Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

IMPLEMENTACIÓN DE APLICACIONES ON-LINE PARA TELÉFONOS BLACKBERRY, A TRAVÉS DE JAVA MICRO EDITION (CLDC 1.1, MIDP 2.0)

Christian Fernando López López

Asesorado por el Ing. David Haroldo Herrera López

Guatemala, julio de 2012

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

IMPLEMENTACIÓN DE APLICACIONES ON-LINE PARA TELÉFONOS BLACKBERRY, A TRAVÉS DE JAVA MICRO EDITION (CLDC 1.1, MIDP 2.0)

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

CHRISTIAN FERNANDO LÓPEZ LÓPEZ
ASESORADO POR EL ING. DAVID HAROLDO HERRERA LÓPEZ

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, JULIO DE 2012

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Murphy Olympo Paiz Recinos
VOCAL I	Ing. Alfredo Enrique Beber Aceituno
VOCAL II	Ing. Pedro Antonio Aguilar Polanco
VOCAL III	Ing. Miguel Ángel Dávila Calderón
VOCAL IV	Br. Juan Carlos Molina Jiménez
VOCAL V	Br. Mario Maldonado Muralles
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Murphy Olympo Paiz Recinos
EXAMINADOR	Ing. Edgar Estuardo Santos Sutuj
EXAMINADOR	Ing. Ludwing Federico Altán Sac
EXAMINADOR	Ing. Pedro Pablo Hernández Ramírez
SECRETARIA	Inga. Marcia Ivónne Véliz Vargas

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

IMPLEMENTACIÓN DE APLICACIONES ON-LINE PARA TELÉFONOS BLACKBERRY, A TRAVÉS DE JAVA MICRO EDITION (CLDC 1.1, MIDP 2.0)

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, con fecha 15 de octubre de 2011.

Christian Fernando López López

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERIA
ESCUELA DE CIENCIAS Y SISTEMAS

Ref. ASESOR 02-02

Guatemala 24 de abril de 2012

Señores
Comisión de Revisión de Trabajos de Graduación
Carrera de Ciencias y Sistemas
Facultad de Ingeniería
Universidad de San Carlos de Guatemala
Guatemala, Ciudad

Respetables Señores:

El motivo de la presente es informarles que como asesor del estudiante Christian Fernando López López he procedido a revisar el trabajo de graduación titulado IMPLEMENTACIÓN DE APLICACIONES ON-LINE PARA TELÉFONOS BLACKBERRY, A TRAVÉS DE JAVA MICRO EDITION (CLDC 1.1, MIDP 2.0) y que de acuerdo a mi criterio el mismo se encuentra concluido y cumple con los objetivos definidos al inicio.

He tenido reuniones periódicas con el estudiante y luego de haber revisado cuidadosamente el trabajo, considero que cumple con los requisitos de calidad y profesionalismo que deben caracterizar a un futuro profesional de la Informática.

Aprovecho para informarle que he leído detenidamente el documento Ref. ASESOR 01-02 y aplicando las recomendaciones que se dan en el mismo procedo a firmar de revisado el trabajo de tesis.

Sin otro particular me suscribo de ustedes.

Atentamente,

Ing. David Herrera López
JEFE DEPTO. DE MANTENIMIENTO DE SISTEMAS
GERENCIA DE INFORMÁTICA

Ing. David Haroldo Herrera López

David Haroldo Herrera López
Ingeniero en Ciencias y Sistemas
Colegiado 8715

Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 30 de Mayo de 2012

Ingeniero
Marlon Antonio Pérez Turk
Director de la Escuela de Ingeniería
En Ciencias y Sistemas

Respetable Ingeniero Pérez:

Por este medio hago de su conocimiento que he revisado el trabajo de graduación del estudiante **CHRISTIAN FERNANDO LÓPEZ LÓPEZ** carné **95-16329,** titulado: **"IMPLEMENTACIÓN DE APLICACIONES ON-LINE PARA TELÉFONOS BLACKBERRY, A TRAVÉS DE JAVA MICRO EDITION (CLDC 1.1, MIDP 2.0)",** y a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo, según el protocolo.

Al agradecer su atención a la presente, aprovecho la oportunidad para suscribirme,

Atentamente,

Ing. Carlos Alfredo Azurdia
Coordinador de Privados
y Revisión de Trabajos de Graduación

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
INGENIERIA EN CIENCIAS Y SISTEMAS
COORDINACION TESIS

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERÌA
ESCUELA DE CIENCIAS Y SISTEMAS
TEL: 24767644

ESCUELA DE CIENCIAS Y SISTEMAS

El Director de la Escuela de Ingeniería en Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer el dictamen del asesor con el visto bueno del revisor y del Licenciado en Letras, del trabajo de graduación titulado ***"IMPLEMENTACIÓN DE APLICACIONES ON-LINE PARA TELÉFONOS BLACKBERRY, A TRAVÉS DE JAVA MICRO EDITION (CLDC 1.1, MIDP 2.0)"*** *presentado por el estudiante CHRISTIAN FERNANDO LÓPEZ LÓPEZ, aprueba el presente trabajo y solicita la autorización del mismo.*

"ID Y ENSEÑAD A TODOS"

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION DE INGENIERIA EN CIENCIAS Y SISTEMAS

Ing. Marlon Antonio Pérez Turk
Director, Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 12 de julio 2012

Universidad de San Carlos
de Guatemala

Facultad de Ingeniería
Decanato

DTG. 354.2012

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería en Ciencias y Sistemas, al trabajo de graduación titulado: **IMPLEMENTACIÓN DE APLICACIONES ON-LINE PARA TELÉFONOS BLACKBERRY, A TRAVÉS DE JAVA MICRO EDITION (CLDC 1.1 MIDP 2.0)**, presentado por el estudiante universitario **Christian Fernando López López**, autoriza la impresión del mismo.

IMPRÍMASE:

Ing. Murphy Olympo Paiz Recinos
Decano

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANO
FACULTAD DE INGENIERIA

Guatemala, 25 de julio de 2012.

/gdech

ACTO QUE DEDICO A:

Dios	Porque únicamente con su ayuda fue posible alcanzar este sueño.
Mi familia	Por el apoyo, paciencia y amor que me brindaron.

AGRADECIMIENTOS A:

Dios	Por brindarme salud, fuerza, sabiduría y paciencia, para permitirme concluir una etapa importante en mi vida.
Mis padres	Rosanio López y Melanie López, porque invirtieron en mí educación y siempre me motivaron a ser mejor.
Mi esposa	Clara Guevara, quien creyó siempre en mí y por ser ayuda en los momentos más difíciles.
Mis hijos	Kate, Fernando y Gisela López, quienes han sido el motor que mueve mí vida.
Mi hermano	Kevin López, quien fue mí ayuda para alcanzar este éxito.
Mi familia	Abuela Elsa Rivera, tíos, primos, en especial a tío René López, por ser ejemplo de buena actitud y valentía.
Mis amigos	A mis líderes, amigos universitarios, de SAT, RIC, ACS y SIAF, porque de ellos he recibido mucho apoyo y enseñanza.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

GLOSARIO

BB Mobile Data Service — Éste es, por defecto, el tipo de transporte de datos que emplea un dispositivo *BlackBerry*.

BlackBerry — *Smartphone* creado por la empresa RIM.

Bytecode — Código intermedio entre el código desarrollado por el usuario y el código que ejecuta la computadora.

Clases abstractas — Clases que definen como se utilizan, sin tener que implementar los métodos, permitiendo que tenga diferentes implementaciones.

Interfaz gráfica — Programa informático que actúa como medio para que el usuario pueda manipular las opciones que ofrece el programa.

Librería — Conjunto de clases que proveen una funcionalidad específica que puede ser reutilizada.

On-Line — Indica que el traslado de información es accesible a través de Internet.

Píxel	Menor unidad homogénea en color que forma parte de una imagen digital, ya sea ésta una fotografía, un fotograma de vídeo o un gráfico.
Recolector de basura	Encargado de administrar correctamente el uso de la memoria volátil.
Smartphone	Teléfono móvil, que incorpora componentes de *hardware* muy potentes que pueden ser comparados con las de una computadora personal.
Tablet	Tipo de computadora portátil, de mayor capacidad de *hardware* que un *smartphone*.
TCP Direct	Tipo de transporte de datos que puede utilizar el Blackberry y se basa en la pila TCP de la operadora de telefonía móvil.
TCP/IP	Protocolo de Control de Transmisión/Protocolo de Internet (en inglés Transmission Control Protocol/Internet Protocol), un sistema de protocolos que hacen posibles servicios de red, tales *Internet*, *E-mail*, y otros.
Thread	Proceso de ejecución independiente con recursos propios.

WAP	Protocolo de aplicaciones inalámbricas, estándar seguro que permite que los usuarios accedan a información de forma instantánea a través de dispositivos inalámbricos como *smartphone*, *tablet*, etc.
WI-FI	Mecanismo de conexión de dispositivos electrónicos de forma inalámbrica.

RESUMEN

Para la ejecución de *software* desde dispositivos pequeños, Java provee un conjunto de librerías y una máquina virtual, denominada Java Micro Edition.

En el capítulo uno se describe la arquitectura de bajo nivel, la cual se divide en cuatro componentes: el sistema operativo, la máquina virtual de Java, la configuración y los perfiles. El sistema operativo, es el corazón del dispositivo, por medio de este *software* se obtiene su correcto funcionamiento. La máquina virtual de Java es el *software* que se encarga de interpretar el *bytecode* de Java, para que el programa sea ejecutado posteriormente por el sistema operativo. La configuración, es un grupo de clases de Java, agrupadas como librerías, estas librerías tienen como objetivo el proporcionar el medio para el desarrollo de aplicaciones. Los perfiles son librerías, que proporcionan una funcionalidad específica, adicional a lo que provee la configuración.

La Connected Limited Device Configuration, es una de las dos configuraciones disponibles en Java Micro Edition que se detalla en el capítulo dos. Sobre esta configuración serán soportados los perfiles que formarán parte de la aplicación.

Su función es la de proveer un mecanismo para el traslado seguro de contenidos sobre diferentes tipos de red, además de proveer las librerías que permitan el desarrollo de aplicaciones de terceros, facilitando al programador el no conocer detalles, como por ejemplo, los protocolos de transmisión de datos para una red.

El Mobile Information Device Profile, el cual se muestra en el capítulo tres, es un perfil que se ejecuta sobre la Connected Limited Device Configuration, proporcionando los componentes necesarios para el desarrollo de aplicaciones Java, para dispositivos móbiles. El programa creado, recibe por nombre MIDlet, y este perfil en conjunto con el sistema operativo, se encargan de manejar los diferentes estados, recursos y eventos que son creados por el MIDlet. Una de las librerías más importantes que provee MIDP, es para la interfaz gráfica de usuario, que es el medio con el cual una persona interactúa con el programa, y la otra es la que permite realizar la conexión a la red, ambas ocultan detalles que facilitan el trabajo de desarrollo al programador.

