Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

Unidad Zacatenco

Departamento de Computación

Servicio de Comunicación por Eventos en Ambientes Heterogéneos con Registro Implícito de Dispositivos

Tesis que presenta

Christian Gatica Nava

para obtener el Grado de

Maestro en Ciencias en Computación

Director de la Tesis

Dr. José Guadalupe Rodríguez García

México, D.F. Marzo 2010

Resumen

Actualmente, con los avances de las tecnologías en electrónica y dispositivos para las tecnologías de la información, se han desarrollado reproductores multimedia portables (iPod), PDA's (asistentes digitales personales) y teléfonos móviles. Estos dispositivos están haciendo que la comunicación a través de redes inalámbricas sea más accesible, además se convierten en un medio de acceso a la información en cualquier lugar, para entornos conocidos como ubicuos. Los propósitos y ambientes donde se usan son muy variables, por lo tanto es necesario evaluar la usabilidad de estos dispositivos.

Debido a la proliferación de distintos tipos de dispositivos dentro de las organizaciones o empresas, es necesario implementar una gestión de dispositivos, con la finalidad de ayudar a reducir costos de mantenimiento e incrementar la eficiencia en cuanto a la comunicación.

La evolución de las redes inalámbricas y los contenidos multimedia son dos tendencias que rápidamente están emergiendo. Las aplicaciones que generan varios tipos de información (*e.g.*, e-mail, aplicaciones Java, audio, video, mensajería, multimedia, servicios de localización, etc.), están teniendo un papel importante en la forma de comunicarnos. En consecuencia, se ha generado el interés en el desarrollo de nuevos modelos y sistemas o servicios enfocados a la comunicación inalámbrica, haciendo frente a varios desafíos, principalmente la movilidad de los usuarios y la heterogeneidad de los dispositivos.

En esta tesis se presenta el diseño e implementación de un servicio de comunicación por eventos en ambientes heterogéneos con registro implícito de dispositivos. Nuestra propuesta integra los siguientes aspectos: (i) registro del usuario, para determinar el tipo de mensajes que recibirá después de estar registrado; (ii) registro de dispositivos móviles, para conocer las características y capacidades del dispositivo, e.g., sistema operativo, capacidad de almacenamiento, conectividad, memoria, servicios, etc; (iii) está enfocado a dos tipos de conectividad, red alámbrica e inalámbrica; (iv) servicio de mensajes para la notificación de eventos y (v) filtro de mensajes. El objetivo es atender el problema que se genera en las organizaciones o instituciones, donde los avisos, recordatorios y otro tipo de información se envía a través de correo electrónico, ocasionando una sobrecarga de e-mails en las bandejas de entrada. El servicio proporciona una interfaz Web donde el usuario edita un perfil, que servirá para hacer un filtrado en los mensajes que puede recibir y que son generados de acuerdo a ciertos eventos.

Para la evaluación del servicio, se realizaron pruebas que muestran su aplicabilidad. La fase de registro se realizó con éxito, se evaluó el servidor encargado de generar los mensajes, obteniendo resultados satisfactorios en el filtrado de los mensajes.

Abstract

With the advances on electronics we have now mobil devices with more capabilities . With this increase on capabilities new types of services are requested for example Multimedia, Text Messages, Voice over IP applications. Implementing those service on mobile devices gives as result that communications through wireless IP networks are becoming more popular. Users want to have access to those services whenever they have network connection, this means that environment can change very often as result it is necessary to know the mobile device capabilities.

Today organizations can have several mobile devices for everyday activities, those devices can have different capabilities. This gives as result the need to have some service for the administration of mobile devices in order to improve the communication and to reduce maintenance expenses.

The evolution of wireless networks and the emergence of multimedia services are two factors that are evolving quickly. IP Wireless applications are producing several kinds of traffic (e.g. e-mail, audio, video, multimedia messages, service location, etc.). This kind of applications are evolving the way the user are communicating.

As result of this evolution, new types of systems and services are needed, those services and applications must make face to some problems related to wireless networks, i.e. mobility of users and heterogeneity of devices.

In this work we present the design and implementation of a communication system for heterogeneous environments with implicit registering of mobile devices. Our proposition is composed of several mechanisms: i) the user register service, moreover of giving their personal data, user sets the type of messages which he is interested to receive, ii) At the same time while user supply their data, mobile device is registered automatically to discover their characteristics (e.g., operating system, storage capacity, connectivity, etc.), iii) our proposition works on wireless and wired networks, iv) an event notification service, and v) a filter mechanism for messaging transmission.

With this proposition we want to make face to the problem found on organizations where is needed a more dynamic system of communication, it is well known that some communication mechanisms like the e-mail are not dynamic in the sense that user can forget to read messages or do not open their inbox messages very often.

We have tested our service on the WLAN of the mobile computing laboratory of the Computing Sciences Department of the CINVESTAV. We have tested users an devices register, filters and notification event services with successful results.

Agradecimientos

A Dios, por darme la oportunidad de tener un logro más en mi vida.

Un agradecimiento especial al pilar más importante en mi vida, mi familia...

A mis padres: *Gracias por la vida maravillosa que me han dado, por los consejos, regaños, por eseñarme que con esfuerzo y dedicación todos los objetivos se pueden lograr, ustedes siempre están en mi corazón, los quiero mucho.*

A mis hermanos:
Gracias por su apoyo, su compañia en lo momentos dificiles, tomen en cuenta que con empeño y sacrificio los sueños pueden volverse realidad.

A Madai:
Gracias por lograr conmigo un nuevo desafío, por estar en los momentos mas difíciles de mi vida, por todo el amor que me has dado, por darme muchas lecciones de vida y sobre todo por permitirme estar a tu lado. Te amo negrita.

A la memoria de Tere y Adalid:
Hermanitos, siempre viviran en mi corazón, los quiero y los extraño.

A mi asesor, mi más profundo agradecimiento y admiración...

Dr. José Guadalupe Rodríguez García:
Gracias por haberme dado la oportunidad y brindado la confianza para realizar este trabajo de tesis. Por el tiempo y dedicación que ha implicado dirigir este trabajo y sobre todo por transmitirme sus conocimientos y experiencias.

A mis revisores:
Dra. Sonia Mendoza Chapa y Dr. Dominique Decouchant por el tiempo dedicado a la revisión de esta tesis.

Agradezco al Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional por brindarme la formación y las herramientas suficientes durante mis estudios de maestría.

Gracias al Consejo Nacional de Ciencia y Tecnología por haberme proporcionado el apoyo económico para sustentar mi estancia durante el tiempo que realicé mis estudios de maestría.

Sofía Reza:
Gracias por el apoyo y las palabras de aliento que me brindó durante todo este tiempo.

A mis compañeros:
Agradezco a mis mejores amigos Gox, Cheche, July, Pam, por todo el apoyo que me brindaron durante el transcurso de la maestria, principalmente por su amistad y sin olvidar a Ray, Lil, Anita, Pau, Jhony, Beto, Miguelon, Edgar, Lupita, es todo un placer contar con su amistad.

Índice general

Índice de figuras

Lista de Tablas

Capítulo 1

Introducción

Anteriormente, para poder estar conectado a Internet, revisar correo electrónico o realizar transferencias de datos, necesitábamos una conexión física a una red. Esas redes están constituidas por cables, en consecuencia, siempre teníamos que buscar un punto de conexión para poder conectar una computadora. La necesidad de estar siempre conectados a Internet y sobre todo la movilidad, provocaron que las redes inalámbricas aparecieran poco a poco en empresas, casas, aeropuertos, hoteles, hospitales, etc. Estas redes permiten que una persona pueda moverse, sin necesidad de tener una conexión física para estar conectado.

1.1 Comunicación móvil

Los sistemas de comunicación móvil en ambientes inalámbricos, hacen transparente la conexión entre los dispositivos que soporten esta tecnología. La historia de la comunicación inalámbrica se divide en tres generaciones; la primera generación (1G) hace referencia a la telefonía inalámbrica, alojada en estándares para teléfonos celulares análogos, que fue introducida en los años 80's, extendida con el desarrollo de teléfonos digitales en los 90's. La segunda generación (2G) fortaleció la tecnología digital en comunicaciones, con el sistema global de comunicación móvil (GSM), los celulares digitales personales (PDC) y estándares CDMA (acceso multiple por división de código) originarios de Estados Unidos (IS-95). La tercera generación (3G) fue iniciada en octubre de 2001 al momento de lanzarse la WCDMA (acceso múltiple por división de código de banda ancha) en Japón. La característica principal fue la introducción de servicios de transferencia de voz y datos.

Recientemente la empresa TeliaSonera [59] anunció el 14 de diciembre del 2009, en su portal de Internet, el lanzamiento del servicio de la red 4G en Suecia y Noruega, donde afirman que los usuarios podrán navegar a una velocidad teórica de 100 megabits por segundo, diez veces más rápida que la 3G.

Los avances en sistemas celulares, redes inalámbricas y redes de áreas personales (PAN's) están teniendo un papel importante en la forma de comunicarnos [1]. En la actualidad, existen dispositivos que soportan aplicaciones que hacen uso de la tecnología inalámbrica, por ejemplo, el e-mail, las aplicaciones Java, los navegadores Web, video conferencia, video bajo demanda, audio bajo demanda, mensajería multimedia y servicios de localización.

1.2 Tecnologías inalambricas

Dentro de las tecnologías de comunicación inalámbrica, la más popular es *Wi-Fi* (red inalámbrica) o estándar IEEE 802.11. Permite velocidades de transferencia de datos de 11 Mbps (802.11b) hasta 54Mbps (802.11g) y su cobertura abarca de 100 a 150 metros. Es ideal para una infraestructura de red inalámbrica local [51].

Hoy en día existen varias tecnologías inalámbricas, por ejemplo, *Wimax* [22] y *Wibro* [44], las cuales ofrecen un mayor alcance, mayor ancho de banda e incrementan la velocidad de transmisión de datos. Este auge tecnológico ha permitido desarrollar nuevas aplicaciones para diferentes servicios en los sistemas de comunicación móvil. Debido al crecimiento de las tecnologías inalámbricas, es necesario definir algunas especificaciones. La IEEE (Instituto de Ingenieros en Electricidad y Electrónica) es una de las principales organizaciones que elabora estándares. IEEE define normas que involucran una amplia gama de industrias y tecnologías, *e.g.*, poder y energía, tecnologías de la información, telecomunicaciones, nanotecnología, entre otras más. Una de las normas mas conocidas es el grupo de estándares 802 LAN/MAN, los cuales incluyen: *Ethernet* (802.3), redes inalambricas de area local (802.11), redes inalámbricas de área personal (802.15) y acceso inalámbrico a través de banda ancha (802.16).

El interés y la demanda de aplicaciones multimedia basadas en *Wi-Fi* está creciendo rápidamente, impulsado por nuevos dispositivos y la gran cantidad de usuarios de redes inalámbricas. Aplicaciones tales como VoIP, transferencia de audio y video en tiempo real y juegos interactivos son cada vez más solicitadas en redes *Wi-fi*.

1.3 Dispositivos para la comunicación móvil

En los últimos años, Internet ha experimentado una notable expansión. En consecuencia, la demanda de servicios multimedia basados en el Protocolo de Internet (IP), ha aumentado [3]. Los recientes desarrollos tecnológicos en los cuales se utilizan dispositivos personales como aparatos de monitoreo de la salud, *e.g.*, presión arterial, electrocardiogramas, oxímetro de pulso y monitoreo de azúcar en la sangre, tales avances han hecho que el cómputo ubicuo sea una herramienta que ayuda, principalmente, en la asistencia sanitaria.

En una organización, donde hay una movilidad constante por parte del personal, es importante tomar iniciativas para hacer frente a los problemas de conectividad y gestión de dispositivos, los cuales se proporcionan para auxiliar a los empleados de la organización en las actividades que realizan.

Con los avances en las telecomunicaciones, los dispositivos móviles han evolucionado para poder dar soporte a las nuevas formas de comunicación. Una de las áreas que más han evolucionado en los últimos años son precisamente las dedicadas al desarrollo de dispositivos móviles. Teléfonos y PDAs, han dejado de ser aparatos electrónicos sencillos, para convertirse en auténticas computadoras pequeñas, con características multimedia, donde la conectividad es uno de sus aspectos más importantes. Hoy en día es posible realizar diferentes actividades con los teléfonos móviles, con una libertad y movilidad que años atrás era difícil de lograr. La introducción de redes inalámbricas en una organización

Figura 1.1: Evolución de los dispositivos móviles

o en las empresas han impulsado el desarrollo de aplicaciones que tienen que ver con movilidad y cómputo ubícuo. Un caso especial a tratar ocurre debido al incremento, en número, de los diferentes dispositivos que se usan dentro de una organización. Se hace necesaria la implantación de módulos de gestión los dispositivos móviles, los cuales van a interactuar con los usuarios.

1.4 Diseño de aplicaciones basada en eventos

En el diseño de una aplicación, la mayoría de las veces, se define el orden de las operaciones que llevará a cabo dicha aplicación, con la ayuda de estructuras de control tales como secuencias, condicionales y ciclos. En este tipo de programas no se toma en cuenta los sucesos exteriores, *i.e.*, la interacción del usuario, los accesos a bases de datos y otros aspectos que pueden afectar las secuencias de control, debido a que no son procesos dinámicos. En este caso, la aplicación siempre decide cuando ejecutar sus funciones y procedimientos.

Una aplicación guiada por eventos es diferente, el programa no especifica la secuencia de las operaciones, las organiza como un conjunto de servicios listos para ser activados en respuesta a ciertos acontecimientos, *e.g.*, al momento que el usuario da clic en un botón, un sensor detecta un cambio en la temperatura de un lugar o cuando un mensaje llega en un puerto de comunicación. Cada evento determina el servicio que se debe realizar. Una vez que el servicio ha llevado a cabo su función, el programa vuelve a la espera de acontecimientos [52].

1.5 Gestión de dispositivos móviles

Un aspecto importante es lo que se desea gestionar. Además si se manejan dispositivos con características heterogéneas esto representa un factor más que debe ser tomado en cuenta. Existen tres características principales que un servicio o sistema de gestión de dispositivos debe proporcionar. Las caracterísitcas son las siguientes [58]:

- Inventario de *Hardware*: una de las primeras peticiones que se hace a una aplicación de gestión de dispositivos móviles, es poder hacer un inventario del *Hardware*, *i.e.*, la posibilidad de poder conocer sin intervención manual, la cantidad de dispositivos con los que se cuenta, así como el modelo, el sistema operativo, el espacio disponible en memoria, etc. Con una herramienta de este tipo se evita que una persona se dedique a recolectar la información, interrumpiendo el trabajo de los usuarios para tomar nota de las características. Un aspecto importante son las características que la aplicación será capaz de gestionar, *i.e.*, nivel de la batería y espacio libre en la memoria, esto dependerá del sistema operativo instalado en el dispositivo.

- Sincronización de la información: cuando se habla de procesos de sincronización, se habla de la transferencia de información y su integración en bases de datos, por lo que se necesita un adecuado procedimiento de sincronización entre la información del servidor de gestión y los dispositivos.

- Monitoreo y gestión remota: Esta característica es relativamente poco encontrada en las aplicaciones de gestión de dispositivos. Se tarta de hacer un seguimiento de los eventos del sistema operativo, generados por las actividades del usuario, *i.e.*, permite conocer las actividades que el usuario realiza con el dispositivo y qué uso hace de las aplicaciones instaladas o servicios proporcionados por la red. Mediante el monitoreo del *Hardware*, se hace el seguimiento de las características y propiedades del dispositivo, las cuales podrían tomarse en cuenta para llevar a cabo alguna actividad.

En el sector empresarial es común encontrar incertidumbre, en cuanto al uso de la tecnología de comunicaciones y redes de datos, los altos ejecutívos se preguntan qué tan confiables son las tecnologías que pudieran implementarse [58], *e.g.*, las actividades cotidianas, como las ventas de algún producto a través de una PDA o un *Smartphone*. Estos factores están caracterizados principalmente por la incredulidad y la falta de información, además, agregando la característica de movilidad de los nuevos dispositivos, se origina una nueva circunstancia que debe tratarse.

Con el éxito de las tecnologías inalámbricas y el desarrollo de dispositivos de última generación, se ha generado el interés en el desarrollo de nuevos modelos y diseños de sistemas o servicios enfocados a la comunicación inalámbrica, haciendo frente a varios desafíos, principalmente la movilidad de los usuarios y la heterogeneidad de los dispositivos.

Cuando se ha decidido por una solución para administrar y controlar los diferentes dispositivos móviles, que son usados dentro de una organización, el siguiente paso es definir el procedimiento que se llevará a cabo para hacer dicho módulo o servicio de gestión. Si en su momento fue difícil gestionar a los dispositivos de escritorio o fijos, ahora al tratarse de

dispositivos móviles, se agrega un poco más de complejidad al problema, precisamente a la característica de movilidad. La solución ideal para este problema es: tener un servicio de registro automático para todos los dispositivos móviles, donde el usuario no se preocupe por llevar su dispositivo con el administrador de la red para que le de alta en el sistema, el mismo dispositivo, al conectarse un servidor encargado del registro, se va a registrar automáticamente.

1.6 Planteamiento del problema

Dentro de las organizaciones o instituciones, el medio de divulgación de información más utilizado es el correo electrónico. Por este medio se envía cualquier tipo de aviso o recordatorio a los integrantes de la organización. En ocasiones, los *mails* contienen muy pocas líneas de texto debido a que son mensajes que de alguna manera se envían pensando que los integrantes de la institución están revisando la bandeja de entrada cada segundo; cosa que a menudo no sucede.

Por otro lado, al usar el servicio de correo electrónico como medio de difusión para enviar cualquier tipo de información, puede ocasionar una sobrecarga de mensajes en las bandejas de entrada de los usuarios. En ocasiones los correos electrónicos recibidos ni siquiera son del interés para el usuario o peor aún, el usuario se da cuenta de un e-mail importante mucho tiempo después.

Por lo tanto, si alguno de los e-mails, que requieren de una atención importante, se pudieran enviar a través de notificaciones a los dispositivos móviles (*smartphones*, PDA's o teléfonos celulares con interfaz inalámbrica) de los usuarios, los mensajes tendrían una atención más temprana que un correo electrónico.

Con lo mencionado anteriormente, se ha propuesto otra alternativa para divulgar información que hasta aquí resulta ser una buena opción. Sin embargo, al enviar los mensajes a todos los integrantes de la organización puede ocasionar algunas molestias por parte de los usuarios, debido a que podrían recibir mensajes que no son de su interés. Mediante un filtro de mensajes, la información puede encaminarse a un grupo específico de usuarios, entonces se evitaría el envío del mensaje a todos los integrantes de la organización y solo lo recibirían los usuarios realmente interesados.

Para completar la infraestructura de comunicación de mensajes cortos, se debe proporcionar un mecanismo de recepción de los mensajes para los dispositivos móviles. Por lo tanto, se debe facilitar al usuario la aplicación que se tiene que instalar en el dispositivo móvil.

En una organización suelen tener estructurados de manera jerárquica a sus integrantes, o existen clasificaciones de grupos de usuarios, a los que se les proporciona ciertas preferencias para recibir información a cerca de la misma organización. De alguna manera las preferencias son especificadas por cada integrante de la institución. Para poder conocer las prioridades de cada integrante debe existir un mecanismo que ayude a obtenerlas, y que al final genere un perfil específico del usuario.

Las soluciones más usadas para generar un perfil de usuario son basadas en entornos Web, a través de un sistema de este tipo se daría soporte al registro del usuario incluyendo la edición de su perfil, donde especificaría las preferencias de los tipos de mensajes que

desee recibir en su dispositivo móvil.

Debido a que los mensajes son dirigidos al dispositivo móvil del usuario, es necesario registrar también el dispositivo que va a usar. El motivo del registro del dispositivo es brindar la posibilidad de poder comunicar eventos que incluyan no solo mensajes de texto, si no mensajes multimedia que, tal vez se integren al servicio. Los tipos de información multimedia pueden ser: imágenes, audio, video o VoIP. Por lo tanto es necesario saber las capacidades y caraterísticas de los dispositivos para verificar si pueden dar el soporte a un determinado servicio multimedia. Tanto el registro del usuario, la edición del perfil, el registro del dispositivo y la aplicación de recepción de mensajes para los dispositivos móviles se pueden proporcionar dentro de un servicio en Web.

1.7 Objetivos de la tesis

Objetivo general

Incorporar un servicio Web de registro y un servicio de mensajes, el objetivo general de la presente tesis es proponer el servicio SEGEMENS, el cual da soporte a la propuesta para la difusión de la información a través de mensajes, dirigida a los dispositivos móviles de los usuarios. Además, por medio de filtros se evita que los mensajes sean enviados a todos los usuarios del servicio, *i.e.*, solo se envía el mensaje a los usuarios que les interesa la información que contiene.

Objetivos particulares

Para lograr el objetivo general, es necesario desarrollar algunos objetivos particulares, los cuales son:

- desarrollar el servicio de registro de usuario y dispositivos móviles, donde se establezcan los mecanismos para el registro de la información en la base de datos y las funciones para la edición del perfil
- implementar el servicio de mensajes cortos
- implementar mecanismos de filtros de mensajes
- implementar el envío de mensajes a través de mecanismos basados en eventos, donde los mensajes pueden ser enviados al ocurrir un evento provocado por el administrador del servicio de mensajes o al producirse un evento automático.
- definir la arquitectura de comunicación entre los componentes del servicio
- desarrollar un sitio web orientado a dispositivos móviles
- desarrollar la interfaz de usuario (GUI) con la cual el administrador del servicio de mensajes va a interactuar
- desarrollar la aplicación encargada de recibir los mensajes en el dispositivo móvil, la cual debe tener la particularidad de ser portable, *i.e.*, debe de instalarse sin problemas en otros dispositivos móviles sin importar sus características heterogéneas de *Hardware* y *Software*

1.8 Organización de la tesis

El presente trabajo de tesis se encuentra organizado en cinco capítulos de la siguiente manera. El capítulo 2 se menciona el estado del arte, *i.e.*, los trabajos más sobresalientes relacionados con nuestro proyecto de investigación. Además de mencionar las plataformas de programación existentes enfocadas a dispositivos móviles.

En el capítulo 3 se describe la propuesta de solución del problema, se presenta el diseño de la arquitectura de cada elemento que conforma la aplicación. Se justifica el uso de la herramienta para el desarrollo del sitio Web que se encarga de realizar el registro de los dispositivos y usuarios en la base de datos, también se explican los detalles de la aplicación encargada del servicio de mensajes, se describe el diseño del servidor encargado del filtro de los mensajes, también se explica el diseño de la aplicación cliente que se instala en cada uno de los dispositivos móviles donde llegarán los mensajes.

El capítulo 4 presenta el desarrollo de las aplicaciones para el registro de usuarios y dispositivos móviles, se describe la programación de los *servlets* encargados del registro de usuario y dispositivos, además se explica el desarrollo de los programas hechos en el lenguaje de programación *Java* que se encargan del envío y recepción de los mensajes. También, se describen las evaluaciones propuestas, los resultados obtenidos y un análisis detallado de cada una de las pruebas establecidas. Los resultados presentados tienen que ver con el filtrado de mensajes y la respuesta de los dispositivos móviles al interactuar con el sitio web encargado del registro

En el capítulo 5 se describe las conclusiones y se finaliza la investigación correspondiente a la tesis, además se mencionan los trabajos futuros.

Capítulo 2

Marco Teórico

2.1 Sistemas de comunicación móvil

Las comunicaciones móviles aparecieron comercialmente a finales del siglo XX. Países como Noruega, Suecia y Dinamarca, fueron de los primeros en tener sistemas de telefonía móvil, posteriormente surgieron sistemas más avanzados, *e.g.*, radiobúsquedas (GPS) y redes móviles privadas. Después apareció la telefonía digital, computadoras portátiles y agendas personales, capaces de conectarse vía inalámbrica con otros dispositivos. Actualmente el auge principal es la integración de las comunicaciones móviles y la Internet, esto a dado como resultado el surgimiento de un amplio campo de investigación para el desarrollo de aplicaciones.

2.1.1 Evolución de los sistemas de comunicación móvil

Nuevos paradigmas para la creación de redes de telecomunicaciones han aparecido, los cuales requieren una administración que permita la gestión integrada de infraestructuras de redes heterogéneas de gran complejidad (cableadas, inalámbicas, ad hoc, edge) y además soporten conectividad ubicua, *i.e.*, conexión en cualquier lugar, a cualquier hora, con cualquier dispositivo y cualquier servicio [14].

Hasta ahora se han dado tres generaciones distintas de las redes de comunicación móvil incluyendo sus versiones intermedias, la primera generación (1G) se basa en las tecnologías de telefonía inalámbrica mediante los estandares de telefonía celular analógica, los cuales se introdujeron en los años 80 y continuaron hasta que fueron sustituidos por los teléfonos celulares digitales en 1990. La segunda generación de (2G) fue basada en la tecnología celular digital, los modelos más populares en esta generación son el Sistema Global de Comunicaciones Móviles (GSM) y el Acceso Múltiple por División de Código (CDMA). La tercera generación (3G) iniciada en octubre de 2001, surgió al lanzarse en Japón las redes WCDMA (Acceso Múltiple por División de Código de banda ancha), la cual ofrece velocidades de datos mucho más altas en dispositivos móviles y portátiles que las ofrecidas en las anteriores generaciones. El 14 de diciembre del 2009 la empresa TeliaSonera publicó en su página Web la inauguración de la nueva red 4G [59] en los países de Suecia y Noruega, dicha empresa se autodenomina como el primer proveedor en el mundo de la tecnología 4G, ofreciendo una velocidad de banda ancha de 100 Mbps.

Tecnología	1G	2G	2.5G	3G	3.5G	4G
estándar	AMPS, TACS NMT, ETC	TDMA, CDMA GSM, PDC	GPRS, EDGE 1xRTT	WCDMA, CDMA200	HSDPA WiBro Wimax	único estándar
Implementación	1984	1991	1999	2002	2006	2010
Velocidad de datos	1.9 Kbps	14.4Kbps	384 Kbps	2Mbps	10-50Mbps	100Mbps-1Gbps
Multiplexado	FDMA	TDMA, CDMA	TDMA, CDMA	CDMA	CDMA, OFDMA	CDMA, OFDMA, ...
Servicio	voz analógica, sicronización de datos a 9.5Kbps	voz digital, mensajes cortos	Altas capacidades, datos empaquetados	Altas capacidades, banda ancha arriba de los 2Mbps	Internet inalámbrico de alta velocidad, multimedia	completamente orientado a IP, multimedia, velocidad arriba de 1Gbps

Tabla 2.1: Generaciones de la comunicación móvil

Durante el primer cuarto del año 2010 estará abierto el servicio a un bajo costo, con la finalidad de que los usuarios adopten esta nueva conexión de banda ancha móvil. En en cuadro 2.1 se presenta otra vista general de las diferentes generaciones que ha tenido la comunicación móvil.

Los avances en sistemas de telecomunicación tienen el objetivo de jugar un rol importante en la manera de comunicarnos. Se espera que en el futuro todas las conexiones sean inalámbricas, en consecuencia, al incrementarse las capacidades de los sistemas de comunicación móvil se podrá necesitar del desarrollo de aplicaciones y servicios más amplios [8]. En la actualidad algunas de las aplicaciones más populares dentro de los sistemas de comunicación móvil se encuentran los siguientes:

- E-mail: Aunque es una aplicación independiente de los sistemas de comunicación móvil, es posible recibir y enviar mensajes de correo electrónico a través de un dispositivo móvil. Es uno de los servicios más populares en Internet, incluyendo a los usuarios que lo usan a través de un *smartphone* o un PDA.

- Navegar por Internet: Los teléfonos de última generación, son capaces de conectarse a Internet por medio de una interfaz inalámbrica y un navegador, han convertido a este servicio en toda una herramienta de uso cotidiano.

- Servicio de mensajeria multimedia (MMS): Es un servicio que además de permitir el envío de mensajes de texto, puede transmitir varios tipos de contenidos multimedia (audio, video, imágenes). Es importante tener en cuenta que el dispositivo móvil debe tener el soporte para poder generar este tipo de información, *i.e.*, si el dispositivo tiene una cámara integrada, posiblemente puede enviar un video grabado con ella a través de MMS.

- Aplicaciones *Java* : La mayoría de los dispositivos móviles actuales son capaces de ejecutar aplicaciones basadas en *Java* . Dichas aplicaciones se ejecutan sobre una máquina virtual con soporte en *kilobytes* (KVM) que subyace en el dispositivo. Es una capa de software que permite ejecutar la misma aplicación *Java* en diferentes dispositivos, sin importar sus componentes de hardware y sin hacer modificaciones en el código.

- Descarga de videoclips y música: Las redes 3G permiten al usuario descargar audio o video con mayor velocidad. Los dispositivos móviles de hoy en día cuentan con reproductores multimedia y pantallas con gran resolución, lo que permite tener acceso a diferentes tipos de contenidos multimedia.

- Servicios de localización: Actualmente hay redes móviles que ofrecen servicios de localización y de posicionamiento, ya sea a través de dispositivos dedicados o sobre dispositivos móviles de propósito general. Estos servicios permiten al usuario móvil tener acceso a información acerca de restaurantes, hospitales, hoteles, centros comerciales y localización de calles a través de mapas. El servicio más famoso es el sistema de posicionamiento global (GPS).

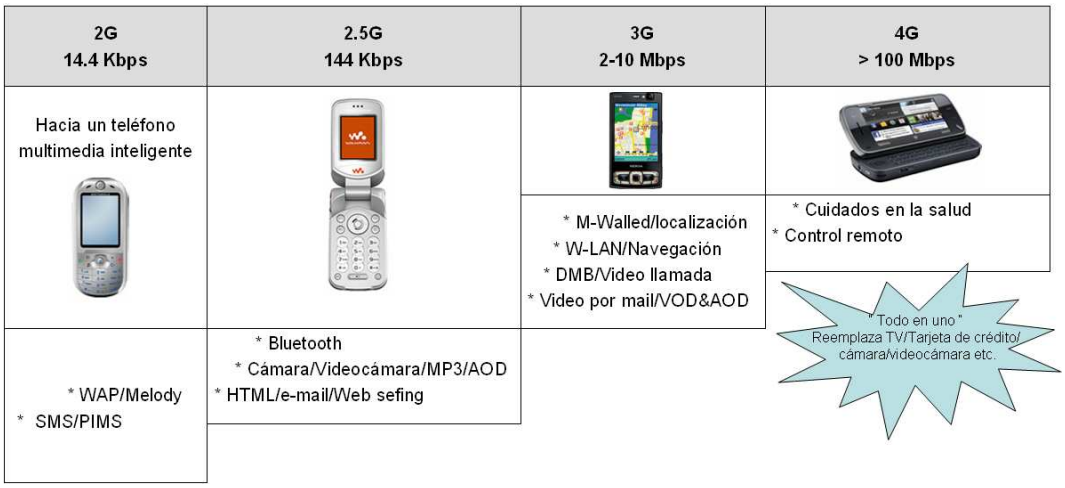

Figura 2.1: Evolución de los servicios y aplicaciones móviles

En la próxima generación de los teléfonos celulares, servicios tales como seguridad en el hogar, control de máquinas de manera remota, asistencia en los aereopuertos, seguimiento de la entrega de paquetes y servicios de medición remota de magnitudes físicas podrían estar disponibles de manera comercial. Con esto se espera que los dispositivos móviles sean un componente muy importante para las próximas generaciones de aplicaciones y servicios. En la Fig 2.1 se puede observar la evolución de los servicios que se han ido proporcionando dependiendo de las diferentes generaciones en las comunicaciones móviles.

Las tecnologías de comunicación más utilizadas actualmente para la transmisión de información en las redes son denominadas de acceso múltiple, debido a que más de un usuario puede utilizarlas. Las principales son las siguientes [5]:

- FDMA (Acceso Múltiple por División de Frecuencia). Tiene acceso a las celdas dependiendo de las frecuencias; separa el espectro en distintos canales de voz al dividir el ancho de banda en varios canales uniformemente, según las frecuencias de transmisión.

- TDMA (Acceso Mútiple por División de Tiempo). Divide el canal de transmisión en particiones de tiempo. Comprime las conversaciones digitales y luego las envía utilizando la señal de radio por un periodo de tiempo.

- CDMA (Acceso Múltiple por División de Códigos). Luego de digitalizar la información, la transmite a través de todo el ancho de banda del que se dispone, a diferencia de TDMA y FDMA. Las llamadas se sobreponen en el canal de transmisión, diferenciadas por un código de secuencia único.

- GSM es un estándar mundial llamado Sistema Global para las Comunicaciones Móviles. Combina TDMA y FDMA con un par de canales de frecuecia *duplex*. Las conexiones pueden utilizar tanto voz como datos, lo que permitió el avance del envío y consumo de datos a través de los celulares. Las implementaciones más veloces de GSM son GPRS, EDGE y UMTS [6]:

  - GPRS (General Packet Radio Service) Se trata de una comunicación basada en paquetes de datos. Los intervalos de tiempo son asignados mediante un sistema basado en la necesidad a la conexión de paquetes, *i.e.*, si no se envía ningún dato, las frecuencias quedan libres.
  - EDGE (Enhanced Data Rates for Global Evolution). Puede ser usada en cualquier transferencia de datos basada en conmutación de paquetes, como lo es la conexión a Internet. Los beneficios de EDGE sobre GPRS se pueden ver en las aplicaciones que requieren una velocidad de transferencia de datos o ancho de banda alta, como video y otros servicios multimedia.
  - UMTS (Universal Mobile Telecommunications System). Tecnología utilizada para lo móviles de tercera generación, sus tres grandes características son la capacidad multimedia, velocidad de acceso a Internet y una transmisión de voz con calidad similar a las redes fijas.

Las redes y la Internet se han desarrollado de manera asombrosa en los últimos años. Ahora no sólo se utilizan para el transporte de datos (e-mail, FTP, WWW), se han desarrollado nuevas aplicaciones, las cuales aumentan la demanda de la infraestructura de las redes, *e.g.* telefonía IP, transferencia de audio y video, videoconferencias y juegos (colaborativos) en línea. A todo ésto se le puede llamar servicios de comunicación interactiva en redes IP, las cuales ganan la aceptación de los clientes si ofrecen calidad de servicio [4].

2.1.2 Dispositivos móviles

Existen varias maneras de referirse a los dispositivos móviles, en inglés suelen llamarlos *information device*, *information appliance, consumer electronic, embedded device o small device*. Generalmente estos dispositivos tienen las siguientes características:

- son pequeños,
- tienen algunas capacidades de procesamiento,
- tienen conexión a una red,
- cuentan con memoria limitada,
- son fabricados para realizar funciones específicas, pero pueden usarse para realizar otras actividades

- la mayoría de las veces son de uso individual.

La movilidad es una característica muy importante en este tipo de dispositivos [28], por su tamaño son fáciles de transportar y se pueden usar al momento de desplazarse de un sitio a otro. El término inalámbrico también se encuentra presente, los dispositivos con esta particularidad son aquellos que son capaces de comunicarse o acceder a una red sin cables [55], cada vez son más completos, permiten estar conectados a infinidad de dispositivos, de una manera muy simple. La primer referencia de los dispositivos móviles se dió oficialmente en 1983, al aparecer de manera comercial el Motorola dynaTAC 8000X, diseñado por el ingeniero Martin Cooper, el primer radioteléfono totalmente móvil [60].

