FACULTAD DE CIENCIAS DE LA ADMINISTRACIÓN

ESCUELA DE INGENIERÍA DE SISTEMAS

Monografía previa a la obtención del Titulo de
Ingeniería de Sistemas

TEMA:

"PRODUCTOS CON SERVICIO A DOMICILIO A TRAVÉS DEL ENVÍO DE MENSAJES SMS IMPLEMENTADO EN J2ME"

AUTORES:

Javier Enrique Auquilla Castro
Ángel Celestino Juca Guaillas

DIRECTOR:

Ing. Pablo Esquivel

Cuenca-Ecuador
2007

Las ideas y opiniones vertidas en la presente monografía son de exclusiva responsabilidad de sus autores

Javier Auquilla Castro

Ángel Juca Guaillas

DEDICATORIA

A todas aquellas personas que me apoyo y no solamente a las que me apoyaron, sino también para todo aquel que se pueda beneficiar de este trabajo, esta hecho con toda dedicatoria, lo cual producirá una gran satisfacción en poder ayudar a quien así lo considera.

Javier Auquilla

DEDICATORIA

Dedico la presente monografía por sobre todo a Dios por darme vida, salud y la oportunidad de cumplir ésta meta.

A mis padres Ángel y María que con cuyo inmenso sacrificio tuve la oportunidad de estudiar y por sus sabios consejos que me animaron a continuar adelante.

A mi esposa Carmen y a mis hijos Andrés y Sebastián por su paciencia y ser la fuente de inspiración para seguir luchando cada día.

A mis hermanos, especialmente Jorge y Rosa por su apoyo incondicional a lo largo de mis estudios.

A todas las personas que en cierto momento me apoyaron de cualquier manera y me dieron ánimos para llegar a éste objetivo, a mis suegros, amigos y a todos quienes la presente monografía pueda servirles en algo.

Ángel Juca Guaillas.

AGRADECIMIENTO

Gracias a todos los que, de una forma u otra, han contribuido a que por fin haya realizado el Proyecto de fin de Carrera. Y a mis padres, Marcelo y Sara, por confiar en mí, por su apoyo y haberme dado la gran oportunidad de estudiar una carrera universitaria. A ellos dedico este proyecto. Nunca podré compensar todo lo que han hecho por mí. Muchas Gracias.

Javier Auquilla

AGRADECIMIENTO

Agradezco a Dios, ya que confiando en Él y con esmero se puede alcanzar lo que se propone. A mis padres por la confianza que siempre me brindaron de que sí podía llegar a cumplir ésta meta tan anhelada, por sus palabras, consejos y valores que inculcaron en mí para llegar a ser profesional y sobre todo un buen ser humano y porque con gran esfuerzo me brindaron el mejor regalo que es la educación y por esto les estaré eternamente agradecido. A mi esposa y a mis hijos por darme las fuerzas y las ganas de superarme cada día más. A mis hermanos por todo el apoyo brindado. A la Universidad del Azuay, a mis maestros que supieron infundir los conocimientos y valores. A nuestro director de monografía, el Ingeniero Pablo Esquivel por su permanente predisposición, voluntad y por sus valiosas sugerencias.
Un agradecimiento a todas las personas que me conocen, a toda mi familia, mis suegros, compañeros y amigos que siempre estuvieron a mi lado en las buenas y en las malas, gracias eternas les llevo en mi corazón, que Dios les bendiga a todos..!!.

Ángel Juca Guaillas.

Índice de Contenidos

INTRODUCCIÓN

El mercado del comercio electrónico a través de dispositivos móviles, como teléfonos móviles, denominado m-comercio, está considerado como uno de los mercados tecnológicos de mayor crecimiento en los próximos años. Las aplicaciones de Productos con servicio a domicilio requieren que el móvil esté conectado a Internet utilizando GPRS. El uso de la mensajería SMS, nos permiten enviar mensajes de los pedidos efectuados que son enviados desde el Terminal al centro servidor de mensajes cortos o SMSC, que a su vez se encarga de hacer llegar el mensaje al móvil destinatario.

1. INTRODUCCIÓN

1.1. Objetivos del Proyecto

El objetivo del presente proyecto es el desarrollo una aplicación para hacer pedidos de negocios con productos para llevar a través del teléfono móvil. El cliente tendrá la aplicación-menú en su móvil con todos los últimos productos. Esta aplicación, que contiene los productos, permitirá seleccionar el producto deseado y confeccionar el pedido. La aplicación calcula el precio y envía el pedido a la central del negocio.

La aplicación a desarrollar permitirá ahorrar tiempo a los compradores permitiendo confeccionar su pedido tranquilamente en cualquier lugar con sólo disponer de un móvil.
También se desarrollará la aplicación para la central del negocio, que permitirá, definir la opción de confirmar el pedido. El pedido es confirmado vía SMS.

1.2. Breve referencia a la tecnología J2ME

La plataforma J2ME es una familia de especificaciones que definen varias versiones minimizadas de la plataforma Java 2. Es la versión de Java orientada a los dispositivos móviles. Debido a que los dispositivos móviles tienen una potencia de cálculo baja e interfaces de usuario pobres, es necesaria una versión específica de Java destinada a estos dispositivos, ya que el resto de versiones de Java, J2SE o J2EE, no encajan dentro de este esquema. J2ME es por tanto, una versión “reducida” de J2SE. La arquitectura J2ME se basa en familias y categorías de dispositivos. Una categoría define un tipo de dispositivo particular: teléfonos celulares, buscapersonas y organizadores personales. Una familia de

dispositivos está compuesta por un grupo de categorías que tiene requisitos similares de memoria y capacidad de procesamiento. J2ME debe estar lista para adaptarse a los nuevos dispositivos por ello es modular y escalable, características que están definidas en un entorno global constituido por cuatro capas básicas (figura 1.2) sobre el sistema operativo del dispositivo:

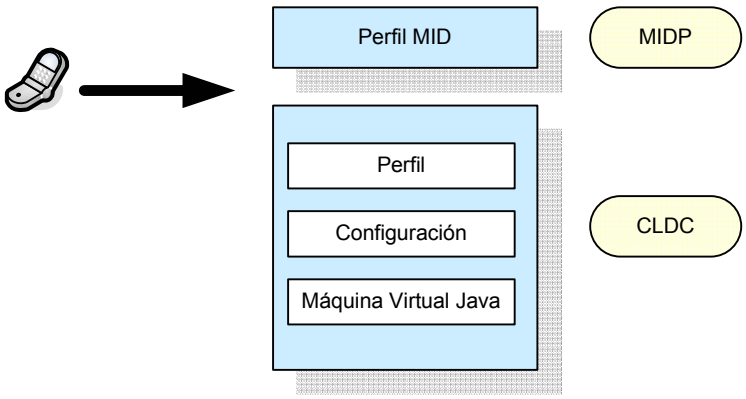


Figura 1.2 Capas de la arquitectura J2ME

- Capa máquina virtual Java: esta capa corresponde a una implementación específica de la máquina virtual para cada uno de los dispositivos, de forma que pueda adaptarse y soportar el sistema operativo que se ejecute en ese dispositivo.
- Capa de configuración: orientada al dispositivo, define el mínimo conjunto de características de la máquina virtual Java y de las librerías de clases Java que están disponibles para un conjunto de dispositivos.
- Capa de perfil: orientada a la aplicación, define el mínimo conjunto de API´s disponibles para una determinada familia de dispositivos, a una categoría vertical de dispositivos, incluyendo librerías de clases que son mucho más específicas que las de una configuración. Las aplicaciones se escriben para un perfil específico, de modo que cualquier dispositivo que soporte ese perfil pueda ejecutarla.
- Capa de Perfil para Dispositivos de Información Móvil (MIDP) consiste en un conjunto de API´s Java que permiten la creación de interfaces de usuario, conexiones de red, manipulación de datos, sonido, seguridad, etc.

La combinación de las tres primeras capas constituye la configuración CLDC (Configuración para Dispositivos con Conexión Limitada), que junto con la capa MIDP forman el entorno de ejecución estándar para las aplicaciones y servicios que se pueden descargar dinámicamente sobre los dispositivos de los usuarios finales.

1.3. ¿Qué es un MIDlet?

Un midlet es para J2ME lo que un applet para J2SE. Al igual que un applet, un midlet no tiene método main(). Un applet es subclase de la clase Applet, mientras que un midlet lo es de la clase MIDlet. Ambos tienen que implementar unos métodos concretos: init(), start(), stop(), destroy(), en el caso de un applet y startApp(), pauseApp(), destroyApp(), en el de un midlet. Y mientras un applet tiene que ser ejecutado sobre un navegador web, un midlet lo tiene que hacer en un dispositivo con soporte J2ME (teléfonos móviles, PDA's,...).