En el capítulo cuatro se presenta la confirmación que el *smartphone* BlackBerry, soporta la tecnología Java Micro Edition, con CLDC 1.1 y MIDP 2.0, además de permitir utilizar el medio que tiene para conectarse a Internet y así realizar transacciones o consultar información, adicionalmente, que las herramientas de desarrollo facilitan el trabajo al programador para generar *software* de gran utilidad.

OBJETIVOS

General

Aplicar los conceptos de desarrollo de servicios, utilizando la *web* para la transferencia de información a través del *smartphone* BlackBerry, que permitan brindar una solución eficiente para mejorar los procesos de carga y descarga de información en tiempo real.

Específicos

1. Estudiar las herramientas de desarrollo que ofrece Java ME, para teléfonos BlackBerry.

2. Identificar las ventajas y desventajas de utilizar *software* libre.

3. Identificar las posibles dificultades que pueden existir en el desarrollo de *software* de éste tipo.

4. Demostrar la importancia del uso de la tecnología moderna en el mercado guatemalteco.

INTRODUCCIÓN

En la actualidad, ha crecido en gran manera la demanda de teléfonos móviles y/o *smartphones*, y a la vez los usuarios esperan obtener cada día mejores beneficios. Es por ello que los fabricantes de *smartphones* están proporcionando teléfonos con más y mejores características, tales como: Conexión a *internet*, envío y lectura de correos electrónicos y video llamadas. A la vez, están dando la oportunidad para que terceros incorporen *software* adicional al dispositivo.

También es necesario considerar que los usuarios de telefonía móvil, se encuentran en la postura de requerir a los proveedores de servicio telefónico, una mejora de sus servicios tanto en velocidad y cobertura, como también a proporcionar otros servicios como acceso a redes sociales, servicios de localización, mensajes de texto, Internet móvil, *roaming*, etc.

Es por esto que en la actualidad no sólo se producen aplicaciones para servidores o computadoras personales, sino también se pueden crear aplicaciones que pueden ser ejecutadas en otros dispositivos, las cuales deben ser adaptados con base en sus características computacionales, tales como: *tablets*, *smartphones*, teléfonos móviles, televisores o electrodomésticos.

Este trabajo se enfoca en el desarrollo de aplicaciones para BlackBerry, que pueden enviar y recibir información en tiempo real, haciendo uso de uno de los servicios que ofrecen los proveedores de servicio telefónico, Internet móvil, con lo cual permite a personas individuales o empresas, reducir costos y/o aumentar sus ganancias.

Además de la utilización de un lenguaje libre para el desarrollo de este tipo de aplicaciones, como: Java, que se encuentra altamente posicionado.

En el capítulo 1 se encuentra la descripción, historia y características de la tecnología Java Micro Edition.

Los capítulos 2 y 3 se enfocan en la definición de dos de los elementos de la tecnología Java ME, el Connected Limited Device Configuration y el Mobile Information Device Profile, respectivamente. En ambos casos, el enfoque es la explicación de los elementos básicos para la realización de una aplicación real, la cual se encuentra definida en el capítulo 4.

1. JAVA MICRO EDITION

Java Micro Edition es la plataforma de ejecución orientada al desarrollo de aplicaciones para dispositivos pequeños, con capacidades computacionales y gráficas muy reducidas, entre los cuales figuran los teléfonos móviles, PDAs, electrodomésticos inteligentes, y otros.

Debido a la gran variedad de tecnología, *Java* ha desarrollado soluciones para cada ámbito tecnológico, realizando divisiones en ediciones distintas:

- Java Enterprise Edition (Java EE), orientada al entorno empresarial.

- Java Standard Edition (Java SE), orientada al desarrollo de aplicaciones de estaciones de trabajo independientes.

- Java Micro Edition (Java ME), orientada al desarrollo de aplicaciones de dispositivos con capacidades restringidas.

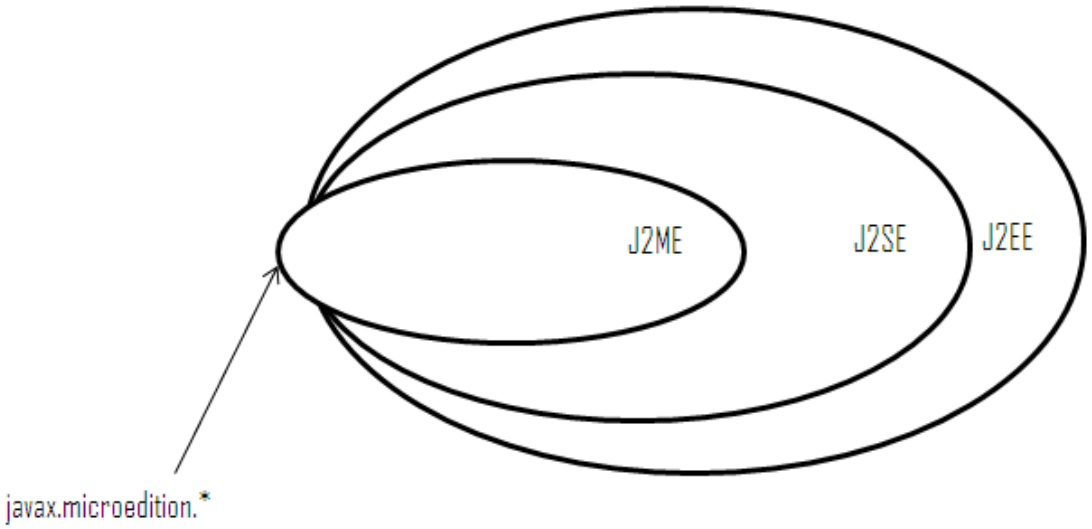
En la figura 1 aparece la arquitectura general para cada una de las ediciones.

Figura 1. **Ediciones Java**

Fuente: http://leo.ugr.es/J2ME/INTRO/intro_2.htm. Consulta: 20 de octure de 2011.

Java ME contiene una parte mínima de las librerías de Java y además una Máquina Virtual de Java (Java Virtual Machine, JVM) diferente a las otras dos ediciones, esto debido a que ocupan espacio de memoria que en los dispositivos pequeños no se encuentra disponible. La figura 2 muestra esta relación.

Figura 2. **Relación de librerías de ediciones Java**



Fuente: http://creacionesmoviles.blogspot.com/2011/12/1-introduccion-j2me-parte-1.html. Consulta: 20 de octubre de 2011.

1.1. Historia

La Java ME fue presentada en 1999 por Sun Microsystems, con el propósito de proveer aplicaciones Java en dispositivos pequeños o de bajos recursos.

En esta primera versión se proporcionó una pequeña Java Virtual Machine, que podía ejecutarse en dispositivos Palm.

La tardía aparición de esta edición se debió a que las necesidades de los usuarios de telefonía móvil han cambiado en estos últimos años, demandando más servicios y prestaciones, tanto de los terminales como de las empresas proveedoras del servicio de telefonía móvil.

1.2. Características

Las características que ofrece esta tecnología son las siguientes:

- Todos los dispositivos tienen un sistema operativo, encargado de ejecutar las instrucciones de los programas.

- Cada dispositivo cuenta con una Java Virtual Machine que se encarga de interpretar el *bytecode* de Java.

- La configuración es un conjunto de clases básicas orientadas a conformar el corazón de las implementaciones, para dispositivos de características específicas.

- Los perfiles son librerías de clases orientadas a implementar funcionalidades específicas de más alto nivel para cada familia específica de dispositivos.

En la figura 3 se presenta la arquitectura de bajo nivel que forma el entorno de ejecución:

Figura 3. **Arquitectura de bajo nivel, ambiente de ejecución**

Fuente: http://caraballomaestre.blogspot.com/2009/05/introduccion-j2me.html. Consulta: 20 de octubre de 2011.

1.2.1. Sistema operativo

El sistema operativo es el *software* principal que se instala en el dispositivo; contiene instrucciones programadas que indican al microprocesador qué hacer, y sobre él se instalan los programas que pueden ser aplicaciones finales, el cual es uno de los objetivos de este documento.

Para las aplicaciones de BlackBerry, el programador no necesita considerar el sistema operativo sobre el cual trabajará, ya que éste es instalado por el fabricante.

El sistema operativo móvil para el *smartphone* BlackBerry se llama BlackBerry OS, que cuenta con las siguientes características:

- Fue desarrollado por Research In Motion (RIM)

- Tiene la capacidad de soportar multitarea, es decir, que permite que varios procesos puedan ser ejecutados a la vez.

- Soporta una variedad de canales de comunicación de entrada al dispositivo fabricado por RIM, tales como la *trackwheel*, *trackball*, *touchpad* y las pantallas táctiles.

- Fue orientado especialmente para el área profesional, como un gestor de correo electrónico y agenda.

- Soporta mensajería instantánea y multimedia

- Debutó en 1999 con su versión 1,0 y en la actualidad se encuentra en la versión 7,0.

1.2.2. Java Virtual Machine (JVM)

Todas las aplicaciones Java son ejecutadas a través de la JVM. La JVM es un programa encargado de interpretar *bytecode* de los programas *Java* precompilados, generando código máquina ejecutable. Además se encarga de efectuar las solicitudes necesarias al sistema operativo y observar las reglas de seguridad y corrección de código definidas para el lenguaje.

Las implementaciones de JVM para las otras ediciones, son muy pesadas en relación a espacio en memoria y en otros requerimientos computacionales. *Java* ME define 2 JVM, cada una adecuada al ámbito de los dispositivos electrónicos, éstas son la KVM y CVM.

1.2.2.1. KVM

Es la JVM más pequeña desarrollada originalmente por Sun. Se trata de una implementación de JVM reducida y especialmente orientada a dispositivos con bajas capacidades computacionales y de memoria. Su nombre proviene de *kilobyte*, haciendo referencia a la baja ocupación de memoria. La KVM está escrita en lenguaje C y fue diseñada con las siguientes características:

- Pequeña: con una carga de memoria entre los 40KB y los 80 KB, dependiendo de la plataforma y las opciones de compilación.

- Alta portabilidad: permitiendo ejecutar las aplicaciones en diversos dispositivos sin realizar cambios al código fuente.

- Modulable: permitiendo dividir las aplicaciones en varias partes.

- Completa y rápida: lo mayor posible y sin sacrificar características para las que fue diseñada.

1.2.2.2. Compact Virtual Machine (CVM)

La CVM ha sido tomada como una JVM que soporta las mismas características que la JVM de Java SE. Está orientada a dispositivos electrónicos con procesadores de 32 *bits* de gama alta y en torno a 2MB o más de memoria RAM. Las características que presenta ésta JVM son:

- Sistema de memoria avanzado

- Tiempo de espera bajo para el recolector de basura

- Separación completa de la JVM del sistema de memoria
- Recolector de basura modularizado
- Portabilidad
- Rápida sincronización
- Ejecución de las clases Java fuera de la memoria de sólo lectura (ROM)
- Soporte de *threads* de lenguajes nativos
- Baja ocupación en memoria de las clases
- Proporciona soporte e interfaces para servicios en sistemas operativos de tiempo real.
- Conversión de *threads* Java a *threads* de lenguajes nativos
- Soporte para todas las características de Java v1,3 y librerías de seguridad, referencias inválidas, Interfaz Nativa de Java (JNI) que permite ejecutar código de lenguaje nativo (por Ejemplo C, C++) desde Java y viceversa, invocación remota de métodos (RMI), Interfaz de depuración de la JVM (JVMDI) que permite la recepción de operaciones mediante el protocolo de depuración de Java y que comunica los cambios de estado producidos en la JVM hacia el servidor.

