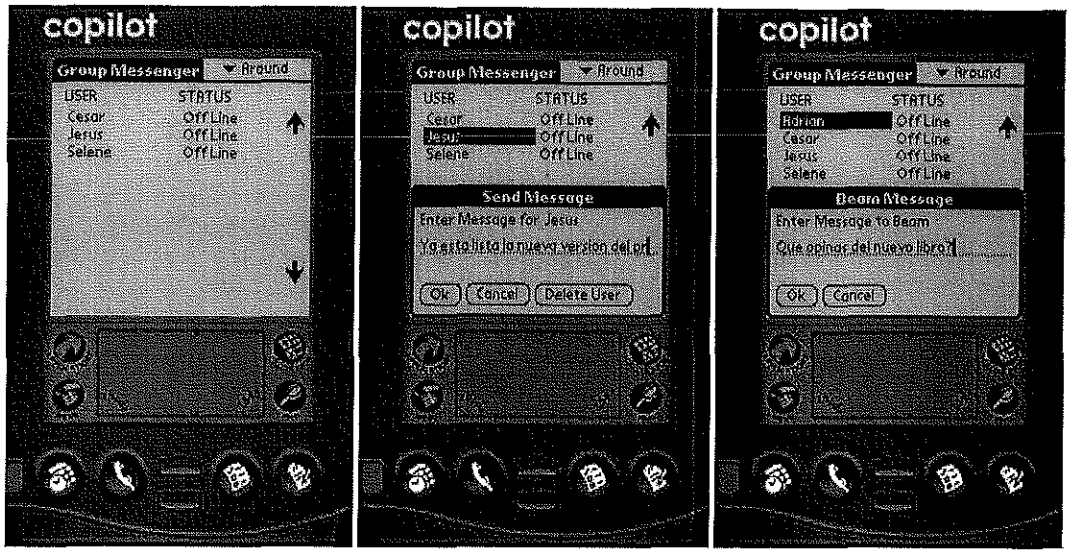


Figura 18: Pantallas de la aplicación móvil de Group Messenger. A la izquierda se muestra la pantalla principal, al centro el usuario se encuentra enviando un mensaje y a la derecha el usuario se dispone a transmitir un mensaje a través del dispositivo de comunicación infrarroja.

Desde la aplicación móvil también se pueden enviar mensajes y datos de contactos por medio del dispositivo de comunicación infrarroja con que cuentan algunos PDAs Palm (ver Figura 18). Además, la aplicación también esta preparada para recibir tales datos por la misma vía, la transmisión y recepción por este método se realiza utilizando la biblioteca Palm COMAL.

En este capítulo se presentó la aplicación Group Messenger, un SOPE desarrollado con el fin de probar el entorno de desarrollo COMAL, se describió su estructura, cada uno de sus componentes, su funcionalidad y la forma en que las bibliotecas COMAL fueron utilizadas para su construcción. En el siguiente capítulo se describen algunas de las pruebas realizadas a esta aplicación y a algunos componentes de las bibliotecas COMAL, y se presentan y analizan los resultados obtenidos.

Capítulo VI.

Pruebas y resultados.

En este capítulo se documentan las pruebas realizadas a las bibliotecas de desarrollo COMAL. La mayoría de las pruebas se realizaron utilizando Group Messenger, la aplicación que se construyó utilizando COMAL para el desarrollo de este trabajo de tesis, esto debido a que la aplicación utiliza muchas de las funciones básicas de las bibliotecas. Otras pruebas se realizaron con ayuda de un par de sencillos programas llamados `palm_connect` y `irobex_palm3`. `palm_connect` fue construido específicamente para desplegar bases de datos que residen en dispositivos Palm, mientras que `irobex_palm3` es un programa ejemplo que forma parte del software Open OBEX con el cual se pueden intercambiar datos entre una computadora personal que utiliza el sistema operativo Linux y un dispositivo Palm.

La prueba ideal de la plataforma COMAL sería proporcionar la herramienta a varios desarrolladores para que construyeran algunas aplicaciones colaborativas móviles, para que, una vez construidas las aplicaciones, se obtuvieran estadísticas en cuanto a que tan complejo les resultó el desarrollo utilizando la arquitectura, bibliotecas y documentación de COMAL, comparado con el esfuerzo que les ha tomado desarrollar aplicaciones similares, sin utilizar COMAL. Sin embargo, debido a que este es un ejercicio que requiere de mucho tiempo y planeación, no pudo realizarse esta ocasión.

Cada una de las bibliotecas fue probada de manera independiente, de la misma manera que las funciones de las bibliotecas. Este tipo de pruebas son llamadas pruebas por módulo o por unidad, y se definen como el proceso de probar los subprogramas, subrutinas o procedimientos de un programa en forma individual [Myers, 1979]. De esta manera, antes de probar un sistema como un todo, primero se realizan pruebas a bloques más pequeños del programa, lo que tiene tres grandes ventajas: se manejan más fácilmente las posibles respuestas del sistema, ya que se analiza sólo una pequeña porción de código a la vez, la tarea de depuración del sistema se facilita, debido a que cuando se encuentra un error se sabe que módulo es al que afecta y además, con las pruebas por módulo existe la posibilidad de probar secciones del sistema en forma paralela (probando dos o más módulos a la vez) [Myers, 1979]. El objetivo principal de este tipo de pruebas es constatar que cada uno de los módulos del sistema se comporta de la manera en que fue establecido en sus especificaciones.

De acuerdo a [Stroustrup, 1997], un sistema debe ser diseñado y construido para que sea fácil de probar, incluso puede tener incluidos mecanismos que faciliten el desarrollo de las pruebas del sistema. Algunas veces no se realiza lo anterior por miedo a que los tiempos de ejecución de la aplicación se hagan innecesariamente largos o que se provoque que los archivos y estructuras de datos del sistema ocupen demasiado espacio de almacenamiento. La mayoría de las veces tales miedos son infundados ya que el código para realizar las pruebas puede ser removido del sistema antes de generar la versión final [Stroustrup, 1997].

Las bibliotecas libCOMAL y libCOMALIr cuentan con código y algunas funciones para auxiliar en la depuración y prueba de las subrutinas. Estas líneas de código auxiliares pueden ser removidas y añadidas según sea necesario. Cuando la variable `DEBUG_TRACE` se encuentra definida en el archivo `Makefile` que controla la compilación

de las bibliotecas, el código se añade, y cuando dicha variable no está definida, el código es removido. El valor de `DEBUG_TRACE` es utilizado dentro de las subrutinas de las bibliotecas COMAL por medio de las instrucciones del preprocesador del lenguaje C `#ifdef` y `#endif`. Por ejemplo, en el siguiente segmento de código se muestra como, en una de las rutinas donde libCOMAL manipula una estructura que contiene información acerca de un evento, la variable `DEBUG_TRACE` decide si se muestra al usuario la información del evento, una vez que ha sido modificada.

```
#ifdef DEBUG_TRACE
    g_print_event_info( *event );
#endif
```

La función `g_print_event_info` simplemente imprime en pantalla los miembros y valores de una estructura tipo `EventInfo`, el tipo de dato de COMAL que contiene la información referente a un evento (`g_print_event_info` recibe como parámetro un apuntador a una estructura de este tipo). En las secciones siguientes se describen las pruebas realizadas a los módulos de cada una de las bibliotecas.

VI.1 Pruebas a libCOMAL.

Las pruebas realizadas a la biblioteca libCOMAL se dividieron en dos partes:

- Pruebas realizadas a las llamadas para accesar, almacenar y eliminar los tipos de datos que maneja COMAL en los archivos con formato GDBM.
- Pruebas a las llamadas que sincronizan datos entre computadoras de escritorio y los dispositivos Palm.

Con el fin de probar los accesos a los archivos GDBM se utilizaron las funciones auxiliares `g_print_event_info`, `g_print_contact_info` y `g_print_message_info`. A continuación se describe como se utilizó `g_print_contact_info` para probar si libCOMAL accesaba, modificaba y almacenaba de forma correcta estructuras tipo `ContactInfo`, las cuales encapsulan los datos acerca de un contacto.

Para realizar esta prueba se utilizó Group Messenger. Primeramente, se compilaron la biblioteca y la aplicación (Group Messenger) con la variable `DEBUG_TRACE` activada. Una vez terminado el proceso de compilación, se ejecutó el programa `messenger_client` (es el nombre del programa ejecutable de Group Messenger), debido a que el programa fue compilado con `DEBUG_TRACE` activado, se despliegan en pantalla una serie de mensajes relacionados con la ejecución del programa y aparece la ventana principal (ver Figura 15). El siguiente paso fue seleccionar del menú de la aplicación Group Messenger la opción para editar la información personal del usuario actual (*Edit* → *Personal Info...*), lo que provoca que aparezca la ventana que se muestra en la Figura 19. Después de modificar algunos datos en la interfaz gráfica se presionó el botón *Save*, para que los cambios fueran registrados en el archivo GDBM correspondiente.