El gestor de aplicaciones o AMS (Application Management System) es el software encargado de gestionar los MIDlets. Este software reside en el dispositivo y es el que nos permite ejecutar, pausar o destruir nuestras aplicaciones J2ME.
Un midlet, cuando está ejecutándose, puede estar en tres estados diferentes:

- Activo, si se está ejecutando. Ocupa en memoria principal todos los recursos que necesite. Puede pasar al estado de pausa, llamando al método MIDlet.pauseApp(), o al estado destruido, llamando al método MIDlet.destroyApp().

- Pausa, no está actualmente en ejecución. En este estado el midlet no utiliza ningún recurso compartido. Para volver a estar en ejecución, estado activo, tiene que hacer una llamada al método MIDlet.startApp(). Desde este estado también se puede pasar al estado destruido.

- Destruido, se liberan todos los recursos ocupados por el midlet. A este estado se pasará si se finaliza la ejecución del midlet o si se necesitan los recursos ocupados por éste para una aplicación de mayor prioridad (ver Figura 1.3).

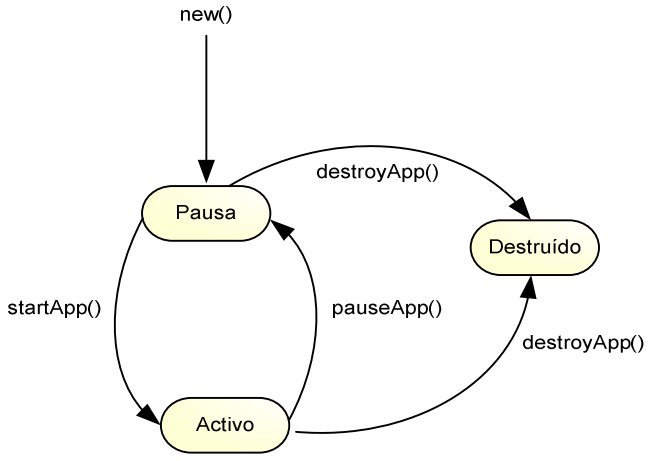


Figura 1.3 Estados de un MIDlet en ejecución

2. HERRAMIENTAS DE SOFTWARE UTILIZADAS

2.1. Eclipse 3.3

Eclipse es un entorno independiente de la plataforma, de código abierto, para crear aplicaciones clientes de cualquier tipo. La primera y más importante aplicación que ha sido realizada con este entorno es el afamado IDE Java llamado Java Development Toolkit (JDT) y el compilador incluido en Eclipse, que se usaron para desarrollar el propio Eclipse.

Eclipse fue creado originalmente por IBM. Ahora lo desarrolla la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

El entorno integrado de desarrollo (IDE) de Eclipse emplea módulos (en inglés plugin) para proporcionar toda su funcionalidad, a diferencia de otros entornos monolíticos donde las funcionalidades están generalmente prefijadas, las necesite el usuario o no. El mecanismo de módulos permite que el entorno de desarrollo soporte otros lenguajes además de Java.

Una de sus grandes ventajas es que basa su funcionamiento en plugins con lo que es ampliable para que haga prácticamente cualquier cosa, desde edición de XML a control del Tomcat, pasando por plugins para otros lenguajes como Perl o Shell Script.

2.2. J2ME Wireless Toolkit 2.2

El J2ME Wireless Toolkit es un completo entorno de desarrollo para escribir probar y depurar aplicaciones MIDP. También pertenece a la firma Sun Microsystems.
Esta herramienta incluyo los APIs MIDP y CLDC. Como herramienta de desarrollo incluye la KToolbar que permite compilar, construir y ejecutar una aplicación Java en el emulador que se indique. Este emulador proporciona un log de eventos, recolector de basura, carga de clases, manejo de excepciones e, incluso, una opción para ejecutar herramientas desde la propia línea de comandos.
No dispone de editor integrado, pero puede integrarse directamente con otras herramientas de desarrollo Java, como Sun ONE Studio o JBuilder.

2.3 El servidor de servlets: Tomcat 5

Tomcat (también llamado Jakarta Tomcat o Apache Tomcat) funciona como contenedor de servlets y es desarrollado bajo el proyecto Jakarta en la Apache Software
Foundation. Implementa las tecnologías Java Servlet 2.4 y JavaServer Pages 2.0 (JSP) de Sun Microsystems.

Tomcat es un servidor de aplicaciones que, a diferencia de un servidor Web, como es por ejemplo Apache, incluye un contenedor Web que puede servir páginas dinámicas (a diferencia del servidor Web, que sólo sirve páginas HTML estáticas).

Incluye el compilador Jasper, que compila páginas JSPs y las convierte en servlets.
Además, funciona con cualquier sistema operativo que disponga de máquina virtual Java, ya que fue escrito en este mismo lenguaje.

2.4 MySQL 5.0

En su nacimiento, MySQL no tenía elementos esenciales de las bases de datos relacionales, como podían ser la falta de integridad referencial y de transacciones. Pero gracias a su simplicidad y a su licencia de código abierto, todas las características que le faltaban fueron sumadas al proyecto tanto por la empresa que la mantiene, MySQL AB, como por desarrolladores de software libre.

Características a destacar son:

- Amplio subconjunto del lenguaje SQL.
- Indexación y búsqueda de campos de texto.
- Manejo seguro de claves foráneas y transacciones.
- Flexibilidad: esta disponible en multitud de plataformas y sistemas.
- Conectividad asegurada.

Otro dato a tener en cuenta es que MySQL fue seleccionada para formar parte de las llamadas Soluciones LAMP (Linux, Apache, MySQL, PHP/Perl/Phyton), que describen las aplicaciones web creadas por la combinación anterior de herramientas.
Fue popularizada por la editorial O`Reilly, de gran peso en el mundo informático.

3. INTERFACES DE USUARIO

3.1. Introducción

La API que nos permite crear interfaces de usuario en J2ME es MIDP. Esta API nos suministra los siguientes paquetes (exclusivos de J2ME):

- javax.microedition.midlet
- javax.microedition.lcdui
- javax.microedition.io
- javax.microedition.rms
-

El paquete más importante de todos es javax.microedition.midlet, el cual nos provee una única clase, MIDlet, que nos permite la ejecución de aplicaciones en dispositivos móviles.

El paquete javax.microedition.lcdui nos suministra una serie de clases e interfaces que nos permitirán crear las interfaces de usuarios (tanto en alto como en bajo nivel). Este paquete nos permite trabajar en alto nivel, con screens, creando menús, editores de texto,... y en bajo nivel, canvas (trabajamos a nivel gráfico).

Ambas clases, Screen y Canvas, heredan de la clase Displayable (ver Figura 3.1.).
Todo lo que mostremos por la pantalla del dispositivo hereda de Displayable.

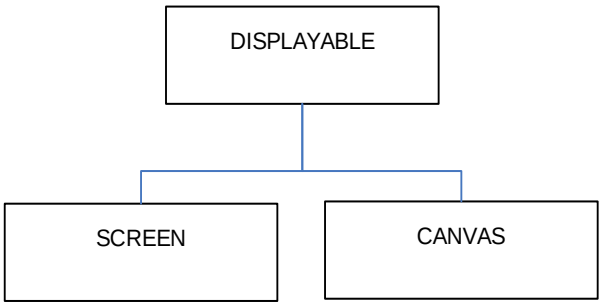
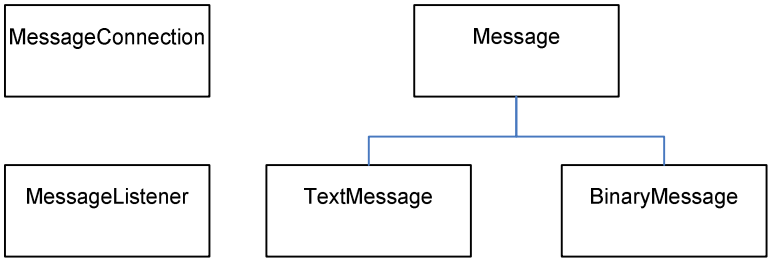


Figura 3.1 Relación de clases

3.2. WMA: Wireles Messaging API

API que proporciona a las aplicaciones MIDP la capacidad de enviar/recibir mensajes SMS. Las interfaces de la API de mensajería WMA están definidas en el paquete javax.wireless.messaging, este paquete define todas las interfaces necesarias para enviar y recibir mensajes wireless, tanto binarios como de texto.