1.2.3. Configuración

Una configuración es el conjunto mínimo de librerías Java que permiten desarrollar aplicaciones para un grupo particular de dispositivos.

Estas librerías describen las características básicas y comunes a todos los dispositivos, y además maneja su propia JVM.

Existen dos configuraciones en Java ME: La Connected Limited Device Configuration y la Connected Device Configuration.

1.2.3.1. Connected Limited Device Configuration (CLDC)

La CLDC está orientada a dispositivos dotados de conexión y con limitaciones en cuanto a capacidad gráfica, cómputo y memoria. Un ejemplo de estos dispositivos son: teléfonos móviles, *smartphones*, PDAs, organizadores personales, etc.

Algunas de estas restricciones vienen dadas por el uso de la KVM, necesaria al trabajar con la CLDC debido a su pequeño tamaño. Los dispositivos que usan CLDC deben cumplir los siguientes requisitos:

- Disponer entre 160 KB y 512 KB de memoria total disponible
- Procesador de 16 ó 32 *bits* con al menos 25 MHz de velocidad
- Ofrecer bajo consumo, debido a que estos dispositivos trabajan con suministro de energía limitado, normalmente baterías.

- Tener conexión a algún tipo de red, normalmente sin cable, con conexión intermitente y ancho de banda limitado (unos 9,600 bps).

La CLDC aporta las siguientes funcionalidades a los dispositivos:

- Un subconjunto del lenguaje Java y todas las restricciones de su *virtual machine* (KVM).

- Un subconjunto de las librerías Java del núcleo

- Soporte para E/S básica

- Soporte para acceso a redes

- Seguridad

1.2.3.2. Connected Device Configuration (CDC)

La CDC está orientada a dispositivos con pocas capacidades computacionales y de memoria, pero que poseen más recursos que los dispositivos CLDC. Por ejemplo, decodificadores de televisión digital, televisores con internet, algunos electrodomésticos y sistemas de navegación en automóviles. CDC utiliza la Compact Virtual Machine (CVM). Es una *virtual machine* similar en sus características a la de Java SE, pero con limitaciones en el apartado gráfico y de memoria del dispositivo.

La CDC está enfocada a dispositivos con las siguientes capacidades:

- Procesador de 32 *bits*

- Disponer de 2 MB o más de memoria total, incluyendo memoria RAM y ROM.

- Poseer la funcionalidad completa de la JVM

- Conectividad a algún tipo de red

1.2.4. Perfiles

Un perfil es un conjunto de librerías orientado a un ámbito de aplicación determinado.

Los perfiles identifican un grupo de dispositivos por la funcionalidad que proporcionan (electrodomésticos, teléfonos móviles, etc.) y el tipo de aplicaciones que se ejecutarán en ellos. Las librerías de la interfaz gráfica son un componente muy importante en la definición de un perfil.

El perfil establece librerías que definen las características de un dispositivo, mientras que la configuración hace lo propio con una familia de ellos. Esto hace que a la hora de construir una aplicación se cuente tanto con las librerías del perfil como de la configuración. Debe considerarse que un perfil siempre se construye sobre una configuración determinada. Por lo tanto, un perfil es como un conjunto de librerías que dotan a una configuración de funcionalidad específica.

Para la configuración CDC existen los siguientes perfiles:

- Foundation Profile: define una serie de APIs sobre la CDC orientadas a dispositivos que carecen de interfaz gráfica como, decodificadores de televisión digital.

  Si una aplicación requiriera una GUI (*Graphical User Interface* o interface gráfica de usuario), entonces sería necesario un perfil adicional.

- Personal Profile: es un subconjunto de la plataforma Java SE v1.3, y proporciona un entorno con un completo soporte gráfico. El objetivo es el de dotar a la configuración CDC de una interfaz gráfica completa, con capacidades web y soporte de Applets Java (Son aplicaciones hechas en Java que pueden ser ejecutadas desde un navegador de *internet*). Este perfil requiere la implementación del Foundation Profile.

- RMI Profile: este requiere una implementación del Foundation Profile que se construye encima de él. El perfil RMI soporta un subconjunto de las librerías *Java* SE RMI.

  Y para la configuración CLDC existen los siguientes perfiles:

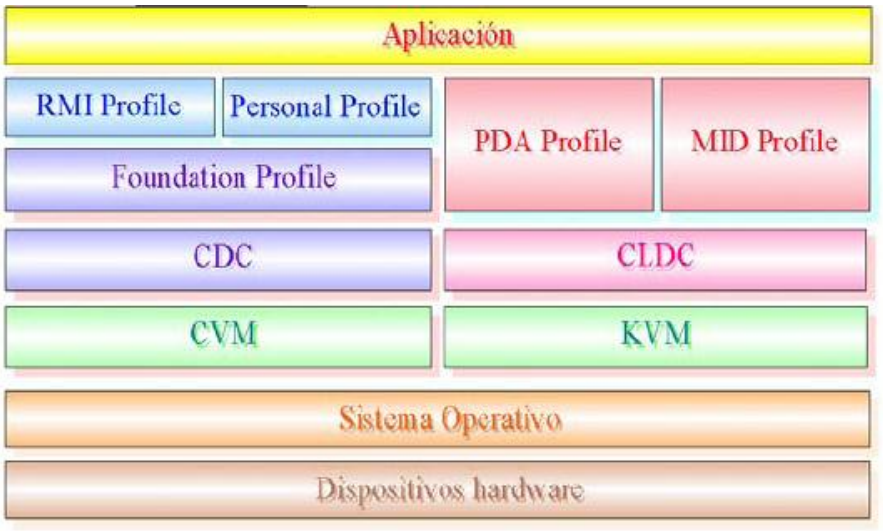
- PDA Profile: está construido sobre CLDC. Pretende abarcar PDAs de gama baja, tipo Palm, con una pantalla y algún tipo de puntero (ratón o lápiz) y una resolución de al menos 20,000 píxeles (al menos 200x100 píxeles) con un factor 2:1.

- Mobile Information Device Profile (MIDP): este perfil está construido sobre la configuración CLDC. Así como CLDC fue la primera configuración definida para Java ME.

MIDP fue el primer perfil definido para esta plataforma y permite la ejecución de aplicaciones java en dispositivos móviles y en el *smartphone* BlackBerry.

En la figura 4 se encuentra la forma cómo están distribuidos los perfiles para cada configuración.

Figura 4. **Arquitectura completa de Java ME**



Fuente: http://caraballomaestre.blogspot.com/2009/05/introduccion-j2me.html. Consulta: 20 de octubre de 2011.

Los *smatphones* BlackBerry son dispositivos que se encuentran soportados de la configuración CLDC, que trabajan sobre la máquina virtual KVM. A través del perfil MIDP es posible el desarrollo de MIDlets que son las aplicaciones que se ejecutan en estos dispositivos.

2. CONNECTED LIMITED DEVICE CONFIGURATION

CLDC es el corazón de la tecnología que será utilizada como base para uno o más perfiles.

Esta configuración es el resultado del trabajo de la Java Community Process (JCP), que se trata de un grupo de expertos en el cual forman parte un número de proveedores industriales.

2.1. Objetivo general

Definir un estándar, para dispositivos pequeños que se encuentran conectados y con recursos limitados.

2.2. Objetivos específicos

La capacidad de permitir fácilmente el extender sus recursos y habilitar el desarrollo de terceras partes son los dos objetivos que a continuación se describen.

2.2.1. Extensibilidad

Uno de los beneficios que ofrece Java para los dispositivos pequeños es la distribución dinámica y segura del contenido y aplicaciones interactivas sobre diferentes tipos de red.

En el pasado, estos dispositivos se fabricaron con características fuertemente definidas y con pocas capacidades de ser extensible. Los fabricantes de estos dispositivos comenzaron a buscar soluciones que permitieran construir dispositivos extensibles. Actualmente, esto ya es una realidad.

2.2.2. Desarrollo de terceras partes

El estándar CLDC proporciona librerías de alto nivel suficientes para el desarrollo de aplicaciones por terceras partes. Por ejemplo, las librerías de red incluidas en CLDC proporcionan al programador una abstracción de alto nivel. Provee la capacidad de transferir archivos, aplicaciones o páginas *web*, en lugar de requerir que el programador conozca los detalles de los protocolos de transmisión para una red específica.

2.3. Requerimientos

Para el correcto funcionamiento de esta configuración es necesario considerar los siguientes requerimientos.

2.3.1. Requerimientos de *hardware*

CLDC permite ejecutarse sobre una amplia variedad de dispositivos pequeños, desde aparatos de comunicación inalámbrica como teléfonos móviles, *smartphones* o *tablets*, hasta organizadores personales y terminales de venta. Las capacidades del *hardware* de estos dispositivos varían y por eso, los requerimientos mínimos solicitados por CLDC son los siguientes:

- 160 KB a 512 KB de memoria total disponible para la plataforma Java. Como mínimo se debe disponer de 128 KB de memoria no volátil para la JVM y las librerías CLDC, y 32 KB de memoria volátil para la JVM en tiempo de ejecución.

- Un procesador de 16 *bits* a 32 *bits*

- Conectividad a algún tipo de red, frecuentemente inalámbrica, con conexión intermitente y ancho de banda limitado (alrededor de 9,600 bps o menos).

2.3.2. Requerimientos de *software*

Al igual que el *hardware*, el *software* incluido en los dispositivos CLDC tiene una gran variación. Por ejemplo, algunos aparatos poseen un sistema operativo completo que soporta múltiples procesos ejecutándose a la vez y con sistema de archivo jerárquico. Otros dispositivos pueden poseer un sistema operativo muy limitado en el que ni siquiera cuentan con un sistema de archivos.

CLDC define las características mínimas de *software*: generalmente, CLDC asume que el dispositivo contiene un mínimo sistema operativo encargado del manejo del *hardware*. Este sistema operativo debe proporcionar, al menos una entidad de planificación para ejecutar la JVM. El sistema operativo no necesita soportar espacios de memoria separados o procesos, ni debe garantizar la planificación de procesos en tiempo real o comportamiento latente.

2.4. Seguridad en CLDC

Derivado de las características que ofrecen estos dispositivos, como la necesidad de descargar aplicaciones y la ejecución de éstas en los dispositivos y, además de almacenar información personal, se hace imprescindible considerar mecanismos de seguridad.

CLDC define un modelo de seguridad que está basado en el modelo de Applets, en el que se permiten ejecutar algunas acciones que se consideran seguras. Este modelo se realiza en dos áreas:

- Seguridad de bajo nivel a través de la máquina virtual: ésta permite ejecutar aplicaciones garantizadas por el verificador de *classfile* de Java, asegurando que el *bytecode* de *Java* y otros ítems almacenados en el *classfile* de *Java* no puedan contener referencias invalidas a localizaciones de memoria o áreas de memoria que se encuentran fuera del espacio de memoria de Java.