La rutina de libCOMAL que se encarga de almacenar la información del usuario en archivos GDBM contiene el siguiente segmento de código:

```
#ifdef DEBUG_TRACE
    g_print_contact_info (*contact );
#endif
```

Como Group Messenger y libCOMAL fueron compilados con la opción `DEBUG_TRACE` habilitada, en la pantalla se muestra la siguiente información:

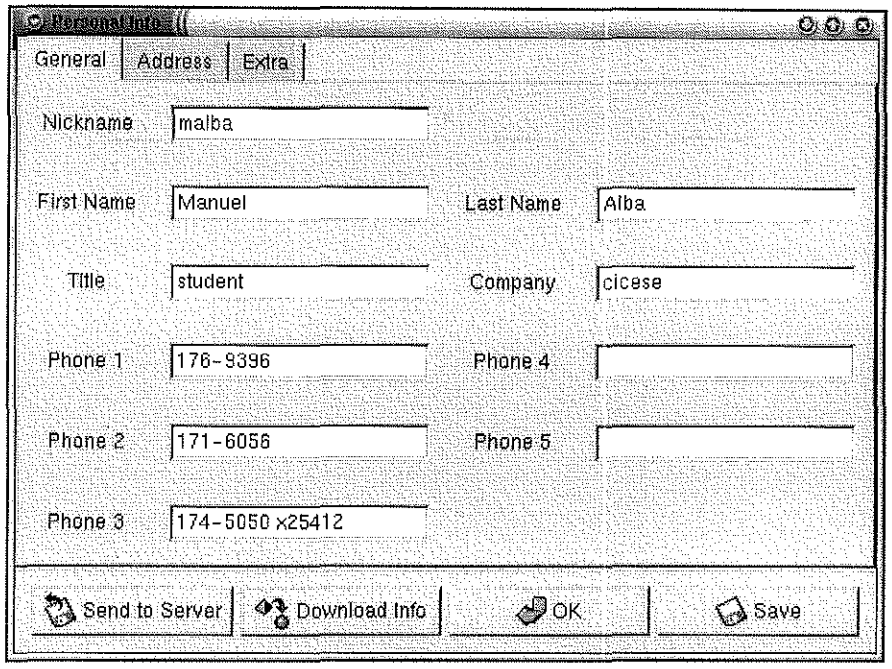


Figura 19: Ventana de edición de información personal de un usuario de Group Messenger.

```
USER_ID[1]
NICKNAME[malba]
FIRST_NAME[Manuel]
LAST_NAME[Alba]
TITLE[student]
COMPANY[cicese]
PHONE1[176-9396]
PHONE2[171-6056]
PHONE3[174-5050 x25412]
ADDRESS[Clemente Rojo 2666, Colonia]
CITY[Ensenada]
STATE[Baja California]
ZIPCODE[22850]
COUNTRY[Mexico]
NOTE[Esta es una nota, ahora una nota completa no crees?]
```

Además, la rutina de libCOMAL que se encarga de registrar de manera individual cada uno de los datos, en el archivo GDBM, contiene el siguiente segmento de código:

```
gdbm_store_data( GDBM_FILE gdf, gchar *key, gchar *value )
{
#ifdef DEBUG_TRACE
    g_print( "STORE[%s:%s]\n", key, value );
#endif
```

se presenta entonces en pantalla la información de cada uno de los pares llave/valor que son almacenados en el archivo GDBM:

```
STORE[1.user_id:1]
STORE[1.alias:malba]
STORE[1.first_name:Manuel]
STORE[1.last_name:Alba]
STORE[1.title:student]
STORE[1.company:cicese]
STORE[1.phone1:176-9396]
STORE[1.phone2:171-6056]
STORE[1.phone3:174-5050 x25412]
STORE[1.phone4:]
STORE[1.phone5:]
STORE[1.address:Clemente Rojo 2666, Colonia]
STORE[1.city:Ensenada]
STORE[1.state:Baja California]
STORE[1.zipcode:22850]
STORE[1.country:Mexico]
STORE[1.custom1:]
STORE[1.custom2:]
STORE[1.custom3:]
STORE[1.custom4:]
STORE[1.note:Esta es una nota, ahora una nota completa no crees?]
```

Como se puede observar, la información se almacena de manera correcta. Esta prueba se realizó varias veces, con diferentes tipos de datos, borrando campos, modificándolos y añadiendo nueva información; la biblioteca pasó las pruebas satisfactoriamente. Un procedimiento similar se realizó para verificar que los datos eran accesados debidamente por libCOMAL, ya que las rutinas de acceso también contienen código de

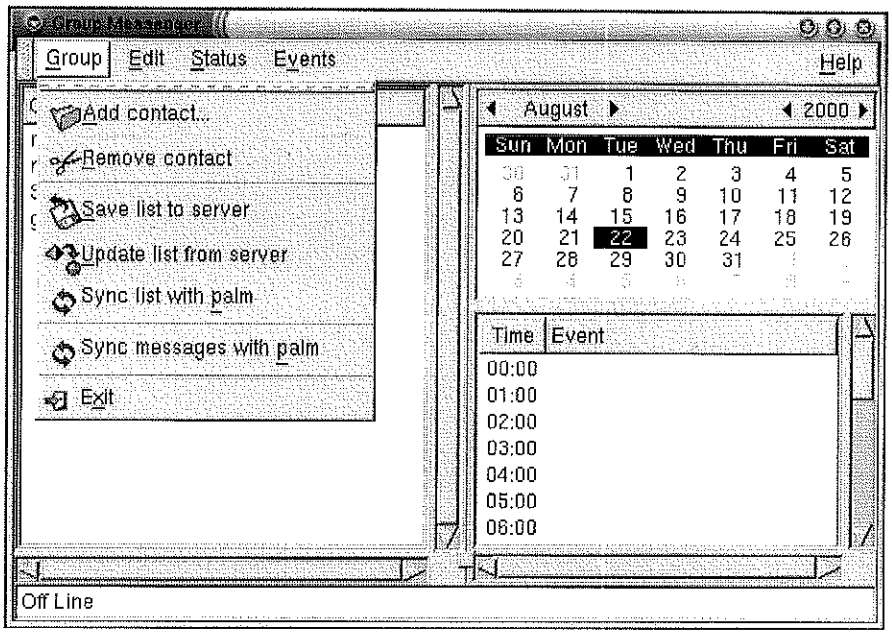


Figura 20: Uno de los submenús de Group Messenger.

depuración y prueba que puede ser utilizado con la variable `DEBUG_TRACE`. Las otras rutinas que manipulan el resto de los tipos de datos que libCOMAL soporta fueron probadas de manera similar.

Las funciones para la sincronización de datos de libCOMAL fueron probadas por medio de Group Messenger y utilizando `palm_connect`. Una vez almacenada la información en los archivos GDBM correspondientes, la información era sincronizada con el dispositivo Palm, utilizando la opción del menú *Group* → *Sync list with palm* (para sincronizar la lista e información de los contactos) o *Group* → *Sync messages with palm*, para sincronizar los mensajes entre la computadora de escritorio y el cliente móvil (ver Figura 20). Otra de las opciones de sincronización de información se encuentra bajo el submenú *Events*.

Después de haber realizar las sincronizaciones se realizaba una verificación de los datos que eran almacenados en el dispositivo Palm, primero visualizándolos con la

aplicación móvil y después utilizando el programa `palm_connect`. Por ejemplo después de realizar el procedimiento de sincronización de la lista de usuarios, `palm_connect` arrojó la siguiente información:

```
[malba@mercury desktop-palm]$ ./palm_connect
Press the 'sync' button on the pilot...Connected? -> 4
==================================
[ Record 0 ] : 30
gallo 1974-2000
==================================
Off Line
==================================
[ Record 1 ] : 30
Selene
==================================
Off Line
==================================
[ Record 2 ] : 30
rafilla
==================================
Off Line
==================================
[ Record 3 ] : 30
Adrian
==================================
Off Line
```

De esta forma se comprobó que las rutinas de libCOMAL para la sincronización de datos también funcionan de la manera deseada.

VI.2 Pruebas a libCOMALIr.

Las pruebas a libCOMALIr se restringieron únicamente a la validación de las rutinas para enviar y recibir los tipos de datos definidos en el entorno COMAL. Debido a que solamente es posible enviar y recibir una sola unidad de información por cada

conexión que se realiza, las pruebas fueron realmente simples.

Los datos individuales eran enviados desde la computadora de escritorio y posteriormente recibidos de manera adecuada en el cliente móvil. La verificación de los datos en el dispositivo Palm se realizó únicamente por medio de la información que despliega el cliente móvil de Group Messenger.

VI.3 Pruebas a Palm COMAL.

Esta biblioteca fue probada por medio del cliente móvil de la aplicación Group Messenger y el programa `palm_connect` mencionado anteriormente. En este caso no se incluyó dentro de la biblioteca algún segmento de código que pudiera ser utilizado para depurar o probar las rutinas contenidas, debido a que en los dispositivos Palm no es tan fácil contar con una pantalla donde desplegar mensajes de las aplicaciones, como en el caso de las aplicaciones de escritorio. Así que las pruebas consistieron simplemente en utilizar el cliente móvil de Group Messenger con diferentes tipos de datos, comprobando siempre si eran enviados y recibidos de manera correcta.

La prueba de recepción del envío de datos con Palm COMAL se realizó por medio del programa `irobex_palm3.c` elaborado por Pontus Fuchs y Dag Brattli. A continuación se muestra el resultado de la ejecución del programa y la recepción de un mensaje enviado desde el cliente móvil, a través de Palm COMAL.

```
[malba@mercury apps]$ ./irobex_palm3
Send and receive files to Palm3
Waiting for files
OBEX_ServerRegister()
irobex_listen()
obex_create_socket()
```

```
obex_register_async()
irobex_listen(), transport mtu=509
obex_input_handler() Got some input!
obex_server() Got CMD_CONNECT
irobex_palm3: wrote /tmp/message.msg (290 bytes)
BEX_TransportDisconnect()
irobex_disconnect_request()
obex_delete_socket()
[malba@mercury apps]$
```

En este capítulo se han mostrado las pruebas que se realizaron a las bibliotecas que forman parte de la plataforma de desarrollo COMAL. Las pruebas se realizaron de manera modular y los resultados que se obtuvieron fueron satisfactorios, de acuerdo a lo especificado en los requerimientos de COMAL. En el capítulo siguiente se presentan las conclusiones obtenidas con la realización de este trabajo, además de que se presentan algunas ideas y trabajo futuro que podrían incorporase a COMAL.

