SISTEMA DE CONTROL DE INVENTARIOS (SCI) MEDIANTE EL USO DE TEGNOLOGÍAS DE INTERNET Y COMUNICACIÓN INALÁMBRICA

Hugo Camilo Robayo Ayala[1], Gerardo Rodrigo Romero Romero[2], Galo Valverde[3]

1 Candidato a Ingeniero en Computación Esp. Sistemas de Información 2005, mail: hrobayo@ieee.org
2 Candidato a Ingeniero en Computación Esp. Sistemas de Información 2005, mail: rromero@espol.edu.ec
3 Director de Tesis, Ingeniero en Computación 1998, Maestría en Dirección Estratégica y Gestión de Innovación, mail: gvalverd@espol.edu.ec

Abstracto.- Este documento ofrece una descripción general de las tecnologías inalámbricas, varias herramientas para la aplicación de estas así como la explicación del sistema desarrollado para mejorar la administración de inventarios en la ESPOL. El desarrollo de este sistema tomo como base los procesos llevados a cabo en las bodegas de la ESPOL, además se integra a estos a través del uso compartido de la misma base de datos. Se incluye las consideraciones hechas en el proceso de análisis tales como las operaciones que el sistema abarca, la presentación al usuario, varios aspectos necesarios en el proceso de diseño así como la descripción de las pruebas y resultados hechos sobre el sistema para verificar su rendimiento.

Finalmente se presentan las conclusiones a las cuales llegamos luego de realizado el sistema, en estas exponemos las ventajas de las tecnologías empleadas, beneficios para que el sistema entre a producción, y como se manejo los escasos recursos que nos brinda los dispositivos móviles para lograr una aplicación eficiente.

Abstract.- This paper offers a general description about wireless technologies, some tools for applying these technologies and the explanation of the developed system for improving the manage inventories in ESPOL. To develop of this system has its base in the process made in ESPOL stores; also this system is incorporated to ESPOL systems through the same database. In this paper is included the considerations made in the analysis process, operations made for the system, presentation to users, stages in the process design and at last tests and results made on the system for verifying the performance.

Finally are showed the conclusions, its shows advantages of technologies used in the system, reasons for deploying the system in a production environment and how handled little resources in Mobil devices for achieving an efficient application was.

Términos clave—computación móvil.- J2ME, J2EE, multi capa, inventario, MVC.

1. INTRODUCCION

La necesidad de reducir costos y aumentar la productividad en el trabajo de los personas que realizan los ingresos de inventarios en la bodega de la ESPOL, junto con la aparición de dispositivos móviles inalámbricos con mayores capacidades en poder de procesamiento, tamaño de la memoria y vías de comunicación son la base para el desarrollo de un sistema automatizado que ayude en el trabajo del personal de bodega que se lo ha llamado: Sistema de Control de Inventarios (SCI).

Actualmente, la ESPOL tiene un sistema financiero el cual también maneja la información de inventarios llamado SAF. El SAF está desarrollado sobre la plataforma de Visual Age de IBM y con el lenguaje de programación SmallTalk, debido a que este ambiente no permite la integración con dispositivos móviles el SCI esta desarrollado sobre las Tecnologías Java. El SCI se basa en una arquitectura multicapa y aprovecha las ventajas que ofrecen las especificaciones J2EE y J2ME. La integración con la plataforma actual de la ESPOL es hecha a través del acceso a la base de datos, es decir,

los dos sistemas trabajan con los mismos datos.

El SCI está dividido en una aplicación cliente basada en MIDP 2.0 y CLDC 1.1 ambos APIs de J2ME [7], además, puede ejecutarse en cualquier dispositivo móvil que soporte éstas especificaciones y la aplicación que se ejecuta en el servidor de aplicaciones compatible con las especificaciones J2EE.