- Seguridad a nivel de aplicación: el verificador de *classfile* se encarga únicamente, de la administración adecuada de la memoria, pero existen otras áreas que deben ser ejecutadas con seguridad, por ejemplo, acceso a recursos externos, al sistema de archivos, impresoras, dispositivos infrarrojos, *wifi*, *bluetooth* y redes.

Para permitir acceso controlado a recursos externos de aplicaciones Java, se provee el concepto de administrador de seguridad. El administrador de seguridad es llamado desde varias partes de una aplicación Java o el sistema de ejecución de Java cuando necesita acceso a recursos protegidos. Este modelo provee un mecanismo de seguridad que consiste en nociones formales de permisos, accesos controlados y políticas de seguridad.

2.5. Librerías CLDC

El objetivo general, para el diseño de librerías Java de CLDC es proporcionar un conjunto mínimo de librerías útiles para el desarrollo de aplicaciones y definición de perfiles para una variedad de pequeños dispositivos.

Las librerías de CLDC se encuentran más limitadas, comparada con las otras dos ediciones (Java EE y Java SE), esto porque requieren muchos *megabytes* de memoria para poder ser ejecutados y los dispositivos pequeños no los tienen.

Las librerías de CLDC pueden ser divididas en dos categorías:

- Las clases que son un conjunto de librerías estándar de Java SE
- Las clases que son específicas de CLDC (pero no de Java SE)

2.5.1. Clases heredadas de Java SE

CLDC provee un conjunto de clases heredadas de la plataforma *Java SE*. En total son aproximadamente 37 clases provenientes de los paquetes java.lang, java.net, java.util y java.io.

2.5.1.1. Clase System

Conjunto de clases que se encuentran íntimamente relacionadas con la máquina virtual de Java.

2.5.1.2. Clases de tipos de datos

Estas clases incluyen tipos de datos básicos que el paquete java.lang soporta.

2.5.1.3. Clases Collection

Es un conjunto de clases que ofrece manipulación de un grupo de objetos, tratados como elementos propios, se conocen también como arreglos.

2.5.1.4. Clases de entrada/salida

Conjunto de clases que ofrecen el manejo de datos para entrada y salida.

2.5.1.5. Clases de calendario y tiempo

Son clases abstractas utilizadas para realizar conversiones entre objetos tipo *Date* y un conjunto de campos enteros como año, mes, día, hora, entre otros.

2.5.1.6. Clases utilitarias

CLDC provee dos clases utilitarias que son la java.util.Random que provee un generador simple de números aleatorios y la clase java.lang.Math que provee métodos de mínimo, máximo y valor absoluto que son frecuentemente utilizadas.

2.5.1.7. Clases de errores y excepciones

Son las que permiten obtener los errores que pueden ocurrir dentro de la aplicación.

2.5.2. Clases propietarias de CLDC

La plataforma Java SE contiene los paquetes java.io y java.net encargados de las operaciones de entrada y salida. Debido a las limitaciones de memoria de los dispositivos, no es posible incluir el total de clases dentro de CLDC.

Las librerías heredadas de Java SE no incluyen clases relacionadas con la entrada y salida de archivos, esto debido a la gran variedad de dispositivos que abarca CLDC, donde estas clases no son necesarias para algunos dispositivos que no soportan el almacenamiento de archivos. Tampoco se incluyen las clases del paquete java.net, basado en comunicaciones TCP/IP, ya que los dispositivos CLDC no están obligados a basarse en este protocolo de comunicación.

Es por esto que, CLDC posee un conjunto de clases más genérico para la entrada y salida y la conexión en red que recibe el nombre de Generic Connection Framework (GCF), que están incluidas en el paquete javax.microedition.io.

3. MOBILE INFORMATION DEVICE PROFILE

MIDP está diseñado para operar en la parte superior de la CLDC, el cual proporciona los paquetes, clases e interfaces necesarios para el desarrollo de aplicaciones Java, que pueden ser utilizados en dispositivos móviles. Los dispositivos que soportan éste perfil se conocen como MIDs.

3.1. Requerimientos de MIDP

MIDP siendo el perfil para el desarrollo de aplicaciones en dispositivos móviles, necesita de los siguientes requerimientos:

3.1.1. Requerimientos de *hardware*

Los MIDs deben tener como mínimo, las siguientes características:

- Pantalla:
    - Tamaño: 96 X 54 píxeles (anchura, altura)
    - Colores: 1 *bit* de profundidad (blanco y negro)
    - Forma del píxel (proporción de aspecto): aproximadamente la relación de altura y anchura de 1:1.

- Entrada:

    - Uno o más de los siguientes mecanismos de entrada para el usuario: Un teclado de una mano, un teclado de dos manos o una pantalla táctil, cada uno de estos como parte del dispositivo.

- Memoria:

    - 128 KB de memoria no volátil para las librerías de MIDP

    - 32 KB de memoria volátil para la ejecución de aplicaciones

    - 8 KB de memoria no volátil para datos persistentes, que serán utilizados por las aplicaciones creadas, para almacenamiento de información.

- Red:

    - De dos vías, inalámbrico, puede ser por medio de WI-FI, WAP, BlackBerry Mobile Data Service o TCP Direct.

3.1.2. Requerimientos de *software*

Para los dispositivos con las características anteriormente indicadas deben tener los siguientes requerimientos mínimos de *software*:

- Un *kernel* mínimo, para manejar las características de *hardware* a bajo nivel.

- Un mecanismo que lea y escriba en memoria persistente

- Acceso de lectura y escritura a la red inalámbrica del dispositivo

- Capacidad mínima de escritura a una pantalla gráfica, estos valores están definidos en los requerimientos de *hardware*.

- Un mecanismo para capturar entradas del usuario de cualquiera de los tres mecanismos de entrada, mencionados en los requerimientos de *hardware*.

- Un mecanismo para manejar el ciclo de vida de las aplicaciones del dispositivo.

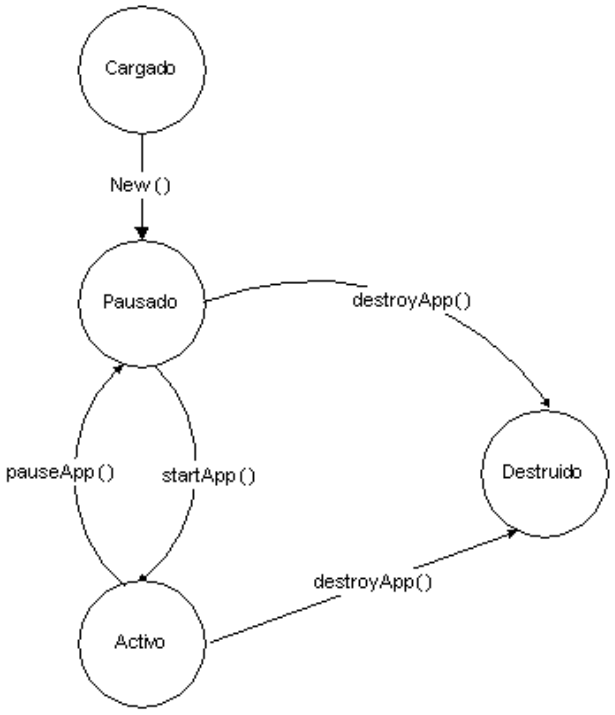
3.2. MIDlet

Se denomina MIDlet al *software* Java realizado usando la especificación MIDP, es decir, un MIDlet será una aplicación que puede usar las funcionalidades aportadas por MIDP y por CLDC.

Los métodos de esta clase son la interfaz del MIDlet que permiten a la aplicación crear, empezar, parar y destruir un MIDlet, y además, permiten al dispositivo poder manejar múltiples MIDlets, sin tener que estar todos ejecutándose en el mismo entorno. El sistema del dispositivo puede seleccionar qué MIDlet está activo, usando los métodos correspondientes para empezar o pausar.

El ciclo de vida de un MIDlet se compone de los siguientes estados: pausado, activo o destruido. La figura 5 muestra cómo se traslada de un estado a otro:

Figura 5. **Ciclo de vida de un MIDlet**



Fuente: http://leo.ugr.es/J2ME/MIDP/aplicaciones.htm. Consulta: 18 de noviembre de 2011.

Cuando un MIDlet es cargado en memoria (método New()), inicialmente pasa al estado pausado, entonces se realiza la inicialización de la clase (método startApp()). Si el MIDlet lanza una excepción durante la ejecución de su constructor, se destruye. El MIDlet puede pasar de activo a pausado, cuando por ejemplo, un teléfono móvil recibe una llamada y el MIDlet debe liberar los recursos como pantalla y teclado.

3.2.1. El administrador de aplicaciones

Derivado que, en algunos MIDs no existen líneas de comando para poder ejecutar las aplicaciones, reside en él un *software* llamado administrador de aplicaciones o Application Management System (AMD), que se encarga de almacenar y ejecutar los MIDlets, además de administrar los recursos que éstos ocupan y administrar su ciclo de vida.

3.3. Interfaces gráficas de usuario

La especificación MIDP creó un conjunto de clases adaptadas a la interfaz gráfica de los dispositivos móviles, es pequeño y bastante sencillo de utilizar.

Dentro de MIDP existen dos alternativas de realizar el interfaz gráfico de los MIDlets: con la librería de interfaces de usuario de alto nivel o la librería de interfaces de usuario de bajo nivel.

Con la librería de interfaz de usuario de alto nivel, es la máquina virtual del dispositivo quien se encarga de colocar los componentes, las barras de desplazamiento, la navegación y características visuales como: el color, las formas, fuentes y los elementos a visualizar. Además provee un sistema de entrada de alto nivel asociado.

Con la librería de interfaz de usuario de bajo nivel, la aplicación tiene un control mayor sobre la pantalla. Esta librería es de mayor utilidad para el tipo de aplicaciones que necesitan un control y precisión absoluto al momento de dibujar elementos en pantalla, por ejemplo, los juegos. Pero, obviamente un mayor control, significa también, una mayor responsabilidad: hay que dibujar todo lo que aparece en pantalla e interpretar cualquier entrada del usuario. Evidentemente, esta librería también tiene asociado un sistema de entrada de bajo nivel.

También se debe considerar que cualquier aplicación que use esta librería podría dejar de ser portable o más complejo y laborioso su desarrollo, debido a que tiene que comprobar los recursos del dispositivo, por ejemplo, debe saber el tamaño de la pantalla para no dibujar píxeles más allá de los límites.