Capítulo VII.

Conclusiones.

En este capítulo se presentan las conclusiones resultado de este trabajo, así como algunas actividades que pueden realizarse en el futuro, tomando como base las herramientas que se desarrollaron durante la elaboración de esta tesis.

Primeramente, el objetivo propuesto inicialmente, de diseñar y construir una herramienta para el desarrollo de aplicaciones colaborativas móviles, se cumplió de manera satisfactoria. Debido a que la plataforma propuesta fue elaborada en base a las características y necesidades más comúnmente encontradas en aplicaciones del Handheld CSCW, COMAL puede ser utilizado para el desarrollo de una amplia variedad de aplicaciones de este tipo, siempre y cuando su estructura se ajuste a la arquitectura COMAL que sirve como guía para la utilización de las bibliotecas.

Además, Group Messenger, la aplicación colaborativa móvil que se construyó utilizando COMAL, sirvió como prueba de que este entorno de desarrollo puede utilizarse como herramienta para construir otras aplicaciones. Gracias a esta aplicación también fué posible explorar la forma en que COMAL puede ser extendido para manejar nuevos tipos de datos.

Con el planteamiento de COMAL y el desarrollo de Group Messenger utilizando la plataforma de desarrollo, se pudo comprobar además que los Asistentes Digitales Personales pueden ser utilizados exitosamente en aplicaciones colaborativas, y que, de hecho, sus características de portabilidad y convivencia permanente con el usuario los destina a ocupar un lugar muy importante dentro del dominio de las aplicaciones colaborativas.

También se concluye de este trabajo que, a pesar de la popularidad actual de los PDAs, es realmente poco el trabajo que se ha realizado con el objetivo de integrarlos en aplicaciones realmente colaborativas, y además de la necesidad existente de este tipo de aplicaciones, la demanda de herramientas para construir tales aplicaciones es mucho *mayor*.

El desarrollo del cliente móvil de Group Messenger en el dispositivo Palm se realizó sin más contratiempos que los relacionados con el cambio de plataforma de cómputo. Las bibliotecas del PalmOS resultaron fáciles de utilizar y muy intuitivas, gracias a esto se pudo constatar la razón por la cual la Plataforma de Cómputo Palm es la más popular de entre todas las disponibles en PDAs.

Otra de las conclusiones a las que se llegó con este trabajo es la conveniencia de utilizar un protocolo de comunicación inalámbrica como IrOBEX. La implementación de IrOBEX con el sistema operativo Linux, el Open OBEX, es fácil de integrar en aplicaciones escritas en el lenguaje C, y muy cómodo para los usuarios de dichas aplicaciones, ya que los libra de tener que lidiar con conexiones alámbricas, fomentando la movilidad y las interacciones usuario a usuario, por medio de los PDAs.

Las pruebas realizadas a los componentes de las bibliotecas COMAL descritas en el capítulo anterior demuestran que las rutinas funcionan de manera correcta en los

escenarios en los que han sido utilizados hasta el momento. Sin embargo, es todavía necesario realizarles más pruebas, de preferencia, mientras se desarrollan otras aplicaciones colaborativas móviles, éstos y otros aspectos relacionados con la evolución futura de COMAL se describen a continuación.

VII.1 Trabajo futuro.

Aún cuando COMAL actualmente maneja algunos de los tipos de datos comúnmente encontrados en las aplicaciones colaborativas tradicionales, existen otros tipos que pudieran ser incorporados. Ejemplos de estas nuevas estructuras son documentos HTML ó XML sencillos, imágenes pequeñas (que pudieran ser desplegadas en un PDA) u otros tipos de datos específicos a la aplicación que se desarrolla. Entre más tipos de datos soporte el entorno COMAL, mayor será la gama de aplicaciones que se pueden construir con él.

El enfoque actual de la implementación de COMAL esta orientado a la comunicación entre un grupo de personas, al libre intercambio de información, sin embargo, existen situaciones en las cuales la seguridad en la transferencia de información se vuelve una prioridad. En el estado en el que se encuentra COMAL el nivel de seguridad es dictado únicamente por la aplicación (en Group Messenger se utilizan contraseñas para proteger la información de los usuarios), sería entonces interesante que COMAL proporcionara mecanismos de seguridad dentro de las bibliotecas, que pudieran ser utilizados de manera transparente a través de las aplicaciones construidas con COMAL.

Un ejercicio interesante con COMAL consistiría en desarrollar otras aplicaciones colaborativas móviles utilizando la herramienta. Esto permitiría conocer los verdaderos alcances de sus bibliotecas además de poder establecer criterios de comparación con

otras herramientas Handheld CSCW.

Un área más que se podría explorar es la de la integración de COMAL con otras plataformas handheld, como WindowsCE, y dejar de lado la restricción de utilizar COMAL únicamente con dispositivos Palm.

Bibliografía.

[AvantGo, 2000] AvantGo Inc. 2000. *AvantGo Developer.* `http://www.avantgo.com/`.

[Bachmann, 1999] Bachmann, Glenn. 2000. *Palm Programming.* SAMS Publishing. Primera edición. Carmel, Indiana.

[Dittrich y Thomas, 1998] Dittrich, Jurgen y Becker Thomas. 1998. *Middleware for Development of CSCW Handheld Applications.* Handheld CSCW Workshop, ACM Conference on CSCW, CSCW '98, Seattle, Washington.

[Ellis *et al.*, 1991] Ellis, C. A., Gibbs, S. J. y G. L. Rein. 1991. *Groupware: Some Issues and Experiences.* Communications of the ACM. 34 (1): 39-58 p.

[Fowler y Scott, 1997] Fowler, Martin y Kendall Scott. 1997. *UML Distilled.* Addison-Wesley.

[Gellersen, 1998] Gellersen, Hans-W. 1998. *Handheld CSCW.* `http://www.teco.edu/hcscw/workshop.html`.

[Greenberg *et al.*, 1999] Greenberg, Saul, Boyle, Michael y Jason Laberge. 1999. *PDAs and Shared Public Displays: Making Personal Information Public, and Public Information Personal.* Personal Technologies. Aceptado en marzo, 1999.

[Heuser, 1999] Heuser, Werner. 1999. Linux IR HOWTO. 65 pp.

[HUC, 1999] 1999. *International Symposium on Handheld and Ubiquitous Computing HUC '99.* `http://www.teco.edu/huc/`.

[IDC, 1999] International Data Corporation. 1999. *Worldwide Smart Handheld Device Forecast Bolstered by Personal Companion Dominance.* `http://www.idc.com`.

[Kortuem *et al.*, 1999] Kortuem, Gerd, Bauer, Martin y Segall Zary. 1999. *NETMAN: The design of a collaborative wearable computer system.* Mobile Networks and Applications. (4): 49-58 p.

[Lehikoinen *et al.*, 1999] Lehikoinen, Juha, Holopainen, Jussi, Salmimaa, Marja y Angelo Aldrovandi. 1999. *MEX: A Distributed Software Architecture for Wearable Computers.* 3rd. International Symposium on Wearable Computers. San Francisco, California.

[Malone y Rockart, 1991] Malone, Thomas W. y John F. Rockart. 1991. *Computers, Network and the Corporation.* Scientific American.

[Mann, 2000] Mann, Steve. 2000. *A GNU/Linux Wristwatch Videophone.* Linux journal. (75): 86-91 p.

[McPherson, 2000] McPherson, Frank. 2000. *What is ActiveSync?.* `http://www.fmcpherson.com/knowce/what/what2.htm`.

[Microsoft, 2000] Microsoft Corp. 2000. *WindowsCE - Overview.* `http://www.microsoft.com/catalog/display.asp?site=120\&subid=22\&pg=1`.

[Myers, 1979] Myers, Glenford J. 1978. *The Art of Software Testing.* John Wiley & Sons Inc. Primera edición. New York. 177 pp.

[Myers *et al.*, 1998] Myers, Brad A., Stiel, Herb y Robert Gargiulo. 1998. *Collaboration Using Multiple PDAs Connected to a PC.* ACM Conference on CSCW, CSCW '98, Seattle, Washington.

[Palm, 2000] Palm Inc. 2000. *HotSync Technology.* `http://www.palmpilot.3com.com/newspromo/corporate/hotsync.html`.

[Roseman y Greenberg, 1996] Roseman, M. y S. Greenberg. 1996. *Building Real Time Groupware with GroupKit, A Groupware Toolkit.* ACM Transactions on Computer Human Interaction. 3 (1): 66-106 p.

[Rhodes y Mckeehan, 1999] Rhodes, Neil y Julie McKeehan. 1999. *Palm Programming, The Developer's Guide.* O'Reilly & Associates. Sebastopol, California.

[Shaw y Garlan, 1996] Shaw, Mary y David Garlan. 1996. *Software Architecture: Perspectives on an Emerging Discipline.* Prentice Hall.

[Soriano y Favela, 1997] Soriano, Ma. Teresa y Jesús Favela. 1997. *Trabajo Cooperativo Asistido por Computadora.* Reporte Técnico, CICESE.

[Stroustrup, 1997] Stroustrup, Bjarne. 1997. *The C++ Programming Language.* Addison Wesley. Tercera edición. Reading, Massachusetts. 910 pp.

[Sun, 2000] Sun Microsystems Inc. 2000. *Jini(tm) Connection Technology.* `http://www.sun.com/jini/`.

[UML-RTF, 1999] Object Management Group Inc.. 1999. *OMG Unified Modeling Language Specification.* Framingham, Massachusetts.

[Valley *et al.*, 1992] Valley, John J., Kochan, Stephen J. y Patrick H. Wood. 1992. *C Programming for UNIX* SAMS Publishing. Primera edición. Carmel, Indiana. 644 pp.