3.2.1. Clases en javax.wireless.messaging



INTERFAZ	DESCRIPCION	METODOS
Message	Interfaz base, de la que derivan TextMessage y Binary Message	getAddress(),getTimestamp(), setAddress()
BinaryMessage	Subinterfaz de Message que proporciona métodos para fijar y detectar el payload binario	getPayloadData(), setPayloadData()
TextMessage	Subinterfaz de Message que proporciona metodos para fijar y detectar el payload de texto	getPayloadData(), setPayloadData()
MessageConnection	Subinterfaz de la interfaz Connection de GCF, que proporciona una factoría de Messages, y métodos para enviar y recibir Messages.	newMessage(), receive(), send (), setMessageListener(), numberOfSegements()
MessageListener	Define la interfaz listener para implementar la notificación asincrona de objetos Message	NotifyIncomingMessage()

3.2.1.1. Interfaz Message

Java.wireless.messaging.Message es la interfaz base para todo tipo de mensajes comunicados mediante WMA:

- Un tipo Message es lo que se envía, recibe.
- En ciertos aspectos, similar a un datagrama (Datagram):
  - tiene direcciones origen y destino,
  - tiene payload ,
  - tiene formas para enviar un mensaje y bloquearse a la espera de un mensaje.

WMA proporciona funcionalidad adicional:

- soporte para mensajes de texto y binarios,
- interfaz listener para recepción asíncrona de mensajes.

3.2.1.2. Interfaces BinaryMessage y TextMessage

BinaryMessage:

- representa un mensaje con payload binario,
- declara métodos para enviar y recibir mensajes.

TextMessagge:

- representa un mensaje con payload de texto.
- proporciona métodos para leer y escribir payloads de texto.

3.2.1.3. Interfaz MessageConnection

- Es una subinterfaz representa de javax.microedition.io.Connection.
- Proporciona los métodos newMessage(), send(), y receive() para crear, enviar, y recibir objetos Message.
- MessageConnection.numberOfSegements()
  - Determina información de segmento acerca del Message antes de ser enviado.
- Define dos constantes String.
  - BINARY_MESSAGE y TEXT_MESSAGE
- Crea una Connection (en este caso, una MessageConnection) hay que llamar a javax.microedition.io.Connector.open()
- Para cerrarla, javax.microedition.Connection.close()
- Un MessageConnection puede crearse como:
  - Cliente(solo puede enviar mensajes)
  - Servidor(puede enviar y recibir mensajes)
- El tipo de MessageConnection se especifica mediante URL
  - Cliente: (MessageConnection)Connector.open(“sms://+091072184:6064”);
  - Servidor: (MessageConnection)Connector.open(“sms://:6064”);

  Los SMS que lleguen al puerto 6064 serán atendidos por el MIDLet.

En realidad, "6064" NO es un puerto sino un IDENTIFICADOR que le indica a la plataforma java que desea tratar los SMS que lleguen al terminal con ese identificador.

3.2.1.4. Interfaz MessageListener

Implementa el patrón de diseño Listener para recibir objetos Message de forma asíncrona.

- Define un solo método: notifyIncomingMessage()
  - es invocado por la plataforma cada vez que se recibe un mensaje,
  - para registrarse , MessageConnection.SetListener().

3.3. El Push Registry de MIDP 2.0

El concepto Push indica la capacidad o el mecanismo para recibir información de forma asíncrona, es decir, cuando la información está disponible. Nuestra aplicación estará en modo pasivo y en un determinado momento será avisada de que dispone de nueva información y por lo tanto actuará en consecuencia. Dentro de MIDP, la tecnología Push Registry permite que los MIDlets sean lanzados automáticamente, sin interacción directa del usuario. Tendremos dos posibles formas de lanzar las aplicaciones J2ME, tendremos activación por red (mediante una conexión externa) o por tiempo (pasado un determinado tiempo). Push Registry es encapsulado dentro de una única clase, la clase javax.microedition.io.PushRegistry. Los métodos de esta clase son:

- listConnections() Retorna la lista de conexiones registradas para un midlet o midlet suite.
- registerConnection() Registra una conexión push.
- unregisterConnection() Elimina el registro correspondiente a una conexión push.

```
conexiones = PushRegistry.listConnections( true ); if(( conexiones == null) ||
(conexiones.length = =  0 )) { texto = new StringItem("Esperando en puerto
"+PUERTO_SMS+"..",""); formulario.append(texto);}
```

Código: PushRegistry.listConnections()

3.4. Interfaz de usuario de alto nivel

Para crear interfaces de alto nivel, J2ME aporta con cuatro clases. Éstas clases heredan de Screen y son las siguientes: Alert, Form, List, TextBox (ver Figura 3.2.).

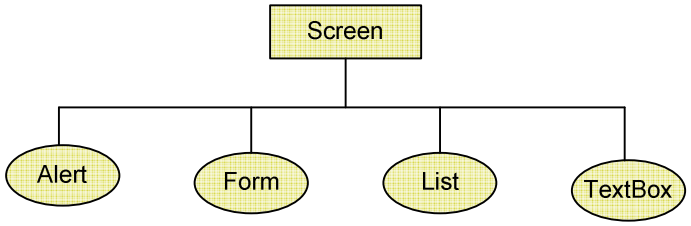


Figura 3.2 Interfaces de alto nivel

Un *midlet* puede estar compuesto de varios elementos de este tipo, pero sólo podemos mostrar uno cada vez ya que ocupan la pantalla entera. Para cambiar de un elemento a otro utilizamos el método setCurrent() de la clase Display.

void setCurrent (Displayable siguienteInterfaz)

3.4.1. La clase Alert

Ésta clase permite mostrar información por pantalla al usuario, normalmente de errores, mediante un texto o imagen, o ambas, durante un tiempo determinado o hasta que se produzca un evento de comando. El constructor de la clase Alert es el siguiente:

Alert (String titulo)

El único parámetro es el nombre de la alerta, el cual se mostrará en la cabecera de la pantalla. Para asignar un texto y una imagen a nuestra alerta utilizaremos los siguientes métodos respectivamente:

public void setImage (Image img)
public void setString (String str)

Para asignar un tipo de alarma, sólo se diferencian en el tipo de sonido que emiten, utilizamos el siguiente método:

public void setType (AlertType tipo)

Cuyo argumento puede ser una de las siguientes constantes:

- ALARM
- CONFIRMATION
- ERROR

- INFO
- WARNING

Si queremos mostrar la alerta durante un determinado tiempo, le pasaremos como parámetro, al método setTimeout(int tiempo), el tiempo en milisegundos. (ver Figura 3.3.).

Figura 3.3. Ejemplo de la clase Alert.

3.4.2. La clase List

Con esta clase podemos realizar listas de opciones para la creación de menús. Su constructor es el que se muestra a continuación:

List (String titulo, int tipoLista)

El primer parámetro es el mismo que para la clase Alert anteriormente explicada.
El segundo es el tipo de lista a crear, a elegir entre las siguientes constantes:

- MULTIPLE: Permite la selección múltiple.
- EXCLUSIVE: Sólo se puede seleccionar un elemento.
- IMPLICIT: Se selecciona el elemento que está marcado.

Para el primero tipo de listas anterior, el que vamos a utilizar en nuestro proyecto, tenemos un método que nos permite saber cuál ha sido la opción elegida por el usuario. Este método es getSelectedFlags(), que nos devuelve un entero, siendo este valor, comenzando por cero, la posición del valor elegido. Si este valor lo pasamos como parámetro al método getString(), nos devolverá el texto de la opción elegida.

Para añadir opciones a la lista nos encontramos con el método append(...):

```
public int append (String nombreOpcion, Image imagenOpcion)
```

Añade un elemento a la lista, siendo el primer parámetro el texto y el segundo una imagen que acompaña a éste (si no queremos imagen le pasaremos el valor null).
Para borrar una opción de la lista tenemos el método delete(...):

```
public void delete (int opcion)
```

Figura 3.4. Ejemplo de la clase List.

3.4.3. La clase Form

Con la clase Form podemos realizar interfaces de usuario más complejas que las anteriores. Esta clase es de tipo contenedor, por lo que podemos incluir más de un elemento en pantalla. El constructor con el que crearemos un *form* vacío es el siguiente:

```
Form (String titulo)
```

Los posibles elementos que puede contener un *form* son:

- StringItem
- ImageItem
- TextField
- DateField
- ChoiceGroup
- Gauge

En la figura 3.5. se presenta de una forma global como se relacionan todas las clases de la rama de alto nivel de la clase Displayable.