Conforme han evolucionado se agregaron más funcionalidades, actualmente no sólo funcionan como teléfonos, ahora también permiten tomar fotos, grabar video, mensajería, etc. En un principio los sistemas de comunicación eran análogos, lo cual generaba altos niveles de congestionamiento. Cuando las telecomunicaciones se adaptaron para trabajar con tecnologías digitales los datos se convirtieron en códigos binarios, para poderlos comprimir, así se disminuye de 3 a 10 veces el espacio que utilizaba una señal analógica. Esto produce un aumento drástico en la capacidad de los sistemas en cuanto a la manipulación de datos y procesamiento de los mismos. Para lograr esta compresión y descompresión de los datos, los dispositivos procesan millones de cálculos por segundo, la mayoría de éstos cuenta con los siguientes componentes físicos:

- un microprocesador llamado DSP (Digital Signal Processor). Realiza todas las operaciones del dispositivo como lo hace un microprocesador en una computadora. Las velocidades de estos microprocesadores alcanzan los 40 MIPS (Millones de Instrucciones Por Segundo). Realiza tareas de compresión y descompresión, procesa los eventos del teclado, gestiona los comandos, controla señales, envía la información a la pantalla, entre otras funciones,
- una tarjeta de circuitos similar a la tarjeta madre de una computadora,
- un altavoz por donde el aparato emite el sonido, después de su descompresión y decodificación en el microprocesador,
- una pantalla de cristal líquido (LCD) donde se muestra la información de manera visual,
- un teclado a través del cual el usuario interactúa, en la actualidad se cuenta con pantallas táctiles que están sustituyendo a los teclados,
- una antena receptora/emisora de señales,
- una batería que proporciona la energía eléctrica al dispositivo.

Es indudable el vertiginoso avance de las posibilidades que tienen los dispositivos móviles actuales en la comunicación, el acceso de manera remota y móvil y el manejo de aplicaciones que hasta hace poco tiempo sólo eran reservadas a plataformas fijas.

Además de las características y capacidades mencionadas anteriormente, otro de los aspectos importantes es el sistema operativo con el que trabaja cada unos de los dispositivos. Actualmente se habla mucho de *IPhone OS 3.0* de Apple y también de *Andriod* de *Google*, dos sistemas operativos mantienen una competencia por dominar el mercado. Sin embargo existe una gran diversidad de sistemas operativos para dispositivos móviles, principalmente:

- *Symbian OS*, es el sistema operativo de los dispositivos *Nokia*, nació en 1981 y se presento hasta 1998. El núcleo de *Symbian* esta formado por una parte básica (microkernel y los controladores de dispositivos), *middleware* (servidores, seguridad y marcos de trabajo para aplicaciones) y comunicaciones (telefonía, mensajería y acceso a redes). Un aspecto a resaltar es su robustez, ya que permite el diseño de aplicaciones multiplataforma. La mayoría de sus componentes han sido creados en C++. La última versión estable es la 9.5. Aunque está estrechamente relacionado con *Nokia*, en su momento fue el sistema operativo más utilizado por diferentes compañias como Sony Ericsson, PSION (compañia creadora de EPOC, sistema anterior a *Symbian OS*), *Samsung*, *Siemens*, *Arima*, *Benq*, *Fujitsu*, *Lenovo*, *LG*, *Motorola*, *Mitsubishi Electric*, *Panasonic*, *Sharp*, etc [20].

- *Windows Mobile*, sistema operativo de Microsoft para dispositivos móviles, se popularizó hasta la aparición de la versión 5.0, donde empezó a tener éxito en el mercado. Fue implementado por varias empresas para sus agendas electrónicas tales como *HP*, *Samsung*, *Qtek* y hasta en la misma *Palm*. *Windows Mobile* se caracteriza por sus aplicaciones de oficina, gracias a eso fue muy bien recibida por empresarios y ejecutivos. A diferencia de *Symbian*, este sistema no fue diseñado para ahorrar batería, en la version 6.0 Microsoft corrigió muchos errores. *Windows Mobile* usa una interfaz gráfica muy similar al *Windows* de escritorio, ya que utiliza una barra de tareas. A partir de la salida del *smartphone HTC Touch*, el sistema operativo empezó a adaptarse mejor para el control con los dedos en las pantallas táctiles [40].

- *iPhone OS* es el sistema operativo móvil de Apple, uno de los más recientes del mercado, fue desarrollado para el dispositivo *iPhone* y para el *iPod Touch*. Está basado en el kernel del sistema *Mac OS X* (sistema operativo de las MAC), utiliza poco espacio en disco, alrededor de medio giga, está diseñado para que las aplicaciones anteriores funcionen en las nuevas versiones del sistema. La última versión es la 3.0, su interfaz gráfica la conforman varios escritorios donde se organizan las aplicaciones, lo cual la hace sencilla y agradable al usuario [65].

- *Android*, relativamente es el sistema más reciente, es de código libre y gratuito, desarrollado por *Google*. Las aplicaciones escritas para este sistema operativo están programadas en *Java* y controlan los dispositivos a través de bibliotecas desarrolladas por *Google*. Tiene una interfaz gráfica adaptada para pantallas táctiles. Cuenta con 3 escritorios donde se observa la organización del contenido por carpetas, *widgets* y aplicaciones. Al ser un sistema operativo de código abierto, ha sido utilizado por empresas para implementar el mismo sistema operativo con distintas interfaces gráficas [57].

El sistema operativo con más presencia en el mercado es *Symbian OS*, le sigue *Windows Mobile*, *RIM BlackBerry, Linux, iPhone OS, Android y Palm OS* respectivamente. Hoy en día, las tendencias de las comunicaciones, especialmente las móviles, y el desarrollo constante de nuevos dispositivos móviles, van encaminándose hacia el uso en la vida cotidiana, *i.e.*, se pretende integrar dispositivos inteligentes como herramientas de uso diario, para que las personas puedan interactuar a través de estos dispositivos de manera transparente en cualquier circunstancia y situación. A toda esta tendencia se le da el nombre de cómputo ubicuo.

2.1.3 Cómputo ubicuo

El cómputo ubicuo se refiere a la integración de las tecnologías de la información en la vida cotidiana, el objetivo es que una computadora se vea como un elemento más en el entorno o ambiente, además se desea insertar componentes computacionales que interactúen con las personas y con sus actividades diarias de forma natural. Mark Weiser introduce en su artículo "*The Computer for the 21st Centuruy*" [15] los conceptos que definen al cómputo ubicuo.

Weiser propone cuatro ideas para construir un entorno ubicuo:

La primer idea se refiere al propósito de una computadora, el cual es ayudar al usuario a realizar otras tareas, *i.e.*, actualmente la interacción entre la computadora y una persona se basa en una pantalla que visualiza una serie de ventanas, las cuales contienen los elementos necesarios para llevar a cabo una tarea específica. En este caso, la persona debe estar atenta a la pantalla de la computadora, ocasionando que el usuario solo pueda realizar pocas tareas y que dependen de la misma computadora con la que esté trabajando.

La segunda idea propone que la mejor computadora es la que proporciona servicios de manera transparente, *i.e.*, varias computadoras interactúan con un usuario en un lugar al mismo tiempo, estas computadoras deben perderse en el ambiente de tal manera que sean invisibles, discretas y deben dar la sensación de que forman parte del entorno de forma transparente.

La tercer idea expresada por Weiser dice que las computadoras deben extender el subconsciente; si una computadora es desarrollada para actuar de manera intuitiva y natural, se crearía una computadora más eficiente para realizar actividades. Como ejemplo, cuando se viaja en automóvil, se pueden ver las señales de tránsito, estos señalamientos pueden ser interpretados por una persona sin necesidad de leer específicamente su significado.

Por último la cuarta idea menciona que la tecnología debe crear calma; actualmente estamos rodeados de varios elementos que atraen la atención de las personas tales como: dispositivos móviles, Internet, correo electrónico, televisión, radio, etc. Esto hace que las personas presten más atención de la necesaria a estos elementos, menciona que la atención del ser humano debe ser dividida en dos partes, la parte central y la parte periférica. La parte central se origina cuando se presta una total atención a algún objeto o evento, mientras que la antención periférica se describe como cuando sabemos que hay algo presente, pero no es el centro de atención y es ignorado la mayor parte del tiempo.

El cómputo ubicuo es un paradigma latente y vivo en la actualidad, donde se pueden introducir todas las tecnologías actuales y anteriores, unidas o aisladas para facilitar la vida a las personas y además, se pretende que se vuelvan parte de las actividades

cotidianas. Cada avance o desarrollo que va apareciendo, después se convierte en dispositivos o tecnologías comunes y necesarias, de tal manera que se ven como si siempre hubieran existido.

Pareciera que el Internet siempre ha existido, pero la realidad es que solo lleva medio siglo de vida, eso mismo sucede con los dispositivos móviles con los que hoy en día las personas interactuar diariamente. El desarrollo de prototipos basados en cómputo ubicuo determinará la tendencia de las tecnologías y dispositivos empleados, de este modo se descubrirá donde puede ser aplicada y en que aspectos puede ser mejorada.

2.2 Arquitectura Orientada a Servicios (SOA)

A lo largo de los años, nuevas tecnologías y arquitecturas han surgido en el mundo de las tecnologías de la información (TI), algunas de ellas han sido fuertemente promovidas, pero sólo pocas han sido ampliamente aceptadas. Una de las más destacadas es SOA (Arquitectura Orientada a Servicios). Se pueden encontrar diferentes definiciones para esta metodología de modelado y diseño de aplicaciones [18], la W3C (el consorcio *World Wide Web*) define a SOA como un conjunto de componentes que pueden ser invocados y cuyas interfaces se pueden publicar y descubrir, esto suena a servicios Web.

Este punto de vista se nota demasiado técnico y probablemente no sea del todo correcto según Sprott y Wilkes [62], definen a SOA como las políticas y marcos de trabajo que permiten funcionalmente que las aplicaciones sean proporcionadas y solicitadas como un conjunto de servicios. Dichos servicios pueden ser invocados, publicados y descubiertos, además son independientes de la implementación, *i.e.*, los servicios pueden estandarizarse para poder integrarlos en diferentes sistemas que respeten la lógica de SOA.

Los sistemas informáticos tradicionales han sido organizados en sitios independientes e incompatibles, que contienen tanto los procesos de negocio como sus funciones automatizadas. Por ejemplo, el proceso de contratación de una póliza de seguro y las funciones que calculan la prima, la emisión de recibos, etc., forman parte del mismo bloque. Los sistemas diseñados de esta forma han conseguido una mejor productividad en las empresas, automatizando procesos de negocio, pero por su naturaleza los cambios y adaptaciones a nuevas necesidades puede hacerlos más lentos y costosos.

Para que los sistemas sean más ágiles es necesario poder combinar los distintos componentes del sistema, donde los sistemas hechos en forma centralizada plantean muchas restricciones. La arquitectura SOA separa los procesos de negocios de las funciones automatizadas, colocando a las funciones en un diccionario de servicios como módulos individuales, permitiendo su utilización por parte de cualquier área de la organización o empresa. SOA es una arquitectura que trata de estructurar las aplicaciones de negocio y la tecnología para responder de forma rápida y flexible, puede verse como una evolución de la arquitectura tecnológica y de negocio de toda empresa.

Hoy en día los negocios exigen crear aplicaciones cada vez más complejas, en menos tiempo y con menos presupuesto. En ocasiones, para crear estas aplicaciones, se necesita de ciertas funcionalidades que ya están implementadas en otros sistemas. Es aquí donde se llega al punto de elegir dos caminos, reutilizar la funcionalidad ya implementada que implica menos tiempo pero es más compleja o reimplementar dicha funcionalidad, aunque

implica más tiempo y la mayoría de las veces es la más fácil y segura. Si bien implica más tiempo, la segunda opción es la más elegida.

La metodología para el modelado y diseño de aplicaciones SOA se conoce como análisis y diseño orientado a servicios. Para que un proyecto SOA tenga éxito, los desarrolladores deben tomar en cuenta modelos de planificación, herramientas e infraestructura. En ocasiones se van a encontrar con algunos inconvenientes técnicos externos a su modelo, *i.e.*, se debe utilizar un sistema de mensajería confiable, las respuestas del servicio puede ser afectada por aspectos como problemas en la red, configuración, etc. Todo esto se tiene que tomar en cuenta diseñando mecanismos de eventualidad para evitar que las aplicaciones y servicios dependientes se queden parados. Algunas características principales de la arquitectura SOA se mencionan a continuación:

- la funcionalidad se encapsula en servicios independientes entre sí,
- debe existir una alta interoperabilidad entre los servicios,
- la aplicación final será la encargada de dirigir la ejecución de los servicios,
- su diseño e implementación, se realiza bajo cualquier infraestructura cliente-servidor,
- generalmente son diseñados como servicios web,
- son arquitecturas multicapa y se utilizan diferentes tecnologías para cada aplicación, *e.g.*, para los procesos de negocio se usan servicios web, para organización de servicios se usa un lenguaje de ejecución de procesos de negocios llamado BPEL y para las interfaces gráficas se utiliza una interfaz Web (JSP, PHP, JSF, etc).

En resumen, el paradigma principal de SOA se refiere a la máxima interoperabilidad entre plataformas heterogéneas, las arquitecturas SOA empiezan a aplicarse como modelo de referencia a soluciones más optimizadas y adaptables, reducir tiempos, costos, y agilizar los procesos. La arquitectura SOA puede llegar a ser un avance hacia la evolución de la Web.

2.3 Plataformas de desarrollo para dispositivos móviles

Cuando se habla de equipos móviles, principalmente se hace referencia a PDA's, *handhelds* y *smartphones*. Son equipos que pueden llevarse a cualquier parte, se conectan a alguna red y son fáciles de manejar, sin utilizar mucho espacio y no tienen el peso de una laptop.

El concepto de movilidad es un aspecto importante, se refiere a tener los datos, aplicaciones y los dispositivos en cualquier lugar. Al fusionar estos dos elementos es posible desarrollar aplicaciones que faciliten el uso de los dispositivos, y que auxilien en las actividades del usuario.

Todos los programadores que desarrollan aplicaciones para dispositivos móviles se enfrentan a un dilema: decidir con qué plataforma trabajar. Para poder decidir con qué programar existe una variedad de consideraciones de acuerdo al propósito y escenario

para el que van ser utilizadas, todavía en ocasiones se tiene la creencia equivocada de que desarrollar aplicaciones móviles es lo mismo que desarrollar una aplicación de escritorio.

Por eso es importante tener en cuenta las consideraciones a seguir de acuerdo al objetivo de la aplicación, *e.g.*, si se desea abarcar una gama de dispositivos o la mayor cantidad de dispositivos posible, se tiene que tomar en cuenta aún más aspectos. Algunas de ellas son: el tipo de aplicación que se desea crear, los sistemas operativos y plataformas de desarrollo, capacidades de cada dispositivo, limitaciones de conectividad, lenguajes para navegadores, entre otras.

Los tipos de aplicaciones van desde las basadas en mensajería (SMS/MMS), basadas en Internet (Wap: WML-WAP 1.0, Web: XHTML-WAP 2.0 y Web enriquecido: Internet, Ajax y Plug-ins), las llamadas *STAND-ALONE* (lenguaje nativo y lenguaje intermedio) y aplicaciones mixtas. Del mismo modo, existen varios sistemas operativos que rigen los dispositivos, los más populares son: *Symbian OS*, *Windows Mobile (Windows CE)*, *iPhone OS*, *Palm OS*, *Android* y *BlackBerry OS*

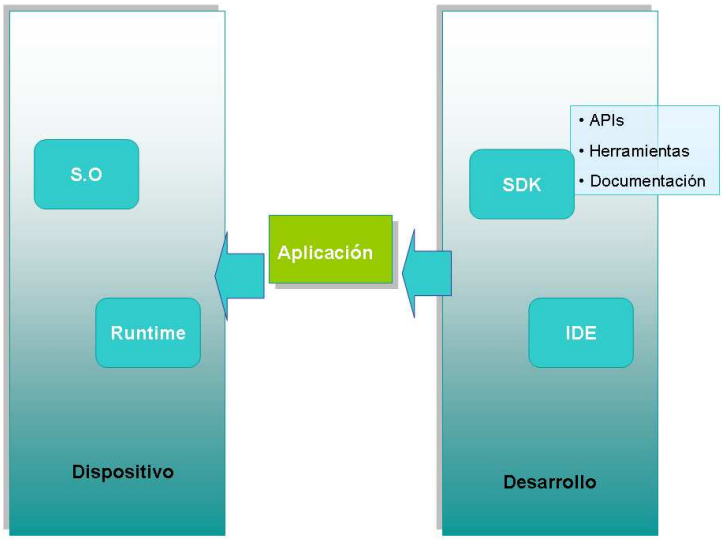

Figura 2.2: Herramientas de desarrollo, ambiente general

Existen dos factores que dan la pauta para elegir la plataforma con la que se van a desarrollar las aplicaciones: El primero es tener en cuenta el tipo de dispositivo en el cual se va a programar y el conocimiento de software de desarrollo. El segundo es lo que se desea desarrollar. En la Fig. 2.2 se observa un esquema general de las herramientas de desarrollo para aplicaciones móviles. En la actualidad existen los llamados kit de desarrollo de software (SDK's) y los ambientes de desarrollo integrados (IDE's), los cuales cuentan con bibliotecas, herramientas y documentación para poder realizar una aplicación. A continuación se presentan tres de las plataformas de desarrollo para dispositivos móviles

más populares en la actualidad.

2.3.1 Java 2 Micro Edition (J2ME)

J2ME es una arquitectura orientada a dispositivos y sistemas para teléfonos móviles, PDA's y otros productos [32]. Está formada por un conjunto de API's que permite a las aplicaciones desarrolladas explotar las características multiplataforma de *Java* , esto hace que una misma aplicación funcione en otros dispositivos (portabilidad). *Java* maneja varias arquitecturas, para aplicar alguna de éstas depende del tipo de dispositivo y las características del mismo. En la Fig. 2.3 se observa el diagrama de las arquitecturas en forma de bloques, de acuerdo a la familia del dispositivo se toma una u otra opción [33]. El entorno de ejecución *Java* para J2ME debe cumplir los siguientes requisitos: configuración, perfiles y paquetes opcionales. Con la combinación de estos elementos se optimiza la memoria, el poder de procesamiento y las capacidades de E/S de los dispositivos.

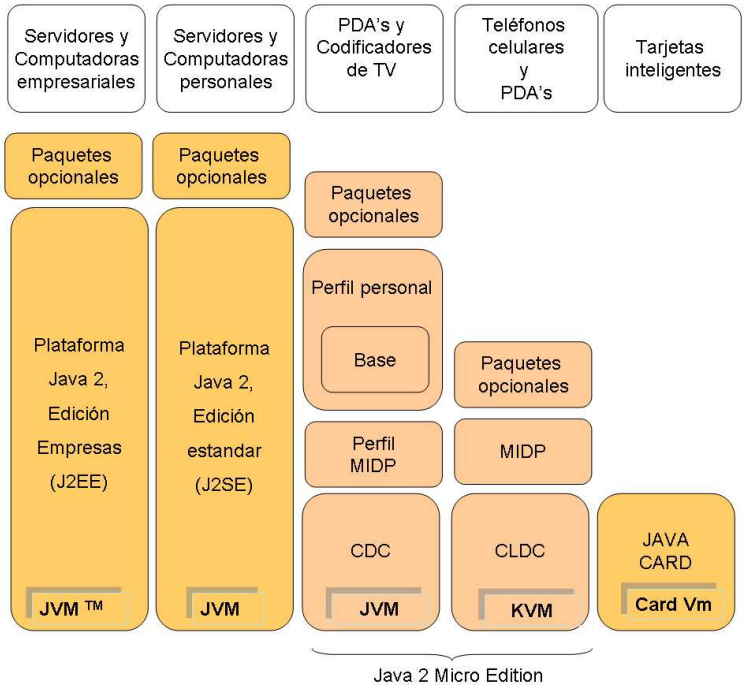

Figura 2.3: Arquitectura J2ME

La configuración se compone de una máquina virtual y un conjunto de bibliotecas. Proporcionan funcionalidades básicas para un entorno de dispositivos que tienen características similares, *e.g.*, gestión de memoria o conectividad a la red. Actualmente existen dos configuraciones J2ME: CLDC (*Connected Limited Device Configuration*) [53], esta configuración esta diseñada para dispositivos con conexiones de red intermitentes, procesadores lentos y memoria limitada. Comúnmente trabajan con dispositivos que tienen CPU's de 16 o 32 bits y con memoria disponible entre 128 y 256 KB para la implementación de la plataforma *Java* y sus aplicaciones asociadas. CDC (Connected Device

Configuration) [34], es la otra configuración diseñada para dispositivos con más memoria, procesadores más rápidos, está orientada a dispositivos con CPU de 32 bits y memoria disponible mínima de 2 MB para la plataforma *Java* y las aplicaciones.

Los perfiles son APIs de un nivel más alto y se combinan con las configuraciones para conformar un entorno de ejecución completo orientado a una categoría de dispositivos. El perfil diseñado para teléfonos móviles y PDAs es el MIDP (*Mobile Information Device Profile*) [54], contiene las funcionalidades básicas para las aplicaciones móviles, incluyendo la interfaz de usuario, conectividad a redes, almacenamiento local de datos y gestión del ciclo de vida de las aplicaciones. Al combinarse con la configuración CLDC, MIDP proporciona un entorno de ejecución *Java* completo, incrementa la capacidad de los dispositivos móviles y reduce el consumo de memoria y energía de los mismos.

Es posible ampliar la plataforma J2ME al combinar varios paquetes opcionales con CLDC y CDC, y sus perfiles correspondientes. Estos paquetes se crearon para solventar algunos requisitos específicos de algún cliente, además ofrecen varios estándares para poder utilizar tanto tecnologías actuales como emergentes, *e.g.*, *bluetooth*, servicios Web, *wireless*, capacidades multimedia o conectividad a bases de datos.

2.3.2 Python para S60

S60 es una plataforma para dispositivos móviles (teléfonos inteligentes o PDAs) que utilizan el sistema operativo *Symbian* [16]. La plataforma consiste de un conjunto de bibliotecas y aplicaciones informáticas estándar, algunas de ellas son telefonía, herramientas de gestión personal y reproductores multimedia. *Python* para *S60* es un lenguaje de programación que puede manejar todas las funciones de un teléfono como cámara, contactos, calendario, grabación, reproducción de audio, entre otras cosas. Python es un lenguaje de código abierto, es administrado por *Python Software Fundation* [57].

Para crear *PyS60* se modificó *Python*, adaptándolo para la serie *S60* del sistema operativo *Symbian*, utilizado por varios dispositivos móviles. Permite acceder a las funciones de los teléfonos llamados "inteligentes", *e.g.*, cámara fotográfica, calendario, grabadora de sonidos, contactos, bluetooth, etc. *Python* es relativamente fácil de usar para programadores de otros lenguajes tales como C, C++, *Java* y *Visual Basic*, en consecuencia, puede ayudar a incrementar la productividad del programador. Es caracterizado como un lenguaje ágil, que promueve el desarrollo de aplicaciones en corto tiempo e incluye un marco de prueba para construir aplicaciones más robustas.

Python es utilizado regularmente en el desarrollo de grandes sistemas para reducir costos de software, además es el lenguaje base de las aplicaciones que son utilizadas por muchas compañias y organizaciones a nivel mundial, debido a esto, la cantidad de usuarios está creciendo a gran velocidad [17]. Sus características importantes son [16]:

- código abierto, razón por la cual es considerado como un buen lenguaje, esta siendo mejorado en su desempeño cada día más.
- lenguaje de alto nivel
- portable, *i.e.*, puede ser utilizado para distintas plataformas (*Windows*, *Linux*, *Mac*, *Windows CE*, *Pocket PC*)

- interpretado, solo basta con ejecutar los programas, no necesita un compilador ni cargar librerías

- orientado a objetos, se puede combinar con otros lenguajes como C y contiene librerías extendidas, *i.e.*, su librería estándar es muy amplia y ofrece soporte para muchas aplicaciones

- contiene librerías que permiten manipular imágenes como el *Python imaging library*

2.3.3 Visual Studio para Windows Mobile

Visual Studio es otra plataforma de desarrollo de aplicaciones *Smart client* [58] (cliente inteligente). Proporciona las herramientas y el marco necesario para desarrollar aplicaciones destinadas a *Pocket PC*, teléfonos *Smartphone* y otras plataformas basadas en *Windows CE*. De manera general, con Visual Studio hay dos formas de desarrollar aplicaciones para dispositivos móviles: aplicaciones Web que se ejecutan en un servidor Web y se presentan en diferentes formatos en varios tipos de dispositivos móviles, que cuentan con un navegador, y aplicaciones cliente enriquecidas bajo *Windows CE*, que se ejecutan en el propio dispositivo, a los cuales se le conoce como "aplicación para dispositivos inteligentes".

Al programar para dispositivos inteligentes se usa el mismo entorno de Visual Studio, que se utiliza para aplicaciones de escritorio, pero existen algunas diferencias:

- se necesitan herramientas adicionales para establecer conexión con un dispositivo remoto y buscar errores en el mismo,

- al seleccionar un tipo y una pantalla de proyecto, al momento de crear un proyecto, se debe seleccionar el dispositivo en el cual se va a ejecutar y depurar la apliación. El dispositivo puede ser físico conectado al equipo de desarrollo, un dispositivo en red o un emulador de dispositivo que se ejecute en el equipo de desarrollo,

- las clases y sus miembros pueden ser diferentes al programar dispositivos móviles. Para saber si una clase y sus miembros están o no disponibles, se debe consultar la documentación de dicha clase.

Es posible utilizar Visual C# o Visual Basic para crear aplicaciones administradas que se ejecuten en .NET *Compact Framework* o bien es posible crear aplicaciones nativas usando Visual C++. Independientemente del lenguaje que se utilice, se puede usar el mismo editor de código, diseñadores e interfaz de depuración que se utilizarían si se programara aplicaciones de escritorio. Visual Studio proporciona herramientas que simplifican el empaquetado de la aplicación y sus recursos en archivos CAB para su implementación en los dispositivos.

Visual Studio proporciona tres lenguajes de programación diferentes y varios tipos de proyectos para el desarrollo de dispositivos móviles, además proporciona plantillas de proyectos para Pocket PC, Windows Mobile Smartphone, Pocket PC y dispositivos de Windows CE. Los lenguajes de programación que se pueden elegir son: Visual C#, Visual Basic y Visual C++.

Visual C# es un lenguaje moderno y orientado a objetos. Las características de la recolección de elementos no utilizados y la compatibilidad con las claves de .NET *Compact Framework* hacen que sea un idioma ideal a la hora de desarrollar aplicaciones móviles confiables y seguras [41]. Visual C# para dispositivos móviles incluye una amplia cantidad de controles para crear de forma rápida una interfaz gráfica y las clases de *Compact Framework* tales como GDI+, XML y servicios Web. También puede llamar a funciones de Windows CE.

El lenguaje Visual Basic para dispositivos móviles, simplifica en gran medida la tarea de trasladar una aplicación de escritorio a un dispositivo móvil o de crear rápidamente una aplicación cliente. También utiliza .NET *Compact Framework*. Los desarrolladores ya familiarizados con Visual Basic, podrán trasladar las aplicaciones existentes o crear otras nuevas de forma muy rápida. Del mismo modo que Visual C#, Visual Basic puede tener acceso a funciones nativas de Windows CE [38].

Visual C++ es el lenguaje de desarrollo que se prefiere para dispositivos móviles al momento de hacer hincapié en el rendimiento o a la hora de desarrollar aplicaciones a nivel de sistema o controladores de dispositivos. No admite .NET *Compact Framework*, pero en su lugar proporciona un subconjunto del conjunto API Win32. Esto se puede utilizar en aplicaciones escritas en código de C# o de Visual Basic para tener acceso a código de C++ que se encuentra en archivos DLL a través de interoperabilidad [58].

2.4 Java Servlets

Dentro del desarrollo de sitios Web, al principio las páginas estaban entrelazadas entre ellas por medio del lenguaje HTML. En la actualidad, los sitios Web incluyen multimedia, aplicaciones de comercio electrónico, y otros tipos de aplicaciones basadas en Web tales como VoIP y transacciones en línea [66]. Además de los avances en contenido, se han producido avances en las tecnologías de servidores Web y tecnologías de creación de contenido dinámico, *e.g.*, desarrollo de servidores de aplicaciones y creación de páginas JSP y ASP respectivamente.

En esta sección se describe de manera general la programación de *Java servlets*, esta tecnología fue usada para crear el sitio Web que se encarga de llevar a cabo el registro de los dispositivos móviles, debido a su arquitectura funcional, fue elegida como la mejor opción para esta tarea.

Los *servlets* son objetos que se ejecutan dentro del contexto de un servidor de *servlets* (*e.g.*, *Tomcat*) o dentro de un servidor de aplicaciones. La palabra *servlet* se deriva de otro concepto, *applet*, el cual se hace referencia a pequenãs aplicaciones que de ejecutan en el contexto de un navegador Web [13]. De manera contraria, un *servlet* es un programa que se ejecuta en un servidor. Los *servlets* se usan regularmente para generar páginas Web de forma dinámica a partir de ciertos parámetros enviados en una petición a través de un navegador de Internet.

Los *servlet* son componentes de un servidor, dichos componentes pueden ser ejecutados en cualquier plataforma o en un servidor, debido a la tecnología *Java* utilizada para implementarlos. Los *servlets* ayudan a incrementar la funcionalidad de una aplicación Web, son cargados de forma dinámica por el entorno de ejecución *Java* del servidor cuando

se necesitan. Al momento de recibir una aplicación del cliente, el servidor Web ejecuta el *servlet* requerido. El *servlet* procesa la petición del cliente y direcciona la respuesta al cliente[11].

La interacción que se genera entre el cliente y el servidor, ambos basados en Web, usa el protocolo HTTP [46], el cual es un protocolo sin estados basado en un modelo de petición y respuesta con varios métodos de petición tales como: GET, POST, HEAD, OPTIONS, PUT, TRACE, DELETE, CONNECT, etc. La mayoría de las aplicaciones Web basadas en *servlets* se construyen bajo el modelo de petición/respuesta HTTP.

Para ejecutar un *servlet* es necesario contar con conocimientos previos de *Java* y HTML, las aplicaciones y las páginas Web que sean iniciadas a través de *servlets* deben ejecutarse en servidores Web con contenedores Web integrados (e.g., *Tomcat*), el API *servlet* debe estar integrado en el servidor Web. Cuando el servidor se ha configurado correctamente, el siguiente paso es programar los *servlets*. En el API *servlet* hay dos paquetes, `javax.servlet`, que contiene las clases del marco de trabajo de los *servlets* y paquetes de protocolos, y `javax.servlet.http` [63], el cual se usa específicamente el protocolo HTTP.

2.4.1 Estructura de un servlet

La estructura de un *servlet* se basa en una interfaz (`Public Interface Servlet`), es una colección vacía de métodos. Los siguientes métodos están declarados en la interfaz `Servlet`:

- `public abstract void init (ServletConfig config) throws ServletException`. Este método se usa para inicializar los parámetros proporcionados por el objeto `ServletConfig`. Es invocado sólo una vez, a menos que el *servlet* sea reiniciado al haberse destruido y vuelto a cargar. En este método regularmente se inicializan los parámetros de configuración como la conexión con una base de datos, inicialización de archivos y variables de entorno.

- `public abstract ServletConfig getServletConfig()`. Este método proporciona el objeto `ServletConfig` para inicializar los parámetros del *servlet*. Se pueden crear parámetros adicionales especificándolos en el archivo `servlet.properties`. Después de que se hayan especificado en este archivo, se puede acceder a ellos usando el objeto ServletConfig.

- `public abstract void service (ServletRequest req, ServletResponse res) throws ServletException, IOException`. Este método es el punto principal del modelo petición/respuesta de protocolo HTTP. Recibe las peticiones del cliente en forma de objeto `ServletRequest`. Los parámetros son enviados junto con el objeto de petición al servidor. La respuesta o resultado del procesamiento de los parámetros se envían al cliente usando el objeto `ServletResponse`.

- `public abstract String getServletInfo()`. Este método se usa para extraer algunos datos del *servlet*, como el autor, versión del *servlet* y alguna otra información referente al *copyright*.

- `public abstract void destroy()`. El método *destroy* se invoca para liberar todos los recursos solicitados como base de datos y otros recursos del servidor. Al igual que el método *init*, este método solo se llama una sola vez.

El ciclo de vida de un *servlet* es especificado en la interfaz `javax.servlet.Servlet`. El servidor Web es el responsable de crear una instancia del *servlet* y de invocar al método `init`. Si un cliente ha enviado una petición al servidor Web, esa petición se pasa al método *servicio* del *servlet* y se envia una respuesta de regreso al servidor Web que posteriormente es direccionada al cliente que hizo la petición. Por último, cuando el *servlet* ha cumplido su propósito, el servidor Web llama al método `destroy`. En la Fig 2.4, se observa el ciclo de vida de un *servlet*.

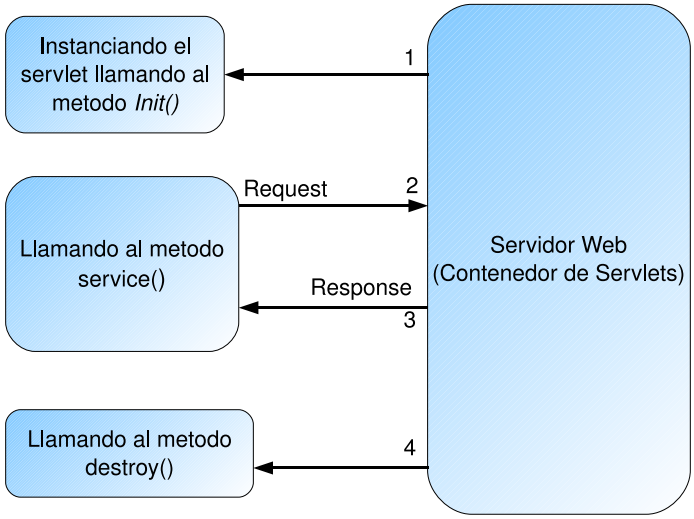

Figura 2.4: Ciclo de vida de un Servlet

2.4.2 Ventajas de los servlets

Los *servlets* agregan un comportamiento dinámico a los servidores. El desarrollo y escritura de un *servlet* es relativamente sencillo de programar, para aplicaciones basadas en Web, sin necesidad de tener que concentrarse en los detalles de bajo nivel de los protocolos HTTP, formatos de petición y encabezados. Todo esto gracias al API de programación. Debido a que los *servlets* tienen un ambiente de trabajo similar a *Java*, son independientes de la plataforma y del servidor, además, pueden ser enlazados con diferentes bases de datos.

Los *servlets* son extensiones de un servidor Web, son objetos *Java* que se cargan dinámicamente por el entorno de ejecución de *Java* (JRE) cuando se necesitan. Cada petición a un *servlet* genera un procesamiento ligero, el cual es controlado por el servidor. Dentro de dicho hilo de lleva a cabo el cambio de contexto y los hilos se pueden pasar de activos a inactivos fácilmente. Un aspecto importante en los *servlets* es el rendimiento al hacer consultas a bases de datos, *i.e.*, es posible almacenar la información de una consulta

en memoria, esto evita los costosos e innecesarios accesos a la base de datos. También cuentan con el manejo de excepciones.

La razón principal del porqué se utilizó esta tecnología basada en *Java* es justificada por su orientación hacia los dispositivos móviles. Debido a que tienen capacidades limitadas de procesamiento, se decidió que el servidor haga el trabajo duro y solo presente los resultados a través de paginas HTML, esto es una buena opción para dar solución al proceso de registro de los dispositivos. En la actualidad la mayoría de los dispositivos móviles son capaces de trabajar bajo el protocolo HTTP, en consecuencia no existe impedimento para poder desarrollar aplicaciones Web basada en modelos de petición/respuesta y con la tecnología *Java* , en este caso los *servlets*.

2.5 Trabajos relacionados

En esta sección se mencionan algunos de los trabajos relacionados más significativos para esta tesis, el primero es una plataforma de un servicio móvil empresarial (*iMobile EE*), es uno de los trabajos pioneros que utiliza dispositivos móviles para comunicarlos entre sí, y proporciona el acceso a contenidos corporativos de una empresa grande como *AT&T*. Esta plataforma fue ampliada para ofrecer servicios multimedia personalizados como la transmisión de video en tiempo real.

Por otro lado, existen varios tipos de aplicaciones para la gestión de dispositivos móviles, uno de los más importantes es *Afaria*. Desarrollado por la compañia de software empresarial *Sybase*, se trara de una aplicación que proporciona una gestión completa y aspectos de seguridad para garantizar que los datos y dispositivos móviles estén actualizados, sean confiables y seguros. Permite tener un buen control para gestionar de forma segura a varios dispositivos móviles, datos, aplicaciones y comunicaciones, independientemente del ancho de banda disponible.

