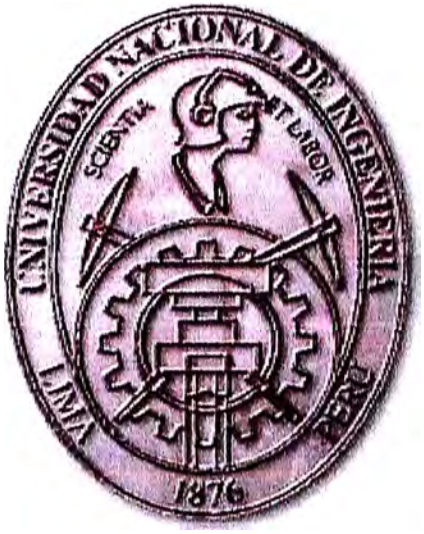
UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA



APLICACIONES MÓVILES CON PROTOCOLO WAP

INFORME DE SUFICIENCIA

PARA OPTAR EL TÍTULO PROFESIONAL DE:

INGENIERO ELECTRÓNICO

PRESENTADO POR:

ROGER REYNAN GUTIERREZ FLORES

PROMOCIÓN
2004 - II

LIMA – PERÚ
2009

APLICACIONES MÓVILES CON PROTOCOLO WAP

DEDICATORIA

Dedico este proyecto a mis Padres por estar ahí cuando más los necesité; en especial a mi madre por su ayuda y constante cooperación para ampliar mis conocimientos y estar más cerca de mis metas profesionales. Esto fue posible primero que nadie con la ayuda de Dios, gracias por otorgarme la sabiduría y la salud para lograrlo. Gracias a los intercambios y exposiciones de ideas con mis compañeros y amigos de estudios durante el curso de actualización.

SUMARIO

En este trabajo, se presenta el desarrollo y la arquitectura empleada en el proceso de implementación de la tecnología móvil.
Además del Framework .NET como una alternativa para desarrollar programas reutilizables y extensibles que permitan sacar provecho de las ventajas que ofrecen los dispositivos móviles actuales. Como otra alternativa de desarrollo se presenta a Java.

Adicionalmente se presenta un resumen de UML, el cual ofrece a los arquitectos de sistemas una notación estándar para modelar sistemas. Ahora, igual que los arquitectos leen los planos, un modelador de objetos puede coger cualquier diseño y entender qué se está capturando. UML también es bueno para la comunidad del modelado - en vez de gastar tiempo acordando cómo expresar la información que se captura, los modeladores pueden resolver el problema real, lo cual es diseñar el sistema.

INDICE

INTRODUCCIÓN

El desarrollo de software ha extendido sus alcances, así, las aplicaciones tecnológicas y los medios para usarlas han evolucionado de tal forma que su uso ya no esta limitado a una computadora personal (PC), si no que se ha extendido al uso de dispositivos móviles (PDA, teléfonos celulares, Smartphone2 y TabletPC3) para lograr un mayor alcance de la aplicación y obtener los beneficios que el cómputo móvil ofrece.

Este modelo basado en el uso de dispositivos móviles se ha desarrollado desde hace varios años, dando como resultado varios proyectos de investigación y algunos productos comerciales.

La tecnología móvil ha evolucionado en los últimos años, lo que ha llevado a un crecimiento en el mercado de dispositivos móviles personales a un costo menor del que tenían hace algún tiempo, permitiendo que llegue a más personas en donde los usos que les dan van desde el recreativo, medio de comunicación o cómputo empresarial. Por otro lado, los actuales dispositivos cuentan con capacidades ricas en recursos multimedia como audio, video, fotografías, conectividad y comunicación, lo que ha elevado las capacidades de cómputo y flexibilidad de los dispositivos. Así, de manera paralela el desarrollo de aplicaciones de software para este tipo de dispositivos también ha crecido. En ese sentido, existen en el mercado aplicaciones móviles empresariales como clientes de correo electrónico, bases de datos reducidas con contenido de un tópico en particular como recetas de cocina, horarios de trenes, mapas de ciudades, entre otras; aplicaciones de entretenimiento como juegos; reproductores de música o video y clientes de televisión portátil; además existen aplicaciones de uso general como agendas, calculadoras o planificadores de tareas.

CAPITULO I
LA EVOLUCIÓN DE LAS APLICACIONES MÓVILES

Los dispositivos móviles están avanzando a una velocidad sorprendente. Pero ¿es esta rápida evolución del otrora humilde teléfono celular una cuestión de dispositivos que enloquecieron o una progresión racional? La respuesta es probablemente una combinación de ambos. Pero el hecho es que los dispositivos que ahora están en etapa de diseño llevarán a los servicios móviles del mañana a lugares que sólo podemos soñar hoy.

Todos hemos visto la visión del futuro móvil de James Bond. Dispositivos móviles discretos que registran y transmiten sonidos e imágenes instantáneamente, controlan un amplio espectro de dispositivos desde autos hasta refrigeradoras de champagne, y pueden ser descartados y reemplazados sin dudarlo. Adivinen que: 007 ya no es algo especial – y en un futuro predecible, el tipo de magia que Q le proporciona a Bond costará sólo unos pocos dólares en una tienda de la esquina.

Esto sucederá debido a que la evolución de los dispositivos manuales está ingresando a una nueva fase, a medida que los teléfonos celulares se transforman en dispositivos multimedia de alta capacidad que procesan interfaces intuitivas de usuario. Queda por verse si estas capacidades adicionales de los dispositivos garantizarán altos ingresos o una mejor experiencia para el cliente. Pero al menos podemos estar seguros que estos pequeños adminículos continuarán sorprendiéndonos, tanto por la reducción de su tamaño como por su funcionalidad exponencialmente mayor.

En los próximos cuatro años, la agrupación de nueva tecnología en dispositivos manuales será cada vez mayor, convirtiendo a los teléfonos celulares tradicionales en dispositivos audiovisuales de avanzada, procesados en redes rápidas, inalámbricas y de alta capacidad. Gartner predice que el precio minorista de dispositivos de low-end se reducirá a la mitad en 2008, y que para el 2009, existirá un porcentaje de mercado casi igual en las ventas de estos dispositivos entre los modelos de tercera generación (3G) y los de 2.5G. Todos los teléfonos 3G tendrán capacidades de doble modalidad (2.5G y 3G).

El software móvil avanzará igualmente rápido. En el 2005 veremos una aceleración mayor en la adopción de aplicaciones Java, especialmente en el campo de rápido crecimiento

de los juegos. Otras plataformas tales como BREW (Binary Runtime Environment for Wireless) open source de Qualcomm en Estados Unidos ya construyeron una sólida posición. BREW, por ejemplo, es usada por Verizon y algunos participantes regionales de Code-Division Multiple Access (CDMA) en EE.UU. y es ofrecida comercialmente en más de 100 dispositivos por fabricantes líderes y Original Device Manufacturers (ODMs).

La demanda de los clientes también impulsará el diseño de hardware. Los usuarios quieren dispositivos bonitos que soporten e-mails y aplicaciones más complejas de información personal (Personal Information Management – PIM), web-browsing y juegos. La industria responderá creando dispositivos funcionales pero modernos, con pantallas más grandes, permitiendo un rápido y continuo cambio entre aplicaciones. Los dispositivos también incorporarán sensores que dispararán aplicaciones industriales y personales, desde las más obligatorias hasta las críticas para la vida. Algunos actuarán como monitores del cuidado cardíaco, otras mejorarán y expandirán la experiencia de los juegos, siendo sensibles al movimiento, y otras medirán la temperatura ambiente y responderán conforme a ello.

1.1 ¿Qué modelo logrará funcionar?

Este panorama móvil emergente abrirá una amplia gama de posibilidades comerciales a las empresas en el espacio móvil y más allá de éste. Del lado del usuario, todas las industrias, desde la de servicios financieros, distribución y ventas minoristas hasta la de salud, seguridad, incluso los departamentos de policía que ya están utilizado la transmisión de fotos de delitos por telefonía celular) necesitan adaptarse a estos dispositivos, y ver cómo potenciarlos en su propio contexto único.

El primer paso para lograrlo es modelar cómo las nuevas capacidades de video, ubicación y demás impactarán la forma de vida de sus clientes y de sus propios empleados, y cómo esto abrirá nuevas formas de hacerlo.

Del lado de la industria, los fabricantes de transmisores y dispositivos deben colaborar mutuamente para permitir que los terceros incorporen las aplicaciones adecuadas rápida y fácilmente a la plataforma subyacente. Es crucial que la industria asegure que toda el área no se convierta en una pesadilla para la atención al cliente (CRM), con frustraciones de ambos lados y gente tan confundida por la tecnología que sea incapaz de recordar cómo realizar una llamada telefónica básica. A medida que se migran las nuevas capacidades, surgen nuevas áreas de oportunidad:

1.1.1 El video está a punto de despegar

Las aplicaciones handset incluyen ahora videoconferencias, video clips y difusión de TV digital. El uso de estas tecnologías despegará una vez que se superen los desafíos técnicos

– el tamaño de la pantalla, la claridad, la lenta inicialización de llamadas y la corta vida útil de las baterías. Tecnologías tales como Organic LED mejorarán las pantallas drásticamente, proporcionarán una mayor velocidad de visualización, mayor brillo y nitidez con una menor fuente de alimentación. En el sector de los medios, la calidad de la experiencia visual en dispositivos móviles será clave.

1.1.2 Una mejor imagen

A fines del 2004, las cámaras-teléfonos de 2 mega pixels declararon la guerra a las cámaras digitales. Esta tendencia continuará con el surgimiento de aplicaciones móviles para cámaras, tales como lectores de código de barras, permitiendo a los clientes compartir detalles personales en las tarjetas e-business, comprar entradas, comparar precios y solicitar ofertas. También habrá webcams móviles y aplicaciones de video vigilancia.

1.1.3 De cara a la música

ITunes de Apple marcó una tendencia con su modelo de comercialización y entrega de música digital. Los handsets equipados con reproductores de MP3, radios AF/FM y grabadores de voz son cada vez más comunes, y los servicios musicales online permiten a los usuarios escuchar canciones en sus teléfonos digitales. Nokia está avanzando mucho más allá con sus capacidades de "Radio Visual", permitiendo a las estaciones de radio agregar una dimensión interactiva a las transmisiones.

1.1.4 Juegos

Grandes desafíos y grandes recompensas – se estima que los juegos electrónicos en dispositivos móviles valdrán US$ 2.800 millones a nivel mundial para el 2009. Este mercado plantea importantes desafíos técnicos y de contenido, y la compleja cadena de valor de crear, empaquetar y vender juegos móviles requiere nuevos roles para cada participante. Sin embargo, para el 2009, se espera que 114 millones de personas en todo el mundo jueguen juegos online, comparado con los 50 millones actuales.

1.1.5 Los sistemas de pago están de moda

Los medios de pago móviles finalmente están ganando terreno entre los consumidores, empresas minoristas, bancos, proveedores y redes de pago. Para el 2009, se espera que las transacciones móviles en EE.UU., Europa Occidental y Japón valgan $7.000 millones, $15.000 millones y $12.000 millones respectivamente. El desafío más grande será actualizar los sistemas de POS en miles de bocas de expendio para manejar los medios de pago móviles. Pero el premio es enorme: de acuerdo con Gartner, el 15% de los encuestados en Europa ya están interesados en pagar productos o servicios mediante su teléfono celular.

1.1.6 Apertura del mercado industrial

Los dispositivos móviles componen la red física de las nuevas aplicaciones industriales utilizando telemetría de sensores, GPS y tecnologías de Identificación de Radio Frecuencia (RFID). Las aplicaciones incluirán video vigilancia, lectura de medidores, monitoreo ambiental, transporte, gestión de inventario y stock, y muchos más. El mensaje general es claro: las continuadas tendencias en el diseño y capacidades de los dispositivos handsets no ocurren en forma aislada. Por el contrario, el rápido avance de los dispositivos móviles produce implicancias y oportunidades importantes para cada segmento de la industria móvil. Y los operadores estarán a la vanguardia de este cambio, a medida que buscan obtener una porción más grande de las billeteras de los clientes, agrupando y proporcionando las aplicaciones y los servicios móviles que éstos quieran.

CAPITULO II
WIRELESS ACCESS PROTOCOL (WAP)

2.1 Entendiendo la Tecnología Móvil

Debido al crecimiento de la industria inalámbrica, Ericsson, Nokia, Motorola y Phone.com (ahora Openwave) fundaron el WAP Forum en 1997 para unificar las tecnologías de acceso móvil. El WAP Forum, que ahora asciende a más de 400 miembros, ha contribuido a la adopción de WAP y WML (Wireless Markup Language), como estándares de fabricación. Usted encontrará WAP y WML en casi todos los teléfonos móviles digitales en la actualidad.

WAP es un conjunto de especificaciones (incluyendo WML) que se basa en las variaciones de los modernos estándares abiertos Web. WML, por ejemplo, se basa en XML. Las especificaciones Wireless fueron necesarias porque los protocolos Web actuales (TCP e IP), no están bien adaptados a dispositivos móviles.

WAP es el resultado del interés compartido por los líderes de la industria por crear un estándar abierto que permita ofrecer aplicaciones móviles avanzadas y acceso a los contenidos de Internet a los usuarios de teléfonos móviles. Pero ¿Por qué este nuevo protocolo? El entorno móvil es muy diferente al tradicional de las tecnologías de la Información (IT). Así, las especificaciones WAP se basan tanto en los estándares de Internet como en los nuevos protocolos basados en Internet, optimizados específicamente para el entorno móvil. WAP tiene en cuenta también el factor limitativo de la red móvil -necesidad de comprimir los datos, tiempo de espera y limitado ancho de banda - y las limitaciones en los terminales: CPUs menos potentes, menor capacidad de memoria, autonomía limitada, pequeños pantallas y diferentes dispositivos de entrada. Unas limitaciones que no impedirán que WAP sea el próximo líder en el próximo Boom de Internet desde el Worid Wide Web. Con WAP, los usuarios accederán a Internet y otros servicios móviles mientras estén en ruta, independientemente de los fabricantes y operadores, gracias a la compatibilidad de los productos y soluciones, al tratarse de una plataforma común y abierta. Al disponer de un modelo común de programación y un mismo lenguaje para el desarrollo de aplicaciones, se reduce el riesgo de la fragmentación del mercado, a la vez que se le moviliza para la rápida adopción de un estándar consistente, lo que beneficia a todos: usuarios finales, operadores y la

industria de las telecomunicaciones. Los usuarios de teléfonos móviles se acostumbrarán rápidamente a los servicios WAP, ya que no será necesario aprender un nuevo y complejo interface en los aparatos móviles. Además, con el uso de la tecnología estándar de Internet, será posible optimizar los contenidos a las características de las redes actuales y futuras de telefonía móvil. Como un estándar abierto que es, WAP proporcionará la misma tecnología a todos los vendedores independientemente de los sistemas de redes. Así, habrá terminales y soluciones compatibles con WAP de múltiples fabricantes. De hecho, ha sido adoptado ya por fabricantes que representan un 75% del mercado mundial y por operadores que actualmente cuenta con más de 100 millones de abonados en todo el mundo. Al ofrecer una vía tecnológica también abierta, los operadores pueden seleccionar la mejor alternativa entre una amplia gama de productos. Asimismo, WAP ofrece potenciales economías de escala, al animar a los fabricantes de móviles y otros dispositivos a invertir en el desarrollo de nuevos productos compatibles. Infraestructura de Redes WAP WAP soporta Servicios como Mensajes Cortos (SMS), Circuit Switched Data, Unstructures Supplementary Service Data (USSD) así como el inminente General Packet Radio Service (GPRS). Por otra parte, la compañía tiene soluciones de red completas para la creación e implementación de GPRS Data Service, que incluye en su núcleo tanto el protocolo de Internet (IP) como infraestructuras de redes de radio en este sistema, que ofrece la mejor conectividad para las aplicaciones WAP.

2.2 Historia de WAP

WAP es un estándar global y no está controlado por una sola empresa. Ericsson, Nokia, Motorola y Unwired Planet fundaron el Foro WAP en el verano de 1997 con el propósito inicial de una industria esencial de las especificaciones para el desarrollo de aplicaciones sobre redes de comunicaciones inalámbricas.

En un primer momento Phone.com, Ericsson, Nokia y muchos otros comenzaron a elaborar las normas por separado el uno del otro, pero tan pronto entendieron que tendría más sentido centrarse en torno a la mejora de un estándar común, todos ellos quisieron establecer un esquema común de Internet para las transferencias a teléfonos móviles, sin tener que modificar las páginas de Internet para mostrar el particular, diferente en cada teléfono móvil u organizador personal. Con WAP Forum fueron capaces de encontrar ese protocolo.

2.3 ¿Qué es WAP?

El WAP (Wireless Application Protocol) es una especificación abierta, global que permite a los usuarios móviles con dispositivos inalámbricos acceder e interactuar fácilmente a la información y servicios de inmediato.

El WAP es una especificación para un conjunto de protocolos de comunicación para estandarizar la forma en que los dispositivos inalámbricos, como teléfonos celulares y radio transmisores, puedan ser utilizados para el acceso a Internet, incluyendo el correo electrónico, World Wide Web, y grupos de noticias.

WAP o lo que es lo mismo Protocolo para aplicaciones sin hilos. WAP también define un entorno de aplicaciones. WAP es escalable, permitiendo así Todo lo que se refiere a teléfonos móviles, Palmtops (Ordenadores de mano), portátiles, y cualquier otro acceso a redes con dispositivos sin conexión física. WAP es una solución unificada para los servicios de valor añadido existentes y futuros. El protocolo incluye especificaciones para las capas de la torre OSI de sesión y de transporte, así como funcionalidades de a las aplicaciones disponer de las capacidades de pantalla y recursos de red según su necesidad y en un gran número de tipos de terminales. Los servicios podrán ser aplicables a pantallas de una sola línea o a terminales mucho más complejos. Como cualquier estándar, las ventajas son múltiples a la hora de desarrollar aplicaciones, fabricar terminales o estructurar la red.

