UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES
DE ARAGÓN

"SISTEMA DE VIDEO-VIGILANCIA MULTIPLATAFORMA CON INTEGRACIÓN A DISPOSITIVOS MÓVILES"

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

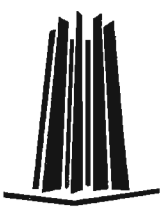
INGENIERO EN COMPUTACIÓN

P R E S E N T A :

DAVID AURELIO ACEVEDO MORENO

A S E S O R :

ING. ALEJANDRO RENÉ GONZÁLEZ PONCE



MÉXICO, 2005.

Agradecimientos.

A mi madre:
Por el sacrificio y dedicación que has puesto en tu vida para que tus hijos tengan una educación y todas las oportunidades para salir adelante. Pero mas que nada, por el amor que siempre nos has dado, a mi hermana y a mi, porque sin el, no seriamos las personas que somos ahora.

A mi hermanita:
Por estar siempre a mi lado, tu ejemplo, cariño y ayuda han sido muy valiosos, para buscar mi propia superación.

A Gaby, Sandy y Alf:
Por brindarme su increíble amistad durante todo este tiempo, por compartir conmigo sus vidas y tantos momentos inolvidables, por aguantar mi mal humor, por su apoyo en momentos difíciles y porque los 4 Fantásticos no tendrían sentido sin ustedes.

A Alejandro y Liliana
Porque ustedes no solo fueron mis maestros, sino también mis amigos y me han dado todo su apoyo y confianza, para obtener mi título y superarme día con día como ingeniero.

Y a todos aquellos amigos que han sido parte de mi vida, porque por cada uno de ustedes, siempre hay algo bueno que contar.

INDICE

Capítulo 1. Descripción de Sistemas de Vigilancia Actuales, Dispositivos y aplicaciones móviles 1

1.1 Sistemas de Vigilancia 1

1.2 Elementos que componen un sistema de CCTV 3
1.2.1 Elementos captores de imagen 3
1.2.2 Elementos reproductores de imagen 3
1.2.3 Elementos grabadores de imagen 3
1.2.4 Elementos de transmisión de la señal de video 4
1.2.5 Elementos de control 4

1.3 Sistemas CCTV basados en PC 4

1.4 Cámaras de Red 7

1.5 Dispositivos y aplicaciones móviles 8
1.5.1 Dispositivos móviles 8
1.5.2 Tecnología inalámbrica 9
1.5.3 Aplicaciones móviles 11

Capítulo 2. Herramientas de Desarrollo 12

2.1 La Tecnología Java 12
2.1.1 El Lenguaje de Programación Java 12
2.1.2 La Plataforma Java 15

2.2 Java 2 Platform, Standard Edition 16

2.3 Java 2 Platform, Enterprise Edition 19
2.3.1 Aplicaciones Distribuidas Multinivel 19
2.3.2 Contenedores J2EE 20
2.3.3 Web Services 21
2.3.4 J2EE 1.4 APIs 21

2.4 Java 2 Platform, Micro Edition 24
2.4.1 Configuraciones 25
2.4.2 Perfiles 26
2.4.3 Paquetes Opcionales 28
2.4.4 Java Technology for the Wireless Industry 30

2.5 Herramientas de Programación 31
2.5.1 Ambiente Integrado de Desarrollo 31
2.5.2 Netbeans IDE 33

Capítulo 3. Gestión de Video 34

3.1 Time-Based Media 34
3.1.1 Flujo de Datos 34
3.1.2 Presentación de los Datos 36
3.1.3 Procesamiento de Datos 37
3.1.4 Captura o Media Capture 38

3.2 Java Media Framework 39
3.2.1 Arquitectura 39
3.2.2 Presentación 44
3.2.3 Procesado 45
3.2.4 Captura 47
3.2.5 Almacenamiento y Transmisión de Datos 47

Capítulo 4. Desarrollo de Aplicaciones Móviles 49

4.1 MIDlets Suite 49

4.2 Modelo de Seguridad 50

4.3 Instalación 51

4.4 Desinstalación 52

4.5 Ciclo de Vida 52

4.6 Aplicación ¡Hola Mundo! 53

Capítulo 5. Análisis y Desarrollo de un Caso Práctico 58

5.1 Descripción del Problema 58

5.2 Elementos de un Sistema Multiplataforma 58
5.2.1 Hardware 58
5.2.2 Software 59

5.3 Configuración Física del Sistema 60

5.4 Funcionamiento del Sistema 61
5.4.1 Nets Desktop 62
5.4.2 Nets Server 66
5.4.3 MicroNets 67

5.5 Sistemas Similares 72

CAPITULO 1

DESCRIPCION DE SISTEMAS DE VIGILANCIA ACTUALES. DISPOSITIVOS Y APLICACIONES MOVILES.

1.1. Sistemas de Vigilancia (CCTV)

Un Circuito Cerrado de Televisión (CCTV) es un sistema formado por un conjunto de cámaras y equipo especializado dedicados a tareas de vigilancia por medio del video. Aunque en la mayoría de los casos sus componentes se encuentran enlazados a través de cables también es posible encontrar sistemas inalámbricos que funcionan de forma muy similar a un sistema de emisión de televisión.

Los sistemas de CCTV donde la imagen es vista o grabada, pero no emitida fue inicialmente desarrollada por motivos de seguridad en los bancos. Actualmente su uso se ha difundido tanto que podemos encontrar éstos sistemas en muchas partes diferentes, por ejemplo los casinos, donde son utilizados para vigilar cada uno de los juegos en busca de personas haciendo alguna clase de trampa o empleados beneficiando a ciertos jugadores. En otro caso, los centros comerciales cuentan con un área bastante grande donde la vigilancia se hace especialmente difícil por lo cual utilizan las cámaras para que con menos personal puedan vigilar la mayor área posible y ubicar cualquier acto sospechoso de manera más eficiente. Algunos aeropuertos han implantado sistemas de Circuito Cerrado de Televisión como una medida antiterrorista. Por otro lado los gobiernos de diferentes países utilizan esta tecnología en las calles, tanto para monitorear el tráfico vehicular como para vigilancia de zonas con alto nivel delictivo.

Conforme avanza la tecnología estos sistemas se han hecho más simples, eficientes y menos costosos permitiendo su uso a pequeñas empresas para la supervisión de empleados o incluso su uso domestico.

Los Circuitos Cerrados de Televisión se ha extendido a lo largo de las ciudades, ya se manejados por el gobierno o por civiles, teniendo como justificación que detienen el crimen. Sin embargo, hay quien se opone a su uso argumentando que violan la privacidad de las personas bajo vigilancia pero aun más importante, aseguran que desplaza el crimen más que reducirlo.

Aunque el uso más difundido de estos sistemas es el de la vigilancia, existen otras aplicaciones como es la investigación o el cuidado infantil. Supongamos que se esta realizando una investigación sobre el comportamiento de cierta especie en peligro de extinción, sin embargo, la presencia humana interfiere en su comportamiento natural, la solución perfecta es un monitoreo a distancia, el cual puede ser fácilmente implementado por un sistema CCTV que además

ayudará a los investigadores a tener un gran archivo de video para su posterior análisis.
Las primeras cámaras de CCTV usadas en áreas públicas eran muy rudimentarias, eran cámaras a blanco y negro de baja resolución sin la habilidad para hacer acercamientos (zoom) o moverse de un lado a otro (pan/tilt). Las cámaras modernas son a color, de alta definición y no solo pueden realizar acercamientos, sino que además, enlazadas a una computadora principal pueden seguir el movimiento de los objetos a lo largo de una escena.

Actualmente los Circuitos Cerrados de Televisión utilizan técnicas para el análisis de las imágenes digitales con lo cual pueden rastrear las placas de los autos como parte de sistemas más complejos para la localización de autos robados, críticos de estos sistemas señalan que esto produce una potencial fuente de información para la localización de personas o grupos, pues tecnológicamente, no existe limitación para que un conjunto de cámaras conectadas en red puedan seguir el movimiento de individuos.

Una de las tecnologías más controvertidas en el uso de cámaras de CCTV es el reconocimiento facial, con la cual sería posible determinar la identidad de cualquier persona sin siquiera detenerla o sin que se percate de que su identidad esta siendo verificada o almacenada. Los sistemas podrían identificar miles de rostros almacenados en bases de datos en unos segundos. Para lugares donde el tráfico de personas se cuenta por miles y es imprescindible una rápida atención a todas ellas, como en aeropuertos o estadios, se convierte en una herramienta de mucha utilidad sin embargo el bajo poder de discriminación y un alto número de falsos positivos (Un falso positivo, también llamado falsa alarma, existe cuando el sistema encuentra, incorrectamente una señal cuando en realidad no existe ninguna) hacen que esta tecnología sea aún ineficiente.

Otra variante del uso de estas tecnologías permite que los operadores no tengan que estar interminablemente observando los monitores, lo cual también significa que una sola persona puede operar más de un CCTV a la vez. Estos sistemas no observan a las personas directamente, en su lugar rastrean su comportamiento en busca de tipos particulares de movimientos, o tipos específicos de ropa o equipaje. La teoría detrás de esto dice que en lugares públicos las personas tienden a comportarse de maneras predecibles. Ejemplo de esto puede presentarse en una calle poco transitada, donde un ladrón de autos no se comportará de la misma manera que otras personas, lo cual la computadora podría identificar sus movimientos y alertar al operador de que algo fuera de lo normal esta sucediendo, sin embargo, una persona esperando encontrarse con otra potencialmente provocaría una alerta en el sistema.

1.2. Elementos que componen un sistema de CCTV[1]

1.2.1. Elementos captores de imagen.

Están constituidos por cámaras de video y los accesorios que las complementan como son los objetivos, carcasas de protección, soportes, etc. Actualmente las cámaras utilizan pequeños componentes electrónicos llamados CCD (Charge-Coupled Device), estos elementos son los encargados de recoger la luz y convertirla en una señal eléctrica. Los CCD se encuentran organizados en arreglos donde cada elemento representa un píxel de la imagen capturada, por lo tanto, mientras más grandes sean estos arreglos, mayor será la resolución de la imagen.

1.2.2. Elementos reproductores de imagen

Los elementos que nos permiten la visualización de las imágenes captadas por las cámaras son los monitores, los cuales son muy similares a un televisor doméstico, aunque generalmente carece de circuitos de radiofrecuencia, contiene un selector de impedancia para la señal de entrada. Comúnmente el tamaño de la pantalla suele ser de 9 pulgadas en monitores blanco y negro, hasta 14 pulgadas para monitores a color, sin embargo dependiendo las necesidades del sistema es posible utilizar monitores de mayor tamaño.

1.2.3. Elementos grabadores de imagen

Una de las características más importantes de un sistema de CCTV es su capacidad para grabar las imágenes producidas por las cámaras, esto es logrado comúnmente por dispositivos que utilizan cintas magnéticas, el ejemplo más claro lo encontramos con una videograbadora casera que puede permitir hasta unas 8 horas de grabación continua, sin embargo, este lapso de tiempo suele ser insuficiente para un sistema de vigilancia que opera las 24 horas del día. Existen equipos especializados que permiten la grabación de hasta 960 horas. Una manera más eficiente de grabación se realiza al digitalizar las imágenes y almacenarlas en discos duros, lo cual dependiendo del sistema no solo podría incrementar la capacidad de almacenamiento, sino que además permitiría una mayor facilidad en su administración.

[1] Ramon F. Mateo. (n.d.). Diseño de un sistema de TV de Circuito Cerrado CCTV. Monografias.com http://www.monografias.com/trabajos/cctelevis/cctelevis.shtml

1.2.4. Elementos de transmisión de la señal de video

La señal de video obtenida de la cámara debe ser transmitida en las mejores condiciones posibles al monitor o monitores correspondientes, para lo cual se emplean las líneas de transmisión y amplificadores de señal en caso de que sea necesario para recorrer distancias más grandes. La transmisión por cable coaxial es una de las más comunes, sin embargo, no es la única pues dependiendo de las necesidades del sistema también es posible encontrar transmisiones a través de líneas telefónicas, radiofrecuencias o sobre redes de datos. Aunque se debe tener en cuenta que para ello es necesario dispositivos tales como conversores, transductores, módems o conjuntos emisor/receptor, adecuados a cada caso.

1.2.5. Elementos de control

Aunque en el caso más simple estos elementos no son necesarios, son imprescindibles para sistemas más completos donde se hace necesario un selector de video para manipular la señal proveniente de múltiples cámaras y mostrar sus imágenes en uno o varios monitores. Estos elementos permiten manipular la imagen mostrada de tal forma que en un monitor se puede mostrar secuencialmente la imagen de cada una de las cámaras disponibles. En otros casos, tal vez sea necesario mantener una visualización constante de las escenas minimizando el número de monitores a utilizar, esto es posible mediante un equipo que reciba diferentes señales enviando una única señal al monitor en el cual la pantalla es dividida en sectores donde en cada sector se presenta una de las señales originales

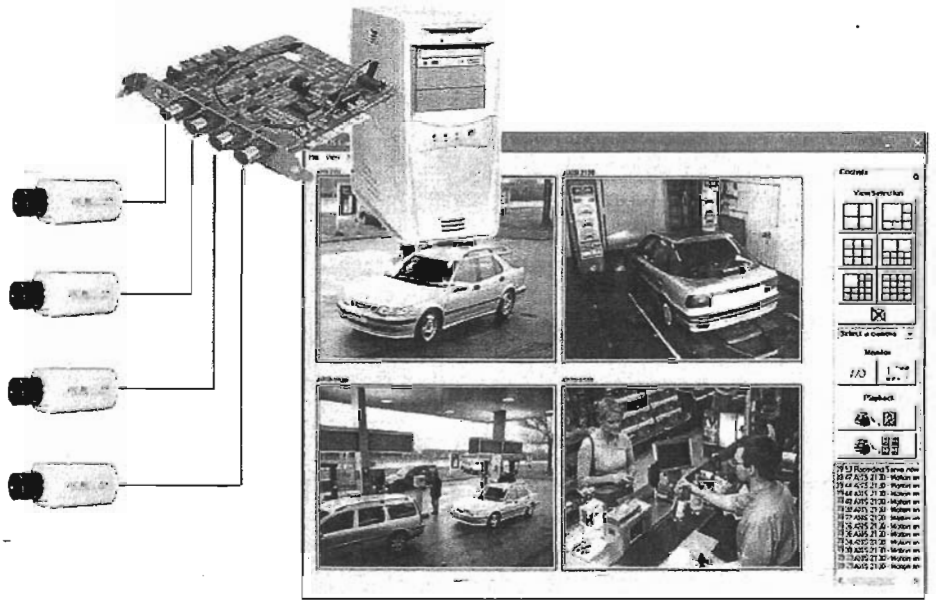
Además en algunos casos pueden existir cámaras más sofisticadas que permitan un cierto nivel de movimiento, el cual puede ser controlado remotamente o incluso hasta programado.

1.3. Sistemas CCTV basados en PC

Los sistemas analógicos han venido resolviendo las necesidades de observación, sin embargo, los usuarios se han visto con engorrosos procedimientos de grabación, visualización y administración. Con los problemas asociados de eventual perdida de nitidez, legibilidad y/o borrado de estas cintas a través del tiempo.

Con la constante evolución de la tecnología se han reemplazado los sistemas analógicos por digitales, mejorando la calidad de la imagen, la eficiencia en su almacenamiento y administración. Algunos de los sistemas actuales consisten en equipos con configuraciones muy similares a una computadora de escritorio, con tarjetas especializadas que permiten la conexión con las cámaras

tradicionales de vigilancia y realizan una transformación de la señal analógica a un medio digital para su almacenamiento en discos duros.



Sistema de video-vigilancia basado en PC.

En estos casos es común encontrar que el fabricante de la tarjeta que captura el video también proporcione un Software especializado que permita la visualización de las cámaras así como su grabación y añadiendo diversas herramientas que no solo ayudan a facilitar la administración del sistema sino que además incrementan su funcionalidad.

Dentro de las opciones que ofrece este tipo de sistemas podemos encontrar:

- Grabación y visualización de video en vivo de múltiples cámaras simultáneamente.

- Visualización de video en vivo de hasta 16 cámaras simultáneamente. Generalmente la interfaz de usuario proporciona un variado número de opciones que permiten la forma en que se distribuirán las imágenes en pantalla.

- Grabación continua permitiendo la configuración del lapso de tiempo entre cada imagen para maximizar el espacio de almacenamiento.

- Grabación programada, esto es, el sistema puede iniciar la grabación de cierta cámara en un horario determinado.

- Grabación activada por eventos. El sistema puede definir ciertas condiciones para que se inicie una grabación.
- Detección de movimiento con filtros de exclusión en la imagen.
- Grabaciones digitales donde la calidad es definida por las imágenes capturadas por la cámara.
- Limitaciones de grabación definidos por Hardware.
- Funciones de búsqueda de grabaciones.
- Mensajes del sistema en pantalla y en bitácoras.
- Múltiple reproducción de grabaciones a una velocidad controlable.
- Grabación de imágenes y videos en diferentes formatos.
- Notificación de eventos por medio de sonido o correo electrónico.
- Control de movimiento (pan/tilt/zoom) para cámaras, incluyendo la opción de establecer posiciones especificas.
- Accesos remotos para la visualización de video en vivo a través de Internet por medio de navegadores Web o programas cliente.
- Administración remota del sistema.
- Sistema de control descentralizado.

En general, los sistemas CCTV basados en computadoras personales proporcionan una alta funcionalidad debido a la utilización de todos los recursos disponibles en la computadora, sin embargo, ésta es también la que puede limitar sus capacidades debido a una baja velocidad del procesador, memoria o capacidad de almacenamiento insuficiente.

La característica más notable de estos sistemas radica en la manera en que integra toda su funcionalidad por medio de Software, en un CCTV tradicional, generalmente es necesario un equipo adicional cuando se requiere de una función especial, por ejemplo, cuando se requiere la grabación del video, es común utilizar videograbadoras especializadas, cuando es evidente el incremento en el número de cámaras puede también ser necesaria la adquisición de nuevos equipos como monitores, switchers o multiplexores; todos estos equipos requieren de una inversión importante, sin embargo, un sistema

basado en PC resuelve la mayoría de estos problemas con sencillas configuraciones de software.

1.4. Cámaras de Red[2]

Una cámara de red puede ser descrita como una cámara y un servidor integrados en un mismo equipo. Captura y transmite imágenes digitales en vivo directamente a través de cualquier red IP (por ejemplo: LAN, Intranet o Internet), permitiendo a los usuarios ver y/o manejar la cámara de forma remota a través de un navegador Web en cualquier lugar y en cualquier momento.

Una cámara de red posee su propia dirección IP y funciones de servidor integradas lo cual la hace totalmente autónoma. Todo lo necesario para ver las imágenes a través de la red está incluido dentro de la cámara. La cámara se conecta directamente a la red como cualquier otro dispositivo de red, tiene su propio software integrado, el cual incluye un servidor Web como principal herramienta para interactuar con el usuario, además cada fabricante puede agregar otros servicios y soporte a otros protocolos como son FTP, HTTPS, PING, SMTP, ARP, DHCP, etc.

Según el modelo de cámara podrá ir equipada con muchas otras funciones como son la detección de movimiento, salida de vídeo analógico, movilidad PAN/TILT, sensibilidad para luz infrarroja, etc.

Para mostrar de forma más detallada el funcionamiento de estos equipos, tomemos como ejemplo las cámaras de red del fabricante sueco Axis, uno de los líderes a nivel mundial en este rubro.

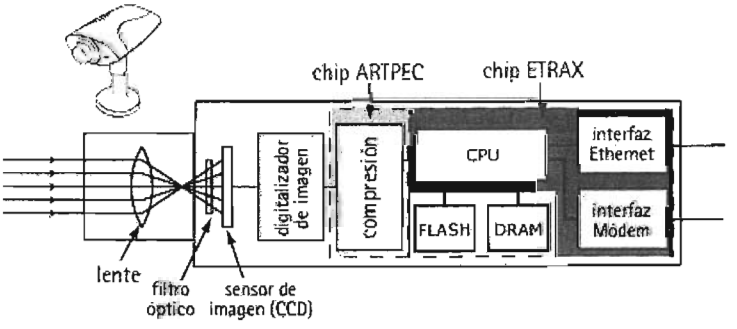


Diagrama interno de una cámara de red Axis

La lente de la cámara enfoca la imagen en el sensor de imágenes (CCD). Antes de llegar al sensor la imagen pasa a través del filtro óptico, que elimina cualquier luz infrarroja para que los colores mostrados sean "correctos". El sensor de

[2] Axis Communications. (Abril 24, 2003) Network camera technology. http://www.axis.com/products/video/camera/about_cameras/netcam_tech.htm

imagen convierte la imagen, compuesta por información lumínica, en señales eléctricas. Estas señales eléctricas digitales están ya en un formato que puede comprimirse y enviarse a través de la red.

El chip ARTPEC (Axis Real Time Picture EnCoder), desarrollado por Axis, es el que realiza las funciones de control de la cámara como son la gestión de la exposición, el balance de blancos (ajusta los niveles de color), la nitidez de la imagen y otros aspectos de la calidad de la imagen. El chip ARTPEC también incluye un componente de compresión de vídeo que comprime la imagen digital para su eficiente envío a través de la red. Actualmente es común encontrar que las cámaras de red utilizan una compresión JPEG, técnica propuesta por el Joint Photographic Experts Group que proporciona a las imágenes captadas un reducido tamaño así como una alta calidad. Cámaras mas profesionales, pueden incluso proporcionar video con calidad cercana al DVD gracias a la compresión MPEG-2 o MPEG-4.