Los mensajes cortos son actualmente uno de los principales elementos en el mundo de las comunicaciones inalámbricas, dentro de este ambiente existe un trabajo que introduce algunos métodos y procesos para desarrollar un sistema de gestión de mensajes cortos utilizando J2ME y el `toolkit` 668i de *Siemens*. El sistema aplica algunas clases para realizar un agrupamiento, listas y edición de mensajes. Con este sistema se pretende incrementar eficientemente la gestión de los mensajes cortos en los teléfonos móviles.

2.5.1 iMobile EE

iMobile EE[19] es una plataforma móvil de servicios empresariales que hace posible la comunicación entre dipositivos móviles con recursos limitados, además, permite que tengan acceso con seguridad a contenidos y servicios corporativos. Esta arquitectura ofrece soluciones en ambientes móviles, algunas de ellas son las siguientes:

- servicios escalables: es capaz de manejar un gran número de solicitudes provenientes de diversas redes inalámbricas y cableadas,
- autentificación: regularmente los usuarios solo tienen acceso a Internet, la plataforma cuenta con una puerta de enlace (*gateway*) para permitir el acceso a la información

de la empresa a través de su Intranet. El servicio de autentificación es indispensable dentro de la arquitectura de la plataforma *iMobile EE*,

- políticas de seguridad: de acuerdo al usuario, se puede dar acceso a los recursos corporativos. Se autoriza el acceso en base a la identificación del usuario, canal de seguridad y políticas corporativas,

- fiabilidad: es capaz de reconfigurarse dinámicamente cuando algunas computadoras fallen o lleguen a sobrecargarse.

Dentro de esta plataforma un dispositivo móvil siempre interactúa con una puerta de enlace *iMobile* para tener acceso a los servicios del sistema. La arquitectura de la plataforma *iMobile EE* está compuesta de varios *gateways* (puertas de enlaces) para cada uno de los servicios que proporciona. Los *gateways* alojan a diferentes controladores o protocolos encargados del envío y la recepción de mensajes, de acuerdo al tipo de servicio que se solicita. Estos *gateways* pueden aumentar dinámicamente dependiendo de la demanda del servicio solicitado. A continuación se describe de forma general la arquitectura del sistema, la cual cuenta con los siguientes elementos:

- puerta de enlace para HTTP/WAP: La puerta de enlace HTTP maneja las solicitudes de servicios HTTP y la autentificación de los usuarios asociados a un dispositivo móvil. Del mismo modo, el *gateway* es compatible con WAP (protocolo de aplicaciones inalámbricas), una especificación abierta que ofrece un método estándar para el acceso a Internet basado en contenidos y servicios desde dispositivos inalámbricos tales como teléfonos móviles y PDA's. El *gateway* esta implementado como un conjunto de *Java Servlets*, bajo un servidor *Jakarta/Tomcat* y un contenedor de aplicaciones Web de *Oracle OC4J*.

- puerta de enlace de e-mail: Permite a los usuarios móviles acceder a los datos corporativos, a través del envío de solicitudes por medio de correos electrónicos. A cada usuario que solicite una dirección de correo electrónico se le debe asignar un identificador en la base de datos antes de que se autorice la solicitud.

- puerta de enlace de mensajería instantánea: Esta puerta de enlace actúa como un cliente de mensajería instantánea, interpreta los mensajes que recibe como solicitudes de servicio y devuelve las respuestas como mensajes. El *gateway* asigna un identificador a cada mensaje instantáneo como un identificador *iMobile* antes de enviar la solicitud del servicio. Funciona de manera similar a un chat, la comunicación es entre el usuario móvil y la plataforma *iMobile*.

- puerta de enlace SMS: Permite a los usuarios de *iMobile* enviar una solicitud de datos a través de mensajes cortos por GSM. El *gateway* SMS responde a la petición del mismo modo, por medio de un mensaje corto.

La arquitectura es flexible, permite añadir nuevos dispositivos móviles y protocolos dentro de su marco de trabajo, sin necesidad de hacer cambios en la lógica operacional del sistema. La plataforma *iMobile* actúa como un agente de la red que permite a los

dispositivos móviles acceder a los servicios personalizados. Los servicios multimedia son parte estos servicios, como: el servicio remoto de grabación de video, de control de cámaras y peticiones de video pre-grabado o de video en tiempo real.

2.5.2 Afaria

La empresa de software *Sybase*, dentro de su conjunto de aplicaciones móviles empresariales, cuenta con un desarrollo llamado *Sybase iAnywhere*, el cual fue reconocido como líder mundial en gestión de dispositivos móviles en el año pasado por las firmas analistas de la industria tecnológica (IDC) [67]. Dentro de este software se encuentra el modulo llamado *Afaria* [36], el cual es capaz de gestionar de forma proactiva y segura a múltiples dispositivos, aplicaciones, datos y comunicaciones de importancia, independientemente del ancho de banda disponible. *Afaria* combina la gestión de los dispositivos móviles y la seguridad desde una consola, ofreciendo la mejor protección contra amenazas a la seguridad. Este módulo proporciona tres funcionalidades principales:

- asignación: Esta etapa es considerada como la fase de configuración adecuada para la gestión de tareas que se implementarán, de acuerdo a las necesidades que se requieran. Se incluye la asignación de pertenencia a un grupo y la configuración de la conectividad del dispositivo. Cuando se trata de seguridad, *Afaria* permite gestionar los requisitos de seguridad de forma centralizada, incluyendo el establecimiento de políticas de seguridad, inicialización de contraseñas, instalación de antivirus, firewall y la configuración de puertos y controles periféricos del dispositivo,

- producción: Durante esta etapa, *Afaria* debe llevar a cabo la administración de manera eficaz, además automatiza las tareas en curso. Algunas de las tareas son la actualización y reparación de software, el seguimiento del rendimiento, mantenimiento y modificación del dispositivo, configuración de la aplicación, así como la distribución y actualización de datos y archivos,

- desmantelamiento: Un dispositivo móvil puede entrar en la fase final del ciclo de vida dentro del sistema por varias razones. El dispositivo se pudo haber perdido, fue robado, se dió de baja o se volvió a dar de alta para otros propósitos diferentes al original. Cuando un dispositivo fue robado o se perdió, *Afaria* permite darlo de baja, de manera remota, desde la consola administrativa.

La arquitectura combina la gestión de dispositivos y de seguridad, para la más amplia gama de plataformas conectadas a una sola red, a través de una interfaz Web. El servidor de *Afaria* ofrece un mantenimiento mínimo y menos costoso. Algunas de las características de la arquitectura son las siguientes:

- se puede integrar sin problemas con bases de datos SQL *Anywhere* de Sybase *iAnywhere*. Es un manejador de base de datos personalizado por la misma empresa que desarrollo este sistema,

- integra tecnologías de acceso a directorios, *e.g.*, LDAP (Protocolo Ligero de Acceso a Directorios); el cual permite el acceso a un servicio de directorios ordenados y distribuidos para buscar información en el entorno de red,

- tiene soporte para bases de datos como *Microsoft SQL Server*, *SQL Anywhere* y *Oracle*,
- la consola basada en Web es compatible con las tecnologías .NET,
- varias opciones de conectividad son soportadas para operar con entornos móviles, tales como TCP/IP a través de una conexión inalámbrica o conexión por cable,
- soporta los protocolos HTTP y HTTPS para permitir a los usuarios conectarse a través de Internet y así tener un punto de acceso seguro por medio del *firewall* de la empresa,
- el servidor *Afaria* se ejecuta en *Windows Server 2003 Standar* o *Enterprise*,
- proporciona una mayor seguridad dentro de la Intranet de la empresa.

El sistema es capaz soportar a varios dispositivos, tales como:

- laptops basadas en sistemas operativos *Win32*,
- dispositvos con *Windows CE/Pocket PC*,
- dispositivos con *Windows Mobile Profesional*,
- dispositivos con *Windows Mobile Estándar*,
- dispositivos con *Symbian E serie 60 3rd edición* y *N series*,
- dispositivos con *PalmOS versión 5*,
- dispositivos *BlackBerry RIM*

2.5.3 Gestión de mensajes cortos SMM

El sistema de gestión de mensajes cortos basado en J2ME (SMM)[43] fue desarrollado con la *API* de *Siemens*, el resultado fue un *midlet* que permite al usuario gestionar los mensajes cortos en su teléfono móvil de manera eficiente. La función principal del sistema se encarga de dividir los mensajes cortos en grupos, editar, insertar y eliminar operaciones sobre la lista de mensajes y también sobre el contenido de los mismos.

La herramientas de desarrollo principal que se utilizó fue *J2ME*, la cual incluye una serie de *plugins* tales como: *SUN Wireless Toolkit*, *UltraEdit*, varios emuladores, *Builder*, *Mobile Set* y *Nokia SDK*. Se desarrolló en el editor *Jcreator* y *MotoSDK*, ademas se usó el *toolkit* de *Siemens SMTK6688i*. El sistema se compone de 12 clases que realizan la gestión de los mensajes cortos y algunas otras operaciones. Básicamente el usuario puede usar los grupos que el sistema establece de acuerdo a la lista de mensajes, también puede borrar los grupos, crear nuevos grupos de mensajes y renombrarlos.

Otras de las funciones del sistema se realizan dependiendo del contenido de los mensajes o del teléfono móvil, en otras palabras el usuario puede editar, borrar y modificar; en cuanto al teléfono móvil, cuando se agrega un mensaje en la lista también se agrega el

número del teléfono. De acuerdo a esto los mensajes quedan clasificados principalmente por el contenido y así el usuario sabe a qué números generalmente envía ese tipo de mensajes. En la Fig 2.5 se puede observar la estructura de cada una de las funciones del sistema de gestión de mensajes cortos.

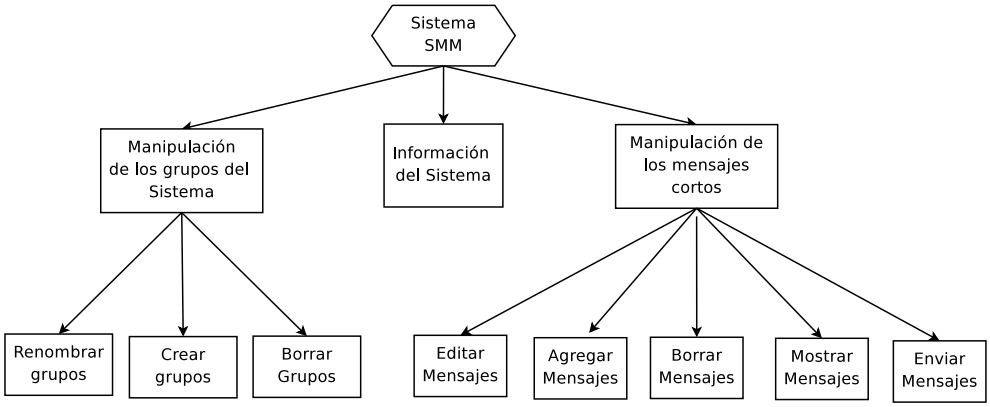

Figura 2.5: Estructura del sistema SMM

Este trabajo es un buen referente, ya que en él se maneja una organización de los mensajes por grupos parecida a la clasificación que se elaboró en la presente tesis. La diferencia radica en el propósito de la organización de los grupos, en el trabajo citado el objetivo parte del contenido de los mensajes, en el presente trabajo terminal el objetivo se centra en los destinatarios.

Capítulo 3

Diseño del servicio SEGEMENS

En este capítulo se describe un panorama general de los sistemas distribuidos, en especial se hace énfasis en la arquitectura cliente-servidor y en los mecanismos de comunicación basados en eventos. También se proporciona una vista general de los ambientes heterogéneos donde las tecnologías en telecomunicación están presentes. De acuerdo a la propuesta de solución se presenta el análisis que se llevó a cabo para seleccionar las herramientas que se usaron en el desarrollo del servicio, además se presenta el diseño de las aplicaciones y los componentes de SEGEMENS.

3.1 Sistemas Distribuidos

Desde los inicios de la computación, se han presentado muchos cambios, desde las enormes computadoras que solo permitían realizar un número limitado de tareas y que eran usadas exclusivamente por pocas organizaciones, hasta las computadoras de escritorio y portátiles con capacidades muy superiores a las primeras, este aumento en capacidades ha provocado que las computadoras sean usadas cada vez más en las actividades cotidianas de las personas, principalmente los dispositivos móviles tales como PDA's, teléfonos celulares y *Smartphones*. El uso de dispositivos de cómputo se ha hecho posible gracias a dos factores básicos:

- desarrollo de microprocesadores: se redujo su tamaño y el costo de los mismos, además se incrementaron sus capacidades
- el desarrollo de las redes: con la aparición de las redes de área local y las telecomunicaciones se permitió conectar varias computadoras con el objetivo de transferir e intercambiar datos

Bajo este contexto surge el concepto de "Sistemas Distribuidos" el cual es muy popular en la actualidad, debido a los campos de estudio en los cuales se fundamenta, principalmente en redes, *e.g.*, Internet, redes de telefonía celular, redes inalámbricas, entre otras [21]. Existen varias definiciones de los sistemas distribuidos:

- "Un sistema distribuido es una colección de computadoras autónomas que trabajan conjuntamente para dar la apariencia de un sistema coherente único." [23]

- "Un sistema distribuido es aquel en el que los componentes de hardware o software, están localizados en computadoras conectados mediante una red, comunican y coordinan sus acciones sólo mediante paso de mensajes." [21]
- "Un sistema distribuido es una colección de dispositivos individuales de computación que pueden comunicarse unos con otros." [25]
- "Un sistema distribuido es un sistema compuesto de varias computadoras que se comunican mediante una red, almacenando procesos que utilizan un conjunto de protocolos distribuidos para soportar la ejecución coherente de actividades distribuidas." [26]
- "Un sistema distribuido es un sistema de procesamiento de información que contiene múltiples computadoras independientes, que cooperan unas con otras sobre una red de comunicaciones para alcanzar un objetivo especifico". [27]

En los conceptos anteriormente mencionados se pueden observar algunos elementos en común, con éstos podría establecer una definición más completa resaltando las siguientes características [31]:

- Varias computadoras o procesos, denominados nodos.
- Red de comunicaciones o de computadoras, denominada simplemente red.
- Intercambio de mensajes (coordinación).
- Objetivo en común, *e.g.*, compartir un recurso, brindar un servicio, o ejecutar una aplicación.
- Apariencia de un sistema único.

Las aplicaciones distribuidas ejecutan parte de su código en distintas computadoras, las cuales coordinan la ejecución mendiante mensajes. Hoy en día, la computación distribuida se ha afianzado como la base para el desarrollo de tecnologías *e.g.*, *clusters*, *grids*, comunicaciones punto a punto (P2P) y redes para dispositivos móviles. Para poder desarrollar un sistema distribuido se deben tener en cuenta las siguientes características [64]:

- Heterogeneidad: Pueden interactuar distintos dispositivos (computadoras, PDAs, *Smartphones*, etc) con distintos sistemas operativos y distintos lenguajes de programación para el desarrollo de aplicaciones.
- Seguridad: Puede existir información compartida, valiosa para los usuarios, se pueden usar técnicas de cifrado para que siga siendo lo más confidencial posible.
- Escalabilidad: Se debe poder añadir nuevos recursos, usuarios, etc; sin que se pierda el rendimiento y el control; los cuellos de botella que se generan, a la larga, disminuyen el rendimiento del sistema.
- Fallos: El sistema debe ser tolerante a fallos y ser capaz de recuperarse completamente cuando el fallo se haya solventado.

- Concurrencia: Debe permitir que varios usuarios puedan acceder de manera simultánea a un recurso y tomar otros recursos sin ocasionar errores en los resultados.

- Transparencia: Algunos aspectos del sistema están ocultos a las aplicaciones, *i.e.*, acceder a un recurso sin saber su ubicación.

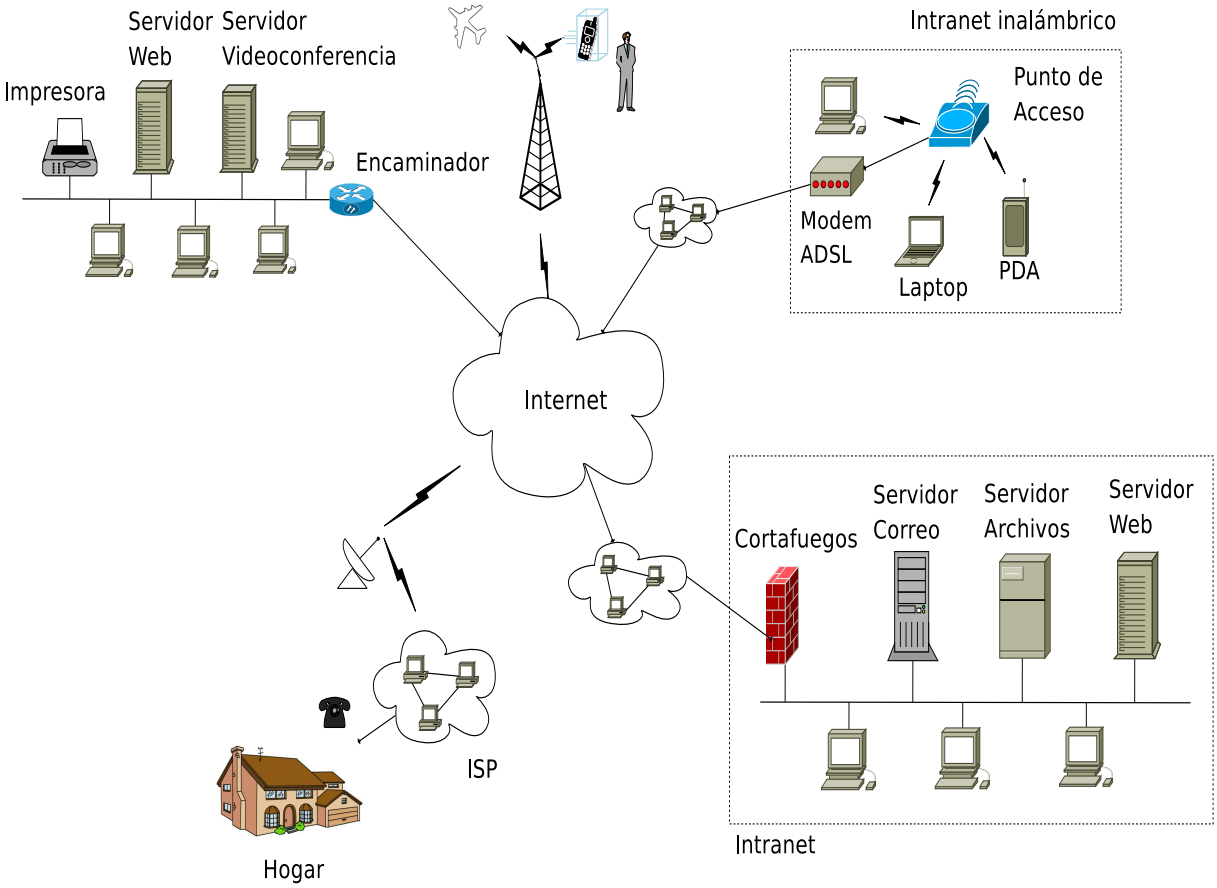

Figura 3.1: Sistemas Distribuidos

En la Fig 3.1 se pueden observar varios sistemas distribuidos, *e.g.*, el servidor Web que esta dentro de la *Intranet* puede dar servicio tanto a los usuarios que estén dentro de la organización, como a los que están fuera de ella, puede ser que una persona se conecta desde su casa a través de un proveedor de Internet (ISP, Servicio Proveedor de Internet). También se tienen servicios tales como el correo electrónico, transferencias de archivos, servicios multimedia y de impresión, los cuales están a disposición tanto de los usuarios externos como internos. El desarrollo de la tecnología inalámbrica permite que varios dispositivos móviles (*laptops*, PDA's, teléfonos celulares) sean capaces de conectar a los usuarios mientras se están moviendo.

3.1.1 Arquitectura de comunicación cliente-servidor

El principal objetivo, que se sigue al construir un sistema distribuido, es compartir recursos, se debe diferenciar entre los componentes que tienen los recursos y los componentes que solicitan o requieren el recurso. En el modelo cliente-servidor hay dos tipos de procesos, los clientes son procesos que hacen peticiones de servicio y los servidores proveen esos servicios. La tarea de administrar tales recursos la llevan a cabo los servidores, los cuales por medio de aplicaciones ponen a disposición los recursos u otros servicios, *e.g.*, un servidor Web gestiona un conjunto de páginas Web, los clientes acceden a éstas a través de navegadores de Internet (*browsers*). En estos casos es recomendable que el trabajo pesado lo realice el servidor, *i.e.*, los tipos de procesamiento como las consultas a bases de datos, cálculos estadísticos, tratamiento de imágenes, simulaciones, etc.

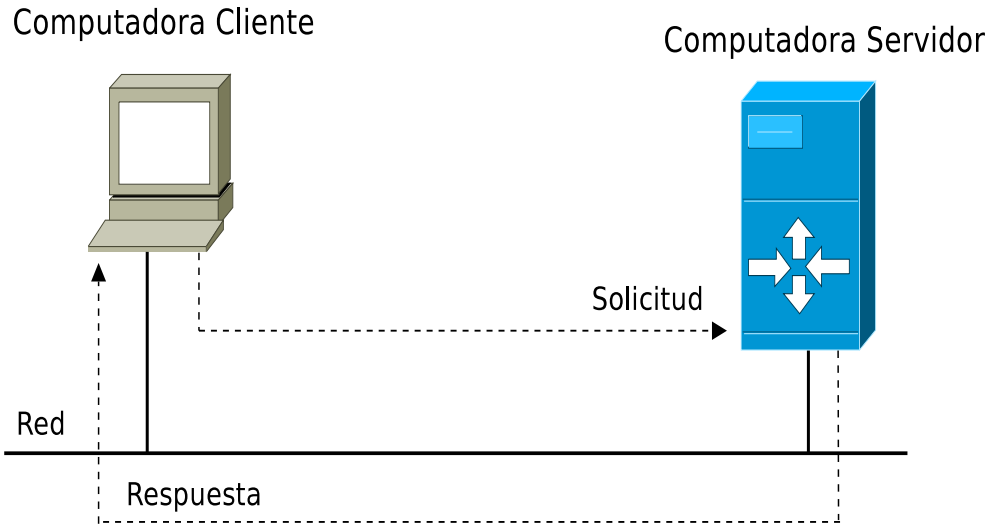

Figura 3.2: Arquitectura Cliente-Servidor

En la Fig 3.2 se muestra la interacción entre el cliente y el servidor. Cuando el cliente necesita un recurso o solicita un servicio, envía un mensaje de petición al servidor. Del otro lado, el servidor está a la espera para atender alguna petición. Al recibir la solicitud realiza un procesamiento para responder al cliente, posteriormente devuelve el resultado a través de un mensaje de respuesta. Existen dos maneras de obtener los servicios que ofrece un servidor, los mecanismos son *Push* y *Pull* [23]:

- El modo *Push* se presenta cuando el servidor toma la iniciativa de enviar la información a clientes ya determinados. Un ejemplo práctico son los correos electrónicos, los cuales envían los mensajes o avisos a los usuarios.

- El modo *Pull* se presenta cuando el cliente toma la iniciativa de hacer una petición al servidor y éste le responde, *i.e.* el cliente actúa mediante un mecanismo de *polling* (de consulta periódica). Este es el modelo de operación tradicional, un ejemplo de una aplicación que usa el método *Pull* es un sitio Web que proporciona el estado del tiempo, un cliente se conecta a la página, solicita la información y el servidor responde el pronóstico ambiental.

Una de las ventajas del modelo cliente-servidor es que permite la aplicación de concurrencia, *i.e.*, proporciona servicios de manera simultánea a múltiples clientes. De esta manera el servidor no espera a que finalice de atender a un cliente para atender a otro. Aunque se trata de dos elementos (cliente y servidor), el servicio o recurso está centralizado en un solo servidor. Esto podría ocasionar cuellos de botella al conectarse más clientes o al producirse alguna falla. En la Fig 3.3 se observa la capacidad que puede adoptar un servidor para atender a varios clientes.

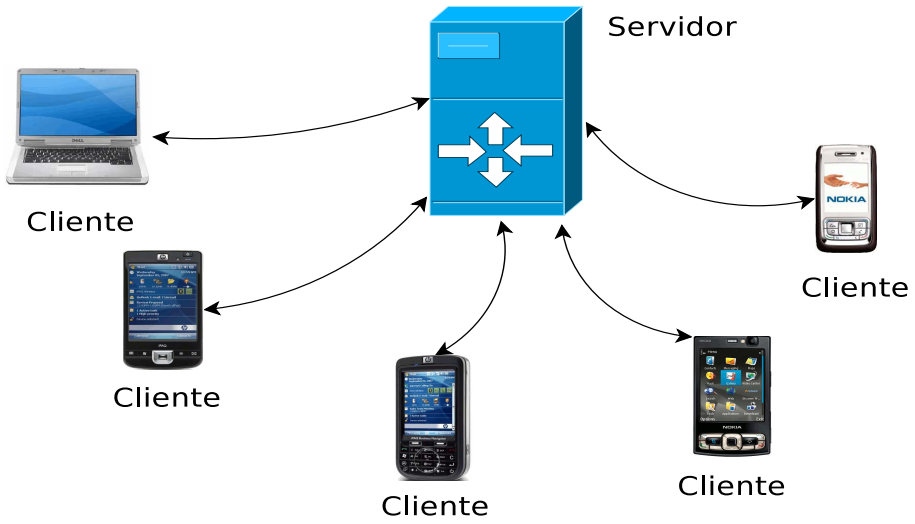

Figura 3.3: Arquitectura cliente-servidor con múltiples clientes

3.1.2 Comunicación basada en eventos

La notificación basada en eventos es un paradigma de comunicación usado en llamadas a procedimientos remotos, en componentes distribuidos y en sistemas orientados a servicios [45]. Desde el punto de vista informático, un evento indica la aparición de un estímulo que puede disparar una transición de estados (un objeto pasará de un estado a otro). Es la especificación de un acontecimiento significativo, como una señal recibida, un cambio de estado o el simple paso de un intervalo de tiempo [37]. Desde el contexto de programación, un evento es un vínculo entre una ocurrencia en el sistema y una porción de código que responde a esa ocurrencia. Esa parte de código es lo que se denomina como manejador de eventos. Los eventos pueden ser provocados por el usuario, por el sistema o por el medio, algunos ejemplos se describen a continuación:

Los eventos provocados por el usuario pueden ser:

1. un clic del mouse,
2. oprimir una tecla,
3. seleccionar un menú,
4. mover el mouse.

Los eventos provocados por un sistema o el medio pueden ser:

1. Los intervalos del reloj,
2. el paso del tiempo,
3. el cierre del sistema,
4. un mensaje que proviene de otra aplicación.

El objetivo de desarrollar sistemas basados en eventos es principalmente la comunicación asíncrona, *i.e.*, durante la ejecución del sistema basado en eventos, mientras los usuarios comparten el espacio de trabajo, pueden aparecer eventos que ejecuten ciertas funciones en las aplicaciones de los usuarios [42].

Otra característica de este tipo de sistemas es que los eventos son condiciones o acciones que se observan por el propio sistema, pero provienen desde afuera del control del mismo. Los eventos son previstos, pero no son planeados; *i.e.*, se conoce le hecho de que el evento puede ocurrir, pero tal vez no se conozcan las circunstancias o el momento en que ocurrirá. A continuación, en la Fig 3.4, se muestra un diagrama de estados de una aplicación simple utilizando programación basada en eventos:

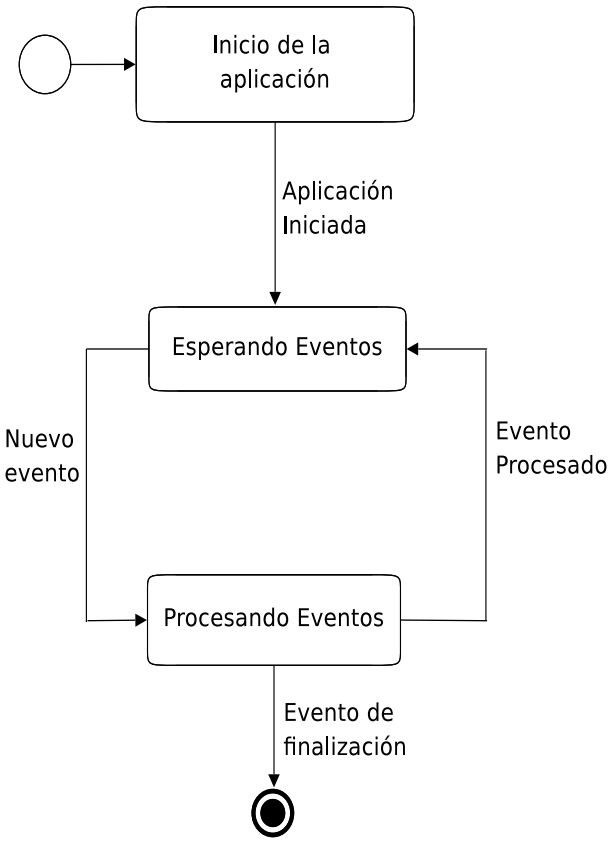

Figura 3.4: Aplicación simple basada en eventos

3.2 Sistemas Heterogéneos

Un sistema heterogéneo está compuesto por hardware con características físicas distintas, y software con características operativas distintas, pero que se pueden comunicar utilizando medios comunes [61]. Actualmente se tiene una gran variedad de medios y dispositivos interconectados, software para comunicación y sistemas operativos con diferentes características, los cuales ocasionan que las redes de comunicación no posean una base homogénea. Existen dos tipos de heterogeneidad [29]:

1. Heterogeneidad de conexíon: significa que los usuarios pueden conectarse a una red sin especificar el tipo de conexión, *i.e.*, pueden conectarse de dos maneras inalámbrica y alámbricamente.

2. Heterogeneidad de dispositivo: es la capacidad que tiene un sistema o servicio de atender a un usuario desde diferentes tipos de dispositivos, *e.g.*, una PC, *Laptop*, PDA o *Smartphone*.

3.2.1 Ambientes heterogéneos

Un ambiente heterogéneo es constituido por sistemas distribuidos que tiene la capacidad de establecer una comunicación entre sí, a través de una red alámbrica o inalámbrica. Además se cuenta con la característica de atender a los usuarios sin importar el dispositivo con el cual accedan al sistema.

Al incorporar tecnología de red inalámbrica en este tipo de ambientes se incrementa las opciones para desarrollar aplicaciones que sean capaces de ejecutarse en dispositivos móviles con distintas plataformas. Debido a la heterogeneidad también pueden ser incorporados los dispositivos fijos. El resultado de esta fusión genera un ambiente móvil y heterogéneo.

3.3 Análisis del servicio

Las organizaciones y empresas de la actualidad para poder sobrevivir en los ambientes competitivos de negocios, académicos u otros; deben tener la habilidad de manejar distintas tecnologías, para mejorar los servicios que ofrecen a sus clientes o usuarios. El concepto de movilidad se ha hecho muy popular hoy en día, las soluciones móviles permiten la operatividad remota y la gestión eficiente de la información, aumentando el rendimiento de las organizaciones.

Durante la última década se ha visto un aumento en las investigaciones en el área de las comunicaciones inalámbricas, debido a la demanda de conectividad. Esta última inicialmente fue impulsada por la telefonía celular, pero actualmente está siendo opacada por el desarrollo de aplicaciones para el manejo de datos sobre tecnología inalámbrica [24]. Muchas organizaciones han optado por introducir soluciones simples y útiles para el trabajo móvil que se ha generado, debido a la creciente demanda de movilidad, creando nuevos retos dentro de las mismas organizaciones. Para poder hacer uso de la tecnología inalámbrica es necesario contar con los dispositivos adecuados, éstos deben ser capaces

de dar el soporte a los servicios que se desean proporcionar a los usuarios. La utilización de dispositivos móviles de tipo PDA's y *Smartphones* en las organizaciones han escalado posiciones en la jerárquia de soluciones de adminsitración de datos; debido principalmente a la comodidad de transportarlos, sus interfaces son más sencillas respecto a una PC, así como también la posibilidad de conectarse inalámbricamente a redes de datos; con todo esto los dispositivos móviles son de gran utilidad y poseen cada vez más prestaciones [39].

Multimedia es un término que se aplica a cualquier objeto que usa simultáneamente diferentes formas de contenido informativo como texto, sonido, imágenes, animación y video (ver Fig 3.5); para informar o entretener al usuario [56]. Este concepto es usado en las redes de distribución de contenidos, las cuales son un conjunto de tecnologías diseñadas para apoyar el transporte efectivo de contenido a través de una red hacia un dispositivo final [9].

Figura 3.5: Información multimedia

Con la evolución de las redes móviles e inalámbricas, y la modernización de los distintos dispositivos móviles, ha surgido la necesidad de desarrollar aplicaciones capaces de combinar ambas tecnologías, con la finalidad de contar con servicios útilies, los cuales deben tener un rol importante en la forma de comunicarnos.

En la actualidad existen varios tipos de dispositivos móviles, que pueden ser utilizados para la comunicación interna o externa en una organización, dentro de éstas se debe tener un control adecuado de los dispositivos que van a usarse y analizar los servicios que se les puede proporcionar, debido a que algunos tienen más capacidades que otros. Entonces es importante contar con un registro de los dispositivos que se encargue, además de su registro, de conocer las características de cada uno de ellos. Esta es una tarea con un grado de complejidad especial, si tener un almacén de datos con la información de cada computadora fija, es un problema complejo por sí mismo, con la característica de móvilidad se presenta un aspecto más a tomarse en cuenta. Además es importante que el registro se lleve de forma automática, *i.e.*, sin la intervención manual.

Teniendo el conocimiento de las capacidades de cada dispositivo, imaginemos una aplicación que se instala en varios dispositivos heterogéneos y mediante un procedimiento de

sincronización actualiza la información en el dispositivo. Dicha aplicación se basa en una arquitectura cliente-servidor.

Un ejemplo de este tipo de aplicaciones se describe en el siguiente escenario: Dentro de una organización es necesario envíar datos en diferentes formatos (texto, imágenes, sonido, video) y se pretende envíar la información hacia dispositivos móviles.

Dentro de la organización existen diferentes tipos de usuarios los cuales, de acuerdo a su dispositivo, podrán recibir toda o solo cierta información. La forma de distribución de la información es provocada por la ocurrencia de ciertos eventos que pudieran ser disparados por el administrador del servicio o por el propio sistema. *e.g.*, en la sala de un aeropuerto el pasajero puede ser orientado para realizar todos los procedimientos protocolarios que se llevan a cabo para abordar el avión. La detección de la conexión del usuario dispararía el primer evento que tal vez sería un flujo de actividades el cual se envía al dispositivo a través de un mensaje de texto.

Otro caso sería que el servicio este monitoreando el tiempo de holgura que le queda al pasajero para abordar el avión y mediante avisos proporcionar el tiempo que le resta para realizar todas las actividades previas al abordaje. Posiblemente el usuario también reciba información a cerca del viaje, los servicios que se proporcionan dentro del avión, información del destino, etc; todo esto a través de un video que recibirá en su dispositivo.

Cabe destacar que de acuerdo al tipo de boleto que pago el cliente se le proporcionaran los servicios de información, además se debe de contar con el dispositivo adecuado para recibir los datos sin problemas.

En el escenario descrito anteriormente se presentan dos casos principales, que requieren de cierto análisis. En el primero es necesario llevar a cabo un registro del usuario, editar un perfil, registrar el dispositivo, etc. Estas actividades se pueden realizar automáticamente o por parte del mismo usuario, entonces es necesario desarrollar una aplicación que sea capaz de proporcionar una interfaz amigable y fácil de manejar para realizar dicho registro.