[WAP Forum, 1998] WAP Forum Ltd. 1998. *Wireless Application Protocol Arquitecture Specification.*

Apéndice A

Algunas características de las comunicaciones infrarrojas.

A.1 Introducción.

Las comunicaciones inalámbricas constituyen el área de la industria de la comunicación que crece de manera más acelerada en la actualidad. En 1997 los usuarios de sistemas de comunicación celular eran cerca de 200 millones, a los cuales se suman un promedio de 150,000 usuarios cada día. La demanda de servicios básicos de transferencia de voz y datos empuja los límites de la tecnología inalámbrica al tratarse de satisfacer la demanda del mercado.

Además de la telefonía celular, existen otras tecnologías a través de las cuales se puede realizar transferencia de información de manera inalámbrica. Son utilizadas por ejemplo, las frecuencias de radio y las comunicaciones infrarrojas.

A.2 Comunicaciones infrarrojas.

Las ondas de tipo infrarrojo son frecuentemente utilizadas para establecer canales de comunicación de corto alcance. Los controles remotos de televisiones, videocaseteras

y estéreos utilizan este tipo de tecnología. Las comunicaciones de este tipo son relativamente direccionales, fáciles de construir y baratas, pero tienen la desventaja de no pasar a través de objetos sólidos. Esta última desventaja, sin embargo, también tiene sus ventajas, ya que un sistema infrarrojo en una habitación o edificio no interfiere a algún sistema similar ubicado en algún lugar adyacente. Además, la seguridad de éste tipo de comunicación contra recepciones no deseadas es mejor que la de tecnologías de comunicación que utilizan ondas de radio. Esto trae como consecuencia que no sea necesaria una licencia emitida por el gobierno para operar un equipo de este tipo.

Estas propiedades han convertido a los dispositivos infrarrojos en muy buenos candidatos a utilizar en redes locales inalámbricas en interiores. Por ejemplo, si las computadoras y oficinas de un edificio se encuentran equipadas con emisores y receptores infrarrojos, se pueden incorporar a la red local computadores portátiles sin necesidad de conectarlas físicamente a algún cable o dispositivo especial. De esta forma, si se organiza una reunión, los participantes simplemente se presentan con sus computadoras portátiles en la habitación, pudiéndose interconectar totalmente sin la necesidad de realizar conexiones.

Las comunicaciones infrarrojas no pueden ser utilizadas en exteriores ya que el sol brilla con la misma intensidad en la franja infrarroja que en la franja de luz visible del espectro electromagnético.

Actualmente se utilizan ampliamente las comunicaciones infrarrojas alrededor del mundo y una organización que mantiene un conjunto de protocolos que utilizan esta tecnología es IrDA.

A.3 IrDA, Infrared Data Association.

IrDA es una organización no lucrativa fundada en 1993 con la finalidad de crear y promover estándares de interconexión punto a punto, interoperables y de bajo costo. Los estándares que IrDA produce son soportados por una amplia variedad de aparatos electrónicos y dispositivos de cómputo y comunicaciones. Esta asociación esta compuesta por alrededor de 120 compañías como 3Com, NEC, *M*icrosoft, Compaq y Toshiba. La IrDA especifica tres estándares para la comunicación infrarroja: *IrDA Data*, *IrDA Control* y un nuevo protocolo llamado *AIr*.

A.3.1 *IrDA Data.*

IrDA Data es el estándar de transmisión diseñado para la transferencia de datos sobre una distancia de 0 a 1 metro a velocidades de 115 kb/s a 4 Mb/s (la velocidad de transmisión de 16 Mb/s se encuentra aún en desarrollo). Esta tecnología se diseñó para proveer conectividad inalámbrica a dispositivos que usualmente utilizarían cables. El *IrDA Data* está diseñado para transmitir datos punto-a-punto con un ángulo de variación de 30°. Algunas de las características del *IrDA Data* son:

- Instalado en más de 150 millones de dispositivos alrededor del mundo.
- Soportado por una amplia variedad de plataformas de hardware y software.
- Diseñado como reemplazo de cables en conexiones punto-a-punto.
- No produce interferencias con otros dispositivos electrónicos.
- Permanece compatible con las nuevas versiones del estándar.
- Altas velocidades de transmisión.

El *IrDA Data* es utilizado frecuentenente para transmitir datos por computadoras personales, computadoras portátiles, asistentes digitales personales (PDAs), impresoras, teléfonos y pagers, modems, cámaras fotográficas, dispositivos de acceso a redes locales de datos, relojes de pulso, etc. Además de contar actualmente con 150 millones de dispositivos, que hacen uso del estándar, el *IrDA Data* crece un 40% cada año. El amplio uso y aceptación de esta especificación ha acelerado la adopción de los protocolos de *IrDA* por otras organizaciones. La adopción universal de éste estándar facilita además la interoperación de hardware y software que transmite y/o recibe datos utilizando esta tecnología.

IrDA Data está compuesto por un conjunto de protocolos obligatorio y un conjunto de protocolos opcionales. Los protocolos obligatorios son: PHY (capa de señalización física), IrLAP (protocolo de acceso al enlace) e IrLMP (protocolo de manejo de enlace). Los protocolos que forman parte de la especificación *IrDA Data* se muestran en la Figura 21.

Características de la capa física.

Las características de la capa física son:

- Alcance: operación continua desde el punto de contacto hasta cuando menos una distancia de 1 metro (típicamente se alcanzan los 2 metros).
- Comunicación bi-direccional.
- Transferencia de datos desde 9600 b/s hasta 4 Mb/s.
- Los paquetes de datos se protegen utilizando CRC-16 para transmisiones de hasta 1.152 Mb/s y CRC-32 a 4 Mb/s.

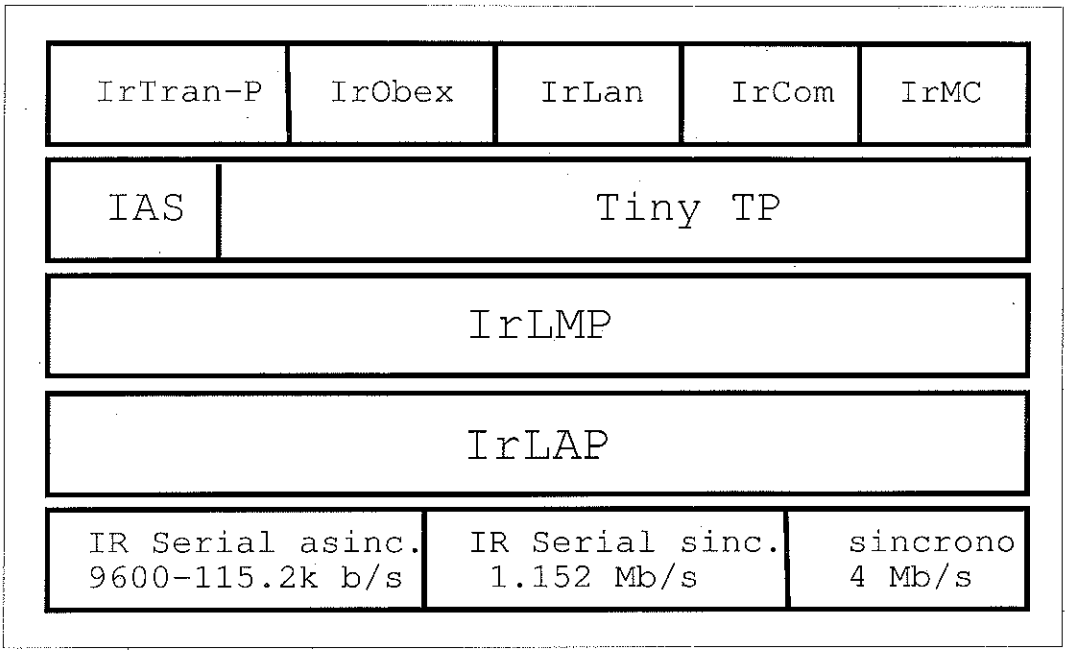


Figura 21: Los protocolos de *Irda Data*.

Características del IrLAP.

Las características más significativas del IrLAP son:

- Provee una conexión dispositivo-a-dispositivo para la transmisión de datos ordenada y confiable.
- Contiene procedimientos para descubrir dispositivos al alcance.
- Maneja nodos ocultos.

Características del IrLMP.

Las características del IrLMP son:

- Provee multiplexado de la capa IrLAP, el manejo de múltiples canales sobre sólo una conexión IrLAP.

- Provee procedimientos para descubrir protocolos y servicios a través del Information Access Service (IAS).

Protocolos opcionales.

Los protocolos opcionales que forman parte de la especificación *IrDA Data* son:

Tiny TP – Provee un control de flujo sobre las conexiones IrLMP con un servicio de segmentación y reensamblado opcional.

IrCOMM – Emula puertos COM (serie y paralelos) para aplicaciones anteriormente desarrollados para este tipo de puertos y para modems y dispositivos de impresión.

IrOBEX – Permite el intercambio de objetos de manera similar al HTTP.

IrDA Lite – Provee métodos para reducir el tamaño del código IrDA, manteniendo total compatibilidad con implementaciones completas.

IrTran-P – Es un protocolo para el intercambio de imágenes utilizado en dispositivos y cámaras para capturar imágenes digitales.

IrMC – Especificaciones para el intercambio de información entre teléfonos y dispositivos de comunicación, esto incluye entradas de calendario, mensajes y datos personales.

IrLan – Describe un protocolo utilizado para accesar una red local de datos usando una conexión inalámbrica infrarroja.

Seguridad.