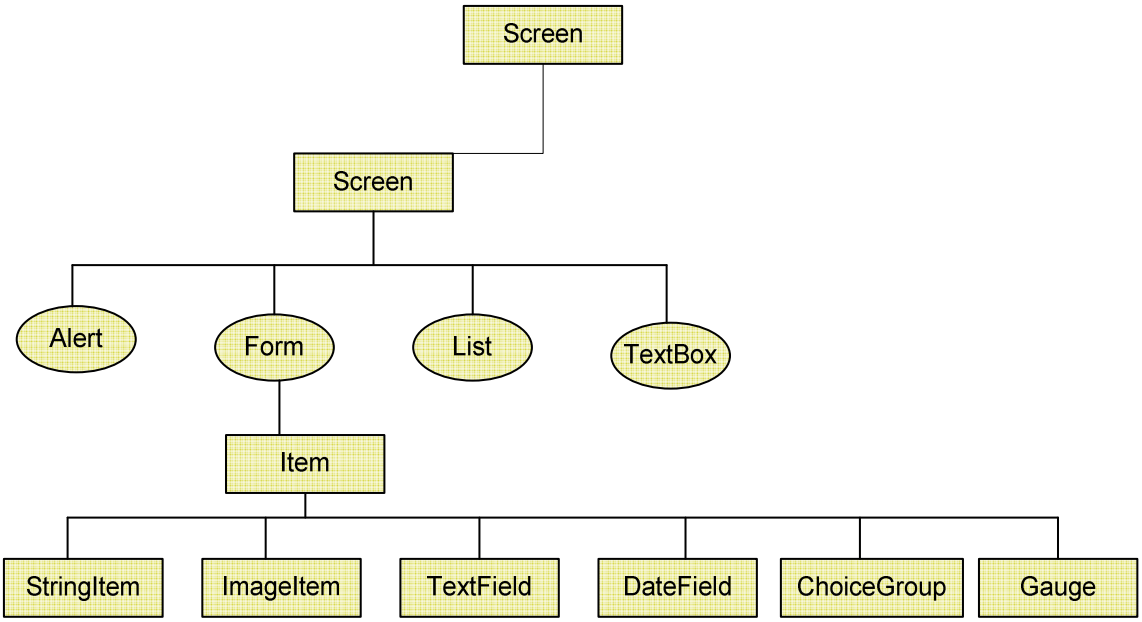


Figura 3.5. Relación completa de la rama de alto nivel de Displayable.

Todos estos elementos heredan de la clase Item y éste a su vez de la clase Form.
Un *item* es un elemento visual que no ocupa toda la pantalla, por lo que nos permite una mayor variedad a la hora de hacer interfaces de usuario. En nuestro proyecto los únicos elementos que vamos a utilizar de la clase Item son TextField, StringItem.
Los métodos principales para la creación de interfaces de la clase Form son:

int append (Item elemento)

Con este método añadimos un nuevo elemento al formulario. Nos devuelve, si todo va bien, el índice que se le ha asignado a dicho elemento dentro del formulario (al primero se le asigna el valor 0). Este es el único método, de la clase Form, que vamos a utilizar.

3.4.3.1 La clase ChoiceGroup

```
public class ChoiceGroup extends Item implements Choice
```

Un componente ChoiceGroup es un grupo de elementos que podemos seleccionar. Es prácticamente lo mismo que el componente List, pero dentro de un formulario.
Para construir un objeto ChoiceGroup realizaremos una llamada a su constructor con los siguientes parámetros:
ChoiceGroup(String etiqueta, int tipo)

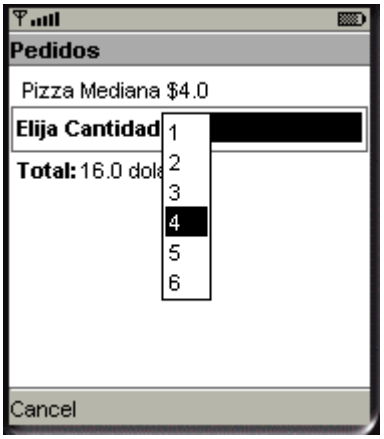


Figura 3.6. Ejemplo de ChoiceGroup.

3.4.3.2. La clase TextField

Ésta clase TextField proporciona un editor de texto diseñado para ser usado dentro de los forms.

El constructor sería de la siguiente forma:

```
TextField (String etiqueta, String texto, int tamañoMax, int limitacion)
```

El primer parámetro es la etiqueta del campo de texto, como cantidad, dirección, etc. El segundo y tercero son, respectivamente, el cuerpo y el tamaño máximo del texto a escribir. Y el cuarto y último, son las restricciones a las que sometemos al texto (sólo correo electrónico, sólo número, etc). Ver figura 3.7. para un ejemplo gráfico utilizado en el proyecto.

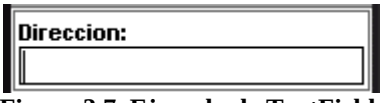


Figura 3.7. Ejemplo de TextField.

3.4.3.3. La clase StringItem

La clase *StringItem* hereda de la clase *Item.* Esta clase es usada principalmente para mostrar texto a un usuario en un *Form*. El constructor de esta clase es:

```
StringItem(String label, String text)
```

Para mostrar un solo texto, podemos indicar el parámetro label como una cadena vacía. Ver figura 3.8. para un ejemplo gráfico utilizado en el proyecto.

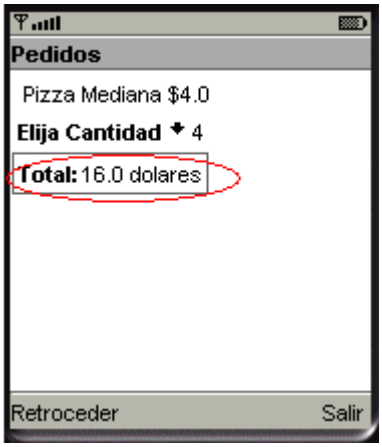


Figura 3.8. Ejemplo de StringItem.

4. ENVÍO Y RECEPCIÓN DE MENSAJES SMS

4.1. Pasos que hay que realizar para enviar un mensaje (SMS)

1. Obtener un MessageConnection en modo cliente.
2. Crear el mensaje a través de la interfaz MessageConnection.
3. Especificar el contenido y el destinatario del mensaje.
4. Usar el método send de la interfaz MessageConnection para enviar el mensaje.

```
try {      // Paso 1: Abrimos la conexion con el destinatario
      con = (MessageConnection)Connector.open( destino );
     // Paso 2: Creamos el SMS
```

```
  TextMessage mensaje = (TextMessage)con.newMessage(
    MessageConnection.TEXT_MESSAGE );
  // Fijamos el destinatario
  mensaje.setAddress( destino );
  // Paso 3: Establecemos el contenido del SMS
  mensaje.setPayloadText( tbMensaje.getString() );
 // Paso 4: Enviamos el SMS
  con.send( mensaje );
  // Indicamos al usuario que el mensaje se ha enviado
  Alert aviso = new Alert( "Envío SMS","  Mensaje enviado",
    null,AlertType.ERROR);
  display.setCurrent( aviso,formulario );
  } catch( Throwable e ) {
    e.printStackTrace();       }
 // Paso 5:Cerramos la conexión abierta con el destinatario
 if( con != null ) {
   try {       con.close();
   } catch( IOException ie ) {}     }
 }  }
```

Código: Envío mensaje

4.2. Pasos que hay que realizar para recibir un mensaje (SMS)

1. Obtener un MessageConnection en modo servidor.
2. Implementar la interfaz MessageListener en una de nuestras clases.
3. Asociar la clase anterior al MessageConnection.
4. Cuando el método notifyIncomingMessage() de la interfaz MessageListener sea invocado, significa que se ha recibido un mensaje.
5. Se debe invocar el método receive() de la interfaz MessageConnection en un hilo independiente.
6. Realizar el tratamiento del mensaje.

```
      // Paso2: Implementar la interfaz MessageListener
public class PushMid extends MIDlet
   implements CommandListener,Runnable,MessageListener {}
try { // Paso1: Abrimos la conexión en la dirección fijada
       con = (MessageConnection)Connector.open( direccion);
       // Paso3: Activamos el midlet como receptor de mensajes
       con.setMessageListener( this );}

      //Paso 4: indicamos que el midlet está en ejecución
public void notifyIncomingMessage( MessageConnection con ) {
   if( th == null ) {
     enEjecucion = true;
     th = new Thread( this );
     th.start();
     }
   }

  //Paso 5: Método receive()  que se encarga de la recepción de mensajes
```

```
try {
      mensaje = con.receive();
      if( mensaje != null && mensaje instanceof TextMessage ) {
        // Origen del envío
        String origen = mensaje.getAddress();

       //Paso 6:  Creamos un mensaje indicando el origen y el texto del
        // mensaje recibido
        texto = new StringItem( "",origen+"\n ->"+
          ((TextMessage)mensaje).getPayloadText() );
        // y lo presentamos en pantalla
        formulario.append("\n SMS Recibido:" + texto );     }
```

Código: Recepción de mensaje

4.3. Emulación de envió de mensajes SMS

Para comprobar que la porción de código encargada del envío de un SMS funcione, será necesario configurar adecuadamente el emulador y utilizar las consolas WMA que proporciona J2ME Wireless Toolkit 2.2. Esta herramienta no hace sino utilizar la consola de Wireless Toolkit. Habrá que configurar el número de teléfono que se va a asociar al emulador del móvil.