De acuerdo al perfil especificado del usuario se tiene que hacer un filtro del tipo de información que se enviará al cliente, este sería el segundo caso que debe tratarse, es que no todos los usuario tienen los mismos privilegios, además dependen del soporte del dispositivo para recibir la información correctamente. Como se puede observar hay circunstancias que se deben tomar en cuenta, específicamente la heterogeneidad de los dispositivos y los mecanismos de organización de la información que será proporcionada. En la Fig 3.6 se puede observar los tipos de información multimedia que el servidor proporciona a los dispositivos, dependiendo de las característica y la funcionalidad de los mismos.

Desarrollar una aplicación que pueda ejecutarse en distintos dispositivos móviles es una tarea difícil, además para proporcionar al usuario un entorno donde pueda editar su perfil, se podría pensar en ambientes Web para dar solución a una parte del problema. La organización y distribución de la información es otro tema a tratar, desarrollando un servidor con las funciones necesarias para realizar un filtrado adecuado de datos y la gestión de eventos, apoyándose en lenguajes de programación, ayudarían a tener un servicio robusto y aceptable.

Figura 3.6: Dispositivos con diferentes soportes de servicios multimedia

3.4 Arquitectura general del servicio

De acuerdo al análisis descrito anteriormente se propone como solución, un Servicio de Comunicación por Eventos con Registro Implícito de Dispositivos, que además de realizar un registro de usuarios y de los distintos dispositivos, proporciona un servicio de notificación de mensajes cortos que se encarga de generar avisos de acuerdo a ciertos eventos. La arquitectura general del servicio se puede apreciar en la Fig. 3.7.

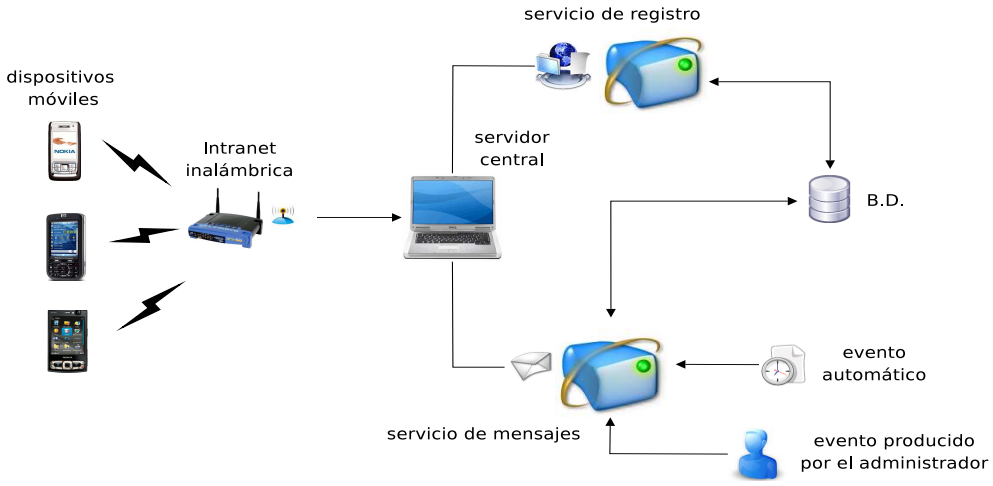

Figura 3.7: Arquitectura general

A continuación se describirá los elementos de la arquitectura general:

- dispositivos móviles: a través de una aplicación, los dispositivos recepcionan y visualizan en pantalla el mensaje enviado por el servidor de mensajes. Por medio del navegador de Internet, que ya tienen instalado los dispositivos, los usuarios pueden acceder al servicio de registro basado en Web para crear una cuenta de usuario y editar su perfil,
- intranet inalámbrica: es una red privada local, por medio de ella se realiza la conexión tanto al sitio Web como al servidor de mensajes,
- servidor central: es el equipo de cómputo donde se encuentran alojados el servicio de registro y el servicio de mensajes,
- servicio de registro: mediante una interfaz Web, el usuario va a registrarse en el servicio, se debe realizar todo un procedimiento para la edición de un perfil, el cual ayuda en la selección de los tipos de información que el usuario puede recibir en su dispositivo. Este componente también permite registrar a los dispositivos a través de la misma interfaz Web, esta tarea se realiza de forma automática y transparente para el usuario,
- servicio de mensajes cortos: permite envíar avisos a través de mensajes, los usuarios reciben la información adecuada gracias al proceso de filtrado que se realiza de acuerdo al perfil editado por el mismo usuario,
- evento automático: el servicio de mensajes es capaz de producir un evento de manera automática, el cual se dispara diariamente para enviar un mensaje a los usuarios,
- evento producido por el administrador: en este caso lo eventos son producidos manuálmente por el administrador del servicio de mensajes, cada evento producido representa el envío de algún mensaje,
- B.D.: Se cuenta una base de datos donde se almacenan todos los registros de los usuarios incluyendo su perfil y también los registros de los dispositivos que usa.

3.5 Diseño de las aplicaciones e interfaces del servicio

Para el desarrollo de cualquier sistema, es necesario elegir un lenguaje de programación que sea adecuado para el mismo [30]. Hay muchas opciones en cuanto a lenguajes de programación, desde lenguajes tipo ensamblador hasta los lenguajes visuales. Algunos de los más utilizados son: *Java* [35], *C*, *C++*, *Python* [17], *Visual Basic*, *Delphi*, *HTML*, etc.

La rápida evolución de los sistemas inalámbricos, en particular las redes celulares, que incorporan conceptos de comunicación; han originado un nuevo enfoque para el desarrollo de sistemas de aplicaciones de datos [24], principalmente en el desarrollo de programas para dispositivos móviles.

Es importante tomar en cuenta que la tecnología usada en la fabricación de dispositivos móviles avanza muy rápido, por lo tanto, los lenguajes de programación deben

proporcionar más herramientas para aprovechar al máximo las características de los dispositivos. Entonces para el desarrollo de aplicaciones móviles es necesario que el lenguaje de programación sea una versión más ligera, comparada con los lenguajes utilizados en computadoras de escritorio o *Laptops*.

Para elegir el lenguaje de programación que se utilizó en el desarrollo del servicio se llevo a cabo un análisis de requerimientos, el cual nos indicaría cual sería la mejor opción. Las puntos principales que aborda el servicio desarrollado en esta tesis se describen a continuación:

- el servicio cuenta con un registro de usuarios el cual se realiza a través de una aplicación Web,
- se ejecuta en distintos sistemas operativos: *Windows*, *Linux (Ubuntu)*, *Windows Mobile* y *Symbian*,
- soporte de comunicaciones basado en conexiones TCP (*sockets*),
- soporte en el desarrollo de interfaces,
- soporte para el desarrollo de aplicaciones multihilos,
- la aplicación es portable, se debe ejecutar sin problemas en los distintos dispositivos,
- los dispositivos móviles tienen capacidades limitadas en comparación con una PC o *Laptop*.

Lenguaje	Características					
	Versión para Móviles	Portabilidad	Soporte Multihilo	Soporte Sockets	Diseño de Interfaces	Diseño Web
Java	si cuenta	es portable, gracias a la maquina virtual	si soporta	si soporta	fácil de crear pero dificil de implementar	cuenta con tecnologías como jsp y servlets
Python	si cuenta	se portable, sus funciones básicas se ejecutan sin modificar el código	si soporta	si soporta	relativamente sencillas de crear e implementar	cuenta con un *framework* llamado Django
C y C++	si cuenta	es portable,	si soporta	si soporta	difíciles de crear e implementar	no da soporte para diseño Web

Tabla 3.1: Cuadro comparativo de lenguajes de programación

En el cuadro 3.1 se observa una comparativa de los tres lenguajes de programación más populares y que cuentan con versiones para el desarrollo de aplicaciones móviles.

Se eligió *Java* porque cumple con los requerimientos necesarios para el desarrollo del servicio, principalmente por el soporte para el diseño Web, además contiene clases que ayudan al desarrollo de interfaces gráficas. Gracias a la máquina virtual para dispositivos móviles, la aplicación es portable, el mismo código se ejecuta en varios dispositivos sin problemas.

3.5.1 Sitio Web basado en servlets

Básicamente la aplicación que se encarga de llevar a cabo el registro de los dispositivos, está desarrollada en *Java servlets*. Se eligió esta tecnología por la funcionalidad que ofrece, *i.e.*, todo se procesa y se ejecuta dentro un servidor, y los dispositivos solo reciben la respuesta que se solicitó en forma de páginas Web. Por lo tanto, en los dispositivos no se lleva a cabo ningún procesamiento, solo la visualización de los resultados en formato *HTML*. Esto es un punto a favor ya que se trabaja con dispositivos móviles con procesamiento limitado.

La mayoría de los dispositivos móviles de última generación cuentan con interfaces de red inalámbrica, por lo tanto también tienen un navegador Web, además soportan el protocolo HTTP para la transferencia de datos a través de la red. Sabiendo esto se procedió a desarrollar un sitio Web que se encarga de registrar a los dispositivos móviles.

Tomando en cuenta que el sitio Web tiene que visualizarse en dispositivos móviles, se debe analizar las dimensiones que debe tener la interfaz. Los tamaños de las pantallas de los dispositivos móviles son pequeños, por lo tanto el diseño de la aplicación debe adaptarse al tamaño de la pantalla, además se cuenta con diferentes navegadores los cuales no soportan algunas tecnologías como *javascript* o animaciones flash.

La entrada de datos a través de un sitio Web orientado a dispositivos móviles es un aspecto que se debe pensar muy bien. En una computadora de escritorio o *Laptop* se cuenta con teclados de 102 teclas para introducir datos, en un dispositivo móvil el manejo del teclado es más complicado, entonces si el usuario va a introducir datos, la solicitud de los datos debe ser lo más sencilla que se pueda.

La presente tesis esta enfocada a diferentes dispositivos móviles pero que cuentan con interfaces inalámbricas. El servicio de registro de usuarios y dispositivos móviles almacena y recupera información de una base de datos, las consultas e inserciones son realizadas por los *servlets* alojados en el servidor, de tal modo que no se realiza ningún procesamiento del lado de los dispositivos móviles.

3.5.2 Aplicación basada en J2ME

Para mejorar las aplicaciones orientadas a dispositivos móviles se han propuesto varios lenguajes de programación. El desarrollo de software para teléfonos móviles inicialmente sólo se centró en los servicios básicos de voz, despues se fueron agregando más servicios, entonces surgió la necesidad de crear funciones para tener acceso a los nuevos servicios a través de redes inalámbricas o por medio de Intenet [43].

J2ME es un lenguaje desarrollado por SUN *MicroSystems* con el fin de aplicar *Java* en dispositivos de comunicación móvil, dispositivos empotrados y aparatos electrodomésticos. El enfoque de solución de este lenguaje se basa en dos puntos: (1) *Java* se ejecuta en diferentes plataformas, en consecuencia las aplicaciones *Java* eventualmente pueden instalarse en dispositivos móviles; (2) *J2ME* ofrece soporte para la programación sobre protocolos de Internet tales como HTTP, TCP y UDP [35].

Actualmente, en las comunicaciones móviles, *J2ME* es visto como un esquema de solución debido a que ofrece múltiples servicios. Algunos de los principales son: información móvil, transacciones bancarias, comercio electrónico, tratamiento médico móvil,

servicios multimedia, mensajería, etc, los cuales pueden desarrollarse con la plataforma *J2ME*.

El soporte para la programación multihilos de la plataforma *J2ME* permite realizar aplicaciones basadas en esquemas de solicitud-respuesta, *e.g.*, cliente-servidor y arquitecturas punto a punto (P2P). La ventaja de usar *Java*, sobre otras herramientas para el desarrollo de aplicaciones enfocadas a dispositivos móviles es la portabilidad. Una aplicación escrita con API's MIDP de *J2ME* puede ser portable con cualquier dispositivo MIDP, esta característica esta presente en la mayoría de dispositivos móviles actuales [32].

En esta tesis desarrollamos aplicaciones con *J2ME* de tipo cliente, su función principal es conectarse al servidor de mensajes. Aprovechando el soporte de programación multihilo, la aplicación cliente permanece con la conexión abierta, esperando los mensajes que el servidor pudiera enviar.

3.5.3 Interfaces de las aplicaciones

El servicio desarrollado en la presente tesis cuenta con tres aplicaciones: la aplicación servidor, la aplicación cliente y el sitio Web. Para el diseño de interfaces gráficas en *Java* existen varias herramientas, una de las más populares es la biblioteca gráfica *Swing* [47], que proporciona componentes como botones, tablas, marcos, etc; para crear una interfaz gráfica de usuario (GUI). La clase *Swing* es fácil de usar debido a que existe una gran cantidad de documentación. Una de las características princpales de *Swing* es que hereda el manejo de eventos del API *AWT* (*Abstract Window Toolkit*), *i.e.*, permite que un componente de la aplicación reaccione a eventos del teclado, el mouse, la ventana, etc. No es necesario instalar el paquete *Swing*, viene incluido en el *Java* JDK en la version *Estandar Edition*. El paquete *Swing* existe desde el JDK 1.1.

Una aplicación *Swing*, se construye usando componentes bajo las siguientes reglas:

- debe existir un contenedor principal (contenedor de alto nivel o *Top-Level Container*), que se encarga de proveer el soporte que los componentes *Swing* necesitan para dibujar la interfaz y el manejo de eventos,
- otros componentes que se agregan dentro del contenedor principal (éstos pueden ser contenedores o componentes simples).

El paquete *Swing* fue seleccionado para diseñar la interfaz gráfica del servidor de mensajes, la cual interacciona con el administrador del servidor. Al dar clic en los botones de control, se generan los mensajes que posteriormente pasan al mecanismo de filtrado y después hacia los destinatarios.

La aplicación cliente cuenta con una interfaz que se activa cada vez que el usuario recibe un mensaje. Se trata de una ventana emergente, la cual es iniciada al ejecutar la aplicación en el dispositivo móvil. En realidad no se utiliza ninguna API especial para la interfaz gráfica de la aplicación cliente, solo se usa la clase *Alert* que permite mostrar una pantalla de texto durante un tiempo o hasta que se produzca un comando tipo *OK*.

Para el diseño de una interfaz Web orientada a dispositivos móviles se analizaron los siguientes puntos:

- la visualización: debido a que los dispositivos móviles tienen pantallas pequeñas que van de 120x120 a 340x240 pixeles o más, se debe definir el tamaño de las dimensiones de la interfaz, de tal manera que pueda adaptarse a las pantallas de la mayoría de los dispositivos móviles,

- entrada de datos: los teclados de los dispositivos móviles son más difíciles de manipular, por lo tanto se debe analizar la forma de introducir datos, *i.e.*, entre menos se utilice el teclado y los datos se introduzcan de una forma mas dinámica, la interacción con el usuario será más sencilla de usar,

- navegadores móviles: los dispositivos móviles con interfaces inalámbricas cuentan con navegadores de Internet, la mayoría de ellos no soportan la ejecución de contenidos dinámicos como el *javascript*. Por lo tanto, para navegar por un sitio Web móvil es necesario hacer uso de enlaces (*links*).

Como se puede observar es importante tomar en cuenta la diversidad de pantallas en el proceso de diseño de una interfaz Web móvil, también se debe de cumplir con los estándares de diseño Web *XHTML* y *CSS*, *i.e.*, la creación de un código con los estándares mencionados anteriormente debe de poder usarse en cualquiera de los futuros proyectos para dispositivos móviles, aportando beneficios, tiempo y productividad en los siguientes proyectos.

3.6 Diseño de la base de datos

Dada la naturaleza del servicio, al llevarse a cabo un registro de usuarios y dispositivos, es necesario contar con una base de datos que permita agregar y recuperar información en el momento que se requiera. Por lo tanto, es necesario elegir un sistema gestor de bases de datos (SGBD) que permita hacer una colección de datos, interrelacionarlos y manipularlos, a través de un conjunto de programas los cuales accedan a los datos.

Existen muchas metodologías a seguir para diseñar una base de datos (BD), en esta tesis particularmente se realizaron los siguientes pasos:

- diseño conceptual: en esta etapa se debe construir un esquema de información, el cual es usado en la organización, institución o empresa. Al momento de la construcción del esquema, se descubre la semántica (significado) de los datos, *i.e.*, se descubren las entidades, atributos y relaciones. Esta información es obtenida directamente de los requerimientos de la organización,

- diseño lógico: en este punto se crea un nuevo esquema de información de la organización, basándose en un modelo de base de datos, sin tomar en cuenta el SGBD que se va a utilizar posteriormente. En esta etapa se transforma el diseño conceptual en diseño lógico utilizando algún modelo de BD, en nuestro caso se uso el modelo relacional. Además se realiza el proceso de normalización de la BD, con el objetivo de eliminar datos redundantes,

- diseño físico: en esta etapa se produce toda la descripción realizada en las fases anteriores, a través de estructuras de almacenamiento y métodos de acceso. Para llevar a cabo este punto se debe decidir por el SGBD que se va a utilizar. Con el diseño físico y el SGBD se obtiene un conjunto de tablas y las restricciones de las mismas, además de las estructuras de almacenamiento y los métodos de acceso a los datos,

Diseño conceptual de la BD

El diseño de una BD siempre comienza con la identificación de los datos que se almacenarán. La información puede ser de todo tipo, cada elemento informativo (nombre, apellidos, etc.) es lo que se conoce como dato, se debe conocer de donde surgen y cúal es la importancia que tendrán en la estructura de la BD.

Para obtener los datos se realizó un análisis a cerca de la información real que se desea almacenar en la BD, definiendo así un esquema que describe el conjunto de datos, obteniendo como resultado el grupo de entidades, los atributos de las entidades y las relaciones entre las entidades.

Una entidad no es una propiedad concreta que represente información por sí sola, es un objeto que puede poseer múltiples propiedades llamadas atributos. En nuestro caso se definieron un conjunto de entidades, una de ellas es la entidad *usuarios*, los atributos de la misma se pueden apreciar en la Fig 3.8, donde se observa que los atributos son los encargados de definir la información que representa la entidad *usuarios*.

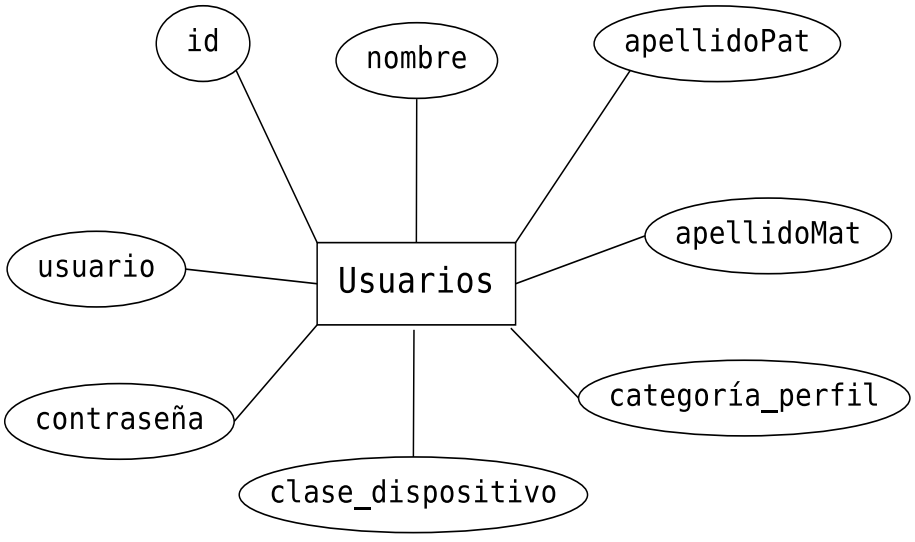

Figura 3.8: Entidad *usuarios*

En el proceso de registro de usuarios se identificaron las entidades *usuarios* y *dispositivos*. En la Fig 3.9 se puede observar el esquema conceptual, a través de un diagrama entidad-relación, entre las dos entidades mencionadas. El punto importantes en este esquema es la relación que existe entre *usuarios* y *dispositivos*.

Para definir las relaciones se analiza la asociación que puede presentarse entre las entidades, en este caso, la relación se presenta debido a que un usuario puede utilizar varios

dispositivo y un dispositivo puede ser usado por varios usuarios, en consecuencia la relación identificada es *usar*.

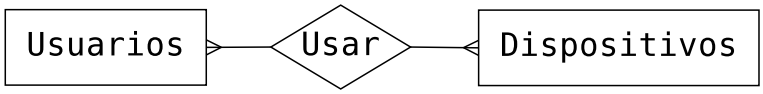

Figura 3.9: Diagrama entidad-relación entre *usuarios* y *dispositivos*

Para completar el diseño conceptual de la BD, se siguió con el análisis de los datos restantes, una vez obtenido el esquema conceptual se procede a diseñar el esquema lógico de la BD.

Diseño lógico de la BD

Una vez establecido el modelo conceptual de la BD, el diseño lógico de los datos permite que éstos se puedan representar como datos estructurados y también ayuda a identificar las posibles restricciones de la BD. El objetivo es convertir el esquema conceptual de datos en un esquema lógico, el cual debe ajustarse al gestor de BD que se utilizará en el diseño físico de la BD.

La elección de un modelo de BD, se realiza una vez que se tenga el conocimiento de los datos. En el cuadro 3.2 se presentan las ventajas y desventajas de los diferentes modelos de bases de datos, en base a éste análisis se eligió el modelo que más satisfacía las necesidades para el desarrollo de nuestro sistema.

Modelo de B.D.	Independencia Estructural	Ventajas	Desventajas
Jerárquico	no	conceptos simples, maneja relaciones padre-hijo, proporciona integridad y eficiencia con relaciones 1:N	limitaciones de ejecución, no soporta relaciones N:M, carece de lenguaje de definición de datos
Red	no	conceptos simples, soporta relaciones N:M, incluye lenguaje de definición y manipulación de datos	es muy complejo por lo tanto es poco eficiente
Relacional	sí	basado en tablas, provee herramientas que garantizan evitar la duplicidad de registros, garantiza la integridad referencial y favorece la normalización por ser más comprensible y aplicable	es deficiente a la hora de representar datos gráficos, multimedia y sistemas de información geográfica, tienen un nivel de abstracción inferior al modelo E-R
Orientado a objetos	sí	la representación visual incluye contenido semántico	sistema de navegación complejo, sus requerimientos son elevados por lo tanto las transacciones son lentas

Tabla 3.2: Cuadro comparativo de modelos de bases de datos

El modelo relacional es utilizado habitualmente en el proceso de diseño de bases de datos, se basa en registros. Los modelos basados en registros se denominan así porque la base de datos se estructura en registros de formato fijo de varios tipos.

Cada tabla contiene registros de un tipo en particular. Cada registro define un número fijo de campos, o atributos [48]. Las columnas de la tabla corresponden a los atributos del tipo de registro.

Después de analizar las características que ofrece el modelo relacional, se decidió utilizarlo para el diseño de la base de datos del servicio. El modelo relacional es el más adecuado debido a que se necesita tener una base bien estructurada e interrelacionada.

La entidad *usuarios* repesentada de manera formal en base al modelo relacional, queda de la siguiente manera:

Usuarios(id, nombre, apellidoPat, apellidoMat, usuario, password, usuarios_id, clase_disp_id, estado)

Donde: la entidad *usuarios* del esquema conceptual se representa como una relación o *tabla*. Los atributos se convierten en campos que corresponden a las *columnas* de la *tabla*, al conjunto de atributos se le denomina *grado*, a cada registro se le llama *tupla* y al número de registros o *tuplas* se les llama *cardinalidad*.

El próximo paso a realizar es definir las relaciones. Para poder relacionar las tablas se deben cumplir ciertas condiciones:

- cada tabla debe contener un solo tipo de filas o *tuplas*,
- cada fila debe ser única (sin repeticiones),
- cada columna tiene un nombre único,
- cada columna toma un valor.

Después de analizar las condiciones mencionadas, se define una combinación de atributos que, tomados en conjunto, identifican de forma única cada *tupla*, a esta combinación se le llama *llave*. Si se cuenta con más de una llave se tiene que definir una principal y las demás pueden ser alternas o foráneas.

Para determinar una llave principal se agrega un campo que almacenará un valor único e irrepetible. Para completar las relaciones entre las tablas se tienen que definir llaves foráneas, las cuales representan uno o más campos de una tabla que hacen referencia a la llave principal de otra tabla.

En la Fig 3.10 se observa la relación que existen entre las tablas *usuarios* y *repositorio*. Dado que la relación expresa que varios usuarios pueden utilizar varios dispositivos descritos en la tabla *repositorio*, se genera una tabla intermedia que llamaremos *dispositivos*.

A través de la tabla *dispositivos* se manifiesta la relación varios a varios, donde el campo *usuarios_id* de la tabla dispositivos representa una llave foránea que se relaciona con el campo *id* de la tabla *usuarios*, y el campo *repositorio_id* es la llave foránea que se relaciona con el campo *id* de la tabla *repositorio*.

La cardinalidad es un concepto que se presenta en el modelo relacional de una BD, se refiere al número de filas o *tuplas* contenidas en la relación, *i.e.*, en la Fig 3.10 se puede apreciar que, del lado de la entidad usuarios, sólo esta contenida una fila en la relación. Del otro lado, en la entidad dispositivos, pueden estar contenidas una o mas filas en la relación

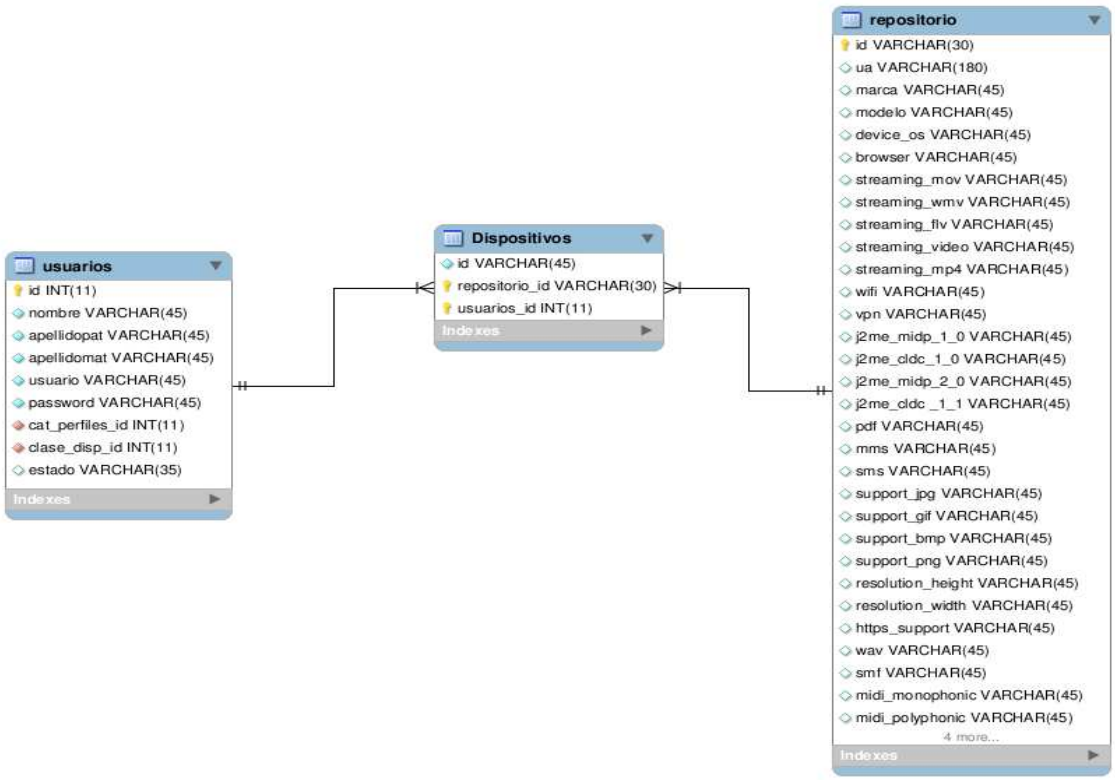

Figura 3.10: Relación de tipo varios a varios entre la tabla *usuarios* y la tabla *repositorio*

Diseño físico de la BD

Mientras que en el diseño lógico se especifica qué se guarda, en el diseño físico se especifica cómo se guarda. En esta etapa se debe conocer la funcionalidad del sistema de gestión de bases de datos (SGBD) que se vaya utilizar en la creación de la estructura física de la BD. El diseño físico no es una etapa aislada, ya que algunas decisiones que se tomen durante su desarrollo, por ejemplo al proponer una mejora, pueden reestructurar el esquema lógico.

La metodología utilizada para la estructura física en esta etapa de diseño de la BD toma en cuenta los siguientes puntos:

- traducir el esquema lógico: aquí se diseñan las relaciones en base al SGBD seleccionado

- diseñar la representación física: uno de los objetivos del diseño físico es almacenar los datos de modo eficiente, esto involucra a la producción de transacciones, el tiempo de respuesta de las transacciones y el espacio en disco. Para mejorar las prestaciones de una BD, se debe conocer la interacción que se presenta entre los dispositivos involucrados y cómo afecta a las prestaciones. La memoria principal, el CPU, la entrada/salida en disco y la red son recursos que pueden empeorar o mejorar la funcionalidad de la BD.

Para poder definir los datos en un sistema de bases de datos es necesario un lenguaje de

definición de datos, del mismo modo, para hacer consultas y modificaciones a una base de datos se necesita un lenguaje de manipulación de datos. En la práctica, ambos lenguajes no son separados, forman parte de un único lenguaje, el más usado es *SQL* (lenguaje de consulta estructurado) y también es incluido como una herramienta más en el diseño de la base de datos del servicio desarrollado en la presente tesis.

A continuación se describen las tablas más importantes y las relaciones que tienen con otras tablas.

- tabla *Usuarios*: es una de las principales tablas de la base de datos, en ella se almacena la información correspondiente al registro del usuario. Además del nombre de usuario y la contraseña, también almacena el perfil general y la clase de dispositivo que utiliza el usuario en el servicio de mensajes. Se relaciona con otras siete tablas a través de su campo *id*,

- tabla *dispositivos*: en esta tabla se almacenan los dispositivos registrados, cuenta con dos llaves foráneas que la relacionan con la tabla *usuarios* y con la tabla *repositorio*. Esta última se usa para relacionar las características y capacidades del dispositivo,

- tabla *repositorio*: contiene registros de diferentes tipos de dispositivos móviles. Esta tabla es muy importante durante el proceso de registro, el campo *ua* corresponde al user-agent, es fundamental para registrar el dispositivo de forma transparente para el usuario,

3.7 Servicio de comunicación por eventos con registro implícito de dispositivos(SEGEMENS)

En la siguiente sección se describe cada componente del servicio, además se describe detalladamente el diseño de cada aplicación, también se presenta la arquitectura del servicio y la comunicación entre los módulos del servicio.

3.7.1 Componentes

SEGEMENS es un servicio conformado por dos servicios los cuales son: 1) registro de usuarios y dispositivos, y 2) comunicación a través de mensajes cortos. Los componentes del servicio de registro se observan en la Fig 3.11, se trata de una aplicación Web basada en *servlets*, que realiza el registro del usuario y su respectivo dispositivo móvil, además proporciona la opción de modificar el perfil a los usuarios ya registrados.

El sitio Web permite agregar nuevos dispositivos en el servicio, además deja disponible la aplicación que se encarga de recibir los mensajes en el dispositivo móvil, la cual puede descargarse desde el dispositivo para posteriormente instalarla en el mismo.

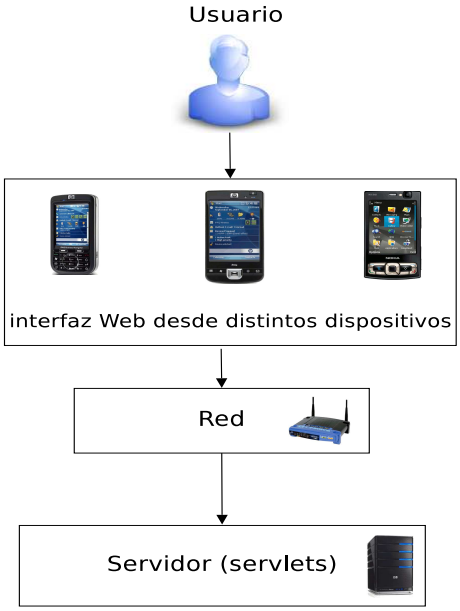

Figura 3.11: Componentes del servicio de registro

En la Fig 3.12 se puede apreciar un esquema general de la comunicación de la aplicación servidor encargada del servicio de mensajes para las aplicaciones clientes instaladas en los dispositivos móviles. Dentro de este componente destacan los mecanismos para el filtro de mensajes y el manejo de eventos, los cuales se disparan para enviar los mensajes a los usuarios. En las siguientes subsecciones se describirán cada módulo.

Figura 3.12: Componentes del servicio de mensajes

3.7.2 Aplicación Web

La Web móvil se refiere a los sitios Web que se acceden desde dispositivos móviles como *Smartphones* o PDA's [49]. Las capacidades de los dispositivos móviles ha crecido enormemente los últimos años. Casi todos los teléfonos móviles pueden ahora acceder a Internet a través de diferentes medios. La mayoría de los teléfonos móviles y PDA'S cuentan con pequeños navegadores (Web *browser*), por medio de éstos los usuarios tienen acceso a la Web.

Para el diseño de un sitio Web móvil es importante conocer las características y capacidades de los dispositivos móviles. Existe una gran variedad de telefónos móviles, los cuales cuentan con un procesador especialmente diseñado y ejecutan sistemas operativos especiales, *Symbian* y *Windows Mobile* son los más utilizados, aunque muchos fabricantes desarrollan sus propios sistemas.

En los últimos tres años, los teléfonos móviles han tenido numerosas innovaciones, ocasionando la existencia de una gran variedad de dispositivos en el mercado, de simples teléfonos han llegado a convertirse en teléfonos inteligentes (*Smartphones*).

Los tamaños de las pantallas, la resolucón y la densidad de color cambian de un dispositivo a otro; se pueden encontrar pantallas con tamaños de 120x120 a 320x240 pixeles o más. Los teclados para la entrada de datos son conformados por 12 teclas y en ocasiones son complementados con otras teclas mas para funciones especiales.

Una de las capacidades especiales de los *Smartphones* es que se puede desarrollar aplicaciones para extender la funcionalidad de los mismos. La conectividad es otra característica de los teléfonos móviles, son capaces de conectarse a redes a través de GPRS, 3G, Bluetooth o Wi-Fi (interfaz inalámbrica).

Otra clase de dispositivos móviles son los PDA's (Asistentes digitales personales), los cuales contienen muchas aplicaciones de negocios, e-mail, oficina móvil y otros tipos de software. Existen otros dispositivos como las mini *laptops* o consolas de video-juegos que también pueden ser utilizados como clientes Web.

Después de analizar las características y capacidades de los dispositivos móviles, el punto principal para el diseño de un sitio Web móvil, es averiguar cuales son realmente las dimensiones de pantalla de un dispositivo móvil. Por ejemplo:

- el iPhone es de 320 pixeles de ancho por 480 de alto,
- muchos dispositivos de Nokia N-Series son de 240 pixeles de ancho por 320 de alto,
- los dispositivos más nuevos invierten de forma espontánea la anchura y la altura,
- los dispositivos Nokia más populares disponen de pantallas que van de 176x208 hasta 352x416 pixeles.,
- las resoluciones de pantalla Blackberry varían de 160x160 hasta 324x 352 pixeles.

Para que el sitio Web móvil pueda adaptarse a una amplia gama de dispositivos móviles y tamaños de pantalla se deben definir estrategias eficaces de adaptación y orientación. La plasticidad es un concepto que sin duda resalta en la búsqueda de resolver este tipo de problemas.

El diseño de una interfaz gráfica de usuario se puede llevar a cabo de dos formas, estática o dinámicamente, según algunos de los conceptos del tema de plasticidad [50]. La estrategia utilizada para la solución, particularmente en el diseño de la interfaz del sitio Web móvil, fue definir estáticamente los elementos que se van a visualizar en el dispositivo móvil.