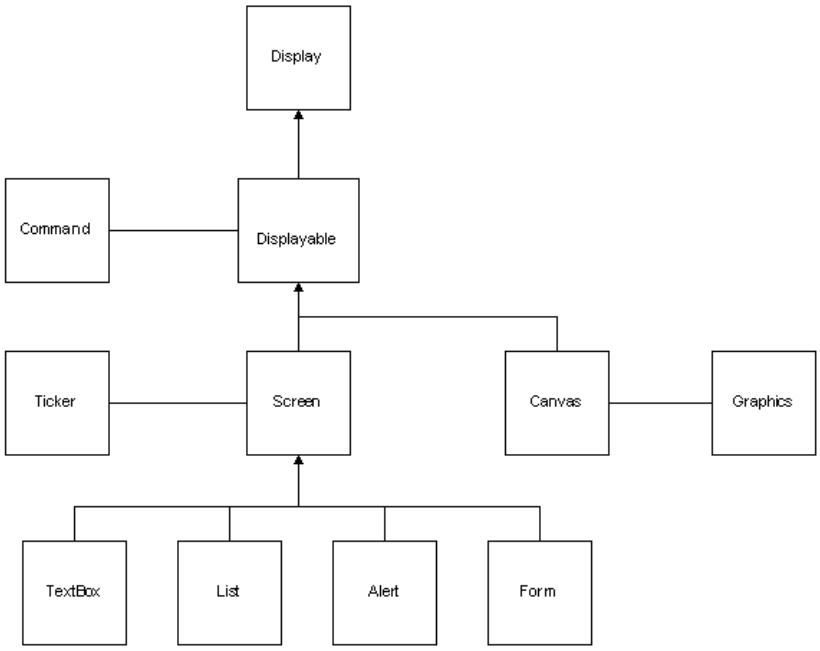
Las librerías de interfaz de usuario, tanto de alto como de bajo nivel se encuentran en el paquete javax.microedition.lcdui.

3.3.1. Clases Display y Displayable

La clase javax.microedition.lcdui.Display representa el controlador lógico de la pantalla del dispositivo, donde el MIDlet colocará su interfaz, por lo que esta clase es responsable de controlar la pantalla y la interacción con el usuario.

Displayable es la clase base para los interfaces de usuario de un MIDlet, pero por ser abstracta no puede utilizarse. Las clases que pueden utilizarse para realizar interfaces de usuario son sus clases derivadas, cuya jerarquía se muestra en la figura 6.

Figura 6. **Jerarquía de clases derivadas de Display**



Fuente: http://leo.ugr.es/J2ME/MIDP/graficos.htm. Consulta: 30 de noviembre de 2011.

Las dos clases que descienden de Displayable: Screen y Canvas, son también abstractas. Estas clases corresponden a las librerías de alto y bajo nivel del interfaz de usuario, respectivamente.

Este documento está orientado a proporcionar una introducción al desarrollo de aplicaciones para BlackBerry, por lo que únicamente se explica la librería de alto nivel.

La clase Screen es la base, la cual es utilizada en la librería de alto nivel del interfaz de usuario. Extiende la clase Displayable e implementa el *ticker*, que es una cadena de texto que se desplaza por la pantalla de derecha a izquierda.

Las subclases Screen pueden agruparse en dos tipos: el primer tipo encapsula completamente los ítems del interfaz de usuario siguiendo una estructura predefinida. Dentro de este tipo existen las clases: TextBox, List y Alert, este tipo no permite que se añadan o quiten otra clase de componentes. El segundo tipo: la clase Form que permite a las aplicaciones añadir o quitar *items* de la pantalla, tal y como desee el programador del MIDlet.

3.3.2. Clase Command e Interface CommandListener

Un objeto de la clase Command mantiene información sobre un evento. Una analogía de éste es un botón de Windows. Generalmente, son implementados en los MIDlets cuando es necesario detectar y ejecutar una acción.

Los tipos de Command que existen son los siguientes:

- *Ok*: verifica que se pueda hacer una acción
- *Cancel*: cancela una acción
- *Stop*: detiene una acción
- *Exit*: salir del MIDlet
- *Back*: envía al usuario a la pantalla anterior
- *Help*: solicita ayuda

- *Item*: un comando específico de la aplicación que es relativo a un determinado ítem del Screen.

- *Screen*: un comando específico de la aplicación que es relativo al actual Screen.

Para que el objeto Command realice una acción de acuerdo a su tipo, es necesario implementar la interfaz CommandListener. La interfaz CommandListener incluye el método commandAction(Command c, Displayable d) en donde se indica la acción que debe realizar cuando se produzca un evento en el Command que se encuentra en el objeto Displayable.

3.3.3. Clase Alert

El objeto Alert representa un aviso. Normalmente se utiliza para indicar al usuario un evento, por ejemplo, un error. Un Alert está formado por un título, texto e imágenes (este último es optativo).

El tiempo que el aviso permanecerá en pantalla puede ser uno de los siguientes:

- *Modal*: el aviso permanece un tiempo indeterminado hasta que es cancelada por el usuario.

- *No Modal*: el aviso permanece un tiempo definido por el programador

Los tipos de alerta que pueden ser definidos son los siguientes:

- *Alarm*: aviso de una petición previa

- *Confirmation*: indica la aceptación de una acción

- *Error*: indica que ha ocurrido un error

- *Info*: indica algún tipo de información

- *Warning*: indica que puede ocurrir algún problema

3.3.4. La clase List

Permite construir aplicaciones donde es necesario mostrar una lista de opciones, implementa la interfaz Choice, y esto da la posibilidad de crear 3 tipos distintos de listas, que son:

- *Exclusive*: lista en la que un sólo elemento puede ser seleccionado a la vez.

- *Implicit*: lista en la que la selección de un elemento genera un evento

- *Multiple*: lista en la que cualquier número de elementos pueden ser seleccionados al mismo tiempo.

3.3.5. La clase *TextBox*

Un TextBox es un objeto que permite editar texto. También permite colocar restricciones al texto, las cuales se encuentran en la clase TextField, íntimamente relacionada con TextBox. Las restricciones son:

- *Any*: acepta editar cualquier carácter

- *Constraint_Mask*: se utiliza cuando se necesita determinar el valor actual de las restricciones.

- *Emailaddr*: permite caracteres válidos para direcciones de correo electrónico.

- *Numeric*: permite sólo números

- *Password*: oculta los caracteres introducidos mediante una máscara para proporcionar privacidad.

- *PhoneNumber*: permite caracteres válidos sólo para números telefónicos

- *Url*: permite caracteres válidos sólo para direcciones *URL*

3.3.6. Clases Form e Item

Form es un contenedor de ítems del interfaz gráfico, donde todos los ítems que se pueden añadir a Form descienden de la clase Item, la cual contiene las siguientes:

- ImageItem: clase que permite colocar imágenes en el interfaz gráfico

- StringItem: ítem para colocar una cadena en el interfaz

- TextField: un objeto de edición de texto

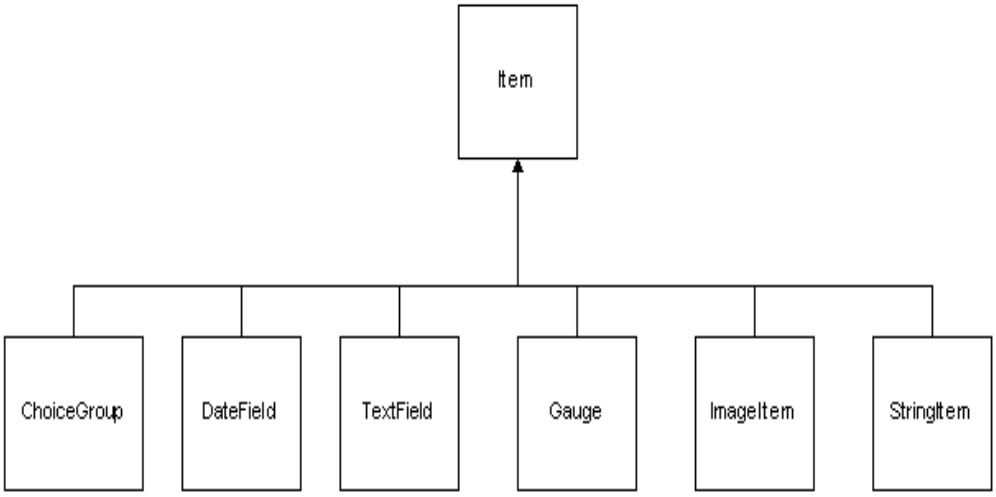
- DateField: una versión de TextField utilizada para la introducción de fechas.

- ChoiceGroup: un componente que proporciona una lista de opciones exclusivas o no.

- Gauge: un ítem que es utilizado para ver en la pantalla del MID, el progreso de una operación.

La figura 7 muestra la jerarquía de la clase Item.

Figura 7. **Jerarquía de clases derivadas de Item**



Fuente: http://jcp.org/en/jsr/detail?id=037. Consulta: 9 de enero de 2012.

3.4. Comunicaciones

Las comunicaciones a través de una conexión de red son fundamentales en cualquier dispositivo móvil, esto porque es necesario el intercambio de información, que al mismo tiempo ofrece ventajas sobre cualquier otro dispositivo con comunicación a red, asimismo, se realiza a través de un aparato pequeño y que actualmente es factible adquirirlo.

En CLDC se implementan conexiones genéricas y en MIDP y librerías opcionales se realizan los protocolos concretos.

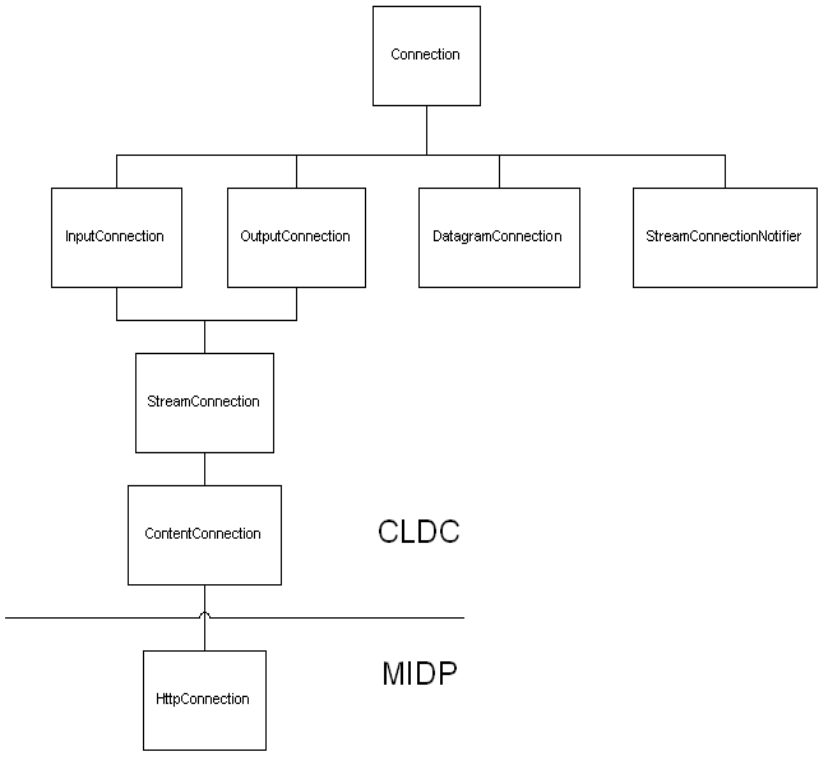
Debido a las dificultades para proveer soporte en funciones de red a nivel de configuración por la variedad en los dispositivos, el CLDC delega esta parte a los perfiles. Para realizar esta delegación de forma satisfactoria, el CLDC ofrece un marco general de trabajo en red, conocido como el Generic Connection Framework.