2.4 ¿Cómo Trabaja WAP?

En el escenario típico web, un usuario tipea una URL en el navegador. El navegador envía una solicitud HTTP al servidor Web, el cual interpreta la petición y determina que recursos recuperar o ejecutar.

Si la URL especifica un archivo, entonces el servidor envía de vuelta. Si la URL especifica una página ASP, su código se ejecuta antes de devolver los resultados al cliente. En ambos casos, el servidor Web escribe una cabecera HTTP para el archivo de salida o ASP y lo envía de vuelta al navegador.

El navegador interpreta la respuesta y muestra el contenido al usuario. La Fig. 2.1 compara el típico servidor / navegador web con un proceso de aplicación Web móvil escenario.

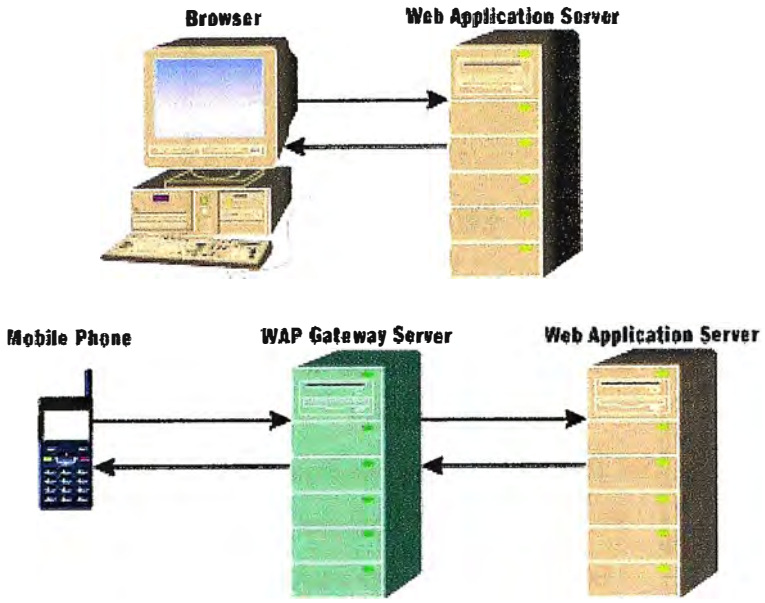


Fig. 2.1 Escritorio vs. Móvil

2.5 Arquitectura WAP

Para acceder a un sitio Web desde un teléfono móvil, tenemos que escribir la URL en el navegador. Usando WTLS (Wireless Transport Layer Security), el navegador del teléfono móvil genera una URL encriptada y la envía sobre WSP (Wireless Session Protocol) a un servidor gateway WAP. WSP es una variante de HTTP que transfiere información en formato binario, en lugar de texto basado en el formato.

El servidor gateway WAP interpreta la petición, lo traduce en una petición HTTP convencional, y lo envía al servidor Web.

Tras recibir la solicitud, el servidor Web lo interpreta y determina que recursos recuperar o ejecutar. Si la URL indica un archivo, el servidor envía el archivo al cliente. Si la URL indica una página ASP, entonces el servidor Web ejecuta el código ASP, antes de enviar los resultados al servidor gateway WAP. En este método el contenido devuelto debe estar en la forma de un documento WML.

El servidor gateway quita las cabeceras innecesarias, traduce el documento WML a binario, y envía la respuesta al navegador del teléfono móvil. El navegador interpreta el WML y lo muestra al usuario. Esta es la arquitectura detrás de WAP.

2.6 Configuración de WAP

Aquí tienes los datos necesarios para configurar una conexión WAP en tu teléfono.

Movistar	Airtel	Amena
Página de inicio:.....	Página de inicio:.....	Página de inicio:.....
Tipo de conexión: Continua	Tipo de conexión: Continua	Tipo de conexión: Continua
Portador: Datos	Portador: Datos	Portador: Datos
Número de acceso: 556	Número de acceso: 607 100 300	Número de acceso: +34 656 200111
Dirección IP: 194.224.26.30	Dirección IP: 212.73.32.10	Dirección IP: 10.132.61.10
Autentificación: Seguro	Autentificación: Normal	Autentificación: Normal
Tipo de Llamada de datos: RDSI	Tipo de Llamada de datos: RDSI	Tipo de Llamada de datos: RDSI
Velocidad de llamada: 9.600	Velocidad de llamada: 9.600	Velocidad de llamada: 9.600
Nombre de usuario: WAP	Nombre de usuario: wap	Nombre de usuario: CLIENTE
Contraseña: WAP	Contraseña: wap125	Contraseña: AMENA

TABLA Nº 2.1 Datos para conexión WAP

2.7 ¿Qué es WML y WML Script?

WML (Wireless Markup Language) es el lenguaje en el que se definen las páginas. Se recomienda tener conocimientos de HTML, ya que WML es muy parecido a este último.

WMLS (Wireless Markup Language Script) es el lenguaje en el que se definen los scripts. Tiene el formato típico de un script. Se recomienda tener nociones de JavaScript, ya que es muy parecido.

Como ejemplo creamos un archivo de nombre wap.wml e introducimos el siguiente código wml:

```
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
<card id="t1" title="TITULO">
<p align="center"><b>Hola amigos de mygnet</b></p>
```

```
<p align="center"><small>http://www.mygnet.com</small></p>
<p align="center">Ejemplo 1</p>
</card>
</wml>
```

TABLA Nº 2.2 Código fuente ejemplo

Y lo ejecutamos en el emulador de WAP, el cual tendrá la siguiente salida:

Fig. 2.2 Resultado del código en un simulador

Fig. 2.3 Resultado del código en un celular

2.7.1 Un poco sobre el lenguaje WML

a) Baraja: Es el equivalente a una página Web en Internet. Generalmente esta baraja es de un tamaño pequeño, cercano al Kbyte, debido a las restricciones que impone la comunicación inalámbrica.

b) Carta: Una carta contiene información de formatos, contenidos visibles en la pantalla. Un conjunto de cartas forma la baraja.

Cada carta de una baraja debe contener uno o más elementos.

c) Elementos: El código de una aplicación programada en wml está formado por una serie de elementos. Los elementos de bloques <elemento>...</elemento> y los que no tienen contenido como
 que es un salto de línea.

Vamos analizar un bloque de código wml:

```
<?xml verson="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml ">
<wml>
<card>
```

```
<p align="center" mode="wrap">
 Telefonía Móvil
</p>
</card>
</wml>
```

TABLA Nº 2.3 Bloque de código WML

1. Este bloque del programa debe incluirse siempre:

```
<?xml verson="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml ">
```

TABLA Nº 2.4 Cabecera de código WML

Aquí se define la versión como la definición XML del lenguaje a utilizar.

2. Y el código que esta entre<wml>…</wml> es a lo que se le llama la baraja, la cual se subdivide en cartas <card>…</card>, dentro de las cartas pueden contener elementos como párrafos … los cuales pueden definirse ciertas propiedades.

Como center = Parrafo centrado

Modo wrap = Modo de presentación envuelto, el cual garantiza que el texto de ser muy largo se verá en la línea siguiente.

Por lo pronto ya tenemos algunas nociones muy básicas del lenguaje wml.

2.8 ¿Qué puedo hacer con WAP?

Empresas: Envío de faxes y email al servidor de la intranet para una difusión de los mismos a menor coste o multienvio de email a los móviles de la compañía. Recordatorio, agenda, trabajo en grupo, programación de viajes. Acceso a las bases de datos de la compañía (Stocks, clientes, proveedores, catálogos)

Personales: Las últimas noticias de actualidad al alcance de tu mano (política, sociedad, economía, cultura, etc.). Todas las noticias y novedades que se producen diariamente en el mundo del deporte. Una agenda deportiva para saber en todo momento cuándo se produce un determinado partido y estar permanentemente informado de fechas y horas. Programación diaria de todas las cadenas de TV con posibilidad de búsqueda por tipo de programas, por fraja horaria, etc. Cartelera de todos los cines de las capitales de provincia españolas, actualizado semanalmente, indicando salas, películas y horarios. Resultados y

premios de todos los sorteos. Toda la información metereológica, con predicciones, estado actual del tiempo, imágenes vía satélite, previsiones para viajes de 24 y 48 horas, etc. Anuncios clasificados para comprar, vender, buscar y cambiar todo lo que se te ocurra.

CAPITULO III
PROTOCOLO DE APLICACIONES INALAMBRICAS WAP

3.1 ¿Qué es el protocolo de Aplicaciones inalámbricas?

El Protocolo de Aplicaciones Inalámbricas surge como la combinación de dos tecnologías de amplio crecimiento y difusión durante los últimos años: Las Comunicaciones Inalámbricas e Internet. Mas allá de la posibilidad de acceder a los servicios de información contenidos en Internet, el protocolo pretende proveer de servicios avanzados adicionales como, por ejemplo, el desvío de llamadas inteligente, en el cual se proporcione una interfaz al usuario en el cual se le pregunte la acción que desea realizar: aceptar la llamada, desviarla a otra persona, desviarla a un buzón vocal, etc. Para ello, se parte de una arquitectura basada en la arquitectura definida para el World Wide Web (WWW), pero adaptada a los nuevos requisitos del sistema. En la Figura se muestra el esquema de la arquitectura WAP.

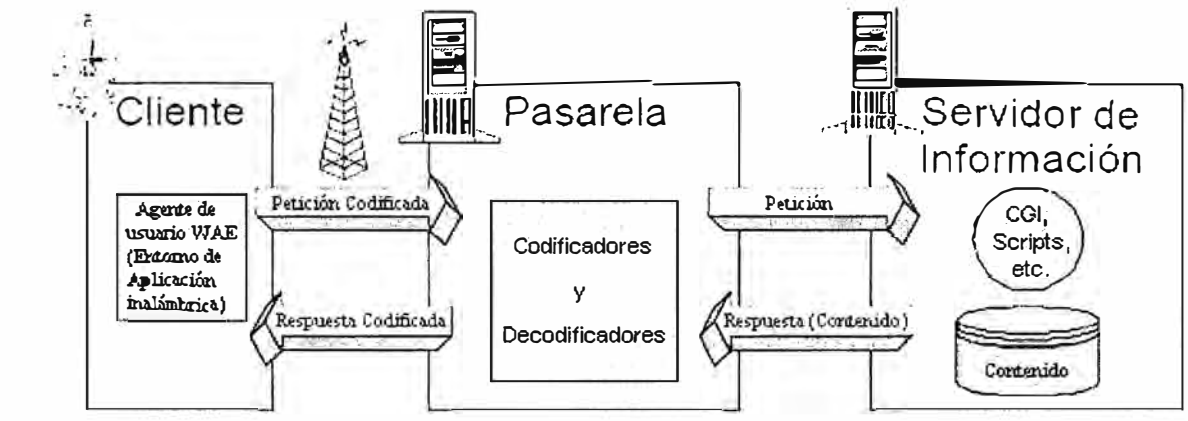


Fig. 3.1 Modelo de funcionamiento del WAP

De esta forma, en el terminal inalámbrico existiría un "micro navegador" encargado de la coordinación con la pasarela, a la cual la realiza peticiones de información que son adecuadamente tratadas y redirigidas al servidor de información adecuado. Una vez procesada la petición de información en el servidor, se envía esta información a la pasarela que de nuevo procesa adecuadamente para enviarlo al terminal inalámbrico.

Para conseguir consistencia en la comunicación entre el terminal móvil y los servidores de red que proporcionan la información, WAP define un conjunto de componentes estándar:

✓ Un modelo de nombres estándar. Se utilizan las URIs definidas en WWW para identificar los recursos locales del dispositivo (tales como funciones de control de llamada) y las URLs (también definidas en el WWW) para identificar el contenido WAP en los servidores de información.

✓ Un formato de contenido estándar, basado en la tecnología WWW.

✓ Unos protocolos de comunicación estándares, que permitan la comunicación del *micro navegador* del terminal móvil con el servidor Web en red.

Veamos ahora un modelo global de funcionamiento de este sistema en la Figura.

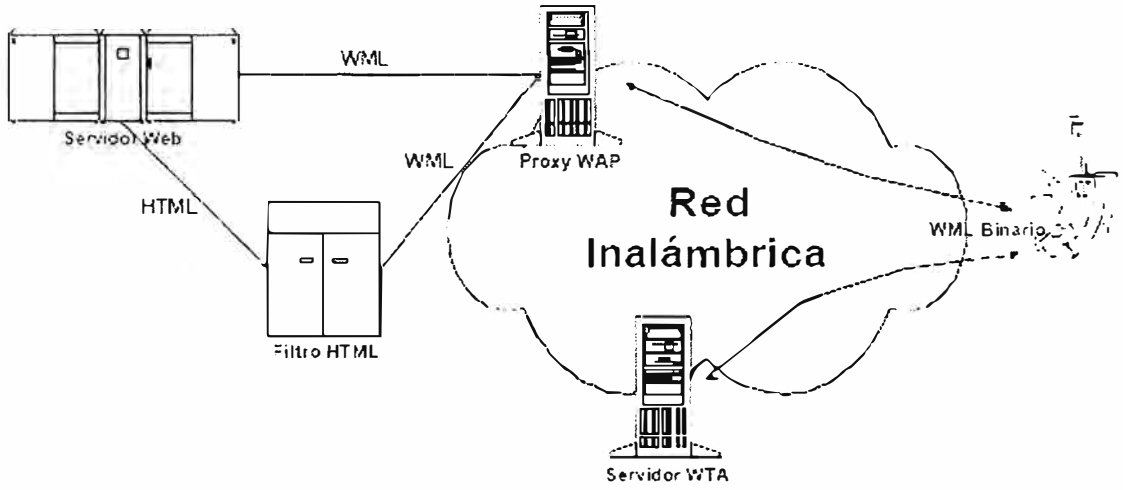


Fig. 3.2 Ejemplo de una red WAP

En el ejemplo de la figura, nuestro terminal móvil tiene dos posibilidades de conexión: a un proxy WAP, o a un servidor WTA.

El primero de ellos, el proxy WAP traduce las peticiones WAP a peticiones Web, de forma que el cliente WAP (el terminal inalámbrico) pueda realizar peticiones de información al servidor Web. Adicionalmente, este proxy codifica las respuestas del servidor Web en un formato binario compacto, que es interpretable por el cliente. Por otra parte, el segundo de ellos, el Servidor WTA está pensado para proporcionar acceso WAP a las facilidades proporcionadas por la infraestructura de telecomunicaciones del proveedor de conexiones de red.

3.2 Componentes de la arquitectura WAP

Una vez introducido el sistema, vamos a ver la arquitectura que le da consistencia. La arquitectura WAP está pensada para proporcionar un "entorno escalable y extensible para el desarrollo de aplicaciones para dispositivos de comunicación móvil". Para ello, se define una estructura en capas, en la cual cada capa es accesible por la capa superior así como por otros servicios y aplicaciones a través de un conjunto de interfaces muy bien definidos y especificados. Este esquema de capas de la arquitectura WAP la podemos ver en la Fig. 3.3.

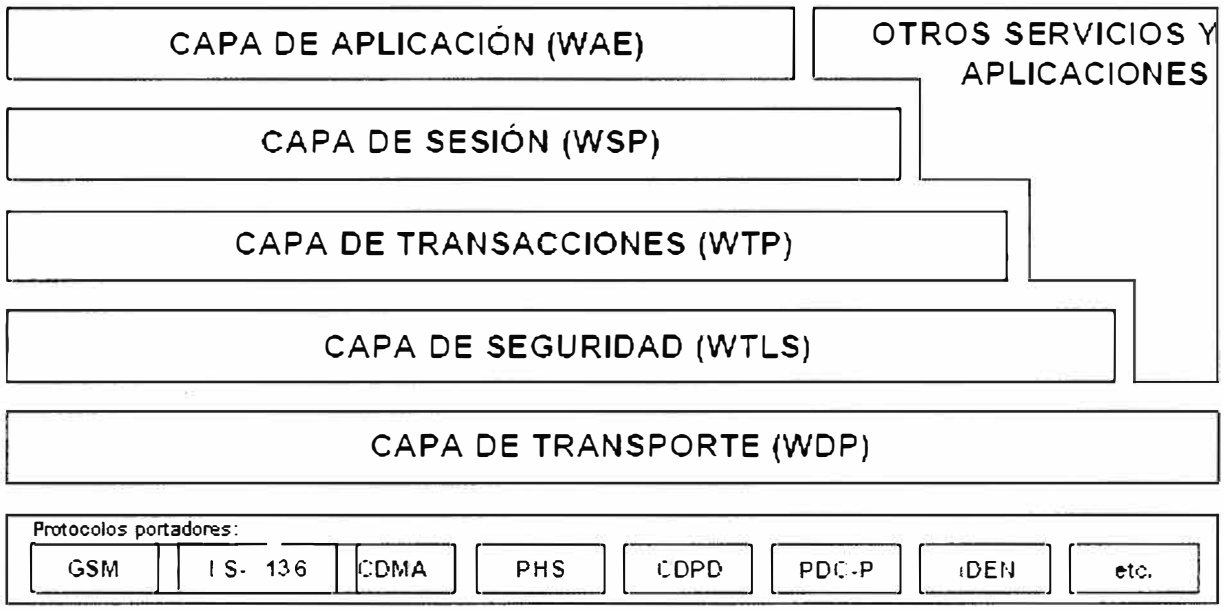


Fig. 3.3 Arquitectura de WAP

Hagamos un recorrido por estas capas de forma breve, antes de pasar a analizarlas con más profundidad.

3.2.1 Capa de Aplicación (WAE)

El *Entorno Inalámbrico de Aplicación (WAE)* es un entorno de aplicación de propósito general basado en la combinación del *World Wide Web* y tecnologías de Comunicaciones Móviles. Este entorno incluye un *micro navegador*, del cual ya hemos hablado anteriormente, que posee las siguientes funcionalidades:

✓ Un lenguaje denominado WML similar al HTML, pero optimizado para su uso en terminales móviles.