La conexión Ethernet de la cámara se consigue gracias al chip ETRAX, una solución de sistema en un chip que permite conectar periféricos a la red. El ETRAX incluye una CPU de 32 bit, conectividad 10/100 Mb Ethernet, funcionalidad DMA (Direct Memory Access) avanzada y un amplio rango de interfaces de Entrada/Salida.

La CPU, memoria Flash y la memoria DRAM representan la parte más innovadora de estas cámaras pues están diseñadas específicamente para aplicaciones de red. Juntas, controlan la comunicación con la red y el servidor Web.

1.5. Dispositivos y aplicaciones móviles

1.5.1. Dispositivos móviles

El mercado de los dispositivos móviles ha crecido enormemente en los últimos años. Los primeros dispositivos eran simples teléfonos celulares para voz y agendas electrónicas con muy limitadas capacidades para datos personales. Conforme avanza la tecnología, más funciones se agregan a los pequeños dispositivos y nuevos productos son creados. Actualmente una gran variedad de dispositivos móviles están disponibles ofreciendo pantallas sensibles al tacto, un mayor poder de procesamiento y memoria, baterías de larga duración y pantallas más grandes y con capacidad de mostrar más colores. Muchos de estos dispositivos incorporan capacidades de cómputo y comunicación en un mismo equipo.

En el mercado, los dispositivos móviles incluyen laptop's, PDA's (Personal Digital Assistants), teléfonos celulares y teléfonos inteligentes. Las computadoras portátiles (laptop's) poseen capacidades de memoria y procesamiento muy

similares a sus contrapartes de escritorio. Pueden llevarse a viajes, a conferencias, pueden ser utilizados en aeropuertos y cuartos de hotel siendo de gran utilidad para hombres de negocio en sus viajes. Pero aún siguen siendo muy grandes para llevarlas a todo lugar todo el tiempo, por lo cual, no pueden considerarse en su totalidad como un dispositivo móvil. Las computadoras de mano o PDA's son más pequeñas y ligeras, pero sus funciones son más limitadas que las de una laptop. Estos dispositivos suelen ser considerados como una extensión de las computadoras de escritorio por sus capacidades para sincronización de datos. Los teléfonos celulares son utilizados principalmente para la comunicación por voz aunque también poseen la capacidad para transmitir datos; los hay de muchas formas y tamaños, en general, suelen ser más pequeños que un PDA aunque también suelen consumir más energía pues deben transmitir una señal de radiofrecuencia con suficiente poder para comunicarse. Los teléfonos inteligentes toman la capacidad de comunicación de los teléfonos celulares y la combinan con todas las posibilidades que puede dar un PDA.

En general podemos mencionar que los dispositivos móviles tienen ciertas características que los distinguen de otros equipos, estos son:

- Son aparatos pequeños y ligeros de modo que puedan transportarse fácilmente.
- Sus capacidades de procesamiento son limitadas.
- Tienen una memoria limitada la cual puede ser de unos cuantos Kilobytes hasta varios Megabytes.
- Están diseñados para funciones específicas, pero pueden llevar a cabo otras más generales.
- Poseen cierta capacidad de comunicación con otros equipos para transmitir o recibir datos, la cual puede ser a través de cables o de forma inalámbrica.

1.5.2. Tecnología inalámbrica

Las comunicaciones inalámbricas son un campo muy amplio, que cubren desde transmisiones de radio y televisión hasta localizadores, teléfonos móviles y comunicaciones satelitales, todas ellas apoyadas bajo el mismo principio de transmisión por medio de radiación electromagnética.

Concretamente las comunicaciones se realizan en el espectro de radio que comprende una banda de radiaciones de baja frecuencia. Las ondas de radio

operan en el rango de 3 KHz a 300 GHz, lo cual corresponde a longitudes de onda de 3 centímetros hasta 300 metros. Existen varias razones para usar esta banda de frecuencias: A diferencia de las longitudes de onda cortas, que son fácilmente absorbidas por objetos, las longitudes de onda largas pueden atravesar obstáculos (bosques, montes, pueblos, ciudades, etc.) sin una distorsión importante. Estas pueden viajar largas distancias, transmitida por satélites y reflejarse a medio camino para cruzar el continente. Además, las ondas de baja frecuencia se caracterizan por ser de baja energía, lo cual disminuye el riesgo sobre la salud humana. Finalmente, baja energía significa una transmisión más barata.

El espectro de radio, esta dividido en canales, cada uno ocupando una banda de 30 KHz. Estos pueden ser canales de control o comunicación. Los canales de comunicación transportan voz y datos mientras que los canales de control, que operan a una ligera frecuencia más alta, manejan las señales de entrada y salida, además de otras tareas de transmisión.

El área de los dispositivos móviles, incluyendo teléfonos celulares, se esta extendiendo rápidamente, al mismo tiempo estándares y protocolos son adoptados, usados, actualizados y en algunos casos descartados. En las comunicaciones inalámbricas; otra de las áreas que crece a un ritmo acelerado es el de las redes WLAN (Wireless Local Area Networks) o Wi-Fi. Apoyado por una amplia aceptación del estándar IEEE 802.11. Aunque las redes inalámbricas parecen un caso especial, es en realidad más intuitiva y natural que una red cableada. Actualmente los fabricantes de computadoras y dispositivos móviles ofrecen cada día más productos con soporte a estas tecnologías lo cual hará en un futuro no muy lejano que la conexión de una laptop a la red o la comunicación entre dispositivos a través de cables sea innecesaria.

Conceptualmente, las comunicaciones inalámbricas pueden dividirse en dos tipos: área local y área amplia. Existe una amplia variedad de dispositivos que cubren un rango local como los teléfonos inalámbricos de 900 MHz, juguetes de radio control, equipos con tecnología Bluetooth, redes Wi-Fi. Todos ellos operan en distancias cortas típicamente de unos cuantos metros.

Las comunicaciones de área amplia operan en distancias mucho más grandes. Un teléfono celular es el más claro ejemplo, donde se pueden establecer comunicaciones a cualquier teléfono sobre el planeta. Estos dispositivos de largo alcance se basan en una red terrestre mas elaborada. Un teléfono móvil no tiene más poder que un juguete, como para transmitir a grandes distancias, lo que realmente tiene es una red de antenas de radio (células) cuidadosamente colocadas, de modo que el teléfono puede seguir operando mientras se encuentre en un rango de distancia de por lo menos una célula. El teléfono recibe el servicio a través de un proveedor, la empresa encargada de operar la red terrestre.

1.5.3. Aplicaciones móviles

Las limitaciones fundamentales del hardware de los dispositivos móviles tienen un impacto importante en el diseño tanto del sistema operativo, como de las aplicaciones. El software tiene que ser pequeño para cumplir con las necesidades de memoria; los algoritmos tienen que ser eficientes para optimizar el uso de la batería y el poder de cómputo.

Cada dispositivo móvil contiene las aplicaciones para el uso específico al que fue diseñado, conforme estos dispositivos aumentan sus capacidades tanto de procesamiento como de memoria, se hace posible el correr aplicaciones más generales, sin embargo, un dispositivo móvil no debe considerase como un sistema de computo de propósito general para ejecutar simulaciones complejas o grandes aplicaciones debido a sus recursos limitados que no superaran a los de una computadora de escritorio.
Podemos dividir las aplicaciones para dispositivos móviles en cinco categorías:

- Aplicaciones "standalone" como juegos y utilerías.
- Software de productividad general (procesadores de textos, calendarios, agendas).
- Aplicaciones de Internet como son clientes de correo electrónico y navegadores Web.
- Aplicaciones de localización como guías turísticas y localización geoposicionada.
- Aplicaciones diseñadas a la medida.

CAPITULO 2

HERRAMIENTAS DE DESARROLLO

2.1. La Tecnología Java

La tecnología Java, creada por un equipo de desarrollo de Sun Microsystems encabezado por James Gosling, esta basada en un ambiente donde la redes de computadoras crecían día a día y en la idea de que un mismo software debería correr en cualquier tipo de computadora y otros dispositivos.

La tecnología Java se divide en dos componentes principales: El lenguaje de programación Java y la plataforma Java.

2.1.1. El Lenguaje de Programación Java

La tecnología Java tuvo su origen como herramienta de programación en un proyecto llamado "The Green Project" el cual pretendía crear una nueva convergencia entre diferentes dispositivos y las computadoras. El equipo iniciado por James Gosling, Patrick Naughton y Mike Sheridan creó un pequeño dispositivo multimedia llamado *7 ("StarSeven") el cual era capaz de controlar diferentes equipos a distancia, mientras mostraba animaciones en su pantalla a color LCD de cinco pulgadas. Aunque no tuvieron éxito para introducir su nueva creación al mercado, se dieron cuenta de que el lenguaje "Oak" (rebautizado como Java) creado para el proyecto era ideal para el creciente y popular Internet.

El lenguaje de programación Java ha madurado desde entonces definiendo un enfoque distinto a otros lenguajes de alto nivel y con ciertas características que lo hacen único y la mejor opción para programar sistemas robustos en ambientes distribuidos.[3]

2.1.1.1. Sencillo

Una de las principales características de Java es que es un lenguaje sencillo que puede ser utilizado sin un extenso entrenamiento. Los conceptos fundamentales de la Tecnología Java son comprendidos de manera rápida con lo cual los programadores pueden ser productivos desde el inicio.

[3] Sun Microsystems Inc. (1997). The Java Language Environment. Design Goals of the Java Programming Language. http://java.sun.com/docs/white/langenv/Intro.doc2.html#334

2.1.1.2. Orientado a Objetos

Java es un lenguaje diseñado a ser orientado a objetos desde el principio. La tecnología orientada a objetos ha demostrado su eficiencia en el software actual, donde las necesidades de los sistemas basados en una arquitectura cliente-servidor coinciden con los paradigmas de encapsulación del software basado en objetos.

Los desarrolladores tienen acceso a bibliotecas que proveen gran funcionalidad desde tipos básicos de datos, interfaces para tareas de red y operaciones de I/O, así como herramientas para ambientes de alto contenido gráfico. Todas estas bibliotecas pueden ser extendidas para darle una nueva funcionalidad.

2.1.1.3. Familiar

Antes de la creación de Java el lenguaje más utilizado por programadores era C++, por lo que era evidente que un nuevo lenguaje debería tener un estilo similar a C++ pero eliminando las innecesarias complejidades de éste, lo cual permitiría migrar fácilmente a la plataforma Java.

2.1.1.4. Robusto

El lenguaje de programación Java fue diseñado para crear software altamente fiable. Provee una extensa revisión en tiempo de compilación, seguida por una segunda revisión en tiempo de ejecución. Las características del lenguaje ayudan a los programadores a seguir buenos hábitos de programación.

El modelo de manejo de memoria es extremadamente simple: los objetos son creados con el operador *new*. No existe la aritmética de punteros y el reclamo de memoria se realiza automáticamente (garbage collection). Éste simple manejo de memoria elimina muchos de los errores de programación que se producción en C y C++.

2.1.1.5. Seguro

La tecnología Java esta diseñada para funcionar en ambientes distribuidos, lo que significa que la seguridad es de vital importancia. Con características en el lenguaje y en el sistema de ejecución, es posible construir aplicaciones seguras contra la intrusión de código sin autorización que intente crear virus o alterar los sistemas de archivos.

2.1.1.6. Arquitectura Neutral y Portátil

La tecnología Java esta diseñada para crear aplicaciones para un ambiente heterogéneo que puedan ser capaces de ejecutarse en una amplia variedad de arquitecturas de hardware y diferentes sistemas operativos. Para lograr esto el compilador de java genera *bytecodes,* el cual es un formato intermedio diseñado para transportar el código de manera eficiente a través de múltiples plataformas tanto de hardware como de software. La característica inherente de Java de ser un lenguaje interpretado ayuda a resolver el problema de distribución de archivos binarios; los mismos archivos creados por el compilador se ejecutarán en cualquier plataforma.

La portabilidad de un sistema no solo depende de su capacidad para funcionar en cualquier plataforma. Java toma la portabilidad de forma más estricta dentro de la definición misma del lenguaje especificando el tamaño para cada uno de los tipos de datos básicos y el comportamiento de sus operadores aritméticos. De esta forma los programas son los mismos y no hay incompatibilidad de tipos de datos de una arquitectura de hardware y software a otra.

La neutralidad en la arquitectura y la portabilidad del lenguaje Java es posible por lo que se conoce como *Java Virtual Machine* que no es más que la especificación de una maquina abstracta para la cual los compiladores generan su código. La implementación de la Java Virtual Machine para un plataforma de hardware y software especifico proveen una concreta realización de la maquina virtual.

2.1.1.7. Alto Desempeño

El desempeño es siempre un factor a considerar. La plataforma Java basa su desempeño adoptando una estrategia en la cual el intérprete se ejecuta a toda velocidad sin la necesidad de verificar el ambiente de ejecución. El garbage collector corre como una tarea de baja prioridad, asegurando una alta probabilidad de que la memoria este disponible cuando se requiera, de vital importancia para un mejor desempeño. Aplicaciones que requieren de gran cantidad de poder de procesamiento pueden ser diseñadas para que aquellas secciones de computo intensivo puedan ser rescritas en código nativo e interconectadas a la plataforma Java.

Desde su creación las aplicaciones Java han sido criticadas por su inferior desempeño en comparación con aplicaciones escritas en lenguajes como C++. La característica de Java de ser un lenguaje interpretado es la causa de este problema, sin embargo, nuevos intérpretes, una configuración óptima del ambiente de ejecución y algoritmos bien diseñados, permiten a cualquier aplicación Java tener un desempeño muy cercano a aquellas escritas en código nativo.

2.1.1.8. Interpretado

La característica de Java de ser un lenguaje interpretado no solo le permite ejecutar los bytecodes en cualquier plataforma donde se haya implementado la máquina virtual sino que además permite ciclos de desarrollo más rápidos y simples.

2.1.1.9. Multitarea

En la actualidad, aplicaciones como un navegador de páginas Web requieren de realizar diversas tareas al mismo tiempo, por ejemplo: un usuario podría correr diferentes animaciones mientras se descarga una imagen al mismo tiempo que se desplaza a través de la página. La capacidad de multitarea de la tecnología Java proporciona una plataforma para crear aplicaciones con muchos hilos de actividad, lo cual resulta en un alto grado de interactividad al usuario.

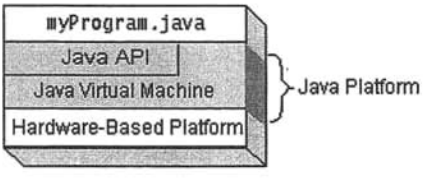
2.1.1.10. Dinámico

Mientras que el proceso de compilación de un programa Java se hace de manera muy estricta y estática, la fase de enlace se realiza de manera dinámica. Las clases son enlazadas solo cuando son requeridas. Nuevos módulos pueden ser enlazados de diversas fuentes incluso a través de la red, lo cual permite una forma simple y transparente de actualizar las aplicaciones.

2.1.2. La Plataforma Java

Una plataforma es un ambiente de hardware o software en el cual se ejecuta un programa, algunos ejemplos de estos son: Microsoft Windows, Linux, Solaris y MacOS. La mayoría de las plataformas pueden ser descritas como una combinación de un sistema operativo y hardware. La plataforma Java difiere sustancialmente de otras en que es una plataforma basada únicamente en software la cual se ejecuta sobre otras plataformas basadas en hardware.

La plataforma Java esta formada por dos componentes: The Java Virtual Machine (JVM) y The Java Application Programming Interface (Java API). La siguiente figura representa un programa corriendo en la plataforma Java. La figura, muestra como la plataforma aísla al programa del hardware.



2.1.2.1. The Java Virtual Machine

Anteriormente se mencionó que la Java Virtual Machine (JVM) es la responsable de la independencia de hardware de la tecnología Java.
La JVM es una computadora abstracta, esto es, tiene el mismo funcionamiento que una computadora real con la diferencia es que esta compuesta únicamente por software.

La JVM solo reconoce los archivos en el formato *class.* Los archivos en el formato class son aquellos creados por el compilador de Java en los cuales se almacenan las instrucciones para la maquina virtual (bytecodes), así como una tabla de símbolos y otra información relacionada.

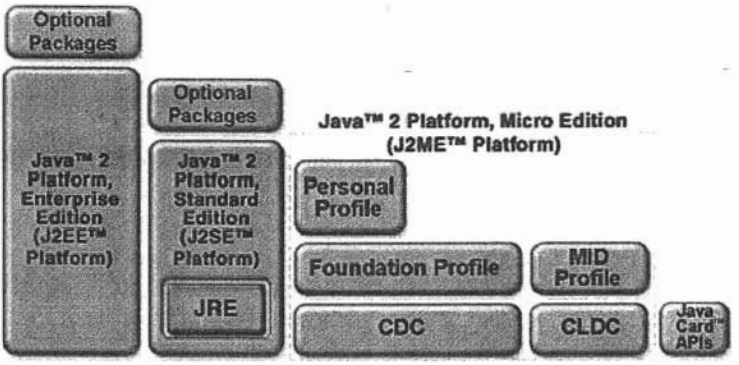
2.1.2.2. The Java Application Programming Interface

La Java Application Programming Interface (Java API) es una larga colección de componentes de software que proveen una amplia variedad de utilidades, como son componentes para la creación de interfaces de usuario (GUI), operaciones de I/O, conexiones de red, criptografía, manipulación de datos, etc. Ésta se encuentra organizada en bibliotecas de clases e interfases relacionadas; estas bibliotecas son conocidas como *paquetes* o *packages.*

2.2 Java 2 Platform, Standard Edition

Desde sus inicios la Tecnología Java fue creada pensando en un ambiente de red con una gran variedad de plataformas que variaban desde pequeños dispositivos con muy limitados recursos de memoria y procesamiento, una computadoras de escritorio hasta grandes servidores de alto desempeño con varios procesadores y grandes cantidades de memoria. Debido a la dificultad que se presenta el desarrollo de una plataforma que satisfaga los requerimientos de tan diversos sectores, el 15 de Junio de 1999 Sun Microsystems anunció la división de su Tecnología Java en tres diferentes plataformas: Java 2 Platform, Micro Edition (J2ME), Java 2 Platform, Standard Edition (J2SE), y Java 2 Platform, Enterprise Edition (J2EE).[4]

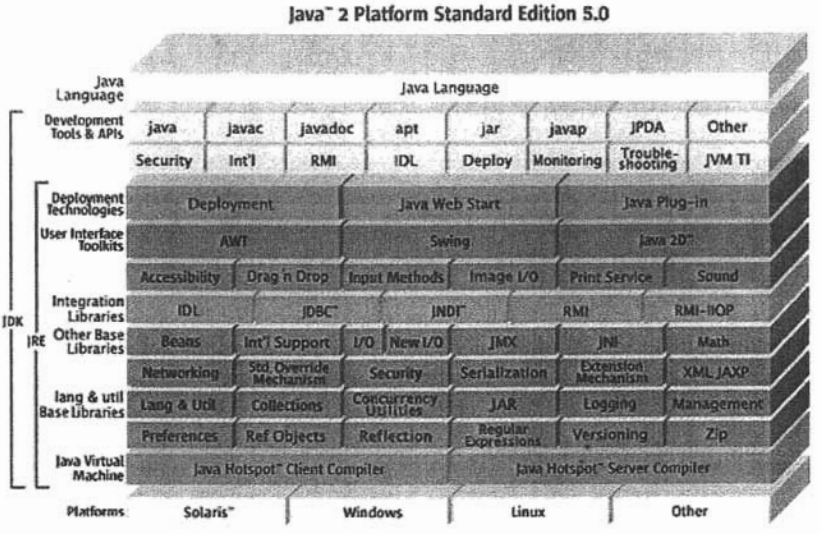
[4] Sun Microsystems Inc. (n.d.) The Java Platform: Five Years in Review. Sun Developer Network. http://java.sun.com/features/2000/06/time-line.html



Las diferentes versiones de la plataforma Java. http://java.sun.com/java2/whatis/

Existen dos distribuciones en la plataforma J2SE: J2SE Runtime Environment (JRE) y J2SE Development Kit (JDK). El JRE proporciona las Java APIs, la maquina virtual de Java y otros programas necesarios para ejecutar applets y aplicaciones escritos en el lenguaje de programación Java. Este es también la base para las tecnologías en la plataforma J2EE para el desarrollo e implementación de aplicaciones empresariales. El JDK contiene todo lo que esta en el JRE, además de herramientas como compiladores y depuradores para el desarrollo de aplicaciones.

El siguiente diagrama ilustra la forma en como esta integrada la plataforma J2SE.[5]



[5] Sun Microsystems Inc. (n.d.) Java 2 Platform Standar Edition Overview Sun Developer Network. http://java.sun.com/j2se/overview.html

La J2SE API consiste en una serie de tecnologías, las cuales se organizan en dos grupos: Core Java y Desktop Java.