La diversidad en las pantallas y el tamaño de pixel son dos aspectos que se deben tomar en cuenta a la hora de estandarizar los elementos que conformaran la interfaz de usuario. Al diseñar para la Web móvil, se puede suponer que las páginas se desplazan de hacia arriba y hacia abajo, por lo tanto, de alguna forma se omite la preocupación por la altura de la pantalla, lo cual permite concentrarse en el tratamiento de la diversidad en el ancho de las pantallas de los dispositivos móviles.

Con un desarrollo estratégico, teniendo en cuenta los tamaños de pantalla (principalmente el ancho), se puede construir sitios Web escalables para una amplia gama de dispositivos móviles.

Una estrategia simple es crear un diseño de referencia por defecto, *i.e.*, se debe seleccionar un dispositivo de referencia, este dispositivo se usará como base durante el proceso de diseño. La mayoría de los dispositivos tiene un ancho de 240 pixeles, a partir de este tamaño se define un diseño de referencia para la interfaz Web.

El contenido que aparece en un sitio Web móvil es diferente al contenido que se visualiza en los sitios Web normales. En nuestro caso se descartó la posibilidad de implementar contenidos dinámicos, debido a las limitantes de los navegadores de Internet de los dispositivos móviles. Los elementos para navegar son principalmente enlaces y botones de acción. El diseño de la página Web se ilustra en la Fig 3.13

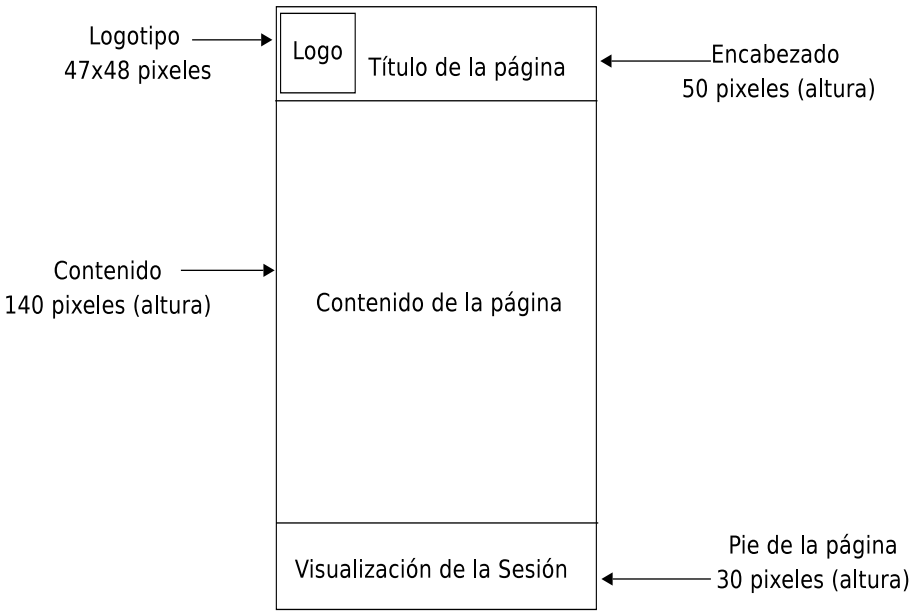

Figura 3.13: Diseño del sitio Web de acuerdo al tamaño referencial

El contenido del sitio se describe a continuación:

- el logotipo del sitio está adaptado para cada agrupación de dispositivos. Con esto se garantiza la visualización de la imagen,

- los encabezados están colocados al 100% del ancho de la pantalla utilizando una imagen de fondo repetitiva,
- las imágenes de contenido no son mayores a 200 pixeles de ancho,
- se cuenta con una hoja de estilos para establecer los valores de cada contenido de acuerdo al tamaño referenciado del dispositivo.

3.7.3 Aplicación cliente

La aplicación cliente tiene la tarea de visualizar los mensajes que llegan al dispositivo móvil. Se basa en un mecanismo de comunicación basado en sockets TCP, con el cual se lleva a cabo la conexión con el servidor y la recepción de los mensajes. La aplicación cliente tiene la capacidad de ejecutarse en segundo plano (*background*), *i.e.*, se puede salir de la aplicación y al momento de recibir un mensaje, se visualizará una ventana de alerta que muestra el mensaje recibido.

En la aplicación cliente se cuenta con un proceso de entrada al servicio, *i.e.*, se debe escribir el nombre de usuario y contraseña. El objetivo de este procedimiento es saber que usuarios son los que están en línea, también se conocería el tipo de dispositivo con el cual se han conectado.

Gracias al soporte de programación multihilos es posible diseñar una aplicación que se ejecute fuera de la interfaz del usuario y aparecer nuevamente después de la ocurrencia de un evento, *e.g.*, después de un intervalo de tiempo, la detección de una conexión o la llegada de un mensaje. En la Fig 3.14 se puede apreciar el diseño de la aplicación cliente, esta puede ser manipulada por dos eventos:

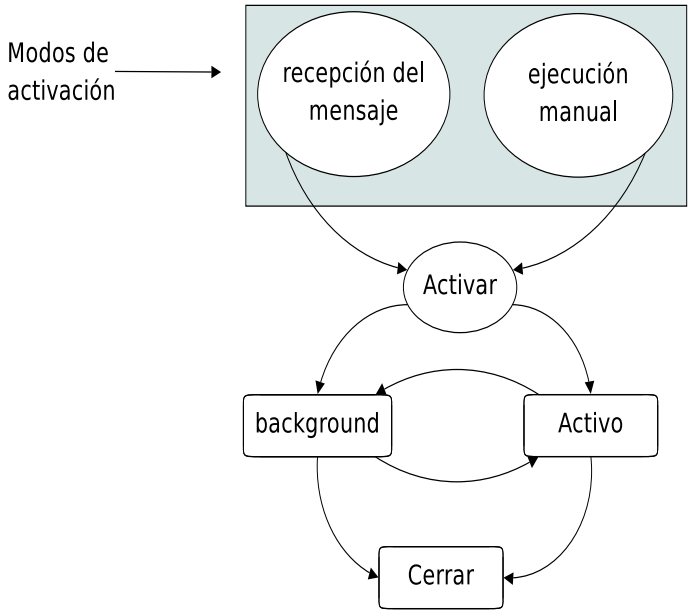

Figura 3.14: Diseño de la aplicación cliente

- ejecución manual: la aplicación debe de ser ejecutada directamente por el usuario
- recepción de mensaje: una vez ejecutada la aplicación se queda activa en segundo plano a la espera de un mensaje.

Para lograr que la aplicación se ejecute en segundo plano, se cuenta con un proceso que permite salir de la interfaz de usuario y deja un subproceso activo. El subproceso tiene la capacidad de visualizar, por medio de ventanas de alerta, el mensaje que se ha recibido, además vuelve a colocar la interfaz de usuario en la pantalla, para que se pueda dejar la aplicación ejecutándose o cerrarla. En la Fig 3.15 se observan los componentes de la aplicación cliente.

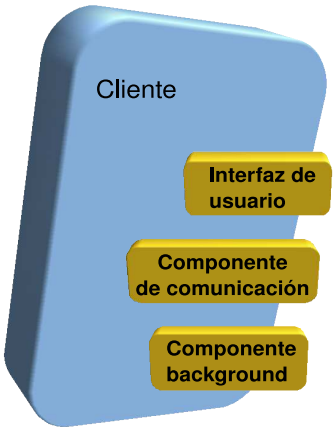

Figura 3.15: Componentes de la aplicación cliente

3.7.4 Aplicación Servidor

Una aplicación servidor es un proceso dedicado a escuchar, atender y responder las peticiones realizadas por clientes. Si se desea diseñar una aplicación servidor que tenga la capacidad de atender a varios clientes, es necesarios aplicar mecanismos de programación multihilo, *i.e.*, se define un hilo para cada cliente.

Para el servicio de mensajes cortos, se cuenta con un servidor el cual se encarga de aceptar conexiones de varios clientes. El servidor tiene una característica especial, toma la iniciativa de enviar los mensajes hacia los clientes. Esta particularidad se toma de los mecanismos de actualización de información en sistemas distribuidos con arquitecturas cliente-servidor.

Como se menciona en la sección 3.4 de este capítulo, se cuenta con un perfil de usuario, el cual se toma en cuenta para que el servidor envíe mensajes cortos a los clientes. El

servidor analiza el perfil editado por el usuario, donde se especifíca el tipo de información que el usuario desea recibir. Por lo tanto, cuando se genera un mensaje, el servidor se encarga de dirigirlo a los usuarios que marcaron el interés por tipo de información que se va a enviar.

En resumen, el servidor se encarga de hacer un filtro para el envío de los mensajes, el cual se lleva a cabo basándose en el perfil que el usuario ha especificado. Los componentes generales del servidor se pueden apreciar en la Fig 3.16.

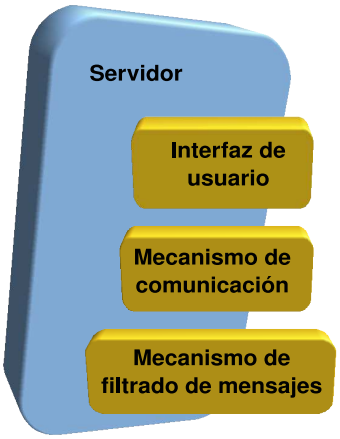

Figura 3.16: Componentes de la aplicación servidor

La descripción de cada componente se describen a continación:

- interfaz de usuario: éste componente se encarga de dibujar la interfaz gráfica de usuario. Debido a que el servidor está bajo el manejo de un administrador, la interfaz muestra una serie de botones de acción que sirven para visualizar las opciones necesarias para editar y enviar los mensajes,

- mecanismo de comunicación: se encarga de comunicar el servidor con los clientes a través de sockets TCP. El servidor es el que envía los mensajes después de generarse un evento producido por el administrador, sin necesidad de recibir alguna petición por parte de los clientes. Los mensajes también son generados automáticamente, el servidor cuenta con un mecanismo de envío de mensajes basado en eventos.

- mecanismo de filtrado de mensajes: Los mensajes son dirigidos directamente a los usuarios interesados por la información contenida en el mensaje. Para poder enviar los mensajes específicamente a un grupo de usuarios se realiza un filtro basado en el perfil de los usuarios.

Una de las funciones del servidor es validar cada cliente que se conecta con él, *i.e*, se realiza una consulta a la base de datos para verificar que el cliente esta registrado, si es

así, el servidor modifica en la base de datos el registro que indica que el usuario está en línea.

Como ya se ha mencionado, parte del servicio de mensajes es controlado por el administrador del servidor de mensajes, por lo tanto, la interfaz que se muestra en la pantalla del servidor visualiza las opciones, ya sea para editar el mensaje o para activar el evento que provoca el filtrado de acuerdo a los perfiles de los usuarios. En la interfaz se muestran los botones que indican los grupos de usuarios que administra el servidor.

Cada grupo de usuario puede recibir distintos tipos de mensajes. Al dar clic sobre un grupo en particular pueden desplegarse otros botones de acción que representan a los subgrupos que puede contener el grupo. Al seleccionar un subgrupo, se muestra la ventana de edición de mensajes.

Para editar un mensaje, primero se debe seleccionar alguna categoría de la lista, posteriormente se escribe el mensaje que se va a enviar y, a través del botón de acción, se llama al componente encargado de realizar el filtro para el envío del mensaje. El diseño de las ventanas de la intefaz gráfica se observa en la Fig 3.17.

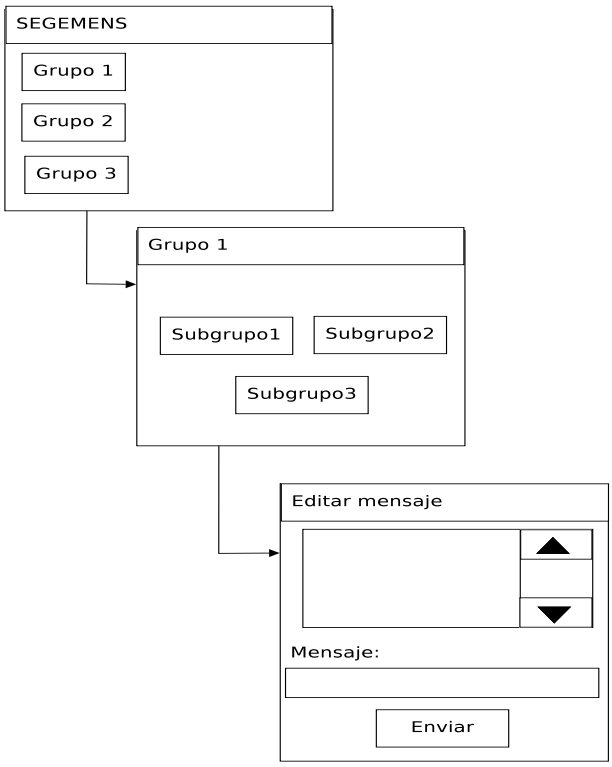

Figura 3.17: Vista de la interfaz del lado del servidor

En la Fig 3.18 se puede observar los componentes que integran al servidor de mensajes. El componente encargado del filtro de los mensajes, realiza una consulta a la base de datos, para analizar las preferencias de los usuarios conectados, las cuales se relacionan con el contenido del mensaje que se va a enviar.

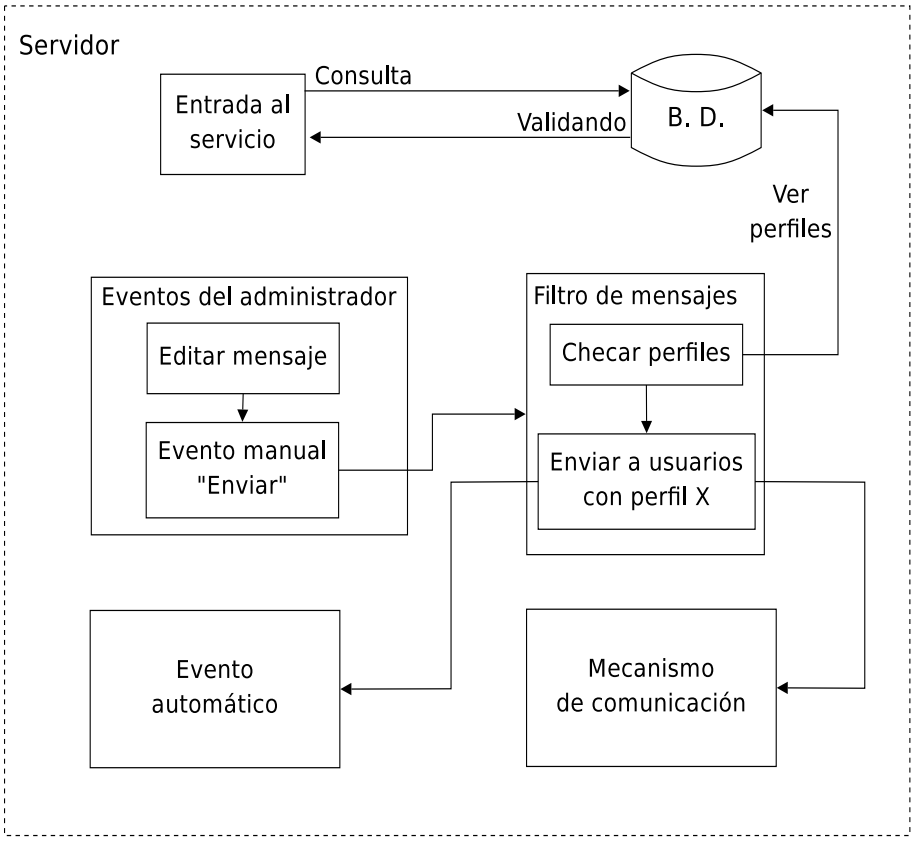

Figura 3.18: Componentes del servidor de mensajes

3.8 Arquitectura de SEGEMENS

Después de presentar los componentes del servicio, a continuación de describe la arquitectura de SEGEMENS. En primer lugar se describe la arquitecura del servicio de mensajes cortos. Se trata de una arquitectura distribuida de tipo cliente-servidor donde, de manera general, los clientes se conectan al servidor y se quedan en espera de la información que el servidor envíe. De lado del servidor es donde se toma la iniciativa de comunicación con los clientes. Esta característica es representada por el mecanismo de actualización de datos de tipo *Push* [23], definido como replicas permanentes iniciadas por el servidor.

Posteriormente se describe la aquitectura del sitio Web orientado a dispositivos móviles, por medio del cual se realiza el registro de los usuarios y dispositivos, así como la edición del perfil de usuario.

3.8.1 Arquitectura del servicio de mensajes cortos

En la Fig 3.19 se muestra la arquitectura cliente-servidor de SEGEMENS. El servidor se encarga de atender las conexiones iniciadas por los clientes, además se encarga de filtrar y enviar los mensajes editados por el administrador del servicio. Una vez establecida la conexión, el cliente queda a la espera de los datos que el servidor le envíe.

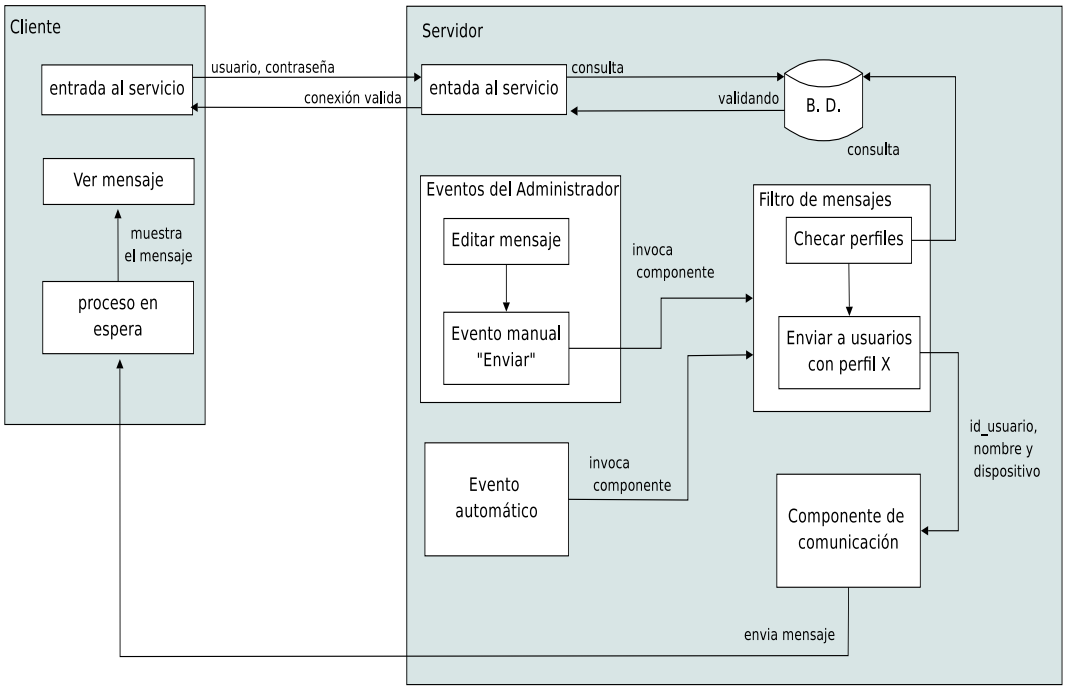

Figura 3.19: Arquitectura del servicio de mensajes cortos

El proceso de conexión se presenta cuando el cliente tiene que escribir su nombre de usuario y la contraseña, estos datos se envían al servidor a través del dispositivo móvil. El servidor hace una consulta en la base de datos para corroborar la identidad del usuario. Si la verificación es correcta, el servidor coloca en línea al cliente, indicando el identificador de usuario, el nombre y el tipo de dispositivo que está utilizando en la base de datos; además, envía un mensaje de confirmación de la conexión al cliente. Esta interacción se observa en la Fig 3.20 donde se representa la secuencia de la entrada al servicio.

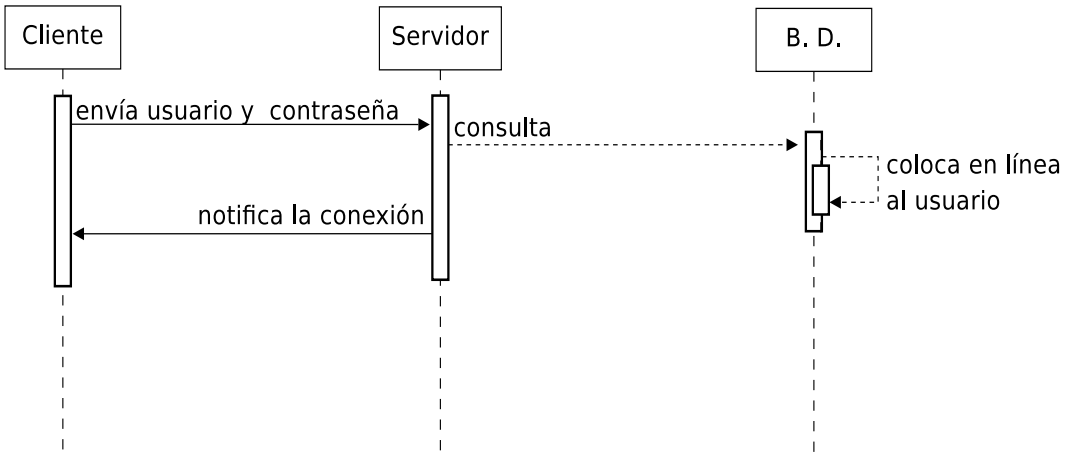

Figura 3.20: Diagrama de secuencia de la entrada al servicio

Para poder enviar un mensaje corto se debe seleccionar el grupo de usuarios destinatario, dentro de los grupos de usuarios pueden existir subgrupos, en el caso de que

existan se debe elegir el subgrupo destinatario. Para cada usuario existen categorías o clasificaciones que representan los distintos tipos mensajes cortos que pueden recibir. Las categorías se visualizan en la pantalla del servidor, las cuales son seleccionadas por el administrador para señalar el tipo de mensaje que se pretende enviar. También se presenta un cuadro de texto donde se edita el mensaje. Finalmente se envía el mensaje a los usuarios. En la Fig 3.21 se observa lo descrito anteriormente.

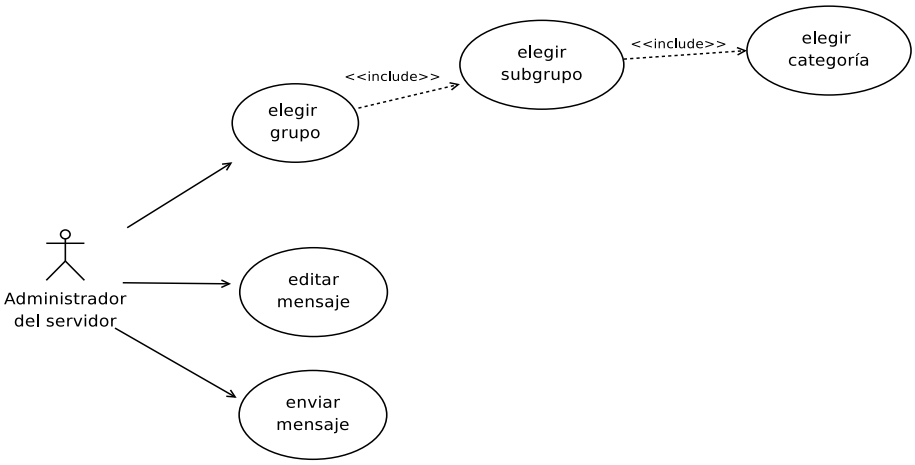

Figura 3.21: Casos de Uso de la interfaz gráfica para enviar un mensaje

La categoría seleccionada es un dato fundamental, al momento de direccionar el mensaje hacia los usuarios. Al saber la categoría se tendrá el conocimiento del tipo de mensaje que se va a enviar, en consecuencia, se debe saber qué usuarios han marcado como preferencia ese tipo de información. Para saber las preferencias de los usuarios se hacen consultas a la base de datos, donde ya se tiene especificada ésta información. Todo esto se define en el perfil de usuario almacenado. El procedimiento descrito anteriormente (ver Fig. 3.22) es parte del módulo de filtración de los mensajes cortos, la información final de esta fase es el contenido del mensaje y los usuarios destino. Estos datos son enviados al módulo de comunicación.

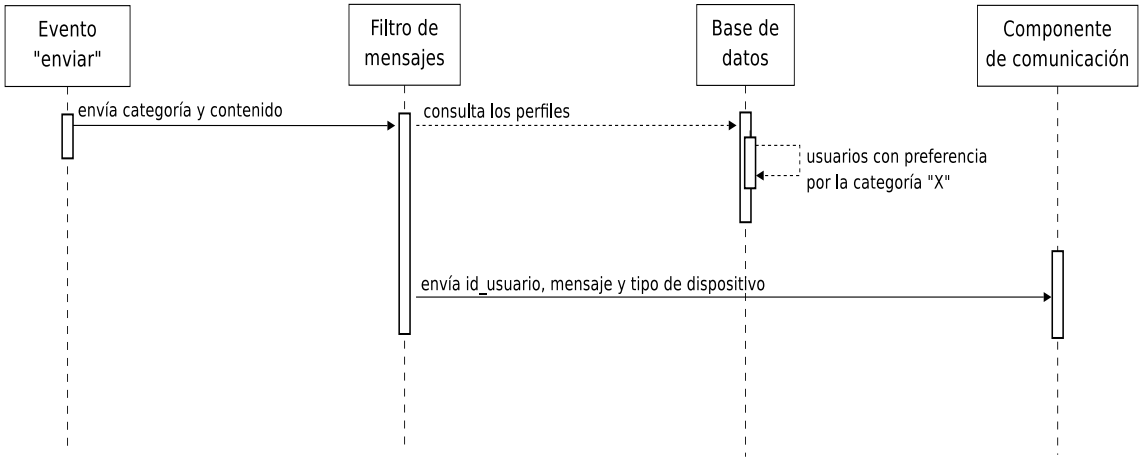

Figura 3.22: Diagrama de secuencia del filtro de mensajes

El módulo de comunicación se encarga de enviar, a través de la red, el mensaje a un grupo específico de usuarios, esta es la parte final del ciclo que se lleva a cabo para enviar los mensajes cortos (ver Fig 3.23). Del lado del cliente, se recibe el mensaje y se visualiza en una ventana de alerta. Básicamente este es el funcionamiento del servidor de mensajes cortos.

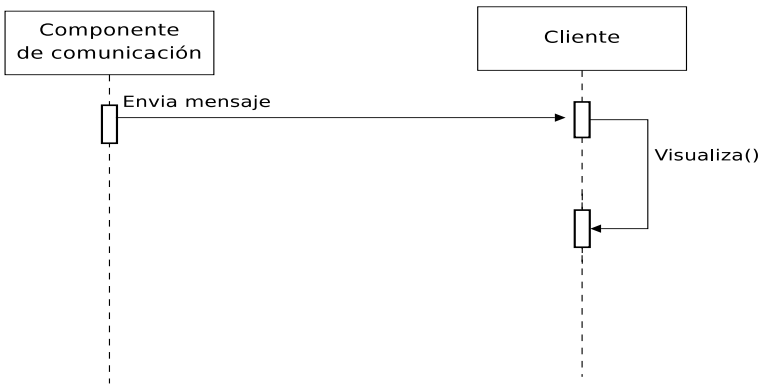

Figura 3.23: Diagrama de secuencia del componente de comunicación

3.8.2 Arquitectura del sitio Web

La otra parte del servicio SEGEMENS es el registro de usuarios y dispositivos. Como ya se ha mencionado en la sección 3.7.1 el registro se lleva a cabo a través de un sitio Web totalmente orientado a dispositivos móviles. Este sitio Web proporciona los mecanismos para modificar el perfil o registrarse, si es un usuario nuevo y posteriormente crear su perfil.

La arquitectura, al igual que el servicio de mensajes cortos, también utiliza el esquema cliente-servidor, la diferencia radica en que los clientes sí solicitan la información al servidor Web (ver Fig 3.24), *i.e.*, la iniciativa de comunicación se da por parte de los clientes. A este mecanismo de actualización de datos se le conoce como *Pull* [23].

Dentro del servicio de registro se proporcionan dos funciones, naturalmente el registro tanto del usuario como del dispositivo móvil y la modificación del perfil de usuario. La arquitectura del módulo de registro es iniciada al dar clic sobre el botón de acción *Registrarse* que se visualiza en la página de inicio del sitio Web. El usuario realiza la primer solicitud al servidor, al solicitar la interfaz de registro. En el navegador del dispositivo se visualiza un formulario, el cual consiste de ciertos campos que el usuario debe llenar. Posteriormente los datos son enviados al servidor Web y la información es recibida en un *servlet*.

Cuando los datos llegan al *servlet*, encargado de crear la cuenta del usuario, lo primero que hace es leer el encabezado HTTP que se envía por parte del cliente. La finalidad es conocer el tipo de dispositivo con el cual se está conectando el usuario. Cabe señalar que dentro del encabezado HTTP se encuentra el campo *user-agent*, por medio de él se proporciona información del dispositivo *e.g.*, el fabricante, modelo del dispositivo, sistema opertativo, nombre del navegador de Internet, entre otras configuraciones. De toda esta información, la que más interesa es el sistema operativo que tiene instalado el dispositivo,

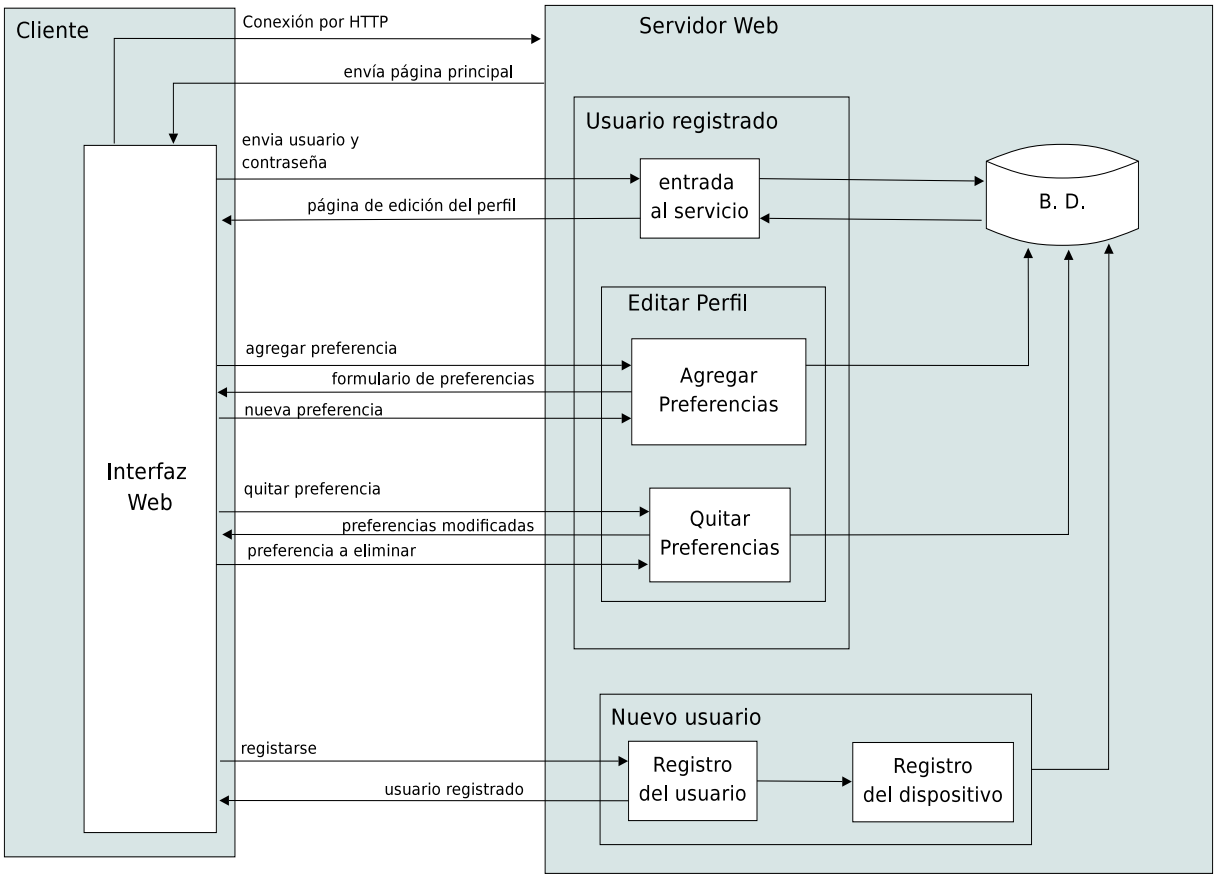

Figura 3.24: Arquitectura general del servicio de registro

dado que el servicio de registro tiene la capacidad de detectar si el dispositivo que se está usando es un dispositivo móvil, un dispositivo de mayores capacidades como una *laptop* o una PC.

Posteriormente se debe verificar que el nombre de usuario especificado esté disponible, *i.e.*, que no haya sido ya utilizado por otro usuario. Esta tarea se lleva a cabo al realizar una consulta a la base de datos. Después de verificar que el nombre de usuario éste disponible se procede a insertar al nuevo usuario en la base de datos. Para el caso del registro del dispositivo, se cuenta con un repositorio de datos referente a distintos dispositivos móviles. El dato importante de dicho repositorio es el *user-agent*, el cual es comparado con el *user-agent* del dispositivo utilizado por el usuario. Si el resultado de la comparación es verdadera (coinciden los *user-agent*) los demás datos del repositorio se adjuntan al dispositivo del nuevo usuario y se almacena en la base de datos. Con esto se puede saber las caraterísticas del dispositivo del usuario que se está registrando. El proceso de registro del dispositivo se realiza de forma transparente para el usuario. De la misma manera, también se almacenan los datos del usuario.

El *servlet* tiene mecanismos de excepciones, *e.g.*, cuando un usuario no llena todas las cajas de texto, el nombre de usuario ya está registrado, el dispositivo no es reconocido, entre otros. Todo el procedimiento descrito anteriormente se puede apreciar en la Fig 3.25.

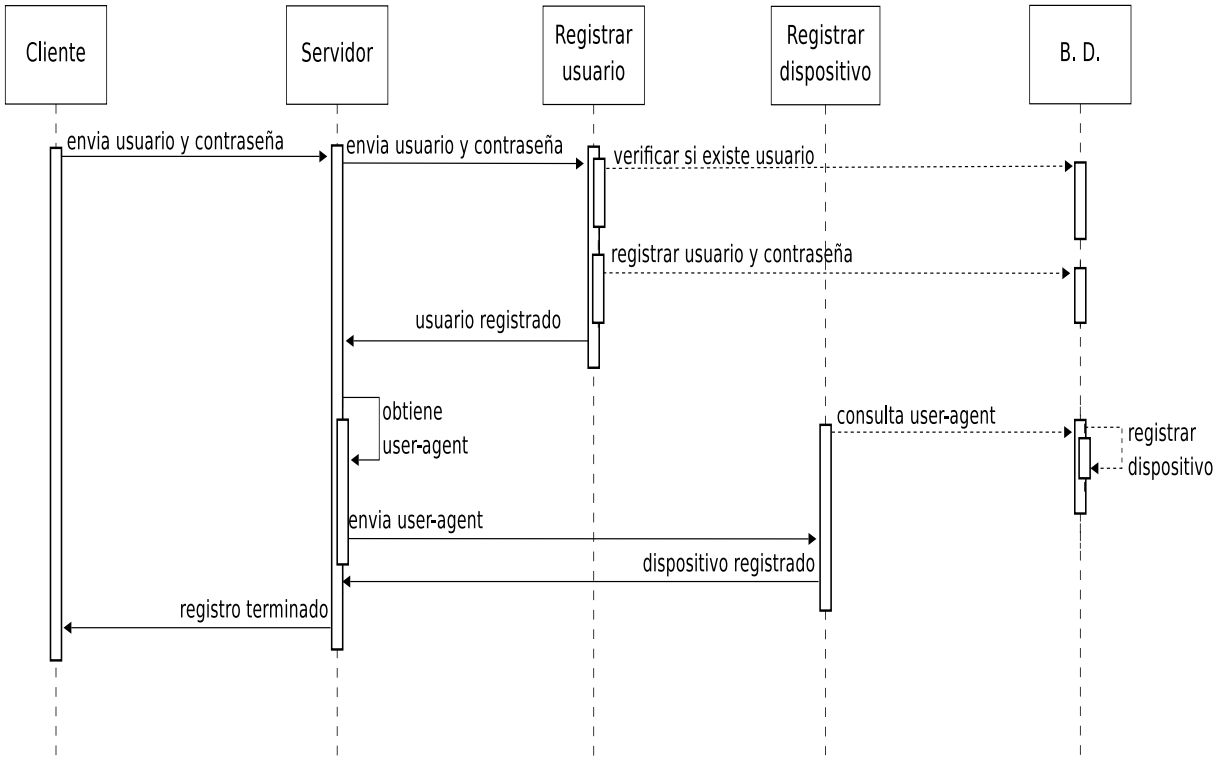

Figura 3.25: Diagrama de secuencia del registro de un nuevo usuario

Si un cliente ya está registrado, puede modificar su perfil desde el mismo sitio Web. La página de inicio muestra un formulario donde el cliente debe escribir el nombre de usuario y la contraseña para entrar al sitio. Los datos escritos se envian al *servlet* encargado de verificar si se encuentran en la base de datos. Una vez verificada la información y si todo está en orden, se identifica el tipo de perfil del usuario, dependiendo del tipo de perfil se direcciona a una página específica.