La naturaleza direccional de *IrDA Data* proporciona cierto nivel de seguridad debido a que se requiere que el transmisor y el receptor se encuentren a la vista, uno respecto del otro. Sin embargo, es posible intervenir una transmisión detectando la luz reflejada y eliminando el ruido ambiental de la señal. *IrDA Data* no provee mecanismos de seguridad en el nivel de transmisión de datos, la seguridad la proporcionan los protocolos de más alto nivel en la arquitectura y las aplicaciones que utilizan *IrDA Data* en caso de que se requiera restricción de acceso o encriptamiento especial.

A.3.2 *IrDA Control* y *AIr.*

IRDA Control es el estándar para comandos bi-direccionales y control diseñado para la utilización inalámbrica de dispositivos de entrada de datos tales como: teclados, ratones, joysticks, unidades de control remoto y asistentes digitales personales. Fué diseãdo para proveer enlaces punto-a-punto en los que la velocidad de respuesta es crítica.

AIr es un estándar en desarrollo para proveer conectividad multi-punto-a-multi-punto dentro de un cuarto. El alcance y velocidad de transmisión son variables y van desde 250 kb/s a una distancia de 8 metros a 4 Mb/s con una distancia de 4 m. Este estándar esta diseñado para conexiones inalámbricas a múltiples dispositivos periféricos y para aplicaciones de salas de reuniones y trabajo colaborativo.

Apéndice B

Documentación del servidor Group Messenger.

A continuación se presentan los comandos que los clientes mandan al servidor Group Messenger para realizar alguna operación. La sintaxis de los comandos es:

"comando" arg1 arg2 ... argN

Donde *"comando"* es la acción a realizar y *arg1... argN* son sus argumentos. [*arg*] significa que el argumento *arg* es opcional. El servidor envía respuestas a los clientes que hicieron una petición, solamente al utilizar ciertos comandos y en ciertas situaciones. Estas respuestas contienen el resultado obtenido al ejecutar el comando que los clientes solicitan.

COMANDOS que pueden ser interpretados por el servidor Group Messenger.

`login user_id [user_passwd]`

Permite al usuario `user_id` accesar al servidor, utilizando la contraseña `user_passwd`. Este comando se debe ejecutar antes que cualquier otro, con el fin de que el usuario pueda accesar su información, así como contactar a otros usuarios. Cuando el usuario no tiene ninguna contraseña definida `user_passwd` debe ser omitido.

Respuesta del servidor:

`LOGIN_OK`	El usuario ha accesado el sevidor.
`NO_SUCH_CONTACT`	Usuario `user_id` inexistente.
`INVALID_PASSWD`	La contraseña no es correcta.

`close`

Termina la conexión actual.

Respuesta del servidor:

Mensaje de despedida.

`contacts`

Regresa el contenido de la lista de contactos del usuario actualmente dentro del sistema.

Respuesta del servidor:

`INVALID_CONNECTION`	El usuario no ha sido validado.

`LIST_BEGIN`

`contacto1`

`contacto2`

...

contactoN

LIST_END — La lista de contactos del usuario actual, la lista empieza con LIST_BEGIN y termina con LIST_END.

`add_contact user_id`

Añadir al usuario user_id a la lista de contactos del usuario actual.

Respuesta del servidor:

INVALID_CONNECTION — El usuario no ha sido validado.

NO_SUCH_CONTACT — El usuario user_id no existe.

ADDED_OK — El contacto fue añadido exitosamente.

`remove_contact user_id`

Remueve el usuario user_id de la lista de contactos del usuario actual.

Respuesta del servidor:

INVALID_CONNECTION — El usuario no ha sido validado.

NO_SUCH_CONTACT — El usuario user_id no existe.

REMOVED_OK — El contacto fue removido exitosamente.

`send_message user_id "message"`

A través de este comando, el usuario actual puede mandar un mensaje de texto al usuario `user_id`. Se considera como parte del mensaje a todo el texto que se encuentra después del destinatario (`user_id`).

Respuesta del servidor:

Ninguna.

`get_status [user_id]`

Regresa el status del usuario representado por `user_id`. Si el argumento `user_id` no es proporcionado, el servidor regresa el status del usuario actual. Los valores de status válidos son:

`OFF_LINE`

`ON_LINE`

`BUSY`

`AWAY`

`AROUND`

Otras respuestas del servidor:

`INVALID_CONNECTION`	El usuario no ha sido validado.
`NO_SUCH_CONTACT`	El usuario `user_id` no existe.

`set_status valor_status`

Establece el status del usuario actual al valor `valor_status`. Los valores válidos para `valor_status` son los mismos que los descritos en la descripción del comando `get_status`.

Respuesta del servidor:

`INVALID_CONNECTION`	El usuario no ha sido validado.
`NO_SUCH_STATUS`	El valor `valor_status` es desconocido.
`SET_OK`	El nuevo status fue selecionado

`get_personal_info`

Este comando despliega la información personal del usuario actual.

Respuesta del servidor:

`PERSONAL_INFO_BEGIN`

ificador del usuario (unico en el sistema).

`ALIAS`: Alias del usuario.

`PASSWD`: Clave del usuario.

`FIRST_NAME`: Primer nombre.

`LAST_NAME`: Apellido.

`TITLE`: Cargo.

`COMPANY`: Empresa a la que el usuario se encuentra relacionado.

`PHONE1`: Datos.

`PHONE2`: Datos.

`PHONE3`: Datos.

`PHONE4`: Datos.

`PHONE5`: Datos.

ADDRESS: Datos.

CITY: Ciudad.

STATE: Estado.

ZIPCODE: Zona postal.

COUNTRY: Nombre.

CUSTOM1: Datos extra.

CUSTOM2: Datos extra.

CUSTOM3: Datos extra.

CUSTOM4: Datos extra.

NOTE: Nota de texto acerca del usuario.

PERSONAL_INFO_END

remove_all_contacts

Remueve todos los elementos de la lista de contactos del usuario.

Respuesta del servidor:

Ninguna.

add_contact user_id

Agrega un elemento a la lista de contactos del usuario actual.

Respuesta del servidor:

NO_SUCH_CONTACT	El usuario user_id no existe.
ADDED_OK	El contacto fue incorporado sin problemas.

`get_contact_info user_id`

Despliega la información personal del usuario `user_id`.

Respuesta del servidor:

`CONTACT_INFO_BEGIN`

ificador del usuario (unico en el sistema).

`ALIAS`: Alias del usuario.

`FIRST_NAME`: Primer nombre.

`LAST_NAME`: Apellido.

`TITLE`: Cargo.

`COMPANY`: Empresa a la que el usuario se encuentra relacionado.

`PHONE1`: Datos.

`PHONE2`: Datos.

`PHONE3`: Datos.

`PHONE4`: Datos.

`PHONE5`: Datos.

`ADDRESS`: Datos.

`CITY`: Ciudad.

`STATE`: Estado.

`ZIPCODE`: Zona postal.

`COUNTRY`: Nombre.

`CUSTOM1`: Datos extra.

`CUSTOM2`: Datos extra.

`CUSTOM3`: Datos extra.

`CUSTOM4`: Datos extra.

NOTE: Nota de texto acerca del usuario.

CONTACT_INFO_END

`get_messages`

Esta función muestra al usuario actual los mensajes que le han sido enviados por otras personas utilizando Group Messenger.

Respuesta del servidor:

MESSAGES_BEGIN

`contacto1 mensaje1`	Mensaje enviado por `contacto1` con contenido `mensaje1`.
`contacto2 mensaje2`	Mensaje enviado por `contacto2` con contenido `mensaje2`.
...	
`contactoN mensajeN`	Mensaje enviado por `contactoN` con contenido `mensajeN`.

MESSAGES_END

`set_alias [alias]`

Cambia el valor del alias del usuario, si `alias` es omitido, el alias actual es eliminado.

Respuesta del servidor:

Ninguna.

`set_passwd [passwd]`

Cambia el valor de la contraseña del usuario, si `passwd` es omitido, la contraseña actual es eliminada.

Respuesta del servidor:

Ninguna.

`set_first_name [nombre]`

Cambia el valor del nombre del usuario, si `nombre` es omitido, el nombre actual es eliminado.

Respuesta del servidor:

Ninguna.

`set_last_name [apellido]`

Cambia el valor del apellido del usuario, si `apellido` es omitido, el apellido actual es eliminado.

Respuesta del servidor:

Ninguna.

`set_title [título]`

Cambia el valor del título del usuario, si `título` es omitido, el valor de título actual es eliminado.

Respuesta del servidor:

Ninguna.

`set_company [compañía]`

Cambia el valor del nombre de la compañía con el cual el usuario actual se encuentra asociado. Si el valor de `compañía` es omitido, el valor actual de compañía es eliminado.

Respuesta del servidor:

Ninguna.

`set_phone1 [número]`

Cambia el valor del número almacenado en el registro PHONE1 del usuario actual. Si `número` es omitido, el valor actualmente almacenado en PHONE1 es eliminado.

Respuesta del servidor:

Ninguna.

`set_phone2 [número]`

Cambia el valor del número almacenado en el registro PHONE2 del usuario actual. Si `número` es omitido, el valor actualmente almacenado en PHONE2 es eliminado.

Respuesta del servidor:

Ninguna.

`set_phone3 [número]`

Cambia el valor del número almacenado en el registro PHONE3 del usuario actual. Si `número` es omitido, el valor actualmente almacenado en PHONE3 es eliminado.

Respuesta del servidor:

Ninguna.

`set_phone4 [número]`

Cambia el valor del número almacenado en el registro PHONE4 del usuario actual. Si `número` es omitido, el valor actualmente almacenado en PHONE4 es eliminado.

Respuesta del servidor:

Ninguna.

