ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN
UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Proyecto Fin de Carrera

Evaluación de Alternativas de Comunicacion entre PC-Teléfono Móvil sobre Java

AUTOR: Antonio Guillén Pascual
DIRECTOR(ES): Juan Ángel Pastor Franco

12 / 2007

Autor	Antonio Guillén Pascual
E-mail del Autor	Antonio.guillen@gmail.com
Director(es)	Juan Ángel Pastor Franco
E-mail del Director	Juanangel.pastor@upct.es
Título del PFC	Evaluación de Alternativas de Comunicación entre PC-Teléfono Móvil sobre Java
Descriptores	Java ME, comunicación móvil, visulalización móvil.

Resúmen

El principal fin de este proyecto es la evaluación de diferentes alternativas de comunicación entre un proceso Java ejecutándose en un PC y un programa Java ejecutándose en un teléfono móvil. Los objetivos que persigue este proyecto se pueden dividir en tres puntos fundamentales:

→ Evaluación de alternativas de comunicación que ofrece el lenguaje Java para comunicar un teléfono móvil con un PC.

→ Proporcionar ejemplos didácticos para la docencia.

→Control de la maqueta de trenes que se encuentra en el DSIE a través de un teléfono móvil.

Titulación	Ingeniería Técnica de Telecomunicación, Especialidad Telemática
Intensificación	
Departamento	TIC
Fecha de Presentación	12- 2007

INDICE

FIGURAS

TABLAS

Capítulo 1.

Introducción

1.1. Contexto y motivación

El mundo de las telecomunicaciones avanza muy deprisa y cada día que pasa no hace más que ofrecer nuevas posibilidades tecnológicas para la comunicación entre diversas plataformas. Un ejemplo en este sentido es la comunicación entre un teléfono móvil y un PC. Hace no mucho tiempo este tipo de comunicación era impensable, puesto que ni los teléfonos móviles ni la infraestructura de servicios que ofrecían las compañías telefónicas tenían capacidad para ofrecer este servicio. Actualmente, gracias al avance espectacular que están teniendo tanto los teléfonos móviles como los nuevos servicios que ofrecen las compañías telefónicas y la aparición de lenguajes como Java MicroEdition, en continua evolución que ofrece cada día que pasa más prestaciones para el desarrollo de aplicaciones para móviles, ya no hace falta usar la imaginación para ver las enormes posibilidades que ofrece este tipo de comunicación entre PC y móvil.

Dentro de este contexto, la principal finalidad de este proyecto es la evaluación de diferentes alternativas de comunicación entre un proceso Java ejecutándose en un PC y un programa Java ejecutándose en un teléfono móvil.

1.2. Objetivos del proyecto

Los objetivos que persigue este proyecto se pueden dividir en tres puntos fundamentales:

→ Evaluación de alternativas de comunicación que ofrece el lenguaje Java para comunicar un teléfono móvil con un PC.

→ Proporcionar ejemplos didácticos para la docencia.

→ Control de la maqueta de trenes que se encuentra en el DSIE a través de un teléfono móvil.

El primer objetivo que se quiere alcanzar con este proyecto es la evaluación de diferentes alternativas de comunicación que ofrece el lenguaje Java para comunicar un teléfono móvil y un PC. La finalidad es ofrecer alternativas para el intercambio de información entre ambas plataformas a través de Internet. La información que se desea intercambiar puede ser texto o imágenes. Se pretende que el lector de este documento, sea capaz de ver las posibilidades que ofrece cada una de las alternativas que se muestran.

Otro objetivo que se propone es que este documento proporcione ejemplos didácticos para la docencia, ya que abre una nueva vía de estudio en las comunicaciones, en este caso entre dos plataformas diferentes como son la tecnología móvil y un PC, de reciente aparición.

Finalmente, es también objetivo de este proyecto la creación de una aplicación para teléfonos móviles desde la cual realizar el control de la maqueta de trenes situada en el DSIE. Con este proyecto se pretende dar soporte para la creación de una aplicación que se ejecuta desde el móvil capaz de mandar comandos de control sobre la maqueta de trenes y además poder visualizar imágenes del estado de la maqueta en tiempo real.

1.3. Planteamientos y fases

El planteamiento que se ha de seguir para la realización del proyecto es:

1. Estudio del lenguaje Java MicroEdition, para obtener las posibilidades de comunicación que puede ofrecer para definir diferentes alternativas de comunicación.

2. Estudio de la infraestructura necesaria para realizar cada una de las alternativas obtenidas del estudio del lenguaje.

3. Definir las diferentes alternativas junto con la infraestructura necesaria para realizar la comunicación.

4. Estudio de cada uno de los componentes que intervienen en cada una de las infraestructuras.

5. Estudiar y definir un protocolo de comunicación para cada una de las alternativas.

6. Elección de programas externos necesarios para la construcción de cada una de las alternativas.

7. Desarrollo de los componentes de cada uno de los elementos que componen la infraestructura. Este desarrollo siempre será en una de la diferentes API’s que ofrece Java.

8. Depuración y corrección de errores en los componentes.

9. Comprobación del correcto funcionamiento de la comunicación entre el móvil y el PC bajo el uso de cada alternativa.

1.4. Estructura de la memoria

Con el fin de que este documento sirva como herramienta docente, esta memoria se ha estructurado de tal forma que el lector se vaya introduciendo poco a poco en las posibilidades que ofrece el lenguaje Java para la comunicación entre dos plataformas diferentes como son la de un teléfono móvil y un PC.

Esta memoria está dividida en 6 capítulos, en los se irán mostrando al lector diferentes alternativas para el intercambio de información entre ambas plataformas. El Capítulo 1 presenta esta introducción que espera servir para que el lector obtenga la idea fundamental sobre lo que trata este proyecto. El Capítulo 2 presenta las primeras alternativas de comunicación entre ambas plataformas para el intercambio de información en modo texto. En el Capítulo 3 se presentan otras alternativas para la visualización de imágenes a través del teléfono móvil. En Capitulo 4 se muestra un ejemplo práctico a través del cual se podrá controlar una maqueta de trenes situada en el DSIE. El Capitulo 5 muestra las conclusiones obtenidas del desarrollo de todas las alternativas mostradas en el proyecto. El último capítulo, el Capítulo 6, muestra la bibliografía utilizada para el estudio y desarrollo de todas las alternativas que se presentan en este documento.

Capítulo 2. Comunicación entre un móvil y un PC.

En este capítulo se describen dos alternativas para el intercambio de información en modo texto entre un teléfono móvil y un PC. La finalidad de este capítulo es que la aplicación que se ejecuta en el teléfono móvil sea capaz de mandar información al PC para que este la trate y mande la correspondiente respuesta. La información que se manda desde el móvil podrá verse como comandos a ejecutar en la aplicación que se ejecuta en el PC.

Persiguiendo este fin, se presentan dos alternativas:

- **Comunicación a través de Servlets**

  Esta alternativa de comunicación utiliza un Servlet como pasarela entre el teléfono móvil y la aplicación Java que se ejecuta en el PC. Dentro de esta alternativa se presentan todos los componentes que forman parte de la infraestructura necesaria para poder realizar esta comunicación, la implementación de los componentes, el protocolo, la estructura y el comportamiento de los mismos.

- **Comunicación directa a través de Sockets**

  Esta alternativa presenta una forma directa de comunicar un teléfono móvil y un PC directamente usando Socket. Dentro de esta alternativa se presentan todos los componentes que forman parte de la infraestructura necesaria para poder realizar esta comunicación, la implementación de los componentes, el protocolo, la estructura y el comportamiento de los mismos.

Capítulo 3. Visualización de imágenes en tiempo real.

Con el fin de entender mejor como es el proceso de visualización de imágenes a través del móvil, se proponen dos alternativas diferentes, la primera de ellas a través del protocolo TCP y la segunda usando el protocolo UDP. Como ya se ha mencionado anteriormente este documento pretende servir como herramienta docente, por lo que estas dos alternativas son una forma sencilla y didáctica para aprender sobre la visualización de imágenes en tiempo real a través del móvil. Hay otros mecanismos más complejos y eficientes como la visualización de imágenes en tiempo real mediante el uso del protocolo RTP/RTCP, que hay que tener en cuenta, pero que no se han introducido en este proyecto, ya que se escapan de la finalidad del mismo.

Con estas alternativas para la visualización, se pretende que desde un teléfono móvil se pueda ver las imágenes en tiempo real ofrecidas por una webcam. Las dos alternativas que se presentan son:

- **Visualización usando el protocolo TCP**

  Esta alternativa presenta la posibilidad de visualizar imágenes en tiempo real por la pantalla del teléfono móvil usando el protocolo de transporte TCP/IP. Para realizar esta visualización solo hace falta una aplicación que capture imágenes en formato JPG y servidor Web al que el móvil le solicitara las imágenes. Además se mostrarán los componentes que forman la infraestructura junto con su protocolo, su estructura y comportamiento.

- **Visualización usando el protocolo UDP**

  Esta alternativa presenta otra posibilidad, más eficiente, pero más compleja, que la anterior, para visualizar las imágenes en tiempo real en el teléfono móvil. Es más compleja porque precisa de un protocolo para el control de visualización. Gracias a este protocolo de control se podrá iniciar o finalizar la visualización de las imágenes. Se presenta además la infraestructura necesaria, los componentes que la componen, la estructura de los mismo y su comportamiento y el protocolo de comunicación que se sigue.

Capítulo 4. Ejemplo de Aplicación. Control de una maqueta de trenes.

En este capítulo se explica como haciendo uso de las alternativas mostradas en los capítulos anteriores se puede construir una aplicación capaz de realizar el control sobre una maqueta situada en el DSIE. La infraestructura básica para realizar este control sobre la maqueta la componen tres nodos: el móvil, el Servidor de Red y la maqueta de trenes. Los componentes que componen la aplicación que se ejecuta en el nodo móvil y en el Servidor de Red se han desarrollado en lenguaje Java, y tienen una descripción para su instalación, ejecución y manejo de los mismos.

Capítulo 5. Conclusiones.

Una vez se han presentado todas las alternativas, se mostraran las ventajas y desventajas que ofrecen cada una de ellas, posibles mejoras para la evolución de las alternativas, requerimientos de Hardware, etc…

Capítulo 6. Bibliografía.

En este capítulo se puede encontrar los libros y enlaces WEB que se han usado para el desarrollo de este proyecto.

Apéndice A. Introducción a Java MicroEdition.

En este apéndice se explican conceptos básicos de Java ME porque es interesante, aunque no necesario, conocer ciertos conceptos relacionados con java y esta versión Microedition (J2ME).

Apéndice B. Manual básico de Servlet.

En este apéndice se resumen los conceptos básicos de un Servlet a la vez que se enumeran los pasos necesarios para la creación de los mismos.

Capítulo 2.

Comunicación entre un móvil y un PC

2.1. Introducción

Las comunicaciones móviles cada día tienen mayor importancia, es una tecnología en constante evolución, que ofrece cada día más y mejores servicios para la comunicación. Lo mismo sucede con los terminales, los bien conocidos "móviles", los cuales avanzan acordes con los nuevos servicios que ofrecen los operadores. De este modo, actualmente hay terminales que son capaces de navegar por Internet, hacer llamadas de voz, videoconferencias en tiempo real, reproducción de música o de vídeos, etc.

Con el auge de los servicios de Internet en los operadores móviles y los nuevos terminales que ofrece actualmente el mercado, ya se puede pensar que el móvil puede servir para algo más que para hacer llamadas de voz, videoconferencias o navegar por Internet.