Debido a que los tipos de perfiles de usuarios son distintos, la información para cada uno de ellos cambia, *i.e.*, para cada tipo de perfil de usuario se tiene una página especifica. Las operaciones que el usuario puede realizar son:

- modificar la información del perfil: en este punto el usuario puede modificar las preferencias de los tipos de mensajes que desea recibir en su dispositivo móvil
- agregar dispositivo: si el usuario desea utilizar otro dispositivo que no está registrado, se puede agregar a través de esta opción
- descargas: en esta sección se proporciona la aplicación cliente que se utiliza para el servicio de mensajes cortos.

Cuando el usuario elige modificar su perfil, el sitio Web le permite agregar más preferencias o en su defecto quitar algunas en cuanto a los tipos de mensajes. Si se elige agregar un dispositivo, el proceso de registro se lleva a cabo de manera transparente para el usuario, tal y como se realiza en la fase de registro de usuario. Por último, la opción de

descargas permite obtener la aplicación cliente en el dispositivo móvil que se desea usar para el servicio de mensajes cortos (ver Fig. 3.26).

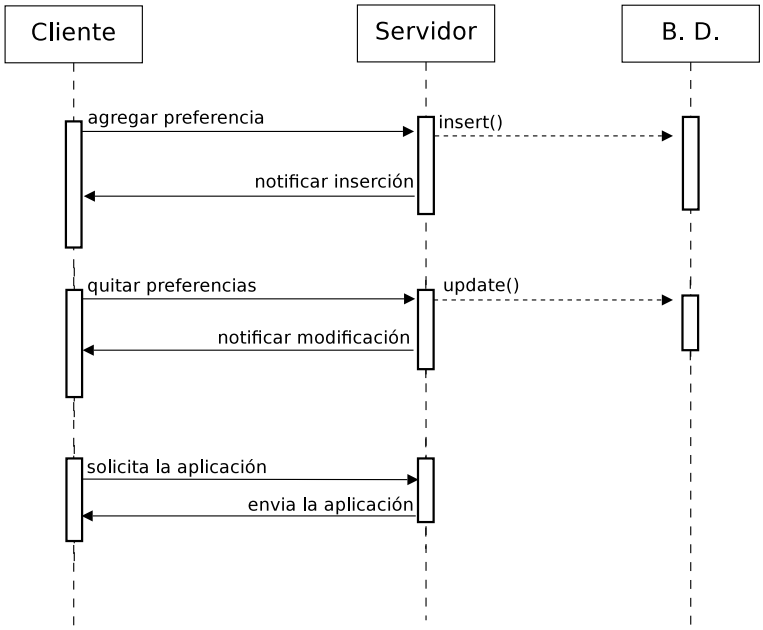

Figura 3.26: Operaciones para agregar o quitar preferencias, y descargar la aplicación cliente del servicio de mensajes cortos

Capítulo 4

Implementación del servicio SEGEMENS

A continuación se describe detalladamente la implementación del servicio SEGEMENS. Con los componentes definidos en el capítulo 3, se presenta el diagrama de las clases correspondiente a la implementación del servicio, también se presenta la interacción y el funcionamiento entre las clases desarrolladas, además, se describe la implementación de la base de datos. Como ya se describio nuestra propuesta consta de dos subservicios, a) registro de usuarios y dispositivos y b) servicio de mensajes cortos. En éste capítulo se mencionan las tecnologías usadas para el desarrollo de cada subservicio. Al final se describen las pruebas realizadas y los resultados obtenidos.

4.1 Ambiente de implementación

La solución propuesta fue desarrollada tanto en código *Java Stándar Edition* como en lenguaje *J2ME* para dispositivos móviles. El servicio fue implementado con un conjunto de clases *Java*, que usan los métodos de la API estándar de *Java* (*sockets*, *threads*, conexión a bases de datos) para lograr la funcionalidad deseada.

Para el desarrollo del servicio de mensajes cortos es importante señalar que se utilizó la técnica de programación multihilo, la cual forma parte de la plataforma *Java* y que permite desarrollar aplicaciones basadas en la arquitectura cliente-servidor. El objetivo es realizar aplicaciones independientes, ésta es una característica requerida por los entornos distribuidos. También se ha utilizado el lenguaje *J2ME*, para desarrollar una aplicación *cliente* que se ejecuta en un dispositivo móvil. Con esto se completa la arquitectura distribuida cliente-servidor aplicada en esta tesis.

En el servicio de registro de usuario y dispositivos móviles hemos utilizado la tecnología *servlets* de *Java*, los cuales crean instancias de objeto que se ejecutan en un contenedor de *servlets* (*e.g.*, *Tomcat*), *i.e.*, se ejecutan en un servidor. El objetivo es generar páginas dinámicas, a partir de los parámetros enviados por los navegadores de Internet de los dispositivos móviles.

Para la implementación de la base de datos se ha utilizado *MySQL*. Es uno de los manejadores de bases de datos más populares y que permite un acceso relativamente

sencillo a través de una implementación nativa del *driver* de *Java*. Es muy utilizado en aplicaciones Web, debido a que en estos entornos la lectura de datos es intensa. *MySQL* es una herramienta ideal para este tipo de aplicaciones.

4.2 Implementación de los componentes del servicio

Una vez que se han especificado todos los componentes en el capítulo 3, se presenta la implementación de cada elemento introducido en la arquitectura general del servicio. El servicio está compuesto por dos subservicios, uno de ellos es el servicio de registro de usuarios y dispositivos. El registro se realiza a través de un sitio Web, donde el usuario crea una cuenta para poder iniciar una sesión. Una vez registrada la cuenta, el sitio Web proporciona una página que está compuesta por tres apartados: la edición del perfil, la sección para agregar dispositivos y una sección de descargas

El otro componente es el servicio de mensajes cortos, dentro de él existen mecanismos de comunicación basados en *sockets* TCP. El filtro de mensajes es el elemento más importante para este servicio, se encarga de direccionar los mensajes a un grupo de usuarios. Para determinar el destino de los mensajes se evalúa el perfil de cada cliente, específicamente sus preferencias. Todos los procesos descritos anteriormente son realizados por una aplicación servidor, desarrollada en el lenguaje de programación *Java*, la cual se encarga de aplicar mecanismos de comunicación basados en eventos.

El complemento del servicio de mensajes es una aplicación de tipo cliente que se obtiene del sitio de descargas de la página Web y que se instala en el dispositivo móvil. Con dicha aplicación se reciben los mensajes que envía el servidor. Ambos servicios realizan consultas a una BD, donde se encuentran almacenados los datos correspondientes a los usuarios, dispositivos y preferencias de mensajes

El medio físico por el cual se comunican los dispositivos con el servidor central, es una red inalámbrica administrada localmente.

4.2.1 Servicio de Registro

A continuación se describen las clases de los componentes implementados para lograr el funcionamiento del servicio de registro, además se presentan los componentes internos y procedimientos encargados de realizar las tareas correspodientes al registro de usuarios y de los dispositivos móviles. En primer lugar se presentan las clases implementadas en el sitio Web basado en *servlets* de *Java*.

El servicio de registro está constituido por una aplicación Web para el registro de usuarios y dispositivos, la cual está conformada por las siguientes clases: `crear`, `checaReg`, `invalida`, `subperfil` y `guardaperfil`.

- clase `crear`: recibe los parámetros enviados desde el dispositivo móvil. Define los métodos de conexión con la base de datos, para realizar el registro del nuevo usuario.

- clase `checaReg`: se encarga de dar acceso al sitio Web. Recibe los parámetros *usuario* y *contraseña*, los cuales son utilizados para verificar su existencia en la base de datos, si los datos son correctos, por medio del método `sendRedirect`, se

direcciona a una página Web donde se visualizarán las opciones que el usuario puede realizar de acuerdo al perfil su general. De lo contrario, se manda una notificación al usuario donde se informa que sus datos no son correctos,

- clase `subperfil`: recibe los parámetros correspodientes al subperfil, que identifica la pertenencia a un grupo de usuarios dentro del servicio. Los datos recibidos son utilizados para proporcionar una interfaz de selección, donde el usuario puede elegir, de acuerdo al subperfil especificado, sus posibles preferencias,

- clase `guardaperfil`: recibe los parámetros de la página Web donde se muestra la interfaz de selección de preferencias, la cual envía las preferencias marcadas por el usuario. Contiene métodos para la inserción de los datos en la BD,

- clase `invalida`: esta clase es invocada para cerrar la sesión iniciada, misma que se mantiene al navegar por todo el sitio Web.

Registro de usuarios y dispositivos móviles

En el sitio Web, los usuarios deben crear una cuenta para poder iniciar una sesión. Una vez registrada la cuenta, el sitio Web proporciona una página que permite al usuario realizar tres tareas:

- editar el perfil del usuario: el perfil permite especificar el grupo de usuario y las preferencias, acerca del tipo de información que el usuario desea recibir en su dispositivo móvil,

- agregar dispositivos: solo los dispositivos que estén registrados son los que tendrán acceso al servicio de mensajes, por lo tanto, si se desea utilizar otro dispositivo diferente al que se registro originalmente, se debe agregar con esta opción,

- descargas: envía al cliente la aplicación que se tiene que instalar en el dispositivo móvil para poder usar el servicio de mensajes cortos.

Cualquier cambio que se desee realizar en el perfil, se hará a través del sitio de registro descrito y naturalmente sólo los usuarios registrados tienen acceso a estas opciones. Para el desarrollo del sitio Web se utilizó la tecnología *servlets* de *Java*. Con el objetivo de lograr que todas las funciones del sitio Web se realicen correctamente se desarrollaron los siguientes componentes internos:

- *nuevo usuario*: se encarga de recibir los datos personales proporcionados por el usuario, extrae el *user-agent* del dispositivo que se está conectando al sitio, almacena la información del usuario en la base de datos, analiza el *user-agent* del dispositivo, siguiendo dos propósitos: el primero es buscar el *user-agent* obtenido en el repositorio de dispositivos móviles, con la finalidad de saber de que dispositivo se trata y adjuntarle las características y capacidades que estan señaladas en el repositorio, el segundo propósito es identificar si se trata de un dispositivo móvil o fijo, con esto el servicio de mensajes se puede extender hacia computadoras de

escritorio. Esta clase cuenta con un manejo de excepciones que pueden presentarse cuando el usuario envía los datos dejando alguna caja de texto vacía o cuando el *user-agent* no se encuentra en el repositorio de dispositivos.

- *usuario registrado*: proporciona las tareas que el usuario puede llevar a cabo en el sitio Web. En este componente se atiende el proceso de conexión al servicio, el cual analiza que el nombre de usuario y contraseña sean correctos. Una vez validado el proceso de entrada, se analiza el perfil general que el usuario especificó al momento de registrarse. Cada usuario puede agregar o quitar las preferencias de los tipos de mensajes. Otra de las opciones administrada por éste componente es la que permite registrar más dispositivos. Por último, también proporciona una página de descarga, donde el usuario obtiene la aplicación *cliente* que debe instalarse en los dispositivos para el servicio de mensajes cortos.

Cabe señalar que tanto el componente *nuevo usuario* y *usuario registrado* realizan consultas e inserciones a la base de datos. El proceso de registro de un nuevo usuario se presenta en el **procedimiento 1**.

Procedimiento 1 Proceso de registro de un nuevo usuario

Entrada: `nombre, apellidoPat, apellidoMat, user, pass, confir, perfil, clase_disp`

```
1: conectarse al sitio Web
2: dar clic en el botón registrarse
3: llenar los datos correspondientes al registro
4: si (pass = confir) entonces
5:   insertar datos de entrada en la BD
6: si no
7:   la contraseña de confirmación no es la misma, intentar de nuevo
8: fin si
```

Es importante resaltar la comparación que se hace entre la variable *pass* y *confir* en el **paso 4**, con esto se confirma la contraseña que se escribió. Si la confirmación es correcta se procede a insertar los datos para generar un nuevo registro en la BD.

El **procedimiento 2** se encarga de realizar el proceso del registro del dispositivo móvil, el cual se ejecuta de manera trasparente para el usuario, *i.e.*, no se introduce ningún dato. El paso más importante de este procedimiento es el **número 4**, como puede observarse, se analiza si el sistema operativo del dispositivo es SymbianOS o CE (Windows Mobile). De acuerdo a ésta comparación, se lleva a cabo o no el registro del dispositivo. Hasta el momento, sólo es posible registrar a los dispositivos móviles que cuenten con los sistemas operativos mencionados, se tomó esta característica porque son los sistemas operativos más populares y están instalados en la mayoría de los dispositivos móviles. Además, las herramientas para programar sobre estós sistemas operativos son libres, *i.e.*, no se tiene que pagar una cuota por el uso de la herramienta, y mucho menos para poder probar las aplicaciones en los dispositivos.

Procedimiento 2 Registro del dispositivo móvil

```
1: obtiene user-agent del encabezado de la conexión HTTP
2: almacena el user-agent en una cadena de caracteres
3: analiza la cadena para identificar el sistema operativo instalado en
   el dispositivo
4: si (subcad = CE) o (subcad = SymbianOS) entonces
5:   band = 1          // coloca band = 1 para indicar que se trata de un
     dispositivo móvil
6:   consulta al repositorio de la BD para buscar el user-agent obtenido
     en el paso 1
7:   si ua = user-agent entonces
8:     obtiene el id del dispositivo de la tabla repositorio de acuerdo
       al user-agent
9:     obtiene el id del usuario de la tabla usuarios
10:    inserta en la tabla dispositivos el id del dispositivo y el id del
       usuario nuevo
11:  si no
12:    se marca como dispositivo desconocido
13:  fin si
14: si no
15:  el dispositivo es una PC o una laptop          // hasta ahora no hay
     tratamiento para estos dispositivos
16: fin si
```

Una vez realizado el registro completo se direcciona al usuario la página de inicio del sitio Web. Ahora el usuario debe realizar el proceso de entrada al sitio, el cual se observa en el **procedimiento 3**.

Procedimiento 3 Procedimiento de entrada al sitio Web

```
Entrada: usuario, contraseña
1: conectarse al sitio Web
2: ingresar el nombre de usuario y la contraseña
3: si (user = usuario) y (contra = contraseña) entonces
4:   verificar el perfil general del usuario
5:   direccionar la página correspondiente al perfil general del usuario
6:   session = true          // inicia la variable sesión en el sitio Web
7: si no
8:   usuario o contraseña incorrectos
9: fin si
```

Generalmente la página direccionada por el servidor al dispositivo móvil (como se indica en el **paso 5** del **procedimiento 3**, despliega las opciones *Perfil*, *Dispositivos* y *Descargas*. La opción *Perfil* es la única que cambia su funcionamiento dependiendo del perfil general. Las otras dos opciones realizan la misma tarea para todos los perfiles generales.

Para la edición del perfil, el usuario debe seleccionar la opción *Perfil* de la página

direccionada, de acuerdo al perfil, el usuario puede pertenecer a un grupo específico. Las opciones para editar el perfil son distintas entre los grupos de usuarios, *i.e.*, las preferencias, que representan el tipo de mensaje que se desea recibir, son distintas.

En el **procedimiento 4** se observan los pasos para agregar las preferencias de mensajes, esta opción forma parte de la edición del perfil, además de la opción *agregar*, también se pueden quitar las preferencias. En el **paso 4** del procedimiento las preferencias se presentan en forma de varias casillas, ahí el usuario elige las de su interés y posteriormente las envía al servidor para hacer el registro en la BD.

Procedimiento 4 Pasos para editar perfil-agregar preferencias

1: seleccionar la opción *Perfil* en la página //direcciona a la página donde se indica el subperfil del usuario
2: seleccionar el subperfil en la página //direcciona a la página de edición del perfil
3: seleccionar la opción agregar de la página de edición del perfil
4: seleccionar las casillas correspondientes al tipo de mensajes se desea recibir
5: enviar los datos al servidor

En el **procedimiento 5** se observa el proceso de registro de las preferencias. Una vez capturados los parámetros enviados a través de la página Web, que representan a cada preferencia marcada, se procede a insertarlos en la BD. El objetivo de este procedimiento es tener un registro que nos permita saber cuales son las preferencias marcadas por cada usuario.

Procedimiento 5 Registrar preferencias

Entrada: $datos[], i = 0$
1: $datos[] \leftarrow parametros$
2: **mientras** `(datos.Masparametros)` **hacer**
3: `cadena = dato.nextparametro`
4: $prefer[] \leftarrow cadena$
5: **fin mientras**
6: **para** $prefer[j]$ `hasta` $prefer[j++]$ **hacer**
7: `elemento =` $prefer[j]$
8: `insert(elemento)`
9: **fin para**

4.2.2 Servicio de mensajes

En esta sección se describe el servicio de mensajes cortos, particularmente se explica el desarrollo de las clases que intervienen en el funcionamiento. Las clases y sus respectivas relaciones, los componentes y la información están representadas de manera conceptual. En la Fig 4.1 se puede apreciar las clases implementadas en el servicio de mensajes.

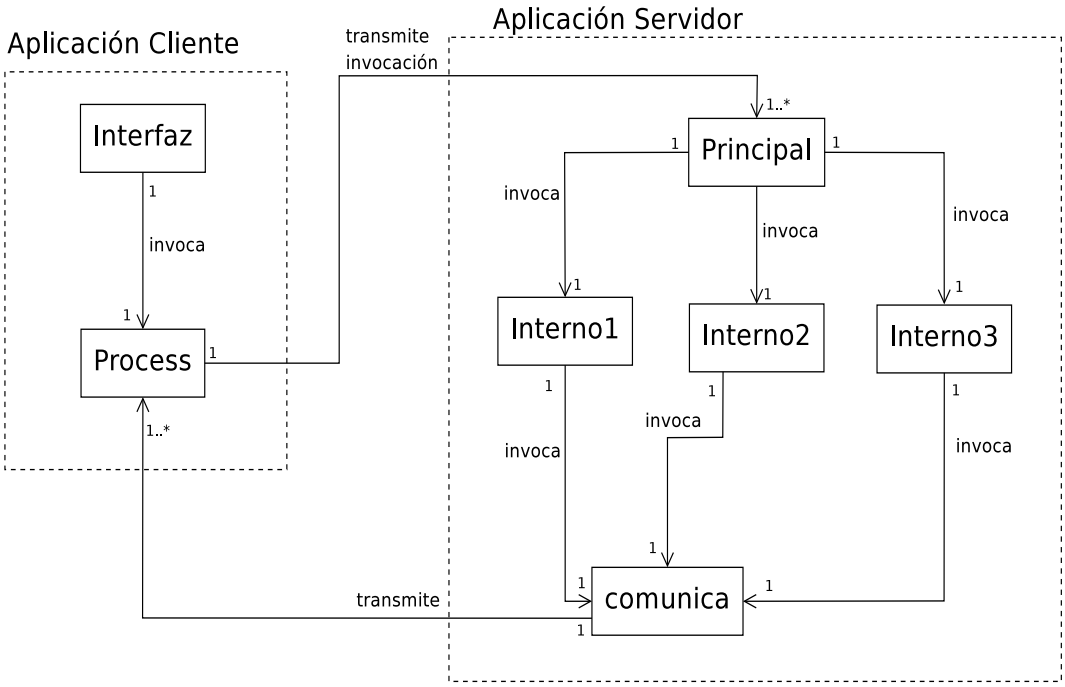

Figura 4.1: Diagrama conceptual del servicio de mensajes

En el servicio de mensajes, también se especifica el componente de comunicación basado en *sockets* TCP. El filtro de mensajes es el componente más importante para este servicio, se encarga de direccionar los mensajes a un grupo de usuarios. Para determinar el destino de los mensajes se evalúa el perfil de cada cliente, específicamente sus preferencias. Todos los procesos descritos anteriormente son realizados por una aplicación servidor desarrollada en el lenguaje de programación *Java*.

El servicio de mensajes se complementa con la implementación de una aplicación *cliente* orientada a dispositivos móviles, la cual se desarrolló en el lenguaje de programación *J2ME*. Dado que la mayoría de los dispositivos móviles cuentan con una versión reducida de la máquina virtual de Java, se puede ejecutar la misma aplicación *cliente* en distintos dispositivos, proporcionando portabilidad de ejecución, sin importar los diferentes sistemas operativos instalados.

La aplicación *cliente* tiene la responsabilidad de recibir los mensajes enviados por el servidor y visualizarlos en la pantalla. Una particularidad que resalta es que la aplicación puede cambiar de estado, *i.e.*, puede permanecer totalmente activa o pasar a segundo plano (*background*).

Para dar una mejor explicación de la implementación del servicio de mensajes cortos se describe cada componente por separado, posteriormente se presenta la interacción entre todos los componentes del servicio de mensajes.

Aplicación cliente

La aplicación *cliente* está conformada por dos clases: `interfaz` y `Process`:

- `interfaz`: es la clase principal de la aplicación *cliente*, invoca la clase *Midlet* e implementa la interfaz `Comandlistener`, la cual se encarga de capturar todos lo

eventos realizados por el usuario. Se implementa una interfaz de usuario que está conformada por dos formularios, el primero de ellos (`screen`) permite introducir los datos correspondientes para la entrada al servicio, *i.e.*, recibe el nombre de usuario y la contraseña. A través del comando `enviar`, se encapsulan los datos de entrada y se invoca un hilo de ejecución de la clase `Process`. En la invocación de hilo se agregan, como parámetros, los datos encapsulados en una cadena de caracteres. Además de las operaciones mencionadas anteriormente, el comando `enviar`, invoca el segundo formulario, el cual está preparado para recibir los mensajes enviados por el servidor. Cuenta con un comando llamado `back`; que permite enviar la aplicación a segundo plano.

- `Process`: define un hilo que invoca el método `transmit` que se encarga de realizar la conexion con el servidor y envía los datos del proceso de entrada para su evaluación. Una vez verificados los datos de entrada, la aplicación *cliente* permance en espera de los mensajes enviados por el servidor. Cuando un mensaje es recibido, se invoca el método `vizualiza` y se le adjunta el mensaje que se recibió como parámetro. También cambia el estado de la aplicación a primer plano, en el caso de que estuviera en *background*. El mensaje se visualiza a través del objeto `Alert`, el cual despliega una ventana emergente con el contenido del mensaje.

En la Fig 4.2, se observa la relación que existe entre las clases que conforman la aplicación *cliente*. Se trata de una relación de agregación por valor o también conocida como relación composición, esto es por la fuerte dependencia del objeto instanciado, *i.e.*, para poder crear los objetos en la clase `Interfaz` es necesario una asociación con el método de la clase `Process`.

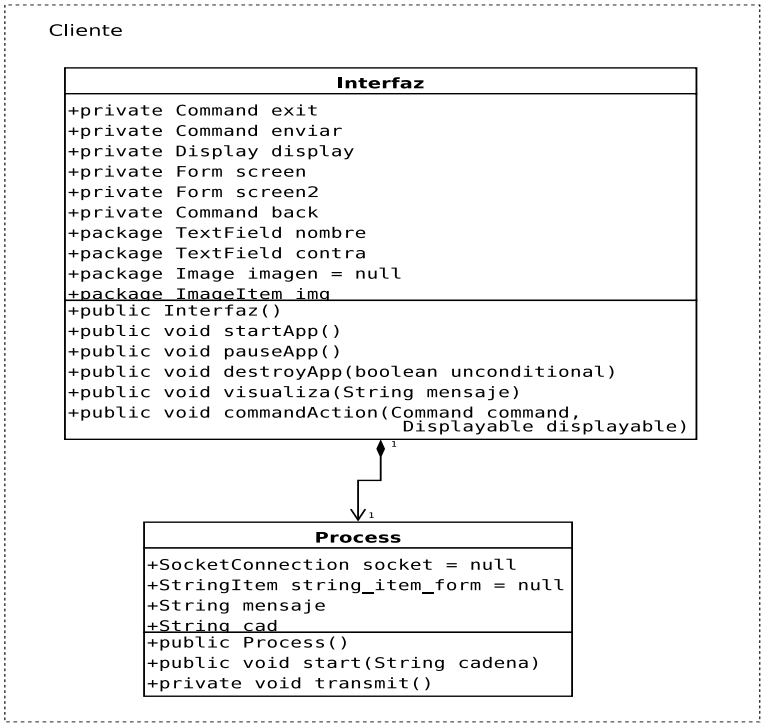

Figura 4.2: Diagrama de clases de la aplicación `cliente`

En el **procedimiento 6** se observan los pasos necesarios para la construcción del *Midlet*, en el cual se definen los elementos que conformarán la interfaz gráfica. El **paso 10** es el encargado de atender el evento que se produce al seleccionar algún comando, se puede cerrar la aplicación por medio del comando *Salir* o iniciar la conexión con el servidor al seleccionar el comando *Enviar*. Además del manejo de la pantalla y los comandos de la misma, se cuenta con una función que se encarga de visualizar los mensajes que llegan al dispositivo móvil. Los mensajes se muestra a través de ventanas de alerta. En el **paso 20** se puede observar la función encargada de visualizar los mensaje en la pantalla del dispositivo móvil

Procedimiento 6 Aplicación cliente

Entrada: `exit, enviar, back, cerrar, display, screen, screen2, nombre, contra, socket, mensaje, cad`

```
1: display = Display.obtener()
2: enviar = Command(‘‘Enviar")                    //agregando los comandos
3: exit = Command(‘‘Salir")
4: cerrar = Command(‘‘Cerrar")
5: back = Command(‘‘Background")
6: exit = Command(‘‘Salir")
7: screen = Command(‘‘Interfaz de usuario")  //declarando los formularios
8: screen2 = Command(‘‘Esperar Mensaje")
9: display.setCurrent(screen)                      //iniciando el Midlet
10: commandAction {                                 //método commandAction
11: si command = Salir entonces
12:   destruirMidlet()
13: fin si
14: si command = Enviar entonces
15:   Obtener usuario y contraseña
16:   crea objeto process()
17:   invoca process.start(cadena)
18: fin si
19: }
20: visualiza {    //función que visualiza en pantalla el mensaje recibido
21: alerta = new Alert(‘‘mensaje")
22: display.setCurrent(alerta) //muestra en pantalla el mensaje
23: }
```

En el **procedimiento 7**, se establece la conexión con el servidor, una vez establecida la conexión, el **paso 5** se encarga de enviar los datos de autentificación del usuario, el **paso 7**es el más importante, ya que se encarga de esperar los mensajes que el servidor envía.

Otro de los componentes que integran la aplicación cliente es el mecanismo *background*, el cual permite salir de la aplicación sin cerrarla por completo, *i.e.*, se implementó un hilo en el lado del cliente, el cual se utiliza para establecer la comunciación entre el usuario y el servidor. Además de establecer la comunicación y recibir los mensajes, se encarga de

cambiar de estado a la aplicación cliente. La aplicación puede estar en primer plano o segundo plano.

Procedimiento 7 Conexión y recepción de mensajes

Entrada: `socket, mensaje, cad, url, is, salida, dato`

```
1: url = socket                                  //asigna la ruta de conexión
2: Socket = SocketConnection.open()        // abre el socket para crear la
   conexión con el servidor
3: Flujo de datos de entrada:  is  // fujos de entrada y salidad de datos
4: Flujo de datos de salida:  salida
5: salida.escribir(‘‘cad")  //mensaje que contiene el nombre de usuario y
   la cotraseña
6: mensaje = is.leer_entrada()  //recibe la notificación de la conexion y
   la validadación de los datos de entrada
7: repetir
8:   dato=flujo_de_entrada //sentencia repetitiva que deja al cliente a la
     espera de los mensajes que envía el servidor
9:   Visualiza(dato)      //llama a la función encargada de visualizar el
     mensaje contenido en la variable dato
10: hasta que dato = salir
```

Aplicación servidor

La función principal de la aplicación servidor es el envío de mensajes, lo cual se realiza de acuerdo a la ocurrencia de dos tipos de eventos. El primero de ellos se refiere a todos los eventos generados por el administrados del servicio, y el segundo es un mecanismo de eventos que se dispara automáticamente después de realizarse una tarea programada.

Para los eventos provocados por el administrador, por medio de una interfaz gráfica, debe seleccionar el grupo de usuarios y la categoría destino:

- grupos de usuarios, se refiere a los diferentes perfiles generales, dentro de ellos puede existir una subclasificación que representa un grupo de usuarios más específico. Cada vez que se selecciona un grupo, la interfaz abre una ventana con sus respectivas subclasificaciones, o de lo contrario, muestra la ventana de edición de mensajes.

- categoría destino, se refiere a los distintos tipos de preferencias que el usuario puede elegir al momento de editar su perfil. Las categorías destino cambian dependiendo del grupo de usuario que se seleccionó. Para el caso de estudio mencionado en la sección 4.4, las categorías que se muestran en la interfaz corresponden a las materias del cuatrimestre vigente, entonces el administrador puede enviar un mensaje destinado a una materia especifica. De esta manera, el mensaje sólo lo van a recibir los alumnos que hayan marcado la preferencia que el administrador ha seleccionado para enviar un mensaje..

La implementación de la aplicación servidor contiene las siguientes clases: `Principal`, `Interno1`, `interno2`, `Interno3` y `comunica`. A continuación se describe más a detalle:

- clase `Principal`: se encarga de crear la ventana principal de la interfaz gráfica, define un contenedor de elementos gráficos, en dicho contenedor se han agregado tres botones de acción que invocan los métodos para crear más ventanas, dependiendo del botón seleccionado

- clase `Interno1`: esta clase es invocada para dibujar una ventana dentro del contenedor de la clase `Principal`, también se encarga de invocar a la clase `comunica` para poder enviar un mensaje al grupo de usuarios que se esté representando

- clase `Interno2`: realiza funciones similares a la clase `Interno1`, invoca la clase `comunica` para poder enviar un mensaje al grupo de usuarios que se esten representando.

- clase `Interno3`: tambien crea una ventana dentro del contenedor de la clase `Principal`, invoca a la clase `comunica` para poder enviar mensajes a los grupos de usuarios que se esten representando.

- la clase `comunica`: se encarga de editar, filtrar y enviar el mensaje.

En la Fig 4.3 se observa el diagrama de clases que se ha implementado para el desarrollo de la interfaz gráfica.

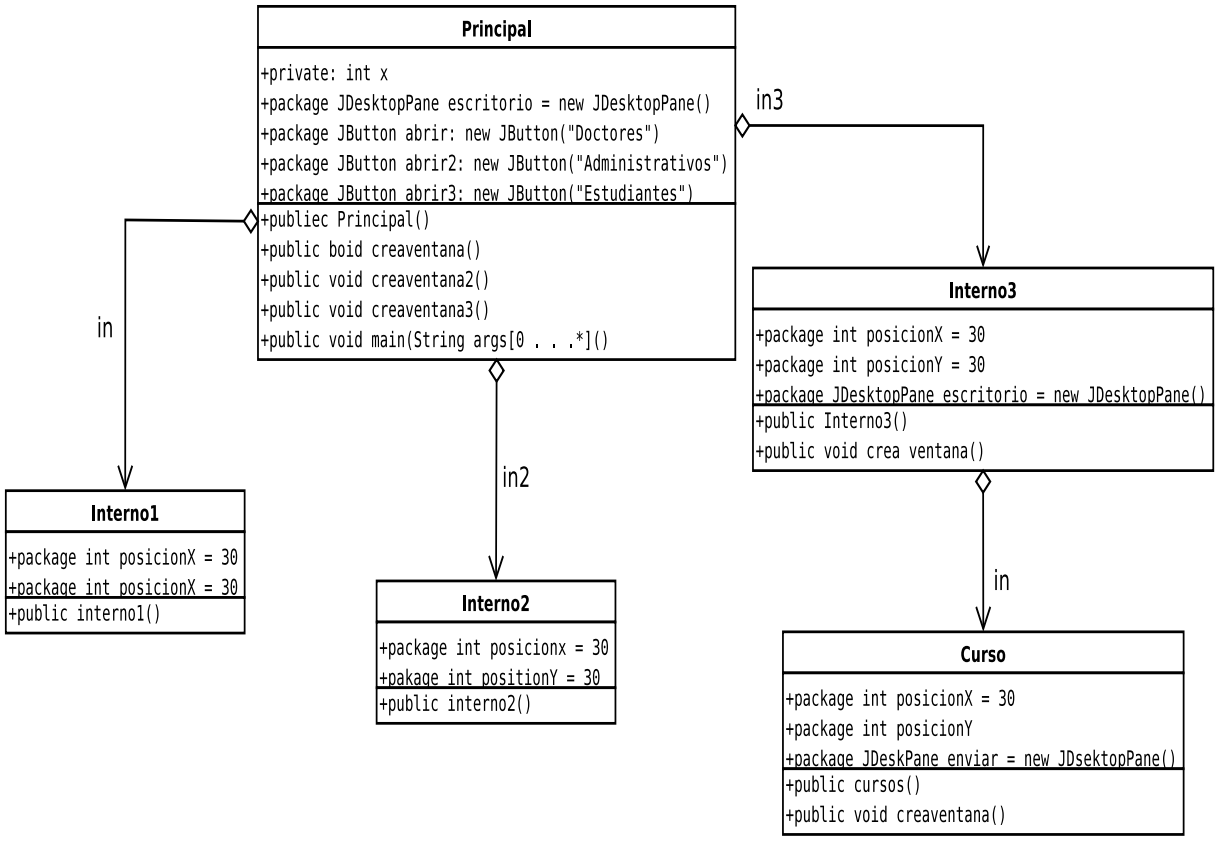

Figura 4.3: Diagrama de clases de la interfaz gráfica

Una vez conocidas las clases, el servidor inicia al ejecutarse la clase `Principal`, la cual implementa la clase `JFrame`, dentro de ésta se define el contenedor y los componentes de la interfaz gráfica. En el constructor de la función se especifican las dimensiones de la ventana, la posición de los botones de acción y se agregan los componentes al contenedor `JDesktopPane`. Otro punto importante de esta clase es que aquí se invoca la clase `Servidor`, el cual se encarga de los mecanismos de comunicación. En el siguiente fragmento de código se observan los elementos de la clase:

```
public class Principal extends JFrame{
public Principal (){...}
abrir.addActionListener(new ActionListener){...}
public void crearventana()
    }
```

El método `addActionLisetener` se encarga de escuchar los eventos que se producen en los `JButton`. Su función principal es responder al evento que se dispara cuando el usuario da clic con el mouse. Cada uno los botones de acción llaman a la función encargada de dibujar una ventana interna, esto se lleva a cabo a través de la invocación de la función `crearventana`.

La función `crearventana` crea un objeto para invocar la clase que se encarga de dibujar una ventana dentro del contenedor de `Principal`. Esta función se presenta en cada botón de la aplicación.

En las ventanas internas puede programarse más invocaciones para crear otras ventanas, las cuales pueden corresponder a los subgrupos de usuarios que existan en la organización. Dentro de las clases que generan las ventanas internas nuevamente se define el constructor, el método `ActionListener` y, cuando se requiera, la función `crearventana` para realizar las mismas funciones descritas anteriormente.