El grupo Core Java proporciona paquetes esenciales para el desarrollo de aplicaciones empresariales en áreas claves como el acceso a bases de datos, seguridad, comunicaciones, etc. Las tecnologías incorporadas en este grupo hasta la versión J2SE 5.0 son:

- Java Authentication and Authorization Service (JAAS)
- Java Cryptography Extension (JCE)
- Java Secure Socket Extension (JSSE)
- Java Database Connectivity (JDBC) Technology
- Java Platform Debugger Architecture (JPDA)
- Javadoc Tool
- Internationalization
- Java Remote Method Invocation (RMI)
- Java Naming and Directory Interface (JNDI)
- Java Management Extensions (JMX) (opcional)
- JMX Remote API (opcional)
- Java Communications API (opcional)
- Java Telephony API (opcional)

Desktop Java contiene una amplia gama de características que permiten la creación de aplicaciones que mejoran la experiencia al usuario con el uso de interfaces gráficas y elementos multimedia. Algunas de las tecnologías incorporadas a Desktop Java son:

- Java Plug-in (parte de JRE)
- Java Web Start (parte de JRE)
- JavaBeans

- Java Foundation Classes (JFC/Swing)
- Java Sound API
- Java Media Framework (JMF) (opcional)
- Java 3D API (opcional)
- Java Advanced Imaging API (JAI) (opcional)
- Java Speech API (opcional)
- Java Help System (opcional)

2.3. Java 2 Platform, Enterprise Edition

La plataforma Java 2 Enterprise Edition (J2EE) ofrece un modelo para aplicaciones distribuidas multinivel (multitiered), componentes reutilizables, un modelo de seguridad unificado, control de operaciones flexible y soporte para servicios Web a través de un intercambio integrado de datos por XML (Extensible Markup Language).[6]

2.3.1 Aplicaciones Distribuidas Multinivel (Distributed Multitiered Applications)

Dentro de la plataforma J2EE la lógica de las aplicaciones esta dividida en componentes de acuerdo a su función, los cuales están instalados en diferentes equipos dependiendo el nivel al que pertenezca. Estas aplicaciones generalmente son consideradas de tres niveles y esto se debe a que están distribuidas en tres partes: el cliente, el servidor J2EE y un servidor de información o base de datos.

Las aplicaciones J2EE se forman de componentes. Un componente J2EE no es más que una unidad de software que es ensamblada dentro de una aplicación con sus respectivas clases y archivos y la cual puede comunicarse con otros componentes.

A continuación de definen los diferentes tipos de componentes existentes en la plataforma J2EE.

---

[6] Eric Armstrong, Jennifer Ball, Stephanie Bodoff, Debbie Bode Carson, Ian Evans, Dale Green, Kim Haase, et al. (Diciembre 16, 2004). The J2EE 1.4 Tutorial. Sun Developer Network. http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html

Clientes J2EE
Un cliente J2EE puede ser de dos tipos:

- *Clientes Web.* El cual consiste en páginas Web creadas dinámicamente por el servidor en diversos lenguajes (HTML, XML, etc.) y un navegador, el cual interpreta dichas páginas para mostrarlas al usuario. Dentro de esta categoría podemos incluir los *applets* que no son más que pequeñas aplicaciones escritas en el lenguaje de programación Java, las cuales se ejecutan dentro de la máquina virtual instalada en el navegador.

- *Aplicaciones Cliente.* Una aplicación ejecutándose en el equipo cliente proporciona una manera en que los usuarios manejen las tareas que requieren de una interfaz de usuario mucho más compleja.

Componentes Web
Los componentes Web están formados por servlets o páginas creadas utilizando la tecnología JSP. Los servlets son clases que procesan dinámicamente las peticiones y construyen una respuesta. Las páginas JSP son documentos de textos que se ejecutan como servlets pero permiten una aproximación más natural para la creación de contenido estático.

Componentes Empresariales
Los componentes empresariales representan la lógica que resuelve las necesidades de cada aplicación, las cuales son manejadas por enterprise beans ejecutándose en el nivel empresarial del lado del servidor.

2.3.2 Contenedores J2EE

Los contenedores son las interfaces entre los componentes y la funcionalidad de la plataforma específica que lo contiene. Antes de que cualquier componente web, enterprise bean, o aplicaciones cliente puedan ser ejecutados, primero deben ser ensamblados en un modulo J2EE desplegado en su contenedor.

Los contenedores proveen a los componentes de diversos servicios como el manejo de los ciclos de vida de servlets y enterprise beans, datos persistentes, modelos de seguridad, acceso a las APIs de la plataforma J2EE, etc.

Existen cuatro tipos de contenedores:

- *Contenedor de Enterprise JavaBeans.* Se encarga de manejar la ejecución de enterprise beans, los cuales se encuentran en el servidor.

- *Contenedores Web*. Maneja la ejecución de páginas JSP y servlets. Tanto los componentes web como el contenedor se ejecutan en el servidor.

- *Contenedores de Aplicaciones Cliente*. Se encarga de la ejecución de los componentes de las aplicaciones cliente.

- *Contenedor de Applets*. Maneja la ejecución de applets. Este contenedor esta formado por un navegador Web y un Java Plug-in corriendo en el equipo cliente.

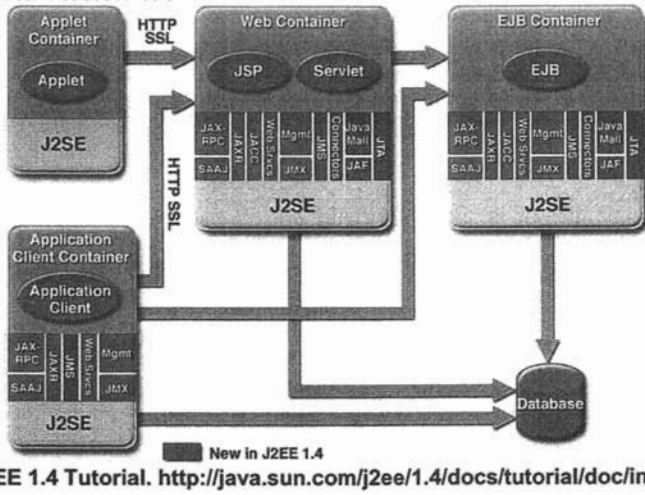
2.3.3 Web Services

Los Web Services o Servicios Web son aplicaciones Web basadas en estándares XML y protocolos de transporte para intercambiar información con clientes. La plataforma J2EE proporciona todas las herramientas necesarias, así como diferentes APIs para manipular documentos XML con lo cual se pueden crear servicios Web y clientes que pueden interactuar completamente con otros.

El uso de estándares XML permite que tanto los servicios Web como los clientes puedan comunicarse sin problemas, sin embargo, esto no significa que toda la información transmitida tenga que incluir etiquetas XML, debido a que la información puede ser por si misma texto plano, datos XML, o cualquier tipo de datos binarios como archivos de audio, video, mapas, programas, etc.

2.3.4 J2EE 1.4 APIs

La siguiente figura muestra la disponibilidad de APIs en cada tipo de contenedor presentes en la versión 1.4



The J2EE 1.4 Tutorial. http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html

A continuación se describe brevemente cada una de estas APIs.

Enterprise JavaBeans Technology
Un Enterprise JavaBeans (EJB) o enterprise bean, es solo una clase que contiene campos y métodos para implementar módulos de lógica empresarial el cual puede ser utilizado por si solo o en conjunto con otros enterprise beans. Frecuentemente estos componentes son los encargados de interactuar con las bases de datos.

Java Servlet Technology
Un servlet es una clase escrita en el lenguaje de programación Java la cual permite extender las capacidades de servidores que contengan aplicaciones que sigan el modelo de programación request-response. Aunque los servlets pueden responder a cualquier tipo de petición (request), son usados comúnmente en aplicaciones en servidores Web.

JavaServer Pages Technology
La tecnología JavaServer Pages o JSP, permite colocar pequeños trozos de código, basado en los servlets, directamente sobre un documento de texto. Una página JSP es un documento de texto que contiene dos tipos de texto: información estática (la cual puede expresarse en cualquier formato de texto como HTML, WML, y XML) y elementos JSP, los cuales determinan como la página construirá su contenido dinámico.

Java Message Service API
La Java Message Service (JMS) API es un estándar de mensajes que permite a los componentes de aplicaciones J2EE crear, enviar, recibir, y leer mensajes.

Java Transaction API
La Java Transaction API (JTA) define interfaces entre un administrador de operaciones y las partes involucradas en un sistema de operaciones distribuidas: la aplicación, el administrador de recursos y el servidor de aplicaciones.

JavaMail API
Las aplicaciones J2EE utilizan ésta API para enviar y leer notificaciones por medio de correo electrónico.

JavaBeans Activation Framework
Java Beans Activation Framework (JAF), utilizada por JavaMail, provee servicios para determinar el tipo de un determinado trozo de datos, encapsula su acceso,

descubre las operaciones disponibles sobre él y crea el componente JavaBean apropiado para realizar dichas operaciones.

Java API for XML Processing
Java API for XML Processing (JAXP) permite el procesamiento de documentos XML utilizando Document Object Model (DOM), Simple API for XML (SAX), y Extensible Stykesheet Lenguaje Transformations (XSLT). JAXP permite leer y transformar documentos XML independientemente de la implementación con la que fue creado.

Java API for XML-Based RPC
La Java API for XML-based RPC (JAX-RPC) utiliza el estándar SOAP y HTTP, para que aplicaciones cliente, puedan realizar llamadas a procedimientos remotos (RPCs) basados en XML a través de Internet proporcionando encriptación de datos y autentificación entre el cliente y el servidor. JAX-RPC también soporta WSDL, lo cual permite interactuar con clientes y servicios que no estén basados en la plataforma Java, como lo es .NET

SOAP with Attachments API for Java
SOAP with Attachments API for Java (SAAJ) en una API de bajo nivel de la cual depende JAX-RPC. SAAJ permite la producción y el consumo de mensajes que forman parte de la especificación SOAP 1.1.

Java API for XML Registries
Java API for XML Registries (JAXR) permite el acceso a registros empresariales y de propósito general sobre la Web.

J2EE Connector Architecture
La arquitectura de conectores sobre la plataforma J2EE es utilizada por vendedores de herramientas J2EE e integradores de sistemas para crear adaptadores de recursos (*resource adapter*) que permitan el acceso a sistemas de información empresarial los cuales pueden ser incluidos en cualquier producto J2EE. Un adaptador de recursos es un componente de software que permite a las aplicaciones acceder e interactuar con sistemas de información específicos de cada fabricante como son las bases de datos.

JDBC API
La JDBC API permite invocar comandos SQL desde métodos en el lenguaje de programación Java. Esta API puede utilizarse en conjunto con los enterprise beans lo cual permite una implementación de acceso a base de datos que no

contiene código JDBC o comandos SQL. También es posible utilizar la API desde servlets o páginas JSP para acceder directamente a la base de datos.

Java Naming and Directory Interface
Java Naming and Directory Interface (JNDI) proporciona métodos para realizar operaciones con directorios, como la asociación de características a objetos y la búsqueda de dichos objetos por medio de sus características.

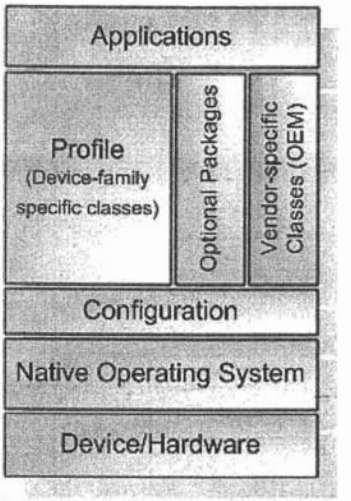
Java Authentication and Authorization Service
Java Authentication and Authorization Service (JAAS) suministra métodos para autentificar y autorizar a un usuario específico o un grupo de usuarios.

2.4. Java 2 Platform, Micro Edition

Java 2 Platform, Micro Edition o J2ME cumple con la finalidad original de la tecnología Java, la cual era proporcionar un ambiente para el desarrollo de aplicaciones en pequeños dispositivos electrónicos y sistemas embebidos con recursos limitados. A diferencia de J2SE y J2EE la plataforma J2ME es una colección de tecnologías y especificaciones que se encuentran enfocadas a diferentes segmentos del mercado de dispositivos electrónicos, los cuales pueden variar enormemente desde la cantidad de memoria disponible y la velocidad de procesamiento hasta sus capacidades de conexión e interfaces para interactuar con el usuario.

La plataforma J2ME tiene una organización multicapa formada por tres tipos de software: configuraciones, perfiles y paquetes opcionales. Una configuración como CLDC proporciona servicios fundamentales para una amplia variedad de dispositivos. Un perfil como MIDP provee de servicios de alto nivel a una clase más específica de dispositivos. Los paquetes opcionales añaden servicios especializados que pueden ser útiles a una gran cantidad de dispositivos pero no necesariamente a todos ellos.

La siguiente figura muestra una pila de software la cual ilustra la organización de la plataforma, la cual agrega funcionalidad conforme se desplaza hacia arriba.



Organización de la plataforma J2ME

2.4.1. Configuraciones

Una configuración define un ambiente de ejecución básico para las aplicaciones, esto incluye la máquina virtual y un conjunto de clases principalmente derivadas de J2SE. Cada configuración esta adaptada a un amplio grupo de dispositivos con capacidades similares. Al momento de escribir este trabajo se tienen definidas dos configuraciones: Connected Limited Device Configuration (CLDC) y Connected Device Configuration (CDC).

Connected Limited Device Configuration
CLDC es la mínima configuración dentro de J2ME, la cual esta diseñada para dispositivos con severas restricciones en poder computacional, duración de batería, memoria, y conectividad. Estas limitaciones afectan directamente el tipo de aplicaciones que pueden soportar.

La configuración CLDC soporta dispositivos con procesadores de 16 o 32 bits con un mínimo de 160 KB de memoria persistente y por lo menos 32 KB de memoria volátil. El nivel de consumo de energía es bajo, debido a que estos dispositivos son comúnmente alimentados por baterías. Sus capacidades de conexión pueden ser intermitentes y de baja velocidad. En el corazón de la configuración encontramos a la máquina virtual llamada K Virtual Machina (KVM) su versión optimizada CLDC Hotspot Implementation.

CLDC define un subconjunto de paquetes del Java Core y añade otras clases como el paquete Generic Connection Framework (GCF), el cual permite soporte para operaciones de I/O sin el uso de extensos paquetes como *java.net* y *java.io*.

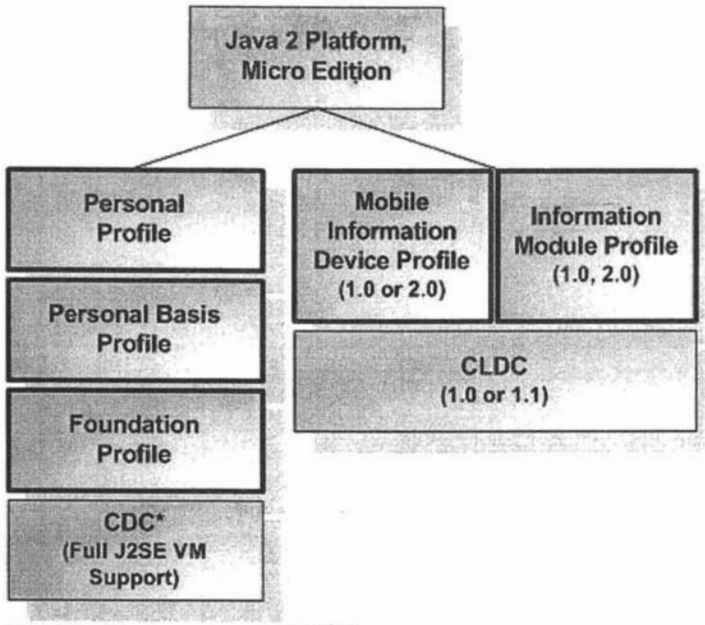
Connected Device Configuration
La configuración CDC esta diseñada para dispositivos más poderosos de los que soporta CLDC, como teléfonos inteligentes, PDAs y dispositivos embebidos más sofisticados.

Los dispositivos con soporte para CDC tienen procesadores de 32 bits, por lo menos 2 MB de memoria principal y 2.5 MB de memoria de solo lectura, y algún tipo de conectividad. Al igual que CLDC el principal componente de esta configuración es la máquina virtual como CDC Hotspot Implementation, que contiene todas las capacidades de determinada versión de J2SE y debido a que CLDC es un subconjunto de CDC, éste contiene también todas las clases y paquetes de CLDC, incluyendo GCF.

2.4.2. Perfiles

Las configuraciones no proporcionan clases para el manejo del ciclo de vida de las aplicaciones, el uso de interfases de usuario, manejo de datos persistentes almacenados en el dispositivo, o el acceso seguro a información almacenada en servidores. Todas estas funcionalidades son definidas por los perfiles, o por paquetes opcionales.

Un perfil agrega clases específicas al conjunto proporcionado por las configuraciones, clases que pretenden un uso específico el cual no se define en la configuración. Al momento existen dos perfiles basados en CLDC: Mobile Information Device Profile (MIDP, en su versión 1.0 y 2.0) y Information Module Profile (IMP). Tres perfiles más se encuentran basados en CDC: Foundation Profile (FP), Personal Basis Profile (PBP), y Personal Profile (PP). La siguiente figura muestra como actualmente se encuentra dividida la plataforma J2ME y como los perfiles se construyen sobre las configuraciones, además de que un perfil puede extender las capacidades de otro.



***CDC 1.0 is based on J2SE 1.3**
***CDC 1.1 is based on J2SE 1.4**

Configuraciones y perfiles de J2ME.

Perfiles Basados en CLDC: MIDP y IMP

MIDP fue el primer perfil creado para J2ME, y es el más adoptado en el mundo, principalmente en PDA y teléfonos celulares.

El Mobile Information Device Profile y el Information Module Profile son muy similares el uno del otro, pues están diseñados para dispositivos con características muy semejantes, sin embargo, IMP fue diseñado para dispositivos con poca o nula capacidad para una interfaz de usuario.

MIDP en su versión 1.0, es ampliamente utilizado por compañías como SonyEricsson, Nokia y Motorola en sus teléfonos celulares aunque cada día estos fabricantes ofrecen más modelos con soporte para MIDP 2.0, lo cual incrementa las capacidades del perfil para crear aplicaciones más robustas. Además de las APIs originales para el manejo de conexiones, interfaces de usuario, datos persistentes y ciclo de vida del MIDlet (aplicación basada en el perfil MIDP), la versión 2.0 agrega nuevas capacidades como soporte para TCP socket streams, datagramas UDP, conexiones seguras, APIs para el manejo de sonido e incluso APIs diseñadas para la creación de juegos.

MIDP 2.0 incluye formalmente la especificación de Over-the-Air (OTA) User-Initiate Provisioning, el cual fue originalmente un agregado a MIDP 1.0, en donde

se describe como los usuarios pueden encontrar y descargar aplicaciones sobre redes inalámbricas.
La especificación también indica el uso de CLDC 1.0, sin embargo, nada impide el uso de CLDC 1.1 como la base de cualquiera de las versiones de MIDP.

Perfiles Basados en CDC: FP, PBP, PP
CDC es una configuración más extensa que CLDC. Fue creada para dispositivos con mayores capacidades de memoria y procesamiento, y sigue el mismo modelo de configuraciones y perfiles. CDC provee un grupo de paquetes generales para el dispositivo, mientras que uno o más perfiles agregan clases apropiadas para la categoría a la que pertenece el dispositivo.

El Foundation Profile es la base sobre la que se han creado otros perfiles. EL FP extiende la configuración agregando clases para seguridad, utilidades y otras, sin embargo, no provee ninguna clase para interfaces de usuario.

El Personal Basis Profile se construye sobre el Foundation Profile agregando interfaces y clases que incluyen el Abstract Windowing Toolkit (AWT), el cual da soporte para imágenes y objetos para la creación de sencillas interfaces de usuario. Este perfil también agrega soporte para la programación de JavaBeans y el nuevo modelo de aplicaciones Xlet.

El Personal Profile hereda todas los paquetes de la configuración y de los perfiles sobre los que esta construido además de agregar más clases de AWT y soporte para Applets. EL Personal Profile provee un ambiente para aplicaciones más completo, uno muy similar a J2SE. Esta dirigido a dispositivos que requieren de una interfaz de usuario más compleja y conexiones de red seguras.

2.4.3. Paquetes Opcionales

Los paquetes opcionales son una parte muy importante en la arquitectura de la plataforma J2ME y como su nombre lo indica, estos paquetes son opcionales por tanto un fabricante de dispositivos puede decidir incluirlos o no en producto en particular.
Los paquetes opcionales pueden verse como una extensión de los perfiles los cuales dan soporte a servicios muy específicos que algunos dispositivos necesiten pero otros no, como lo son el envío de mensajes, capacidades multimedia o servicios de localización. Los perfiles pueden enfocarse a proporcionar un ambiente de ejecución únicamente con las clases que todos los dispositivos necesiten, mientras que los paquetes opcionales agreguen funcionalidades específicas.

El número de paquetes opcionales se incrementan día a día, algunos para CLDC, otros para CDC y algunos más para ambas configuraciones. A continuación se enlistan algunas de las APIs que actualmente forman parte de la plataforma J2ME:

- *Java API for Bluetooth (JABWT).* Bluetooth es una tecnología de radio de corto alcance y una pila de software utilizada para la transmisión de voz y datos. Esta API agrega a la plataforma la funcionalidad de Bluetooth en dispositivos habilitados con esta tecnología.

- *Wireless Messaging API (WMA).* WMA proporciona una API para el envío y recepción de mensajes de texto y mensajes binarios, típicamente mensajes Short Messaging Service (SMS) y Multimedia Messaging Service (MMS).

- *Mobile Media API (MMAPI).* Mobile Media API agrega funcionalidades para el procesamiento multimedia en equipos con capacidades avanzadas de sonido y multimedia.

- *Web Services API for J2ME (WSA).* WSA define dos paquetes opcionales, uno para la interacción con servicios Web remotos y otro para el procesamiento de XML, la cual esta basada en un subconjunto de Java API for XML-Based RPC (JAX-RPC 1.1).

- *Location API for J2ME.* Esta API provee a los dispositivos de clases para la creación de aplicaciones basadas en la localización.

- *Security and Trust Services API (SATSA).* SATSA define varios paquetes que implementa servicios de seguridad.

- *SIP API for J2ME.* Una API enfocada en el Protocolo de Inicio de Sesión (Session Initiation Protocol) el cual es usado en el manejo de sesiones permitiendo a las aplicaciones determinar las capacidades multimedia de los usuarios e invitarlos a unirse a una sesión multimedia, estableciendo, manteniendo y terminando la sesión.