`set_phone5 [número]`

Cambia el valor del número almacenado en el registro PHONE5 del usuario actual. Si `número` es omitido, el valor actualmente almacenado en PHONE5 es eliminado.

Respuesta del servidor:

Ninguna.

`set_address [dirección]`

Cambia el valor de la dirección del usuario actual, si `dirección` es omitida, el valor actual de la dirección es eliminado.

Respuesta del servidor:

Ninguna.

`set_city [ciudad]`

Cambia el valor del nombre de la ciudad del usuario actual, si `ciudad` es omitida, el nombre de la ciudad actual es eliminado.

Respuesta del servidor:

Ninguna.

`set_state [estado]`

Cambia el valor del nombre del estado del usuario actual, si `estado` es omitido, el nombre del estado actual es almacenado.

Respuesta del servidor:

Ninguna.

`set_zip_code [zona]`

Cambia el valor de la zona postal del usuario actual a `zona`, si este valor es omitido, el valor actual de la zona postal es eliminado.

Respuesta del servidor:

Ninguna.

`set_country [país]`

Cambia el valor del nombre del país del usuario actual a `país`, si este valor es omitido, el nombre del país actual es eliminado.

Respuesta del servidor:

Ninguna.

`set_custom1 [datos]`

Cambia el valor de los datos almacenados en el registro CUSTOM1 del usuario actual. Si `datos` es omitido, el valor actualmente almacenado en CUSTOM1 es eliminado.

Respuesta del servidor:

Ninguna.

`set_custom2 [datos]`

Cambia el valor de los datos almacenados en el registro CUSTOM2 del usuario actual. Si `datos` es omitido, el valor actualmente almacenado en CUSTOM2 es eliminado.

Respuesta del servidor:

Ninguna.

`set_custom3 [datos]`

Cambia el valor de los datos almacenados en el registro CUSTOM3 del usuario actual. Si `datos` es omitido, el valor actualmente almacenado en CUSTOM3 es eliminado.

Respuesta del servidor:

Ninguna.

`set_custom4 [datos]`

Cambia el valor de los datos almacenados en el registro CUSTOM4 del usuario actual. Si `datos` es omitido, el valor actualmente almacenado en CUSTOM4 es eliminado.

Respuesta del servidor:

Ninguna.

`set_note [nota]`

Cambia la información de la nota del usuario actual a `nota`. Si este valor es omitido, el contenido actual de la nota del usuario es eliminado.

Respuesta del servidor:

Ninguna.

`ayt`

Comando para verificar que el servidor se encuentra en ejecución.

Respuesta del servidor:

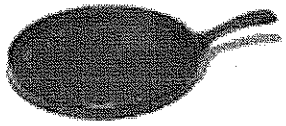
`YES`	En caso de que no existan problemas en la transferencia de datos entre el cliente y el servidor.

Apéndice C

Documentación de las bibliotecas COMAL.

La documentación de las bibliotecas COMAL se encuentra disponible en la dirección `http://dlib.cicese.mx/~malba/comal`. A continuación se incluye una copia impresa de la documentación de la forma en que se encontraba en noviembre de 2000.

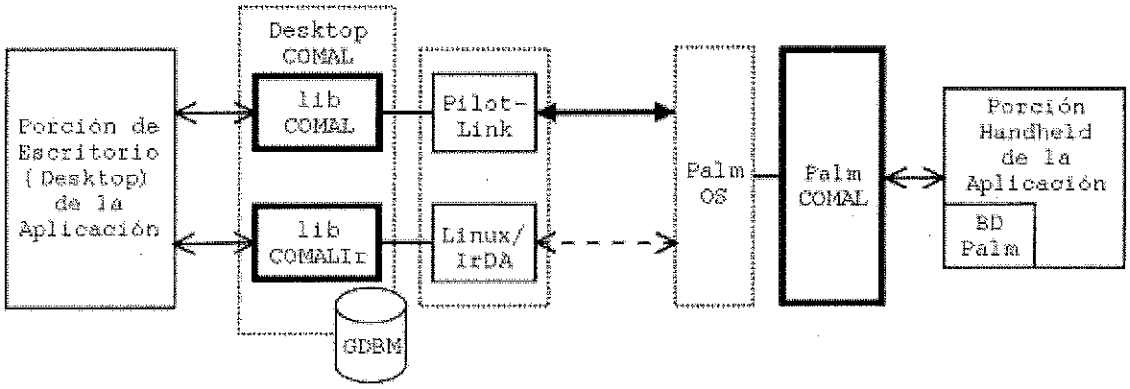
COMAL



El objetivo principal de las bibliotecas COMAL (*Collaborative Mobile Application Library*) es el de proveer una base sobre la cual se puedan construir aplicaciones colaborativas móviles o integrar dispositivos móviles a aplicaciones colaborativas de escritorio existentes. Estas bibliotecas proveen llamadas de funciones para transmitir, recibir y sincronizar datos con, y desde, dispositivos Palm.

Además de poder intercambiar información entre computadoras de escritorio y Palms utilizando el cable serial con el que regularmente se lleva a cabo la sincronización de información, COMAL tiene funciones para enviar y recibir datos a través del dispositivo de comunicación infraroja con que cuentan algunos modelos de computadoras Palm.

En el diagrama siguiente se ilustra la estructura de una aplicación que utiliza COMAL, así como su relación con las 3 bibliotecas que forman parte del entorno de desarrollo.



Bibliotecas COMAL:

libCOMAL - Funciones para intercambiar información entre computadoras de escritorio y dispositivos Palm utilizando el cable serial de sincronización.

libCOMALIr - Funciones para intercambiar información entre computadoras de escritorio y dispositivos Palm utilizando un dispositivo de comunicación infraroja (conectado a la PC) y el puerto infrarojo de los dispositivos Palm.

Palm COMAL - Llamadas de funciones para accesar los elementos de algunas bases de datos de los dispositivos Palm.

CICESE malba@cicese.mx

Noviembre 9, 2000.

libCOMAL

Bibliotecas para transferir datos entre un dispositivo Palm y computadoras de escritorio utilizando las bibliotecas de comunicación pilot-link.

comal_connect

comal_util

comal_event_db

comal_contact_db

comal_sync_event_db

comal_sync_contact_db

comal_sync_messages_db

CICESE malba@cicese.mx

Noviembre 9, 2000.

libCOMAL : connect

---

comal_connect_open_pilot( )

```
gint      comal_connect_open_pilot                ( gchar *port );
```

Abre una conexión con un dispositivo Palm, regresando un descriptor a donde se le puede leer y escribir.

port : contiene la ruta del archivo que apunta al dispositivo Palm e.g. "/dev/pilot"

Regresa : un descriptor abierto para leer/escribir desde el dispositivo Palm ó -1 si un error ocurrió

---

comal_connect_close_pilot( )

```
gint      comal_connect_close_pilot               ( int pilot );
```

Cierra una conexión con un dispositivo Palm previamente abierto.

pilot : el descriptor abierto de un dispositivo Palm

Regresa : 0

---

comal_connect_open_pilot_db( )

```
gint      comal_connect_open_pilot_db             ( int pilot,
                                                    int mode,
                                                    gchar *name,
                                                    int *handle );
```

Abre una base de datos de un dispositivo Palm para leer y escribir datos a ella, regresando una referencia a la base de datos en el parámetro 'handle'.

pilot : el descriptor abierto de un dispositivo Palm

mode : describe el modo de acceso:

Lectura = 0x80

Escritura = 0x40
Exclusivo = 0x20
ShowSecret = 0x10

name : el nombre de la base de datos por abrir, e.g. "contactsDB"

handle : usado para almacenar la referencia de la base de datos abierta

Regresa : comparar con el valor de retorno de la función dlp_OpenDB de la biblioteca pilot-link

---

comal_connect_close_pilot_db( )

```
gint     comal_connect_close_pilot_db                ( int pilot,
                                                       int handle );
```

Cierra una base de datos previamente abierta.

pilot : el descriptor abierto de un dispositivo Palm

handle : referencia de la base de datos abierta

Regresa : comparar con el valor de retorno de la función dlp_CloseDB de la biblioteca pilot-link

libCOMAL : util

---

get_list_size( )

```
guint        get_list_size              ( GDBM_FILE gdf );
```

Regresa el valor asociado con la llave "list_size" almacenado en un archivo GDBM, este valor debe ser un entero.

gdf : un archivo tipo GDBM abierto previamente
Regresa : representación entera del valor asociado con la llave "list_size"

---

set_list_size( )

```
guint        set_list_size              ( GDBM_FILE gdf,
                                          guint size );
```

Almacena un par llave/valor en un archivo GDBM, con llave "list_size" y con valor igual al entero que se pasa como parámetro a la función.

gdf : un archivo tipo GDBM abierto previamente
size : el entero que se asociará a la llave "list_size"
Regresa : 0

---

string_to_datum( )

```
datum        string_to_datum            ( gchar *key );
```

Convierte una cadena (gchar *) en un tipo de dato datum para utilizarlo en archivos GDBM.

key : La cadena que será convertida a tipo de dato datum
Regresa : el datum asociado a la cadena que se la pasó como parámetro a la función

---

datum_to_string( )

```
gint        datum_to_string                 ( datum data,
                                              gchar value[],
                                              int max_length );
```

Convierte un tipo de dato datum a una cadena (gchar *), regresando el resultado en el parámetro 'value'.

data : el datum a ser convertido en (gchar *)
value : buffer utilizado para almacenar la cadena resultado de la conversión
max_length : el límite del índice hasta donde puede realizarse la conversión, este valor debe coincidir con el tamaño del buffer
Regresa : la longitud de la cadena resultante de la conversión

---

gdbm_store_data( )

```
gint        gdbm_store_data                 ( GDBM_FILE gdf,
                                              gchar *key,
                                              gchar *value );
```