✓ Un lenguaje denominado *WMLScript*, similar al *JavaScript* (esto es, un lenguaje para su uso en forma de *Script*)

✓ Un conjunto de formatos de contenido, que son un conjunto de formatos de datos bien definidos entre los que se encuentran imágenes, entradas en la agenda de teléfonos e información de calendario.

3.2.2 Capa de Sesión (WSP)

El *Protocolo Inalámbrico de Sesión (WSP)* proporciona a la Capa de Aplicación de WAP interfaz con dos servicios de sesión: Un servicio orientado a conexión que funciona por encima de la Capa de Transacciones y un servicio no orientado a conexión que funciona por encima de la Capa de Transporte (y que proporciona servicio de datagramas seguro o servicio de datagramas no seguro)

Actualmente, esta capa consiste en servicios adaptados a aplicaciones basadas en la navegación Web, proporcionando las siguientes funcionalidades:

✓ Semántica y funcionalidades del HTTP/1.1 en una codificación compacta.

✓ Negociación de las características del Protocolo.

✓ Suspensión de la Sesión y reanudación de la misma con cambio de sesión.

3.2.3 Capa de Transacciones (WTP)

El *Protocolo Inalámbrico de Transacción (WTP)* funciona por encima de un servicio de datagramas, tanto seguros como no seguros, proporcionando las siguientes funcionalidades:

✓ Tres clases de servicio de transacciones:

- Peticiones inseguras de un solo camino.
- Peticiones seguras de un solo camino.
- Transacciones seguras de dos caminos (petición-respuesta)

✓ Seguridad usuario-a-usuario opcional.

✓ Transacciones asíncronas.

3.2.4 Capa de Seguridad (WTLS)

La Capa Inalámbrica de Seguridad de Transporte (WTLS) es un protocolo basado en el estándar SSL, utilizado en el entorno Web para la proporción de seguridad en la realización de transferencias de datos. Este protocolo ha sido especialmente diseñado para los protocolos de transporte de WAP y optimizado para ser utilizado en canales de comunicación de banda estrecha. Para este protocolo se han definido las siguientes características:

✓ *Integridad de los datos*. Este protocolo asegura que los datos intercambiados entre el terminal y un servidor de aplicaciones no han sido modificados y no es información corrupta.

✓ *Privacidad de los datos*. Este protocolo asegura que la información intercambiada entre el terminal y un servidor de aplicaciones no puede ser entendida por terceras partes que puedan interceptar el flujo de datos.

✓ *Autentificación.* Este protocolo contiene servicios para establecer la autenticidad del terminal y del servidor de aplicaciones.

Adicionalmente, el WTLS puede ser utilizado para la realización de comunicación segura entre terminales, por ejemplo en el caso de operaciones de comercio electrónico entre terminales móviles.

3.2.5 Capa de Transporte (WDP)

El Protocolo Inalámbrico de Datagramas (WDP) proporciona un servicio fiable a los protocolos de las capas superiores de WAP y permite la comunicación de forma transparente sobre los protocolos portadores válidos.

Debido a que este protocolo proporciona un interfaz común a los protocolos de las capas superiores, las capas de Seguridad, Sesión y Aplicación pueden trabajar independientemente de la red inalámbrica que dé soporte al sistema.

Antes de pasar a estudiar en más profundidad cada uno de estos protocolos, veamos tres ejemplos de interconexión de estas capas en la siguiente figura:

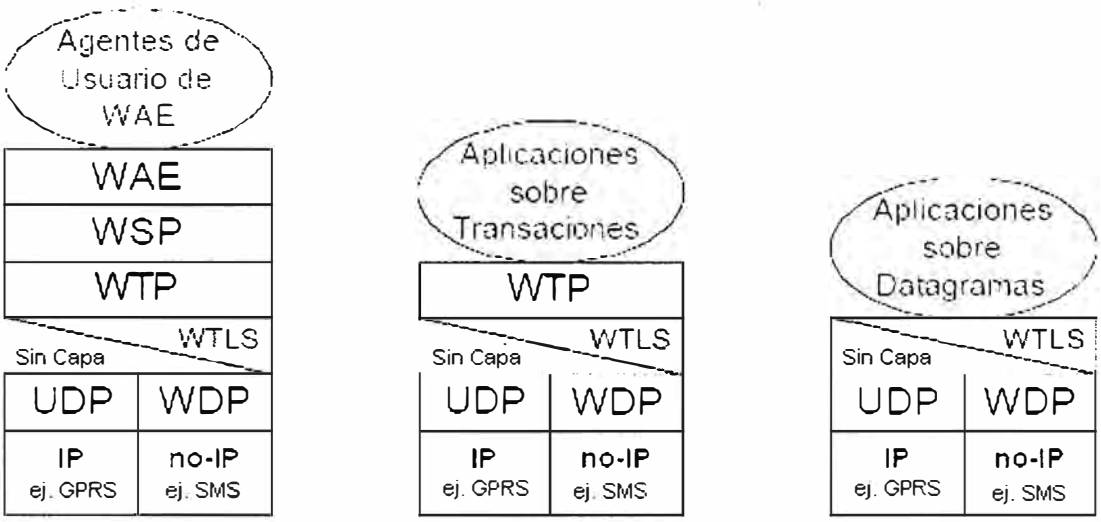


Fig. 3.4 Ejemplo de capas en WAP

Así pues, dependiendo de la aplicación en cuestión, la comunicación se realizará con una determinada capa de la estructura de WAP.

3.3 El Entorno Inalámbrico de Aplicaciones

El objetivo del Entorno Inalámbrico de Aplicaciones es construir un entorno de aplicación de propósito general, basado fundamentalmente en la filosofía y tecnología del World Wide Web (WWW). Principalmente, se pretende establecer un entorno que permita a

los operadores y proveedores de servicios construir aplicaciones y servicios que puedan utilizarse en una amplia variedad de plataformas inalámbricas de forma útil y eficiente.

De esta forma, la arquitectura del Entorno Inalámbrico de Aplicaciones (en adelante WAE) está enfocado principalmente sobre los aspectos del cliente de la arquitectura del sistema de WAP, esto es, de los puntos relacionados con los agentes de usuario. Esto es debido a que la parte que más interesa de la arquitectura es aquella que afecta principalmente a los terminales móviles, esto es, a aquellos puntos en los cuales van a estar ejecutándose los diversos agentes de usuario.

Si volvemos sobre la Fig. 3.1, vemos que entre los agentes de usuario localizados en el cliente (en el terminal móvil) y los servidores de información se define un nuevo elemento: Las Pasarelas. Su función es codificar y decodificar la información intercambiada con el cliente, para así minimizar la cantidad de datos radiados, así como minimizar el proceso de la información por parte del cliente.

Basándonos en esta arquitectura, vamos a profundizar un poco más en los componentes de este Entorno Inalámbrico de Aplicación. Tal y como podemos observar en la Fig. 3.5, se divide en dos partes, dos capas lógicas:

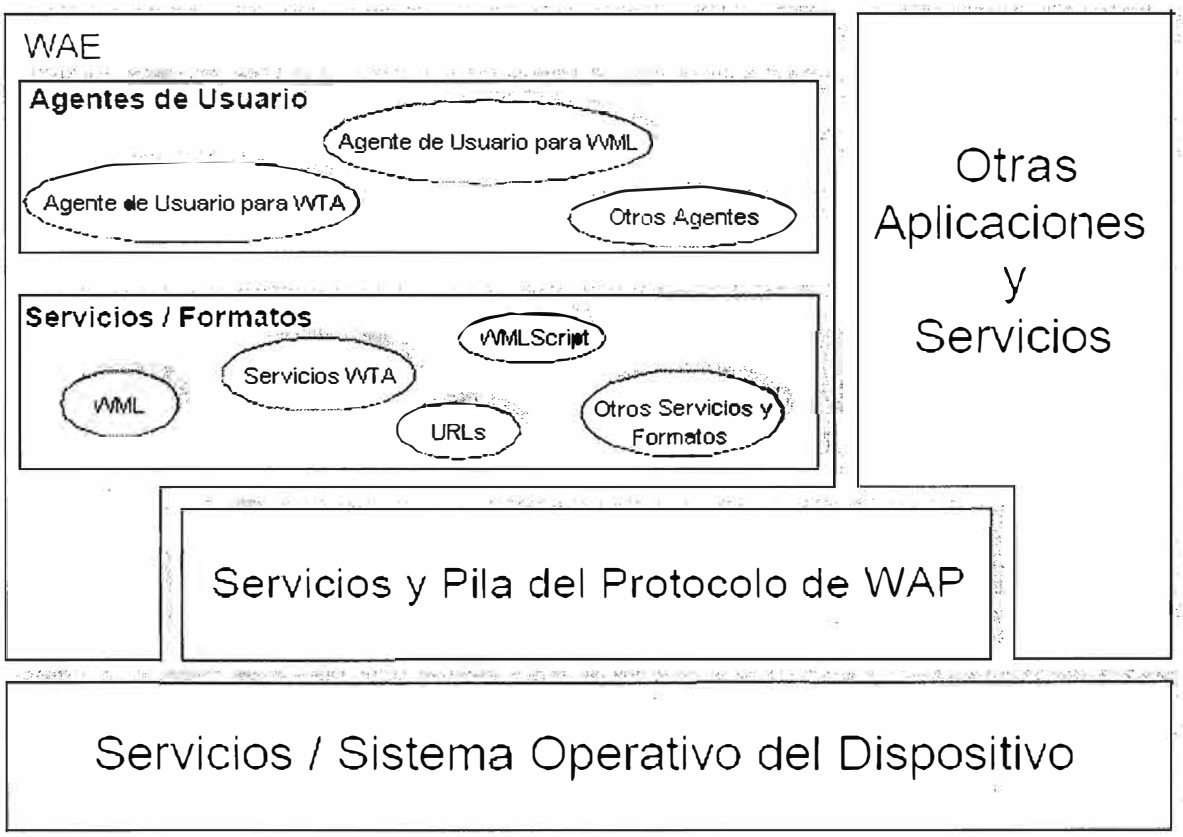


Fig. 3.5 Componentes del Cliente de WAE

✓ Los Agentes de Usuario, que incluye aquellos elementos como navegadores, agendas telefónicas, editores de mensajes, etc.

✓ Los Servicios y Formatos, que incluyen todos aquellos elementos y formatos comunes, accesibles a los Agentes de Usuario, tales como WML, WMLScript, formatos de imagen, etc.

Como se puede ver en la figura, dentro de WAE se separan Servicios de Agentes de Usuario, lo que proporciona flexibilidad para combinar varios Servicios dentro de un unico Agente de Usuario, o para distribuir los Servicios entre varios Agentes de Usuario.

Los dos Agentes de Usuario más importantes son el Agente de Usuario para WML y el Agente de Usuario para WTA.

El Agente de Usuario para WML es el Agente de Usuario fundamental en la arquitectura del Entorno Inalámbrico de Aplicación. A pesar de su importancia, este Agente de Usuario no está definido formalmente dentro de esta arquitectura, ya que sus características y capacidades se dejan en manos de los encargados de su implementación. El único requisito de funcionalidad que debe cumplir este Agente de Usuario, es el proporcionar un sistema intérprete a los lenguajes WML y WMLScript, de forma que se permita la navegación desde el terminal móvil.

Por otra parte, el Agente de Usuario para WTA permite a los autores acceder e interactuar con las características de los teléfonos móviles (p. e. Control de Llamada), así como otras aplicaciones supuestas en los teléfonos, tales como agendas de teléfono y aplicaciones de calendario.

3.4 El protocolo Inalámbrico de Sesión

El *Protocolo Inalámbrico de Sesión* constituye la capa que se sitúa por debajo de la capa de Aplicación, proporcionando la capacidad necesaria para:

✓ Establecer una conexión fiable entre el cliente y el servidor, y liberar esta conexión de una forma ordenada.

✓ Ponerse de acuerdo en un nivel común de funcionalidades del protocolo, a través de la negociación de las posibilidades.

✓ Intercambiar contenido entre el cliente y el servidor utilizando codificación compacta.

✓ Suspender y recuperar la sesión.

Hoy por hoy, este protocolo ha sido definido únicamente para el caso de la navegación, definiéndose como WSP/B. Esta implementación está realizada para el establecimiento de una conexión sobre la base de un protocolo compatible con HTTP1.1.

De esta forma, se han definido un conjunto de primitivas de servicio para permitir la comunicación entre la capa de sesión integrada dentro del equipo cliente y la capa de sesión integrada en el equipo servidor. Estas primitivas, junto con una pequeña descripción de las mismas, pueden verse en la Tabla 3.1:

Nombre de la primitiva	**Descripción**
S-Connect	Esta primitiva se utiliza para iniciar el establecimiento de la conexión, y para la notificación de su éxito
S-Disconnect	Esta primitiva se utiliza para desconectar una sesión, y para notificar al usuario de una sesión que esa sesión no se puede establecer, que ha sido desconectada
S-Suspend	Esta primitiva se utiliza para solicitar la suspensión de la sesión
S-Resume	Esta primitiva se utiliza para solicitar que se recupere la sesión utilizando para las direcciones el nuevo identificador de punto de acceso de servicio.
S-Exception	Esta primitiva se utiliza para notificar aquellos eventos que no están asignados a una transacción en particular, ni provocan la desconexión o suspensión de la sesión.
S-MethodInvoke	Esta primitiva se utiliza para solicitar una operación que deba ser ejecutada en el servidor.
S-MethodResult	Esta primitiva se utiliza para devolver una respuesta a una petición de operación.
S-MethodAbort	Esta primitiva se utiliza para abortar una solicitud de ejecución operación, que no haya sido aún completada.
S-Push	Esta primitiva se utiliza para enviar información no solicitada desde el servidor, dentro del contexto de una sesión de forma y Sin confirmación.
S-ConfirmedPush	Esta primitiva realiza las mismas funciones que la anterior, pero con confirmación.
S-PushAbort	Esta primitiva se utiliza para anular una primitiva anterior del tipo *S-Push* o *SConfirmedPush.*

TABLA N° 3.1 Primitivas de Servicio de Sesión

Adicionalmente, existen cuatro tipos de cada una de estas primitivas, tal y como puede verse en la Tabla 3.2:

Tipo	**Abreviación**	**Descripción**
Request	req	Se utiliza cuando una capa superior solicita un servicio de la capa inmediatamente inferior
Indication	ind	Una capa que solicita un servicio utiliza este tipo primitiva para notificar a la capa inmediatamente superior de las actividades relacionadas con su par, o con el proveedor del servicio
Response	Res	Este tipo de primitiva se utiliza para reconocer la recepción de la primitiva de tipo *Indication* de la capa inmediatamente inferior
Confirm	Cnf	La capa que proporciona el servicio requerido utiliza este tipo de primitiva para notificar que la actividad ha sido completada satisfactoriamente.

TABLA Nº 3.2 Tipos de Primitivas de Servicio.

Por último, reseñar que cada una de estas primitivas está perfectamente definida dentro de la especificación, tanto desde el punto de vista del diagrama de tiempos en el que se tienen que invocar las primitivas, como desde el punto de vista de los parámetros intercambiados.

3.5 El Protocolo Inalámbrico de Transacción

El *Protocolo Inalámbrico de Transacción* se establece para proporcionar los servicios necesarios que soporten aplicaciones de "*navegación*" (del tipo petición/respuesta). Es a este dúo petición/respuesta, lo que vamos a denominar como transacción. Este protocolo se sitúa por encima del *Protocolo Inalámbrico de Datagramas* y, de forma opcional, de la *Capa Inalámbrica de Seguridad de Transporte*, que serán estudiados posteriormente.

Las características de este protocolo son:

✓ Proporciona tres clases de servicios de transacción:

- Clase 0: mensaje de solicitud no seguro, sin mensaje de resultado.
- Clase 1: mensaje de solicitud seguro, sin mensaje de resultado.
- Clase 2: mensaje de solicitud seguro, con, exactamente, un mensaje de resultado seguro.

✓ La seguridad se consigue a través del uso de identificadores únicos de transacción, asentimientos, eliminación de duplicados y retransmisiones.

✓ Seguridad opcional usuario a usuario.

✓ De forma opcional, el último asentimiento de la transacción puede contener algún tipo de información adicional relacionada con la transacción, como medidas de prestaciones, etc.

✓ Se proporcionan mecanismos para minimizar el número de transacciones que se reenvían como resultado de paquetes duplicados.

✓ Se permiten las transacciones asíncronas.