Actualmente, la mayoría de los móviles soportan aplicaciones programadas en Java MicroEdition (Java ME). La mayoría de ellas son juegos para entretener al propietario del móvil. Pero si utilizamos un poco la imaginación podríamos pensar en utilizar dicho lenguaje para realizar aplicaciones acordes a las necesidades de cualquier empresa. Un ejemplo claro, sería controlar una central de domótica desde cualquier lugar del mundo a través del móvil.

2.2. Objetivos

Este capítulo versa sobre el primero de los objetivos del proyecto, establecer los conceptos básicos en la comunicación entre un móvil y un PC para el intercambio de información.

Se desea establecer un flujo de información entre el terminal móvil y la aplicación ejecutada en el PC. Esta información constará de una serie de comandos que el móvil podrá mandar al PC para que este los ejecute. Es decir, la aplicación del móvil ejerce el rol de cliente de los servicios que ofrece la aplicación que se ejecuta en el PC.

Para conseguir estos objetivos se programarán las aplicaciones de cada uno de los nodos en el lenguaje Java.

2.3. Definición de alternativas

Para la consecución de los objetivos planteados, se van a plantear dos alternativas, cada una de ellas con sus ventajas y desventajas. Estas alternativas tienen en cuenta que el flujo de información debe ser bidireccional, es decir, deben asumir, que tanto el móvil como la aplicación de PC, puedan generar o recibir información una vez se establezca la comunicación.

Las dos alternativas planteadas son: la comunicación a través de Servlets y la comunicación directa a través de Sockets.

2.3.1. Comunicación a través de Servlets

Esta alternativa consiste en establecer la comunicación entre el móvil y el PC usando un intermediario que es un servidor WEB con capacidad de ejecutar Servlets (Véase Figura 1). Los **servlets** son módulos java que nos sirven para extender las capacidades de los servidores WEB, con ellos, podemos generar páginas WEB de forma dinámica. Para quien esté familiarizado con el tema de las CGI [13], los Servlets son su equivalente en Java.

De esta forma, el móvil a través de una URL manda los comandos al Servidor WEB (más adelante se explicará como se realiza este paso), éste ejecuta el Servlet que extrae el comando de la URL y posteriormente establece una comunicación con la aplicación del PC mediante sockets, para transmitir el comando recibido desde el móvil. La aplicación genera la respuesta, que se la envía al Servlet, y este a su vez se la transmite al móvil.

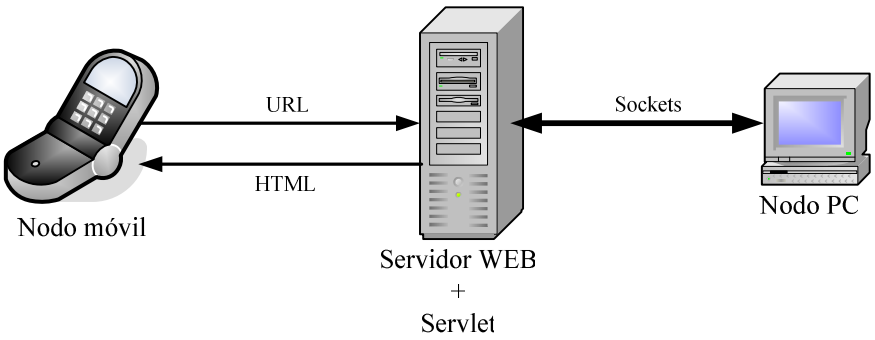


Figura 1. Nodos que intervienen en la comunicación a través de Servlet

2.3.2. Comunicación directa a través de Sockets

La otra alternativa que se plantea es la comunicación entre el móvil y la aplicación del PC haciendo uso de los sockets (Véase Figura 2). Un **socket** es un punto final en una conexión existente entre un programa cliente (en nuestro caso el móvil) y un programa servidor (en nuestro caso la aplicación), pudiendo mediar entre ambos una red de comunicación [1]. De esta manera no es necesaria la utilización de intermediarios.

El proceso que se sigue para desarrollar la comunicación es relativamente sencillo, el móvil manda el comando a través de un conexión de socket a la aplicación. Esta recibe el comando lo ejecuta, y devuelve la información pertinente al móvil en una única conexión, ya que el flujo de información es bidireccional.

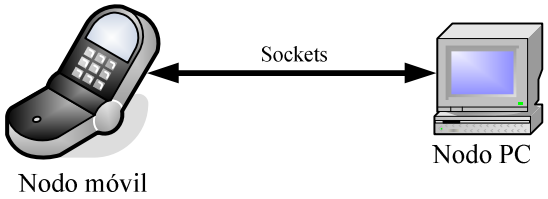


Figura 2: Nodos que intervienen en la comunicación directa a través de Sockets

2.4. Ventajas y desventajas de las alternativas

Las ventajas y desventajas que ofrece cada alternativa son las que se muestran en la Tabla 1:

	Ventajas	Desventajas
Comunicación a través de Servlets	- Compatibilidad con la mayoría de móviles disponibles en el mercado. Es suficiente con que el móvil pueda acceder a Internet. - Compatibilidad con diferentes plataformas. No sólo se podría mandar información desde un móvil, sino que también se podría desde un navegador WEB.	- Necesidad de intermediarios en la comunicación, que implica que si se cae el servidor WEB, aunque los demás nodos funcionen correctamente, la comunicación no se puede establecer. - La comunicación se hace menos segura, ya que el envío de la información se hace a través de la URL. - Comunicación más lenta. - Mayor infraestructura. Más costoso su mantenimiento
Comunicación directa a través de Sockets	- No son necesarios intermediarios, lo que hace que sea una comunicación mucho más ágil. - Puede realizarse una comunicación mucho más segura si fuese preciso. - Comunicaciones más rápidas.	- Incompatibilidad con muchos teléfonos móviles disponibles en el mercado.

Tabla 1. Ventajas y desventajas de las alternativas de comunicación.

2.5. Comunicación a través de Servlets

2.5.1. Infraestructura

Para el buen desempeño de ésta comunicación, es necesaria la participación de 3 nodos en la comunicación: el móvil, el servidor WEB con capacidad de ejecutar Servlets y un PC encargado de ejecutar una aplicación JAVA que hace de servidor (Véase Figura 3).

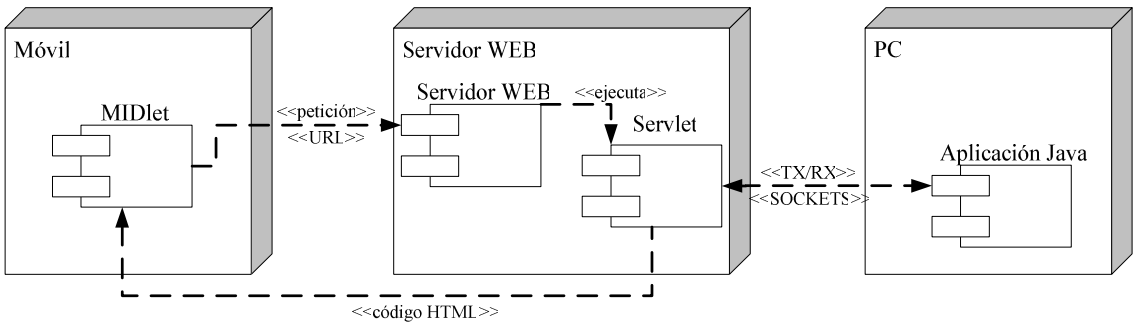


Figura 3. Nodos y componentes de la alternativa de comunicación a través de Servlets

a) Nodo móvil

Se precisa de un dispositivo móvil compatible con las API's MIDP 1.0 y CLDC 1.0 de Java ME (Véase Apéndice A). Es suficiente que dicho móvil sea compatible con estas API's pero es altamente recomendable que sea compatible con MIDP 2.0 y CLDC 1.1, puesto que estas hacen que nuestro dispositivo presente menos problemas a la hora de realizar cualquier tipo de comunicación de red.

Para el desarrollo práctico no hemos utilizado ningún móvil, sino un simulador de móvil ejecutándose sobre un PC. Este simulador se encuentra dentro del "*Java Wireless Toolkit 2.5 for CLDC"*, que no es más que un conjunto de herramientas para el desarrollo de aplicaciones en Java ME.

La aplicación cliente del móvil (MIDlet) será capaz de enviar comandos, predefinidos previamente, vía URL utilizando el método GET del protocolo HTTP. Posteriormente, esperará la respuesta ofrecida por el Servlet en formato HTML, para mostrarla por pantalla.

b) Nodo servidor WEB

Para esta parte de la comunicación se precisa de un servidor WEB con capacidad de ejecutar Servlets.

Para poder establecer la comunicación utilizamos un PC, en el que instalaremos el servidor de Servlets Tomcat. Como ya se ha mencionado, este servidor es capaz de ejecutar Servlets y mostrar páginas de contenido dinámico como nos interesa, pero además, ofrece la posibilidad de mostrar páginas WEB con contenido estático. Este servidor lo podrá descargar de forma gratuita desde http://tomcat.apache.org.

El Servlet es capaz de extraer los comandos de la URL recibida desde el móvil, para posteriormente mandarle dicho comando a la aplicación JAVA (realiza la función de servidor), a través de Sockets. Una vez procesado el comando por la aplicación, se recibe una respuesta de la aplicación JAVA, que el Servlet transmite en formato HTML al móvil (Véase Figura 1).

c) Nodo PC

Este nodo consiste en un PC, en el que se ejecutará una aplicación Java en modo servidor. Es decir, capaz de aceptar conexiones mediante Sockets, para ofrecer un servicio.

En nuestro caso, la aplicación espera conexiones para que le transmitan un comando, cuando dicho comando es recibido, la aplicación cambia su estado según dicho comando. Si se produjo un cambio de estado, devolverá OK a quien le transmitió dicho comando, sino NO_OK. Un comando particular, sería el comando "Estado", que devuelve el estado en el que se encuentra el servidor en dicho momento.

2.5.2. Protocolo

En el diagrama de secuencia de la Figura 4 se muestra como se establece la comunicación y como se intercambia la información entre los distintos nodos.

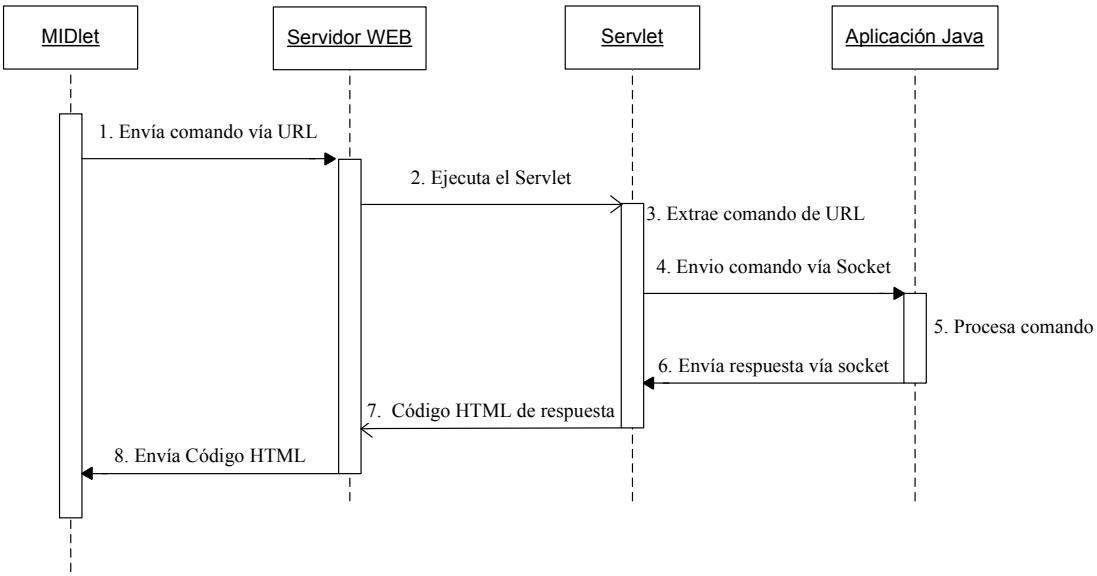


Figura 4. Diagrama de secuencia del protocolo para la comunicación a través de Servlets

2.5.3. Paquetes y Clases JAVA utilizadas

A continuación se mostrarán los paquetes y clases utilizadas en los componentes programados en lenguaje Java: MIDlet, Servlet y Aplicación Java.