En el diálogo de preferencias, se activará la pestaña WMA, donde podrá ser configurado el número de teléfono del primer emulador (First Assigned Phone Number, al que asignaremos el número +10000000) y el del siguiente emulador (Phone Number of Next Emulator, al que asignaremos el número +10000001).

El siguiente paso consiste en acceder a la opción Utilities, para abrir una consola WMA (pulsar el botón Open Console en la sección WMA del diálogo Utilities). De acuerdo con los parámetros introducidos en las preferencias, se habrá abierto una consola WMA, que monitorizará el tráfico de red de un móvil con número de teléfono +10000000.

Ahora bastará con ejecutar la aplicación para comprobar cómo la consola WMA del teléfono +10000000 recibe el mensaje. Para observar detenidamente cómo se ejecuta la aplicación, se aconseja ejecutar la aplicación utilizando el modo de depuración. Previamente, se debería haber activado un punto de ruptura en la línea en que se declara el String url y se inicializa éste con la dirección del móvil destinatario del mensaje.

La ejecución de la suite Contactos provocará la creación de un emulador que ejecutará nuestra aplicación; dicho emulador tendrá asignado el número de teléfono +10000001. Ejecutando instrucción a instrucción se podrá comprobar que:

- Al ejecutar Connection.open, el emulador pide permiso para recibir mensajes de texto. Esto se debe a que cuando hay una conexión abierta, ésta puede ser utilizada tanto para leer mensajes como para escribirlos (ver figura a la izquierda, en la Figura 4).

- Al ejecutar Connection.send, el emulador pide permiso para enviar el mensaje de texto (ver figura a la derecha, en la figura 4).

Figura 4: Peticiones de autorización para recepción y envío de mensajes

- Tras ejecutar Connection.send, la consola WMA muestra un mensaje advirtiendo de la llegada de un SMS y muestra el contenido del mismo.

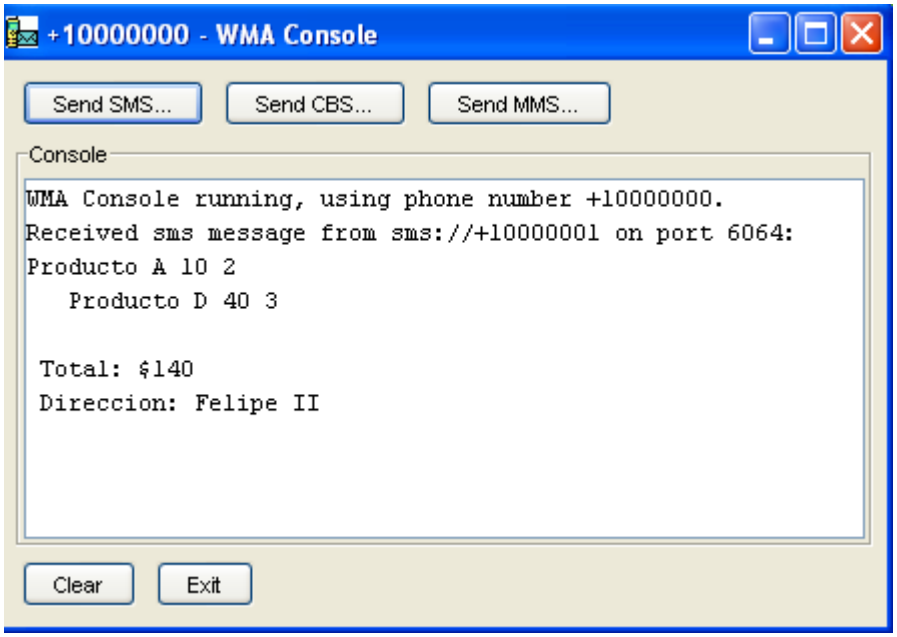


Figura: Consola WMA del teléfono destinatario del SMS

5. ANÁLISIS DE LA APLICACIÓN MÓVIL

5.1 Descripción y requisitos

La aplicación corresponde a la implementación de Productos con Servicio a Domicilio a Través del Envió de Mensajes SMS.

La Aplicación Móvil debe que cumplir los siguientes requisitos:

1. El usuario podrá seleccionar diferentes productos relacionados con el tipo de negocio.
2. Calcular automáticamente el precio. Esto es, el producto por la cantidad.
3. El usuario enviará la información sobre su dirección para su localización.

5.2. Casos de uso

En la figura 5 se puede ver el diagrama principal de casos de uso de la Aplicación del dispositivo Móvil. En el se distinguen dos actores: Cliente y Empleado. El actor cliente representa a los clientes que tendrá la aplicación-menú en su móvil, el actor empleado representa la confirmación del pedido.

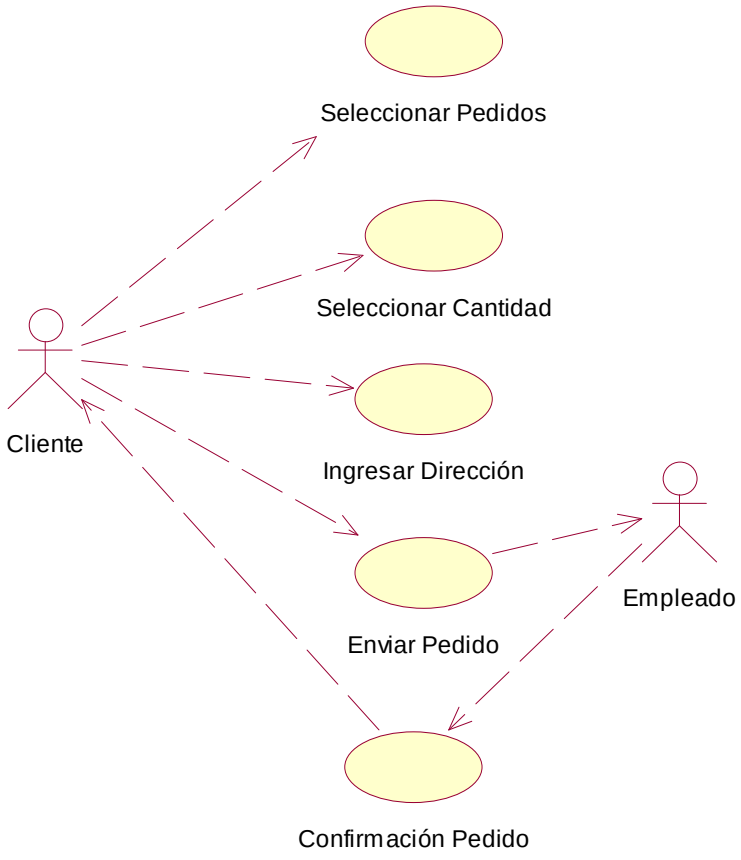


Figura 5 Diagrama de Casos de uso de la aplicación

Caso de Uso	Seleccionar Pedidos
Actores	Cliente
Propósito	Seleccionar el producto deseado

Visión General	El Cliente seleccionara el producto deseado La aplicación validara que al menos un producto se haya seleccionado.
Curso típico de Eventos	
Acción del Actor	**Respuesta de la Aplicación**
1. **Cliente:** selecciona el producto deseado, pulsa "seleccionar"	3. Comprueba si se ha seleccionado al menos un producto: a. Si es correcto visualizara pantalla elegir cantidad b. Si es incorrecto visualizara "no hay elementos seleccionados"
Cursos Alternativos	
• La línea 1: Selecciona Salir. Sale de la aplicación	

Caso de Uso	Elegir Cantidad
Actores	Cliente
Propósito	Elegir la cantidad deseada
Visión General	El Cliente seleccionara la cantidad deseada La aplicación validara que se ingrese algún valor.
Curso típico de Eventos	
Acción del Actor	**Respuesta de la Aplicación**
1. **Cliente:** Seleccionara la cantidad deseada 2. **Cliente**: ingresa la dirección del lugar de localización, pulsa "`Pedir"	3. Comprueba si se ha ingresado algún valor a. Si es correcto se enviara el pedido
Cursos Alternativos	
• Líneas 1,2: Selecciona "Atrás". Vuelve al menú anterior.	