Durante el desarrollo del SCI se utilizó el IDE de IBM Websphere Device Developer basado en Tecnología Eclipse, el cual integra el API de J2ME, emuladores de dispositivos y organizadores de tareas muy útiles para el desarrollo. Las pruebas de rendimiento se las efectuó en un dispositivo Palm Tungsten C, con procesador Intel Xcale de 400 Mhz., 4 MB de memoria dinámica y 51 MB de memoria, este dispositivo ejecutó Palm OS como sistema operativo y J9VM de IBM como máquina virtual de Java [2]. Además, la aplicación de servidor se ejecutó en una computadora Dell Optilex SX-270 con procesador Intel Pentium IV y 680 MB de memoria RAM, y esta ejecutó el servidor de Aplicaciones Sun Java System Server 8 update 1

2. ANALISIS

Actualmente, se dispone de un sistema que administra los inventarios en la bodega, pero aun se llevan varios controles escribiendo los datos en documentos. Un ejemplo del proceso actual de ingreso de inventarios se describe a continuación:

- El personal administrativo recibe la documentación sobre la compra de inventarios.
- Se entrega al personal de bodega hojas con las listas con los datos de los inventarios a ser ingresados.
- El personal de bodega revisa cada ítem y escribe los datos como: marca, modelo, cantidad, etc. de cada inventario en las hojas.
- Este listado es remitido al personal administrativo e ingresado al sistema de la ESPOL.

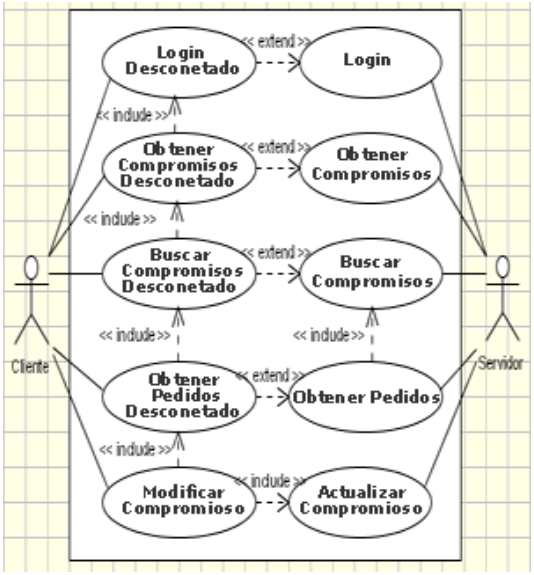

Figura 1. Diagrama de casos de uso del SCI.

2.1. Casos de Uso

La fortaleza en el diseño del SCI radica en proveer dos modos de operación: conectado y desconectado, esta diferencia de modos de operación es debido a las limitaciones dadas por la poca confiabilidad de las redes inalámbricas y para evitar un mayor consumo de energía en el dispositivo móvil. En el modo conectado el cliente siempre dispone de una conexión a la red y puede acceder a los servicios que la aplicación del servidor provee. Sin embargo, cuando la red no está disponible el SCI cambia a modo desconectado y trabaja con los datos almacenados localmente.

La figura 1 muestra la interacción entre los casos de usos de modo conectado y los casos de uso de modo desconectado.

2.2. Interfaz de Usuario

En el desarrollo de las interfaces del SCI, se tomaron en cuenta los siguientes aspectos:

- En las tareas en las cuales se hace uso de los recursos de red, se provee de retroalimentación a través de

animaciones que muestran el avance del proceso.

- El SCI brinda la posibilidad de parar los procesos que se están ejecutando, esto se puede apreciar en las tareas que tienen que ver con la recuperación y envío de datos.
- Las interfaces del SCI son simples y predecible el cual ayuda a guiar al usuario en las tareas que tiene que realizar.

En general, las aplicaciones que se diseñan con MIDP, tienen características similares en cuanto a la distribución de los elementos en la pantalla de los dispositivos, aunque se las aprecie de manera diferente, como lo muestra la figura 2.

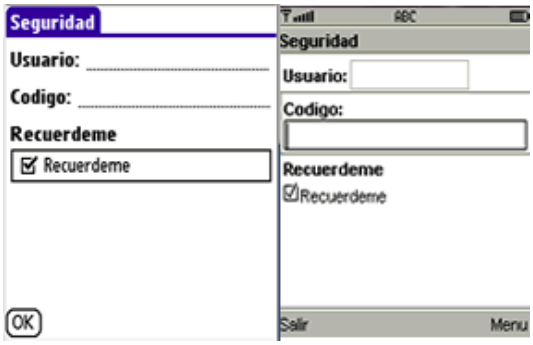

Figura 2. Representación de MIDP de la pantalla de ingreso del SCI en dos dispositivos diferentes.