Al igual que en el protocolo anterior (el protocolo inalámbrico de sección), en la Tabla 3.3 vamos a ver las primitivas de servicio que sustentan la comunicación entre dos capas de transacciones situadas en dos equipos distintos:

Nombre de la primitiva	**Descripción**
TR-Invoke	Esta primitiva se utiliza para iniciar una nueva transacción.
TR-Result	Esta primitiva se utiliza para devolver el resultado de transacción iniciada anteriormente
TR-Abort	Esta primitiva se utiliza para abortar una transacción existente

TABLA Nº 3.3 Primitivas de Servicio de Transacción

A modo de ejemplo, vamos a ver en la Fig. 3.6 la concatenación de Primitivas de Servicio de Sesión y de Transacción para el caso de una petición-respuesta:

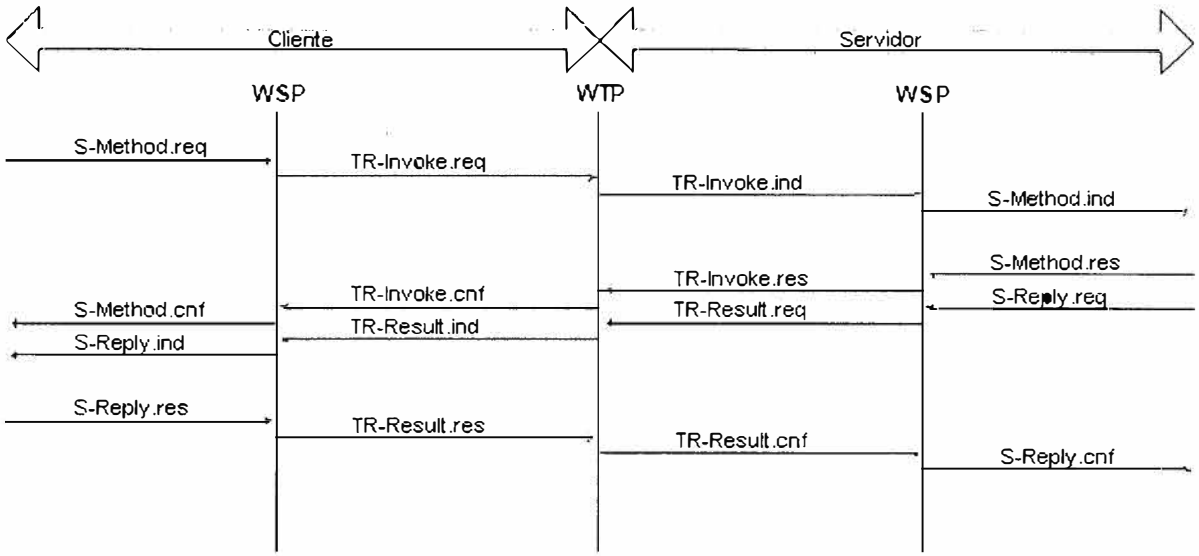


Fig. 3.6 Ejemplo de intercambio de primitivas entre capa Sesión y Transacción

Para finalizar, vamos a detallar un poco más las principales características de este protocolo:

✓ Transferencia de Mensajes.

Dentro de este protocolo se distinguen dos tipos de mensajes: mensajes de datos y mensajes de control. Los mensajes de datos transportan únicamente datos de usuario, mientras que los mensajes de control se utilizan para los asentimientos, informes de error, etc. pero sin transportar datos de usuario.

✓ Retransmisión hasta el asentimiento.

Esta característica se utiliza para la transferencia fiable de datos desde un proveedor WTP a otro, en caso que haya pérdida de paquetes. A modo de comentario, dejar claro que para reducir lo máximo posible el número de paquetes que se transmiten, este protocolo utiliza asentimiento explícito siempre que sea posible.

✓ Asentimiento de usuario.

El Asentimiento de Usuario permite al usuario de este protocolo, confirmar cada mensaje recibido por el proveedor WTP.

✓ Información en el Último Asentimiento.

Se permite, así pues, enviar información en el último, y únicamente en el último, asentimiento de una transacción. De esta forma, se puede enviar, por ejemplo, información del rendimiento proporcionado por el sistema durante la transacción realizada, etc.

✓ Concatenación y Separación.

Podemos definir concatenación como el proceso de trasmitir múltiples Unidades de Datos del Protocolo (PDU) de WTP en una Unidad de Datos del Servicio (SDU) de la red portadora.

Por el contrario, separación es el proceso de separar múltiples PDUs de un único SDU (esto es, el proceso inverso al anterior).

El objetivo de estos sistemas es proveer eficiencia en la transmisión inalámbrica, al requerirse un menor número de transmisiones.

✓ Transacciones Asíncronas.

Para un correcto funcionamiento del protocolo, múltiples transacciones deben ser procesadas de forma asíncrona, debe ser capaz de iniciar múltiples transacciones antes que reciba la respuesta a la primera transacción.

✓ Identificador de la Transacción

Cada transacción está identificada de forma única por los pares de direcciones de los sockets (Dirección fuente, puerto fuente, dirección destino y puerto destino) y por el Identificador de Transacción (TID), el cual se incrementa para cada una de las transacciones iniciadas. Este número es de 16 bits, utilizándose el bit de mayor orden para indicar la dirección.

✓ Segmentación y re-ensamblado. (opcional)

Si la longitud del mensaje supera la Unidad Máxima de Transferencia (MTU), el mensaje puede ser segmentado por el WTP y enviado en múltiples paquetes. Cuando esta operación se realiza, estos paquetes pueden ser enviados y asentidos en grupos. De esta forma, el emisor puede realizar control de flujo cambiando el tamaño de los grupos de mensajes dependiendo de las características de la red.

3.6 La Capa Inalámbrica de Seguridad de Transporte

La *Capa Inalámbrica de Seguridad de Transporte* (en adelante WTLS), constituye una capa modular, que depende del nivel de seguridad requerido por una determinada aplicación. Esta capa proporciona a las capas de nivel superior de WAP de una interfaz de servicio de transporte seguro, que lo resguarde de una interfaz de transporte inferior.

El principal objetivo de esta capa es proporcionar privacidad, integridad de datos y autentificación entre dos aplicaciones que se comuniquen. Adicionalmente, la WTLS proporciona una interfaz para el manejo de conexiones seguras.

Al igual que hemos hecho en los protocolos anteriores, en la Tabla 3.4 vamos a ver las primitivas de servicio que sustentan la comunicación entre dos capas situadas en dos equipos distintos:

Nombre de la primitiva	Descripción
SEC-Unitdata	Esta primitiva se utiliza para intercambiar datos de usuario entre los dos participantes. Sólo puede ser invocada cuando existe previamente una conexión segura entre las direcciones de transporte de lo dos participantes.
SEC-Create	Esta primitiva se utiliza para iniciar el establecimiento una conexión segura.
SEC-Exchange	Esta primitiva se utiliza en la creación de una conexión segura si el servidor desea utilizar autentificación de clave pública o intercambio de claves con el cliente.
SEC-Commit	Esta primitiva se inicia cuando el handshake se complet y cualquiera de los equipos participantes solicita cambia a un nuevo estado de conexión negociado.
SEC-Terminate	Esta primitiva se utiliza para finalizar la conexión.
SEC-Exception	Esta primitiva se utiliza para informar al otro extremo sobre las alertas de nivel de aviso.
SEC-Create-Request	Esta primitiva se utiliza por el servidor para solicitar al cliente que inicie un nuevo handshake.

TABLA Nº 3.4 Primitivas de Servicio de Capa de Seguridad

Hemos hablado anteriormente del proceso de establecimiento de una sesión segura o *handshake*. En la Fig. 3.7 podemos ver este intercambio de primitivas:

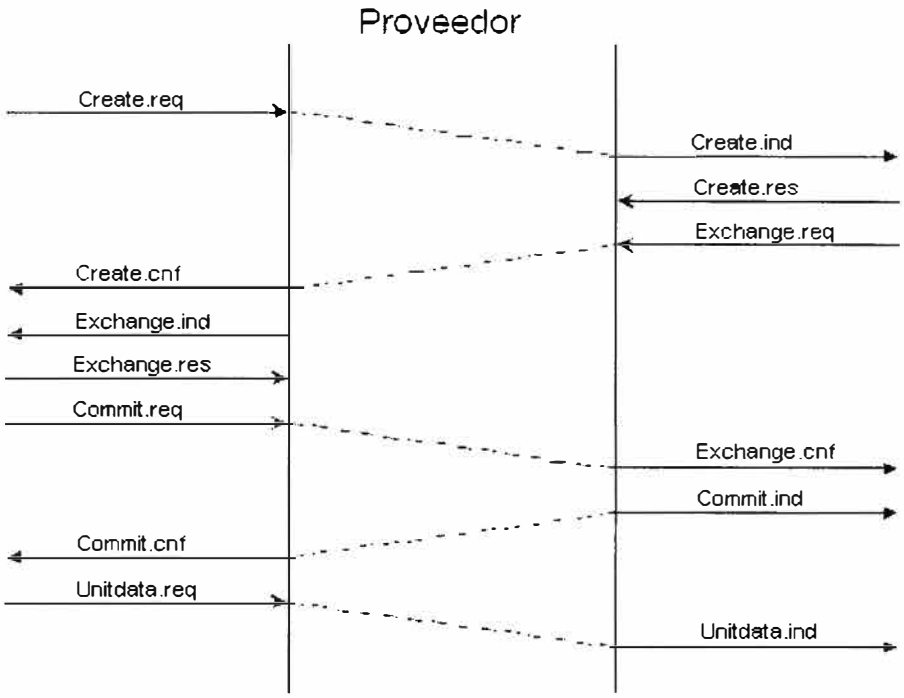


Fig. 3.7 Secuencia de Primitivas para el establecimiento de una sesión segura

3.7 El protocolo Inalámbrico de Datagramas

El *Protocolo Inalámbrico de Datagramas* (en adelante WDP) ofrece un servicio consistente al protocolo (Seguridad, Transacción y Sesión) de la capa superior de WAP, comunicándose de forma transparente sobre uno de los servicios portadores disponibles.

Este protocolo ofrece servicios a los protocolos superiores del estilo a direccionamiento por número de puerto, segmentación y re-ensamblado opcional y detección de errores opcional, de forma que se permite a las aplicaciones de usuario funcionar de forma transparente sobre distintos servicios portadores disponibles. Para ello, se plantea una arquitectura de protocolo como el que se muestra en la Fig. 3.8:

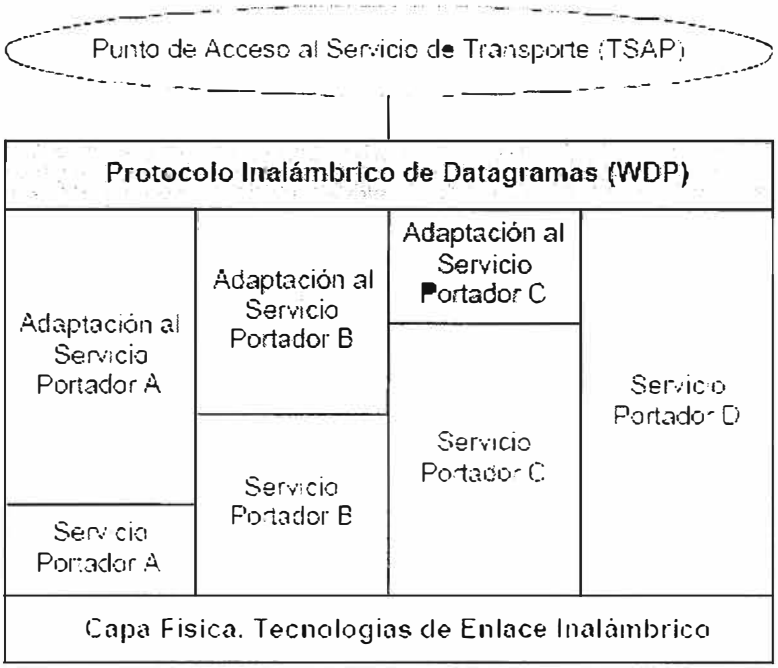


Fig. 3.8 Arquitectura del Protocolo Inalámbrico de Datagramas

Al igual que hemos hecho en los protocolos anteriores, en la Tabla 3.5 vamos a ver las primitivas de servicio que se utilizan en este protocolo:

Nombre de la primitiva	**Descripción**
T-DUnitdata	Esta primitiva es la utilizada para transmitir datos con datagramas. No requiere que exista una conexión para establecerse.
T-DError	Esta primitiva se utiliza para proporcionar información a la capa superior cuando ocurre un error que pueda influenciar en el servicio requerido.

TABLA Nº 3.5 Primitivas de Servicio de la Capa de Datagramas

Por último, vamos a ver la arquitectura de este protocolo dentro de la arquitectura global de WAP, para el caso de utilizarse GSM como servicio portador, que es el protocolo que más nos puede interesar por su amplia implantación en los sistemas de comunicaciones móviles telefónicas existentes hoy en día.

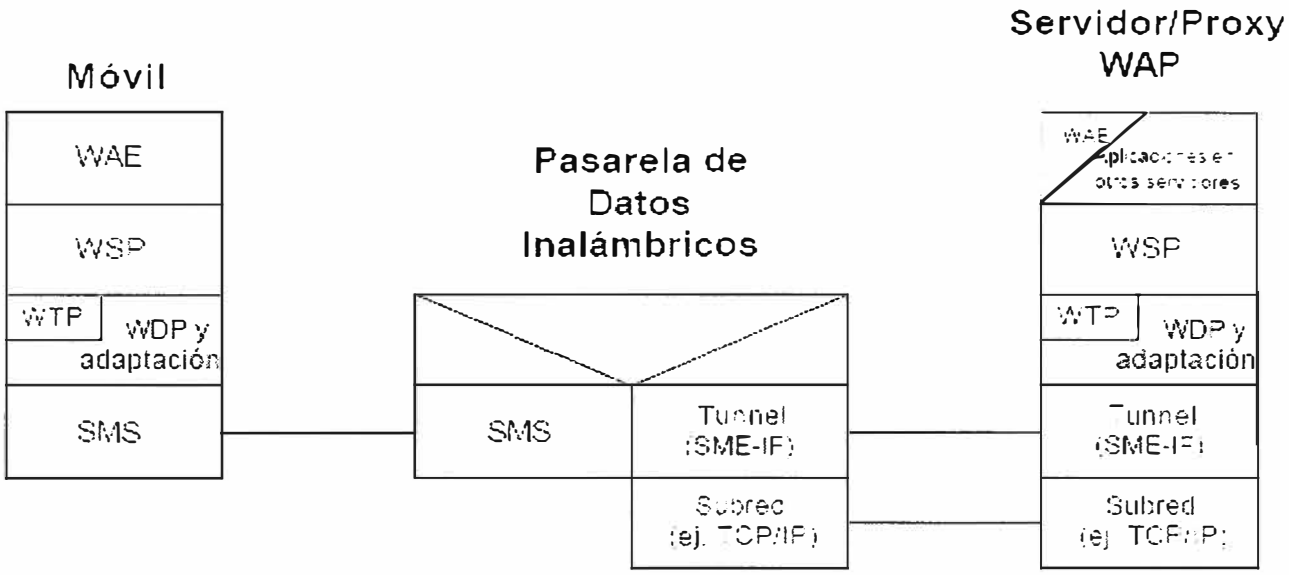


Fig. 3.9 WDP sobre GSM SMS

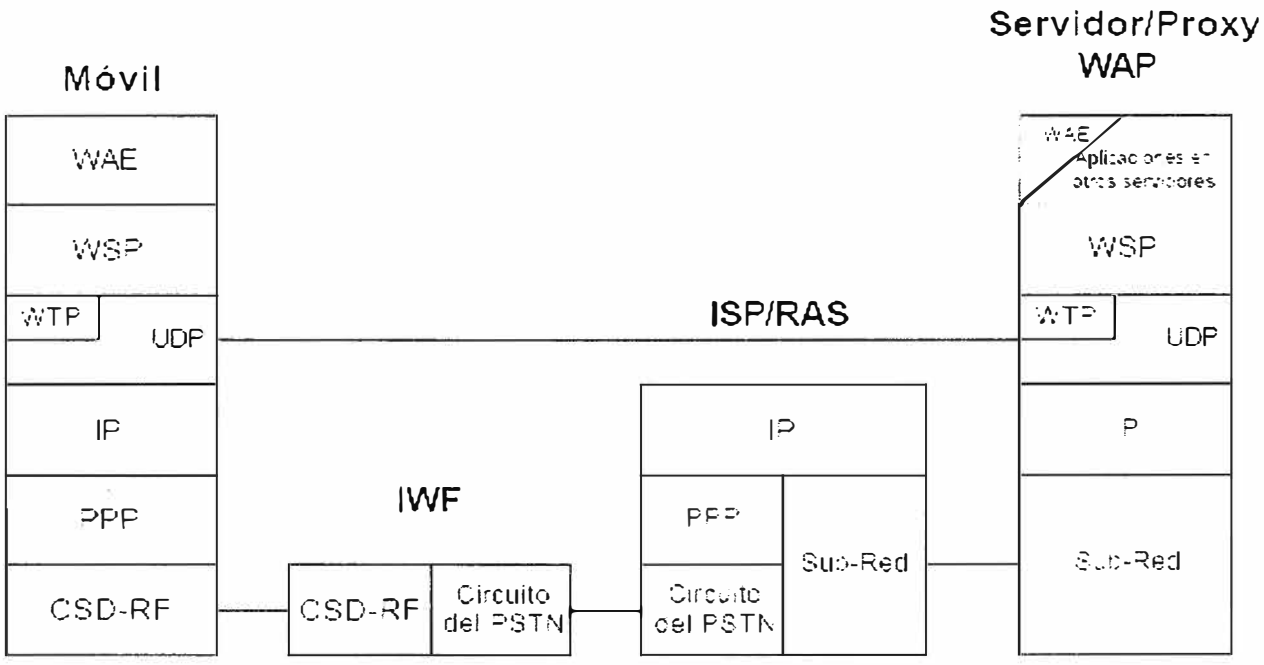


Fig. 3.10 WDP sobre GSM Canal de Datos de Circuitos Conmutados

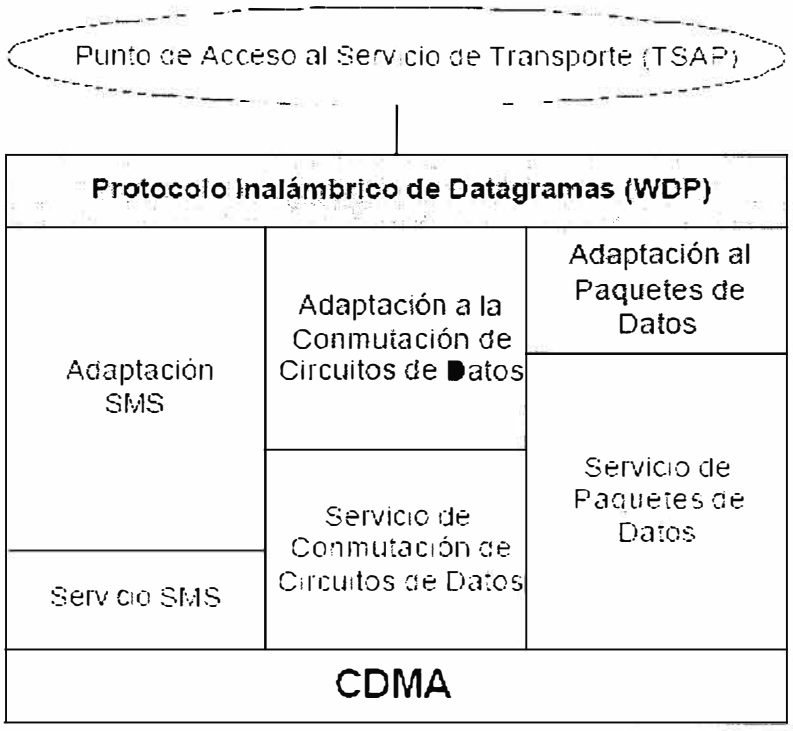


Fig. 3.11 WDP sobre Servicios Portadores CDMA

CAPITULO IV
.NET

4.1 Introducción a .NET

.NET es un proyecto de Microsoft para crear una nueva plataforma de desarrollo de software con énfasis en transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones. Basado en ella, la empresa intenta desarrollar una estrategia horizontal que integre todos sus productos, desde el sistema operativo hasta las herramientas de mercado.