• MIDLET

El diagrama de la Figura 5 muestra los paquetes y clases de Java ME utilizados en la realización del MIDlet, que recuérdese se ejecuta en el nodo móvil (Véase Figura 3):

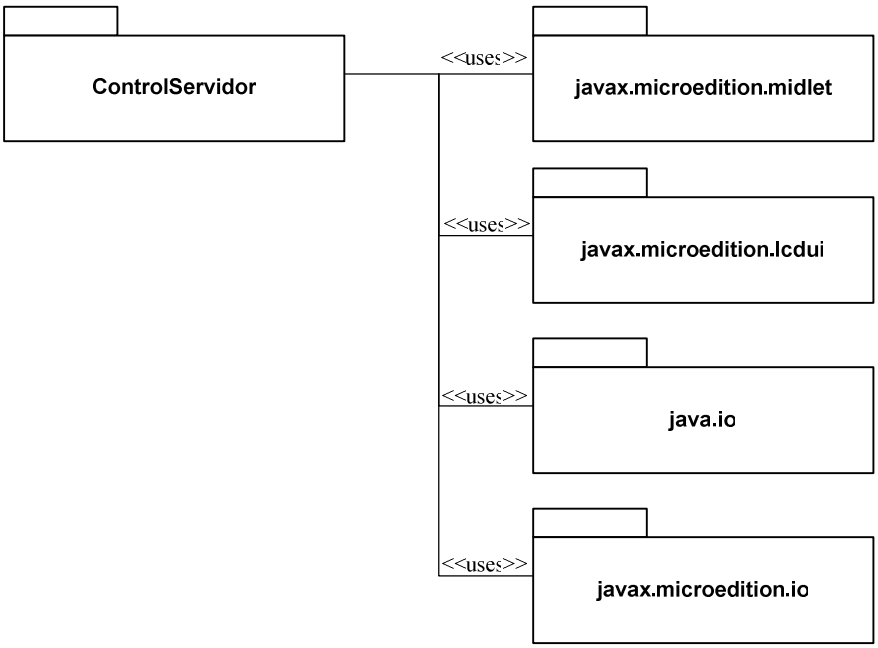


Figura 5. Paquetes Java necesarios para desarrollar el MIDlet

En la Tabla 2, se muestran las clases, interfaces y excepciones utilizadas de cada uno de los paquetes:

Paquete	***Clases***	***Excepciones***	***Interfaces***
javax.microedition.midlet	MIDlet		
javax.microedition.lcdui	Display Form TextBox StringItem TextField ChoiceGroup Command		CommandListener
javax.microedition.io	Connector		StreamConnection
Java.io	InputStream	IOException	

Tabla 2. Clases, excepciones e interfaces necesarios para desarrollar el MIDlet

En la Tabla 3 se destacan los aspectos más relevantes de los elementos utilizados para la realización del MIDlet:

Paquete javax.microedition.midlet	
Clase	**Descripción**
MIDlet	Las aplicaciones deben extender a esta clase para poder ejecutarse en un móvil. Los métodos de esta clase permiten crear, iniciar, pausar o destruir una aplicación para el móvil.
Paquete javax.microedition.lcdui	
Clase	**Descripción**
Display	Clase que se utiliza para manejar lo que se quiere mostrar por la pantalla del móvil.
Form, TextBox	Son subclases que extienden a la clase “screen” del mismo paquete. Sirven como contenedores de elementos para poder ser mostrados por pantalla.
StringItem, TextField, ChoiceGroup,	Clases que sirven para representar información dentro de un objeto “screen” como “Form” y “TextBox”.
Command	Esta clase es un constructor que encapsula información semántica de una acción. La acción pertinente se define en una asociación entre un objeto “Displayable” y el interface “CommandListener” (Véase Apéndice B)
Interface	**Descripción**
CommandListener	Este interface es utilizado por las aplicaciones que necesitan recibir eventos high-level (de alto nivel) desde la implementación.
Paquete javax.microedition.io	
Clase	**Descripción**
Connector	Sirve como factoría de nuevas conexiones, es decir, se utiliza para generar cualquier tipo de conexión desde el MIDlet.
Interface	**Descripción**
StreamConnection	Es un interface que define la capacidad que debe tener una conexión tipo stream. Solo para instancias MIDP 2.0 existe una implementación típica de éste interface.

Tabla 3. Descripción de las clases, excepciones e interfaces más importantes en el desarrollo del MIDlet

• SERVLET

El diagrama de la Figura 6 muestra los paquetes y clases de Java Enterprise Edition (Java EE) utilizados en la realización del Servlet, que recuérdese se ejecuta en el servidor WEB (Véase Figura 3):

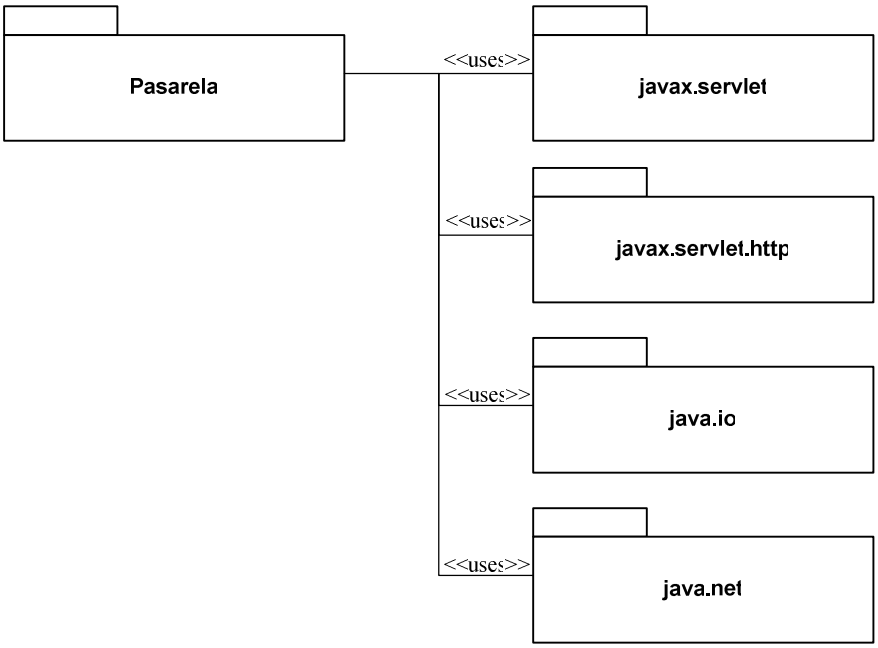


Figura 6. Paquetes Java utilizados en el desarrollo del Servlet

En la Tabla 4 se muestra los paquetes y clases Java utilizados en la realización del Servlet:

Paquete	***Clases***	***Excepciones***	***Interfaces***
javax.servlet		ServletException	
javax.servlet.http	HttpServlet		HttpServletRequest HttpServletResponse
java.io	InputStream OutputStream PrintWriter	IOException	
java.net	Socket		

Tabla 4. Clases, excepciones e interfaces utilizados en el desarrollo del Servlet

En la Tabla 5 se destacan los aspectos más relevantes de los elementos utilizados para la realización del Servlet:

Paquete javax.servlet.http	
Clase	**Descripción**
HttpServlet	Clase abstracta que debe ser extendida por la aplicación para poder ser ejecutado como un Servlet por un servidor WEB capaz de ejecutar éste tipo de aplicaciones.
Interface	**Descripción**
HttpServletRequest, HttpServletResponse	Interfaces que proporcionan la capacidad al Servlet para recibir y generar peticiones HTTP.

Tabla 5. Descripción de las clases e interfaces destacables del Servlet

• APLICACIÓN PC

Se muestran en el diagrama de la Figura 7 el conjunto de paquetes Java Standard Edition (Java SE) para la realización de la aplicación que se ejecuta en el nodo PC (Véase Figura 3):

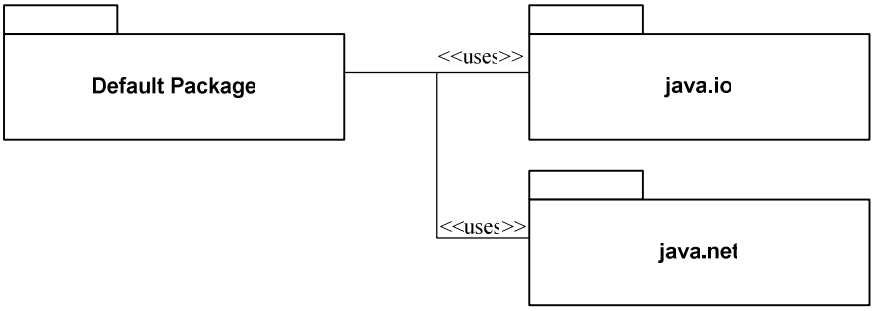


Figura 7. Paquetes Java utilizados en el desarrollo de la aplicación del PC.

Por último, en la Tabla 6 se muestra los paquetes y clases Java utilizados en la aplicación que hace de servidor:

Paquete	*Clases*	*Excepciones*	*Interfaces*
java.io	InputStream OutputStream	IOException	
java.net	ServerSocket Socket		

Tabla 6. Clases, excepciones e interfaces usados en el desarrollo de la aplicación del PC

2.5.4. Estructura y comportamiento

a. MIDlet

En el diagrama de la Figura 8 se puede observar la estructura de la aplicación que se ejecuta en el nodo móvil.

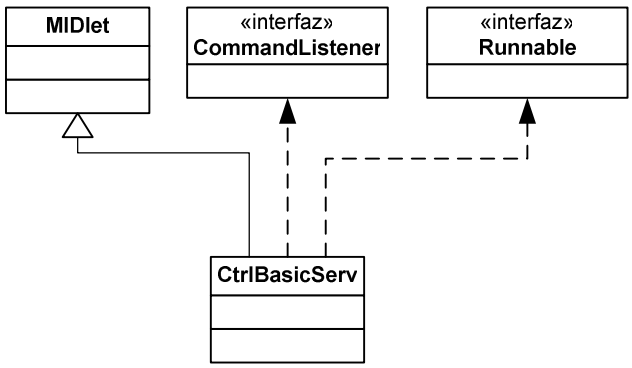


Figura 8. Estructura de clases del MIDlet

La clase `CtrlBasicServ` extiende a la clase MIDlet, lo cual implica, que el nodo móvil es capaz de ejecutar `CtrlBasicServ`. Además como también se puede ver, `CtrlBasicServ` implementa las interfaces `CommandListener` y `Runnable`. La primera para definir las acciones surgidas de eventos de comandos, y la segunda para poder crear nuevos hilos de ejecución, desde los cuales se establecen la comunicación.

b. Servlet

Obsérvese en el diagrama de la Figura 9, que la clase `Pasarela` extiende a la clase abstracta `HttpServlet`, de ésta forma, cualquier servidor de Servlets podrá ejecutar la clase implementada. La clase `HttpServlet` a su vez extiende a la clase `GenericServlet`. Destacar también, que la clase `Pasarela` contiene un `Socket`, que será el encargado de establecer e intercambiar información en la comunicación.