3. DISEÑO

3.1. Arquitectura del Sistema

El SCI se basa en una arquitectura de varias capas, las cuales establecen niveles de desarrollo para resolver problemas como la administración de transacciones, manejo de bases de datos y otros. El SCI está dividido en 4 capas que se listan a continuación:

- Capa 0: Aplicación Cliente.- Se ejecuta en un dispositivo móvil, procesa las entradas del usuario y almacena datos localmente.
- Capa 1: Presentación del servidor.- esta capa representa las interfaces de los servicios a los cuales los clientes tienen acceso.
- Capa 2: Lógica del negocio.- esta capa representa las operaciones y procesos de la aplicación, con clases que abstraen los conceptos como: compromisos de compra, activos fijos, pedidos, etc.
- Capa 3: Lógica de los datos.- esta capa representa los subsistemas que manejan la persistencia de los datos.

3.2. Modelos Aplicados

Para soportar ambos modos de operación (Modo Conectado y Desconectado), la aplicación cliente debe mantener su propio modelo local de datos. El modelo de datos del Cliente es soportado por una parcial o total replicación del modelo de datos del servidor.

Patrón de Diseño Modelo-Vista-Controlador

El Patrón de Diseño ***Modelo-Vista-Controlador (MVC)*** [1], promueve la separación del modelo de datos de la presentación. Además, es ampliamente usado para aplicaciones interactivas, se divide en tres componentes funcionales: modelo, vista y controlador. Cada componente maneja sus tareas específicas y sus responsabilidades hacia los otros dos componentes, como lo muestra la figura 3.

Las consecuencias de usar el patrón MVC se detallan a continuación:

- Reutilización de componentes del Modelo.- La separación de modelo y vista, permite a múltiples vistas usar el mismo modelo. Consecuentemente, los componentes del modelo de una

aplicación son fáciles de implementar, probar y mantener.
- Fácil soporte para nuevos tipos de clientes.- simplemente se escribe la vista y algo de la lógica del controlador e incorporarlo dentro de la aplicación empresarial
- Incremento en la complejidad del diseño. Este patrón introduce muchas clases extra debido a la separación del modelo, vista y controlador.

Patrón de Diseño de Estructura de Fachada

El patrón de diseño de Estructura de Fachada puede ser usado para ocultar la complejidad del modelo de datos del lado del cliente, debido a la complejidad que puede surgir de la implementación de dos modos de operación: conectado y desconectado, especialmente en la replicación, mantenimiento y acceso remoto de los datos.

Formalmente, el patrón de diseño de Fachada hace fácil el uso de un subsistema porque este provee de una interfase de alto nivel que unifica el conjunto de subinterfases. La meta de diseño es reducir la comunicación y las dependencias entre varios subsistemas. Muchos clientes necesitan los servicios de un subsistema como de todo el sistema en sí pero no necesitan conocer cada clase individual. Usando un objeto de fachada, es posible dar a los clientes una simplificada interfaz a los subsistemas.

Patrón de Diseño Proxy

El patrón de Diseño Proxy es usado para abstraer la lógica que maneja el acceso a los datos remotos, como el protocolo de comunicaciones Cliente/Servidor, y optimizaciones relacionadas, como el almacenamiento temporal de datos. Formalmente, un Proxy provee una manera para redirección del control de acceso a otro objeto.

Entre otros usos, un Proxy Remoto provee representatividad local para un objeto en un diferente espacio de direccionamiento, como en un Servidor de Aplicaciones Remoto.

Replicación de Datos

Debido a las limitaciones de memoria de los dispositivos móviles, el modelo de datos del cliente sólo replica un subconjunto relevante del modelo de datos del servidor.

El filtrado de datos puede ayudar mucho cuando se esta replicando porciones del modelo de datos del servidor. Menos data en el lado del cliente mejora el procesamiento y reduce el tiempo para descargar la información

Para proceder a realizar esta replicación se han tenido en cuenta lo siguiente[2]::

- ¿Debe de realizarse una replicación parcial o total de los datos?
- ¿Cuan costoso es replicar toda la estructura de datos del servidor?
- ¿Cuando manejar las operaciones de sólo lectura o lectura/escritura en el cliente?
- ¿Es permitida la modificación concurrente de la data compartida con otros usuarios?
- ¿La data es sensitiva al tiempo y por consecuencia esta expira?

Persistencia de Datos

La especificación MIDP provee de mecanismos para almacenar los datos, este es llamado ***Record Management System (RMS)***; es un API que permite el acceso a las bases de datos de los dispositivos móviles, que generalmente son simples bases de datos en las cuales se pueden crear registros simples de hasta 64 Kb [3]. Cada registro es almacenado y recuperado en un arreglo de bytes. El tamaño de los registros puede variar y el formato de los datos no esta limitado por RMS.