La clase *comunica* también es invocada para aparecer internamente dentro del contenedor de la clase *Principal*, contiene los mecanismos de selección de la categoría, la edición del mensaje y el envío del mismo. En la Fig 4.4 se puede apreciar la secuencia de la interacción entre los objetos de la clase *comunica*. A continuación se describe cada uno de los elementos:

- selección de la categoría: por medio de una consulta a la B.D., se visualiza en una lista, las categorías que corresponden a las preferencias de los mensajes. Los elementos que se muestran en las listas cambian dependiendo de la clasificación del perfil de usuario,

- edición del mensaje: se proporciona una caja de texto para que el administrador del servidor edite el mensaje que desea enviar,

- enviar mensaje: Por medio de un botón de acción se invoca al método encargado de enviar el mensaje al mecanismo de filtración, antes de ser enviado al grupo de usuarios.

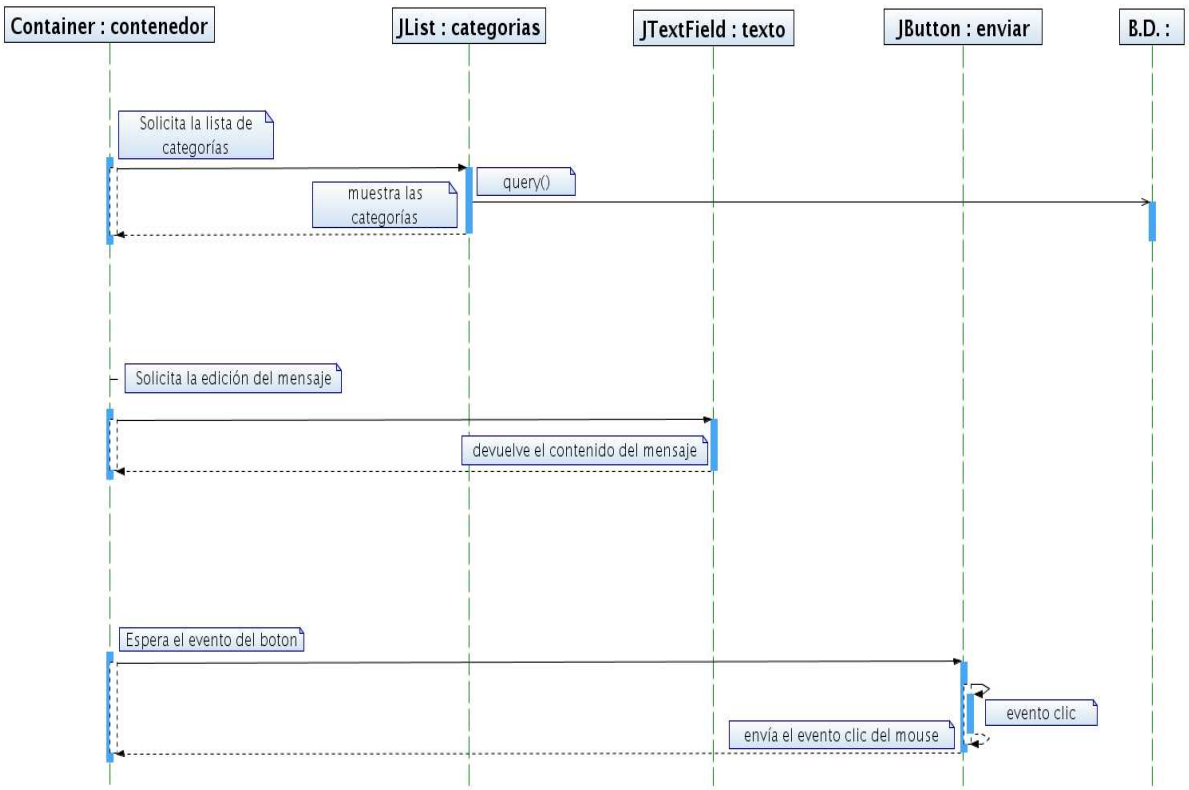

Figura 4.4: Secuencia de los objetos de la clase *comunica*

Antes de enviar el mensaje, en el servidor se lleva a cabo un filtrado sobre las preferencias de los usuarios, *i.e.*, se encarga de analizar las preferencias de los clientes conectados. Este procedimiento inicia cuando un cliente se conecta al servidor, los datos de autentificación son recibidos y analizados con el objetivo de crear un *hilo* para cada *socket*, generado por la conexión del cliente.

Al recibir una conexión entrante en el servidor, se crea una instancia al *hilo* y se le agrega el *socket* como parámetro. Las instancias del *hilo* se almacenan en un vector de objetos, el cual es muy importante en el proceso del filtro de destinatarios. Cuando se realiza la instancia para ejecutar el *hilo* se llevan a cabo dos procedimientos inicialmente:

- autentificación: por medio de la función `autentica`, se realiza una consulta a la base de datos, donde se verifica que el usuario ya esté registrado. Además se modifica la tabla *conectado* de la B.D., para indicar que el usuario está en línea. También

se realiza otra consulta a la B.D. para identificar las preferencias marcadas por el cliente, éstos datos son almacenados en un arreglo global de preferencias (ver el procedimiento 8).

- filtro: a través de la función `getUser`, se analiza el arreglo de preferencias marcadas por el usuario. Esta función se llama cada vez que el administrador del servicio dispara el evento clic al seleccionar el botón de la ventana de edición del mensaje.

Para enviar el mensaje a través del *socket* de comunicación, se invoca la función `enviaMensaje`. El objetivo es recorrer todo el arreglo de instancias de tipo *hilo* que representa a cada cliente conectado.

Procedimiento 8 Proceso de entrada al servidor de mensajes

Entrada: `mensaje`, *datos*, `subcad`
Salida: `estado`, *preferencias*[]

```
1: i = 0
2: mientras (tokens.hasMoreTokens()) hacer
3:   subcad = tokens.nextToken();
4:   datos[i] = subcad
5:   i++
6: fin mientras
7: Query(usuario,contraseña)
8: si (Query=1) entonces
9:   Query2(usuario, online)
10:  estado = online
11:  rs = Query3(usuario)
12:  mientras (rs.next() hacer
13:    preferencias[] ⇐ prefer
14:  fin mientras
15: fin si
```

Antes de enviar el mensaje se verifica si en el usuario ha marcado como preferencia el tipo de mensaje que se va a enviar (ver el procedimiento 9).

Procedimiento 9 Función *autentica2*

Entrada: *preferencias*[], `prefer`
Salida: `bandera`

```
1: j=0
2: para (j=0, j≤ preferencias.size(), j++) hacer
3:   si prefer = preferencias[j] entonces
4:     bandera=true
5:   si no
6:     bandera=false
7:   fin si
8: fin para
```

Para realizar la verificación de las preferencias se llama a la función `autentica2`, pasándole la preferencia como parámetro. La función devuelve *true* (verdadero) si la preferencia se encuentra registrada en el arreglo de preferencias, de lo contrario devuelve *false* (falso).

Una vez que la función `autentica2` devuelve el valor de *bandera* en *true*, se invoca la función `enviamosMensaje` agregando el mensaje como parámetro, el cual fue editado por el administrador. Dentro de la función `enviamosMensaje` se tiene el flujo de salida que viaja a través del *socket*, ahí se agrega el mensaje recibido como parámetro e inmediatamente se envía por la red. Si el valor de *bandera* es *false*, significa que el usuario no marcó como preferencia el tipo de mensaje que se va a enviar, por lo tanto se recorre a la siguiente instacia del *hilo* almacenada en el vector de objetos. De esta manera solo se envía el mensaje a un grupo selecto de clientes, en base a sus preferencias marcadas en la edición del perfil de usuario. Lo descrito anteriormente se puede apreciar en el **procedimiento 10**.

Procedimiento 10 Enviar el mensaje

Entrada: *vector*[]

```
1: i=0
2: para (i=0, i≤ vector.size(), i++) hacer
3:   h=(hilo)vector.get(i)     // invocación al primer elemento del vector
     correspondiente a la conexión de un cliente
4:   si (h.autentica2) entonces
5:     h.enviamosMensaje            // llamda a la función enviamosMensaje
6:   fin si
7: fin para
```

Mecanismo automático de eventos

En el servicio de mensajes se implemento una clase especial, que se encarga de enviar mensajes de manera automática. Se trata de un mecanismo que responde a la ocurrencia de un evento, el cual se dispara dependiendo del resultado de una consulta a la tabla *eventos* de la BD.

La consulta que se hace a la tabla *eventos* tiene el objetivo de comparar la fecha actual con las fechas almacenadas, que hacen referencia al día en el cual se llevara a cabo una actividad preprogramada.

El mecanismo para el envío automático de mensajes, se inicia al ejecutarse el servidor de mensajes. Una vez iniciado el servidor se ejecuta un objeto de tipo *Timer*, el cual invoca a la clase *EventoX*. El objeto de tipo *timer* ha configurado para que invoque a la función *EventoX* diariamente, a una hora específica.

Para invocar la clase *EventoX* a una hora exacta, en la configuración del objeto, se invoca a una función que realiza el cálculo de los tiempos, los milisegundos los transforma a horas y posteriormente, las horas resultantes se envian como parámetro a la configuración del objeto *timer*.

La clase EventoX es la que se encarga de realizar la consulta en la tabla *eventos* de la BD, verifica las fechas de las actividades que se tienen registadas y las compara con la

fecha actual, si alguna coincide se toma el nombre de la actividad y se envía como mensaje a los usuarios. La clase se puede apreciar en el **procedimiento 11**.

Procedimiento 11 Clase **EventoX**

Entrada: conn, día, mes annio, even, fecha, qry

```
1: conn = (Connection) DriverManager.getConnection(ruta, userBd,
   userBdPass);
2: dia = Integer.toString(c.get(Calendar.DATE));
3: mes = c.get(Calendar.MONTH) + 1;
4: annio = Integer.toString(c.get(Calendar.YEAR));
5: fecha = annio + "-" + mes + "-" + dia;
6: qry = "select count(*) from eventos where fecha =" + "'" + fecha +
   "'";
7: rst = stmt.executeQuery(qry);
8: si rst.getInt(1) != 0 entonces
9:   qry="select evento from eventos where fecha =" + "'" + fecha + "'";
10:  rst = stmt.executeQuery(qry);
11:  mientras rst.next() hacer
12:    even =rst.getString("evento");
13:    ser.enviaMensaje(even);
14:  fin mientras
15: fin si
```

La clase EventoX, primeramente realiza la conexión con la BD, posteriormente obtiene el día, el mes y el año para formar la fecha actual. Después realiza la sentencia consulta para verificar si existe algún registro en la BD que tenga el campo *fecha* almacenado. Si la consulta indica que se han encontrado registros con las especificaciones mencionadas, se realiza una nueva sentencia para obtener el nombre de la actividad que se tiene programada en la fecha actual.

El nombre de la actividad se utiliza para dar formato al mensaje que se envía a los usuarios. Para recibir los mensajes generados por el mecanismo automático basado en eventos, el usuario debió marcar alguna de las preferencias extra al momento de editar su perfil, lo que significa que el mensaje generado por el manejador de eventos también es analizado por el filtro de mensajes.

Con el mecanismo automático de envío de mensajes y los eventos producidos por el administrador del servidor se complementa el servicio de mensajes implementado.

4.3 Implementación de la base de datos

Para la implementación física de la base de datos se siguió el proceso descrito en la sección 3.6, dando como resultado doce tablas. A continuación se describe el proceso que se siguió para crear las tablas de la base de datos, así como también se mencionan las herramientas que se utilizaron.

En primer lugar, era necesario contar con una herramienta que nos facilitara el diseño del diagrama EER, donde claramente se identifican cada una de las tablas que integran

la BD. Al elegir *MySQL* como el sistema gestor de bases de datos, se analizaron varias herramientas que permiten construir el diseño del diagrama EER y al mismo tiempo crear las tablas. La herramienta seleccionada fue *MySQL Workbench*, se trata de una ambiente gráfico que, a través de una interfaz, permite crear cada una de las tablas de una BD.

Antes de construir cada tabla fue necesario realizar un análisis para especificar las restricciones que cada tabla debe tener. La herramienta *Workbench* permite agregar restricciones con solo seleccionar el objeto que representa la restricción y trazar una línea uniendo a las tablas involucradas en la restricción. Para que las restricciones se pudieran crear con éxito fue importante indicar los campos llave y campos foráneos de cada una de las tablas

El proceso para la implementación de las tablas fue relativamente sencillo, solo se presentaron algunos problemas al momento de añadir tablas cuando ya se había creado la base de datos físicamente. En esta caso se podía elegir entre dos opciones:

1. agregar la tabla al diagrama EER desde la herramienta *Workbench*, el problema se presentaba cuando ya se contaba con varios registros almacenados en la BD, si se volvía a exportar el diagrama para crear las tablas físicas a partir de él, los registros almacenados se borrarían automáticamente.

2. la otra opción fue usar una herramienta que nos permitiera administrar la BD cuando ya estaba creada, la herramienta seleccionada para esta tarea fue *MySQL Query Browser*, permite gestionar una BD a través de una interfaz gráfica donde principalmente se realizan consultas, inserciones, modificaciones, entre otras cosas. La mayoría de las operaciones se realizan utilizando el lenguaje SQL a diferencia de la herramienta Workbench, aunque para crear las tablas se presenta una ventana donde se debe escribir cada campo con su respectivo tipo de dato.

En la Fig 4.5, se puede apreciar la vista de una tabla desde la herramienta *Workbench*, en este caso se puede observar el campo llave de la tabla, y la especifición de los tipos de datos que representa cada campo que la compone.

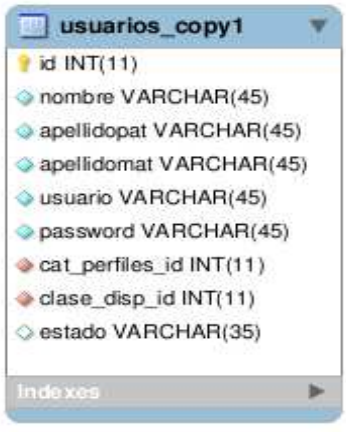

Figura 4.5: Vista de la tabla usuarios desde *Workbench*

El acceso a los datos se realiza desde el propio servicio a través de la implementación de consultas. Si se desea realizar un cambio directo en la BD se debe usar la herramienta M*ySQL Query Browser*. Las consultas que se realizan desde el sistema son básicamente las sentencias de selección, inserción y modificación de datos del lenguaje *SQL*.

Una de las sentencias más significativas utilizadas por el servicio de registro es una consulta que involucra a dos tablas. Para que la consulta se ejecute con éxito se utiliza una sintaxis de consulta anidada, con esto puede involucrarse a dos a más tablas, siempre y cuando las tablas esten relacionadas a través de una restricción.

4.4 Caso de estudio

Dentro del departamento de computación del Cinvestav cualquier tipo de aviso o recordatorio se envía a través de correo electrónico. En ocasiones estos tipos de *mails* contienen muy pocas líneas de texto, debido a que son mensajes que se envían pensando que los integrantes del departamento están revisando la bandeja de entrada constantemente.

En el departamento, las personas que lo conforman se han clasificado en tres grupos a los que les llamaremos grupos de usuario. La clasificación especifíca al grupo *Doctores*, grupo *Administrativos* y grupo *Estudiantes*, los cuales son representados por un botón dentro de la clase `Principal` correspondiente a la interfaz de usuario del administrador del servidor.

La ventana más significativa es la que representa al perfil *Estudiantes*, en ella se basa la implementación central del servicio de mensajes. Dicha ventana se crea al invocar la clase *Interno3*, sus métodos y funciones realizan prácticamente las mismas operaciones que la clase `Principal`, *i.e.*, generan una ventana interna dentro del contenedor de la clase. La interfaz gráfica de `Interno3` visualiza tres botones de acción los cuales representan una subclasificación del perfil general *Estudiantes*, éstos son: *cursos*, *Tesista_maestria* y *Doctorante*.

En el grupo *Estudiantes* la subclasificación de tres subgrupos se refiere a los estudiantes de primer año de maestría, los cuales toman los cursos que se imparten en el departamento; los alumnos tesistas de maestría, que se encuentran en su segundo año, toman solo un curso y eventualmente entran a otros curso de oyentes; los alumnos doctorantes, toman solo un curso y en ocasiones entran a otros cursos para fortalecer algún conocimiento o por ordenes de su asesor.

De acuerdo a la descripción anterior, la aplicación Web ha sido implementada para dar el servicio de registro, contemplando los grupos de integrantes del departamento de computación. El resultado se observa en la Fig 4.6, que muestra el mapa del sitio Web.

Nos interesamos más por el caso del grupo *estudiantes*, debido a que la mayor parte de la disfusión de información dentro del departamento está dirigida especialmente a los alumnos.

Ahora, al incluir mensajes de texto, no solo se puede enviar información acerca de las conferencias y otros eventos, también se pueden enviar mensajes que, de cierto modo necesitan ser vistos con urgencia, *e.g.*, enviar mensajes con información de algún curso, avisos de cambios de horario de alguna clase, recordatorio de fechas de entrega de proyectos, avisos especiales de parte de la coordinación académica, entre otros.

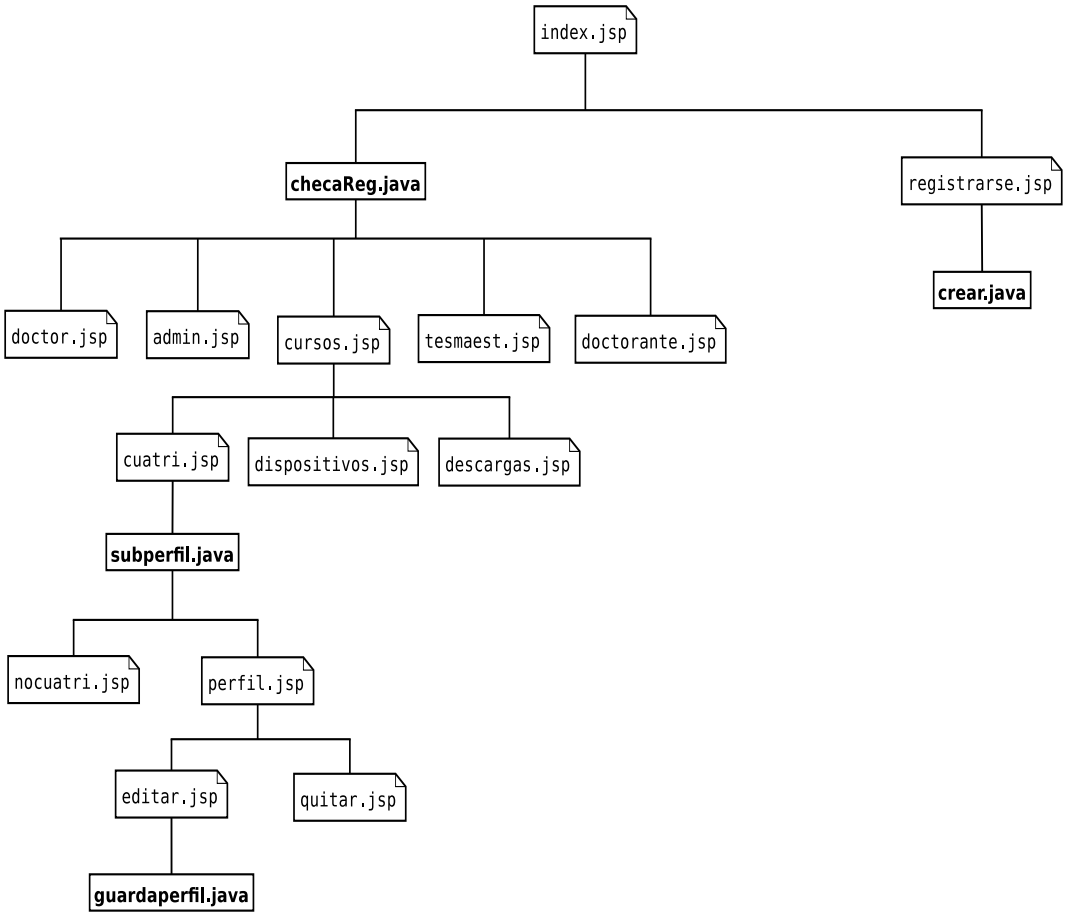

Figura 4.6: Mapa del sitio-servicio de registro de usuarios

Al enfocarnos en el caso *estudiantes*, dentro de la BD se han generado tablas que tiene que ver con la información de los estudiantes, la cual es necesaria para la implementación del servicio de mensajes. Las tablas involucradas son las siguientes:

- tabla *Usuarios*: en ella se almacena la información correspondiente al registro del usuario, el perfil general y la clase de dispositivo que utiliza el usuario,
- tabla *mat_Estud*: en esta tabla se almacenan las materias que representan las preferencias de mensajes, los estudiantes en cursos de primer año solo le interesan los mensajes que tengan que ver con las materias que esta tomando,
- tabla *preferencias_1*: registra otras preferencias que el usuario puede elegir. Algunas de ellas son: eventos, conferencias o invitaciones a exámenes de grado,
- tabla *materias* almacena todos los cursos que se imparten en el transcurso del primer año de la maestría,
- tabla *alta_materias* contiene el registro de las materías dadas de alta en el cuatrimestre actual,

- tabla *clase_disp*: es un catálogo donde se generalizan tres clases de dispositivos móviles: bajo, intermedio y avanzado,
- tabla *usua_cuatri*: almacena el *id* del usuario y el *id* del cuatrimestre que está cursando el estudiante,
- tabla *conectado*: almacena el *id* del usuario y su estado, *i.e.*, indica si está en línea o no.

Los datos para llenar las tablas se obtienen a través del sitio Web al momento del registro y al editar el perfil. Tomando en cuenta un determinado grupo de usuario, en este caso son los estudiantes del departamento, existen tres tipos; los alumnos que están en primer año (en cursos), los alumnos en tesis de maestría y los alumnos doctorantes. Cada uno tiene un perfil distinto, por lo tanto las preferencias que puede elegir también varían un poco, A continuación se presenta el **procedimiento 12**, donde se observan los pasos que un alumno de primer año sigue para editar su perfil, agregando sus preferencias.

Procedimiento 12 Pasos para editar perfil y agregar preferencias

1: seleccionar la opción *Perfil* de la página *cursos*
2: seleccionar el subperfil de la página *cuatri*
3: seleccionar la opción agregar de la página *edición*
4: seleccionar las casillas correspondientes al tipo de mensajes se desea recibir
5: enviar los datos al servidor

El servidor realiza varias operaciones en cada paso que se observa en el **procedimiento 12**, en el caso del **paso 2** el servidor realiza lo siguiente:

- el servidor realiza una consulta a la BD para verificar si el cuatrimestre que se seleccionó está vigente,
- si no está vigente redirecciona una página, informando al usuario a cerca del estado del cuatrimestre,
- si el cuatrimestre está vigente realiza una nueva consulta la BD, con el objetivo de saber si el usuario está editando su perfil por primera vez,
- si el usuario está editando por primera vez su perfil, debe ingresar a la BD un registro indicando el *id* del usuario y el cuatrimestre vigente.

En el **paso 4**, las preferencias se presentan en forma de varias casillas, donde el usuario elige las de su interés y posteriormente las envía al servidor para hacer el registro en la BD.

La página Web que muestra las casillas con los nombres abreviados de las materias, también coloca unas casillas más, que representan otras preferencias, específicamente son: conferencias (información acerca de las conferencias que se van a dar en el departamento), eventos (información acerca de reuniones, celebraciones, etc) y examenes (invitaciones a los exámenes de obtención de grado).

Toda esta información es obtenida como parámatros por el servidor. En el **procedimiento 13** se observa el proceso de registro de las preferencias. En el **paso 4** se realiza una comparación para diferenciar las preferencias que corresponden a las materias y las que preferencias extras. Esto se lleva a cabo para poder almacenar la información en la tabla que le corresponde. En el **paso 13**, se almacena en la BD los datos correspondientes al identificador del usuario y al identificador de la materia. En el **paso 24**, se registran en la BD las preferencias extras que se han marcado.

Procedimiento 13 Registrar preferencias

Entrada: *datos*[],*materias*[], *prefer*[], i, j, id_materia, cadena, elemento, conf, event, exa

```
1: datos[] ← parametros
2: mientras (datos.Masparametros) hacer
3:    cadena = dato.nextparametro
4:    si (cadena = conferencia) o (cadena = eventos) o (cadena = examen ) entonces
5:       prefer[] ← cadena
6:    si no
7:       materias[] ← cadena
8:    fin si
9: fin mientras
10: id_usuario= query(usuario)
11: para materias[i]hastamaterias[i++] hacer
12:    id_materia= query(materias[i])
13:    update(id_usuario,id_materia)
14: fin para
15: para prefer[j] hasta prefer[j++] hacer
16:    elemento = prefer[j]
17:    si elemento = conferencia entonces
18:       conf = 1
19:    si no, si elemento = eventos entonces
20:       event = 1
21:    si no
22:       exa = 1
23:    fin si
24:    update(conf, event, exa)
25: fin para
```

El escenario más común se presenta cuando el administrador del servicio de mensajes selecciona, a través de una interfaz gráfica, un grupo de usuarios a los cuales se les enviará un mensaje corto, *e.g.*, algún aviso, recordatorio, invitación a un exámen de grado, información de alguna conferencia, etc. Solo los usuarios que hayan especificado su preferencia por el tipo de información que se va enviar, van a recibir el mensaje en su dispositivo.

Además de recibir mensajes enviados por el administrados del servicio, el usuario recibe mensajes que son generados automáticamente por el servidor. La información enviada de

forma automática es referente a las conferencias o actividades que se van a realizar en el departamento de computación. Esto es gracias a la implementación de un mecanismo de envío de mensajes basado en eventos.

Los mensajes se envían al dispararse un evento, el cual ocurre cuando se analiza la tabla *eventos* de la BD, donde se compara la fecha actual con las fechas de las actividades registradas, si las fechas son iguales, se notifica a los usuarios que marcaron como preferencia éste tipo de información.

4.5 Pruebas y resultados

En las secciones anteriores se ha descrito detalladamente la implementación, los componentes y el funcionamiento del servicio SEGEMENS. Ahora toca el turno de presentar las pruebas realizadas para validar el servicio y también los resultados obtenidos en el desarrollo de la esta tesis.

Las pruebas realizadas tienen como objetivo comprobar el funcionamiento de todo el servicio, confirmar los requisitos que se definieron y analizar la interacción con el usuario.

Primeramente se presenta el caso de estudio aplicado al servicio SEGEMENS, posteriormente se describen las pruebas realizadas y los resultados obtenidos. Al final de esta sección se presentan las interfaces gráficas utilizadas por el usuario, las cuales corresponden al sitio Web y al servicio de mensajes cortos.

4.5.1 Infraestructura

La etapa de pruebas en los sistemas o servicios aplicativos son muy importantes, por lo tanto se definió una infraestructura necesaria de *hardware* y *software* para poner en práctica la solución desarrollada. Las pruebas que se realizaron al servicio SEGEMENS se hicieron dentro de la red privada *labwless2*, usando el protocolo TCP/IP para el servicio de mensajes cortos y el protocolo HTTP para el servicio Web de registro de usuarios y dispositivos.

En el servicio SEGEMENS se definieron los equipos de cómputo así como también se eligieron tres dispositivos móviles para realizar las pruebas. Para el servicio de mensajes cortos y el sitio Web se dispuso de una *laptop* marca Dell, modelo *inspiron* 1501. Los dispositivos móviles utilizados para acceder al sitio Web, se muestran el la tabla 4.1, donde tambien se especifican algunas de sus características.

Dispositivo	(SMS, MMS, e-mail)	Acceso a Internet	Sistema Operativo
Nokia N95	Si	Si	Symbian OS v9.2 (s60)
iPAQ 610c	Si	Si	Windows Mobile 6.0
Nokia E65	Si	Si	Symbian OS (s60)

Tabla 4.1: Dispositivos móviles

En la Fig 4.7 se pueden apreciar los dispositivos móviles utilizados en las pruebas del servicio. Estos dispositivos tienen características heterogéneas, tanto en el hardware

como en el software, *i.e.*, cuentan con diferentes sistemas operativos, tamaños de pantalla distintos y diferentes fabricados por diferentes compañias.

Figura 4.7: Dispositivos móviles

El equipo de cómputo donde se instaló el servidor de mensajes y el sitio Web tiene las siguientes características:

- procesador AMD Turion(tm) 64X2 Mobile Technology TL-56 1.8 Ghz,
- memoria Ram de 2 Gb,
- sistema operativo Linux Ubuntu 8.04 (hardy),
- interfaz inalámbrica Broadcom IEEE 802.11g.

4.5.2 Pruebas

Las pruebas realizadas al servicio SEGEMENS se concentran en la conectividad entre los dispositivos móviles y el servidor de mensajes, además se realizan una serie de pruebas para corroborar la eficiencia del sitio Web. Para llevar a cabo cada una de las pruebas se siguió un orden específico, las siguientes pruebas se realizaron al sitio Web orientado a los dispositivos móviles:

- conexión con el sitio Web: esta prueba consistió en realizar la conexión con el sitio Web, escribiendo la *url* en el navegador de Internet del dispositivo. Generalmente la conexión fue exitosa, cuando sucedía lo contrario, el servidor no estaba ejecutándose,
- visualización del sitio Web: debido a que el sitio está orientado a dispositivos móviles, se verificó que la visualización de las páginas del sitio se acoplaran a la pantalla del dispositivo,

- proceso de registro: una vez hecha la conexión con la página inicial del sitio Web, se inició el proceso de registro del usuario y el registro del dispositivo móvil. La prueba consiste en comprobar que los datos proporcionados por el usuario sean correctos, además se tiene que confirmar el registro tanto del usuario como del dispositivo móvil,

- proceso de entrada al sitio Web: se introdujeron el nombre de usuario y la contraseña, posteriormente se enviaron al servidor para ser analizados. También se analizaron las excepciones que se generaron, *e.g.*, cuando un usuario dejó un cuadro de texto vacío,

- visualización de consultas: ocasionalmente el sitio Web visualiza algunas opciones en base a ciertas consultas a la BD, *e.g.*, cuando el usuario desea editar su perfil, primero debe especificar el cuatrimestre vigente, de acuerdo a esto, se realiza una consulta a la BD, para saber cuales son las preferencias dadas de alta para el cuatrimestre especificado. Una vez encontradas las preferencias de mensajes se envían al usuario a través de una página Web,

- agregar dispositivo: esta prueba consistió en agregar un nuevo dispositivo para usarlo en el servicio de mensajes,

- descargar aplicación *cliente*: después de agregar un dispositivo más, se tiene que descargar la aplicación, *cliente*. La prueba realizada consistió en descargar la aplicación y ejecutarla en el dispositivo móvil.

Las pruebas que se realizaron al servicio de mensajes se describen a continuación:

- conectividad: la prueba consistió en examinar el comportamiento del servidor al momento de la conexión de varios clientes. El servidor es capaz de atender a varios clientes al mismo tiempo, por lo tanto, no se deben presentar problemas en las conexiones.

- comunicación: se probó el envío y recepción de mensajes entre el servidor y los dispositivos móviles. Inicialmente se preparó al servidor para enviar una cadena de caracteres a todos los clientes conectados. Después se realizó la misma prueba pero ahora el mensaje se envío a un grupo específico de clientes.

- interfaz servidor: se realizaron pruebas de funcionamiento a la interfaz gráfica del servidor, el objetivo fue examinar la navegación entre las ventanas que se desarrollaron. La fase más importante de esta prueba se enfocó en la ventana de edición del mensaje, se verificó que se visualizaran correctamente las distintas categorías o tipos de mensajes que se pueden enviar y que el mensaje editado fuera enviado correctamente a los clientes.

- aplicación *cliente*: en este caso se probó que el proceso de entrada al servicio se llevara a cabo correctamente, se verificó que las excepciones funcionaran tal y como se programaron. Además realizaron pruebas con la interfaz de recepción de los mensajes, el objetivo fue que la información se mostrára correctamente a través de las alertas implementadas.

- pruebas de portabilidad: para la aplicación *cliente* se realizaron pruebas de portabilidad, *i.e.*, la aplicación debe ser capaz de ejecutarse si problemas en los diferentes dispositivos móviles. El requerimiento principal para que esta prueba sea exitosa es que se cuente con una maquina virtual de Java instalada en los dispositivos.

4.5.3 Resultados

El resultado de la implementación ha generado tres aplicaciones, la aplicación cliente, la aplicación servidor y la aplicación Web. A continuación se describe cada una de las aplicaciones que se desarrollaron, se explica su funcionamiento y la aplicabilidad que se les dio en el caso de estudio descrito en la sección 4.8.1.

Aplicacion Web

La aplicación Web está conformada por dos funciones, el registro de usuarios y dispositivos móviles, y la edición del perfil de usuario. En la figura 4.8 se observa la página de inicio del sitio Web. Para poder registrar un nuevo usuario se debe seleccionar el botón *registrarse*.

Figura 4.8: Vista de la página de inicio del sitio Web

Ahora toca el turno de ingresar algunos datos correspondientes al registro, los datos que se deben escribir son: el nombre, apellido paterno, apellido materno, nombre de usuario, contraseña, confirmación de la contraseña, el perfil general de usuario y la clasificación del dispositivo. La página para introducir los datos se puede apreciar en la Fig 4.9.

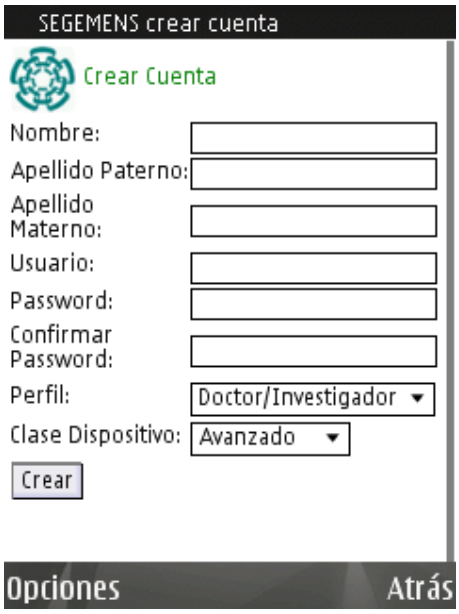

Figura 4.9: Entrada de datos

El registro del dispositivo móvil se realiza de forma transparente para el usuario, si la respuesta del sitio Web es la página de confirmación del registro del usuario quiere decir que la operación ha sido realizada exitosamente. Una vez registrado el usuario el control vuelve a la página principal del sitio, ahora toca el turno del proceso de entrada al sitio Web (ver Fig 4.10) para iniciar con la edición del perfil de usuario.

Figura 4.10: Entrada de datos desde un *Smartphone* iPAQ 610

Después de validar la entrada, se visualiza la página con las opciones que se proporcionan para editar el perfil, agregar un nuevo dispositivo o descargar la aplicación cliente

desde un dispositivo recién agregado. En la Fig 4.11 se muestran las opciones mencionadas anteriormente.

Figura 4.11: Vista de la página de inicio del sitio Web

Para editar el perfil, el usuario selecciona el enlace que lo lleva a la página para agregar o quitar una preferencia. Al seleccionar la opción *agregar*, para el caso del perfil *Estudiantes*, se solicita que especifique el cuatrimestre vigente tal como se observa en la Fig 4.12.