- *J2ME RMI Optional Package (RMI OP).* RMI OP le da a CDC y a Foundation Profile una limitada versión de RMI API en la plataforma J2SE.

- *JDBC Optional Package for CDC/Foundation Profile.* Esta API proporciona clases para que dispositivos CDC tengan acceso a bases de datos.

2.4.4. Java Technology for the Wireless Industry

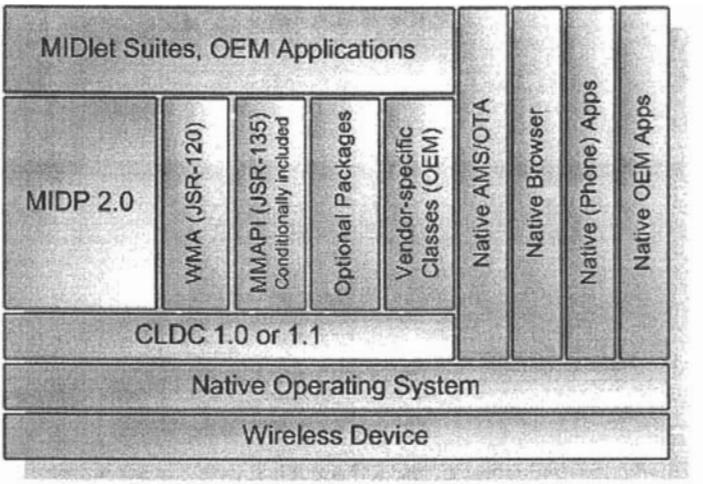
La plataforma J2ME ha producido una amplia variedad de APIs desde su creación, y muchas de estas han ganado gran aceptación. Aunque es importante que todas estas tecnologías sean llevadas a un mercado bien definido, de una manera consistente, para conseguir un alto grado de compatibilidad e interoperabilidad entre teléfonos móviles. Para cumplir esta meta, la especificación *Java Technology for the Wireless Industry* (JTWI) define una arquitectura común en dispositivos basados en las siguientes especificaciones:

- Plataforma J2ME
- Connected Limited Device Configuration
- Connected Limited Device Configuration 1.1
- Connected Device Configuration
- Mobile Information Device Profile
- Mobile Information Device Profile 2.0
- Wireless Messaging API
- Mobile Media API

Dispositivos con JTWI deben cumplir con lo siguiente:

- Una configuración mínima de CLDC 1.0
- Debe cumplir obligatoriamente con MIDP 2.0 y WMA 1.1
- De forma condicional también es necesario de MMAPI 1.1

Definiendo un ambiente de ejecución, APIs y un conjunto de tecnologías asociadas, JTWI asegura un entorno consistente para que las aplicaciones funcionen adecuadamente entre los diferentes dispositivos. Desde el punto de vista de la arquitectura JTWI, define la organización y la relación de sus componentes como lo muestra la figura.



Organización de JTWI.

2.5. Herramientas de Programación

Sun Microsystems proporciona un soporte completo a la tecnología Java incluyendo herramientas para el desarrollo de aplicaciones. Una de las distribuciones más importantes de Java (JDK) contiene todas las herramientas que programador necesita incluyendo: compilador, intérprete, depuradores, etc.

Aunque dentro de las distribuciones JDK se encuentra todo lo necesario para la creación de todo tipo de aplicaciones baja la plataforma Java, existen programas los cuales permiten al programador la creación de aplicaciones de forma más rápida y eficiente.

2.5.1. Ambiente Integrado de Desarrollo

Un ambiente integrado de desarrollo o Integrated Development Environment (IDE) es un programa compuesto por un conjunto de herramientas especializadas en la programación el cual puede estar dedicado a un lenguaje específico o bien, poder utilizar varios.

Algunas de las características más importantes con las que cuenta cualquier IDE son:

- Escribir y editar código fuente.
- Visualizar errores mientras se escribe.

- Resaltar la sintaxis del código.
- Automatizar tareas repetitivas.
- Compilar y/o interpretar código.
- Navegación a través de las estructuras y clases.
- Visualización de documentación.
- Uso de utilidades para una fácil creación de elementos, como lo pueden ser objetos gráficos o la creación de conexiones a bases de datos.

Además algunos IDEs proporcionan otras funcionalidades como:

- Plantillas para un rápido desarrollo de componentes Web.
- Completado de código mientras se escribe.
- Creación automática de clases, métodos y propiedades.
- Integración con repositorios de código como CVS.
- Integración con servidores Web.
- Integración con utilidades de construcción como Apache Ant.
- Monitoreo HTTP para el depurado de aplicaciones Web.
- Interfaz incorporada para la depuración de código.
- Macros y abreviaciones.
- Soporte para UML.

Actualmente existe una gran variedad de IDEs disponibles como JCreator de XINOX Software, IntelliJ IDEA de JetBrains, Eclipse de IBM, JBuilder de Borland, Sun Java Studio de Sun Microsystems, Netbeans IDE, etc. Aunque todos estos IDEs tienen características en común, cada empresa u organización agregan herramientas o funciones únicas, algunas de ellas desarrollan diferentes versiones enfocadas a diferentes plataformas o mercados; por ejemplo Borland ofrece tres versiones de su software JBuilder: *Foundation, Developer y Enterprise* cada uno de los cuales agrega mayor funcionalidad al anterior, sin embargo, solo la versión Foundation es gratuita.

2.5.2. Netbeans IDE[7]

Netbeans es un proyecto establecido en la comunidad Open Source ("código abierto") el cual esta dedicado al desarrollo de dos productos: NetBeans Platform y NetBeans IDE.

NetBeans Platform es una serie de componentes que proporcionan un ambiente con características comunes sobre el cual puede construirse una aplicación. NetBeans IDE es un ambiente integrado de desarrollo basado en NetBeans Platform, ambos desarrollados completamente con tecnología Java, lo cual inherentemente lo convierte en un IDE multiplataforma.

NetBeans IDE es uno de los IDEs más completos de su categoría ya que proporciona un ambiente completo para el desarrollo de aplicaciones en cualquiera de las tres plataformas disponibles en la tecnología Java.

Algunas de las características más importantes que ofrece son:

Soporte J2SE 5.0. NetBeans IDE ofrece un completo soporte de la versión 5.0 de J2SE, incluyendo las nuevas características del lenguaje, Java Beans, Swing, JavaDoc, etc.

Sistema de proyectos basado en Apache Ant. Apache Ant en una herramienta basada en Java para estandarizar y automatizar ambientes para desarrollo de aplicaciones. NetBeans IDE construye sus proyectos sobre Ant almacenando toda la información necesaria en scripts y archivos XML.

Soporte para aplicaciones Web. En la creación de aplicaciones Web, NetBeans IDE ofrece soporte para J2EE 1.3 (Servlet 2.3 y JSP 1.2), así como, J2EE 1.4 (Servlet 2.4 y JSP 2.0). Además cuenta con soporte para el manejo de bases de datos y un servidor Web completo (Apache Tomcat 5.0.28) para la ejecución de servlets y paginas jsp.

Soporte para aplicaciones móviles. Con la adición de un modulo extra, NetBeans IDE permite el desarrollo de aplicaciones para la plataforma J2ME dando soporte completo a las versiones 1.0 y 1,1 de CLDC, así como, la versión 1.0 y 2.0 de MIDP, además soporte OTA, J2ME Wireless Toolkit 2.2 con la posibilidad de la integración de emuladores de terceros para crear un ambiente robusto de pruebas.

Estas características hacen que NetBeans IDE sea la herramienta más adecuada para la creación de aplicaciones en las cuales se requiere del manejo de las plataformas J2SE, J2EE y J2ME.

---

[7] NetBeans.org (n.d.) NetBeans IDE 4.0. http://www.netbeans.org/community/releases/40/

CAPITULO 3

GESTION DE VIDEO*

3.1. Time-Based Media

Cualquier tipo de información que sufre un cambio significativo respecto al tiempo puede ser considerado un *Time-Based Media*, como ejemplos de este tipo de datos encontramos, clips de audio y video, secuencias MIDI y animaciones, las cuales pueden provenir de varias fuentes, como archivos locales o en la red, cámaras, micrófonos, transmisiones en vivo, etc.

En este capítulo de describen las características de Time-Based Media siguiendo el modelo fundamental de procesamiento de datos (entrada – procesamiento – salida).

3.1.1. Flujo de Datos

Una característica clave de todo time-based media es que requiere de un procesamiento y entrega oportunos. Una vez que el flujo de datos comienza, existen estrictos límites de tiempo que deben ser cumplidos, tanto en términos de recepción como de representación de los datos. Por esta razón, todo time-based media también es llamado *streaming media*. La información es entregada en un flujo de datos constante que debe ser recibida y procesada en el momento indicado para producir resultados aceptables.

Por ejemplo, cuando un video es reproducido, si los datos no pueden ser entregados lo suficientemente rápido, pueden presentarse pausas o retrasos en la reproducción, por otro lado, si los datos no pueden ser recibidos o procesados rápidamente, la película puede aparentar dar saltos en la secuencia, pues ciertos datos se pierden o son intencionalmente ignorados en un intento por mantener la velocidad de reproducción.

Content Type
El formato en el cual la información es almacenada es llamado *content type* o tipo de contenido. El tipo de contenido es en esencia un sinónimo de formato de archivo, sin embargo, éste termino no es utilizado debido a que los datos son con frecuencia obtenidos de fuentes diferentes a archivos locales. Como ejemplos de tipo de contenido podemos entontar: QuickTime, MPEG y AVI.

* La gestión de video se basa en el uso de la Java Media Framework API explicada a detalle en la guia: Sun Microsystems, Inc. (Mayo 11, 1998). JMF Programmers Guide. Sun Developer Network. http://java.sun.com/products/java-media/jmf/2.1.1/guide/JMFTOC.html

Media Streams

Un flujo de datos o *media stream* son los datos obtenidos de un archivo local, adquiridos a través de la red, o capturados por medio de una cámara. Un flujo de datos a menudo múltiples canales de datos llamados *tracks*. Por ejemplo, un archivo Quicktime puede contener un track para audio y uno para video. Aquellos flujos de datos que contienen más de un track son llamados *multiplexados* o *complejos*. La demultiplexión es el proceso de extraer cada track individual de un flujo de datos complejo.

El tipo de track define el tipo de datos que contiene, el formato especifica como los datos del track deben ser estructurados.

Un flujo de datos puede ser identificado por su localización y el protocolo utilizado por acceder a él. Por ejemplo un URL puede ser usado para obtener un archivo QuickTime, si el archivo se encuentra en la máquina local, puede ser leído a través del protocolo FILE, mientras que si se localiza en un servidor Web, el archivo puede ser leído por medio del protocolo HTTP.

Existen dos tipos de flujo de datos de acuerdo a como la información es entregada:

- *Pull*. Donde la información es iniciada y controlada por el cliente. Por ejemplo, HTML y FILE ambos son protocolos pull.

- *Push*. En este caso el servidor inicia la transferencia y controla el flujo de datos.

Formatos y Codecs

Cuando se trabaja con contenido multimedia como el video, es de esperarse que hablemos de grandes cantidades de información, la cual no solo aumenta mientras mayor calidad tenga, sino que además debe ser entregada oportunamente para una correcta reproducción. Teniendo en cuenta que un flujo de datos puede provenir de cualquier fuente, se hace evidente la necesidad de aumentar la velocidad con la que se transmiten los datos o desde otro punto de vista, disminuir la cantidad de datos a transmitir, este problema es más notorio cuando el flujo de datos se encuentra en la red.

Un *formato* o *codec* es un método de compresión para el flujo de datos de forma que permita disminuir la cantidad de datos que representan el video. Cada codec esta diseñado para satisfacer ciertas necesidades que depende de el uso final que se le de al video, así por ejemplo, existen codecs, como H.261 y H.263 usados para videoconferencias a través de Internet, sin embargo, el alto grado de compresión de estos formatos, también disminuye la calidad de las imágenes presentadas.

La siguiente tabla muestra algunos de los formatos más comunes identificando las principales características que se deben tomar en cuenta al utilizar cada uno de ellos.
La columna CPU indica el poder de procesamiento necesario para una óptima presentación con el codec especificado. La columna de Ancho de Banda señala la velocidad de transmisión necesaria para enviar o recibir los datos lo suficientemente rápido para una óptima presentación.

Codec	Tipo de Contenido	Calidad	CPU	Ancho de Banda
Cinepak	AVI, QuickTime	Media	Baja	Alta
MPEG-1	MPEG	Alta	Alta	Alta
H.261	AVI, RTP	Baja	Media	Media
H.263	QuickTime, AVI, RTP	Media	Media	Baja
JPEG	QuickTime, AVI, RTP	Alta	Alta	Alta
Indeo	QuickTime, AVI	Media	Media	Media

3.1.2. Presentación de los Datos

La mayoría de los time-based media como el audio y el video son presentados a través de dispositivos de salida como bocinas y monitores. Estos dispositivos son el destino más común, sin embargo, el flujo de datos puede también ser enviados a otros destinos, por ejemplo, puede ser almacenado en un archivo o transmito a través de la red. A este destino se le llama *data sink*.

Control de Presentación
Generalmente, mientras un flujo de datos es representado, éste es acompañado de una serie de controles que le permiten al usuario el control de la reproducción (play, forward, rewind, pause, etc.), sin embargo, estos controles no son un elemento indispensable.

Latencia
En muchos casos, en particular cuando el flujo de datos proviene de la red, la presentación de la información no puede comenzar inmediatamente. El tiempo que existe antes de que la presentación pueda iniciarse es conocido como *latencia de inicio* o simplemente *latencia*.

Calidad de Presentación
La calidad de la presentación depende muchos factores, de los cuales los más importantes son:

- El método de compresión utilizado, lo cual se refiere, directamente al codec utilizado.

- La capacidad de procesamiento del sistema.

- El ancho de banda disponible el cual es especialmente importante en flujo de datos obtenidos en la red.

Generalmente, mientras más alta es la calidad, más grande es el flujo de datos, mayor la capacidad de procesamiento y ancho de banda requeridos. El ancho de banda usualmente se representa como el número de bits que pueden ser transmitidos en un periodo de tiempo especifico, a esto se le conoce como *bit rate*.

En lo que a video de alta calidad se refiere, el número de *frames* o imágenes mostradas en un periodo de tiempo es llamado *frame rate* y este debe ser tan alto como sea posible. Generalmente, un frame rate de 30 frames por segundo es considerado adecuado para la mayoría de las aplicaciones.

3.1.3. Procesamiento de Datos

En la mayoría de los casos el flujo de datos pasa por una serie de procesos antes de ser presentado al usuario:

1. Si el flujo de datos contiene más de un track, éstos son extraídos.
2. Si cada track se encuentra comprimido, estos son decodificados.
3. Si es necesario, cada track es convertido a un nuevo formato.
4. Opcionalmente puede aplicarse algún otro proceso sobre los tracks decodificados. Por ejemplo, la aplicación de algún filtro o efecto especial.

Cada track es entonces enviado a su propio dispositivo de salida. Si el flujo de datos debe ser almacenado en un archivo entonces se deben seguir los siguientes pasos:

1. Cada track incluido debe ser capturado.
2. De ser necesario se aplica algún tipo de proceso especial a cada track.
3. Se codifica cada track, utilizando el codec deseado.
4. El conjunto de tracks son multiplexados en un solo flujo de datos.
5. El flujo de datos obtenido es entonces almacenado en un archivo.

Multiplexores y Demultiplexores
Un demultiplexor extrae cada uno de los track que conforman un flujo de datos complejo o multiplexado. Un *multiplexor* realiza la función opuesta, toma una serie de tracks y los une en un solo flujo de datos multiplexado.

Codecs
Como se menciono anteriormente los codecs son una forma de comprimir el flujo de datos. Se dice que un track es codificado cuando se convierte a un formato comprimido el cual es conveniente para almacenamiento o transmisión; cuando

éste es decodificado es convertido a un formato no comprimido (raw) el cual es más conveniente para su presentación.

Filtros
Un filtro modifica de alguna manera los datos de cada track, comúnmente para crear efectos especiales como el blur. Generalmente, los filtros son aplicados a datos sin comprimir.

Composición o Compositing
El proceso de composición es la combinación de múltiples track en un solo medio de presentación. Por ejemplo, la sobre posición de texto sobre el video es una forma común de composición.

3.1.4. Captura o Media Capture

Un flujo de datos puede ser capturado desde una fuente en vivo (*live source*) para su procesado y reproducción. Por ejemplo, una tarjeta de captura de video puede ser usada para obtener video de una cámara, de la misma forma que se puede obtener el video una cámara habilitada con un puerto Firewire. La captura de video puede ser considerada como la fase de entrada (input) dentro del modelo de procesamiento de datos.

Dispositivos de Captura
Para la captura de video es necesario el uso de hardware especializado, por ejemplo, un puerto firewire comúnmente utilizado por las video cámaras digitales. El formato en el cual se encuentra el flujo de datos depende enteramente de los procesos que realice el dispositivo de captura. Algunos dispositivos realizan muy poco procesamiento con lo cual entregan los datos en formato raw, otros dispositivos pueden entregar un flujo de datos en diferentes formatos como MJPEG, MPEG-4, etc.

Controles de Captura
De la misma forma en que se proporciona controles para la reproducción, existen también controles que permiten el manejo del proceso de captura, dando al usuario la posibilidad de seleccionar el bit rate o el tipo de codec a utilizar.

3.2. Java Media Framework

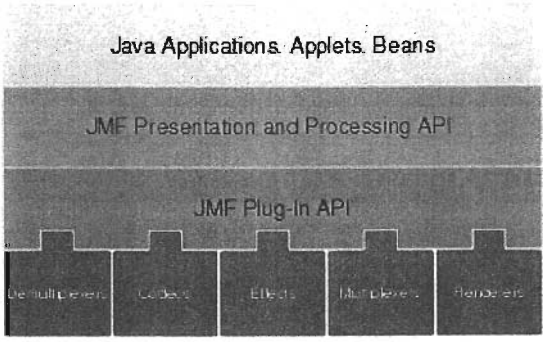
Java Media Framework (JMF) es una API que proporciona una arquitectura y un protocolo de mensajes para el manejo de adquisición, procesamiento, y entrega de datos time-based media. JMF fue diseñado para dar soporte a una gran cantidad de tipos de contenido y formatos, lo cual permite crear aplicaciones multiplataforma con alto contenido multimedia.

3.2.1. Arquitectura

Una video casetera o VCR proporciona un modelo sencillo para la grabación, procesamiento y presentación de un video. Cuando se reproduce una película usando el VCR, se proporciona el flujo de datos a través de una cinta de video, el VCR lee los datos y envía las señales producidas a un televisor y bocinas.

JMF utiliza el mismo principio. Un *data source* o *fuente de datos* encapsula el flujo de datos y un *player* o *reproductor* proporciona el procesamiento y los mecanismo de control de forma similar a un VCR. La reproducción y captura de audio y video con JMF requiere del uso de dispositivos de entrada y salida específicos, como micrófonos, cámaras, bocinas y monitores.

Las fuentes de datos y los reproductores son parte integral en el manejo de captura, presentación y procesamiento dentro de la JMF API, la cual también proporciona una API de bajo nivel que permite la integración de componentes personalizados y extensiones. Esta arquitectura permite a los desarrolladores incorporar a la JMF API de manera sencilla en sus aplicaciones mientras se mantiene la flexibilidad y crecimiento requerido para dar soporte a aplicaciones de gran contenido multimedia y nuevas tecnologías.



Arquitectura de Java Media Framework

Modelo de Tiempo

JMF mantiene el tiempo con una precisión de nanosegundos. Un momento específico en el tiempo es representado por un objeto Time. Las clases que integran el modelo de tiempo de JMF deben implementar a Clock para mantener el seguimiento de un flujo de datos en particular. La interfase Clock define las operaciones básicas de tiempo y sincronización necesarias para mantener el control sobre la presentación de los datos.

Un objeto Clock utiliza TimeBase para dar seguimiento al paso del tiempo mientras un flujo de datos es presentado. La única información que maneja TimeBase es el tiempo actual, que es llamado *time-based time* y el cual no puede ser detenido ni reiniciado.

Dentro de un objeto Clock, el tiempo que representa la posición actual del flujo de datos es llamado *media time*. El inicio del flujo de datos tiene un media time igual a cero, por otro lado el final del stream representa el máximo media time. La *duración* o *duration* del flujo de datos es el tiempo que tarda en presentarse en su totalidad. Los objetos que puedan mostrar la duración deben implementar la interfase Duration.

Para dar un seguimiento del tiempo, Clock utiliza:

- *Time Base Start Time*. El tiempo en que se inicia la presentación, proporcionado por TimeBase.

- *Media Start Time*. La posición en donde el flujo de datos iniciara la presentación.

- *Rate*. El rate es un factor de escala aplicado a TimeBase, e indica que tan rápido correo Clock en relación a éste. Por ejemplo, un rate de 1.0 representa una reproducción normal, mientras que un rate de 2.0 indica que la presentación al doble de la velocidad normal. Un rate negativo indica que Clock esta corriendo en dirección opuesta a TimeBase, lo que podría ser usado para poner una presentación en reversa.

Cuando la presentación inicia, media time es mapeado a time-based time el cual es usado para medir el paso del tiempo. Durante la presentación, el media time actual es calculado utilizando la siguiente formula:

$$MediaTime = MediaStartTime + Rate(TimeBaseTime - TimeBaseStartTime)$$

Cuando la presentación es detenida, media time también se detiene, sin embargo time-base media continua su avance. Si la presentación es reiniciada, media time es remapeada.

Managers
JMF API consiste principalmente de interfaces que definen el comportamiento y la interacción entre los objetos usados para capturar, procesar y presentar los datos. Para permitir una integración sencilla con clases existentes, la API utiliza objetos intermediarios llamados *managers*.