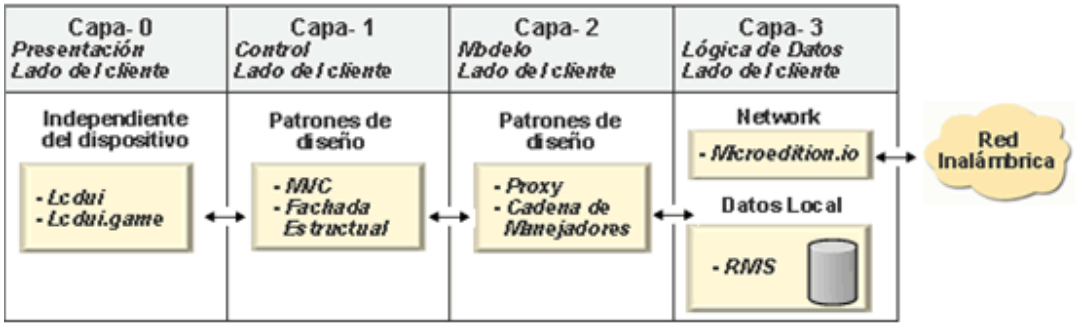

Figura 3. Modelo de la Aplicación Cliente

Debido a que RMS almacena y recupera datos como arreglos de bytes, MIDP no soporta los estándares de Serialización de Java pero esto puede realizarse haciendo que los objetos implementen la interfase java.io.Serialize por lo cual debe de implementarse un mecanismo de Serialización propio. Para el efecto se ha implementado una interfase común llamada *IndexedObjectc* por medio de la cual se logra el almacenamiento y recuperación en forma binaria de las bases de datos del dispositivo móvil así como la comunicación de objetos vía http mediante el envío de un flujo de bytes.

Estrategias aplicadas en el servidor
Patrón de Diseño: Delegación del Negocio

En aplicaciones distribuidas, la búsqueda y el manejo de excepciones para componentes remotos pueden ser muy complejos. Cuando las aplicaciones usas componentes de negocios es muy probable que el código de la aplicación deba cambiar para reflejar los cambios en el API de los componentes.

El ***Patrón Business Delegate*** maneja la complejidad de buscar componentes distribuidos y el manejo de excepciones, puede adaptar la interfase de los componentes de negocios para simplificar la interfase que va a ser usada.

Patrón de Diseño Fachada de Sesión

El patrón de diseño de ***Fachada de Sesión*** define componentes de negocios de alto nivel que centraliza las complejas relaciones de los componentes de bajo nivel. Una Fachada de Sesión está implementada con un EJB de Sesión con una sola interfase para la funcionalidad de la aplicación o un subconjunto de aplicaciones. También descompone las interacciones entre objetos de negocios de bajo nivel, haciendo el modelo más flexible y comprensible. El beneficio que se obtiene con esto se puede apreciar en la figura 4 en donde se ha disminuido el número de llamadas remotas.

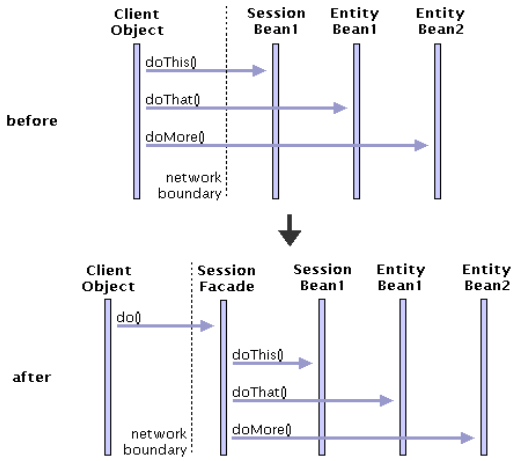

Figura 4. Patrón de Diseño de Fachada de Sesión

4. PRUEBAS DE RENDIMIENTO

Poco ancho de banda y latencia en la red configuración de la conexión toma tiempo, además de esto mejorar la velocidad de la conexión resulta costoso. [4] El ambiente de desarrollo generalmente es en emuladores.