Caso de Uso	Enviar Pedido
Actores	Cliente y Empleado
Propósito	Enviar información del pedido a la Central del Negocio.
Visión General	El Cliente envía el pedido a la aplicación de recepción del teléfono móvil del empleado.
Curso típico de Eventos	
Acción del Actor	**Respuesta de la Aplicación**
1. **Cliente:** selecciona la opción "enviar"	2. Visualizara "El pedidos ha sido efectuado"

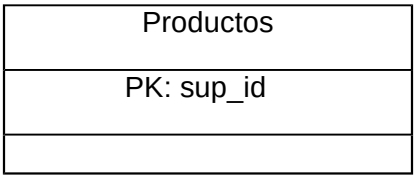
Caso de Uso	Confirmar Pedido
Actores	Cliente y Empleado
Propósito	Confirmar pedido vía SMS.
Visión General	El empleado envía confirmación del pedido.
Curso típico de Eventos	

Acción del Actor	Respuesta de la Aplicación
1.**Cliente:** selecciona la opción “confirmar”	2. Visualizara “Confirmación Pedido”

6. DETALLES DE IMPLEMENTACIÓN

6.1. La base de datos

La base de datos presentada es una base de datos MySql. Se ha creado un modelo entidad/relación para mostrar la una entidad creada es la siguiente:



Seguidamente se pasa detallar los tipos de los campos:

TABLA	ATRIBUTO	TIPO	RESTRICCIONES
productos	sup_id	Integer	Primary
	nom_producto	Varchar(32)	Unique, not null
	precio	Float	
	cantidad	Integer	

Tabla de BD Productos

La tabla de productos contendrá los datos relativos a los productos que tengamos en la base de datos. El atributo sup_id nos proporciona una forma rápida de encontrar un producto por medio de un identificador entero, además de ser la clave primaria. Nombre no puede estar repetido. El atributo Precio sirve para establecer el precio de dicho producto. El atributo Cantidad contiene el número de cantidad que tiene el producto en cuestión.

6.2. Comunicación entre aplicación móvil y aplicación web

Las aplicaciones móviles utilizan el paradigma cliente-servidor para realizar operaciones más complejas que las que permiten las limitadas capacidades de los dispositivos móviles. En el caso de nuestra aplicación móvil (cliente): recibe información de los productos almacenados en el servidor (aplicación web). De esta forma la aplicación móvil puede mostrar los detalles de los productos

6.2.1 Clase Connector

public class Connector

La conexión se realiza mediante el siguiente mensaje genérico:
Connector.open(“protocolo:dirección;parámetros”);
Ejemplo de invocación:
Connector.open(“http://localhost:8080/basic/pedidos”);
La clase Connector se encarga de buscar la clase específica que implemente el protocolo requerido. Si esta clase se encuentra, el método open() devuelve un objeto que implementa la interfaz Connection. Una conexión de tipo Connection se crea después de que un objeto Connector invoque al método open().

Figura 3.5.3. Ejemplo de Clase Conector.

6.3. JDBC

A continuación se citan los métodos y clases más relevantes de la librería JDBC en relación al proyecto.

- Class.forName("com.mysql.jdbc.Driver"); Carga la clase que implementa el driver JDBC
- Connection con=DriverManager.getConnection(urlBD, userBD, pwdBD); Conecta con la base de datos
- Statement orden=con.createStatement(); Crea un objeto capaz de almacenar los datos de la petición
- Resultset result=orden.executeQuery(Orden_SQL); Realiza la petición SQL
- result.next(); Avanza una posición el puntero de lectura del objeto Resultset.
- result.getString(num_columna); Lee un valor del registro del objeto Resultset a que actualmente apunta el puntero de lectura.

```
DriverManager.registerDriver(new Driver());
                this.connection = DriverManager.getConnection
```

```
("jdbc:mysql://localhost/curso",
                          "angel", "juca");

      Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("SELECT SUP_ID, NOM_PRODUCTO, PRECIO,
CANTIDAD FROM PRODUCTOS");


while (rs.next()) {
      int cod = rs.getInt("SUP_ID");
      String s = rs.getString("NOM_PRODUCTO");
      float n = rs.getFloat("PRECIO");
      int  i = rs.getInt("CANTIDAD");
```

Código: JDBC

7. INSTALACIÓN

En este apartado se verá, en primer lugar, como instalar las herramientas necesarias para crear, cargar, compilar o ejecutar un proyecto en nuestro PC, y en segundo lugar, todo lo necesario para pasar la aplicación a un dispositivo móvil.

7.1. Requisitos previos para la instalación

7.1.1. Software Necesario

- Sistema operativo Linux CentOS 4.2
- J2SDK: el kit de desarrollo para compilar, depurar y ejecutar programas en Java
- Eclipse: un IDE visual para poder desarrollar más cómodamente aplicaciones web en Java
- Tomcat: un servidor web para aplicaciones java
- MySQL: un servidor de bases de datos
- Sun Java Wireless Toolkit 2.2

7.2. Instalación y configuración

7.2.1. Instalación de Java

J2SDK 1.5 para linux .

- Crear el directorio `/usr/java`
  ```
  mkdir /usr/java
  ```
  Aunque puede instalarse en otros directorios distintos a los que aquí se especifican.
- Copiar a `/usr/java` y ejecutarlo desde allí (si no permite realizar ésta operación, se debe darle permisos de ejecución con `chmod ugo+x jdk-1_5_0_08-linux-i586.bin`). El programa se descomprimirá en `/usr/java/jdk1.5.0_13`
- Editar el archivo .bashrc de vuestro directorio `HOME` y añadir las siguientes líneas:
  - Para que se pueda ejecutar java desde cualquier directorio
    ```
    export PATH=/usr/java/jdk1.5.0_08/bin:$PATH
    ```
  - Para que Tomcat, Eclipse y otros programas "sepan" dónde está instalado Java
    ```
    export JAVA_HOME=/usr/java/jdk1.5.0_13
    ```
- Abrir una nueva consola y comprobar que Java está instalado ejecutando
  ```
  java -version
  ```

  debería aparecer un mensaje como:
  ```
  java version "1.5.0_08"
  ```
  Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_08-b03)
  Java HotSpot(TM) Client VM (build 1.5.0_08-b03, mixed mode, sharing)

7.2.2. Instalación de Eclipse

Eclipse 3.3 para linux.

- Descomprimir en `/usr/java` con `tar -xzvf eclipse.tar.gz`. Creará un directorio llamado `eclipse.`
- Se necesitan todos los permisos en el directorio `/workspace` y sus subdirectorios. Siendo `root`, teclear:
  ```
  chmod -R ugo+rwx /usr/java/eclipse/workspace
  ```

En caso de no tener permisos de escritura dentro de `workspace`, Eclipse funcionará, pero no se podrá guardar proyectos en dicho directorio y cada cierto tiempo aparecerá un mensaje "Could not save master table" (cuando Eclipse intente autoguardar datos).

- Si se desea ejecutar eclipse desde cualquier directorio, se tendrá que añadir el directorio donde se haya instalado (p.ej. /usr/java/eclipse) a la variable de entorno PATH (modificar el .bashrc si se quiere que los cambios sean permanentes)
  ```
  export PATH=$PATH:/usr/java/eclipse
  ```

- Para comprobar que eclipse funciona, teclear `eclipse` en una terminal.

7.2.3. Instalación de Wireless Toolkit

j2me_wireless_toolkit para linux

Para instalar la herramienta J2ME Wireless Toolkit, el primer paso es tener instalado una versión de J2SE SDK. Esto es debido a que J2ME Wireless Toolkit se ejecuta por encima de J2SE SDK y además hace uso del compilador de java cuando se selecciona la opción de Build.

- Ejecutar el archivo binario de la siguiente forma `./ j2me_wireless_toolkit-2_2-linux-i386`
- Se establece el directorio donde se encuentra nuestra versión de J2SE SDK instalada. Si la herramienta no lo reconoce automáticamente deberemos indicarle que esta se encuentra en nuestro caso en: `/usr/java/jdk1.5.0_08/bin`
- Se establece el directorio donde queremos que quede instalada la herramienta. Por ejemplo en $HOME/J2ME

7.2.4. Instalación MySQL

Mysql 5.0 para Linux

Hay varias posibilidades para hacer la instalación:

- Recomendado: instalado el MySQL que venga con vuestra distribución de Linux. De este modo el proceso de configuración y arranque de MySQL será automático.
- Buscar un paquete ya configurado para vuestra distribución de linux en la **página de descargas** de MySQL (.rpm, .deb o similar). Instalar tanto el cliente como el server. El proceso de configuración y arranque de MySQL será automático.
- Descargar el **binario** de MySQL e instalar y configurarlo con los siguientes pasos.
  - Descomprimir MySQL en `/usr/java`. Creará un directorio `mysql-max-5.0.27-linux-i686.`
  - Se puede seguir las instrucciones en el fichero INSTALL-BINARY incluído en mysql , o bien como root copiar **instalarMySQL.sh** y **my.cnf** en `/usr/java` y ejecutar instalarMySQL.sh (antes darle permisos de ejecución con `chmod ugo+x instalarMySQL.sh`)
- Para comprobar si el servidor de MySQL está en marcha, abrir una consola y teclear `mysql -`

`u root –p`

pedirá el password de root. Si todo va bien se entrará en la consola de mysql mostrando el prompt mysql>. En esta consola podemos teclear órdenes SQL, por ejemplo show databases; para ver las bases de datos existentes (en una instalación nueva de mysql hay al menos una base de datos llamada "mysql" y otra "test"). Para salir de la consola teclear quit.