JMF utiliza cuatro managers:

- Manager. Encargado de la construcción de *Players*, *Processors*, *DataSources*, y *DataSinks*.
- PackageManager. Permite llevar un registro de los paquetes que contienen las clases de JMF.
- CaptureDeviceManager. Mantiene un registro de los dispositivos de captura disponibles.
- PlugInManager. Mantiene un registro de los plug-ins disponibles para JMF, como multiplexores, demultiplexores, codecs, filtros, etc.

Para escribir programas basados en JMF, es necesario utilizar los métodos proporcionados pro Manager para crear los objetos necesarios para la aplicación. Si se están capturando datos desde algún dispositivo, es necesario utilizar CaptureDeviceManeger para encontrar los dispositivos disponibles y poder tener acceso a ellos. Si se requiere tener el control sobre los procesos realizados sobre los datos, o se quiere implementar un nuevo plug-in, éstos deben ser registrados por medio de PlugInManager. Para utilizar un componente personalizado, como un Player o DataSource, se utiliza PackageManager para registrar el paquete correspondiente.

Modelo de Eventos
JMF utiliza un mecanismo de eventos para mantener a las aplicaciones informadas el estado actual de los datos, lo que les permite responder a errores como la insuficiencia de datos. Cualquier objeto JMF que necesite reportar su estado actual genera un MediaEvent. Subclases de MediaEvent se utilizan para identificar diferentes tipos de eventos. Estos objetos siguen el patrón de manejo de eventos establecido por los Java Beans.

Para cada objeto definido en la JMF API que puede arrojar eventos, también se define su correspondiente interfase especializada en capturar dicho evento (listener). Para recibir notificación cuando un MediaEvent es creado, solo se necesita implementar la interfase apropiada y registrar la clase con el método apropiado al objeto que generará el evento.

Modelo de Datos

Los reproductores en JMF generalmente utilizan DataSources para manejar la transferencia de datos. Un DataSource encapsula tanto la localización de los datos como el protocolo y el software usado para entregarlos. Una vez obtenido, este objeto no puede ser utilizado para entregar otro tipo de datos.

Un DataSource se identifica por un MediaLocator o un URL (universal resource locator). Un MediaLocator es similar a un URL y puede ser construido a partir de un URL, sin embargo, un MediaLoctor puede ser construido incluso si el manejador del protocolo correspondiente no esta instalado en el sistema.

DataSource maneja un conjunto de objetos SourceStream. Un data source estándar utiliza un arreglo de bytes como su unidad de transferencia. Un *buffer data source* utiliza al objeto Buffer como su unidad de transferencia.

Como se comento anteriormente, los flujos de datos pueden ser de dos tipos: push y pull De la misma forma JMF define dos tipos de data source:

- *Pull Data Source.* Donde el cliente inicia la transferencia y controla el flujo de los datos. JMF define dos clases que entran en esta categoría: PullDataSource y PullBufferDataSource.

- *Push Data Source.* El servidor inicia la transferencia y controla el flujo de datos. JMF define dos tipos de push data sources: PushDataSource y PushBufferDataSource.

Existen dos tipos más de datas sources: clonables (coneable) y combinados (merging).
Un data source clonable debe implementar la interfase SourceCloneable para ser usado en la creación de clones los cuales pueden ser controlados por medio del DataSource original. Un MergingDataSource puede ser usado para combinar SourceStreams de distintos DataSources en un solo DataSource, lo que permite a un conjunto de DataSources ser controlados desde un mismo objeto.

Dentro de DataSource se define un objeto Format, el cual contiene la información del formato exacto en que se encuentran los datos. JMF define dos subclases de Format: AudioFormat que describe las características para un formato de audio, y VideoFormat utilizado para encapsular la información relacionada con el video. Varias subclases más derivan de VideoFormat para describir los formatos más comunes de video, éstas incluyen:

- IndexedColorFormat

- RGBFormat

- YUVFormat
- JPEGFormat
- H261Format
- H263Format

Controles

La clase Control de la JMF API proporciona un mecanismo para establecer y obtener los atributos de un objeto. Cualquier objeto que permita el acceso a sus correspondientes objetos Control debe implementar la interfase Controls, la cual define métodos para la obtención de los objetos apropiados.

JMF define un gran número de controles que le permiten gran flexibilidad sobre el manejo de los objetos, por ejemplo:

- CachingControl. Si un objeto puede dar información sobre un proceso de descarga, entonces puede implementar esta clase de forma que una barra de progreso pueda ser mostrada al usuario.
- FrameGrabbingControl. Este control puede ser utilizado para la obtención de un frame dentro de una secuencia de video.
- FormatControl. Permite el acceso al objeto Format.
- BitRateControl. Usado para exportar la información de bit rate de un flujo de datos entrante o para controlar el bit rate al momento de la codificación.
- H263Control. Permite control sobre los parámetros del codec H.263.
- QuialityControl. Permite el control de ciertos atributos de un codec que permiten el control sobre la calidad y el uso del CPU.

En ciertos casos, los controles deben ser visibles al usuario final. Los controles implementan métodos los cuales regresan un objeto Component el cual puede ser en cualquier ventana que forme la aplicación.

3.2.2. Presentación

JMF proporciona la interfase Controller sobre la cual se definen los estados básicos y el mecanismo de control que un objeto debe llevar para controlar, presentar o capturar cualquier flujo de datos. Una serie de operaciones deben ser realizadas antes de que los datos puedan ser presentados, Controller genera una serie de MediaEvents específicos con lo que permite saber con exactitud los cambios de estado por los que pasan los objetos.

Player
Un objeto Player procesa un flujo de datos de entrada proporcionado por un objeto DataSource y lo presenta en el tiempo preciso. Un Player no proporciona ningún control sobre el procesamiento que realiza sobre los datos o como los presenta.

Un objeto Player puede estar en uno de seis estados posibles. La interfase Clock de dos estados principales: *Stopped* y *Started*. Para facilitar el manejo de los recursos, Controller divide el estado Stopped en cinco estados más: *Unrealized*, *Realizing*, *Realized*, *Prefetching*, y *Prefetched*.

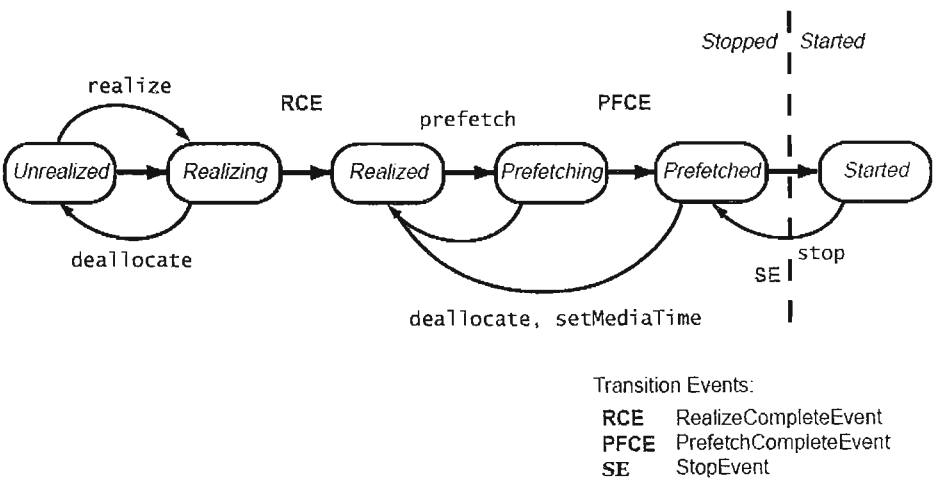


Diagrama de estados de un objeto Player.

Para alcanzar el estado Started, objeto Player debe pasar por cada unos de sus estados:

- Un Player en el estado *Unrealized* ha sido creado, sin embargo, no contiene ninguna información sobre los datos que presentara.

- Cuando se llama al método realize, Player se mueve hacia el estado *Realizing*. En éste estado Player se encuentra en el proceso de determinar los recursos necesarios que necesitará y adquiere aquellos que necesita inicialmente. Estos recursos excluyen a los de uso exclusivo (Los recursos de uso exclusivo son recursos limitados que solo pueden ser usados por un Player a la vez).

- Cuando el objeto Player termina de revisar sus recursos pasa entonces al estado *Realized*. Un Player en el estado Realized conoce los recursos que necesita y contiene información sobre el tipo de datos que presentara, por lo tanto, también puede proporcionar componentes visuales y controles.

- Cuando el método prefetch es llamado, el objeto Player se mueve hacia el estado *Prefetching* en el cual se prepara para presentar los datos. Durante esta fase el objeto carga los datos y obtiene recursos exclusivos. Es posible que en ciertas circunstancias el objeto debe regresar a este estado, por ejemplo, cuando la presentación es reposicionada.

- Cuando Player termina sus operaciones en el estado Prefetching, entonces se mueve al estado *Prefetched* en donde esta lista para ser iniciado (started).

- Llamando al método start pone al objeto Player en el estado *Started*. En éste estado el media time y time-base media son mapeados y el Clock esta corriendo.

El objeto Player genera eventos TransitionEvents cuando se mueve de un estado a otro. JMF proporciona la interfase ControllerListener lo que nos permite determinar en que estado se encuentra Player programar una apropiada respuesta.

3.2.3. Procesado

Un Processor es un tipo de Player el cual toma un objeto DataSource como su entrada, realiza ciertos procesos definidos por el usuario y da como salida los datos procesados.

Un objeto Processor puede enviar sus datos a un dispositivos para su presentación o hacia un nuevo DataSource, si éste es el caso, el DataSource puede ser usado como entrada para ser presentado por un Player, manipulado por otro Processor, o algún otro destino como un archivo local con la ayuda de un objeto DataSink.

La característica más importante de un Processor es permitir al desarrollador de la aplicación definir la clase de operaciones que serán realizadas a los datos. Esto permite e uso de efectos, mezclas y compositing en tiempo real.

El procesado de los datos se divide en varias fases:

- Demultiplexado. El proceso en el cual se extraen los track individuales que componen el flujo de datos.
- Pre-procesado. Es el proceso de aplicar ciertos algoritmos a los datos de cada track extraído.
- Transcoding. Es el proceso de convertir los datos de cada track de un formato a otro. Cuando los datos son convertidos de un formato raw a un formato comprimido se le llama encodig o codificación y al proceso inverso se le conoce como decoding o decodificación.
- Post-procesado. Es el proceso de aplicar ciertos algoritmos a los datos de un track decodificado.
- Multiplexado. Es el proceso de entrelazar los tracks disponibles en un solo fujo de datos.
- Rendering. Es el proceso que se realiza para presentar los datos al usuario.

Cada fase del procesamiento se realiza por un componente separado. Estos componentes son llamados JMF *plug-ins*. JMF define cinco tipos de plug-ins: Demultiplexer, Effect, Codec, Multiplexer y Rederer.

Un objeto Processor, puede estar dentro de los seis estados definidos por Player, sin embargo, Processor agrega dos estados más: *Configuring* y *Configured*.

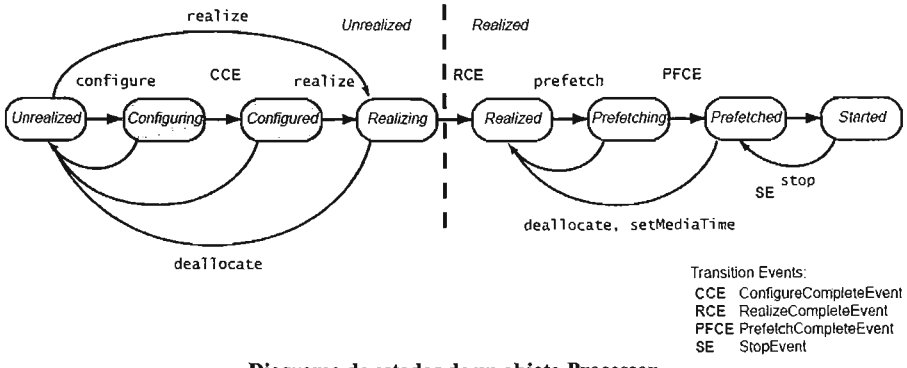


Diagrama de estados de un objeto Processor.

- Un Processor entra al estado de *Configuring* cuando el método configure es llamado. Mientras se encuentra en este estado se realizan el proceso de demultiplexado y se accede a información que contiene el formato de los datos.

- Cuando las acciones dentro del estado *Configuring* terminan el procesador pasa al estado *Configured* generando un evento ConfigureCompleteEvent.

- Llamando al método realize el Processor es llevado al estado *Realized* donde se dice que el objeto esta totalmente construido.

Cuando un Processor se encuentra en su estado *Configured*, el método getTrackControls puede se invocado de forma que se pueda seleccionar las operaciones específicas de procesado que se realizaran a cada track.

No todos los objetos Processor dan una salida a sus datos, en su lugar pueden presentar los datos al usuario de la misma forma que un Player. Por este motivo un Processor puede ser visto como un Player configurable.

3.2.4. Captura

Un dispositivo de captura multimedia puede servir como una fuente de datos multimedia, ejemplos de estos dispositivos pueden ser un micrófono, una tarjeta de captura de video o una cámara digital. JMF permite el uso de la clase DataSource, como una abstracción de cualquier dispositivo de captura. Un programador puede fácilmente extender de DataSource o cualquiera de sus subclases e implementar su propia solución al dispositivo de captura presente.

3.2.5. Almacenamiento y Transmisión de Datos

La clase DataSink es usada para leer un flujo de datos de un objeto DataSource y enviarlos hacia algún destino diferente de cualquier dispositivo de presentación. Un determinado objeto DataSink puede ser usado para escribir datos a un archivo local, escribir los datos a través de la red, o enviar los datos por la red utilizado el protocolo RTP (Real-Time Transport Protocol) lo que permite crear aplicaciones para videoconferencias.

De la misma forma que Player, los objetos DataSink son construidos por medio de Manager usando un DataSource como fuente de datos.

Siguiendo el esquema de eventos de JMF, DataSink genera eventos DataSinkEvent para reportar su estado. Un DataSink puede generar un

DataSinkEvent proporcionando un código de error, o puede utilizar una de las dos subclases existentes:

- DataSinkErrorEvent. Indica que ocurrió un error mientras DataSink estaba escribiendo los datos.

- EndOfStreamEvent. Indica que todo el flujo de datos fue escrito satisfactoriamente.

Para responder a estos eventos, es necesario implementar la interfase DataSinkListener.

CAPITULO 4

DESARROLLO DE APLICACIONES MÓVILES

4.1. MIDlets Suite

Dentro del ambiente de desarrollo de J2ME, específicamente en el uso de la configuración CLDC y el perfil MIDP, las aplicaciones desarrolladas son conocidas como MIDlets. Dentro de cada dispositivo habilitado con J2ME se encuentra una aplicación llamada AMS (Application Management Software) encargada de administrar la descarga y ciclo de vida de los MIDlets. El AMS provee el ambiente de ejecución para un MIDlet, verificando la seguridad, permisos, los diferentes estados de ejecución, así como, proporcionar las clases del sistema.

Los componentes básicos que forman cualquier MIDlet suite para su distribución son el *Java Application Descriptor* (JAD) y *Java Archive* (JAR). Juntos, estos dos archivos forman un MIDlet suite.

El archivo JAD describe al MIDlet suite. La descripción incluye el nombre, la localización y el tamaño de archivo JAR, y los requerimientos que necesita tanto de configuración como de perfil. Este archivo también puede contener otros atributos definidos por MIDP, el desarrollador o ambos. Los atributos que inicien con las palabras *MIDlet-* o *MicroEdition-* se encuentran reservadas para uso del AMS. La sintaxis del archivo JAD es similar a la proporcionada por la clase *java.util.Properties* encontrada en el ambiente J2SE, esto es, cada atributo se define por el nombre del atributo seguido de dos puntos y el valor asignado.

En los dispositivos, un MIDlet suite es identificado por un conjunto de atributos (*MIDlet-Name*, *MIDlet.-Version*, *MIDlet-Vendor*). El archivo JAR será descargado para su instalación de la dirección especificada por *MIDlet-Jar-URL*. El tamaño del archivo a descargar debe coincidir con aquel definido en el atributo *MIDlet-Jar-Size*.

El archivo JAR contiene uno o más MIDlets, así como algunos otros recursos utilizados por éstos. Cada MIDlet es definido por el atributo *MIDlet-<n>* cuya sintaxis es la siguiente:

MIDlet-n : MIDletName, [IconPathname], ClassName

Donde:

- *n* es el número del MIDlet, iniciando en 1 e incrementándose en uno por cada MIDlet.

- *MIDletName* es el nombre del MIDlet que será visible al usuario.

- *IconPathname* es la ruta absoluta de una imagen en formato PNG dentro del archivo JAR que es asignada a la aplicación dentro del sistema.

- *ClassName* es el nombre de la clase inicial del MIDlet que extiende de *javax.microedition.midlet.MIDlet* y contiene un constructor público que no tiene argumentos.

El archivo manifiesto (manifest file), localizado en */META-INF/MANIFEST.MF* de cualquier archivo JAR, tiene la misma sintaxis que el archivo JAD y puede compartir los mismos atributos. El archivo manifiesto es un componente esencial para el modelo de MIDlets firmados en MIDP 2.0. En un MIDlet suite firmado, los atributos del archivo JAD debe coincidir con los del manifiesto, lo cual es una de las formas de verificar la integridad del archivo JAR.

Además de las clases Java y el archivo manifiesto, un archivo JAR, como se menciono anteriormente, puede contener otros recursos. Estos pueden ser imágenes que el MIDlet puede cargar con ayuda del método *javax.microedition.lcdui.Image.createImage(String)*. Es posible también acceder a cualquier tipo de recurso en el archivo JAR utilizando el método *java.lang.Class.getResourceAsStream(String)*, lo que regresa un objeto de tipo *java.io.InputStream* desde el cual pueden leerse los bytes del recurso solicitado.

4.2. Modelo de Seguridad

El modelo de seguridad en J2ME es una versión reducida del modelo de J2SE. Este ha sido adaptado para trabajar en las limitaciones encontradas comúnmente en los dispositivos habilitados para J2ME.

Las reglas son:

- Los archivos de cada clase son verificados antes de su inclusión al archivo JAR.
- Existe un número limitado de APIs: la configuración Connected Limited Device Configuration, uno o más perfiles y ciertas clases autorizadas.
- La descarga y el manejo de las aplicaciones Java son llevadas a cabo por código nativo dentro de la máquina virtual de Java. No existen cargadores de clases definidos por el usuario.
- No existe ninguna implementación de *Java Native Interface* (JNI) y ninguna extensión es permitida.
- Las clases del sistema no pueden ser sobrescritas.

Todas estas reglas garantizan la integridad del ambiente de ejecución de manera que no hay forma de modificarlo o salir de él (*Sand box*).

4.3. Instalación

Existen dos maneras de instalar un MIDlet suite. La primera, llamada directa, involucra cierta conexión entre el dispositivo y la plataforma de desarrollo. Esta conexión puede ser a través de cable serial, conexión infrarroja o un enlace Bluetooth, donde el fabricante del dispositivo puede proporcionar algunas herramientas de software para facilitar la instalación de aplicaciones, el software Nokia PC Suite es un ejemplo de ello. Generalmente, la instalación directa es usada por los desarrolladores para probar sus aplicaciones directamente sobre sus dispositivos, sin embargo, esta opción resulta impráctica cuando la aplicación desea ser distribuida hacia miles o millones de usuarios finales.

La forma más común de instalación de aplicaciones J2ME es por medio de una distribución *Over-the-air* (OTA). Un dispositivo puede instalar un MIDlet suite desde un servidor remoto usando el navegador incluido en mismo dispositivo. La instalación simplemente se inicia accediendo al URL del archivo JAD correspondiente a la aplicación, en general, la instalación se lleva a cabo en los siguientes pasos:

1. El dispositivo cliente manda una petición HTTP GET al servidor con el URL proporcionado.
2. El servidor responde enviando el archivo JAD correspondiente a la aplicación.
3. El cliente verifica la respuesta del servidor y extrae los atributos MIDlet-Jar-File y MIDlet-Jar-Size.
4. El dispositivo cliente envía una petición HTTP GET para obtener el archivo JAR.
5. El servidor responde enviando el archivo JAR solicitado.
6. El dispositivo realiza diferentes tareas de verificación como MIDlets firmados, permisos de uso de clases, entradas al registro push, así como, la integridad misma del archivo JAR
7. El dispositivo solicita permiso al usuario para confirmar la instalación.

Cualquier falla presentada en estos pasos provoca un error de instalación de la aplicación. Otros factores que pueden impedir el término del proceso de instalación son el correcto funcionamiento del servidor Web y el navegador del dispositivo, fallas en la red, así como límites de tamaño impuesto en la red para los archivos JAR.

4.4. Desinstalación.

Debido a que un MIDlet suite es una entidad auto contenida, su eliminación del sistema es sencilla. La mayoría de los dispositivos permiten al usuario seleccionar el MIDlet suite proporcionando una opción de Eliminar. En este punto el dispositivo solicita una confirmación, en donde una respuesta afirmativa remueve por completo el MIDlet suite, incluyendo las entradas al registro y los Record Stores creados por la aplicación.

4.5. Ciclo de Vida

La siguiente figura muestra los diferentes estados en los que se puede encontrar un MIDlet.