.NET podría considerarse una respuesta de Microsoft al creciente mercado de los negocios en entornos Web, como competencia a la plataforma Java de Sun Microsystems y a los diversos framework de desarrollo web basados en PHP. Su propuesta es ofrecer una manera rápida y económica, a la vez que segura y robusta, de desarrollar aplicaciones –o como la misma plataforma las denomina, soluciones– permitiendo una integración más rápida y ágil entre empresas y un acceso más simple y universal a todo tipo de información desde cualquier tipo de dispositivo.

NET es una plataforma de desarrollo que Microsoft presentó en el 2000 en su Conferencia de Desarrolladores Profesionales (PDC). NET Framework se encuentra actualmente en su versión 3.5. La siguiente figura muestra una arquitectura simple de NET Framework para desarrollo de aplicaciones.

Cuando la petición de una página ASP.NET proviene de un cliente web como IE o un dispositivo móvil, IIS recoge la solicitud y lo remitirá al NET Framework. Si la página ASP.NET es solicitada por primera vez entonces NET Framework compilará la página ASP.NET en el lenguaje intermedio (IL). Luego el código IL será compilado a código nativo por un compilador Just-In-Time (JIT). Como puede ver, el NET Framework es el corazón de la aplicación ASP.NET. NET Framework proveerá los recursos necesarios y el compilador de lenguaje de su elección para compilar el código.

Una cosa interesante de los dispositivos móviles es su habilidad de conectarse a Internet y ejecutar aplicaciones Web.

Las aplicaciones móviles ahora pueden ser usadas para enviar cualquier tipo de datos, a cualquier usuario, en cualquier parte del mundo.

Los distintos dispositivos móviles soportan diversos lenguajes de programación. Algunos soportan WAP y WML, algunos soportan HTML o una versión limitada de HTML, y algunos soportan ambos o un lenguaje distinto.

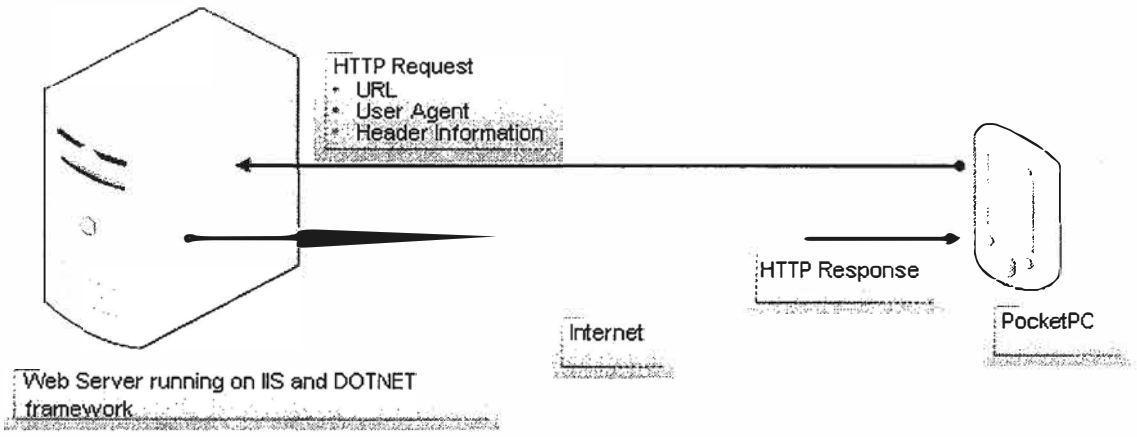


Fig. 4.1 Proceso involucrado en la navegación de una página Web desde una Pocket PC.

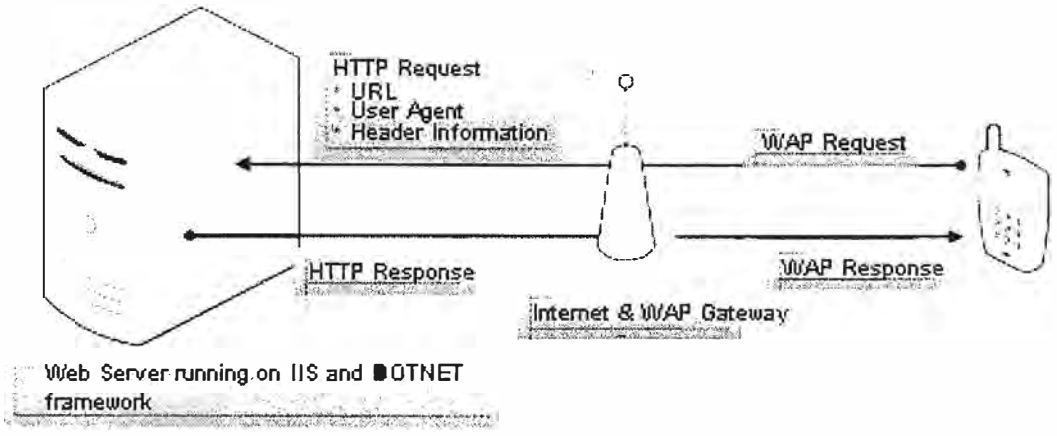


Fig. 4.2 Proceso involucrado en la navegación de una página Web desde un teléfono celular

El WAP gateway traduce la solicitud a HTTP y lo pasa al servidor web sobre Internet. Este gateway es proporcionado por los proveedores de servicios de telefonía móvil.

4.2 Consideraciones

La plataforma .NET de Microsoft es un componente de software que puede ser añadido al sistema operativo Windows. Provee un extenso conjunto de soluciones predefinidas para necesidades generales de la programación de aplicaciones, y administra la ejecución de los programas escritos específicamente con la plataforma. Esta solución es el producto principal en la oferta de Microsoft, y pretende ser utilizada por la mayoría de las aplicaciones creadas para la plataforma Windows.

.NET Framework se incluye en Windows Server 2008 y Windows Vista. De igual manera, la versión actual de dicho componente puede ser instalada en Windows XP, y en la familia de sistemas operativos Windows Server 2003. Una versión "reducida" de .NET Framework está disponible para la plataforma Windows Mobile, incluyendo teléfonos inteligentes.

La norma (incluido en ECMA-335, ISO/IEC 23271) que define el conjunto de funciones que debe implementar la biblioteca de clases base (BCL por sus siglas en inglés, tal vez el más importante de los componentes de la plataforma), define un conjunto funcional mínimo que debe implementarse para que el marco de trabajo sea soportado por un sistema operativo. Aunque Microsoft implementó esta norma para su sistema operativo Windows, la publicación de la norma abre la posibilidad de que sea implementada para cualquier otro sistema operativo existente o futuro, permitiendo que las aplicaciones corran sobre la plataforma independientemente del sistema operativo para el cual haya sido implementada. El Proyecto Mono emprendido por Ximian pretende realizar la implementación de la norma para varios sistemas operativos adicionales bajo el marco del software libre o código abierto.

4.3 Componentes

Los principales componentes del marco de trabajo son:

- El conjunto de lenguajes de programación
- La Biblioteca de Clases Base o BCL
- El Entorno Común de Ejecución para Lenguajes o CLR por sus siglas en inglés.

Debido a la publicación de la norma para la infraestructura común de lenguajes (CLI por sus siglas en inglés), el desarrollo de lenguajes se facilita, por lo que el marco de trabajo .NET soporta ya más de 20 lenguajes de programación y es posible desarrollar cualquiera de

los tipos de aplicaciones soportados en la plataforma con cualquiera de ellos, lo que elimina las diferencias que existían entre lo que era posible hacer con uno u otro lenguaje.

Algunos de los lenguajes desarrollados para el marco de trabajo .NET son: C#, Visual Basic, Delphi (Object Pascal), C++, J#, Perl, Python, Fortran, Cobol y PowerBuilder.

4.4 ¿Cómo trabaja?

La siguiente tabla muestra como trabaja .NET Mobile:

Dispositivos Móviles

Internet

Internet Information Services - IIS

.NET Framework

ASP.NET

.NET Mobile

TABLA N° 4.1 Capas .NET Mobile

- Un cliente solicita una pagina web
- La petición viaja por Internet
- La petición es recibida por IIS
- La petición es manejada por .NET framework
- La pagina solicitada es compilada por ASP.NET
- .NET Mobile maneja los requerimientos del dispositivo móvil

La página es retornada al cliente

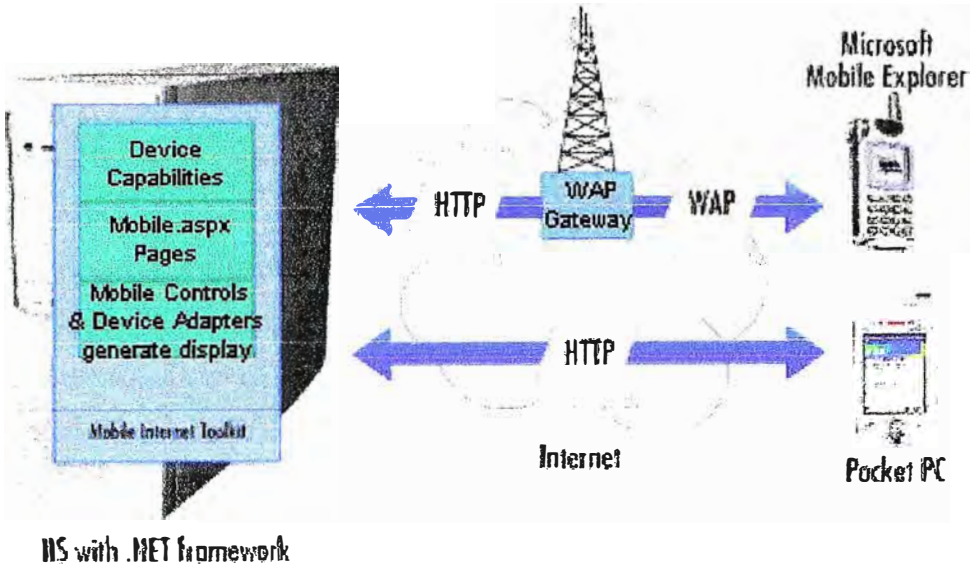


Fig. 4.3 Arquitectura de una aplicación Web móvil

4.5 Requisitos Para Desarrollar Una Aplicación Móvil Con .Net

- Microsoft Windows XP Professional
- Internet Information Services
- Visual Studio 2005
- SQL Server 2005
- SQL Server 2005 Mobile Edition
- ActiveSync 4.0 or higher

ActiveSync 4.0 or higher permite la conectividad entre un dispositivo basado en Windows Mobile y la computadora.

- Windows Mobile 5.0 SDKs

4.6 Microsoft Visual Studio

Microsoft Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros.

Visual Studio permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión net 2002). Así se pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles.

4.7 El futuro de .NET

A largo plazo Microsoft pretende reemplazar el API Win32 o Windows API con la plataforma .NET. Esto debido a que el API Win32 o Windows API fue desarrollada sobre la marcha, careciendo de documentación detallada, uniformidad y cohesión entre sus distintos componentes, provocando múltiples problemas en el desarrollo de aplicaciones para el sistema operativo Windows. La plataforma .NET pretende solventar la mayoría de estos problemas proveyendo un conjunto único y expandible con facilidad, de bloques interconectados, diseñados de forma uniforme y bien documentados, que permitan a los desarrolladores tener a mano todo lo que necesitan para producir aplicaciones sólidas.

Debido a las ventajas que la disponibilidad de una plataforma de este tipo puede darle a las empresas de tecnología y al público en general, muchas otras empresas e instituciones se han unido a Microsoft en el desarrollo y fortalecimiento de la plataforma .NET, ya sea por medio de la implementación de la plataforma para otros sistemas operativos aparte de Windows (Proyecto Mono de Ximian/Novell para Linux/MacOS X/BSD/Solaris), el desarrollo de lenguajes de programación adicionales para la plataforma (ANSI C de la Universidad de Princeton, NetCOBOL de Fujitsu, Delphi de Borland, PowerBuilder de Sybase entre otros) o la creación de bloques adicionales para la plataforma (como controles, componentes y bibliotecas de clases adicionales); siendo algunas de ellas software libre, distribuibles bajo la licencia GPL.

Con esta plataforma Microsoft incursiona de lleno en el campo de los Servicios Web y establece el XML como norma en el transporte de información en sus productos y lo promociona como tal en los sistemas desarrollados utilizando sus herramientas.

CAPITULO V
JAVA

5.1 Introducción a JAVA

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Las aplicaciones Java están típicamente compiladas en un bytecode, aunque la compilación en código máquina nativo también es posible. En el tiempo de ejecución, el bytecode es normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución directa por hardware del bytecode por un procesador Java también es posible.

La implementación original y de referencia del compilador, la máquina virtual y las librerías de clases de Java fueron desarrolladas por Sun Microsystems en 1995. Desde entonces, Sun ha controlado las especificaciones, el desarrollo y evolución del lenguaje a través del Java Community Process, si bien otros han desarrollado también implementaciones alternativas de estas tecnologías de Sun, algunas incluso bajo licencias de software libre.

Entre noviembre de 2006 y mayo de 2007, Sun Microsystems liberó la mayor parte de sus tecnologías Java bajo la licencia GNU GPL, de acuerdo con las especificaciones del Java Community Process, de tal forma que prácticamente todo el Java de Sun es ahora software libre (aunque la biblioteca de clases de Sun que se requiere para ejecutar los programas Java todavía no es software libre).

5.2 Historia

La tecnología Java se creó como una herramienta de programación para ser usada en un proyecto de set-top-box en una pequeña operación denominada the Green Project en Sun Microsystems en el año 1991. El equipo (Green Team), compuesto por trece personas y dirigido por James Gosling, trabajó durante 18 meses en Sand Hill Road en Menlo Park en su desarrollo.

El lenguaje se denominó inicialmente Oak (por un roble que había fuera de la oficina de Gosling), luego pasó a denominarse Green tras descubrir que Oak era ya una marca comercial registrada para adaptadores de tarjetas gráficas y finalmente se renombró a Java.

El término Java fue acuñado en una cafetería frecuentada por algunos de los miembros del equipo. Pero no está claro si es un acrónimo o no, aunque algunas fuentes señalan que podría tratarse de las iniciales de sus creadores: *James Gosling, Arthur Van Hoff, y Andy Bechtolsheim*. Otros abogan por el siguiente acrónimo, *Just Another Vague Acronym* ("sólo otro acrónimo ambiguo más"). La hipótesis que más fuerza tiene es la que Java debe su nombre a un tipo de café disponible en la cafetería cercana, de ahí que el icono de java sea una taza de cafe caliente. Un pequeño signo que da fuerza a esta teoría es que los 4 primeros bytes (el número mágico) de los archivos .class que genera el compilador, son en hexadecimal, 0xCAFEBABE. Otros simplemente dicen que el nombre fue sacado al parecer de una lista aleatoria de palabras.

Los objetivos de Gosling eran implementar una máquina virtual y un lenguaje con una estructura y sintaxis similar a C++. Entre junio y julio de 1994, tras una sesión maratoniana de tres días entre John Gaga, James Gosling, Joy Naughton, Wayne Rosing y Eric Schmidt, el equipo reorientó la plataforma hacia la Web. Sintieron que la llegada del navegador web Mosaic, propiciaría que Internet se convirtiese en un medio interactivo, como el que pensaban era la televisión por cable. Naughton creó entonces un prototipo de navegador, WebRunner, que más tarde sería conocido como HotJava.

En 1994, se les hizo una demostración de HotJava y la plataforma Java a los ejecutivos de Sun. Java 1.0a pudo descargarse por primera vez en 1994, pero hubo que esperar al 23 de mayo de 1995, durante las conferencias de SunWorld, a que vieran la luz pública Java y HotJava, el navegador Web. El acontecimiento fue anunciado por John Gage, el Director Científico de Sun Microsystems. El acto estuvo acompañado por una pequeña sorpresa adicional, el anuncio por parte de Marc Andreessen, Vicepresidente Ejecutivo de Netscape, que Java sería soportado en sus navegadores. El 9 de enero del año siguiente, 1996, Sun fundó el grupo empresarial JavaSoft para que se encargase del desarrollo tecnológico. [2] Dos semanas más tarde la primera versión de Java fue publicada.

La promesa inicial de Gosling era Write Once, Run Anywhere (Escríbelo una vez, ejecútalo en cualquier lugar), proporcionando un lenguaje independiente de la plataforma y un entorno de ejecución (la JVM) ligero y gratuito para las plataformas más populares de forma que los binarios (bytecode) de las aplicaciones Java pudiesen ejecutarse en cualquier plataforma.

El entorno de ejecución era relativamente seguro y los principales navegadores web pronto incorporaron la posibilidad de ejecutar applets Java incrustadas en las páginas web.