3.4.1. Generic Connection Framework (GCF) de CLDC

El GCF permite establecer conexiones de red, independientemente del tipo de red del móvil y está compuesto básicamente por una serie de interfaces de conexión junto con una clase *Conector* que es usada para establecer las diferentes conexiones. Todo se encuentra dentro del paquete javax.microedition.io.

La figura 8 muestra la jerarquía de la clase Connection de CLDC y MIDP.

Figura 8. **Jerarquía de la clase Connection de CLDC y MIDP**



Fuente: http://www.it.uc3m.es/celeste/docencia/j2me/tutoriales/midp2_0/PracticaIO/. Consulta: 23 de enero de 2012.

Las interfaces del paquete javax.microedition.io son:

- Connection: conexión que puede ser abierta y cerrada
- DatagramConnection: conexión para manejar comunicaciones orientadas a paquetes.
- InputConnection: conexión de entrada para las comunicaciones del dispositivo.

- OutputConnection: conexión de salida para las comunicaciones del dispositivo.

- StreamConnectionNotifier: conexión especial para notificaciones, que es usada para esperar que se establezca una conexión.

- StreamConnection: conexión bidireccional para las comunicaciones del dispositivo.

- ContentConnection: flujo (*stream*) de conexión que proporciona acceso a datos web.

- HttpConnection: librería MIDP parte de la base GCF, creando la interfaz HttpConnection, que implementa el protocolo HTTP 1.1

3.4.2. Conectividad en MIDP

MIDP realizó una ampliación al GCF, esto derivado a que el GCF no implementa ningún protocolo. MIDP exige que todas las implementaciones incluyan soporte al protocolo HTTP 1.1, a través de la interfaz HttpConnection, permitiendo así la conectividad en los MIDlets.

El protocolo HTTP es de tipo petición/respuesta. El funcionamiento de este protocolo es el siguiente: el cliente realiza una petición al servidor y espera a que éste le envíe una respuesta. Derivado de esto, la implementación de este protocolo pasa por tres estados en forma consecutiva que son:

- De establecimiento de conexión
- De conexión
- Sin conexión

3.4.2.1. Estado de establecimiento de conexión

En este estado se establecen los parámetros de la comunicación. El cliente indica el tipo de petición a realizar, que puede ser:

- *Get*: petición de información en la que los datos se envían como parte del URL.
- *Post*: petición de información en la que los datos se envían aparte en un stream.
- *Head*: petición de meta-información

Además se puede indicar información adicional a la petición que realizará al servidor, como el formato, idioma, etc. Estos campos forman parte de la cabecera de la petición.

3.4.2.2. Estado de conexión

En este estado realiza el intercambio de información entre el cliente y el servidor. La respuesta del servidor se compone de:

- Línea de estado
- Cabecera
- Cuerpo de la respuesta

3.4.2.3. Estado de cierre

La conexión entra en este estado una vez que se termina la comunicación entre el cliente y el servidor.

4. DESARROLLO DE APLICACIÓN

Para demostrar el correcto funcionamiento de esta tecnología, en este capítulo se describen los pasos para desarrollar un MIDlet, el cual funcionará en un dispositivo *smartphone* BlackBerry, y proporcionará una herramienta de utilidad para las personas que necesiten consultar datos de un vehículo.

- Situación: al realizar la compra de un vehículo usado, los compradores no cuentan con una herramienta que les permita confirmar que el vehículo que quieren adquirir es de dudosa procedencia. Al igual para los agentes de la Policía Nacional Civil, cuando realizan operativos y detienen los vehículos, para confirmar que no existe problema con ellos.

- Problema: el que existe a raíz de esta falta de información en el lugar, es que en la actualidad hay una gran cantidad de personas que han sido estafadas, por realizar compras de vehículos que anteriormente fueron robados, se encuentra una página de consulta, pero está hecha para ser consultada por medio de una computadora y no así desde un *smartphone*, lo que limita el uso de esta consulta.

- Solución: crear un MIDlet que obtenga la información pública que ofrece la Superintendencia de Administración Tributaria (SAT), específicamente en el servicio en línea que permite consultar información del vehículo, el cual será adaptado para que pueda mostrarse de forma personalizada para dispositivos BlackBerry y así el usuario puede confirmar con base en los datos de la tarjeta de circulación y a las características del vehículo que todo se encuentra de una forma correcta.

- Beneficiarios: la aplicación puede ser utilizada por cualquier persona que quiera realizar una compra de un vehículo usado o también, para los agentes de la Policía Nacional Civil que realizan operativos en carretera.

  Para el desarrollo de la aplicación es necesario tener la plataforma de desarrollo que permita la generación de MIDlets.

4.1. Requerimientos funcionales

Para el correcto funcionamiento de la aplicación, el BlackBerry deberá contar con una máquina virtual de Java que permita la ejecución de aplicaciones CLDC 1.1 y MIDP 2.0. Además, el proveedor del servicio de telefonía debe proporcionar el acceso a Internet, para poder realizar la conexión al servidor *web* de la Superintendencia de Administración Tributaria (SAT).

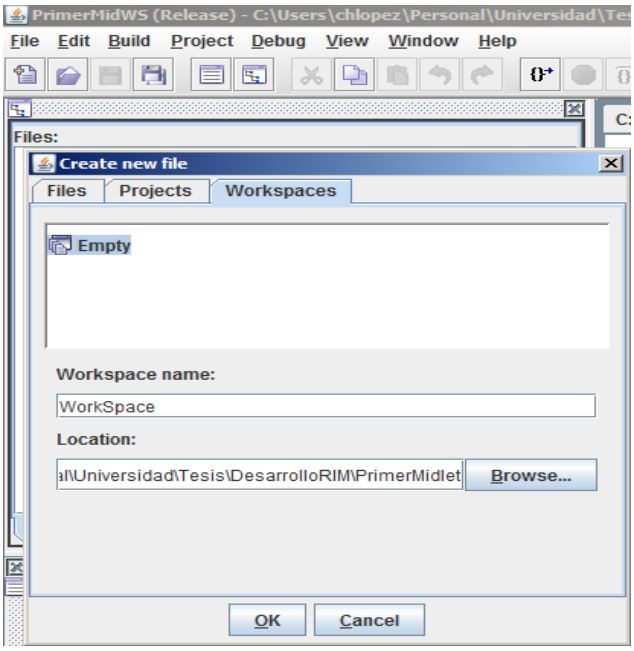
4.2. Herramientas de desarrollo

El único componente a utilizar para el desarrollo de aplicaciones para BlackBerry, es el BlackBerry Java Development Environment, que es un ambiente completo de desarrollo integrado y herramienta de simulación para la construcción de aplicaciones Java ME basadas en *smartphones* BlackBerry. Actualmente tienen disponible hasta la versión 7,1 y puede ser descargado desde esta dirección URL http://us.blackberry.com/developers/javaappdev /javadevenv.jsp, y posteriormente instalado en la computadora a utilizar.

4.3. Preparación del entorno de trabajo

Luego de haber instalado todas las herramientas de desarrollo, debe ser creado un *workspace* dentro del BlackBerry JDE, el cual es un entorno de trabajo que crea una jerarquía de directorios dentro del directorio de aplicaciones *default*. En la figura 9 aparece la vista de la creación del *workspace*.

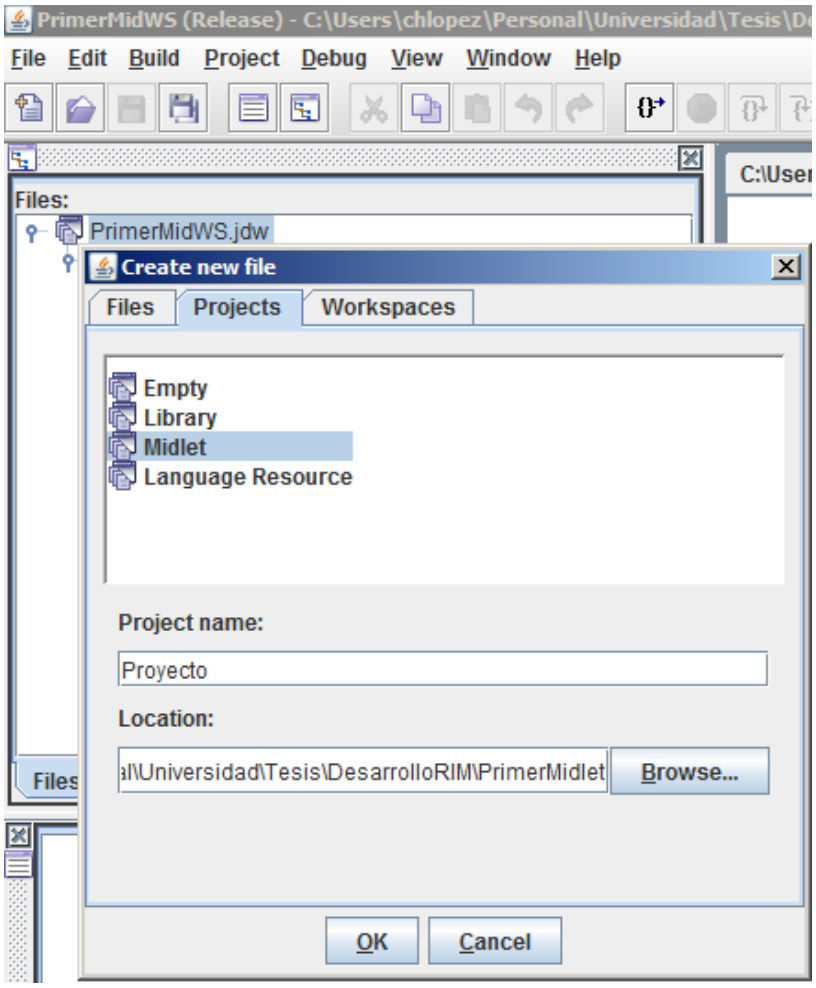
Figura 9. **Creación del *workspace***



Fuente: BlackBerry JDE.

Posteriormente es necesario crear un proyecto, que para la aplicación a desarrollar será llamado proyecto, solamente es necesario indicar que el tipo del proyecto es MIDlet. En la figura 10 aparece una vista de la creación del proyecto.

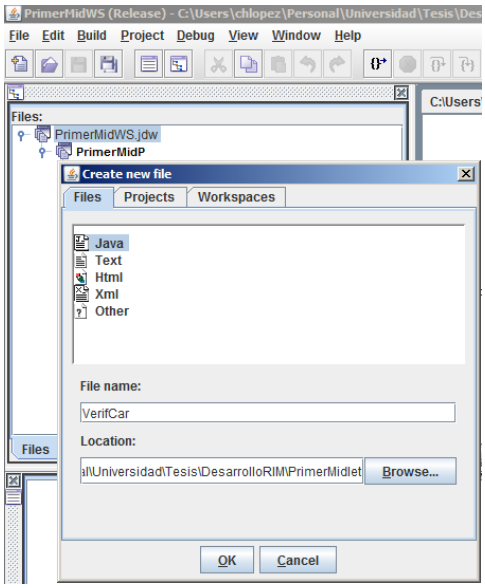
Figura 10. **Creación del Proyecto**



Fuente: BlackBerry JDE.