4.1. Uso de Memoria

En los dispositivos inalámbricos se debe prestar especial cuidado al uso y administración de la memoria ya que este es un recurso limitado y además crítico. [5] [6]. El API de J2ME ofrece métodos muy útiles al momento de examinar el uso de memoria y duración de los procesos. Por ejemplo se dispone de métodos para determinar la duración de un proceso o verificar el uso de memoria en determinado momento como se ilustra en el código siguiente:

```
long inicio, fin;
inicio =
System.currentTimeMillis();
someMethod();
fin = System.currentTimeMillis();
long duracion = fin - inicio;
```

Código de Ejemplo 1: Examinar el tiempo de ejecución de un proceso

```
Runtime runtime =
Runtime.getRuntime();
long antes, despues;
System.gc();
antes = runtime.freeMemory();
Object newObject = new String();
depues = runtime.freeMemory();
long memoria_usado = antes -
despues;
```

Código de Ejemplo 2: Verificación de la memoria usada

Con la ayuda de estos métodos se puede determinar el comportamiento de la memoria y de acuerdo a esto proceder a optimizar la forma en la cual se realizan ciertos procesos.

En el SCI el análisis de la memoria se realizo en base a dos aspectos: uso de memoria y creación de objetos. Los parámetros con los que se configuro el emulador obedecen a las especificaciones de la Palm **Tungsten C** con la cual se probó el sistema, son los siguientes:

- Capacidad de Almacenamiento: 50000 KB
- Tamaño de heap: 4000 KB

Verificando el código de la aplicación se puede establecer, empíricamente, secciones de código en las cuales el uso de memoria y creación de objetos se incrementan rápidamente sin liberar recursos. En aquellas secciones se ejecutó una recolección explícita, los datos de las pruebas así como consideraciones sobre el uso de la memoria son apreciados en la figura 5.

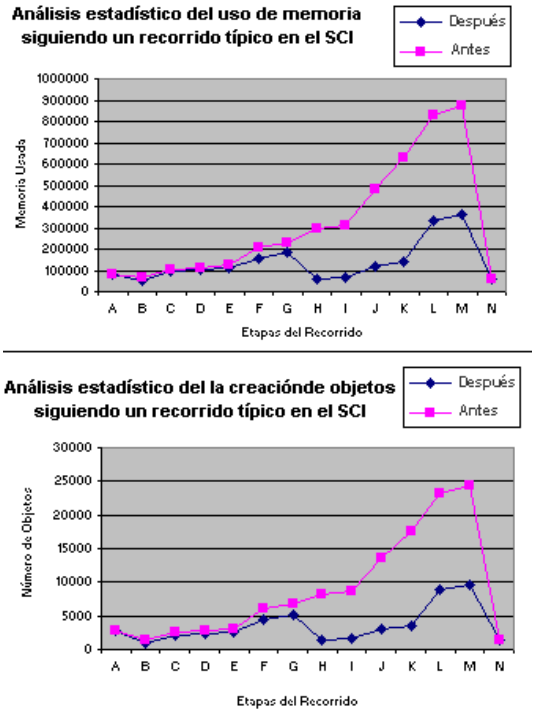

Figura 5.- Uso y optimización de memoria.

En la figura 5, el color fucsia muestra el uso de memoria y la creación de objetos cuando las recolecciones de basura son administradas por el sistema, el color azul denota el efecto sobre los recursos del sistema al ejecutar varias recolecciones de basura de manera explícita, mediante la llamada al sistema System.gc().
Claramente se aprecia, una disminución considerable en los picos en la creación de objetos y uso de memoria.

4.2. Conectividad y Latencia

En lo que respecta a conectividad, colocando el punto de acceso en diferentes lugares se ha probado a instalar la aplicación por distintos medios y se han conseguido tiempos similares tanto en la instalación como en la recuperación de datos cuando la aplicación entra en funcionamiento.

Para conocer los tiempos de latencia se realizaron varias pruebas, a continuación se detalla el resultado de estas.

Operación	Descripción	Tiempo Requerido
Ingreso al Sistema	Solo el personal autorizado puede acceder al sistema (se solicita user y password).	El tiempo requerido en esta operación: de 3 – 5 seg.
Recuperación de compromisos.	La recuperación en primera instancia es solo de los primeros 10 compromisos existentes.	El tiempo requerido en esta operación: De 10 – 15 seg.
Recuperación de los ítems que forman un compromiso	Los ítems que forman un compromiso pueden ser uno o varios. El tiempo de recuperación de estos dependerá de este número.	El tiempo requerido en esta operación: de 5 – 10 seg.
Búsqueda de compromisos	Se solicita el numero de compromiso que se requiere	El tiempo requerido en esta operación: de 3 – 6 seg.