Java ha experimentado numerosos cambios desde la versión primigenia, JDK 1.0, así como un enorme incremento en el número de clases y paquetes que componen la librería estándar.

5.3 Filosofía

El lenguaje Java se creó con cinco objetivos principales:

1. Debería usar la metodología de la programación orientada a objetos.
2. Debería permitir la ejecución de un mismo programa en múltiples sistemas operativos.
3. Debería incluir por defecto soporte para trabajo en red.
4. Debería diseñarse para ejecutar código en sistemas remotos de forma segura.
5. Debería ser fácil de usar y tomar lo mejor de otros lenguajes orientados a objetos, como C++.

Para conseguir la ejecución de código remoto y el soporte de red, los programadores de Java a veces recurren a extensiones como CORBA (Common Object Request Broker Architecture), Internet Communications Engine o OSGi respectivamente.

5.4 Orientado a objetos

La primera característica, orientado a objetos ("OO"), se refiere a un método de programación y al diseño del lenguaje. Aunque hay muchas interpretaciones para OO, una primera idea es diseñar el software de forma que los distintos tipos de datos que use estén unidos a sus operaciones. Así, los datos y el código (funciones o métodos) se combinan en entidades llamadas objetos. Un objeto puede verse como un paquete que contiene el "comportamiento" (el código) y el "estado" (datos). El principio es separar aquello que cambia de las cosas que permanecen inalterables. Frecuentemente, cambiar una estructura de datos implica un cambio en el código que opera sobre los mismos, o viceversa. Esta separación en objetos coherentes e independientes ofrece una base más estable para el diseño de un sistema software. El objetivo es hacer que grandes proyectos sean fáciles de gestionar y manejar, mejorando como consecuencia su calidad y reduciendo el número de proyectos fallidos. Otra de las grandes promesas de la programación orientada a objetos es la creación de entidades más genéricas (objetos) que permitan la reutilización del software entre proyectos, una de las premisas fundamentales de la Ingeniería del Software. Un objeto genérico "cliente", por ejemplo, debería en teoría tener el mismo conjunto de comportamiento en diferentes proyectos, sobre todo cuando estos coinciden en cierta medida, algo que suele suceder en las

grandes organizaciones. En este sentido, los objetos podrían verse como piezas reutilizables que pueden emplearse en múltiples proyectos distintos, posibilitando así a la industria del software a construir proyectos de envergadura empleando componentes ya existentes y de comprobada calidad; conduciendo esto finalmente a una reducción drástica del tiempo de desarrollo. Podemos usar como ejemplo de objeto el aluminio. Una vez definidos datos (peso, maleabilidad, etc.), y su "comportamiento" (soldar dos piezas, etc.), el objeto "aluminio" puede ser reutilizado en el campo de la construcción, del automóvil, de la aviación, etc.

La reutilización del software ha experimentado resultados dispares, encontrando dos dificultades principales: el diseño de objetos realmente genéricos es pobremente comprendido, y falta una metodología para la amplia comunicación de oportunidades de reutilización. Algunas comunidades de "código abierto" (open source) quieren ayudar en este problema dando medios a los desarrolladores para diseminar la información sobre el uso y versatilidad de objetos reutilizables y librerías de objetos.

5.5 J2ME

La plataforma Java Micro Edition, o anteriormente Java 2 Micro Edition(J2ME), es una especificación de un subconjunto de la plataforma Java orientada a proveer una colección certificada de APIs de desarrolllo de software para dispositivos con recursos restringidos. Está orientado a productos de consumo como PDAs, teléfonos móviles o electrodomésticos.

Java ME se ha convertido en una buena opción para crear juegos en teléfonos móviles debido a que se puede emular en un PC durante la fase de desarrollo y luego subirlos fácilmente al teléfono. Al utilizar tecnologías Java el desarrollo de aplicaciones o videojuegos con estas APIs resulta bastante económico de portar a otros dispositivos

Los principales componentes de la Plataforma Java 2, Edición micro (plataforma J2ME) son Connected Device Configurations, Connected Limited Device Configurations y Mobile Information Device Profiles, así como otras muchas herramientas y tecnologías que llevan las soluciones Java a los mercados de consumo y dispositivos integrados.

Las tecnologías J2ME contienen un JRE altamente optimizado, especialmente desarrollado para el mercado de gran consumo, abarcan una amplia gama de aparatos de tamaño muy reducido y permiten ejecutar programas de seguridad, conectividad y utilidades en tarjetas inteligentes, buscapersonas, sintonizadores de TV y otros pequeños electrodomésticos.

Las tecnologías J2ME representan únicamente una parte de la gama de productos de software de Java. Las plataformas Java relacionadas son la Plataforma Java 2, Edición estándar (plataforma J2SE) y la Plataforma Java 2, Edición empresa (plataforma J2EE). La

tecnología Java ofrece, asimismo, métodos de creación de servicios Web, transferencia de información XML, numerosos protocolos de red, kits de herramientas y la aplicación Java Web Stara.

CAPITULO VI
JAVA vs .NET

6.1 Introducción

En los medios ya inicia el oleaje de noticias acerca .NET (DotNet) la nueva iniciativa de Microsoft, sin embargo, los conceptos y descripciones de este Software son confusos; en lo que coinciden los diversos artículos es que .NET (DotNet) es competencia de Java y hasta esto no es del todo exacto, la competencia de Java es C# también desarrollado por Microsoft, .NET es rival de J2EE para ser más exactos; el siguiente documento esta hecho con la intención de clarificar esta madeja de términos y productos de una manera objetiva, las conclusiones finales las tendrá usted.

6.2 .NET (DotNet) son muchas cosas...

.NET (DotNet) es una "plataforma de Software", en este sentido se define como un ambiente donde pueden interoperar diversos componentes independientemente del lenguaje, esto es, en lugar de escribir componentes para una combinación Hardware/Sistema Operativo será escrito para .NET (DotNet). Ahora bien, .NET (DotNet) es el nombre asignado por Microsoft a diversos productos y servicios, en este caso productos como VisualStudio.NET y Windows.NETServer, mientras los Servicios incluyen Passport y Hailstorm que pretenden ofrecer una manera universal de accesar recursos en Internet.

6.3 Cual es la comparativa Java de .NET (DotNet) ?

En el mundo Java esta "plataforma de Software" es denominada J2EE , sin embargo, a diferencia de .NET (DotNet) que es un *concepto global*, J2EE es un grupo de especificaciones que forman lo que es denominado Java Application Server , la principal ventaja que J2EE este basado en especificaciones es *libertad de elección* sobre vendedores, esto es, los componentes escritos en Java son interoperables entre productos J2EE desarrollados por IBM, HP, Sun, BEA.. a diferencia de .NET (DotNet) donde todo gira alrededor *un solo vendedor*: Microsoft.

6.4 ¿Y C#, el lenguaje nuevo de Microsoft?

Aunque C# es el lenguaje nuevo de Microsoft, no forma parte medular de .NET (DotNet), inclusive la única ventaja de C# es la interfase más directa hacia .NET (DotNet).... interfase ?

6.5 .Net Framework/ MSIL : La verdadera obra de Ingeniería

Este componente es el que aporta la mayor funcionalidad a .NET (DotNet), a través de este Software es posible ejecutar e interoperar diversos fragmentos de código escritos en distintos lenguajes, esto es, si usted trabaja en COBOL, C++ o VisualBasic todos estos lenguajes serán convertidos al lenguaje intermediario *MSIL*("Microsoft Intermediate Language") a través de un compilador determinado , este lenguaje intermediario es el que ejecuta el .Net Framework, de esta manera su código escrito en más de 20 lenguajes podrá interoperar!; aunque esto suene como una verdadera maravilla, las complejidades estriban precisamente en el *mapeo* que debe existir entre los diversos lenguajes y *MSIL*, como bien sabe existen muchos paradigmas que no aplican para todos los lenguajes y esto hace complejo la interoperabilidad prometida por *MSIL*; como se mencionó en el párrafo anterior *C'#* es quien ofrece una *interfase directa* hacia MSIL/.NET (DotNet), es esta interfase/mapeo a la que deben llevarse los distintos lenguajes.

6.6 J2EE o .NET (DotNet) ?

La respuesta: "Web Services" (XMLRPC y SOAP) , porque ? A esta Tecnología le están apostando diversas empresas en la Industria, la razón ? Independencia de sistema, si bien J2EE y .NET (DotNet) ofrecen interoperabilidad de sistemas, XMLRPC/SOAP va un paso más allá debido a que esta basado en XML , (más detalles en XMLRPC y SOAP), inclusive un paso en la evolución tanto de J2EE así como .NET (DotNet) es y seguirá siendo "Web Services" (XMLRPC/SOAP) el cual va llegar a ofrecer un puente entre esta rivalidad (J2EE/.NET DotNet) en un futuro no muy lejano.

CAPITULO VII
LENGUAJE UNIFICADO DE MODELADO (UML)

7.1 Introducción

Lenguaje Unificado de Modelado (**UML**, por sus siglas en inglés, *Unified Modeling Language*) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables.

Es importante resaltar que UML es un "lenguaje" para especificar y no para describir métodos o procesos. Se utiliza para definir un sistema de software, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo. Se puede aplicar en una gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el Proceso Unificado Racional o RUP), pero no especifica en sí mismo qué metodología o proceso usar.

UML no puede compararse con la programación estructurada, pues UML significa (Lengua de Modelación Unificada), no es programación, solo se diagrama la realidad de una utilización en un requerimiento. Mientras que, programación estructurada, es una forma de programar como lo es la orientación a objetos, sin embargo, la orientación a objetos viene siendo un complemento perfecto de UML, pero no por eso se toma UML sólo para lenguajes orientados a objetos.

UML (Unified Modeling Language) es un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema software orientado a objetos. Se ha convertido en el estándar de facto de la industria, debido a que ha sido impulsado por los autores de los tres métodos más usados de orientación a objetos: Grady Booch, Ivar Jacobson y Jim Rumbaugh. Estos autores fueron contratados por la empresa Rational Software Co. para crear una notación unificada en la que basar la construcción de sus herramientas CASE. En el

proceso de creación de UML han participado, no obstante, otras empresas de gran peso en la industria como Microsoft, Hewlett-Packard, Oracle o IBM, así como grupos de analistas y desarrolladores.

Esta notación ha sido ampliamente aceptada debido al prestigio de sus creadores y debido a que incorpora las principales ventajas de cada uno de los métodos particulares en los que se basa (principalmente Booch, OMT y OOSE). UML ha puesto fin a las llamadas "guerras de métodos" que se han mantenido a lo largo de los 90, en las que los principales métodos sacaban nuevas versiones que incorporaban las técnicas de los demás. Con UML se fusiona la notación de estas técnicas para formar una herramienta compartida entre todos los ingenieros software que trabajan en el desarrollo orientado a objetos.

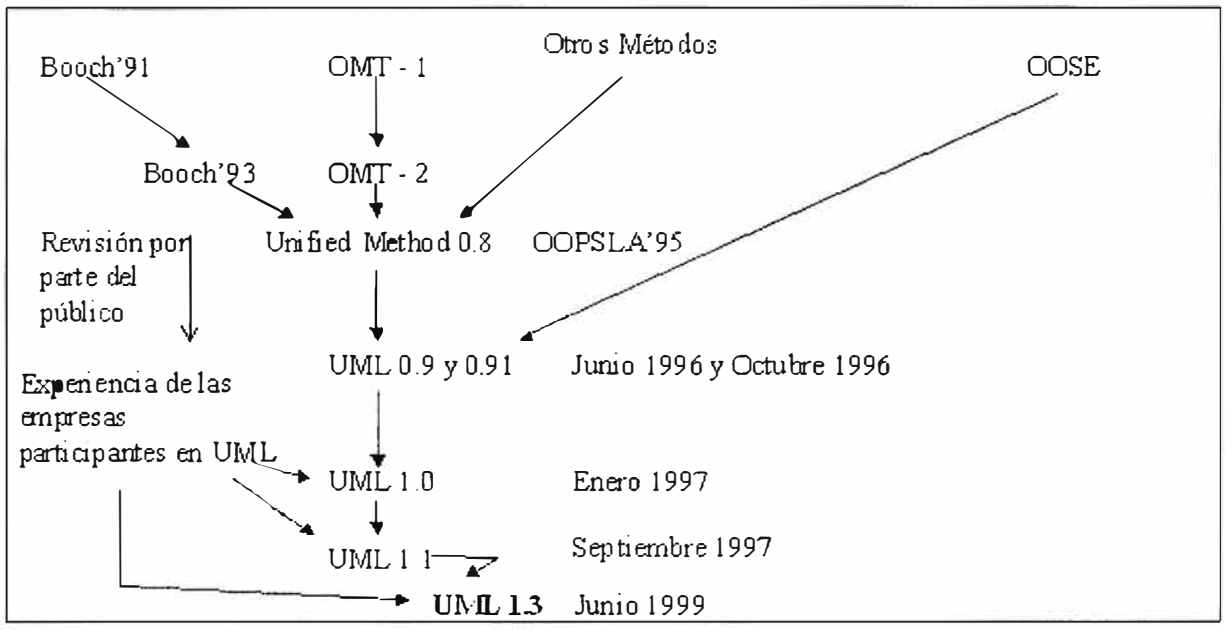


Fig. 7.1 Evolución de UML

Uno de los objetivos principales de la creación de UML era posibilitar el intercambio de modelos entre las distintas herramientas CASE orientadas a objetos del mercado. Para ello era necesario definir una notación y semántica común. En la Fig. 2 se puede ver cuál ha sido la evolución de UML hasta la creación de UML 1.3, en el que se basa este documento. Hay que tener en cuenta que el estándar UML no define un proceso de desarrollo específico, tan solo se trata de una notación. En este curso se sigue el método propuesto por Craig Larman [Larman99] que se ajusta a un ciclo de vida iterativo e incremental dirigido por casos de uso.

En la parte II de este texto se expone la notación y semántica básica de UML, en la parte III se presentan conceptos avanzados de la notación UML, mientras que en la parte IV se

presenta el método de desarrollo orientado a objetos de Larman, que se sirve de los modelos de UML que se han visto anteriormente.

7.2 Modelos

Un modelo representa a un sistema software desde una perspectiva específica. Al igual que la planta y el alzado de una figura en dibujo técnico nos muestran la misma figura vista desde distintos ángulos, cada modelo nos permite fijarnos en un aspecto distinto del sistema.

Los modelos de UML que se tratan en esta parte son los siguientes:

- Diagrama de Estructura Estática.
- Diagrama de Casos de Uso.
- Diagrama de Secuencia.
- Diagrama de Colaboración.
- Diagrama de Estados.

7.3 Elementos comunes a todos los diagramas

7.3.1 Notas

Una nota sirve para añadir cualquier tipo de comentario a un diagrama o a un elemento de un diagrama. Es un modo de indicar información en un formato libre, cuando la notación del diagrama en cuestión no nos permite expresar dicha información de manera adecuada. Una nota se representa como un rectángulo con una esquina doblada con texto en su interior. Puede aparecer en un diagrama tanto solo como unido a un elemento por medio de una línea discontinua. Puede contener restricciones, comentarios, el cuerpo de un procedimiento, etc.

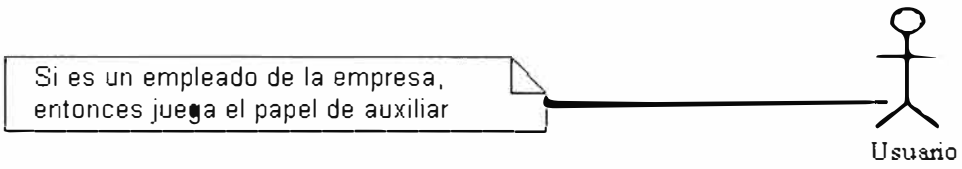


Fig. 7.2 Nota

7.3.2 Dependencias

La relación de dependencia entre dos elementos de un diagrama significa que un cambio en el elemento destino puede implicar un cambio en el elemento origen (por tanto, si cambia el elemento destino habría que revisar el elemento origen). Una dependencia se representa por medio de una línea de trazo discontinuo entre los dos elementos con una flecha en su extremo. El elemento dependiente es el origen de la flecha y el elemento del que depende es el destino (junto a él está la flecha).

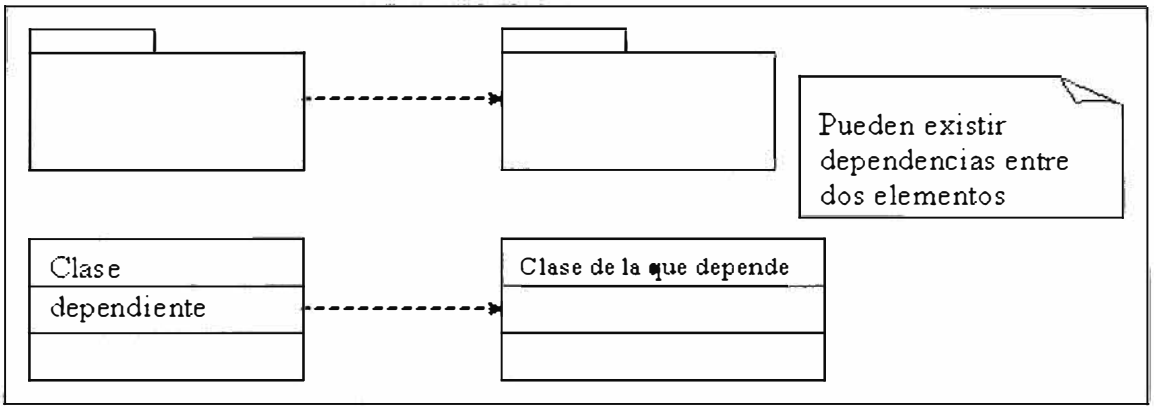


Fig. 7.3 Dependencias

7.4 Diagramas de estructura estática

Los Diagramas de Estructura Estática de UML se van a utilizar para representar tanto Modelos Conceptuales como Diagramas de Clases de Diseño. Ambos usos son distintos conceptualmente, mientras los primeros modelan elementos del dominio los segundos presentan los elementos de la solución software. Ambos tipos de diagramas comparten una parte de la notación para los elementos que los forman (clases y objetos) y las relaciones que existen entre los mismos (asociaciones). Sin embargo, hay otros elementos de notación que serán exclusivos de uno u otro tipo de diagrama.