Almacena un par llave/valor en un archivo tipo GDBM.

gdf : un archivo tipo GDBM abierto previamente
key : la llave del par a almacenar
value : el valor asociado a la llave del par a almacenar
Regresa : el resultado de llamar a la función gdbm_store de las bibliotecas GDBM ó -1 si un error ocurrió

---

gdbm_fetch_data( )

```
gchar *         gdbm_fetch_data             ( GDBM_FILE gdf,
                                              gchar *key);
```

Obtiene el valor asociado a la llave que se pasa como parámetro a la función. El par llave/valor debe haber sido previamente almacenado en el archivo GDBM.

gdf : un archivo tipo GDBM abierto previamente
key : la llave del par del cual queremos conocer el valor
el valor asociado a la llave o NULL si la llave no está asociada con

Regresa : el valor asociado a la llave o NULL si la llave no esta asociada con ningún valor dentro del archivo GDBM

---

gdbm_delete_data( )

```
gint        gdbm_delete_data                ( GDBM_FILE gdf,
                                              gchar *key );
```

Elimina un par llave/valor de un archivo GDBM.

gdf : un archivo tipo GDBM abierto previamente
key : la llave del par que se desea eliminar del archivo GDBM
Regresa : el resultado de llamar a la función gdbm_delete de las bibliotecas GDBM

---

gdbm_key_exists( )

```
gboolean          gdbm_key_exists           ( GDBM_FILE gdf,
                                              gchar *key );
```

Verifica si la llave que se pasa como parámetro a la función ya se encuentra asociada con algún dato dentro del archivo GDBM.

gdf : un archivo tipo GDBM abierto previamente
key : la llave que se desea verificar
Regresa : TRUE si la llave existe o FALSE si la llave no se encuentra asociada a ningún valor dentro del archivo GDBM

comal_event_set_start_time( )

```
guint       comal_event_set_start_time                ( EventInfo *event,
                                                        gint year,
                                                        gint month,
                                                        gint day,
                                                        gint hour );
```

Asigna fecha y hora de inicio a un tipo de dato EventInfo (un evento).

event : el tipo de dato EventInfo al que se le asignará la fecha y hora de inicio
year : el año de la fecha que se le asignará al evento
month : mes del año que se le asignará al evento
day : día del mes de la fecha del evento
hour : hora del día de inicio del evento
Regresa : 0

---

comal_event_set_end_time( )

```
guint       comal_event_set_end_time                  ( EventInfo *event,
                                                        gint year,
                                                        gint month,
                                                        gint day,
                                                        gint hour );
```

event : el tipo de dato EventInfo al que se le asignará la fecha y hora de término
year : el año de la fecha que se le asignará al evento
month : mes del año que se le asignará al evento
day : día del mes de la fecha del evento
hour : hora del día de término del evento
Regresa : 0

---

comal_event_load_info_from_db( )

```
guint    comal_event_load_info_from_db              ( gchar *user_gdbm_datebook_path,
                                                      EventInfo *event,
                                                      gint year,
                                                      gint month,
                                                      gint day,
```

```
                                                gint hour );
```

Carga la información de los eventos que el usuario ha registrado para una hora y fecha en particular.

user_gdbm_datebook_path : ruta del directorio en donde se almacena los archivos con el calendario del usuario
event : estructura donde se guardará la información extraída de la base de datos del usuario
year : año de la fecha en que se desean conocer los eventos registrados
month : mes de la fecha en que se desean conocer los eventos registrados
day : día de la fecha en que se desean conocer los eventos registrados
hour : hora en que se desean conocer los eventos registrados por el usuario
Regresa : -1 si un error ocurrió, 1 si no existen eventos para la fecha y hora especificadas (event = NULL) y 0 si se encontró un evento y la información se almacenó en event

---

comal_event_check_path( )

```
guint     comal_event_check_path          ( gchar *user_gdbm_datebook_path,
                                            gchar *year,
                                            gchar *month );
```

Verifica que exista la jerarquía de directorios apropiada para almacenar los archivos de eventos para el año y mes especificados en los parámetros. Si la jerarquía no existe, esta función trata de crearla.

user_gdbm_datebook_path : ruta del directorio en donde se almacena los archivos con el calendario del usuario
year : el año que se desea verificar que existan los directorios adecuados
month : mes que se desea verificar
Regresa : 1 si un error ocurrió, 0 si los directorios están listos o fueron creados exitosamente

---

comal_event_save_info_to_db( )

```
guint     comal_event_save_info_to_db     ( gchar *user_gdbm_datebook_path,
                                            gchar *local_user_id,
                                            EventInfo event );
```

Almacena un evento dentro de la base de datos del usuario. Almacenar información del usuario "0" es inválido, no se genera ningún error, los datos son simplemente ignorados.

user_gdbm_datebook_path : ruta del directorio en donde se almacena los archivos con el calendario del usuario

local_user_id : identificador del usuario, es una cadena que representa un entero único por usuario, e.g. "23"

event : estructura con la información del evento a almacenar

Regresa : 1 si un error ocurrió, 0 si el evento se almacenó exitosamente

---

comal_event_delete_info_from_db( )

```
guint    comal_event_delete_info_from_db        ( gchar *user_gdbm_datebook_path,
                                                  gint year,
                                                  gint month,
                                                  gint day,
                                                  gint hour );
```

Elimina el evento de la base de datos del usuario que cuenta con la fecha y hora que se especifican como parámetros a la función.

user_gdbm_datebook_path : ruta del directorio en donde se almacena los archivos con el calendario del usuario

year : año de la fecha del evento que se desea eliminar

month : mes de la fecha del evento que se desea eliminar

day : día del mes del evento que se desea eliminar

hour : hora del evento que se desea eliminar

Regresa : 1 si un error ocurrió, 0 si el evento se eliminó de manera exitosa

---

g_print_event_info( )

```
guint       g_print_event_info                  ( EventInfo event );
```

Imprime en la línea de comandos el contenido de una estructura tipo EventInfo. Función utilizada con propósitos de depuración.

event : el evento del cual se imprime el contenido

Regresa : 0

comal_contact_load_info_from_db( )

```
gint     comal_contact_load_info_from_db
                                        ( gchar *user_gdbm_path,
                                          gchar *local_user_id,
                                          ContactInfo *contact,
                                          gchar *user_id );
```

Carga la información de un contacto de la base de datos del usuario.

user_gdbm_path : ruta del directorio en donde se almacena los archivos de las bases de datos del usuario

local_user_id : identificador del usuario local, es una cadena que representa un entero único por usuario, e.g. "23"

contact : estructura donde se almacena la información del usuario que fué obtenida de la base de datos

user_id : identificador del usuario del que se desea obtener información

Regresa : -1 si un error ocurrió, 1 si el usuario no se encuentra en la base de datos (contact es NULL) y 0 si la información fue extraída exitosamente y se encuentra en el parámetro 'contact'

---

comal_contact_get_alias( )

```
guint      comal_contact_get_alias
                                    ( gchar *user_gdbm_path,
                                      gchar *local_user_id,
                                      gchar *buffer,
                                      gchar *user_id );
```

Obtiene la cadena con el alias de uno de los miembros de la lista de contactos. El alias se almacena en el parámetro 'buffer'.

user_gdbm_path : ruta del directorio en donde se almacena los archivos de las bases de datos del usuario

local_user_id : identificador del usuario local, es una cadena que representa un entero único por usuario, e.g. "23"

buffer : utilizado para almacenar el alias del contacto

user_id : identificador del usuario del cual se quiere obtener su alias

Regresa : -1 si un error ocurrió, 1 si el usuario no se encuentra en la base de datos (buffer es NULL) y 0 si el alias fue obtenido exitosamente y se encuentra en el parámetro 'buffer'

---

comal_contact_get_number_of_contacts( )

```
gint    comal_contact_get_number_of_contacts        ( gchar *user_gdbm_path,
                                                      gchar *local_user_id );
```

Obtiene el número de contactos en la base de datos del usuario.

user_gdbm_path : ruta del directorio en donde se almacena los archivos de las bases de datos del usuario

local_user_id : identificador del usuario local, es una cadena que representa un entero único por usuario, e.g. "23"

Regresa : -1 si un error ocurrió, o el número de contactos en la base de datos

---

comal_contact_get_id_by_db_id( )

```
gint     comal_contact_get_id_by_db_id              ( gchar *user_gdbm_path,
                                                      gchar *local_user_id,
                                                      gchar *buffer,
                                                      gint db_id );
```

Obtiene el identificador del usuario dado su identificador dentro de la base de datos local, regresando el identificador en el parámetro 'buffer'.

user_gdbm_path : ruta del directorio en donde se almacena los archivos de las bases de datos del usuario

local_user_id : identificador del usuario local, es una cadena que representa un entero único por usuario, e.g. "23"

buffer : utilizado para almacenar el identificador del contacto

db_id : el identificador del usuario dentro de la base de datos local

Regresa : -1 si un error ocurrió, 1 si el usuario no se encuentra en la base de datos (buffer es NULL) y 0 si el identificador fue obtenido exitosamente y se encuentra en el parámetro 'buffer'

---

comal_contact_get_db_id_by_id( )

```
gint     comal_contact_get_db_id_by_id              ( gchar *user_gdbm_path,
                                                      gchar *local_user_id,
                                                      gchar *contact_id );
```

Obtiene el identificador del usuario, dentro de la base de datos local, dado su identificador único.

user_gdbm_path : ruta del directorio en donde se almacena los archivos de las bases de datos del usuario

local_user_id : identificador del usuario local, es una cadena que representa un entero único por usuario, e.g. "23"

contact_id : identificador del usuario del que se desea obtener su identificador en la base de datos local