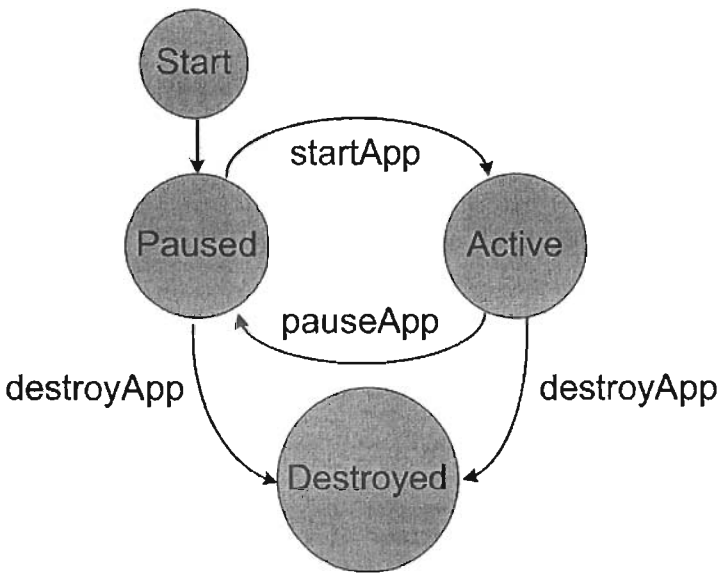


Diagrama de estados de un MIDlet.

Cuando un MIDlet comienza su ejecución, el AMS primero llama al constructor de la clase para crear una nueva instancia de éste. Típicamente el constructor hace poca o nula inicialización. Los diferentes estados permiten un control sobre el manejo de recursos que la aplicación puede necesitar. Cuando el constructor regresa, el AMS coloca al MIDlet en su estado *Paused* (Pausa). Para cambiar al MIDlet a su estado *Active* (Activo) el AMS llama al método *midlet.startApp( )*. De igual forma la transición del estado Activo al de Pausa se lleva a cabo cuando el AMS llama al método *midlet.pauseApp( )*. Un MIDlet en su estado de Pausa no ha sido terminado, sin embargo debería liberar cualquier recurso obtenido en el método *startApp( )*.

Los métodos de transición permiten un control efectivo sobre el manejo de recursos. Típicamente, el método *startApp( )* se utiliza para iniciar recursos como Record Stores, conexiones de red, componentes de interfaz de usuario; de forma contraria, el método *pauseApp( )* es usado para la liberación de éstos recursos.

Un MIDlet puede entrar en el estado *Destroyed* (Destruido) desde cualquiera de los estados de Pausa o Actividad por medio de una llamada al método *midlet.destroyApp( )*. Dentro de este método se libera cualquier recurso obtenido en el constructor y se guarda cualquier información requerida.

Existen ciertas variaciones en las cuales una aplicación puede entrar en cualquiera de los tres estados (Paused/Active/Destroyed). Voluntariamente un MIDlet puede entrar a un estado de Pausa llamando al método *midlet.notifyPaused( )*, de manera similar, una llamada al método *midlet.notifyDestroyed( )* informa al AMS de que la aplicación puede ser considerada para ser terminada (estado Destroyed). Finalmente, el método *destroyApp( )* es llamado con un argumento de tipo boolean. Si el argumento es verdadero, el MIDlet estará en su estado Destruido cuando el método *destroyApp( )* regrese. De lo contrario, si el argumento es falso el MIDlet puede mandar una petición por medio de un *MIDletStateChangeException* para no entrar en su estado de Destruido. Posteriormente, una llamada al método *midlet.resumeRequest( )* le dirá al AMS que el MIDlet esta interesado en entrar a su estado Activo. Es importante aclarar que la llamada a estos métodos son solo peticiones al AMS el cual decidirá dependiendo de sus recursos si permite a la aplicación moverse de un estado a otro.

Uno de los aspectos más importantes relacionados al ciclo de vida de un MIDlet se refiere a cuando son llamados estos métodos para cambiar de un estado a otro. La especificación de MIDP es intencionalmente vaga en este aspecto, debido a que ambiente de ejecución puede tener sus respectivas variantes. Por ejemplo, en uno de los casos más comunes tenemos un teléfono celular el cual puede recibir una llamada o una conexión entrante de SMS. El AMS, por lo general, colocará al MIDlet en su estado de Pausa para liberar recursos necesarios para manejar las conexiones entrantes. Si la aplicación en su estado de pausa no libero suficientes recursos, entonces el AMS puede tomar la decisión de llamar al método *destroyApp( )* a manera de liberar más.

4.6. Aplicación ¡Hola Mundo!

Para ejemplificar el desarrollo de las aplicaciones para la plataforma J2ME se muestra la versión de la ya famosa aplicación de ¡Hola Mundo!

El siguiente código fue generado con la ayuda de Netbeans IDE 4.0 y el suplemento Mobility Pack.

```
/*
 * HelloMIDlet.java
 *
 * Created on 7 de marzo de 2005, 10:16 AM
 */
package hello;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

/**
 * An example MIDlet with simple "Hello" text and an Exit command.
 * Refer to the startApp, pauseApp, and destroyApp methods so see how each handles the requested
 * transition.
 *
 * @author  David
 * @version 1.0
 */
public class HelloMIDlet extends MIDlet implements CommandListener {

    private Command exitCommand; // The exit command
    private Display display;    // The display for this MIDlet

    public HelloMIDlet() {
        display = Display.getDisplay(this);
        exitCommand = new Command("Exit", Command.SCREEN, 2);
    }

    /**
     * Start up the Hello MIDlet by creating the TextBox and associating the exit command and listener.
     */
    public void startApp() {
        TextBox t = new TextBox("Hello MIDlet", "¡¡Hola Mundo!!", 256, 0);

        t.addCommand(exitCommand);
        t.setCommandListener(this);

        display.setCurrent(t);
    }

    /**
     * Pause is a no-op since there are no background activities or record stores that need to be closed.
     */
    public void pauseApp() {
    }

    /**
     * Destroy must cleanup everything not handled by the garbage collector.
     * In this case there is nothing to cleanup.
     */
    public void destroyApp(boolean unconditional) {
    }

    /*
     * Respond to commands, including exit. On the exit command, cleanup and notify that the MIDlet
```

```
    * has been destroyed.
    */
    public void commandAction(Command c, Displayable s) {
        if (c == exitCommand) {
            destroyApp(false);
            notifyDestroyed();
        }
    }
}
```

Como se puede observar en la declaración de la clase, *HelloMIDlet* extiende de la clase MIDlet definida en el paquete *javax.microedition.midlet*, esto le permite heredar los métodos que definen el ciclo de vida de cualquier MIDlet. Adicionalmente, HelloMIDlet implementa la interfase *CommandListener* (javax.microedition.lcdui.CommandListener) la cual permite responder a comandos registrados añadiendo un nivel de interacción con el usuario.

Dentro del constructor de la clase se inicializan dos objetos utilizados a lo largo de la aplicación, el más relevante de ellos es el objeto *display* instanciado a partir del método *Display.getDisplay(this)* la cual regresa una referencia de la clase *Display*. Este objeto representa el administrador encargado del despliegue gráfico y los dispositivos de entrada en el sistema. Solo puede existir una copia de la clase *Display* por aplicación.

El método *startApp( )* inicializa un objeto de tipo *TextBox,* el cual es un objeto gráfico que permite el despliegue y captura de texto en la pantalla del dispositivo, es aquí donde se mostrará el ya famoso mensaje de "¡Hola Mundo!". Una vez creado el objeto que presentará el texto en pantalla, es necesario, establecer los componentes que permitirán la interacción con el usuario, este componente es un objeto de la clase Command.

El método *t.addCommand(exitCommand)* agrega al objeto *exitCommand* (previamente creado en el constructor) de forma que se presente en conjunto con el objeto *TextBox,* sin embargo, cualquier evento generado sobre el objeto *exitCommand* no generará ninguna acción hasta definir un objeto del tipo *CommandListener* que atienda dichas peticiones. Este objeto es el propio MIDlet registrado a partir del método *t.addCommandListener(this)*.

Finalmente después de la construcción y registro de objetos necesarios para el funcionamiento del programa, solo resta mostrar al usuario los componentes gráficos creados. Esto se logra por medio del método *display.setCurrent(t)*.

El último método que vale mencionar en este MIDlet es *commandAction(Command c, Displayable s)*. El cual se hereda de la interfaz CommandListener y se sobrescribe para dar la funcionalidad requerida, en este caso la de salir del programa, llamando para ello a los métodos destroyApp(false) y notifyDestroyed( ) mencionados anteriormente.

El contenido del archivo JAD puede verse en la siguiente tabla

```
MIDlet-1: HelloMIDlet, , hello.HelloMIDlet
MIDlet-Jar-Size: 1427
MIDlet-Jar-URL: HolaMundo.jar
MIDlet-Name: HolaMundo
MIDlet-Vendor: Vendor
MIDlet-Version: 1.0
MicroEdition-Configuration: CLDC-1.0
MicroEdition-Profile: MIDP-1.0
```

Como puede apreciarse, se define un MIDlet de nombre *HelloMIDlet* sin ninguna imagen definida y el cual es ejecutado mediante la clase *hello.HelloMIDlet*.

El valor del atributo *MIDlet-Name* así como el tamaño y el URL del archivo JAR definen claramente al MIDlet suite. Además encontramos otros atributos como la versión y el nombre del vendedor de la aplicación.

Por último es posible ver los requerimientos tanto de configuración como de perfil que este MIDlet suite necesita para su instalación.

Tanto Wireless Toolkit 2.2 como el Mobility Pack de Netbeans IDE 4.0 proporcionan una serie de emuladores que permiten ejecutar el MIDlet como si estuviera corriendo en el dispositivo real, esto permite a los desarrolladores probar sus aplicaciones antes de distribuirlas a los usuarios finales.

Las siguientes figuras muestran una extensión de estos emuladores proporcionada por Sony Ericsson diseñada para sus propios modelos de teléfonos celulares. En este caso en particular la aplicación esta corriendo en el emulador para el modelo K700.

Aplicación ¡Hola mundo! Corriendo en el emulador para el télefono SonyEricsson K700.

CAPITULO 5

ANÁLISIS Y DESARROLLO DE UN CASO PRÁCTICO

5.1. Descripción del Problema

A continuación se describe una aplicación funcional creada para este trabajo con el fin de demostrar las capacidades de la Tecnología Java en el uso de sistemas de video vigilancia, resaltando de igual manera la integración y escalabilidad que presenta el uso de dispositivos móviles en este tipo de sistemas.

El propósito del proyecto es incorporar dentro de una casa o un pequeño negocio, un sistema de vigilancia basado en PC con las siguientes características:

- Sistema multiplataforma.
- Soporte para cámaras locales y/o remotas.
- Visualización de una o más cámaras simultáneamente.
- Grabación continúa de imágenes.
- Grabación de video.
- Detección de movimiento.
- Notificación de eventos por medio de correo electrónico.
- Acceso remoto al sistema y cámaras por teléfono celular.

5.2. Elementos de un sistema multiplataforma

5.2.1. Hardware

Para que un sistema sea multiplataforma, es necesario, que no exista una dependencia del hardware donde se instalará, sin embargo, los sistemas tradicionales basados en computadoras personales incorporan una tarjeta con requerimientos muy específicos dedicada a recibir las señales de cámaras analógicas tradicionales.

Para evitar las dependencias de hardware es necesario utilizar equipos que adopten estándares internacionales para comunicarse. Las redes Ethernet proporcionan un ambiente adecuado de comunicación entre diferentes equipos.

Las cámaras de red o Network Cameras se adaptan perfectamente a los sistemas multiplataforma, proporcionando un pequeño servidor integrado dentro de la cámara, lo que nos permite utilizar cualquier red Ethernet para transferir las imágenes capturadas, directamente al sistema.
El uso de estas tecnologías permite una integración con posibles redes existentes o la creación de una red local de bajo costo, donde el crecimiento del sistema al agregar nuevas cámaras se realice de manera sencilla o incluso adoptando nuevas tecnologías como las redes inalámbricas.

5.2.2. Software

Actualmente, las cámaras de red han tenido un crecimiento muy rápido debido a su principal característica que permite ver las imágenes capturadas desde cualquier navegador de Internet, ya sea utilizando componentes ActiveX o Applets de Java; sin embargo, estas aplicaciones son muy limitadas cuando se requiere de un sistema de vigilancia.

Para cubrir las necesidades de sistemas de vigilancia más complejos, algunos fabricantes de cámaras de red ofrecen sistemas dedicados a éste propósito en los cuales encontramos algunas desventajas importantes.

El primer inconveniente presente en estos sistemas es que no son multiplataforma, en general su único soporte es bajo el sistema operativo Windows y en muy pocos casos existe una versión para Linux creada por terceros.

Un segundo problema de estos sistemas es que son diseñados para trabajar con cámaras analógicas o cámaras de red del mismo fabricante lo que limita sus capacidades de crecimiento. Para mantener la flexibilidad del sistema, éste debe aprovechar las ventajas de las cámaras de red que como parte de su funcionamiento proporcionan un servidor integrado de manera que el video se obtenga de manera similar en equipos de diferentes fabricantes. Desafortunadamente, ésta tecnología aún se encuentra en sus primeros pasos y no existen estándares definidos a los cual apegarse.

Para resolver estos problemas se propone utilizar la Tecnología Java lo que permite la creación de un sistema multiplataforma el cual será capas de obtener imágenes o video de cámaras de diferentes fabricantes.

5.3. Configuración física del sistema

Las características del sistema permiten diversas implementaciones en el uso del equipo o la configuración de la red las cuales producen resultados similares, sin embargo, y debido a los alcances de éste escrito solo se menciona una de ellas.

El sistema propuesto requiere de los siguientes elementos:

- *Cámaras de red.* Las cámaras son la parte más importante del sistema y sus características dependen de el uso específico que se le de a la aplicación. Existen cámaras para cubrir las necesidades de cualquier situación: cámaras para interiores (indoor) o exteriores (outdoor), cámaras con visión nocturna (IR), cámaras inalámbricas, cámaras con movimiento (PTZ).

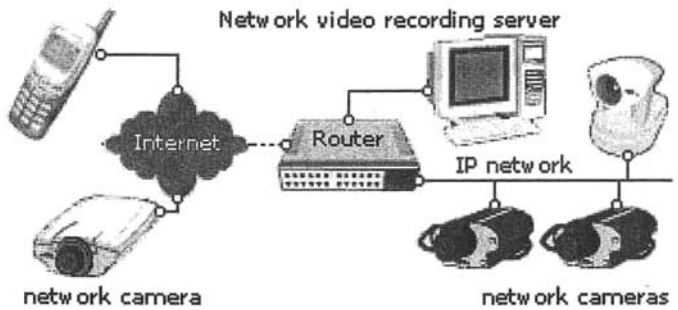
- *Computadora de escritorio.* Este equipo será donde se hospedará el sistema principal, por lo cual, se requiere que tenga cierta capacidad de procesamiento y memoria aunque sus características dependen de diversos factores como son: el número de cámaras a utilizar, la calidad de imagen, el despliegue simultáneo del video, etc. Sin embargo no hay restricciones en cuanto a la plataforma o el sistema operativo que utiliza, siempre y cuando, cuente con soporte para la Tecnología Java.

- *Conexión a Internet de Banda Ancha.* El sistema requiere de una conexión permanente a Internet para el envío de correo electrónico y para su acceso remoto.

- *Red local (LAN).* La comunicación de las cámaras y el sistema se realiza a través de una red local (aunque es posible comunicarse con cámaras de otras redes, el desempeño de estas puede variar considerablemente). La administración de la red se realiza por medio de un Ruteador (Router), el cual también permite la salida hacia Internet.

- *Teléfono celular.* El acceso remoto al sistema requiere de un teléfono celular con acceso a una red de datos, habilitado para correr aplicaciones Java (CLDC 1.0 y MIDP 1.0) y espacio suficiente para la instalación de la aplicación.

La siguiente figura ilustra de forma esquemática la configuración donde se integraran todos estos elementos.



Configuración física del sistema.

El sistema principal se hospeda dentro de una computadora localizada dentro de la red proporcionada por el Router, para lograr las mejores condiciones en el sistema, las cámaras se encuentran conectadas a la misma red permitiendo con ello una transferencia de datos lo más rápida posible, sin embargo, no existe impedimento alguno de que el sistema mantenga una conexión con cámaras publicas accesibles por Internet.

El acceso remoto al sistema se logra por medio de configuraciones en el Router que posibilita el mapeo de puertos permitiendo que una conexión entrante al Router sea direccionada al mismo puerto de un equipo dentro de la red.

En la instalación de un sistema de vigilancia se deben tener muchas consideraciones, como la colocación y el tipo de las cámaras que se utilizaran (interiores, exteriores, nocturnas, etc.), así como el equipo donde se hospeda el sistema, tiempos de grabación y resolución de la imagen, capacidad de almacenamiento para el historial de imágenes y/o videos. Todos estos factores dependen directamente del uso exacto al cual este destinado.

5.4. Funcionamiento del sistema

Desde el punto de vista del software, el sistema se compone de tres aplicaciones:

- Nets Desktop
- Nets Server
- MicroNets

A continuación se describe el funcionamiento de cada uno de estos componentes.

5.4.1. Nets Desktop

Es la aplicación principal del sistema encargada de la administración y visualización de las cámaras registradas, así como, de los eventos asociados a cada una de ellas.

Antes de poder mostrar el video de las cámaras, el sistema inicia varios procesos descritos a continuación:

1. Carga la configuración del sistema. En este paso el sistema buscará su archivo principal de configuración el cual enlista todos los directorios utilizados a lo largo de su ejecución. Este archivo también contiene información sobre la configuración de las conexiones fuera del sistema.

2. Obtener la configuración de las cámaras. El sistema permite registrar cualquier número de cámaras, cada una de las cuales tiene su propia configuración. La información almacenada por cada cámara permite obtener datos tanto para su identificación y funcionamiento en el sistema, algunos de estos datos incluyen: el URL o dirección IP de la cámara, la resolución de la imagen en píxeles, la marca y modelo de la cámara, el estado de la cámara (activa o inactiva, visible o no visible), así como la acción que debe realizar.

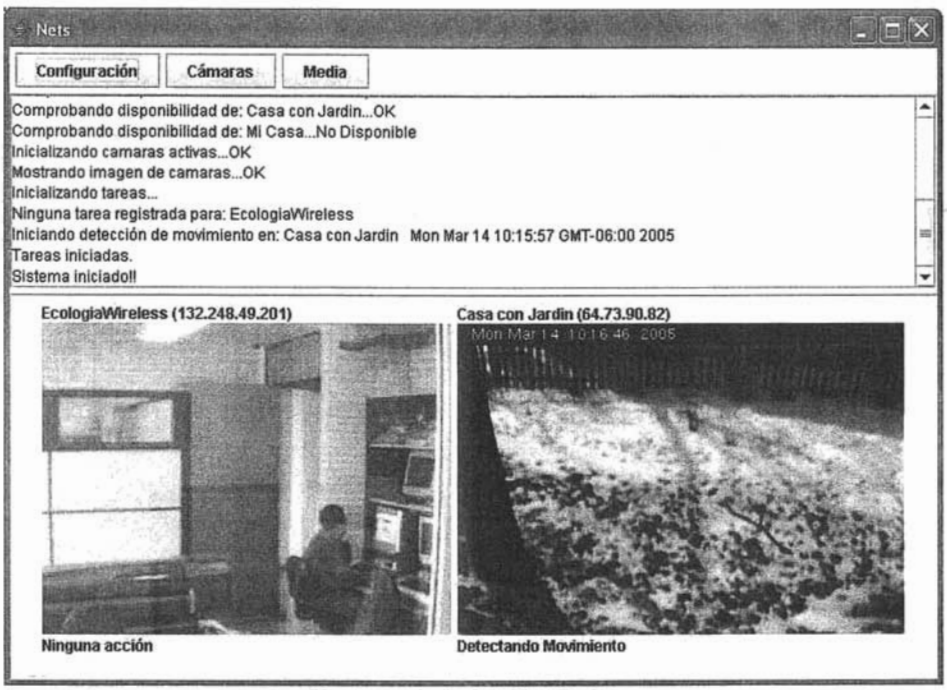
3. Iniciar conexiones. Una vez obtenidos los parámetros del sistema y para cada una de las cámaras registradas debe establecerse la comunicación con cada cámara activa en el sistema de forma que pueda comprobarse que la cámara se encuentra funcionando correctamente. Ya realizada ésta comprobación se inicializan todos los objetos necesarios que mantienen una conexión constante para obtener el flujo de video proporcionado por cada cámara.

4. Iniciar las tareas registradas a cada cámara. El sistema permite asociar ciertas tareas a cada una de las cámaras, de forma que el video recibido pueda ser almacenado. Esto es posible en dos formas. La primera permite almacenar una imagen de la cámara cada cierto periodo de tiempo (típicamente un par de segundos); la segunda opción analiza el video en tiempo real para detectar cualquier movimiento presente con la finalidad de iniciar una grabación de buena calidad y/o el envío de notificaciones por correo electrónico.

5. Presentar el video en pantalla. Con las conexiones y las tareas iniciadas por cada cámara solo resta, obtener los recursos del sistema que permitan la visualización en pantalla.

6. Inicio de otras tareas. Finalmente, el sistema iniciará otras tareas no relacionadas directamente con las cámaras. Tal es el caso del servidor para conexiones remotas.

La interfaz gráfica del sistema esta formada por una ventana principal, integrada por dos paneles; un panel de información donde se muestra detalles de la actividad realizada por el sistema a forma de bitácora, y un segundo panel compuesto por subpaneles que presentan el video, además de cierta información descriptiva de cada una de las cámaras que incluye el nombre, su dirección IP acompañada de la acción que se realiza.



Ventana principal de Nets Desktop.