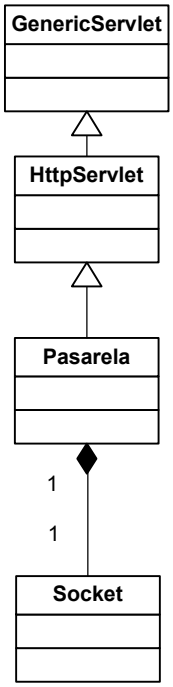


Figura 9. Estructura de clases del Servlet

c. Aplicación Java

En el diagrama de la Figura 10 se destacan los aspectos más relevantes de la aplicación Java que se ejecuta en el nodo PC (Véase Figura 3). Se puede observar que la clase `minimoServidor` contiene el método *main* que ejecuta la aplicación en sí. De dicho método, se destacarán la creación de objetos las clases `ServerSocket` y `Socket`, encargadas de realizar el establecimiento e intercambio de información en la comunicación.

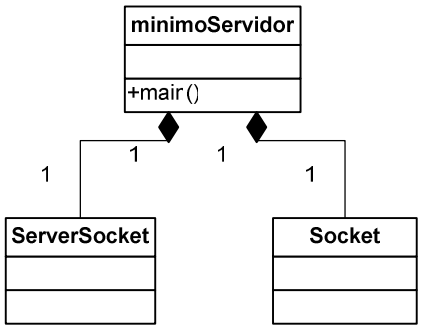


Figura 10. Estructura de clases de la aplicación Java del PC

2.5.5. Comportamiento. Interacción entre objetos.

En el diagrama de la Figura 11 se puede observar como y cuando interviene cada componente en la comunicación que se pretende llevar a cabo. Se destaca, principalmente, la necesidad de crear un nuevo hilo de ejecución (`Thread`) para que se encargue de la transmisión de comando y la recepción de la información que es enviada por el Servlet. De esta forma, se puede realizar la comunicación sin que se generen problemas a la hora de establecer la comunicación, ya que si no se hace esto, la aplicación podría bloquear el móvil impidiendo cualquier acción sobre el mismo.

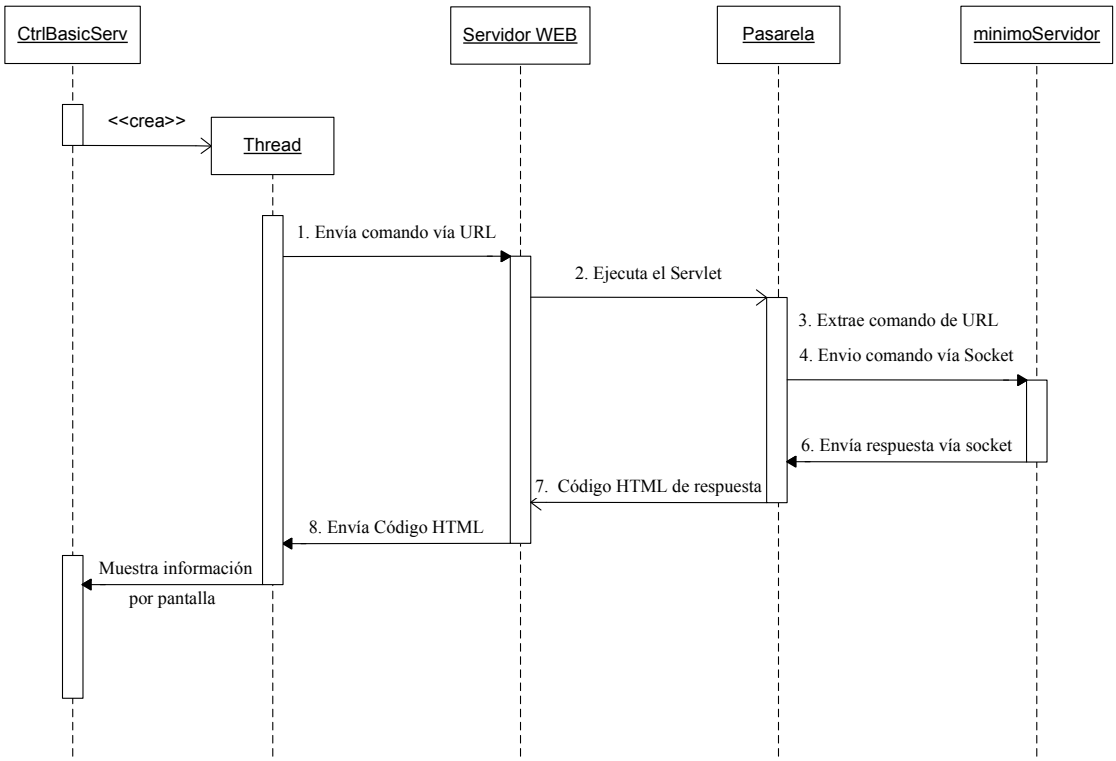


Figura 11. Diagrama de secuencia de la comunicación a través de Servlet

2.5.6. La aplicación. Instalación, ejecución y manejo.

2.5.6.1. Nodo móvil

La aplicación que se ha creado para ser ejecutada en el nodo móvil se llama `CtrlBasicServ`. Para la creación de una aplicación para móviles, se debe utilizar un interfaz de desarrollo Java capaz de desarrollar aplicaciones MIDP en Java ME (Véase Apéndice B si se desea más información de cómo crear una aplicación MIDP para móviles). Un IDE de Java, que cumple con los requisitos necesarios, es la herramienta *Java Wireless Toolkit 2.5 for CLDC*, que se puede descargar de forma gratuita de la página de Sun (http://java.sun.com). Se hará necesario también tener instalado el *Java SE Development Kit* (JDK) y *Java Runtime Environment* (JRE) para el desarrollo de la aplicación. Estos dos elementos también se pueden descargar de forma gratuita de la página de Sun (http://java.sun.com).

En los siguientes apartados se explican los pasos necesarios para la instalación, la ejecución y el manejo de `CtrlBasicServ` utilizando *Java Wireless Toolkit 2.5 for CLDC* (esta herramienta puede encontrarla de forma gratuita en http://java.sun.com).

a. Instalación

Para instalar la aplicación `CtrlBasicServ` en un teléfono móvil son necesarios los ficheros .jad y .jar. El fichero .jar contiene de forma comprimida, el conjunto de clases, librerías y recursos que utiliza la aplicación MIDP. La utilidad del fichero .jad radica en que es usado por el instalador del teléfono para reconocer / instalar / desinstalar la aplicación en cuestión. El entorno *Wireless Toolkit* es capaz de generar estos dos archivos para su posterior ejecución en cualquier teléfono móvil.

La instalación de la aplicación en el móvil, varía dependiendo de la marca y el modelo del mismo, pero como se comentó anteriormente, se puede ejecutar la aplicación desde el simulador que se encuentra en el conjunto de herramientas de *Java Wireless Toolkit for CLDC 2.5*.

b. Ejecución

Como se ha comentado en el apartado anterior (Instalación), la ejecución se llevará a cabo desde el simulador que se encuentra en el conjunto de herramientas *Java Wireless Toolkit for CDLC 2.5*. Hay dos formas de ejecutar la aplicación en el simulador. La primera de ellas es utilizar los archivos .jad y .jar creados por el IDE. La segunda de ellas es ejecutar la aplicación desde el mismo entorno de desarrollo, ya que este utiliza también el simulador para ejecutar la aplicación que se ha creado sin necesidad de crear los archivos .jad y .jar.

En este caso, se va a utilizar la opción de ejecutar la aplicación a través de los archivos .jad y .jar creados desde el entorno de desarrollo. Hay que tener en cuenta que para realizar todos los pasos que se van a explicar a continuación se debe tener instalado el conjunto de herramientas *Java Wireless Toolkit*. Cuando ya se tiene dicha aplicación instalada los pasos a seguir para la ejecución del proyecto serán:

1º) Abrir el explorador y dirigirse a la carpeta “bin” del proyecto Comunicación.

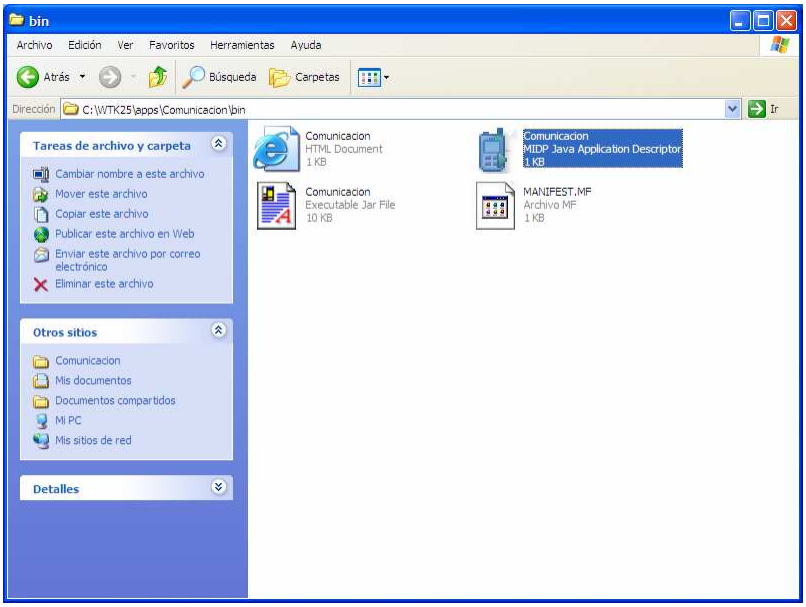


Figura 12. Explorador con la ruta y el jad del MIDlet

2º) Abrimos el archivo .jad del proyecto, en este caso *Comunicacion.jad* que se encuentra señalado en la Figura 12.

3º) Una vez esté abierta la aplicación del proyecto Comunicación seleccionar la opción `CtrlBasicServ` (Véase Figura 13). Recuérdese que así es como se llama la aplicación MIDP que hemos desarrollado para realizar esta comunicación.

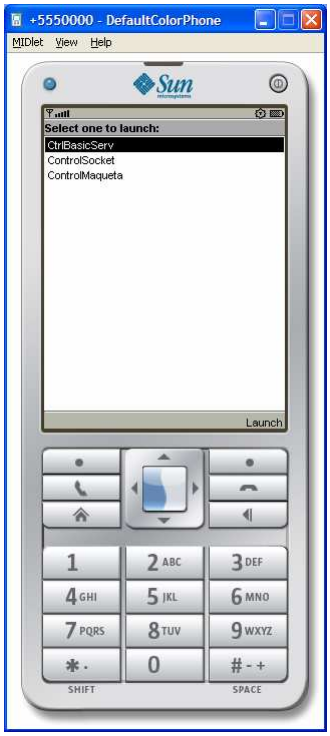


Figura 13. Pantalla para seleccionar el MIDlet

Figura 14. Pantalla inicial del MIDlet

Ahora ya se está ejecutando nuestra aplicación en el nodo móvil (Véase Figura 14).

c. Manejo

Como ya se ha comentado anteriormente, aparecen 3 nodos en la comunicación: el nodo móvil, el servidor WEB y nodo PC. Cada uno de ellos, formado por unos componentes que intervienen directamente en la comunicación (Véase Figura 3 y Figura 11). Esto hace, que si uno de ellos falla, no se está ejecutando o está mal configurado, la comunicación no se pueda realizar.