Regresa : -1 si el usuario no esta en la base de datos, o el identificador del usuario dentro de la base de datos local

---

comal_contact_save_unknown_info_to_db( )

```
gint    comal_contact_save_unknown_info_to_db          ( gchar *user_gdbm_path,
                                                         gchar *local_user_id,
                                                         gchar *contact_id );
```

Almacena la información de un contacto del cual no se tiene mas que su número de identificación único.

user_gdbm_path : ruta del directorio en donde se almacena los archivos de las bases de datos del usuario

local_user_id : identificador del usuario local, es una cadena que representa un entero único por usuario, e.g. "23"

contact_id : identificador del usuario que se dará de alta en la base de datos local

Regresa : -1 si un error ocurrió, 1 si el identificador del usuario desconocido es "0" y 0 si la información fue almacenada de manera exitosa

---

comal_contact_save_info_to_db( )

```
gint    comal_contact_save_info_to_db          ( gchar *user_gdbm_path,
                                                 gchar *local_user_id,
                                                 ContactInfo contact );
```

Almacena la información del contacto contenida en 'contact' dentro de la base de datos local.

user_gdbm_path : ruta del directorio en donde se almacena los archivos de las bases de datos del usuario

local_user_id : identificador del usuario local, es una cadena que representa un entero único por usuario, e.g. "23"

contact : estructura que contiene la información del usuario que será almacenada en la de la base de datos local

Regresa : -1 si un error ocurrió, 1 si el identificador del usuario a almacenar es "0" y 0 si la información fue almacenada de manera exitosa

comal_contact_remove_pos_from_db( )

```
guint       comal_contact_remove_pos_from_db                ( GDBM_FILE gdf,
                                                              gint pos );
```

Remueve la posición del contacto que es removido de la base de datos local, esta función fue creada para uso interno de otras funciones, no para llamarla directamente.

gdf : un archivo tipo GDBM abierto previamente

pos : la posición dentro de la base de datos local del usuario que está siendo removido

Regresa : 1 si la posición no existe dentro de la base de datos, 0 si la posición fue borrada exitosamente

---

comal_contact_remove_info_from_db( )

```
gint     comal_contact_remove_info_from_db              ( gchar *user_gdbm_path,
                                                          gchar *local_user_id,
                                                          gchar *user_id );
```

Remueve la información de un contacto de la base de datos local.

user_gdbm_path : ruta del directorio en donde se almacena los archivos de las bases de datos del usuario

local_user_id : identificador del usuario local, es una cadena que representa un entero único por usuario, e.g. "23"

user_id : identificador del usuario que se desea ser removido de la base de datos

Regresa : -1 si un error ocurrió, 1 si el usuario no se encuentra en la base de datos y 0 si la información del usuario fue removida de manera exitosa

---

g_print_contact_info( )

```
guint       g_print_contact_info              ( ContactInfo contact );
```

Imprime en la línea de comandos la información del contacto contenida en el parámetro 'contact', esta función fue creada con propósitos de depuración.

contact : estructura que contiene la información del usuario a ser imprimida
Regresa : 0

comal_sync_get_event_db( )

```
gint        comal_sync_get_event_db                ( gchar *pilot_dev );
```

Obtiene la base de datos de eventos de un dispositivo Palm ("DatebookDB") y los almacena en la base de datos local.

pilot_dev : contiene la ruta del archivo que apunta al dispositivo Palm e.g. "/dev/pilot"
Regresa : -1 si un error ocurrió, 0 si la base de datos se obtuvo exitosamente

---

comal_sync_event_db( )

```
gint        comal_sync_event_db                    ( gchar *gdbm_path,
                                                     gchar *local_user_id,
                                                     gchar *pilot_dev,
                                                     gint year,
                                                     gint month,
                                                     gint day );
```

Envía los eventos de la base de datos local a un dispositivo Palm, se envían solamente los eventos especificados por la fecha que se define en los parámetros de la función.

gdbm_path : ruta del directorio en donde se almacena los archivos con el calendario del usuario
local_user_id : identificador del usuario local, es una cadena que representa un entero único por usuario, e.g. "23"
pilot_dev : contiene la ruta del archivo que apunta al dispositivo Palm e.g. "/dev/pilot"
year : año de la fecha de los eventos que serán sincronizados con el dispositivo Palm
month : mes de la fecha de los eventos que serán sincronizados
day : día de la fecha de los eventos que serán sincronizados
Regresa : -1 si un error ocurrió, 0 si la sincronización se realizó de manera exitosa

---

g_print_struct_tm_data( )

```
guint       g_print_struct_tm_data                 ( struct tm date );
```

Imprime en la línea de comandos el contenido de una estructura struct tm, esta función fue creada con propósitos de depuración.

date : estructura con información de fecha que se desea imprimir en la línea de comandos

Regresa : 0

---

g_print_appointment_info( )

```
guint     g_print_appointment_info          ( struct Appointment event );
```

Imprime en la línea de comandos el contenido de la estructura struct Appointment (un evento dentro de la base de datos DatebookDB de un dispositivo Palm), función creada con propósitos de depuración.

event : estructura con la información del evento que se desea imprimir en la línea de comandos

Regresa : 0

comal_sync_contact_db( )

```
gint      comal_sync_contact_db                ( gchar *gdbm_path,
                                                 gchar *local_user_id,
                                                 gchar *pilot_dev );
```

Envía todos los contactos de la base de datos local a un dispositivo Palm, almacenándolos en la base de datos "GContactsDB".

gdbm_path : ruta del directorio en donde se almacena los archivos de las bases de datos del usuario

local_user_id : identificador del usuario local, es una cadena que representa un entero único por usuario, e.g. "23"

pilot_dev : contiene la ruta del archivo que apunta al dispositivo Palm e.g. "/dev/pilot"

Regresa : -1 si un error ocurrió, 0 si la sincronización se realizó de manera exitosa

comal_sync_messages_db( )

```
gint      comal_sync_messages_db                ( gchar *gdbm_path,
                                                  gchar *local_user_id,
                                                  gchar *pilot_dev );
```

Envía todos los mensajes de la base de datos local a un dispositivo Palm, almacenándolos en la base de datos "GMessagesDB".

gdbm_path : ruta del directorio en donde se almacena los archivos de las bases de datos del usuario

local_user_id : identificador del usuario local, es una cadena que representa un entero único por usuario, e.g. "23"

pilot_dev : contiene la ruta del archivo que apunta al dispositivo Palm e.g. "/dev/pilot"

Regresa : -1 si un error ocurrió, 0 si la sincronización se relizó de manera exitosa

libCOMALIr

Bibliotecas para transferir información utilizando el dispositivo infrarrojo de los PDAs Palm y el soporte del sistema operativo Linux para comunicación infrarroja (IrDA/Linux).

---

comal_obex_init( )

```
obex_t *                    comal_obex_init                                ( );
```

Inicializa la sesión en la cual se puedan enviar y recibir datos a través de conexiones inalámbricas utilizando COMAL.

Regresa : una referencia a un descriptor de la sesión OBEX de intercambio de datos que sirve como identificador para conexiones futuras.

---

comal_obex_start_connection( )

```
int      comal_obex_start_connection                    ( obex_t *handle );
```

Inicializa una sesión de conexión para transferir datos de manera inalámbrica.

handle : apuntador a un descriptor de una sesión OBEX previamente abierta
Regresa : un valor > 0 si la conexión se inicializó correctamente y < 0 si un error ocurri

---

comal_obex_connect( )

```
int       comal_obex_connect                    ( obex_t *handle );
```

Establece una conexión inalámbrica con un dispositivo móvil, esta función debe llamarse inmediatament antes de intercambiar información.

handle : apuntador a un descriptor de una sesión OBEX previamente abierta
Regresa : 0 si se logró una conexión con el otro dispositivo, otro valor implica que un error ocurrió

---

comal_obex_disconnect( )

```
int       comal_obex_disconnect                    ( obex_t *handle );
```

Termina una sesión de transferencia de datos con otro dispositivo.

handle : apuntador a un descriptor de una sesión OBEX previamente abierta
Regresa : 0 si la sesión terminó con el otro dispositivo, otro valor implica que un error ocurrió

---

comal_obex_put_text_file( )

```
guint       comal_obex_put_text_file                 ( obex_t *handle,
                                                       gchar *file_name );
```

Transmite un archivo de texto a través de una conexión inalámbrica.

handle : apuntador a un descriptor de una sesión OBEX previamente abierta
file_name : ruta del archivo que se transmitirá al dispositivo móvil
Regresa : 0 si el archivo se transmitió de manera correcta, otro valor implica que un err ocurrió

---

comal_obex_put_contact( )

```
guint       comal_obex_put_contact                   ( obex_t *handle,
                                                       ContactInfo contact,
                                                       gchar *status );
```

Transmite la información de un contacto a otro dispositivo inalámbrico.

handle : apuntador a un descriptor de una sesión OBEX previamente abierta
contact : estructura con la información del contacto que se transmitirá
Regresa : 0 si la información del contacto se transmitió de manera cotrrecta y 1 si un er ocurrió

---

comal_build_object_from_contact_record( )

```
                                                    ( obex_t *handle,
```

```
obex_object_t *   comal_build_object_from_contact_record   ( obex_t *handle,
                                                            ContactRecord contact )
```

Construye un objeto OBEX a partir de una estructura tipo ContactRecord para ser enviado a través de un conexión inalámbrica.

handle : apuntador a un descriptor de una sesión OBEX previamente abierta
contact : estructura a partir de la cual se va a construir el objeto OBEX
Regresa : un objeto OBEX que puede ser transmitido por medio de una conexión inalámbrica.

CICESE malba@cicese.mx

Noviembre 9, 2000.