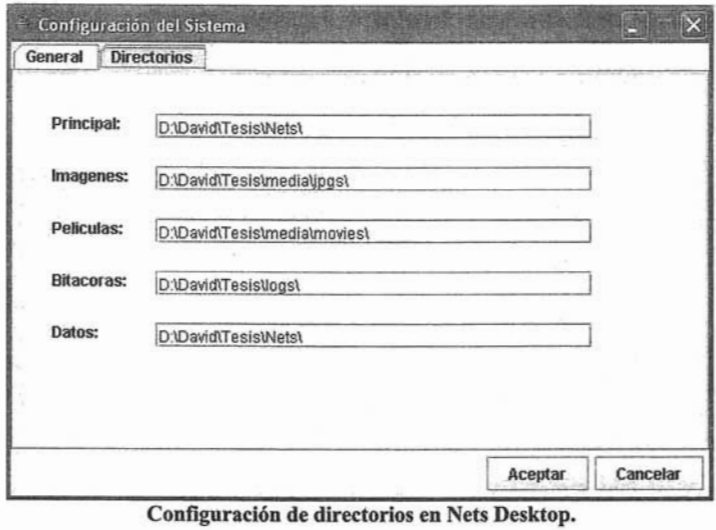
Desde esta ventana puede también apreciarse la existencia de 3 botones en la parte superior que permite acceder a la configuración del sistema (*Configuración*), la configuración de las cámaras (*Cámaras*) y un reproductor de medios (*Media*).

La ventana de configuración del sistema se encuentra formada por dos secciones:

En la sección General pueden configurase opciones como la dirección de correo electrónico que recibirá notificaciones del sistema, así como el servidor de aplicaciones móviles para acceso remoto.

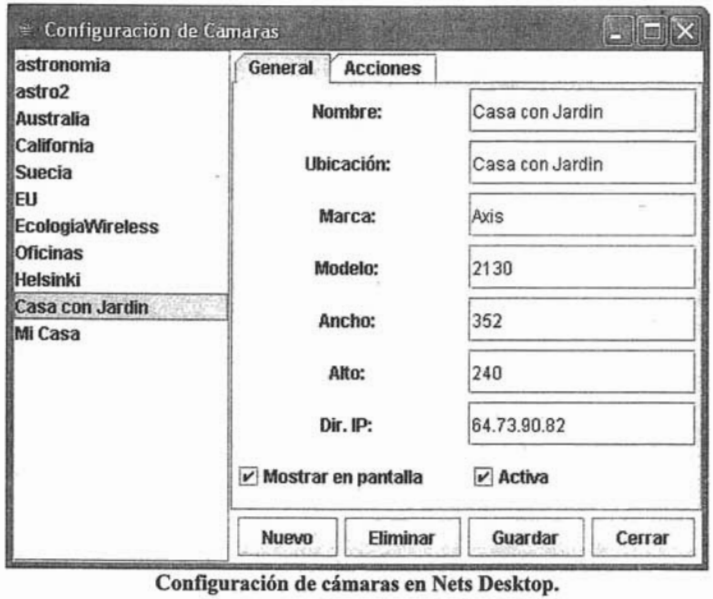
Configuración general de Nets Desktop.

En la segunda sección de esta ventana se encuentran los directorios que el sistema utiliza para los diferentes archivos que maneja ya se de configuración o archivos de imágenes y videos.



Configuración de directorios en Nets Desktop.

La ventana de configuración de las cámaras esta formada por una lista de todas las cámaras registradas en el sistema, sin importar si se encuentran en funcionamiento o no.



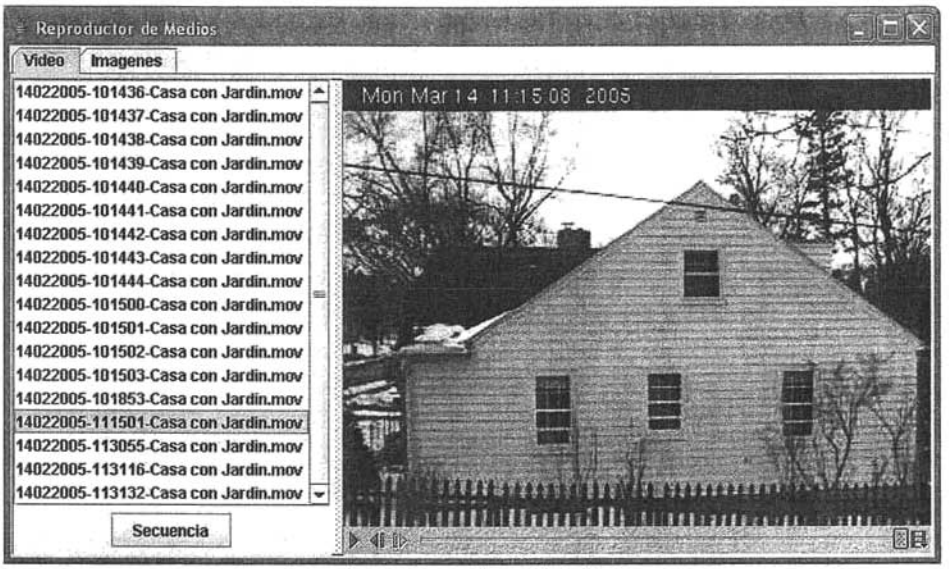
Configuración de cámaras en Nets Desktop.

Al seleccionar cualquier cámara de la lista se muestra la información relacionada incluyendo el nombre, ubicación, marca, modelo, dimensiones en pixeles, y su dirección IP. Todas estas opciones pueden ser editadas directamente y surtirán efecto la próxima vez que el sistema sea iniciado.

El tercer botón en la ventana principal (Media) permite acceder a una herramienta que facilita la visualización de las imágenes y videos capturados. La ventana de reproducción de medios proporciona una lista completa de los videos e imágenes de forma separada.

Como puede apreciarse en la imagen, los nombres de cada uno de los archivos se forman de la fecha y hora en que fueron grabados, así como el nombre de la cámara que lo capturo, esto permite la localización de un momento en especifico de una manera más sencilla y rápida.

Ya que es posible almacenar una gran cantidad de imágenes y videos, esta ventana facilita, por medio del botón *Secuencia*, una visualización consecutiva de todos los archivos. En el caso de los videos, éstos se reproducen uno tras otro hasta alcanzar el último de la lista; por su parte cada imagen es presentada por un par de segundos antes de mostrar la siguiente en la lista.



Reproductor de medios en Nets Desktop.

5.4.2. Nets Server

Una de las características más importantes del sistema propuesto en el presente trabajo, es su capacidad para monitoreo remoto, esto es que a través de un programa cliente se puede acceder a cierta información del sistema o aún más importante se puede obtener imágenes capturadas por las cámaras registradas.

Nets Server se compone de una serie de servlets que se encuentran funcionando sobre cualquier servidor Web con soporte para la plataforma J2EE. Utilizando el protocolo HTTP, el cliente sigue el esquema de petición y respuesta (request-response), en donde se envía una petición por medio del método GET al servlet correspondiente a la información especifica que se solicite, el servidor entonces procesará la información entrante abrirá las conexiones necesarias con el software Nets Desktop, el cual validará la conexión para posteriormente generar la información necesaria y regresarla al servidor Web quien se encargará de entregarle la respuesta al programa cliente.

La siguiente figura ilustra el flujo de información en este proceso.

5.4.3. MicroNets

El tercer y último componente del sistema es un cliente para acceso remoto, esto es utilizando el modelo de conexión a través de Nets Server, el cual no presenta ninguna restricción en cuanto al tipo de cliente que se conectará, esto es, los clientes pueden ser un Applet, una aplicación independiente, o incluso una aplicación desarrollada por otra tecnología diferente de Java.

Actualmente, la telefonía celular ha tenido un gran crecimiento en todo el mundo adoptando nuevos estándares para mejorar la calidad de las comunicaciones y aumentar la velocidad en la transferencia de datos, aunado a esto, cada día salen al mercado nuevos dispositivos con más y mejores capacidades como una mayor cantidad de memoria para almacenar información, pantallas de mayor resolución con profundidad de color, soporte para más protocolos de red y nuevas tecnologías incluyendo soporte para aplicaciones sobre la plataforma J2ME.

Aprovechando las ventajas que proporciona la telefonía móvil se ha creado un cliente llamado *MicroNets*, el cual permite un monitoreo remoto del sistema de vigilancia, desde cualquier dispositivo habilitado con la configuración CLDC 1.0 y el perfil MIDP 1.0 de J2ME.

Esta aplicación cuenta con un menú principal desde donde puede conectarse al sistema remoto, enviar comandos a ejecutarse o configurar las opciones necesarias para el servicio.

Menú principal.

La opción de *Configurar* permite establecer información de dos tipos:

Conexión

- URL. Dirección del equipo donde se hospeda el servicio de Nets Server.

- URL2. Este campo indica el equipo donde se encuentra el sistema Nets Desktop. El sistema esta diseñado para mantener tanto a Nets Server como a Nets Desktop en equipos diferentes, sin embargo, nada impide que ambas aplicaciones se ejecuten en el mismo equipo al mismo tiempo, en este caso el valor más recomendable para URL2 es *localhost*.

- Puerto. El puerto especifico referido por Nets Desktop para su servidor de aplicaciones móviles.

Imagen

- Formato. En el Java Specification Request 37 (MIDP 1.0) se define al formato PNG (Portable Network Graphics) como el estándar para el manejo de imágenes dentro de cualquier aplicación J2ME, sin embargo, en ocasiones este formato no es adecuado cuando se requiere de imágenes con calidad fotográfica, como es el caso de un sistema de vigilancia, por este motivo algunos equipos proveen un soporte para la decodificación de imágenes en el formato JPEG. Los dispositivos con soporte para la especificación JTWI, el uso de este formato es obligatorio. Debido a que no siempre es posible saber si el dispositivo cuenta con soporte para uno u otro formato, el sistema permite la codificación de la imagen en formato PNG o JPEG, de forma que MicroNets pueda mostrar la imagen en cualquier equipo. El campo formato debe entonces tomar el valor *png* o *jpg*.

- Alto y Ancho de la Imagen. Nuevamente debido a la fragmentación, es decir, las diferencias entre los distintos equipos donde se puede ejecutar la aplicación, se hace necesario poder especificar al sistema las dimensiones máximas que requerimos de la imagen para su mejor presentación, esto se hace a través de los campos de *Alto Imagen* y *Ancho Imagen,* donde ambos están definidos en pixeles. Aunque es posible indicar cualquier tipo de tamaño para la imagen, se debe tener cuidado en no solicitar un tamaño demasiado grande que puede traer problemas como un tiempo de esperar más largo o que incluso la imagen requiera de una mayor cantidad de memoria de la que el dispositivo soporta.

Una vez que se ha configurado la aplicación de acuerdo a los requerimientos tanto del servidor como del dispositivo cliente ya es posible el inicio del monitoreo remoto por medio de la opción *Conectar* del menú principal.

Esta opción iniciará una conexión con el servidor por medio del servicio de transporte de datos disponible en el dispositivo, comúnmente CSD, GPRS o HSCSD. Esta conexión realiza una petición a un servlet con el cual se obtiene una lista en formato XML de las cámaras registradas por el sistema.

Como puede verse en la imagen, cada uno de los elementos de la lista están formados por el nombre de la cámara precedido por un pequeño icono que indica intuitivamente el estado en que se encuentra la cámara. Una paloma azul indica un buen funcionamiento, una cruz roja indica que por algún motivo la cámara no esta funcionando, ya sea porque se encuentra inhabilitada desde el sistema o porque no se ha podido establecer una conexión adecuada con ella.

Lista de cámaras.

Una vez que se ya se tiene acceso a la lista de cámaras y se puede observar la disponibilidad de cada una de ellas, ahora es posible obtener información más detallada por medio de un nuevo menú que presenta las opciones mostradas en la siguiente figura.

Menú de cámaras.

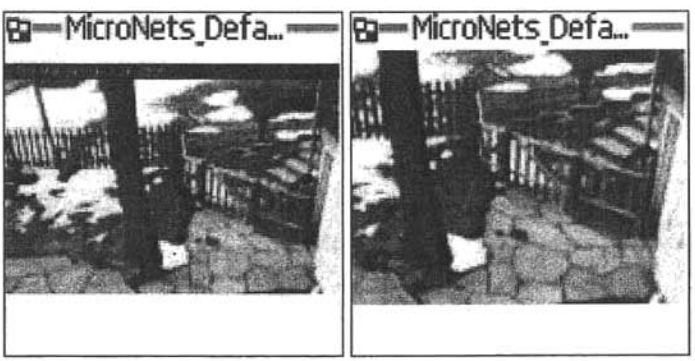
La opción de SnapShot permite al usuario obtener una imagen actual obtenida de la cámara con lo que podemos afirmar el objetivo principal de este sistema, el cual es el monitoreo remoto, se ha cumplido. Sin embargo, existen algunos factores a considerar, el primero de ellos se refiere a la velocidad en la transferencia de los datos, lo que esta estrechamente relacionado con el dispositivo, así como el servicio que proporcione la compañía celular. Por otro

lado, cada imagen enviada tiene un tamaño en bytes que va en relación directa tanto a sus dimensiones como a su complejidad.

Estos factores se reflejan en un tiempo de espera que la aplicación debe pasar antes de poder visualizar la imagen solicitada. Con la finalidad de mantener al usuario informado sobre este tiempo de descarga de la imagen, MicroNets proporciona una vista, que muestra el avance de este proceso, informando además de la cantidad de bytes que componen la imagen y permitiendo al usuario una opción para cancelar la descarga si así lo desea.

Pantalla de descarga.

Aunque la velocidad de transmisión de los datos no es un factor que un usuario pueda controlar, es posible regular el tamaño de la imagen a recibir modificando sus dimensiones, lo que se logra por medio de la opción de configuración del sistema indicando las dimensiones máximas preferentes de la imagen que deseamos recibir. Al ajustar estos valores podemos obtener imágenes de mayor calidad con su consecuente costo en el aumento de tiempo de descarga.



Imágenes descargadas con la configuración de alto y ancho de la imagen en 128x128, 170x170 y 220x220 respectivamente.

Opcionalmente a la descarga de una imagen, la aplicación permite una manera de emular la transmisión de video en vivo, sin embargo, esta opción requiere de una velocidad de conexión mucho más rápida de forma que pueda la aplicación pueda mantener un framerate razonable para que esta opción sea práctica.

La última acción que puede realizarse sobre cada una de las cámaras permite obtener más información sobre su estado y sobre si misma. Esta información incluye las acciones que esta realizando y en su caso la fecha y hora de el último movimiento de detectado.

Adicionalmente al monitoreo remoto, la aplicación MicroNets permite el envío de comandos al sistema de forma que este pueda reaccionar a ciertos eventos solicitados por el usuario. Para demostrar esta capacidad se ha incorporado el comando de alarma, con lo que el sistema Nets Desktop ejecuta una alarma sonora por un cierto periodo de tiempo. Aunque este es un ejemplo de un comando muy básico es posible incorporar nuevas funciones al sistema que le permitan realizar acciones mucho más complejas.

5.5. Sistemas similares.

Actualmente en el mercado podemos encontrar una serie de productos que ofrecen soluciones similares en lo que se refiere a video vigilancia por medio de sistemas basados en PC además de monitoreo remoto a través de teléfonos celulares.

AXIS Camera Station

El fabricante de cámaras IP ofrece varias soluciones para el uso de sus productos, donde resalta el software AXIS Camera Station[8] que provee una serie de funciones de monitoreo, grabación y administración para el uso de sus cámaras y servidores de video. Este sistema fue creado como una solución profesional para la vigilancia de lugares críticos que pueden requerir de una gran cantidad de cámaras.

Este software con un precio aproximado de 960 dólares con una licencia para 10 cámaras proporciona las siguientes funcionalidades:

- Visualización y grabación de video en vivo de múltiples cámaras simultáneamente.
- Diferentes modos de grabación: continua, programada, por alarma y/o detección de movimiento. (La opción de detección de movimiento trabaja obteniendo una imagen cada determinado tiempo y comparando las diferencias en un área específica de la imagen.)
- Soporta, el uso de la detección de movimiento incluida en cada cámara Axis, lo que ahorra ancho de banda en la red pues las imágenes solo son enviadas cuando el movimiento es detectado.
- Grabaciones de alta calidad.
- No existe limitantes de grabación por software.
- Múltiples funciones de búsqueda para eventos grabados.

[8] Axis Communications. (n.d.) AXIS Camera Station.
http://www.axis.com/products/cam_station_software/index.htm

- Acceso remoto por medio de un navegador Web o un cliente de Windows, incluido con el software.
- Permite el control de movimiento PTZ y cámaras de domo.
- Funciones de alerta por medio de sonido o correo electrónico.
- Soporte para audio de dos vías, en tiempo real sin grabación.

Algunos de los requerimientos mínimos que el software solicita del sistema son:

- Procesador Intel Pentium 4, 2 GHz.
- 512 MB en RAM.
- Disco Duro: 1 GB para instalación y 2 GB o más por cámara como base de datos para las imágenes.
- Sistema de archivos NTFS.
- Monitor con resolución XGA (1024x768) o superior.
- Tarjeta de video de 32 MB o más.
- Sistema operativo Microsoft Windows XP Profesional (SP1a), Windows 2000 (SP4), Windows 2003 Server.
- Internet Explorer 6.0, o superior.
- Internet Information Server (IIS) para el cliente Web.
- MS .NET runtime environment.

Software de Monitoreo Remoto de SmartHome

El software disponible para su compra en la pagina www.smarthome.com a un precio de aproximadamente 90 dólares proporciona una solución básica de vigilancia enfatizando el uso de una Web Cam como cámara principal y con el beneficio del monitoreo remoto por medio de un teléfono celular el cual requiere de

de la contratación de un servicio y contrato mínimo de un año por aproximadamente 10 dólares mensuales por video en vivo o 15 dólares de video en vivo y almacenamiento por 24 horas.

Los requerimientos de este sistema son:

- Sistema operativo Microsoft Windows 98 SE, Windows ME, Windows 2000, Windows XP Home o Windows XP Professional.
- 128 MB de RAM.
- 10 MB libres en disco duro para instalación del software.
- Conexión de Internet de banda ancha siempre activa.
- Un puerto USB libre para la Web Cam.
- Teléfono celular habilitado con CLDC 1.0 y MIDP 1.0 para monitoreo remoto.

Logitech MobileVideo

Logitech, el fabricante suizo de periféricos para computadora, ofrece este software como una extensión a sus productos para fomentar el uso de sus Web Cams. El sistema esta formado por dos componentes, el primero de ellos es una aplicación que hará la tarea de servidor de video, instalado en el mismo equipo donde se encuentra la Web Cam compatible. El segundo componente es una aplicación para teléfonos celulares habilitados con BREW con la que es posible ver imágenes en tiempo real de cámaras públicas o privadas protegidas por contraseña.

El software para PC esta disponible para su descarga gratuita desde la página http://mobilevideo.logitech.com, por otro lado, la aplicación móvil puede ser utilizada por una suscripción mensual de 5 dólares, sin embargo, solo esta disponible con algunos proveedores que ofrecen el servicio de alta velocidad en los Estados Unidos.

Los requerimientos para el uso de este sistema son:

- Una Web Cam Logitech compatible.
- Conexión a Internet.

- Sistema operativo Windows 98, Windows 2000, Windows ME o Windows XP.
- Procesador Pentium II 400, Celeron, AMD Athlon o superior.
- 64 MB de RAM.
- 200 MB de espacio libre en disco duro.
- Adaptador de video con una profundidad de color de 16 bits.
- Teléfono celular compatible con BREW.

Como se ha mencionado este sistema requiere de un dispositivo habilitado con BREW, para saber que importancia tiene esto, se presenta una breve descripción de esta tecnología.

BREW (Binary Runtime Environment for Wireless) es una plataforma creada por Qualcomm que ofrece un ambiente de desarrollo y ejecución de aplicaciones en dispositivos móviles. A diferencia de su contraparte, Java 2 Micro Edition, no se encuentra limitado por un lenguaje en particular, aunque aprovecha al máximo en código nativo de C/C++ situándose apenas por encima del chip del software del sistema. BREW cuenta, además con todas las características que se esperan de una plataforma para dispositivos móviles incluyendo un tamaño reducido, modelos de seguridad y distribución, así como una gran flexibilidad para la creación de un gran número de aplicaciones.

Uno de los inconvenientes de esta tecnología es que solo se encuentra disponible para dispositivos habilitados para redes 3G, lo que ha limitado mucho su popularidad fuera de los países con mayores avances en las telecomunicaciones como es el caso de Japón.

CONCLUSIONES

Es indudable la aportación y utilidad de los sistemas de video vigilancia, sin embargo, la selección del sistema adecuado a las necesidades se dificulta al tomar en cuenta la infraestructura requerida y en el caso de sistemas basados en PC, los requerimientos mínimos tanto de hardware como de software.

Una de las aportaciones más significativas de los últimos años a esta área fue la creación de cámaras digitales con la capacidad de integrarse a una red Ethernet. Las Cámaras IP permitieron reemplazar el uso de los costosos cables de las cámaras analógicas por el cable de par trenzado más económico y en ocasiones ya existente en el área a vigilar. Por otro lado, el uso de estas redes proporciona la ventaja adicional de una sencilla transición a sistemas inalámbricos utilizando los estándares IEEE 802.11.

La evolución de los sistemas de vigilancia hacia las redes de computadoras permite de forma más natural el uso de computadoras personales para la administración del sistema tanto para la visualización del video, como su almacenamiento. De esta forma debe ser claro que no debería existir limitante en la plataforma a utilizar siempre y cuando esta pueda conectarse a la red, con lo que el sistema puede alojarse en ambientes Windows, Linux o Mac.

La capacidad multiplataforma de la tecnología Java la hace la mejor opción para el desarrollo de una aplicación en este entorno y su constante evolución y mejoras en rendimiento y APIs disponibles permiten la creación de un sistema de video vigilancia de alto rendimiento que puede competir con cualquier producto actualmente en el mercado.

Por otro lado, el uso cada vez mayor de la telefonía celular, más y mejores servicios de datos, así como la introducción de nuevos dispositivos habilitados con la tecnología Java permiten la creación de aplicaciones que pueden extender las funcionalidades de los sistemas de escritorio.