Los pasos que hay que realizar para mandar un comando a la aplicación Java que hace de servidor son:

1º) Establecer la dirección IP del servidor WEB donde se encuentra el Servlet `Pasarela`.

2º) Establecer puerto por donde ofrece el servicio el servidor WEB.

3º) Seleccionar un comando.

4º) Pulsar el botón de Conectar del móvil. Se pedirá autorización para realizar la comunicación (Véase Figura 15)

5º) Se muestra por pantalla la información recibida desde el Servidor WEB (Véase Figura 16).

La aplicación dispone de un botón "Salir" en la pantalla inicial (Véase Figura 14) para finalizar la ejecución de la misma. Además, cuando se muestra la información recibida del servidor WEB, tenemos la opción de volver a la pantalla inicial pulsando el botón "Volver", pudiendo realizar una nueva comunicación si se desea.

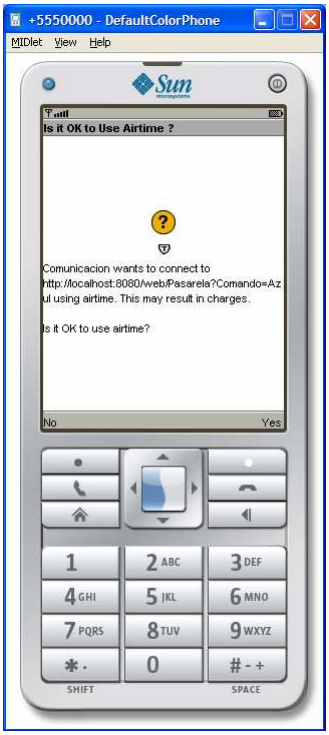


Figura 15. Autorización de conexión

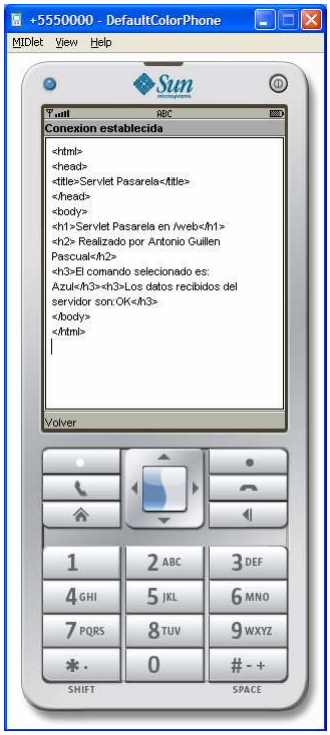


Figura 16. Respuesta recibida del Servlet

2.5.6.2. Nodo Servidor WEB

El Servlet que se ha desarrollado para realizar esta comunicación, se llama `Pasarela`. Para la creación de este tipo de aplicaciones, es necesario tener instaladas las herramientas Java: *Java SE Development Kit* (JDK) y *Java Runtime Environment* (JRE) que se pueden descargar desde la página de Sun (http://java.sun.com).

El interfaz de desarrollo de aplicaciones debe poder generar proyectos de páginas WEB dinámicas. Uno que cumple con esta condición es el *Netbean IDE* que se puede descargar de forma gratuita de www.netbean.org (Véase Apéndice B para más información de cómo crear un Servlet).

a. Instalación

Como ya se ha comentado anteriormente, los **servlets** son aplicaciones java que nos sirven para extender las capacidades de los servidores WEB, con ellos, podemos generar páginas WEB de forma dinámica, con la respuesta que nos ofrezca la aplicación Java.

El servidor WEB que se tiene que utilizar debe ser capaz de ejecutar este tipo de aplicaciones. Un ejemplo de servidor con estas características es el *Apache Tomcat*, que se puede descargar de forma gratuita de http://tomcat.apache.org. Para saber más sobre la

instalación o configuración de este servidor puede verse el manual que se encuentra en http://www.programacion.com/java/tutorial/tomcatintro.

Una vez implementada la aplicación `Pasarela` (el Servlet que se ha desarrollado para esta alternativa de comunicación), desde el entorno de desarrollo que se haya utilizado se puede construir el proyecto. Una vez construido, dentro de la carpeta "build" del proyecto, se encuentra la aplicación `Pasarela`. En la carpeta "build" se encuentra una carpeta raíz de la aplicación (que en nuestro caso se llama web) y dentro de esta contiene dos subcarpetas que son la que contienen la aplicación en sí, META-INF y WEB-INF.

Para instalar dicha aplicación en nuestro servidor Tomcat, tendremos que copiar el contenido de la carpeta "build" del proyecto a la carpeta "webapps" del servidor WEB. Es decir, la carpeta "web" de "build" se debe copiar en la carpeta "webapps". Una vez copiada esta carpeta se debe reiniciar el servidor para que los cambios surjan efecto.

b. Ejecución

Para poder ejecutar el Servlet `Pasarela` se debe saber que éste obtiene la información que debe mandarle a la Aplicación Java usando el método GET del protocolo HTTP [14]. Con este método se puede enviar información a través de la URL, que el Servlet obtendrá para posteriormente mandársela a la Aplicación Java.

Dependiendo de la dirección IP del servidor WEB y el puerto configurado para dar servicio, la URL necesaria para ejecutar dicho Servlet sería:

http://[ip_Servidor]:[puerto]/web/Pasarela[?Comando=].

La ejecución de dicho Servlet se puede realizar desde la aplicación desarrollada para el móvil (el MIDlet), o desde cualquier navegador WEB como puede ser Internet Explorer.

En la Figura 17 se puede observar el código HTML devuelto por el Servlet `Pasarela` mostrado por un navegador cuando la ejecución es:

http://localhost:8080/web/Pasarela?Comando=Azul:

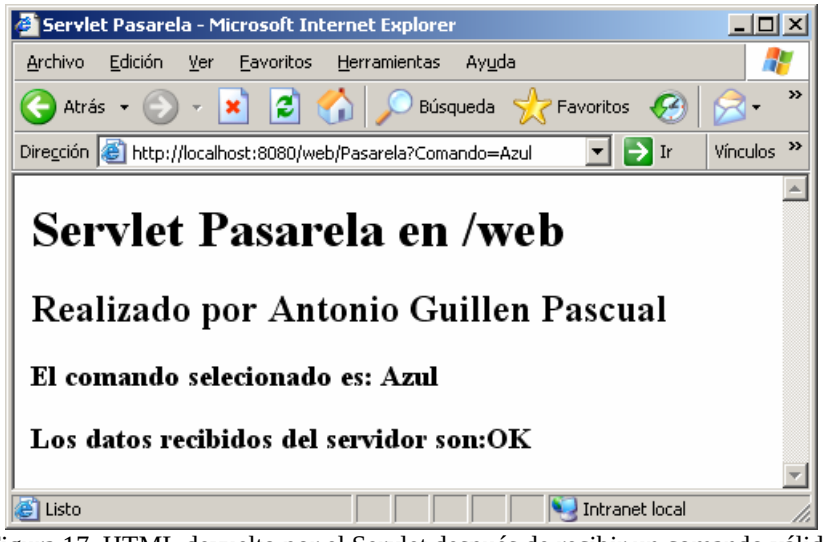


Figura 17. HTML devuelto por el Servlet después de recibir un comando válido

2.5.6.3. Nodo PC

La aplicación Java que se ejecuta en el nodo PC se llama `minimoServidor`. Para desarrollarla es necesario tener instalado el *Java SE Development Kit* (JDK) y *Java Runtime Environment* (JRE) que se pueden descargar de la página de Sun (http://java.sun.com).

Para implementarla se puede realizar desde cualquier interfaz de desarrollo Java, como por ejemplo *Eclipse*. Este se puede descargar de forma gratuita desde www.eclipse.org.

a. Instalación

No requiere instalación.

b. Ejecución

Se ejecuta desde el interfaz de desarrollo, en este caso desde *Eclipse*, eligiendo la opción "Run as Aplication".

c. Manejo

La aplicación `minimoServidor` no requiere manejo alguno, solo espera nuevas conexiones vía `Socket`, cuando estas se establecen, lee la información del lado remoto y lo convierte a una cadena de caracteres (`String`). Esta aplicación esta preparada para recibir 3 tipos distintos de comandos, que son: Azul, Rojo y Estado. Cuando se ha recibido la información se compara para ver si coincide con alguno de estos tres comandos disponibles, si es así la aplicación transmitirá un OK al remoto (Véase Figura 17), mientras que si no, transmite NO_OK (Véase Figura 18).

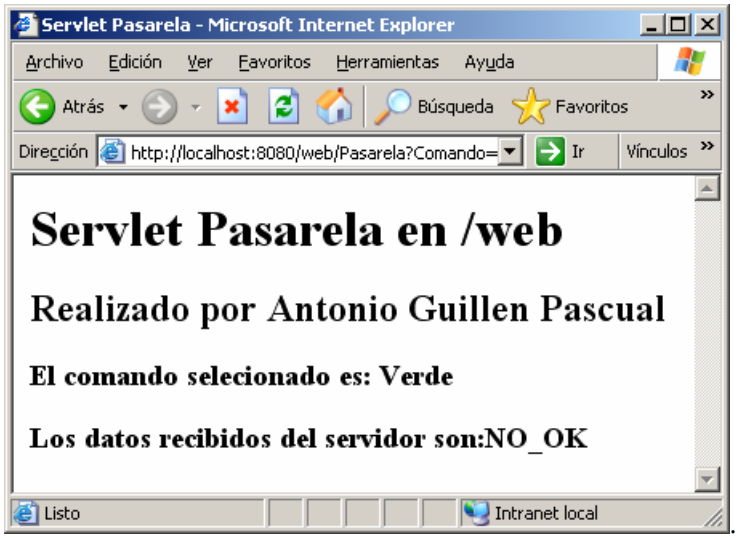


Figura 18. HTML devuelto por el Servlet después de recibir un comando no válido

2.6. Comunicación directa a través de Sockets

2.6.1. Infraestructura

En esta alternativa de comunicación solo son necesarios dos nodos, que transmiten y reciben información mediante Sockets (Véase Figura 19).

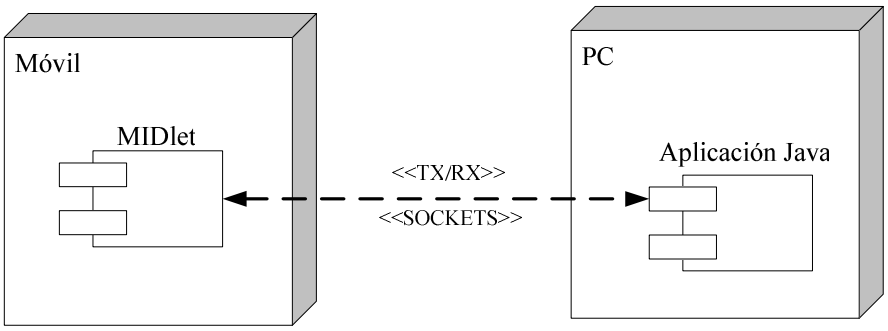


Figura 19. Nodos y componentes de la alternativa de comunicación directa a través de Sockets

a) Nodo móvil

Se precisa de un dispositivo móvil compatible con las API's MIDP 2.0 y CLDC 1.1 de Java ME (Véase Apéndice A). En caso de no disponer de un móvil que sea compatible con estas API's será imposible desarrollar esta alternativa de comunicación.