7.4.1 Clases

Una clase se representa mediante una caja subdividida en tres partes: En la superior se muestra el nombre de la clase, en la media los atributos y en la inferior las operaciones. Una clase puede representarse de forma esquemática, con los atributos y operaciones suprimidos, siendo entonces tan solo un rectángulo con el nombre de la clase. En la Fig. 7.4 se ve cómo una misma clase puede representarse a distinto nivel de detalle según interese, y según la fase en la que se esté.

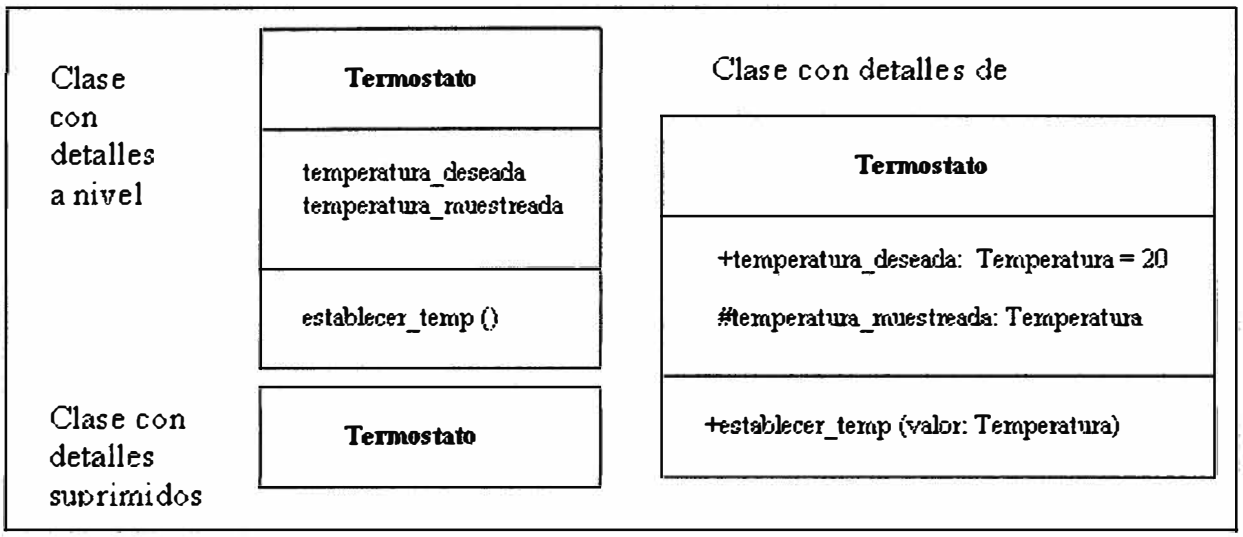


Fig. 7.4 Clases

7.4.2 Objetos

Un objeto se representa de la misma forma que una clase. En el compartimento superior aparecen el nombre del objeto junto con el nombre de la clase subrayados, según la siguiente sintaxis: nombre_del_objeto: nombre_de_la_clase Puede representarse un objeto sin un nombre específico, entonces sólo aparece el nombre de la clase.

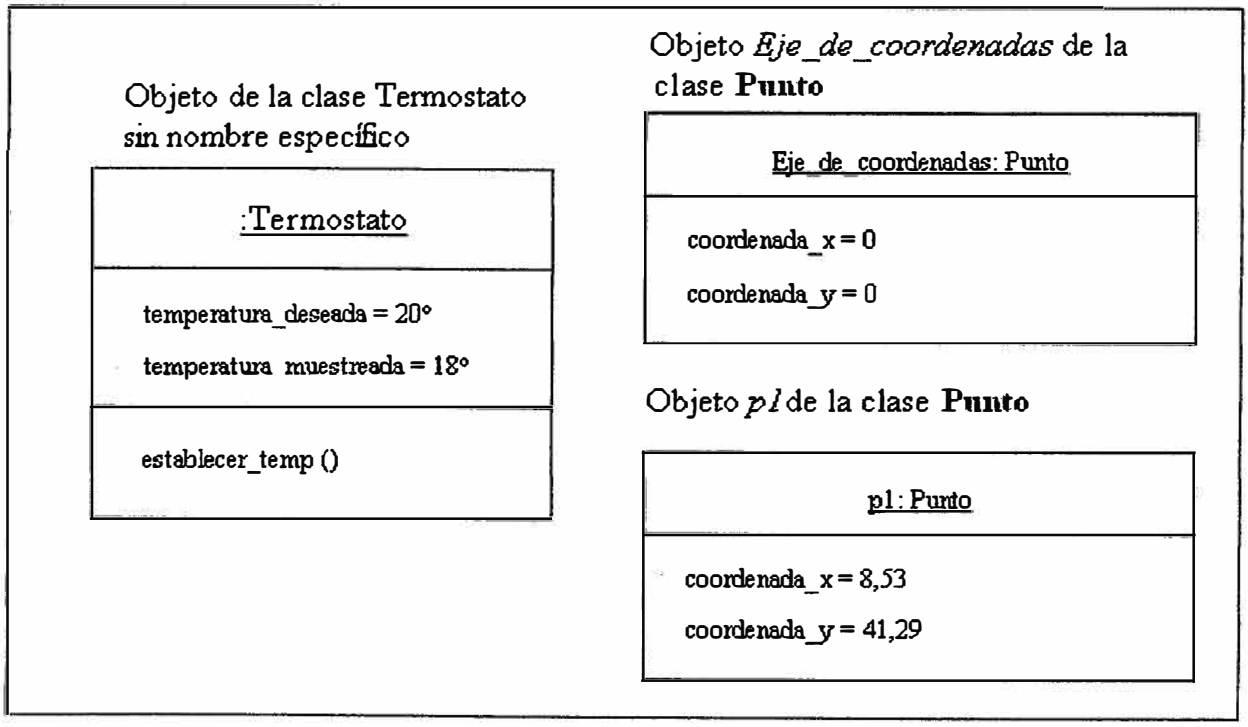


Fig. 7.5 Objetos

7.4.3 Asociaciones

Las asociaciones entre dos clases se representan mediante una línea que las une. La línea puede tener una serie de elementos gráficos que expresan características particulares de la asociación. A continuación se verán los más importantes de entre dichos elementos gráficos.

a) Nombre de la Asociación y Dirección

El nombre de la asociación es opcional y se muestra como un texto que está próximo a la línea. Se puede añadir un pequeño triángulo negro sólido que indique la dirección en la cual leer el nombre de la asociación. En el ejemplo de la Fig. 7.6 se puede leer la asociación como "Director manda sobre Empleado".

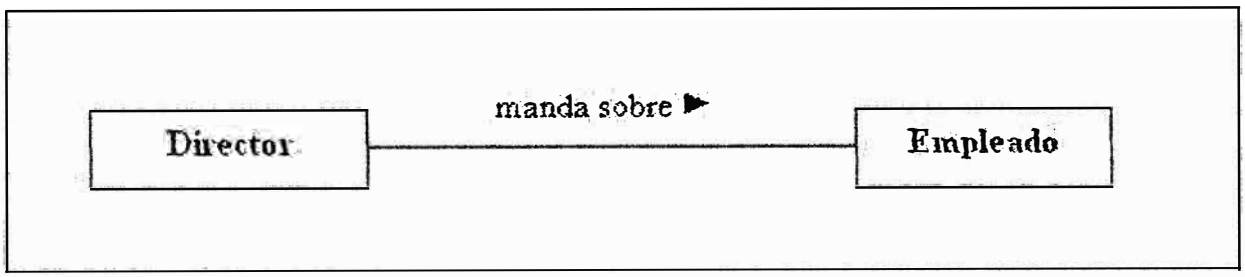


Fig. 7.6 Asociación

Los nombres de las asociaciones normalmente se incluyen en los modelos para aumentar la legibilidad. Sin embargo, en ocasiones pueden hacer demasiado abundante la información que se presenta, con el consiguiente riesgo de saturación. En ese caso se puede suprimir el nombre de las asociaciones consideradas como suficientemente conocidas. En las asociaciones de tipo agregación y de herencia no se suele poner el nombre.

b) Multiplicidad

La multiplicidad es una restricción que se pone a una asociación, que limita el número de instancias de una clase que pueden tener esa asociación con una instancia de la otra clase. Puede expresarse de las siguientes formas:

- Con un número fijo: 1.
- Con un intervalo de valores: 2..5.
- Con un rango en el cual uno de los extremos es un asterisco. Significa que es un intervalo abierto. Por ejemplo, 2..* significa 2 o más.
- Con una combinación de elementos como los anteriores separados por comas: 1, 3..5, 7, 15..*.

• Con un asterisco: * . En este caso indica que puede tomar cualquier valor (cero o más).

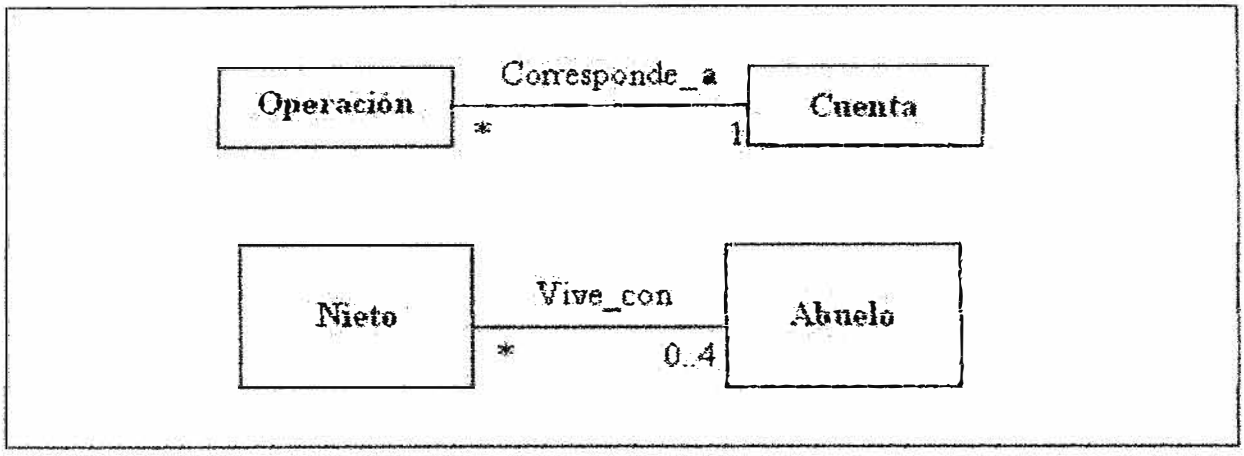


Fig. 7.7 Multiplicidad

c) Roles

Para indicar el papel que juega una clase en una asociación se puede especificar un nombre de rol.

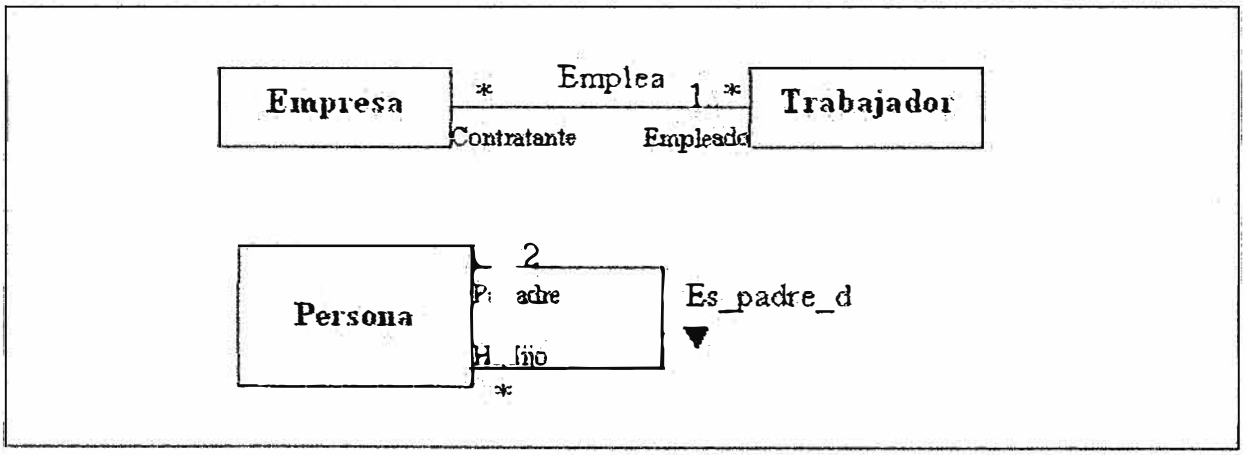


Fig. 7.8 Roles

Se representa en el extremo de la asociación junto a la clase que desempeña dicho rol.

d) Agregación

El símbolo de agregación es un diamante colocado en el extremo en el que está la clase que representa el "todo".

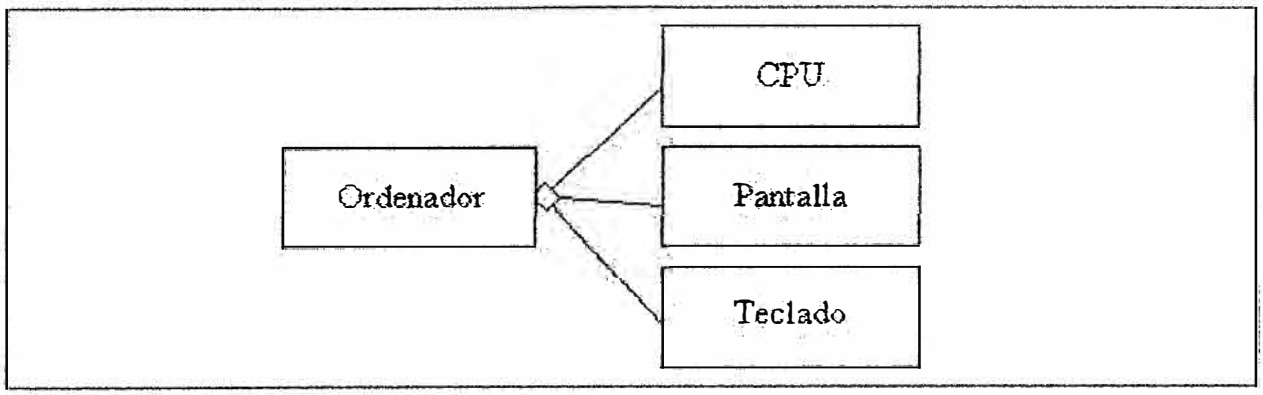


Fig. 7.9 Agregación

e) Clases Asociación

Cuando una asociación tiene propiedades propias se representa como una clase unida a la línea de la asociación por medio de una línea a trazos. Tanto la línea como el rectángulo de clase representan el mismo elemento conceptual: la asociación. Por tanto ambos tienen el mismo nombre, el de la asociación. Cuando la clase asociación sólo tiene atributos el nombre suele ponerse sobre la línea (como ocurre en el ejemplo de la Fig. 7.10). Por el contrario, cuando la clase asociación tiene alguna operación o asociación propia, entonces se pone el nombre en la clase asociación y se puede quitar de la línea.

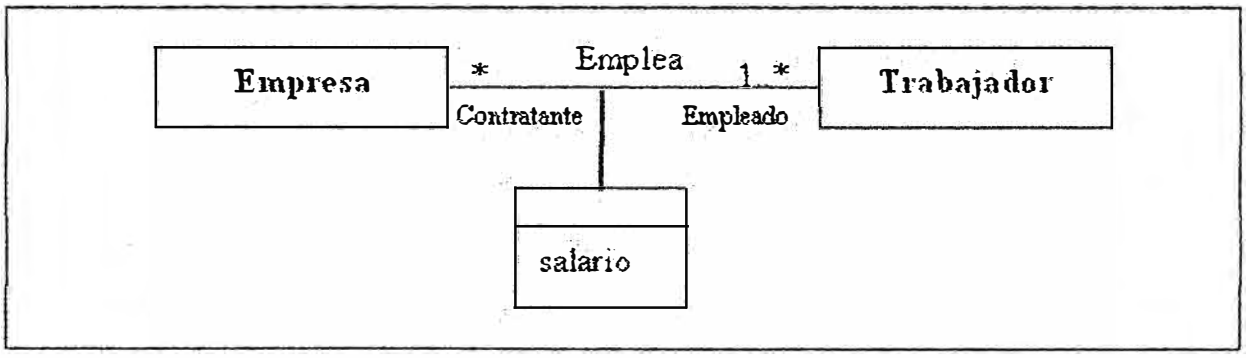


Fig. 7.10 Clases Asociación

f) Asociaciones N-Arias

En el caso de una asociación en la que participan más de dos clases, las clases se unen con una línea a un diamante central. Si se muestra multiplicidad en un rol, representa el número potencial de tuplas de instancias en la asociación cuando el resto de los N-1 valores están fijos. En la Fig. 7.11 se ha impuesto la restricción de que un jugador no puede jugar en dos equipos distintos a lo largo de una temporada, porque la multiplicidad de "Equipo" es 1 en la asociación ternaria.

g) Navegabilidad

En un extremo de una asociación se puede indicar la navegabilidad mediante una flecha. Significa que es posible "navegar" desde el objeto de la clase origen hasta el objeto de la clase destino. Se trata de un concepto de diseño, que indica que un objeto de la clase origen conoce al (los) objeto(s) de la clase destino, y por tanto puede llamar a alguna de sus operaciones.