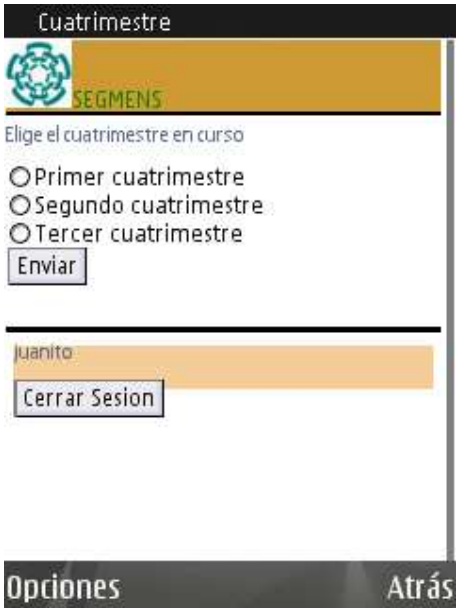

Figura 4.12: Seleccionando el cuatrimestre actual

Siguiendo con el perfil *Estudiantes*, en la Fig 4.13 se observan las preferencias que el usuario puede seleccionar, las cuales principalmente se refieren a las materias (ver inciso

(A) de la Fig 4.13)que se están cursando o también se pueden recibir mensajes referentes a los eventos, conferencias e invitaciones a exámenes de grado (ver inciso (B) de la Fig 4.13) que se llevan a cabo en el departamento de computación.

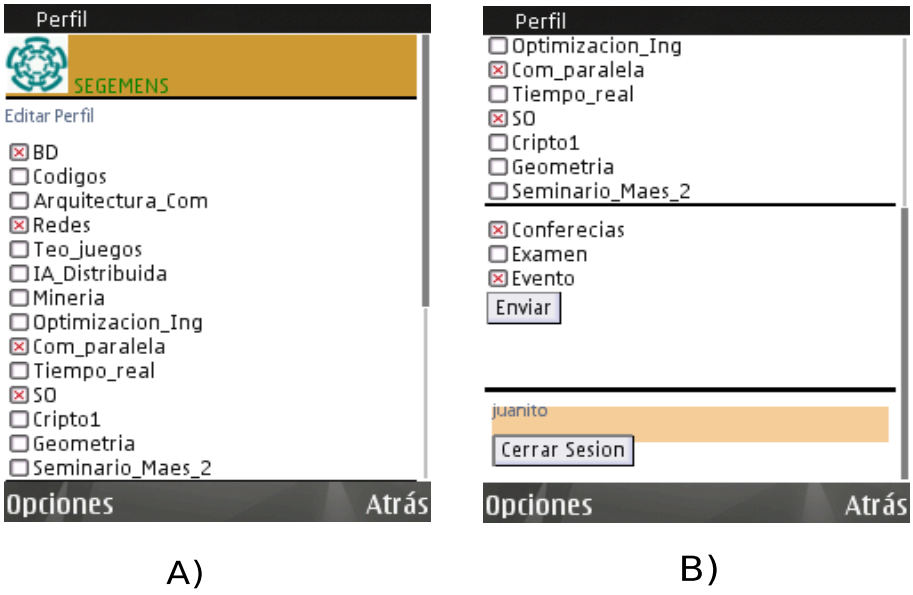

Figura 4.13: Seleccionando las preferencias

Servidor de mensajes cortos

El servidor que se encarga de proporcionar una interfaz al administrador del servicio para enviar los mensajes a un grupo determinado de usuarios. La interfaz principal se muestra en la Fig 4.14, en ella se muestra tres botones de acción que representan los perfiles de usuario generales.

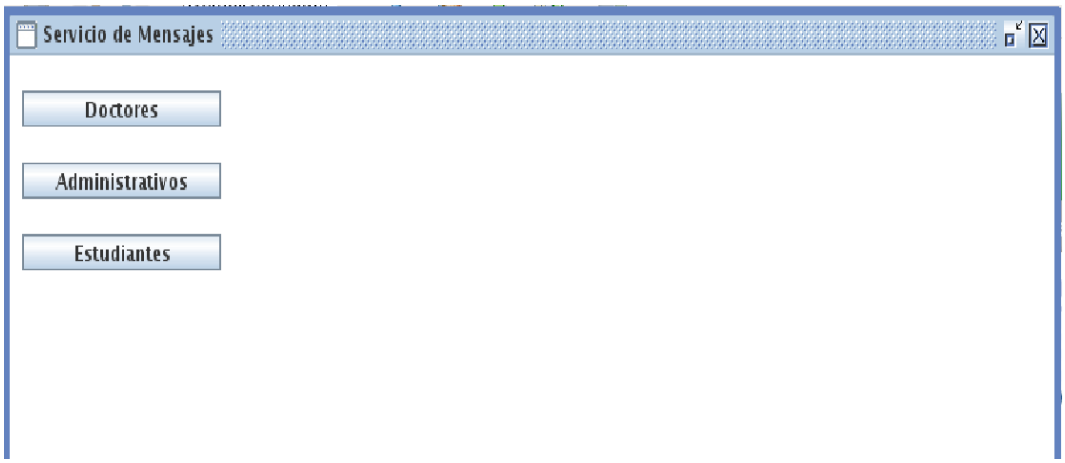

Figura 4.14: Interfaz gráfica del servicio de mensajes

De acuerdo al perfil *Estudiante*, al dar clic en el botón correspondiente se genera una ventana interna, como se observa en la Fig 4.15, donde nuevamente aparecen tres botones

de acción los cuales representan a los estudiantes en cursos, los estudiantes en tesis de maestría y los estudiantes doctorantes. En este caso se desea enviar un mensaje a los estudiantes en cursos, por lo tanto selecciona el botón *cursos* de la ventana interna.

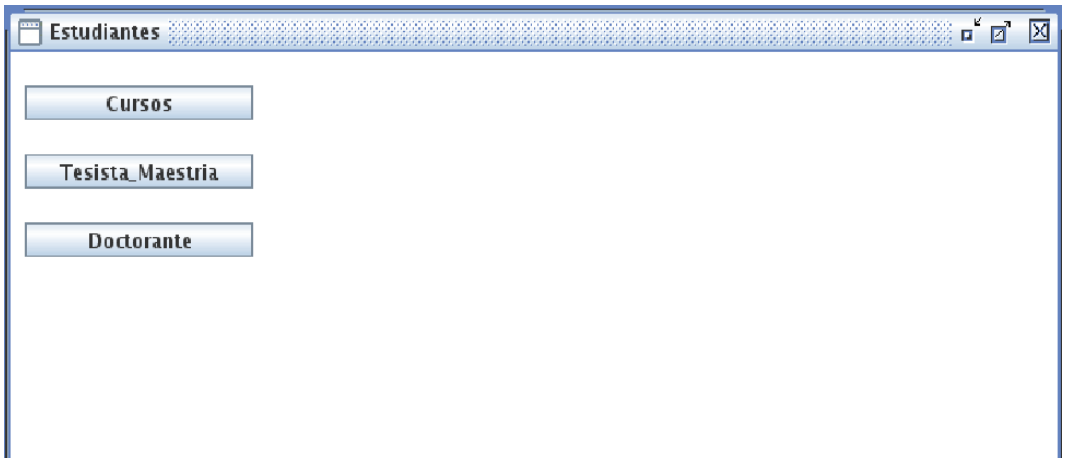

Figura 4.15: Ventana de del perfil *Estudiantes*

La acción anterior genera una nueva ventana interna la cual muestra una lista con los distintos tipos de mensajes que pueden ser enviados a los alumnos en cursos. Además se proporciona una caja de texto para editar el mensaje que se va a enviar y el botón que se encarga de disparar la orden de enviar el mensaje. En la Fig 4.16 se puede apreciar como el administrador del servicio de mensajes edita un mensaje dirigido a los alumnos en cursos que toman la materia de redes de computadoras.

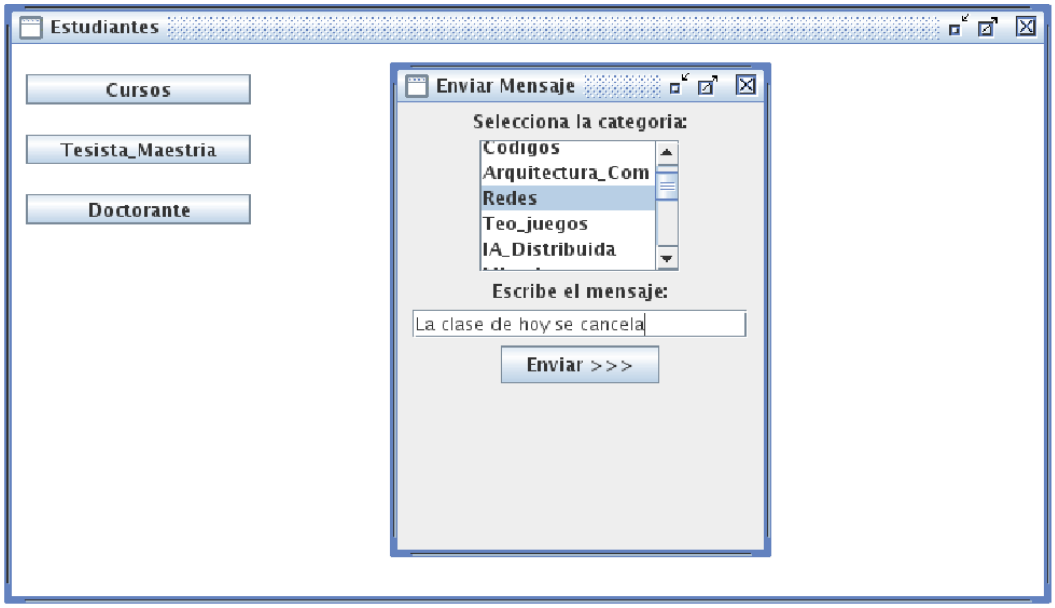

Figura 4.16: Ventana de edición del mensaje

Aplicación *cliente*

Una vez instalada la aplicación *cliente* en el dispositivo móvil, la pantalla que se muestra en la correspondiente al proceso de entrada al servicio (ver Fig 4.17), ahí se debe escribir el nombre de usuario y contraseña correctamente.

Figura 4.17: Interfaz gráfica de la aplicación *cliente*

Después de entrar correctamente al servicio de mensajes, en la Fig 4.18 se observa la pantalla que se encuentra esperando los mensajes enviados por el servidor.

Figura 4.18: Interfaz para la recepción de los mensajes

Al recibir un mensaje se muestra el mensaje que envió el servidor en la pantalla como se muestra en la Fig 4.19.

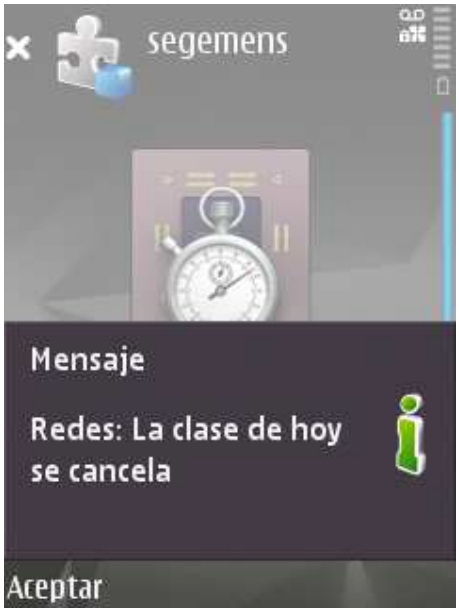

Figura 4.19: Visualizando el mensaje

Visualización de mensajes automáticos

En la Fig 4.20 se muestra en la pantalla el mensaje enviado por el mecanismo de envío de mensajes basado en eventos, la vista presentada es de un dispositivo móvil con *Windows Mobile 6.0*.

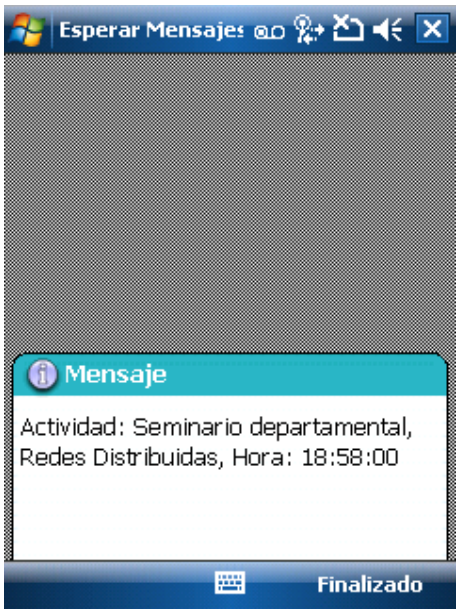

Figura 4.20: Visualización el mensaje en un *iPAQ* 610c

En la Fig 4.21 se puede observar el mensaje anterior desde un teléfono *Nokia* N95, con sistema operativo SymbianOS.

Figura 4.21: Visualización el mensaje en un *Nokia* N95

Capítulo 5

Conclusiones y trabajo futuro

Los avances tecnológicos cada día son más importantes, principalmente los relacionados con las redes de comunicaciones y los dispositivos móviles. Además el manejo de diferentes tipos de información, han motivado la aparición de aplicaciones más complejas y que requieren dar soporte a ciertas características, tales como la movilidad, disponibilidad de conexión, transferencia de información y seguridad. Estos aspectos no pueden ser tratados por una sola computadora, por esta razón, los sistemas distribuidos son arquitecturas computacionales que han sido útiles para dar soporte a las necesidades que demandan las aplicaciones modernas.

Los sistemas distribuidos se han convertido en un modelo necesario en el desarrollo de aplicaciones dedicadas y de propósito general, *e.g.*, sistemas distribuidos de actualización de datos, transferencia de datos en tiempo real, servicios de comunicación, entre otros. Estas aplicaciones forman parte de los diferentes sistemas de comunicación y su utilización es cada vez más común en ambientes distribuidos.

El paradigma de programación orientado a objetos es uno de los más utilizados como herramienta de desarrollo para crear aplicaciones en entornos distribuidos. *Java* es el lenguaje más representativo de este tipo de programación, aportando diferentes distribuciones y API's para crear aplicaciones con grandes capacidades y también a dispositivos con características limitadas de procesamiento.

Nuestro trabajo de tesis representa el desarrollo de una herramienta de comunicación para la ejecución en dispositivos móviles que cuenten con el soporte del lenguaje *Java Micro Edition*. La aplicación desarrollada se complementa con un servidor que se encarga de gestionar el envío de los mensajes que reciben los dispositivos móviles, el cual fue implementado usando el lenguaje *Java*. Además se implementó un servicio de registro para entornos Web, orientado a dispositivos móviles, donde se realiza el registro de usuarios y automáticamente se registran los dispositivos. Como apartado final del presente trabajo, en esta sección se describen los comentarios finales acerca de la tesis desarrollada, exponiéndose las conclusiones, contribuciones y el trabajo futuro.

5.1 Conclusiones

1. El resultado del trabajo de tesis comprende lo siguiente:

   i) Servicio de registro: obtuvimos un sitio Web orientado a dispositivos móviles, el cual permite registrar tanto a los usuarios como a los dispositivos móviles. Cabe señalar que el dispositivo es registrado de forma automática, *i.e.*, de forma transparente para el usuario. El sitio Web permite almacenar el perfil del usuario que se registra, una vez editado el perfil podemos identificar las preferencias que el usuario ha marcado, indicando así, el tipo de información que desea recibir en su dispositivo móvil

   ii) Servicio de mensajes: Desarrollamos una aplicación que se encarga de enviar mensajes a varios dispositivos conectados en una red inalámbrica local. La característica principal incorporada en la aplicación es la implementación de un filtro de mensajes, con la finalidad de que el mensaje solo sea enviado a los usuarios que así lo han marcado en su perfil, como una de sus preferencias

   iii) una arquitectura del servicio basada en clases de *Java*, que sirve como el modelo del diseño para la implementación de cada componente de las aplicaciones desarrolladas

   iv) un servicio de registro en un entornos Web, integrado por herramientas *Java* tales como:

   - *servlets*, son programas que atienden las solicitudes realizadas por un cliente a través de una página Web, teniendo un servidor o contenedor de *servlets* como el encargado de realizar el procesamiento de la solicitud.
   - *JSP*: acrónimo de *Java Server Pages* (páginas de servidor *Java*), es una tecnología orientada para crear páginas Web con soporte de programación en el lenguaje *Java*.

   v) un servicio de mensajes cortos, integrado por siguientes componentes:

   - *Java Swing*: herramienta para desarrollar interfaces gráficas de usario
   - *Sockets*: mecanismo que proporciona *Java* para la comunicación en red. Permite comunicar los dispositivos móviles con el servidor de mensajes.

   vi) una base de datos desarrollada en MySQL, donde se almacenan los datos generados por el servicio de registro, que posteriormente son consultados por el servicio de mensajes, con el objetivo de obtener información necesaria para llevar a cabo los procedimientos programados. Por ejemplo, el filtro de mensajes.

   Las herramientas descritas anteriormente se unen para conformar el servicio SEGEMENS, que proporciona un centro de registro en un entorno Web principalmente enfocado a dispositivos móviles, además proporciona mecanismos de comunicación en una red inalámbrica a través de un servicio de mensajes cortos de texto.

2. Los *servlets* de *Java* tuvieron un papel muy importante en el servicio de registros orientado a entornos Web, debido a su naturaleza, el dispositivo móvil no realizó ningún procesamiento de datos, todo fue responsabilidad del servidor donde están

contenidos los *sevlets*. En los dispositivos móviles solo se presentaban interfaces para ingresar datos de manera sencilla, los cuales se enviaban al servidor, también solo se visualizaban las respuestas enviadas por el servidor a través de páginas Web

3. El diseño de páginas Web con JSP, dió como resultado un sitio orientado principalmente a dispositivos móviles, gracias a la implementación de hojas de estilos. En éstas se definieron las dimensiones que tomarían las paginas Web para visualizarse en un dispositivo móvil, tomando como referencia un valor estándar para el ancho de la página y definiendo los valores de la altura de la misma. La visualización del sitio fue satisfactoria en la mayoría de los dispositivos que se usaron en las pruebas del servicio.

4. En el desarrollo de la solución, la programación multihilo fue pieza clave para resolver los siguientes problemas:

   a) nos permitió agregar el soporte para la concurrencia de la arquitectura cliente-servidor, con esto, el servidor es capaz de atender varias conexiones iniciadas por los clientes al mismo tiempo. De esta forma se procesa cada conexión de manera separada. Al tratar cada cliente por separado se asegura que si alguna conexión falla en algún momento, no afecte o interrumpa a las demás

   b) el tratamiento de cada hilo generado fue parte fundamental al momento de realizar el filtro de mensajes. Ya que cada hilo representaba a un cliente conectado, desde este punto se pudo interactuar con la base de datos para obtener las preferencias que el cliente especificó al momento de editar su perfil.

5. Se desarrolló una estrategia de solución basada la programación guiada por eventos. En estas arquitecturas, los procesos se comunican básicamente a través de la propagación de eventos, los que opcionalmente transportan datos. En el caso del servidor de mensajes, los eventos eran producidos por el administrador del servicio de mensajes y también se cuenta con un mecanismo de produccíon de eventos automáticos, los cuales se disparaban al ordenar el envío del mensaje o al producirse un cambio en la base de datos, específicamente en la tabla de *eventos*. La propagación de los mensajes generada por la ocurrencia de un evento, siguió la idea básica de publicación-suscripción, *i.e.*, se publica un evento al momento de enviar un mensaje que sólo aquellos clientes suscritos a tal mensaje lo recibirán.

6. La parte del servidor de mensajes fue realizada sobre una arquitectura cliente-servidor, con mecanismos de actualización de datos tipo *Push*, *i.e.*, las actualizaciones se propagan hacia los clientes sin que éstos las soliciten. Los protocolos basados en *Push* son eficientes en el sentido de que cada actualización insertada puede utilizarse por uno o más clientes, esta característica nos ayudó a definir la solución para el envío de los mensajes.

7. Las pruebas y los resultados experimentales nos indican que el funcionamiento, la usabilidad y la utilidad del servicio SEGEMENS son validadas de manera exitosa.

5.2 Contribuciones

1. De manera general, el servicio SEGEMENS proporciona:

   a) una aplicación orientado a objetos a través de clases, el cual define una estructura funcional sobre la cual una aplicación distribuida de comunicación basada en eventos puede ser organizada e implementada.

   b) el ambiente de ejecución es orientado a entornos Web, entornos móviles (dispositivos móviles) y fijos (PC's).

2. El diseño de las aplicaciones desarrolladas puede ser reutilizado en proyectos similares al nuestro. Puede tomarse como base para desarrollar una aplicación distribuida basada principalmente en eventos. Se puede extender y personalizar nuestro esquema de clases de acuerdo a sus requerimientos particulares y así, ahorrar tiempo y esfuerzo.

3. Al diseñar el servicio SEGEMENS se ha tomado en cuenta dejarlo preparado para poder integrarlo a todo un sistema de servicios. Hoy en día las arquitecturas orientadas a servicios son muy populares en los ambientes empresariales, el diseño de del servicio desarrollado en la presente tesis puede ser integrado en arquitecturas de éste tipo.

4. Al utilizar la tecnología *servlets* de *Java*, se cuenta con una alternativa más para el diseño de sitios Web móviles. Con los *servlets* se omite el procesamiento de lado del dispositivo móvil y se centraliza en el servidor. Además con el uso de JSP's se dispone de una herramienta flexible para el diseño de interfaces de usuario, las cuales se adaptan a diferentes tamaños de pantallas.

5. La base de datos cuenta con una estructura relacional aceptable, es posible reutilizar su diseño en otros casos particulares.

5.3 Trabajo futuro

Existen varias líneas de trabajos futuros para la presente tesis, a continuación se describen algunas de ellas:

1. El sitio Web orientado a dispositivos móviles está funcionando sobre el dominio de red local *labwless2* del departamento de computación, si se difundiera de manera pública, los usuario podrían cambiar sus perfiles desde cualquier lugar donde exista conexión a Internet. Se necesitaria dar salida a la red global al servidor donde se encuentra alojado el sitio Web.

2. El servicio de mensajes trabaja en la red inalámbrica local, para poder extenderse al exterior se podría proponer hacer uso de la red de telefonía celular y mandar los mensajes vía SMS, el inconveniente es el costo de cada mensaje enviado. Otra alternativa es usar otro tipo de protocolos, SIP es una opción que puede ser útil para extender el servicio de mensajes a la red global.

3. Además del servicio de mensajes, se pueden agregar otro tipo de servicios tales como audio, video, voz, imágenes, etc., con el objetivo de extender la gama de información transferible a través de SEGEMENS.

4. SEGEMENS es capaz de reconocer diferentes dispositivos móviles, pero la diferenciación es muy general, falta dar un tratamiento enfocando a las características y capacidades de cada dispositivo. Actualmente las características son tomadas de un repositorio existente, entonces se propone obtener las capacidades de manera directa en el dispositivo, a través de implementar llamadas a los sistemas operativos. En este punto se buscaría desarrollar una aplicación 100% heterogénea, *i.e.*, que funcione en distintos dispositivos con características diferentes.

5. La interfaz gráfica del servidor puede ser mejorada, tal vez cambiando la forma de navegación a través de las ventanas internas y que solo se maneje por medio de una ventana general. Mejorar la interfaz gráfica del dispositivo móvil, es otro punto importante como trabajo pendiente, si se desea integrar más servicios de comunicación se debe contar con una interfaz adecuada para la visualización de los datos recibidos.

6. El servicio SEGEMENS puede ser extendido y acoplado en otros casos de estudio o ambientes. Puede ser integrado en empresas que desean informar a sus clientes a cerca de sus promociones, ofertas o eventos importantes. También puede ser introducido como un medio más de divulgación de información en aeropuertos, hospitales u organizaciones administrativas.

7. Dado que los mensajes viajan a través de una red, se considera importante pensar en mecanismos de seguridad que protegan los datos. El servicio SEGEMENS no envía datos con información relevante, pero si se implementara en organizaciones donde el envío de mensajes debe ser integro, sería necesario aplicar algorítmos criptográficos para enviar los mensajes de forma cifrada.

Referencias

[1] Park,, Yongwan and Adachi,, Fumiyuki, *Enhanced Radio Access Technologies for Next Generation Mobile Communication*, 1-37, 2007, 1402055315,Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[2] Vilho Räisänen, *Service quality support-an overview*, Elsevier B.V., vol. 27, 2004.

[3] IEEE Computer Society, *IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture*, The Institute of Electrical and Electronics Engineers, Inc., 2002.

[4] Diederich Jörg, Wolf Lars and Zitterbart Martina, *A mobile differentiated services QoS model*, Elsevier, 2004.

[5] K.V. Prasad, *Principles of Digital Communication Systems and Computer Networks*, Ed. Charles River Media, INC, 2004.

[6] Dwenaël Le Bodic, *Mobile Messaging, Technologies and Services SMS, EMS and MMS*, John Wiley & Sons, 2005.

[7] Jorge, Escribano and Carlos, García and Celia, Seldas and José, Ignacio Moreno, *DiffServ como solución a la provisión de QoS en Internet*, Departamento de Ingeniería Telemática, Universidad Carlos III de Madrid, 2002.

[8] Andre Beller, Edgard Jamhour, Mauro Fonseca and Thiago Pereira, *DiffServ Management on Mobile IP Networks using COPS-PR*, Network Control and Engineering for QoS, Security, and Mobility, IV, ed. Gaiti, D., Boston: Springer, pp 187 - 198, 2007.

[9] Denney C. Justin and J.P. Race Nicholas, *The Impact of Wireless Device Access on Content Delivery Networks*, Interactive Multimedia on Next Generation Networks, vol. 2899/2003, pp 1 - 15, Springer Berlin/Heidelberg, 2004.

[10] Wi-Fi Aliance, *WiFi CERTIFIED for WMM-Support for Multimedia Applications with Quality of Service in Wi-Fi Networks*, White paper, september 2004.

[11] Subrahmanyam Allamaraju, Andrew Longshaw, Daniel O'Connor, Gordon Van Huizen, Jason Diamond, John Griffin, Mac Holden, Marcus Daley, Mark Wilcox and Richard Browett, *Professional Java Server Programming J2EE Edition*, Wrox Press, 2000.

[12] Hall Marty and Brown Larry, *Core Servlets and JavaServer Pages*, Prentice Hall and Sun Microsystems Press, Second Edition.

[13] García Javier, Rodríguez José Ignacio and Imaz Aitor, *Aprenda Servlets de Java*, Campus Tecnológico de la Universidad Navarra, White Paper.

[14] Xiaolu Zuo, *Communication Networks: States of Arts*, Springer Berlin/Heidelberg, 2004.

[15] Weiser, Mark, *The computer for the 21st century*, SIGMOBILE Mob. Comput. Commun. Rev., vol. 3, num. 3, issn 1559-1662, pp 3-11, ACM, New York, NY, USA, 1999.

[16] Frank H. P. Fitzer and Frank Reichert, *Python for Symbian Phones*, Springer Netherlands, 2007.

[17] Scheible Jürgen and Tuulos Ville, *Mobile Python, Rapid prototyping of applications on the mobile platform*, John Wiley & sons, Ltd, 2007.

[18] Joachim Rossberg and Rickard Redler, *Pro Scalable .NET 2.0 Application Designs*, Apress, 2006.

[19] Rittwik Jana, Trevor Jim, Matti Hiltunen, Sam John, Huale Huang, Yih Farn Chen, Serban Jora, Radhakrishnan Muthumanickam and Bin Wei, *iMobile EE-An Enterprice Mobile Service Plataform*, Springer, 2006.

[20] Pyssysalo Tino, Bergström Terje, Bocklage Jürgen, Defée Pawel, Granholm Patrik, Huttunen Juuso, Kärkkäinen Ville, Kilponen Matti, Kinnunen Timo, Tomi Koskinen, Matela Mika, Otsala Ilkka, Partanen Antilli, Puronen Timo, Seppälä Jere, Sivola Juha, Tanable Saiki, Tarhonen Jukka, Teräsvirta Tommi, Turunen Tuukka and Välimäki Tommi, *Programming for the Series 60 Plataform and Symbian OS*, John Wiley & Sons, 2003.

[21] Coulouris George, Dollimore Jean and Kindberg Tim, *Sistemas Distribuidos: Conceptos y Diseño*, 3a ed. Madrid: Pearson Educación S.A., 2001.

[22] wimax, Zhang Yan and Chen Hsiao-Hwa, *Mobile WiMAX*, Auerbach Publuications., 2007.

[23] Tanenbaum Andrew and Van Steen Maarten, *Sistemas Distribuidos, Principios y Paradigmas*, Pearson Educación, 2008.

[24] Tse David and Viswanath Pramod, *Fundamentals of Wireless Communication*, Cambridge University Press, 2005.

[25] Attiya Hagit and Welch Jennifer, *Distributed Computing: Fundamentals, Simulatios and Advanced Topics*, 2a. ed, Jhon Wiley & Sons, Inc New Jersey, 2004.

[26] Verissimo Paulo and Rodrigues Luís, *Distributed Systems for System Architects*, Kluwer Academic Publishers, 2001.

[27] Puder Arno, Römer Kay and Pilhofer Frank, *Distributed System Architecture: A Middleware Approach*, Elsevier, 2006.

[28] Choi Kyu Jong, Kim Joon Han, Suk Jin Beom and Ji Yonggu, *Web-based System Development for Usability Evaluation of Ubiquitous Computing Device*, Springer-Verlag Berlin Heidelberg, 2009.

[29] Eduardo Antonio Davalos Camarena, *Servicio de Gestión de Sesiones de Usuarios en Ambientes Móviles y Heteregéneos*, Centro de Investigación y Estudios Avanzados del I.P.N., México, 2009.

[30] Rodrigo Vega García, *Sistema de comunicación con niveles de servicio basado en SIP para ambientes heterogéneos*, Centro de Investigación y Estudios Avanzados del I.P.N., México, 2008.

[31] Christian Ivan Mejia Escobar, *Herramienta autoconfigurable para el desarrollo de aplicaciones distribuidas de tiempo real flexibles y dinámicas*, Centro de Investigación y Estudios Avanzados del I.P.N., México, 2007.

[32] Sing Li and Jonathan Knudsen, *Beginning J2ME: From Novice to Professional*, Editor Apress, 2005.

[33] Sun, *Java 2 Plataform, Micro Edition*, Sun microsystems, White Paper, 2002.

[34] Sun, *CDC: Java Plataform Technology for Connected Devices*, Java Plataform, Micro Edition, Sun microsystems, White Paper, 2005

[35] Agustín Froufe Quintas and Patricia Jorge CÃ¡rdenas, *J2ME. Java 2 Micro Edition. Manual de usuario y tutorial*, Alfaomega, Grupo Editor Ra-Ma, 2004.

[36] afaria, *Afaria Device Management and Security From a Single Console*, Sybase iAnywhere, White Paper.

[37] Greiner Cristina, *Programación Guiada por Eventos, Programación Secuencial, Interactiva y Orientada a Eventos*, Facultad de Ciencias Exactas y Naturales y Agrimensura, Departamento de Informatica, Programación IV.

[38] Evjen Bill, Lhotka Rockford, Hollis Billy, Sheldon Bill, Sharkey Kent, McCarthy Tim and Ramachandran Rama,*Professional VB 2005*, Wiley Publishing, Inc., 2006.

[39] Basterretche Juan, *Dispositivos Móviles*, Universidad Nacional del Nordeste, Facultad de Ciencias Exactas, Naturales y Agrimensura.

[40] Microsoft, *Architectural Overview of Windows Mobile Infrastructure Components*, Microsoft Corporation, White Paper.

[41] Yang Baijian, Zheng Pei and Ni M. Lionel, *Professional Microsoft Smartphone Programming*, Wiley Publishing, Inc, 2007.

[42] Dourish Paul and Bellotti Victoria, *Awareness and Coordination in Share Workspaces*, ACM, Cambridge, 1992.

[43] Zhou Shumin, Zhong Guoyun and Zhang Tiantai, *The short message management system based on J2ME*, IFIP International Federation for Information Processing, Vol. 258, pp. 27-34, Springer, Boston, 2008.

[44] Kim Jae-Young, Lee Ju-Yong, Park Kyung-Joon, Kim Jun-Hyung, Chang Hong-Sung, Lim Geunhwi, Chang Yong, Kim Han-Seok, *WiBro (Wireless Broadband): An 802.16d/e Simulation Model*, Telecomunications Systems Division, Samsung Electronics Suwon Korea, 2006.

[45] Kurzyniec Dawid, Sunderam Vaidy and Slawinska, *REVENTS: Facilitating Event-Driven Distributed HPC Applications*, Springer-Verlag Berlin Heidelberg, pp. 291-301, 2008.

[46] Wong Clinton, *HTTP Pocket Reference*, O'Reilly & Associates, 2000.

[47] Eckstein Robert, Loy Marc, Wood Dave, Elliott James and Cole Brian, *Java Swing*, O'Reilly & Associates, Inc., 2003.

[48] Silberschatz Abraham, F. Korth Henry and SudarShan S., *Fundamentos de Bases de Datos*, Mc Graw Hill, Cuarta edición, 2002.

[49] Mehta Nirav, *Mobile Web Development*, Packt Publishing, 2008.

[50] Tzovaras Dimitrios, *Multimodal User Interfaces*, Springer-Verlag Berlin Heidelberg, 2008.

[51] O'Hara Bob and Petrick Al, *IEEE 802.11 Handbook: A Designer's Companion*, Standards Information Network IEEE Press, 2005

[52] Bertrand Meyer, *Touch of Class*, Springer Berlin HeidelBerg, p.p. 663-698, 2009.

Páginas Web consultadas

[53] Sun, *Connected Limited Device Configuration (CLDC); JSR 30, JSR 139 Overview [en línea]*, Sun microsystems, [Citado: 21-septiembre-2009], http://java.sun.com/products/cldc/overview.html

[54] Sun, *Mobile Information Device Profile (MIDP); JSR 37, JSR 118 Overview [en línea]*, Sun microsystems, [Citado: 25-septiembre-2009], http://java.sun.com/products/midp/overview.html

[55] Juan Manuel Fernández, *Tipos de dispositivos móviles [en línea]*, [Citado: 26-agosto-2009], http://leo.ugr.es/J2ME/INTRO/intro_4.htm

[56] Universidad Nacional Experimental de los Llanos Occidentales Ezequiel Zamora, *Servicio Multimedia [en linea]*, [Citada: 16-noviembre-2009], http://www.edudigital.unellez.edu.ve/portal/index.php?option=com_content&task=view&id=40&Itemid=64

[57] imprescindible.es, *Sistemas operativos móviles actuales [en linea]*, [Citado: 17-septiembre-2009], http://www.imprescindible.es/sistemas-operativos-moviles-actuales

[58] msdn, *Visual Studio [en linea]*, [Citado: 24-septiembre-2009], http://msdn.microsoft.com/es-es/library/52f3sw5c.aspx

[59] TeliaSonera, *TeliaSonera first in the world with 4G services [en linea]*, [Citado: 20-diciembre-2009], http://www.teliasonera.com/press/pressreleases/item.page?prs.itemId=463244

[60] Newstream/Arraycomm, *Martin Cooper-History of Cell Phone [en linea]*, [Citado: 23-septiembre-2009], http://inventors.about.com/cs/inventorsalphabet/a/martin_cooper.htm

[61] Sistemas Heterogéneos, *Instituto Tecnológico del Istmo [en linea]*, [Citado: 5-diciembre-2009], http://www.itistmo.edu.mx/Pag%20Informatica/APUNTES_archivos/page0003.htm

[62] msdn Architecture Center, *Service-Oriented Architecture: Considerations for Agile Systems [en linea]*, [Citado: 18-septiembre-2009], http://msdn.microsoft.com/en-us/library/aa480028.aspx

[63] Sun Microsistems, *Hierarchy for Package javax.servlet [en linea]*, [Citado: 12-agosto-2009], http://java.sun.com/j2ee/tutorial/api/javax/servlet/package-tree.html.

[64] Leandro Navarro Moldes and Joan Manuel Marqués, *Arquitectura de Sistemas Distribuidos [en línea]*, [Consulta: 26-agosto-2009], http://www.luisbernal.es/descargas/index.php?act=download&id=58

[65] Apple Inc, *iPhone OS Technology Overview [en línea]*, 2009, [Citado: 10-diciembre-2009], http://developer.apple.com/iphone/library/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Introduction/Introduction.html

[66] Pursnani Vandana, *Introducción a la programación de Java Servlets [en línea]*, [Citado: 28-septiembre-2009], http://www.acm.org/crossroads/espanol/xrds8-2/servletsProgramming.html

[67] IDC, *IDC Analyzed the Future*, http://www.idc.com/spain/