Para el desarrollo práctico no hemos utilizado ningún móvil, sino un simulador de móvil ejecutándose sobre un PC. Este simulador se encuentra dentro del *Java Wireless Toolkit 2.5 for CLDC*, que no es más que un conjunto de herramientas para el desarrollo de aplicaciones en Java ME.

La aplicación cliente del móvil (MIDlet) será capaz de enviar comandos predefinidos previamente y recibir la respuesta de dicho envía, mediante una única conexión tipo Socket. Para posteriormente mostrar por pantalla la respuesta recibida.

b) Nodo PC

Este nodo consiste en un PC, en el que se ejecutará una aplicación Java en modo servidor. Es decir, capaz de aceptar conexiones mediante Sockets, para ofrecer un servicio.

En nuestro caso, la aplicación espera conexiones para que le transmitan un comando, cuando dicho comando es recibido, la aplicación cambia su estado según dicho comando. Si se produjo un cambio de estado, devolverá OK a quien se lo transmitió, sino enviará NO_OK. Un comando particular, sería el comando "Estado", que devuelve el estado en el que se encuentra el servidor en dicho momento.

2.6.2. Protocolo

El diagrama de la Figura 20 muestra como se establece la comunicación entre la aplicación que se ejecuta en el nodo móvil (MIDlet) y la aplicación Java que se ejecuta en el nodo PC.

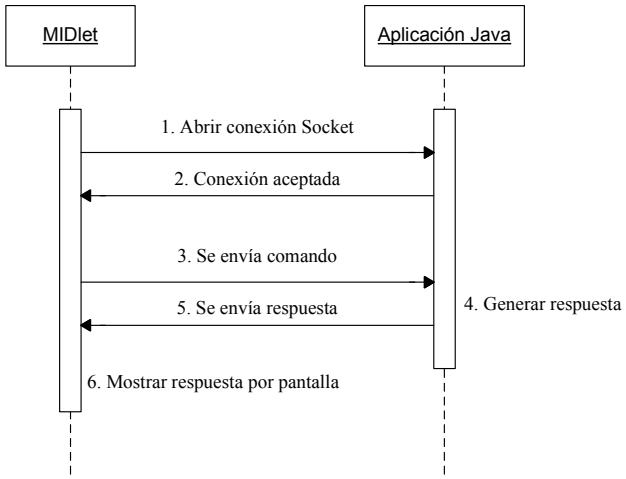


Figura 20. Diagrama de secuencia del protocolo para la comunicación directa a través de Socket

2.6.3. Paquetes y clases Java utilizadas

A continuación se mostrarán los paquetes y clases utilizadas en los componentes programados en lenguaje Java: MIDlet y Aplicación Java.

• MIDLET

En el diagrama de la Figura 21 se muestra los paquetes que se han utilizados para desarrollar el MIDlet, que recuérdese es la aplicación que se ejecuta en el móvil (Véase Figura 20):

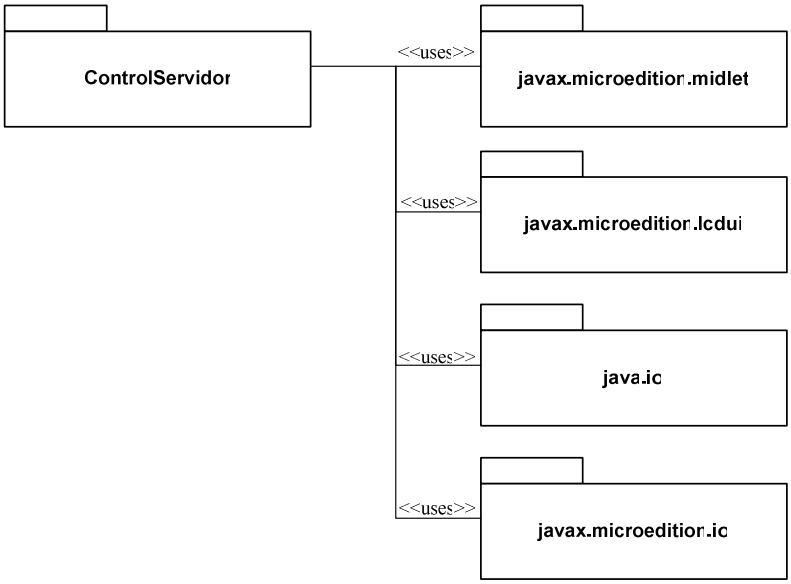


Figura 21. Java necesarios para desarrollar el MIDlet

En la Tabla 7 se muestran las clases, interfaces y excepciones utilizadas de cada uno de los paquetes para el desarrollo del MIDlet:

Paquete	***Clases***	***Excepciones***	***Interfaces***
javax.microedition.midlet	MIDlet		
javax.microedition.lcdui	Display Form TextBox StringItem TextField ChoiceGroup Command		CommandListener
javax.microedition.io	Connector		SocketConnection
Java.io	InputStream OutputStream	IOException	

Tabla 7. Clases, excepciones e interfaces necesarios para desarrollar el MIDlet

En la Tabla 8 se destacan algunas clases e interfaces que se han utilizado para desarrollar esta alternativa de comunicación, pero se recuerda que puede encontrar más información sobre más clases utilizadas en la Tabla 3:

Paquete javax.microedition.io	
Clase	**Descripción**
Connector	Sirve como factoría de nuevas conexiones, es decir, se utiliza para generar cualquier tipo de conexión desde el MIDlet.
Interface	**Descripción**
SocketConnection	Este interface extiende al interface StreamConnection y define una conexión Socket.

Tabla 8. Descripción de las clases, excepciones e interfaces más importantes en el desarrollo del MIDlet

• Aplicación Java

La aplicación java utilizada que se ejecuta en el nodo PC para esta alternativa de comunicación, es la misma que para la anterior, por lo que se insta a ver el apartado "Aplicación Java" del punto "2.5.3. Paquetes y clases Java utilizadas" para obtener más información.

2.6.4. Estructura y comportamiento

a. MIDlet

En el diagrama de la Figura 22 se puede observar la estructura de la aplicación que se ejecuta en el nodo móvil.

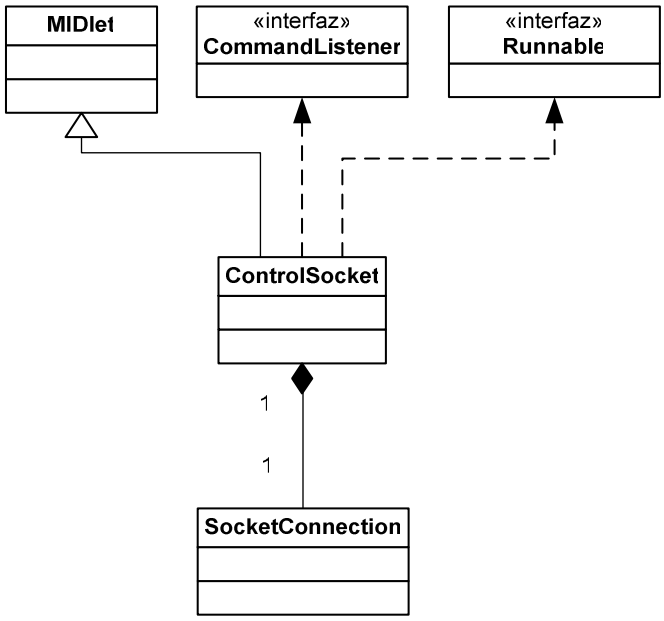


Figura 22. Estructura de clases del MIDlet

La clase `ControlSocket` extiende a la clase MIDlet, lo cual implica, que el nodo móvil es capaz de ejecutarla, es decir, se convierte en una aplicación MIDP. Además, como también se puede ver, `ControlSocket` implementa las interfaces `CommandListener` y `Runnable`. La primera para definir las acciones surgidas de eventos de comandos, y la segunda para poder crear nuevos hilos de ejecución, desde los cuales se establecen la comunicación. Se puede ver también que contiene a un objeto `SocketConnection`, que se utiliza para establecer una comunicación mediante Socket con la aplicación Java que se ejecuta en el nodo PC.

b. Aplicación Java

La aplicación java utilizada que se ejecuta en el nodo PC para esta alternativa de comunicación, es la misma que para la anterior, por lo que se insta a ver el apartado "b. Aplicación Java" del punto "2.5.4. Estructura y comportamiento" para obtener más información.

2.6.5. Comportamiento. Interacción entre objetos.

En el diagrama de la Figura 23 se puede observar el proceso de establecimiento, transmisión y recepción de información entre la aplicación del nodo móvil (`ControlSocket`) y la aplicación del nodo PC (`minimoServidor`). El diagrama muestra en detalle los pasos necesarios para establecer la comunicación por Socket entre ambas aplicaciones, que recuérdese, se ejecutan sobre plataformas diferentes (una en un móvil y la otra en un PC). Se destaca, principalmente, la necesidad de crear un nuevo hilo que ejecute otra instancia de la clase `ControlSocket`, para de esta forma, poder realizar la comunicación sin que se generen problemas a la hora de establecer la comunicación. Si no se hiciera esto, la aplicación bloquearía el móvil impidiendo cualquier acción sobre el mismo. Posteriormente, a través de la clase Connector se intenta abrir una conexión con la aplicación `minimoServidor` mediante el protocolo Socket. Una vez aceptada la conexión por `minimoServidor`, este genera un nuevo socket para ofrecer servicio a la conexión establecida con `ControlSocket` y se lo comunicará a este último. `ControlSocket` entonces crea un nuevo objeto de la clase `SocketConnection`, a través del cual se podrán

establecer los canales de comunicación para el intercambio de información (InputStream y OutputStream).

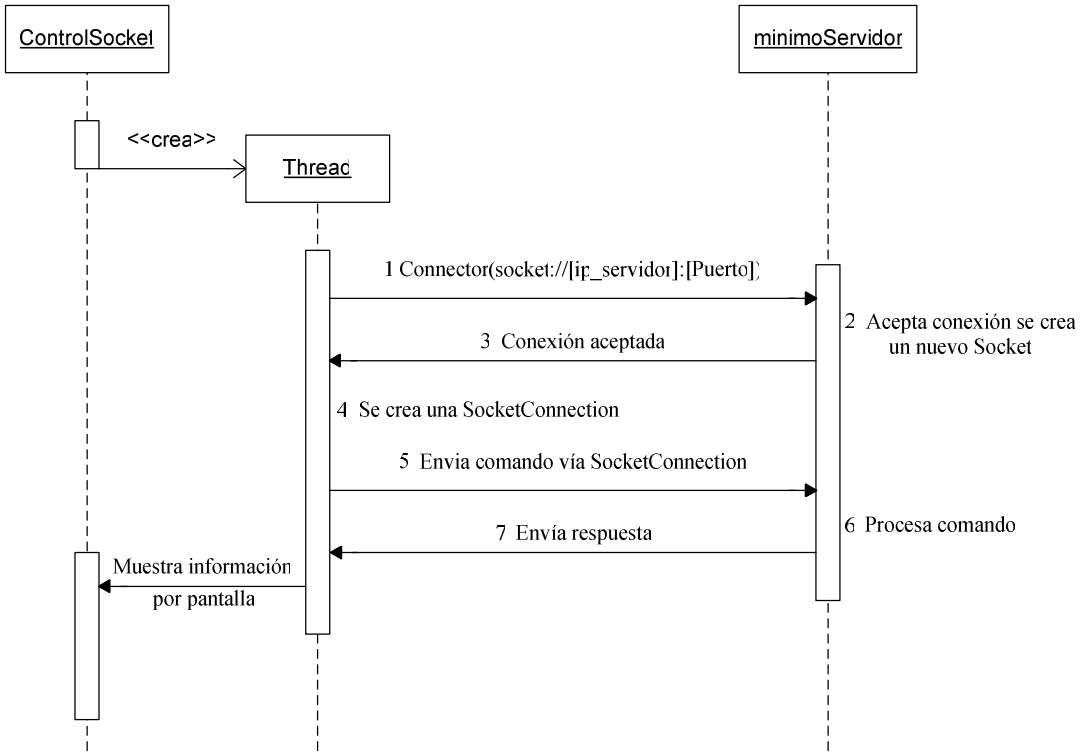


Figura 23. Diagrama de secuencia de la comunicación directa a través de Socket

2.6.6. La aplicación. Instalación, ejecución y manejo.

2.6.6.1. Nodo móvil

La aplicación que se ha creado para ser ejecutada en el nodo móvil se llama `ControlSocket`. (Véase el aparatado 2.5.6.1. Nodo móvil para completar información o Apéndice B si se desea más información de cómo desarrollar una aplicación MIDP para móviles).