7.2.5. Instalación Tomcat

Tomcat 5.5.17 para Linux

- Descomprimir en `/usr/java` con `tar -xzvf apache-tomcat-5.5.17.tar.gz`. Creará un directorio llamado `apache-tomcat-5.5.17` el cual se recomienda renombrar como tomcat5
- Dar todos los permisos en todos los subdirectorios de Tomcat.
  `chmod -R ugo+rwx /usr/java/tomcat5`

7.3. Ejecución de la aplicación web

Se modificó el puerto por donde Tomcat atenderá las solicitudes, por defecto es el 8080, pero se le cambió al 8180, por que el anterior estaba ya siendo utilizada por el proxy:
Abrir el archivo /usr/java/tomcat5/conf/server.xml, y se busca lo siguiente:

```
<!-- Define the Tomcat Stand-Alone Service -->
 <Service name="Catalina">
```

aquí viene mas comentarios y luego esto en la linea 76

```
<!-- Define a non-SSL HTTP/1.1 Connector on port 8080 -->
  <Connector port="8080" maxHttpHeaderSize="8192"
        maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
        enableLookups="false" redirectPort="8443" acceptCount="100"
        connectionTimeout="20000" disableUploadTimeout="true" />
  <!-- Note : To disable connection timeouts, set connectionTimeout value
```

cambiar **8080** a **8180** que es el puerto en el que se ejecutará.

Y ya solo queda instalar la aplicación web en Tomcat y para ello solamente es necesario, situar el fichero ServletJ2ME.war, en nuestra instalación del Tomcat, concretamente dentro de la carpeta webapps. Luego, reiniciar el Tomcat y automáticamente se descomprimirá el directorio y ya está lista la aplicación Web para ser utilizada. Para la creación del archivo con extensión .WAR abrir un terminal y se ejecuta el siguiente comando jar cvf nombre_archivo.war *, para lo cual debe estar correctamente configurado el JDK como ya se indicó anteriormente, el archivo se encuentra en el *cd.*

Luego de esto editar el fichero server.xml ubicado en el directorio /conf del Tomcat, añadiendo la siguiente línea:

```
<Context path="/basic" reloadable="true"
docBase="/usr/java/tomcat5/webapps/ServletJ2ME/webapps"
workDir="/usr/java/tomcat5webapps/ServletJ2ME/work"/>
```

De ésta manera se especifica las rutas del servlet.

Una vez hecho esto la aplicación estará lista para usar. Para comenzar a usar la aplicación ingresar la siguiente dirección en el navegador:

http://190.12.31.135:8180/basic/pedidos

7.4. Ejecución de la aplicación en un PC

Abrir el eclipse instalado anteriormente y desplegar el menú file y seleccionar File System del submenú Import e ingresar la ruta donde esta localizado el directorio del proyecto.

Tras abrir la aplicación se puede compilarla y/o ejecutarla ejecutando los respectivos botones para tal uso, Build para compilar y Run para ejecutar (ver Figura 7).

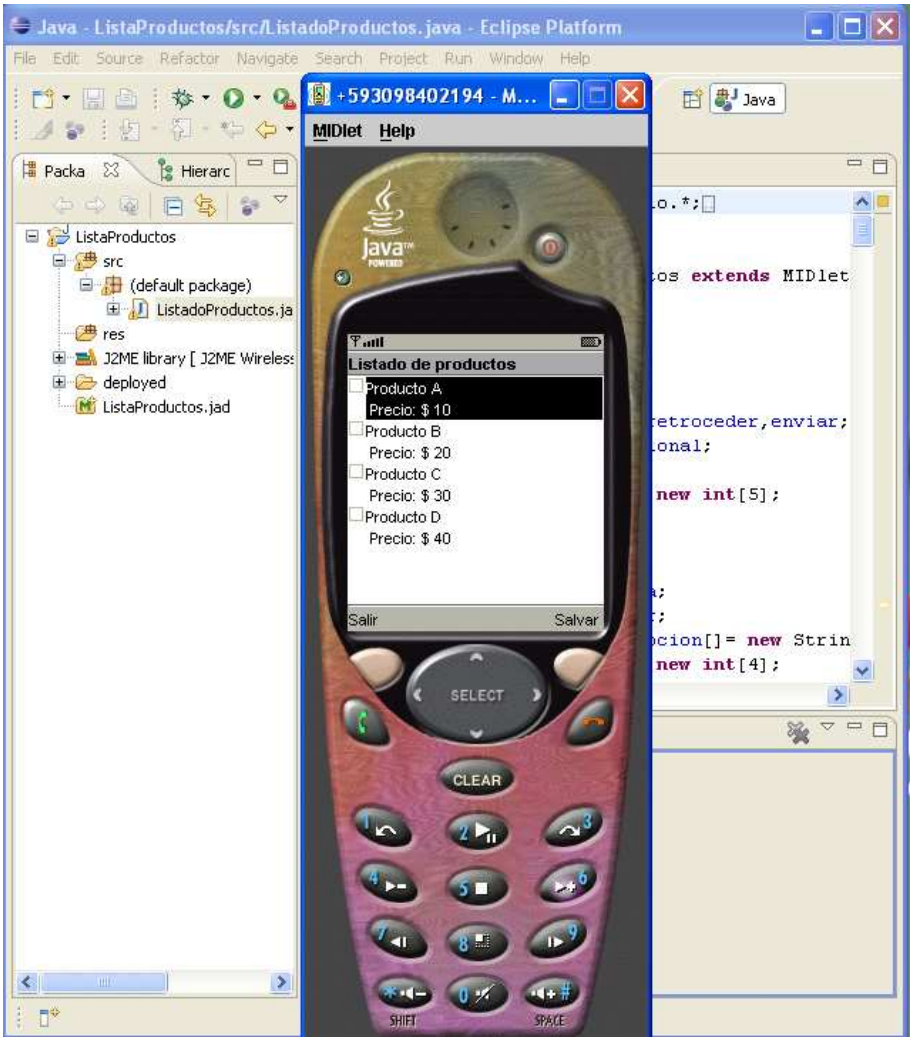


Figura 7 Ejecución de la aplicación.

7.4.1. Transferencia de la aplicación a un dispositivo móvil

Antes de nada se debe anotar que una aplicación J2ME está formada por un archivo JAR que es el que contiene a la aplicación en sí y un archivo JAD (Java Archive Descriptor) que contiene diversa información sobre la aplicación.

El formato del fichero jad es el siguiente:

```
MIDlet-Push-[n]: [url], [clase], [emisor permitido]
```

Dentro de esta línea, los datos son:

- MIDlet-Push-[n] – Identifica el registro *Push* y n es un número secuencial que comienza en 1.
- url – Es una cadena que identifica el punto de conexión entrante. Este valor será el mismo que tendrá el midlet dentro de c*onector.open()* para recoger la conexión.
- clase – Es el nombre completo de la clase que será activada cuando se detecte la conexión entrante.
- emisor permitido – Es un filtro usado para restringir los servidores que pueden activar el midlet. Se pueden usar comodines, como * (1 o más caracateres) y ¿ (un solo carácter).

```
MIDlet-Push-1: sms://:6064, RecibirSMS, *
```

Una vez comprobado que nuestra aplicación funciona correctamente en el emulador, es hora de pasarlo a un dispositivo móvil. Lo primero que hay que hacer es empaquetar el programa y dejarlo listo para descargarlo al dispositivo. Así que una en el entorno Eclipse, se podrá hacer click con el botón derecho sobre el proyecto y selecciona J2ME -> create package (ver Figura 7.1). Una vez creado los archivos conexionSms.jar y conexionSms.jad dentro del directorio deployed de la carpeta del proyecto.
Una vez que se tenga éstos dos archivos, sólo queda transferirlos al dispositivo móvil mediante infrarrojos, bluetooth o el cable de transferencia de datos de nuestro móvil. Otra forma de hacerlo, es subir estos dos archivos a un servidor wap y descargarlo desde el móvil vía OTA. Para esto es necesario que el dispositivo cuente con Internet y soporte GPRS.