3G. Es una abreviatura para tercera generación de telefonía móvil. Los servicios asociados con la tercera generación proporcionan mayor capacidad en las comunicaciones de voz y altas velocidades en la transmisión de datos lo que hace posible la transferencia de video en vivo.

ActiveX. Conjunto de tecnologías de interoperabilidad creadas por Microsoft, que permiten que los componentes de software escritos en diferentes lenguajes funcionen juntos en entornos de red.

AMS. El Application Management Software es una aplicación nativa en la mayoría de los equipos móviles responsable de la instalación y administración de las aplicaciones Java.

API. Abreviación para Application Program Interface. Conjunto de componentes y herramientas para la construcción de software.

Applet. Programa escrito en el lenguaje Java que puede ser insertado dentro de una pagina HTML y visualizado dentro de un navegador Web.

AVI. Audio video Interleave es un formato para archivos multimedia que puede contener tanto video como sonido.

Bluetooth. Tecnología de comunicación inalámbrica que posibilita la transmisión de voz y datos entre diferentes equipos mediante un enlace por radiofrecuencia a distancias no mayores a 10 metros.

Bytecodes. Lenguaje intermedio utilizado por la tecnología Java para proporcionar la capacidad de multiplataforma en sus aplicaciones.

CCD. Un CCD (Charge-Coupled Device) es un circuito integrado que contiene un número determinado de capacitores enlazados que con el control de un circuito interno pueden transferir su carga eléctrica a uno o varios de los capacitores adyacentes. Equipos como cámaras de video, cámaras fotográficas y scanners utilizan CCDs sensibles a la luz como un medio para digitalizar la luz recibida.

CCTV. Closed Circuit Televisión. Es un sistema de monitoreo de diversas áreas a través de video.

CDC. Connected Device Configuration. Es una configuración definida por la plataforma Java2 Micro Edition para dispositivos con configuración similar a un PDA.

CLDC. Connected Limited Device Configuration es una configuración definida en la plataforma Java2 Micro Edition para dispositivos con menores recursos que un PDA, como pueden ser teléfonos celulares.

Codec. Codec es una abreviación de "codificador/decodificador", que describe una especificación implementada en software, hardware o una combinación de ambos, para desempeñar transformaciones bidireccionales sobre datos y señales. En el caso del video digital, los codecs con utilizados para transformar tanto la información de audio como de video en versiones de menor tamaño y con diferentes niveles de calidad para su uso.

CSD. La tecnología Circuit Switched Data es usada por la infraestructura de la telefonía celular como una alternativa para la transferencia de datos.

CVS. Concurrent Versioning System es un sistema de control de versiones que mantiene registro de todo el trabajo y los cambios en la implementación de un proyecto (de software) y permite que distintos desarrolladores colaboren simultáneamente.

Demultiplexor. Ver multiplexor.

DVD. Disco Versátil Digital es un soporte para el almacenamiento de información de igual funcionamiento y tamaño que el CD-ROM, aunque con un significativo aumento en la densidad de información que puede almacenarse en su superficie. Las capacidades de almacenamiento del DVD permiten la distribución de video de alta calidad lo que da lugar a que se realicen comparaciones de calidad tomando como base el DVD.

Frame. Cuadro o imagen individual de las que se componen una animación o video. La sucesión de frames consecutivos con pequeñas diferencias producen la ilusión de movimiento.

Framerate. Es el número de frames en un espacio de tiempo determinado.

GCF. Generic Connection Framework es una API utilizada por la plataforma J2ME que permite la creación de conexiones de entrada/salida como una forma de reemplazo para los paquetes *java.io* y *java.net*.

GPRS. General Packet Radio Service es una tecnología de conexión constante para la transmisión de datos que trabaja sobre el estándar GSM.

GSM. Global System for Mobile communications, formalmente conocido como Group Special Mobile es un estándar mundial de comunicación para dispositivos móviles digitales.

H.261. Es un codec de video originalmente para videoconferencias. H.261 utiliza un algoritmo muy flexible que permite configuraciones para equilibrar la calidad con el ancho de banda o el framerate dependiendo de la cantidad de movimiento que presente el video.

H.263. Es una actualización del codec H.261 que proporciona un mejor desempeño, recuperación de errores y mayores resoluciones de imagen.

HSCSD. High Speed Circuit Switched Data es una mejora a los servicios de datos de las redes GSM. Permite el acceso a servicios diferentes a voz a velocidades mayores a 28.8 Kbps por medio del uso de múltiples canales

IEEE. El Instituto de Ingenieros Eléctricos y Electrónicos (Institute of Electrical and Electronics Engineers) es una asociación internacional sin fines de lucro dedicada a la creación de estándares para la industria electrónica y de computación.

IP. Internet Protocol es un protocolo que define como la información es enviada a través de una red TCP/IP.

IR. El termino IR hace referencia al medio de comunicación (luz infrarroja) por el cual se transmiten datos entre dispositivos.

J2EE. Abreviación de Java2 Enterprise Edition.

J2ME. Abreviatura de Java2 Micro Edition.

J2SE. Abreviatura de Java2 Standard Edition.

JAR. Java Archive es un formato de archivo que permite empaquetar múltiples archivos en un solo archivo. Generalmente un archivo JAR contiene archivos de clase y otros recursos asociados con applets o aplicaciones.

JDK. J2SE Development Kit es un paquete de distribución de Java que proporciona herramientas para ejecutar y desarrollar aplicaciones.

JPEG. Joint Photographic Experts Group es un algoritmo diseñado para comprimir imágenes con calidad fotografica que logra reducir el tamaño de la imagen hasta 20 veces, a costa de disminuir su calidad.

JRE. J2SE Runtime Environment es un paquete de distribución de Java que contienen los arhivos necesarios para que usuarios finales puedan ejecutar aplicaciones Java.

JSP. JavaServer Pages es una tecnología que permite la creación de contenido web dinámico de forma rápida y simple.

JVM. Java Virtual Machine. La Máquina Virtual de Java es un componente indispensable de la tecnología Java encargado de la ejecución de las aplicaciones por medio de la interpretación de los bytecodes.

KVM. K Virtual Machine. La KVM es la máquina virtual diseñada para satisfacer las necesidades de la plataforma J2ME. El uso de la letra K proviene de Kilobyte haciendo mención al limitantes de memoria de los dispositivos a los que esta enfocado.

LAN. Local Area Network o Red de Área Local es una red de computadoras interconectadas distribuida a lo largo de una área relativamente pequeña como puede ser una oficina o un edificio.

LCD. Liquid Cristal Display. Tipo de pantalla basado en la excitación y polarización de cristales líquidos situados entre dos pares de vidrio. La principal característica de las pantallas LCD es su reducido tamaño y bajo consumo de energía.

MIDlet. Aplicación J2ME basada en el perfil MIDP.

MIDP. Mobile Information Device Profile es uno de los perfiles definidos por la plataforma J2ME que se basa en equipos de limitada capacidad de memoria y procesamiento y con ciertas capacidades de despliegue visual como son los teléfonos celulares.

MJPEG. Motion JPEG es un codec de video que se basa en el uso de imágenes individuales cada una almacenada en el formato JPEG.

MPEG. Motion Picture Experts Groups es un conjunto de normas para la compresión de video y audio digital. Algunos de los codecs mas difundidos de este conjunto son MPEG-1, MPEG-2 y MPEG-4 los cuales son usados para la codificación de video de alta calidad como las peliculas en formato DVD.

Multiplexor. Dispositivo que permite la codificación de n número de entradas en una solo canal de salida. Un demultiplexor es un dispositivo que realiza la función opuesta al multiplexor, esto es, permite la división y selección de varias señales a partir de una sola señal compuesta de entrada.

OTA. Over-the-Air es una especificación definida formalmente por el perfil MIDP de J2ME que ofrece una serie de lineamientos para la distribución de aplicaciones para dispositivos moviles.

Pan. Forma en como se conoce al movimiento horizontal sobre su propio eje que puede tener una cámara.

PDA. Personal Digital Assistant. Computadoras de mano originalmente diseñadas como agendas electrónicas que son capaces de ejecutar un gran número de aplicaciones. Algunos de los dispositivos PDA más conocidos son Palm, PocketPC y SmartPhones.

Pixel. Del inglés picture element es la menor unidad en la que se puede descomponer una imagen digital, ya sea una fotografía, un frame de video o la imagen mostrada por cualquier monitor de computadora.

Plug-in. Pequeños componentes de software que interactúan con otras aplicaciones para aportar una función o utilidad específica.

PNG. Portable Network Graphics es un formato de compresión de imágenes.

QuickTime. Arquitectura multimedia desarrollada por Apple consistente en un conjunto de bibliotecas, un formato de archivo y un reproductor multimedia.

Render. Término que se refiere a la generación de un frame o imagen individual.

Router. Componente de hardware o software de interconexión de redes de computadoras que opera en la capa 3 del modelo OSI. Este dispositivo interconecta segmentos de red o redes enteras.

RTP. Real Time Protocol es un protocolo de red diseñado para la transmisión y recepción de flujos de datos en tiempo real. Uno de sus mayores usos es en aplicaciones de videoconferencia.

PTZ. Abreviación de Pan/Tilt/Zoom.

Servlet. Programa escrito en lenguaje Java corriendo del lado del servidor que permite generar contenido Web dinámico.

SMS. Short Message Service es un protocolo de envío de mensajes cortos entre dispositivos móviles a través de redes como GSM.

SQL. El lenguaje de consulta estructurado (Structured Query Language) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas.

TCP/IP. Sistema de protocolos en los que se basa buena parte de Internet. El primer protocolo (Transmission Control Protocol) se encarga de dividir la información en paquetes en el origen, para luego recomponerla en el destino. El segundo protocolo (Internet Protocol) dirige la información adecuadamente a través de la red.

Tilt. Referencia al movimiento vertical que poseen algunas cámaras de vigilancia.

UML. Unified Modeling Language es un lenguaje de modelado de sistemas de software.

URL. Estándar para la representación de direcciones de Internet asi como para recursos específicos dentro de cada dirección.

Web Cam. Pequeña cámara digital, generalmente de baja resolución, conectada a una computadora con la cual puede capturar imágenes y transmitirlas a través de internet.

Wi-Fi. Abreviatura de Wireless Fidelity, es un conjunto de estándares para redes inalámbricas basado en las especificaciones IEEE 802.11.

WLAN. Wireless Local Area Network es una red de computadoras de área local basada en algunas de las especificaciones de redes inalámbricas como IEEE 802.11

XML. Acrónimo del inglés eXtensible Markup Language. Es un lenguaje de descripción de documentos que permite el intercambio de información a través de enfatizar la estructura de la información mas que su representación.

Zoom. Efecto de acercamiento y alejamiento para visualizar el contendido de imágenes digitales o imágenes proporcionadas por una cámara.

REFERENCIAS

Productos y Soluciones Axis

Axis Communications.(n.d.) Application software for network cameras & video servers. http://www.axis.com/products/video/software/index.htm

Axis Communications. (n.d.) Cámaras de Red Axis, Descubra sus beneficios. http://www.axis.com.mx/productos/camaras/index.html

Axis Communications. (n.d.) AXIS Camera Explorer (ACE). http://www.axis.com/products/cam_ace/

Axis Communications. (n.d.) AXIS Camera Station, Flexible surveillance software for Axis network video. http://www.axis.com/products/cam_station_software/

Axis Communications. (n.d.) AXIS Camera Recorder, Software for recording and monitoring. http://www.axis.com/products/cam_rec_software/

Axis Communications. (n.d.) Network Video Developer Pages. http://www.axis.com/techsup/cam_servers/dev/index.htm

Axis Communications. (Abril 24, 2003) Network camera technology. http://www.axis.com/products/video/camera/about_cameras/netcam_tech.htm

Axis Communications. (n.d.) Axis network cameras, When you demand superior quality and performance .http://www.axis.com/products/video/camera/index.htm

Axis Communications. (n.d.) Factors to consider. http://www.axis.com/promotion/security/factors.htm

Axis Communications. (n.d.) Comparativo. http://www.axis.com.mx/productos/camaras/comparativo.htm

Axis Communications. (n.d.) Solutions by industry. http://www.axis.com/solutions/video/index.htm

Cámaras IP

SuperInventos,(Enero 11, 2005). Televigilancia y Seguridad Electrónica. http://www.superinventos.com/S130770.htm

Al Kelly (Abril 26, 1996). What is a CCD camera and how does it work? http://www.ghg.net/akelly/ccdbasic.htm

CCTV

Sistema de video Vigilancia por Internet (n.d.) http://mipagina.cantv.net/redondo/video.htm

Control Electronic Security (2002). Sistemas para Circuitos Cerrados de Televisión. http://www.controlelectronic.com/cameras_cctv_espanol.htm

Ramon F. Mateo. (n.d.). Diseño de un sistema de TV de circuito cerrado CCTV. http://www.monografias.com/trabajos/cctelevis/cctelevis.shtml

eHow. (n.d.) How to Choose a Closed-Circuit TV System for Your Home. http://www.ehow.com/how_4930_choose-closed-circuit.html

Wikipedia, The Free Encyclopedia (Mayo 13, 2005). Closed-circuit television. http://en.wikipedia.org/wiki/Closed-circuit_television

123.CCTV Security Camera Surveillance Equipment (n.d.) http://www.123cctv.com/security.html

CCTVVideo.com (n.d.) http://www.cctvvideo.com/

Dist-Soft, Distribuidor de Software S.A. de C.V. (n.d.) Sistema de Vigilancia Remota. http://www.dist-soft.com/svr/

Tecnología Java

Sun Microsystems, Inc. (n.d.). Introduction to Mobility Java Technology. Sun Developer Network. http://developers.sun.com/techtopics/mobility/getstart/

Sun Microsystems, Inc. (n.d.). The Java Tutorial, Trail: Getting Started http://java.sun.com/docs/books/tutorial/getStarted/index.html

James Gosling, Henry McGilton. (Mayo 1996). The Java Language Environment: Contents. Sun Developer Network. http://java.sun.com/docs/white/langenv/

Sun Microsystems, Inc. (n.d.). Building Compelling Services for the Wireless Market Using Java Technology. Sun Developer Network. http://developers.sun.com/techtopics/mobility/getstart/articles/whyjava/

Sun Microsystems, Inc. (n.d.). The K virtual machine (KVM). Sun Developer Network. http://java.sun.com/products/cldc/wp/

Monica Pawlan. (Marzo 1999). Essentials of the Java Programming Languaje, Part 1. Sun Developer Network http://java.sun.com/developer/onlineTraining/Programming/BasicJava1/

Sun Microsystems, Inc. (n.d.). Java 2 Platform. Sun Developer Network. http://java.sun.com/java2/whatis/

Jon Byous. (1998). Java Technology: The early years. Sun Develpoer Network. http://java.sun.com/features/1998/05/birthday.html

Sun Microsystems, Inc. (Junio 1999). Sun Announces Java 2 Platform, Micro Edition. http://www.sun.com/smi/Press/sunflash/9906/sunflash.990615.10.html

Tim Lindholm, Frank Yellin. (1999). The Java Virtual Machine Specification Second Edition. Sun Developer Network http://java.sun.com/docs/books/vmspec/2nd-edition/html/VMSpecTOC.doc.html

Sun Microsystems, Inc. (n.d.). The Java Platform: Five Years in Review. Sun Developer Network. http://java.sun.com/features/2000/06/time-line.html

Sun Microsystems, Inc. (2005). Java Community Process http://jcp.org/

Sun Microsystems, Inc. (n.d.). Java 2 Platform Standar Edition Overview. Sun Developer Network. http://java.sun.com/j2se/overview.html

Sun Microsystems, Inc. (n.d.). Desktop Java Technology. Sun Developer Network. http://java.sun.com/j2se/desktopjava/index.jsp

Sun Microsystems, Inc. (n.d.). Core Java Technology. Sun Developer Network. http://java.sun.com/j2se/corejava/index.jsp

Eric Armstrong, Jennifer Ball, Stephanie Bodoff, Debbie Bode Carson, Ian Evans, Dale Green, Kim Haase, et al. (Diciembre 16, 2004). The J2EE 1.4 Tutorial. Sun Developer Network. http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html

Dana Nourie. (Marzo 24, 2005). Getting Started with an Integrated Develpment Environment (IDE). Sun Developer Network. http://java.sun.com/developer/technicalArticles/tools/intro.html

Java Media Framework

Sun Microsystems, Inc. (n.d.). Java Media Framework API (JMF). Sun Developer Network. http://java.sun.com/products/java-media/jmf/index.jsp

Sun Microsystems, Inc. (Mayo 11, 1998). JMF Programmers Guide. Sun Developer Network. http://java.sun.com/products/java-media/jmf/1.0/guide/index.html

Budi Kurniawan (Abril, 2001). Program Multimedia with JMF Part 1. Sun Developer Network, reprinted from JavaWorld. http://java.sun.com/developer/technicalArticles/Media/mediaframework/

Sun Microsystems, Inc. (Noviembre 19, 1999). Java Media Framework API Guide. Sun Developer Network. http://java.sun.com/products/java-media/jmf/2.1.1/guide/JMFTOC.html

Desarrollo y Soluciones móviles

Microsoft Corporation. (Junio, 2001) Desarrollar soluciones móviles. Msdn Latinoamérica. http://www.microsoft.com/spanish/msdn/comunidad/comunidades/aplicaciones_moviles/art02/default.asp

Pawlan Jeffrey. (Agosto, 2000). World of Wireless Communications. Sun Developer Network. http://developers.sun.com/techtopics/mobility/getstart/articles/hardware/

Clements Tom. (Julio, 2002). Making Sense of Cellular. Sun Developer Network. http://developers.sun.com/techtopics/mobility/getstart/articles/radio/

Fernández Luna Juan Manuel. (Septiempre, 2004). Programación de dispositivos móviles con Java. DECSAI. http://flanagan.ugr.es/J2ME/INTRO/index2.htm

Marshall Brain. (n.d.). How the Radio Spectrum Works. How Stuff Works http://electronics.howstuffworks.com/radio-spectrum1.htm

Carmen Carmack, Craig Freudenrich. (n.d.). How PDAs Work. How Stuff Works. http://electronics.howstuffworks.com/pda.htm

PalmSource Inc. (n.d.). Palm OS
http://www.palmsource.com/palmos/#print

Scott Jung. (n.d.). For New Users.
http://www.palmsource.com/interests/newusers/

Sony Ericsson Mobile Communications AB. (n.d.). Developer World
http://developer.sonyericsson.com/

Ortiz Enrique (Octubre, 2004). A Survey of J2ME Today. Sun Developer Network.
http://developers.sun.com/techtopics/mobility/getstart/articles/survey/

Pawlan Monica. (Junio, 2001). Introduction to Wireless Technologies. Sun Developer Network.
http://developer.sun.com/techtopics/mobility/getstart/articles/intro/

Marejka Richard. (Febrero, 2005) Leaming Path: MIDlet Life Cycle. Sun Developer Network.
http://developers.sun.com/techtopics/mobility/learn/midp/lifecycle/

Qusay H. Mahmoud. (Octubre, 2000). Wireless Application Programming: MIDP Programming and Packaging Basics. Sun Developer Network.
http://developer.sun.com/techtopics/mobility/midp/articles/getstart/

Ortiz Enrique. (Agosto, 2004). Managing the MIDlet Life-Cycle with a Finite State Machine. Sun Developer Network.
http://developers.sun.com/techtopics/mobility/midp/articles/fsm/

Knudsen Jonathan, Nourie Dana. (Septiembre, 2003). Wireless Development Tutorial Part 1. Sun Developor Network.
http://developer.sun.com/techtopics/mobility/midp/articles/wtoolkit/

Qualcomm Incorporated. (n.d.) Brew. http://brew.qualcomm.com/brew/en/

Mobiledia Corp. (n.d.) Cell Phone List.
http://www.mobiledia.com/phones/list.html

Herramientas de Desarrollo

NetBeans Community. (n.d.). NetBeans IDE.
http://www.netbeans.org/products/ide/index.html

Eclipse.org (n.d.). http://www.eclipse.org/

JetBrains. (n.d.). IntelliJIdea, The Most Intelligent Java IDE. http://www.jetbrains.com/idea/

Wikipedia, La Enciclopedia Libre. (Mayo, 2005). Entorno integrado de desarrollo. http://es.wikipedia.org/wiki/Entorno_integrado_de_desarrollo

Xinox Sofware. (n.d.). JCreator. http://www.jcreator.com/

Borland Software Corporation. (n.d.). JBuilder, The leading development solution for Java. http://www.borland.com/jbuilder/

Sun Microsystems, Inc. (n.d.). Sun Java Studio Creator. http://www.sun.com/software/products/jscreator/index.xml

Gestión de video

Terran interactive, Inc. (1998). Codec Central, Index of Codecs. http://www.siggraph.org/education/materials/HyperGraph/video/codecs/Default.htm

Cutanda López Ramón. (Semptiembre 27, 2003). Guia Rápida. http://www.videoedicion.org/manuales/guia_rapida.htm

Brain Marshall, Reid Roxanne. (n.d.). How Video Editiong Works. How Sfuff Works. http://computer.howstuffworks.com/video-editing.htm

Productos de videovigilancia

Axis Communications. (n.d.) AXIS Camera Station. http://www.axis.com/products/cam_station_software/index.htm

Logitech Inc. (n.d.). Logitech Mobile Video. http://mobilevideo.logitech.com/

U.S. Cellular. (n.d.). Logitech Mobile Video. Easyedge. http://easyedge.uscc.com/easyedge/GameDetail.do?1060

SmartHome. (n.d.) Software Lets You Watch Video Over Your PC, Phone or PDA. http://www.smarthome.com/7787m.html