En los siguientes apartados se explican los pasos necesarios para la instalación, la ejecución y el manejo de `ControlSocket` utilizando *Java Wireless Toolkit 2.5 for CLDC* (esta herramienta puede encontrarla de forma gratuita en http://java.sun.com).

a. Instalación

Para instalar la aplicación `ControlSocket` hay que seguir los mismos pasos que los llevados a cabo con la aplicación `CtrlBasicServ`, por eso se recomienda que se vea el punto 2.5.6.1. Nodo móvil, apartado a. Instalación.

b. Ejecución

Para ejecutar la aplicación `ControlSocket` habrá que seguir los mismos pasos que se describen para la alternativa anterior, pero en esta ocasión se debe elegir la aplicación `ControlSocket` en vez de `CtrlBasicServ` (Véase Figura 13).

Figura 24. Pantalla inicial del MIDlet

Ahora ya se está ejecutando nuestra aplicación en el nodo móvil (Véase Figura 24).

c. Manejo

Como ya se ha descrito anteriormente, en esta alternativa de comunicación aparecen dos nodos, el nodo móvil que ejecuta la aplicación MIDP `ControlSocket`, y el nodo PC que ejecuta la aplicación Java `minimoServidor` (Véase Figura 19 y Figura 23). Esta última es la que presta el servicio a la aplicación `ControlSocket`, por lo que si no se ejecuta anteriormente, la comunicación entre ambas aplicaciones será imposible.

Los pasos que hay que realizar para mandar un comando a la aplicación Java que hace de servidor son:

1º) Establecer la dirección IP del nodo PC donde se está ejecutando la aplicación Java `minimoServidor`.

2º) Establecer puerto por donde ofrece el servicio la aplicación `minimoServidor`.

3º) Seleccionar un comando.

4º) Pulsar el botón de Conectar del móvil. Se pedirá autorización para realizar la comunicación (Véase Figura 25)

5º) Se muestra por pantalla la información recibida de la aplicación `minimoServidor` (Véase Figura 26).

La aplicación dispone de un botón “Salir” en la pantalla inicial (Véase Figura 24) para finalizar la ejecución de la misma. Además, cuando se muestra la información recibida, tenemos la opción de volver a la pantalla inicial pulsando el botón “Volver”, pudiendo realizar una nueva comunicación si se desea.

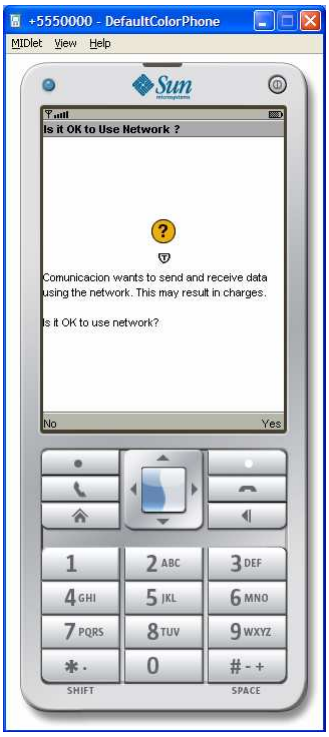


Figura 25. Autorización de conexión

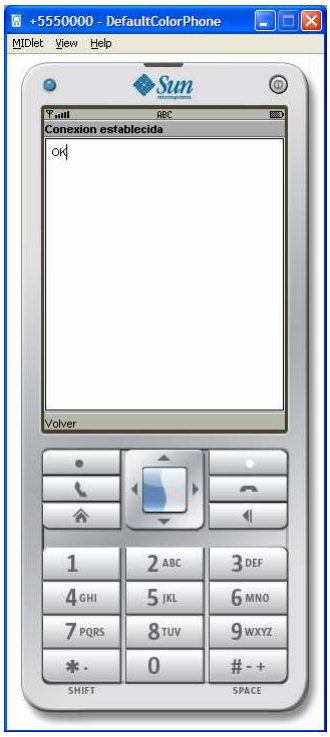


Figura 26. Respuesta recibida de la aplicación Java

2.6.6.2. Nodo PC

La aplicación que se ejecuta en el nodo PC para que esta alternativa de comunicación se pueda llevar a cabo, es la misma que para la alternativa anteriormente mostrada, por lo que para obtener más información sobre la aplicación véase el punto 2.5.6.3. Nodo PC de la alternativa Comunicación a través de Servlets.

Capítulo 3.

Visualización de imágenes en tiempo real

3.1. Introducción

En la actualidad, se están utilizando la visualización de imágenes en tiempo real para diversas aplicaciones, como puede ser chatear con un amigo, emitir imágenes de una zona determinada de tu ciudad por Internet a través de un navegador, para realizar video-vigilancia,… Todas ellas a través de alguna aplicación que se ejecuta en un PC. Con el auge de los teléfonos móviles de ultima generación, que incorporan grandes prestaciones para la visualización de imágenes a través de ellos (mucho de ellos ya están preparados para realizar videoconferencias), se puede empezar a pensar, que a estos teléfonos se les pueden añadir nuevas utilidades que van más allá de la mera conversación telefónica. Entre estas nuevas utilidades, se puede destacar, la visualización de imágenes en tiempo real del estado de nuestra casa o controlar la guardería donde están nuestros niños, desde cualquier lugar del mundo a través de nuestro móvil.

3.2. Objetivo

Este capítulo describe como se ha abordado el segundo de los objetivos que se persigue en este proyecto: la visualización de imágenes en tiempo real a través del teléfono móvil.

Se pretende mostrar por la pantalla del móvil las imágenes capturadas por una cámara Web conectada a un ordenador que dispone de conexión a Internet.

Se pretende conseguir este objetivo estableciendo la menor infraestructura necesaria. Es decir, se persigue que el conste sea el menor posible en cuanto a aplicaciones que intervienen y mantenimiento de las mismas.

Para conseguir estos objetivos se programarán las aplicaciones de cada uno de los nodos en el lenguaje Java.

3.3. Definición de alternativas

Para conseguir los objetivos marcados, se plantean dos alternativas. La primera de ellas, utiliza el protocolo HTTP para conseguir mostrar la imagen, y la segunda, mostraría las imágenes recibidas a través del protocolo UDP. Son dos alternativas muy diferentes conceptualmente, la primera solicita cada una de las imágenes que quiere mostrar, mientras que la segunda espera recibir las imágenes.

3.3.1. Visualización utilizando protocolo TCP.

En esta alternativa para la visualización de imágenes en tiempo real por el móvil, se utiliza el protocolo HTTP. Este protocolo, que pertenece al grupo del nivel de aplicación del modelo de referencia OSI, utiliza el protocolo TCP a nivel de transporte, que hace que la comunicación sea fiable extremo a extremo. Esto quiere decir, que las imágenes se irán mostrando secuencialmente sin posibilidad de que se pierdan ninguna de ellas.

Para desarrollar esta alternativa se necesita un servidor WEB y una aplicación encargada de capturar imágenes de una cámara WEB (Véase Figura 27). La aplicación captura la imagen de la cámara WEB y la almacena en el directorio raíz del servidor WEB. Este servidor espera a que la aplicación del móvil le solicite la imagen, para transmitírsela. Una vez recibe la imagen el móvil, la muestra por pantalla. Este proceso se repite cada cierto tiempo para dar la sensación de que es un vídeo.

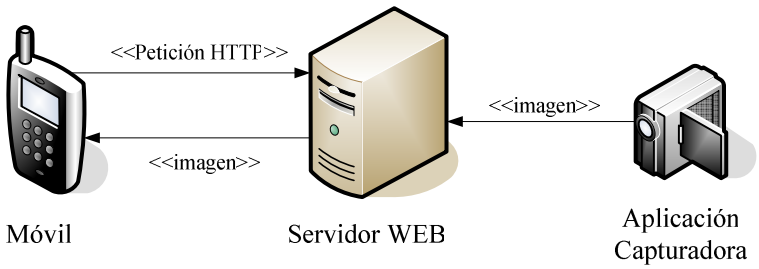


Figura 27. Nodos que intervienen en la visualización utilizando el protocolo TCP

3.3.2. Visualización utilizando protocolo UDP.

En la siguiente alternativa que se propone, se utiliza como protocolo UDP en el nivel de transporte, frente al TCP de la anterior alternativa. De esta manera, se ofrece una comunicación menos fiable extremo a extremo, esto quiere decir, que si una imagen no la recibe el móvil, se descarta, lo que significa que se ajusta más a la visualización de imágenes en tiempo real. En esta alternativa es preciso generar nuestro propio protocolo del nivel de aplicación para iniciar y finalizar la visualización de imágenes (lo que antes hacía el protocolo TCP).

Para el desarrollo de esta alternativa, precisamos de tres nodos en la comunicación: el móvil, una aplicación Java y una aplicación que captura imágenes de la cámara WEB (Véase Figura 28). La aplicación que se encarga de capturar imágenes de la cámara WEB, las almacena posteriormente en una carpeta especialmente creada para la Aplicación Java. Esta última aplicación, se encarga de gestionar el inicio y el fin de la visualización, y de la transmisión de las imágenes hacia el móvil. El móvil se encarga de solicitar el inicio y la finalización de la visualización, y de recibir las imágenes para mostrarlas por pantalla.

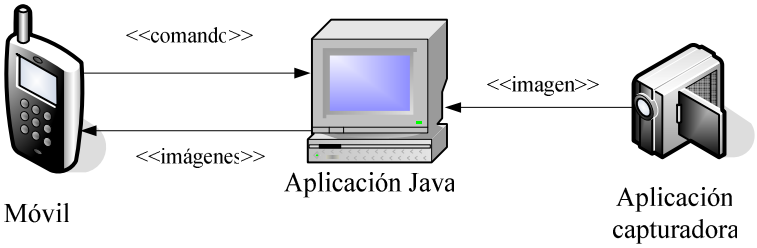


Figura 28. Nodos que intervienen en la visualización utilizando el protocolo UDP

3.4. Ventajas y desventajas de las alternativas

Las ventajas y desventajas que presenta la alternativa de visualización de imágenes en tiempo real usando el protocolo TCP son:

Ventajas	Desventajas
- Una infraestructura sencilla de implementar. - Permite gestionar el intervalo de tiempo entre imágenes. - El desarrollo de la aplicación para el móvil es más sencilla. - No es preciso generar ningún nuevo protocolo de nivel de aplicación para la visualización. Se utiliza HTTP.	- Una solicitud por cada imagen que se desea mostrar. Lo que implica la necesidad de mayor ancho de banda. - Se producen mayores retrasos. Cuando la comunicación falla, se produce retransmisión de imágenes. - Como consecuencia de lo anterior, se pierde el tiempo visualizando imágenes pasadas, cuando nos interesa ver las imágenes actuales.