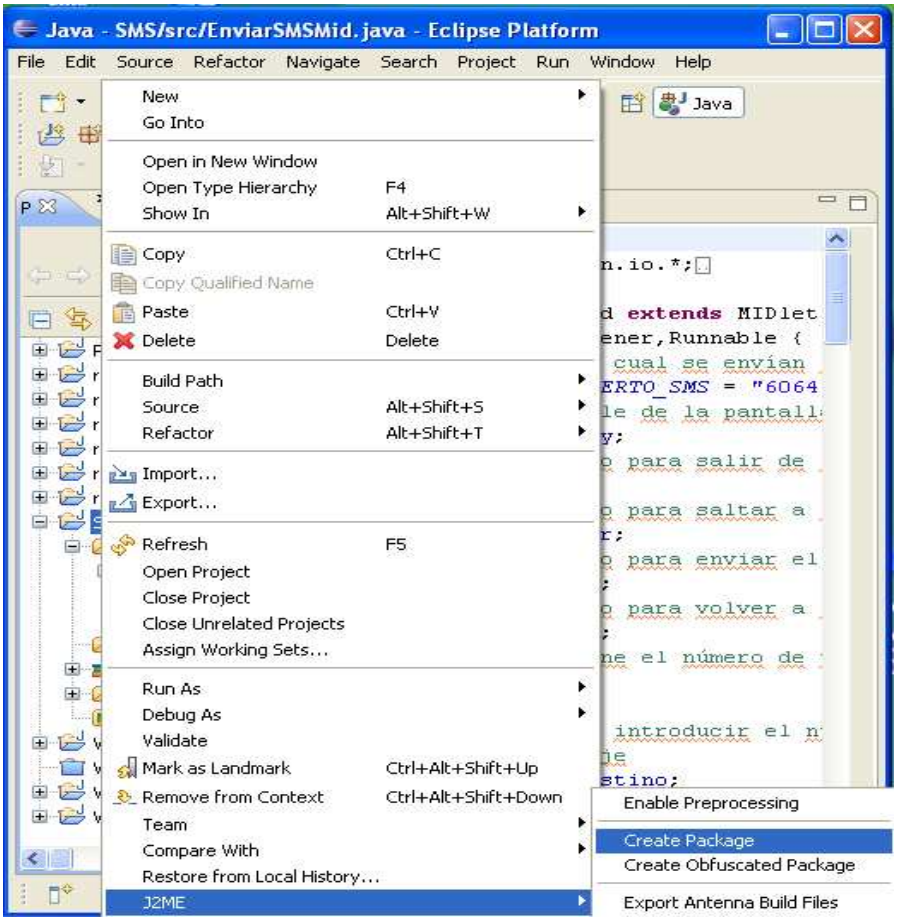


Figura 7.1 Generación de ConexionSms.jar y ConexionSms.jad.

8. CONCLUSIONES Y RECOMENDACIONES

8.1. Conclusiones

Al realizar la presente monografía pensamos que J2ME es el lenguaje idóneo para desarrollar este tipo de proyectos. Quizá el principal problema que notamos es a la hora de realizar aplicaciones de cierta complejidad sean las reducidas, pero obligadas, librerías que lo componen. Es cierto que es obligada porque las mínimas capacidades de cómputo de los teléfonos móviles exigen que así sea. Pero hay que reconocer que el mercado de la telefonía móvil crece día a día y, es de suponer, que Sun Microsystems no dejará sin soporte y ampliará las posibilidades de este lenguaje que es y será el lenguaje de referencia para este tipo de programación.

Al principio se consideró realizar el proyecto en la plataforma Windows XP Professional Edition pero luego de desarrollar y al momento de poner a prueba se lo cambió a la plataforma Linux, ya que el

servidor de la universidad se encuentra en la mencionada plataforma, cabe indicar que el servlet generado se ejecuta de igual manera en las dos plataformas.

Durante el desarrollo del presente trabajo se pensó que se debía instalar las herramientas de desarrollo de software en el servidor, pero durante el período de puesta a prueba se conoció que no es necesario realizar esto, ya que solo se necesitaba un contenedor de servlets, Java como máquina virtual y un gestor de base de datos, por lo se tuvo que instalar solamente el JDK y Tomcat, luego ubicar el archivo .WAR en el directorio /webapps como ya se indicó.

Realizando ésta monografía encontramos dificultades de recibir mensajes en nuestra aplicación receptora de J2ME del dispositivo móvil, esto fue debido a que se utilizó un celular de la operadora Porta y otro de Movistar, por lo que tuvimos que efectuarlo con igual operadora (se probó con móviles activados en la operadora Porta), ya que utilizan diferentes puertos en sus centrales.
Se buscó información acerca de éste inconveniente en los locales de Atención al Cliente de dicha operadora en nuestra cuidad pero no fue posible ya que o desconocían del problema y de sus soluciones o porque tal información se la maneja solamente en Quito y Guayaquil y de forma reservada.

8.2. Recomendaciones

Debido a que la presente monografía debe de ser terminado en un plazo de tiempo prefijado, caso contrario estaríamos incluyendo mejoras prácticamente cada día, existen cosas que no hemos podido incluir en este proyecto pero que pueden servir como avances para el futuro.

He aquí algunas líneas futuras:

- Debido a que una aplicación para un dispositivo móvil puede ser utilizada por cualquier persona que tenga acceso al dispositivo, es recomendable incluir en la aplicación un sistema de control de acceso basado en usuario y contraseña.
- Una buena idea sería la de crear una aplicación para que la base de datos de la central del negocio pueda ser mantenida (borrado/adición de nuevas ofertas, precios, etc.) por los empleados, por supuesto no con todos los privilegios.
- Recomendamos crear una aplicación receptora en el computador conectada al dispositivo móvil que interactúe con la base de datos al confirmar el pedido, actualizándola y permitiendo la impresión del pedido y su respectiva facturación.

Finalmente esperamos que en nuestro medio las operadoras de comunicación inalámbrica pongan mayor información a disposición de usuarios y desarrolladores que usan ésta tecnología, ya que por el

momento solo se encuentra información comercial y la que respecta a lo técnico es muy reducida, pues así permitirían la aparición de nuevas y muy variadas aplicaciones que pueden ayudar a mejorar el estilo de vida de nuestra sociedad, ya que en otros lugares en los que ya se aprovecha ésta tecnología se ha demostrado que existen muchos ámbitos en los que se emplea.

Apéndice A: Manual de instrucciones

Manual de Instrucciones del Midlet Pedidos

Cuando se ejecuta el midlet, lo primero que se mostrará es la Conexión.

Conexión

Lista de Productos

Ahora se debe, pulsar OK. Si la conexión ha sido afirmativa se verá el formulario de Lista de Productos disponibles y podrá seleccionar el o los productos deseados.

Elegir Cantidad

Lista de Pedidos

Hecho esto, el midlet mostrará un nuevo formulario con la lista de pedidos efectuados con su respectiva cantidad disponible. Pulsando “Atrás” se regresará a la pantalla anterior.

Pulsando sobre “Pedir” llegará a un cuadro donde se resume la compra además en el cual se deberá ingresar la dirección. Comprobar que los datos de compra han sido correctos y pulsar “Confirmar”. Se desplegará una última pantalla, donde indica que el pedido ha sido efectuado.

Ahora podemos salir del programa.

Bibliografía

Sitio oficial de Sun Microsystems que trata a cerca de la configuración

CLDC; http://java.sun.com/products/cldc/
5 de agosto de 2007

Tutorial de Java de Sun Microsystems:
http://java.sun.com/docs/books/tutorial/index.html
5 de agosto de 2007

Información sobre Apache Tomcat:
http://jakarta.apache.org/tomcat/tomcat-5.0-doc/
15 de octubre de 2007

Sobre MySQL: http://dev.mysql.com/doc/refman/5.0/en/what-is.html
15 de octubre de 2007

Sitio oficial de Eclipse
http://eclipse.org
28 de julio de 2007

Documentación de J2ME Wireless Toolkit 2.2
http://java.sun.com/j2me
12 de Julio de 2007

Sergio Gálvez Rojas y Lucas Ortega Díaz ,"Java a tope- J2ME (Java 2 Micro Edition)", Edición Electrónica disponible en la web:
http://www.lcc.uma.es/~galvez/J2ME.html
12 de Julio de 2007, nombre del sitio The Java ME Platform- the most Ubiquitous Application Platform for Mobile Devices.