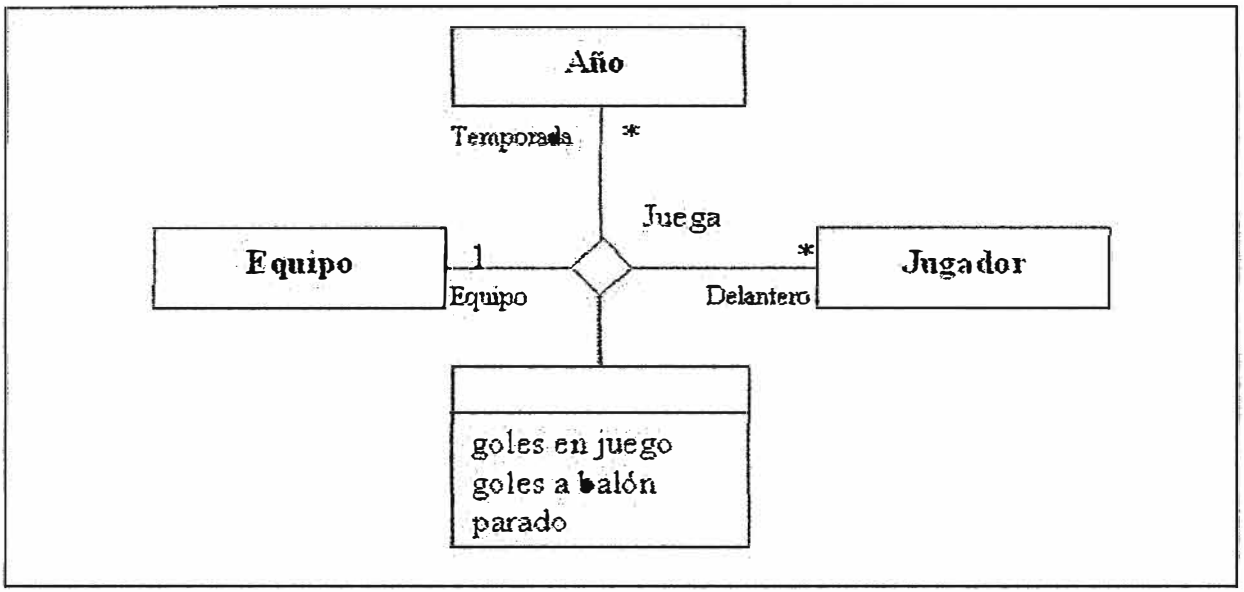


Fig. 7.11 Asociaciones N-Arias

7.4.4 Herencia

La relación de herencia se representa mediante un triángulo en el extremo de la relación que corresponde a la clase más general o clase "padre".

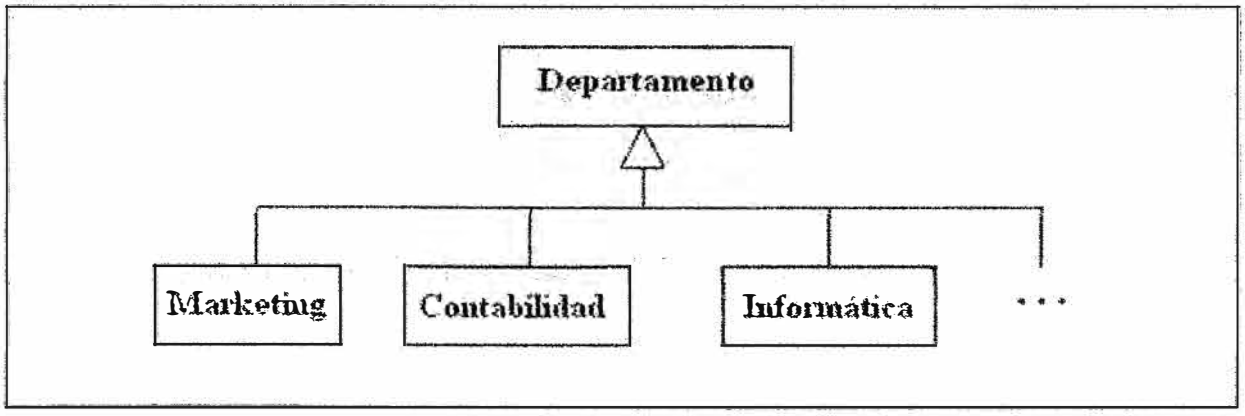


Fig. 7.12 Herencia

Si se tiene una relación de herencia con varias clases subordinadas, pero en un diagrama concreto no se quieren poner todas, esto se representa mediante puntos suspensivos. En el ejemplo de la Fig. 7.12, sólo aparecen en el diagrama 3 tipos de departamentos, pero con los puntos suspensivos se indica que en el modelo completo (el formado por todos los diagramas) la clase "Departamento" tiene subclases adicionales, como podrían ser "Recursos Humanos" y "Producción".

7.4.5 Elementos Derivados

Un elemento derivado es aquel cuyo valor se puede calcular a partir de otros elementos presentes en el modelo, pero que se incluye en el modelo por motivos de claridad o como decisión de diseño. Se representa con una barra "/" precediendo al nombre del elemento derivado.

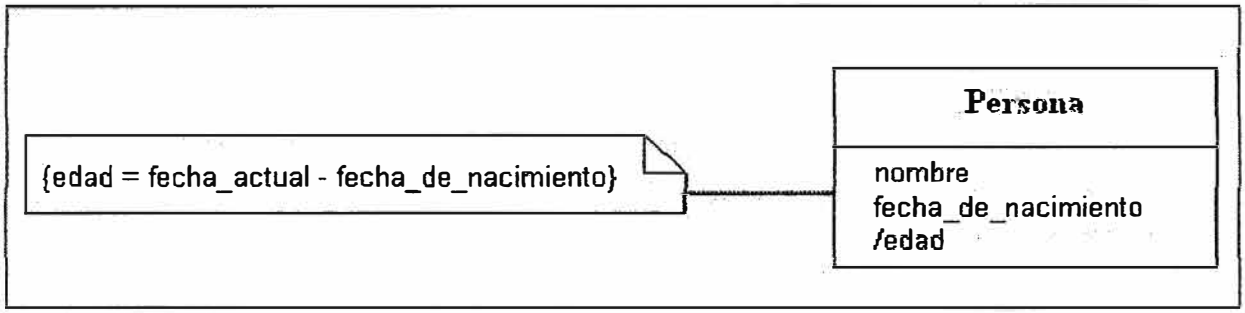


Fig. 7.13 Elementos Derivados

7.5 Diagrama de Casos de Uso

Un Diagrama de Casos de Uso muestra la relación entre los actores y los casos de uso del sistema. Representa la funcionalidad que ofrece el sistema en lo que se refiere a su interacción externa. En el diagrama de casos de uso se representa también el sistema como una caja rectangular con el nombre en su interior. Los casos de uso están en el interior de la caja del sistema, y los actores fuera, y cada actor está unido a los casos de uso en los que participa mediante una línea. En la Fig. 7.14 se muestra un ejemplo de Diagrama de Casos de Uso para un cajero automático.

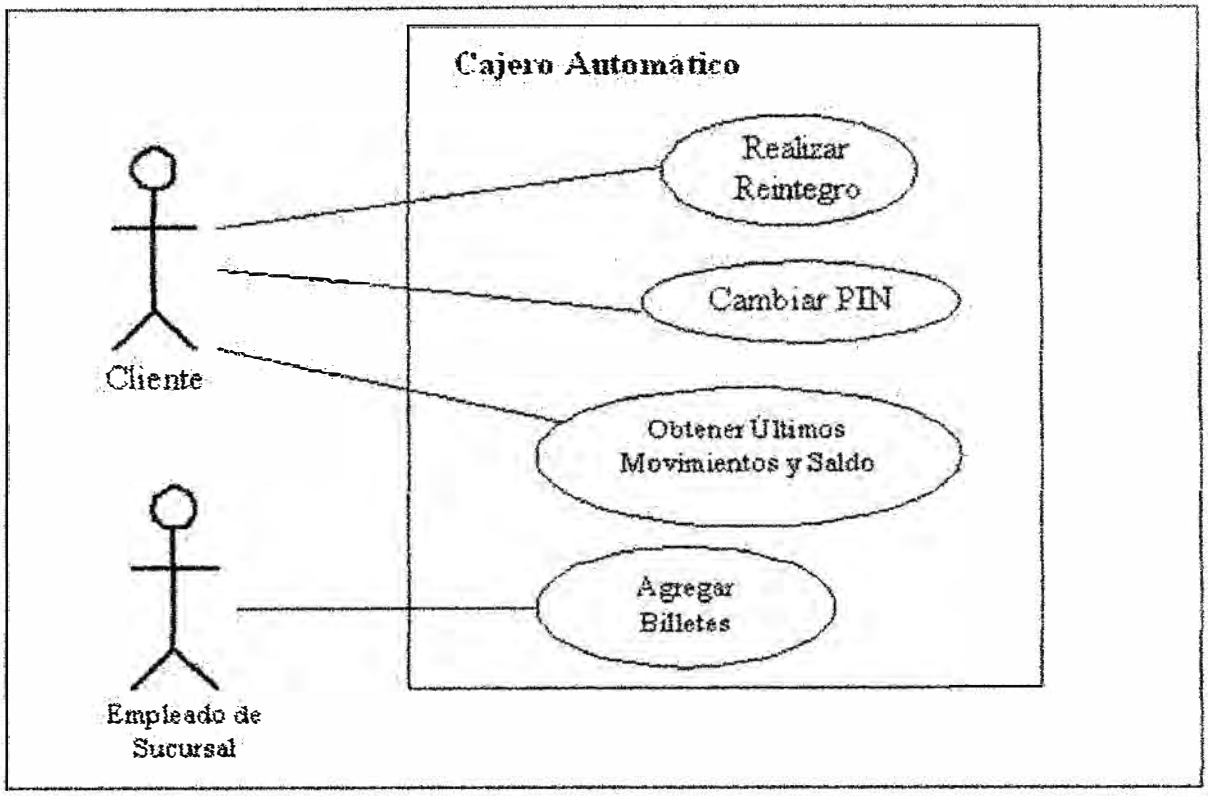


Fig. 7.14 Diagrama de Casos de uso

7.5.1 Elementos

Los elementos que pueden aparecer en un Diagrama de Casos de Uso son: actores, casos de uso y relaciones entre casos de uso.

a) Actores

Un actor es algo con comportamiento, como una persona (identificada por un rol), un sistema informatizado u organización, y que realiza algún tipo de interacción con el sistema.. Se representa mediante una figura humana dibujada con palotes. Esta representación sirve tanto para actores que son personas como para otro tipo de actores.

b) Casos de Uso

Un caso de uso es una descripción de la secuencia de interacciones que se producen entre un actor y el sistema, cuando el actor usa el sistema para llevar a cabo una tarea específica. Expresa una unidad coherente de funcionalidad, y se representa en el Diagrama de Casos de Uso mediante una elipse con el nombre del caso de uso en su interior. El nombre del caso de uso debe reflejar la tarea específica que el actor desea llevar a cabo usando el sistema.

c) Relaciones entre Casos de Uso

Un caso de uso, en principio, debería describir una tarea que tiene un sentido completo para el usuario. Sin embargo, hay ocasiones en las que es útil describir una interacción con un alcance menor como caso de uso. La razón para utilizar estos casos de uso no completos en

algunos casos, es mejorar la comunicación en el equipo de desarrollo, el manejo de la documentación de casos de uso. Para el caso de que queramos utilizar estos casos de uso más pequeños, las relaciones entre estos y los casos de uso ordinarios pueden ser de los siguientes tres tipos: • Incluye (<>): Un caso de uso base incorpora explícitamente a otro caso de uso en un lugar especificado en dicho caso base. Se suele utilizar para encapsular un comportamiento parcial común a varios casos de uso. En la Fig. 7.15 se muestra cómo el caso de uso Realizar Reintegro puede incluir el comportamiento del caso de uso Autorización.

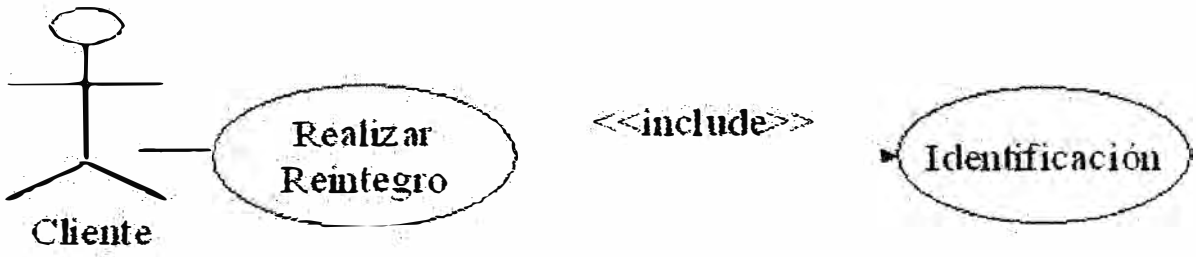


Fig. 7.15 Relaciones entre Casos de Uso

Fig. 7.15 - Ejemplo de Relación <> • Extiende (<>): Cuando un caso de uso base tiene ciertos puntos (puntos de extensión) en los cuales, dependiendo de ciertos criterios, se va a realizar una interacción adicional. El caso de uso que extiende describe un comportamiento opcional del sistema (a diferencia de la relación incluye que se da siempre que se realiza la interacción descrita) En la Fig. 7.16 se muestra como el caso de uso Comprar Producto permite explícitamente extensiones en el siguiente punto de extensión: info regalo. La interacción correspondiente a establecer los detalles sobre un producto que se envía como regalo están descritos en el caso de uso Detalles Regalo.

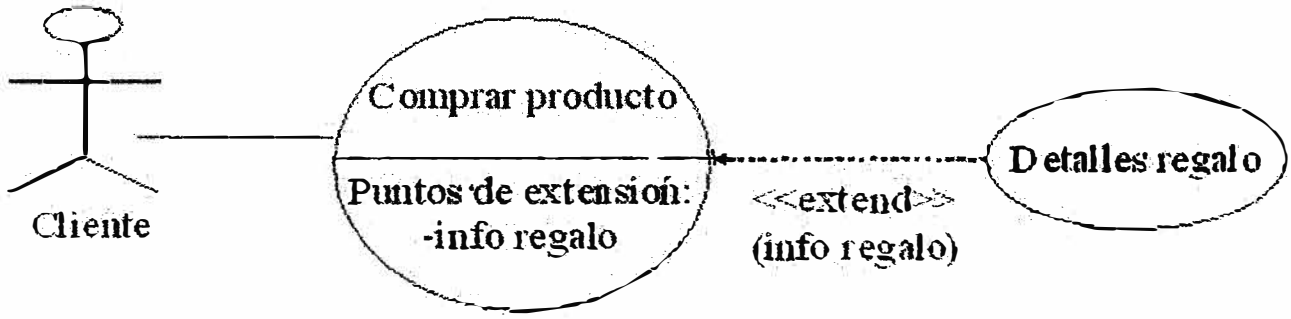


Fig. 7.16 Ejemplo de Relación <>

Ambos tipos de relación se representan como una dependencia etiquetada con el estereotipo correspondiente (<> o <>), de tal forma que la flecha indique el sentido en el que debe leerse la etiqueta. Junto a la etiqueta <> se puede detallar el/los puntos de extensión del

caso de uso base en los que se aplica la extensión. • Generalización ( ): Cuando un caso de uso definido de forma abstracta se particulariza por medio de otro caso de uso más específico. Se representa por una línea continua entre los dos casos de uso, con el triángulo que simboliza generalización en UML (usado también para denotar la herencia entre clases) pegado al extremo del caso de uso más general. Al igual que en la herencia entre clases, el caso de uso hijo hereda las asociaciones y características del caso de uso padre. El caso de uso padre se trata de un caso de uso abstracto, que no está definido completamente. Este tipo de relación se utiliza mucho menos que las dos anteriores.

CONCLUSIONES

Hoy en día los dispositivos móviles son parte de nuestro estilo de vida y la mayoría de nosotros no pueden vivir sin ellas. Cuando estos dispositivos móviles están conectados a Internet, las competencias de los dispositivos móviles son infinitas. Podemos suministrar oportunamente los datos al usuario en cualquier lugar y cualquier momento.

Los teléfonos celulares, PDA y otros dispositivos inalámbricos que conectan con Internet gozan de creciente popularidad, haciendo aplicaciones inalámbricas más importante y especialmente útil para empresas con empleados remotos.

.NET Mobile generará código WML para teléfonos celulares con WAP habilitado y código HTML para dispositivos como la Pocket PC.

Detectando el browser, .NET Mobile producirá el contenido correcto, proveyendo a los desarrolladores una poderosa herramienta para desarrollar aplicaciones únicas que servirán para diferentes dispositivos móviles.

BIBLIOGRAFÍA

1. http://www.digitalforum.accenture.com/DigitalForum/Argentina/CurrentEdition/HotTechnologies/La+evoluci%C3%B3n+de+las+aplicaciones+M%C3%B3viles.htm
2. http://www.monografias.com/trabajos30/cableado/cableado.shtml
3. http://msdn2.microsoft.com/en-us/magazine/cc301785.aspx
4. http://www.c-sharpcorner.com/UploadFile/ggaganesh/DvelopingMobileWebApplicationwith.NET11182005013704AM/DvelopingMobileWebApplicationwith.NET.aspx
5. http://www.wirelessdevnet.com/channels/wap/features/mobilesdk.html
6. http://www.codeproject.com/KB/dotnet/MobileWebAppSecret.aspx
7. http://msdn.microsoft.com/en-us/library/aa454892.aspx
8. http://www.w3schools.com/dotnetmobile/mobile_intro.asp
9. http://es.wikipedia.org/wiki/.NET_de_Microsoft
10. http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_Java
11. http://www.osmosislatina.com/java/dotnetj2ee.htm
12. http://www.clikear.com/manuales/uml/introduccion.aspx
13. http://www.webmovilgsm.com/wap.htm
14. http://www.mygnet.net/articulos/wap/130/
15. http://www.richardcrebeck.com/cclca