Estos tiempos mucho tienen que ver con la calidad de la red que se esta empleando así como también de obstáculos en la línea de vista entre el punto de acceso y el dispositivo inalámbrico

5. CONCLUCIONES

Al finalizar este proyecto obtuvimos las siguientes conclusiones:

- Creemos que el SCI **mejorará la productividad** de los bodegueros al proveer de una herramienta automatizada que ofrece mejoras el método tradicional de recolección de datos. Con la cual se espera disminuir el tiempo en el que se hace un ingreso de inventarios y eliminar el proceso de trascripción de datos realizado por el personal administrativo de la bodega.

- Se espera una mejora en la forma de **manejar la información**, eliminando procesos redundantes, como el que se realiza actualmente al escribir los datos 2 veces (en papel y en el sistema) y tratando de evitar posibles errores en un ingreso de inventario al proveer de un método de ingreso guiado e intuitivo en la aplicación cliente del SCI.
- Entre las tareas que se hacen en el proceso actual de ingreso de inventarios, se tiene que, el personal administrativo prepara documentos para que los bodegueros escriban la información que recolectan de los inventarios y después transcriben esta información al sistema de la ESPOL. Mediante la utilización del SCI, estas dos tareas son eliminadas, dando como resultado una **disminución en las tareas** que tiene el personal administrativo.
- Se ha logrado la **integración del SCI y de nuevas tecnologías** con la plataforma actual de los sistemas de la ESPOL (SmallTalk), mediante la utilización e integración de estrategias de diseño recomendadas por el personal experto del SCI y el acceso a la base de datos del sistema financiero de la ESPOL, la cual es compartida por los dos sistemas.
- El hecho de haber usado las especificaciones MIDP 2.0 y CLCD 1.1 de J2ME **asegura la portabilidad del sistema** sobre dispositivos móviles.
- Se espera mejorar la gestión realizada por el personal de bodega, al proveer de un **flujo de información más eficiente**, lo cual se logra al permitir que el bodeguero realice consultas y búsquedas de datos de forma automática.
- Se concluye que la los mensajes enviados vía HTTP desde el servidor hacia el dispositivo móvil no deben tener un tamaño muy grande, en promedio para este tipo de aplicaciones se recomienda un tamaño no superior a los 750 bytes, para evitar problemas de alta latencia de la red. En este sentido, **el tamaño de los mensajes esta ligado al número de objetos que se transmiten** a través de la red, para el SCI, se llego a la conclusión que 10 objetos a la vez permiten un nivel de rendimiento aceptable.
- Se llega a la conclusión de que a pesar de los escasos recursos que proveen los dispositivos móviles, **el SCI provee un alto grado de usabilidad mediante la aplicación de metáforas** usadas por el sistema de la ESPOL y el uso de términos familiares al personal de bodega.
- Mediante el uso colores y animaciones en la aplicación cliente se logra brindar **retroalimentación sobre las acciones realizadas** por el usuario sobre determinada pantalla.
- En cuanto a los costos de desarrollo e implementación se concluye que son relativamente bajos y **no implican grandes inversiones en equipo**. Sin embargo el tiempo que se le dedica es alto y esto se debe a la ausencia de expertos locales en los sistemas de la ESPOL.

6. REFERENCIAS

[1] Thierry Violleau and Ray Ortigas, Junio 26, 2003, Supporting Disconnected Operation in Wireless Enterprise Applications, Sun Microsystems, http://java.sun.com/blueprints.

[2] WebSphere Micro Environment for palmOne Devices, MIDP 2.0 Porting

Guide (GM Final), Marzo 2004, PalmOne Inc. www.palmone.com/java
[3] JavaOne y Anthony Scian,Senior Software Developer. Research in Motion, Efficient Java 2 Platform, Micro Edition (J2ME) Programming Tips and Techniques, www.rim.com, http://java.sun.com/javaoneonline/
[4] Jonathan Knudsen July 2001, Wireless Java: Developing with Java 2 Micro Edition, Chapter 10 Performance Tuning
[5] Jonathan Knudsen July 2001, Wireless Java: Developing with Java 2 Micro Edition, Chapter 10 Performance Tuning.
[6] José Pérez, Computación Móvil, http://www.monografias.com/trabajos5/compumo/compumo.shtml
MIDP 2.0 y CLDC 1.1 [7]