Luego es necesario crear la clase principal, ésta es la que establece el nombre de la clase MIDlet que será ejecutada en el dispositivo, que en este caso se llamará igual VerifCar. En la figura 11 aparece una vista de la creación de esta clase.

Figura 11. **Creación de la clase MIDlet**



Fuente: BlackBerry JDE.

4.4. Desarrollo de la aplicación

Ahora ya se encuentra listo el entorno de desarrollo, por lo que puede ser creado el archivo código fuente, a través del BlackBerry JDE.

La aplicación *VerifCar* utilizará dos pantallas, una para la captura de parámetros y la otra para mostrar el resultado.

Para la primera pantalla serán necesarios tres parámetros y dos botones:

- Campo NIT del dueño del vehículo
- Campo placa del vehículo

- Campo uso del vehículo

- Botón para revisar el vehículo

- Botón para salir de la aplicación

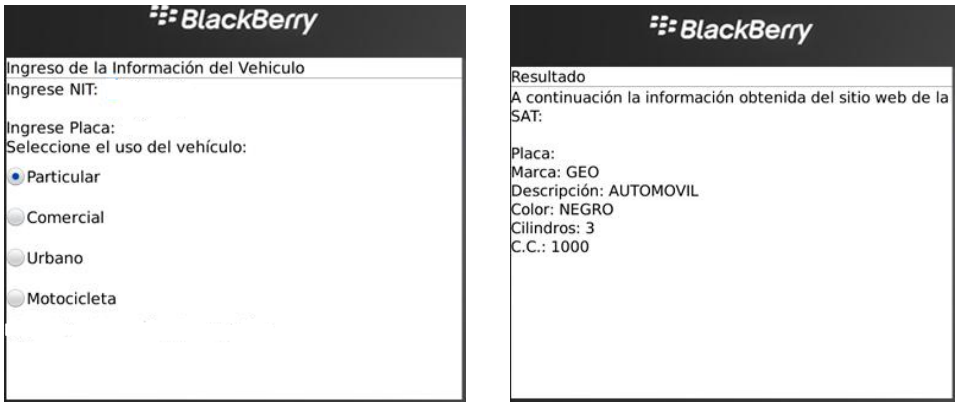
La segunda pantalla utilizará dos componentes:

- Campo para mostrar al usuario el resultado de la verificación

- Botón para regresar a la pantalla de captura, para permitirle al usuario realizar otra revisión

Las pantallas se visualizarán tal y como se muestra en la figura 12.

Figura 12. **Pantalla de captura y resultado**



Fuente: BlackBerry JDE.

Dentro de los MIDlets las pantallas pueden ser creadas a través de formas, en la figura 13 se encuentra el código para la creación de éstas:

Figura 13. **Creación de Pantallas**

```
Form inicioScreen = new Form("Ingreso de la Información del Vehiculo");
Form resultadoScreen = new Form("Resultado");
```

Fuente: BlackBerry JDE.

Para la creación de formas o pantallas, únicamente es necesario un parámetro, que es la etiqueta de la pantalla. En la primer sentencia se encuentra la creación de la pantalla inicial que es la de captura, y en la segunda la creación de la pantalla de resultado.

Los campos editables serán de tipo TextField, ya que estos objetos permiten el ingreso de información. El ejemplo para la creación de los objetos de éste tipo, aparece en la figura 14:

Figura 14. **Creación de objeto TextField**

```
TextField variableTextField = new TextField(“etiqueta”,”valor  inicial”,tamaño,tipo);
```

Fuente: BlackBerry JDE.

Los parámetros del objeto TextField son: la etiqueta que identifica a cada campo, un valor inicial, el tamaño y el tipo de información que permitirá ingresar (los tipos fueron listados en el capítulo 3). El código de los tres campos editables que forman parte de la pantalla de captura, se encuentra en la figura 15:

Figura 15. **Campos editables de la pantalla de captura**

```
nitTextField = new TextField("NIT", "", 10, TextField.ANY);
placaTextField = new TextField("Placa", "", 5, TextField.ANY);
```

Fuente: BlackBerry JDE.

Es necesario crear un último campo de captura, el cual debe ser seleccionado de forma exclusiva, es decir, que sólo una opción del listado puede ser utilizado a la vez. Este campo es el uso del vehículo, por lo que se hace uso del componente ChoiceGroup; en la figura 16 se encuentra el comando para crearlo:

Figura 16. **Campo de captura en listado exclusivo**

```
ChoiceGroup variableTextField = new ChoiceGroup (“etiqueta”,tipo);
```

Fuente: BlackBerry JDE.

Los parámetros del objeto ChoiceGroup son: la etiqueta que identifica al campo y el tipo de información que permitirá ingresar (los tipos fueron listados en el capítulo 3). La configuración del campo ChoiceGroup aparece en la figura 17:

Figura 17. **Configuración del campo ChoiceGroup**

```
ChoiceGroup cgUso = new ChoiceGroup("Seleccione el uso del vehículo:", Choice.EXCLUSIVE);
```

Fuente: BlackBerry JDE.

La instrucción para la creación de botón, se describe en la figura 18:

Figura 18. **Creación de botón**

```
Command variableCommand= new Command("etiqueta", tipo, prioridad);
```

Fuente: BlackBerry JDE.

Los parámetros son: la etiqueta que identificará a cada botón dentro de la pantalla, el tipo de objeto Command (los tipos fueron listados en el capítulo 3) y la prioridad, que para la aplicación a desarrollar todos los objetos son de la misma prioridad. El código de los tres botones que son parte de la pantalla, aparece en la figura 19:

Figura 19. **Definición de botones de la pantalla**

```
Command validarCommand = new Command("REVISAR", Command.OK, 1);
Command salirCommand = new Command("SALIR", Command.EXIT, 1);
Command regresarCommand = new Command("REGRESAR", Command.OK, 1);
```

Fuente: BlackBerry JDE.

El último componente, que formará parte de las pantallas es una etiqueta donde se mostrará el resultado de la verificación del vehículo, que para este caso será de tipo StringItem. La definición se encuentra en la figura 20.

Figura 20. **Definición StringItem**

```
StringItem variableStringItem= new StringItem("etiqueta", valor_inicial);
```

Fuente: BlackBerry JDE.

Los parámetros de este objeto son: una etiqueta y su valor inicial. Dentro de la aplicación queda como aparece en la figura 21:

Figura 21. **Definición de StringItem en la aplicación**

```
private StringItem resultadoStringItem = new StringItem(null, "");
```

Fuente: BlackBerry JDE.

De acuerdo a como se indica en la figura 21, no tiene etiqueta, ni tampoco un valor inicial, ya que su valor se asigna luego de realizar la validación de la placa.

Dentro del constructor del MIDlet, el primer paso que debe realizarse es obtener el manejador de la pantalla y las entradas, a través de la creación de la instancia del objeto Display. Esto se puede observar en la figura 22:

Figura 22. **Definición de objeto Display**

```
Display miDisplay=Display.getDisplay(this);
```

Fuente: BlackBerry JDE.

También, dentro del constructor del MIDlet se puede agregar las opciones del ChoiceGroup, incorporar todos los componentes que forman parte de las pantallas a su respectiva forma, para hacer esto se utilizan los métodos addCommand y append. El primero incorpora a la pantalla los objetos Command o botones y el segundo incorpora objetos que derivan de la clase Item.

Figura 23. **Asignación de componentes a las pantallas**

```
//Agregar opciones del uso del vehículo
     cgUso.append("Particular", null);
     cgUso.append("Comercial", null);
     cgUso.append("Urbano", null);
     cgUso.append("Motocicleta", null);
//Sentencias para agregar los objetos que forman parte de la pantalla de captura
    inicioScreen.append(nitTextField);
    inicioScreen.append(placaTextField);
    inicioScreen.append(cgUso);
    inicioScreen.addCommand(validarCommand);
    inicioScreen.addCommand(salirCommand);

//Sentencias para agregar los objetos que forman parte de la pantalla de resultado
     resultadoScreen.append(resultadoStringItem);
     resultadoScreen.addCommand(regresarCommand);
```

Fuente: BlackBerry JDE.

Derivado a que solamente una pantalla puede encontrarse activa, y la primera que debe aparecer es la de captura, se puede hacer la activación dentro del método startApp(), ya que este método se activa en forma automática luego de la ejecución del constructor. En la figura 24 aparece el código para la activación de la pantalla:

Figura 24. **Activación de pantalla**

```
public void startApp() {
  miDisplay.setCurrent(inicioScreen);
}
```

Fuente: BlackBerry JDE.

Ya se encuentran concluidas las pantallas que forman parte de la aplicación, pero los botones necesitan la implementación de sus eventos. Todo MIDlet que incluya Commands, debe implementar la interfaz CommandListener. Esta interfaz solamente incluye el método commandAction, en donde se indica la acción asociada a cada Command que se encuentre en un determinado objeto Displayable. La implementación de este método se puede observar en la figura 25.

Figura 25. **Implementación de los botones**

```
public void commandAction(Command c, Displayable s) {
    if (c == validarCommand) { //Para el evento validar factura
      //Realizar la validación de la factura
      try{
         ConectarServidor();
      }
      catch (Exception e)
      {
        resultadoString="Msg: " + e.toString();
      }
      //Mostrar el resultado de la verificación
      mostrarResultado();
    } else if (c == salirCommand) { //Para el evento salir
        destroyApp(false);  // Terminar MIDlet y liberar recursos
    } else if (c == regresarCommand) { //Para el evento regresar
        // Regresare a la pantalla de captura
        miDisplay.setCurrent(inicioScreen);
     }
}
```

Fuente: BlackBerry JDE.

La última parte, y la más importante es la conexión por el protocolo HTTP al servidor, que será implementada en el método ConectarServidor.

Figura 26. **Implementación del método ConectarServidor**

```
  private void ConectarServidor() throws IOException{
    HttpConnection http = null;
    InputStream iStrm = null;
    String // Armado de URL
    private String
wap=";deviceside=true;WapGatewayIP=216.230.133.66;WapGatewayPor=8080;WapGatewayAPN=w
ap.ideasclaro";
    url = "http://portal.sat.gob.gt/ws/vehiculo/contenido.jsp" + conexion;
      String requeststring = "vnit" + "=" + nitTextField.getString()
                  + "&vuso" + "=" +  uso
                  + "&vplaca" + "=" + placaTextField.getString();
    try{
     http = (HttpConnection) Connector.open(url,Connector.READ_WRITE);
     // Solicitud cliente
      http.setRequestMethod(HttpConnection.POST);
      http.setRequestProperty("Content-Language","es-ES");
      http.setRequestProperty("User-Agent","Profile/MIDP-2.0 Configuration/CLDC-1.1");
      http.setRequestProperty("Connection","close");
      http.setRequestProperty( "Content-Type", "application/x-www-form-urlencoded" );
      http.setRequestProperty("Content-Length", Integer.toString(requeststring.length()));
     //Envío de parámetros
      OutputStream os = http.openOutputStream();
      os.write( requeststring.getBytes() );
      // Respuesta del servidor
      iStrm = http.openInputStream();
      // Procesar la respuesta
      procesarRespuestaServidor(http,iStrm);
    }finally{
      // Cerrar la conexión
      if (iStrm != null)
        iStrm.close();
      if (http != null)
        http.close();
```

Fuente: BlackBerry JDE.

En la solicitud del cliente, el primer paso que realiza este método es la formación del Uniform Resource Locator (URL).

Como existen varias formas para conectarse desde el dispositivo BlackBerry, es necesario definir la opción que utilizará, esto se logra adicionando parámetros al URL, en este caso se indicará que para conectarse a *internet*, utilizará Wireless Application Protocol (WAP), que es un estándar abierto internacional para aplicaciones que utilizan las comunicaciones inalámbricas.

Es necesario considerar que estos parámetros son propios de cada operador, para esta aplicación, los datos de conexión WAP aparecen asociados a la variable wap y para el operador Claro.

Luego se creará la conexión HTTP, y posteriormente se indicará el tipo de petición a realizar, que para esta aplicación será un método de petición de información de http, en la que los datos se envían por medio de *post*.

Luego se establecen parámetros de información adicional a la petición. Esta información es: el idioma, versiones de CLDC y MIDP, el formato en el que se trasladan los datos al servidor, el tamaño de los datos a enviar, y por último, se indica que cuando se trasladen todos los datos entre la aplicación cliente y el servidor se cierre la conexión.

Por último, el método procesarRespuestaServidor se encargará de obtener el resultado de la verificación del vehículo.

El código fuente completo de la aplicación :

Figura 27. **Código completo**

```
//Importar librerías.
import javax.microedition.midlet.*;
import javax.microedition.io.*;
import javax.microedition.lcdui.*;
import java.io.*;

public class VerifCar extends MIDlet implements CommandListener{
  // Declaración de la variable manejador
  private Display miDisplay;

  //Forma de captura de datos
  private Form inicioScreen = new Form("Ingreso de la Información del Vehiculo");

  //Creación de campos editables, pantalla de captura
  TextField nitTextField = new TextField("Ingrese NIT: ", "", 10, TextField.ANY);
  TextField placaTextField = new TextField("Ingrese Placa: ", "", 12, TextField.ANY);

  //Creación de botones, pantalla de captura
  Command validarCommand = new Command("REVISAR", Command.OK, 1);
  Command salirCommand = new Command("SALIR", Command.EXIT, 1);

  //Forma de resultado
  private Form resultadoScreen = new Form("Resultado");
  private StringItem resultadoStringItem = new StringItem(null, "");

//Creación de botones, pantalla de resultado
  Command regresarCommand = new Command("REGRESAR", Command.OK, 1);

  //Creación de etiqueta de resultado
  private String resultadoString = "";

  public VerifCar(){//Constructor
    //Inicialización de manejador de pantalla
    miDisplay = Display.getDisplay(this);

    //Incorporación de objetos a la forma de resultado
    resultadoScreen.append(resultadoStringItem);
    resultadoScreen.addCommand(regresarCommand);

    //Agregar opciones del uso del vehículo
    cgUso.append("Particular", null);
    cgUso.append("Comercial", null);
    cgUso.append("Urbano", null);
    cgUso.append("Motocicleta", null);
```

Continuación de la figura 27.

```
    //Incorporación de objetos a la forma de captura
    inicioScreen.append(nitTextField);
    inicioScreen.append(placaTextField);
    inicioScreen.append(cgUso);
    inicioScreen.addCommand(validarCommand);
    inicioScreen.addCommand(salirCommand);
    inicioScreen.setCommandListener(this);   }

public void startApp() {//Se coloca la pantalla de captura, como pantalla inicial
    miDisplay.setCurrent(inicioScreen);
  }

  public void pauseApp() {  }

  public void destroyApp(boolean unconditional) {
   notifyDestroyed();  // Indicar al AMS que ha entrado en el estado de Destruido
  }

public void commandAction(Command c, Displayable s) {
    if (c == validarCommand) {
     try{
        ConectarServidor();
     }
     catch (Exception e)
     {
      resultadoString="Msg: " + e.toString();
     }
     mostrarResultado();
    } else if (c == salirCommand) {
      destroyApp(false);  //Llamada a la función para liberar recursos
    } else if (c == regresarCommand) {
      miDisplay.setCurrent(inicioScreen);
    }
  }

  private void mostrarResultado(){
     resultadoStringItem.setText(resultadoString);
     resultadoScreen.setCommandListener(this);
     miDisplay.setCurrent(resultadoScreen);//Se muestra al usuario la forma de resultado
  }

 private void ConectarServidor() throws IOException{
     HttpConnection http = null;
     InputStream iStrm = null;
     // Armado de URL
     String url = url = "http://portal.sat.gob.gt/ws/vehiculo/contenido.jsp" +
";deviceside=true;WapGatewayIP=216.230.133.66;WapGatewayPor=8080;WapGatewayAPN=wa
p.ideasclaro";
```

Continuación de la figura 27.

```
        String requeststring = "vnit" + "=" + nitTextField.getString()
                        + "&vuso" + "=" +  uso
                        + "&vplaca" + "=" + placaTextField.getString();
      try{
        http = (HttpConnection) Connector.open(url,Connector.READ_WRITE);
        http.setRequestMethod(HttpConnection.POST);
        http.setRequestProperty("User-Agent","Profile/MIDP-2.0 Configuration/CLDC-
1.1");//Versiones
        http.setRequestProperty("Connection","close");//Cerrar la conexión después de la petición
        http.setRequestProperty( "Content-Type", "application/x-www-form-urlencoded" );
        http.setRequestProperty("Content-Length", Integer.toString(requeststring.length()));
        OutputStream os = http.openOutputStream();
        os.write( requeststring.getBytes() );
        iStrm = http.openInputStream();
        procesarRespuestaServidor(http,iStrm);
      }finally{
        //Se cierra el Stream y la conexión
        if (iStrm != null)
          iStrm.close();
        if (http != null)
          http.close();
      }
    }

   private void procesarRespuestaServidor(HttpConnection http, InputStream iStrm) throws
IOException{
     if (http.getResponseCode() == HttpConnection.HTTP_OK){//Si logra obtener una conexión
      int length = (int) http.getLength();
      String str;
      if (length != -1){
        byte respuestaData[] = new byte[length];
        iStrm.read(respuestaData);
        str = new String(respuestaData);
      }else{  // Tamaño no permitido
        ByteArrayOutputStream bStrm = new ByteArrayOutputStream();
        int ch;
        while ((ch = iStrm.read()) != -1)
          bStrm.write(ch);
        str = new String(bStrm.toByteArray());
        bStrm.close();      }
      resultadoString=str;
     }else
      resultadoString = new String(http.getResponseMessage());
   }}
```

Fuente: BlackBerry JDE.

El nombre del archivo fuente es: VerifCar.java y queda almacenado dentro del directorio donde se creó el *workspace*. El paso siguiente es compilar la aplicación, utilizando la opción *build*, de BlackBerry JDE que indicará si existe algún error en el código o si el resultado fue exitoso. Además de la compilación, se realiza una preverificación del MIDlet, que es un examen del código del MIDlet para verificar que no viola ninguna restricción de seguridad de la plataforma *Java* ME. De ser exitosa la compilación y la preverificación, es generado un archivo *class* dentro de la carpeta *classes* y la creación de los archivos JAD y COD que es el archivo que se instalará en el dispositivo BlackBerry.

Por último, debe ser creado el archivo ALX, que es utilizado para guiar al dispositivo en la instalación del MIDlet. Esta operación se realiza desde el BlackBerry JDE desde la opción *project* y luego *generate ALX File*.

4.5. Instalación y ejecución de la aplicación

La instalación del MIDlet puede realizarse mediante un cable conectado a una computadora, utilizando el *software* que ofrece BlackBerry para instalación en el dispositivo, BlackBerry Desktop Manager. Cuando un MIDlet está instalado en el dispositivo, todas sus clases, archivos y almacenamiento persistente están preparados y listos para su uso.

El último paso es la ejecución del MIDlet desde el *smartphone* BlackBerry, por lo que es necesario utilizar el menú para trasladarse a los archivos descargados y ejecutar la aplicación VerifCar.

CONCLUSIONES

1. El desarrollo de aplicaciones para el *smartphone* BlackBerry, en el cual la interfaz de usuario no demanda requerimientos gráficos y que permita la comunicación con servidores, para realizar alguna transacción u obtener alguna información, es relativamente sencillo, con un corto tiempo en su implementación, con una respuesta rápida.

2. Ya que un MIDlet de Java ME, es una aplicación estándar, y que los archivos compilados deben ser interpretados por el dispositivo, la aplicación puede ejecutarse en otros dispositivos de generaciones anteriores, y que incluya la JVM, que se encarga de ejecutar el MIDlet.

3. La herramienta de desarrollo facilita de gran manera, el desarrollo de este tipo de aplicaciones, ya que cuenta con un simulador del dispositivo. Pero no todo funciona de la misma forma en el dispositivo, puesto que el tamaño en el que se visualizan los iconos pueden cambiar, así como el funcionamiento, que en este caso requirió la investigación adicional de los diferentes tipos de conexiones a Internet que pueden utilizarse desde el BlackBerry, así como los parámetros necesarios para configurar la conexión.

RECOMENDACIONES

1. Para mejorar el performance de este tipo de aplicaciones que requieren una conexión a Internet, es necesario hacer uso de hilos de trabajo que proporciona Java, conocido también como Thread, para que el proceso de conexión a Internet se ejecute desde un hilo diferente al cual corre la aplicación principal, con el propósito de que si ocurre algún error, este no detenga la manipulación del MIDlet en su pantalla principal.

2. Proporcionar una transferencia de datos genérica, con la cual los datos puedan consumirse de diferentes fuentes, por lo que, recomiendo que el software del lado del servidor se realice por medio de WebService. Por medio de este único WebService, será posible consumirlo desde diferentes tipos de cliente, incluyendo un *smartphone* BlackBerry.

3. Hacer uso del conjunto de librerías BlackBerry proporciona un valor agregado, ya que permiten crear mejores interfaces, aunque de esta forma ya no sea posible utilizarla desde teléfonos de generaciones anteriores, pero se puede obtener una interfaz más agradable.

BIBLIOGRAFÍA

1. *BlackBerry Java ME*. [en línea]: <http://docs.blackberry.com/en/developers/deliverables/5827/Java_ME_an_Java_APIs_for_BB_446980 _11.jsp> [Consulta: 17 de enero de 2012].

2. *Java Platform Micro Edition* (*Java* ME). [en línea]: http://www.oracle.com/technetwork/java/javame/index.html> [Consulta: 15 de octubre de 2011].

3. *Java Net*. [en línea]: <http://today.java.net/pub/a/today/2006/01/24/introduction-to-blackberry-j2me.html> [Consulta: 15 de octubre de 2011].

4. *Schildt Herbert*. *Java* 2: *the Complete Reference.* 4a ed. *Osborne, Kansas*: McGraw-Hill, 2001. 1077p.

5. *Transmisión de datos en dispositivos móviles.* [en línea]: <http://www.tech-zone.org/pdf/canaviri_2010.pdf> [Consulta: 18 de marzo de 2012].