Tabla 9. Ventajas y desventajas de la visualización utilizando el protocolo TCP

La alternativa de visualización de imágenes en tiempo real usando el protocolo UDP presenta las siguientes ventajas y desventajas:

Ventajas	Desventajas
- La perdida de imágenes en la comunicación no implica retrasos en la visualización. - No se pierde el tiempo visualizando imágenes pasadas, siempre se verán las imágenes actuales. - Optimiza mejor el ancho de banda. Solo hace falta la solicitud de inicio y fin de la visualización.	- Precisa generar un nuevo protocolo de aplicación al que deben ajustarse la aplicación del móvil y la aplicación Java. - De lo anterior, se requiere una mayor complejidad a la hora de implementar las aplicaciones. - No se puede ajustar los tiempos entre imágenes.

Tabla 10. Ventajas y desventajas de la visualización utilizando el protocolo UDP

3.5. Visualización utilizando el protocolo TCP

3.5.1. Infraestructura

Para desarrollar esta alternativa de visualización es necesaria la participación de tres elementos fundamentales, que son un móvil, un servidor WEB y una aplicación encargada de la captura de imágenes a través de una cámara WEB (Véase Figura 29).

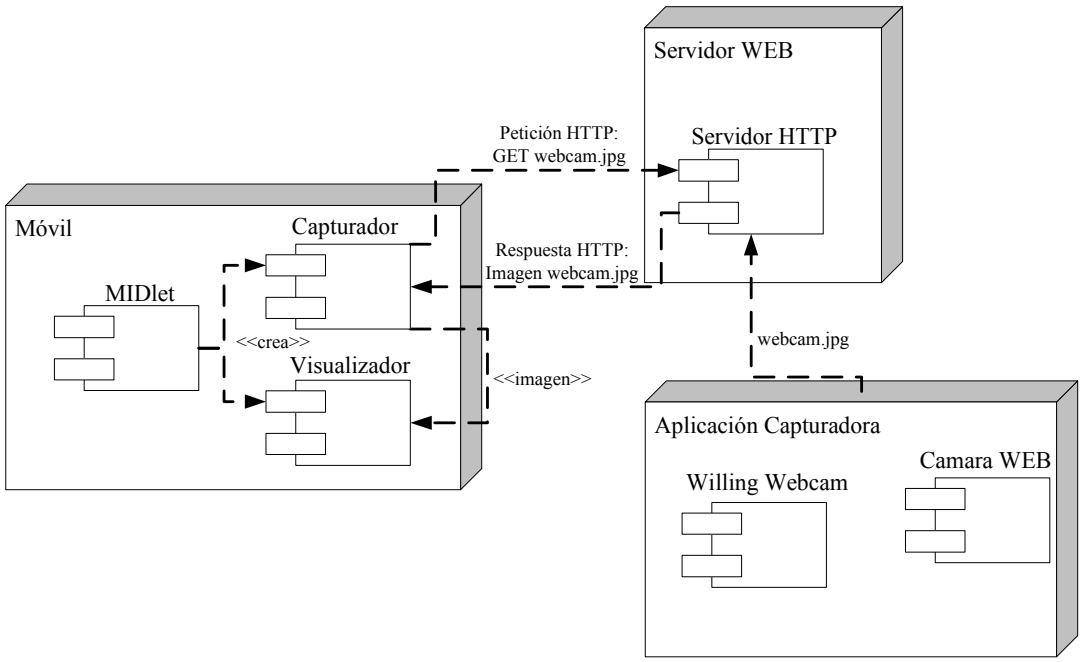


Figura 29. Nodos y componentes de la alternativa de visualización utilizando el protocolo TCP

a) Nodo móvil

Se precisa de un dispositivo móvil compatible con las API's MIDP 2.0 y CLDC 1.1 de Java ME (Véase Apéndice A). En caso de no disponer de un móvil que sea compatible con estas API's será imposible desarrollar esta alternativa de visualización.

Para el desarrollo práctico no hemos utilizado ningún móvil, sino un simulador de móvil ejecutándose sobre un PC. Este simulador se encuentra dentro del *Java Wireless Toolkit 2.5 for CLDC*, que no es más que un conjunto de herramientas para el desarrollo de aplicaciones en Java ME.

La aplicación es la encargada de solicitar la imagen al servidor WEB, utilizando el método GET del protocolo HTTP. Es decir, la imagen se solicita a través de una URL como por ejemplo:

http://[ip_Servidor_WEB]/webcam.jpg

Posteriormente recibirá la imagen que ha solicitado al servidor WEB y la mostrará por la pantalla del dispositivo. El tiempo entre cada solicitud será configurable.

b) Nodo servidor WEB

Es precisa la participación de un servidor WEB para gestionar las solicitudes de imágenes procedentes del móvil. Para ello podemos utilizar cualquier servidor WEB disponible en el mercado, como por ejemplo servidor HTTP *Apache*, que lo puede descargar de forma gratuita de www.apache.org.

El servidor recibirá del móvil una solicitud para que le mande la imagen que se encuentra en el directorio raíz. Después de recibirla le transmitirá la imagen solicitada.

c) Nodo Aplicación capturadora

Para realizar las capturas para la visualización en tiempo real, se utilizará una aplicación, que realizará dichas capturas y las almacenará en el directorio raíz del servidor

HTTP. Es decir, se requiere una aplicación capaz de capturar imágenes de una cámara WEB en un formato legible por el móvil y que lo haga cada cierto intervalo de tiempo. Una aplicación que cumple con estos requisitos es *Willing Webcam* que se puede adquirir en www.willingsoftware.com por unos 30 dólares aproximadamente.

Esta aplicación se encargará de capturar cada segundo una imagen en formato jpg, y la copiará en el directorio raíz del servidor HTTP.

3.5.2. Protocolo

Para establecer la visualización de imágenes usando el protocolo de transporte TCP, se sigue los pasos mostrados en el siguiente diagrama:

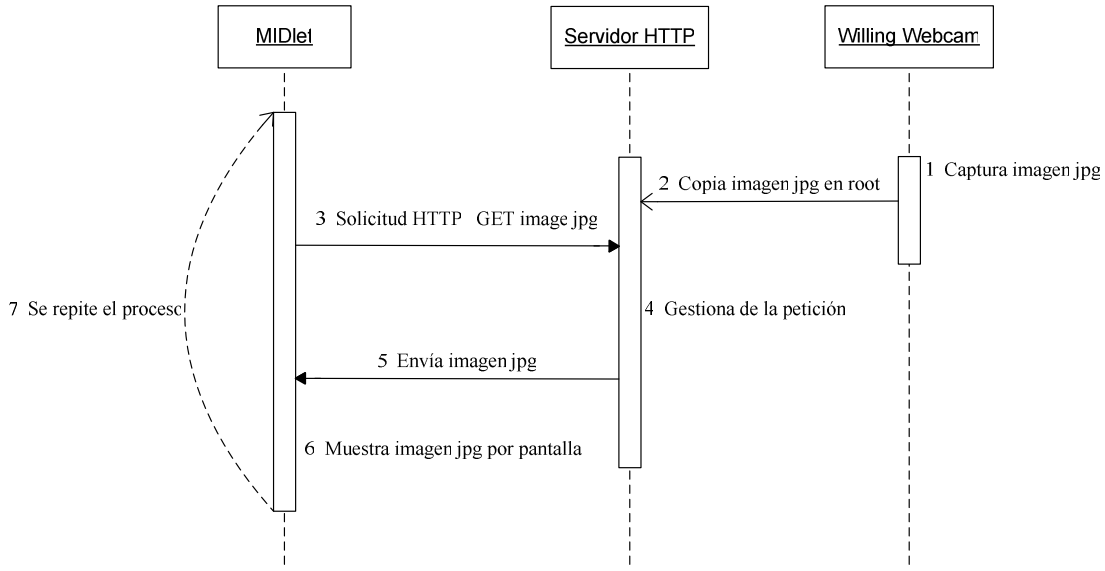


Figura 30. Diagrama de secuencia del protocolo para la visualización usando el protocolo TCP

Como se puede observar, se utiliza el protocolo HTTP para realizar la solicitud de la imagen al servidor HTTP. El móvil solicita la imagen en formato jpg, que anteriormente ha sido capturada por la aplicación *Willing Webcam* de la cámara WEB y copiada en el directorio de raíz del servidor HTTP. Una vez gestionada la solicitud por el servidor, le envía dicha imagen al móvil para que este la muestre por su pantalla. Este proceso se repetirá cada cierto intervalo de tiempo, configurable desde el propio móvil.

3.5.3. Paquetes y clases Java utilizadas

A continuación se mostrarán los paquetes y clases utilizadas en los componentes programados en lenguaje Java. En esta alternativa de visualización solo se va a requerir el desarrollo del MIDlet, que recuérdese es la aplicación que se ejecuta en el móvil (Véase Figura 29).

En el diagrama de la Figura 31 se muestra los paquetes que se han utilizados para desarrollar el MIDlet:

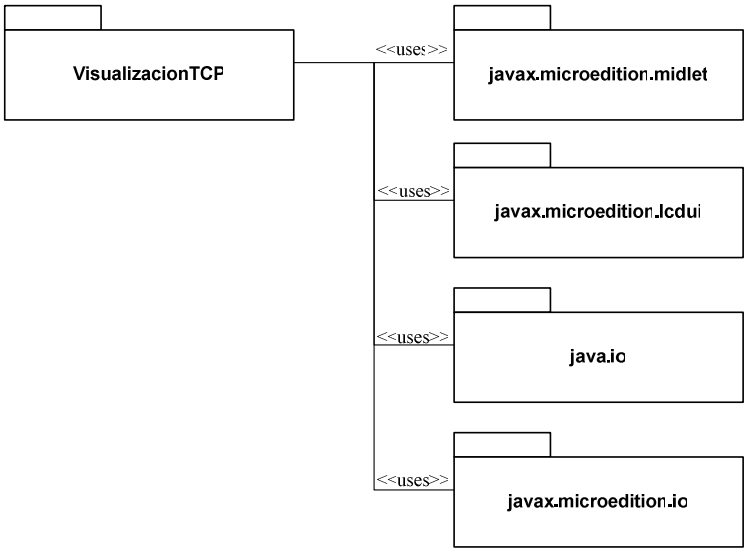


Figura 31. Paquetes Java necesarios para desarrollar el MIDlet

En la Tabla 11 se muestran las clases, interfaces y excepciones utilizadas de cada uno de los paquetes:

Paquete	***Clases***	***Excepciones***	***Interfaces***
javax.microedition.midlet	MIDlet		
javax.microedition.lcdui	Display Form StringItem TextField Command Image Canvas Graphics		CommandListener
javax.microedition.io	Connector		HttpConnection
Java.io	InputStream DataInputStream	IOException	

Tabla 11. Clases, excepciones e interfaces necesarios para desarrollar el MIDlet

En la Tabla 12 se destacan algunas clases e interfaces, pero se recuerda que puede encontrar más información sobre otras clases en la Tabla 3:

Paquete javax.microedition.lcdui	
Clase	**Descripción**
Canvas	Esta clase sirve como base para escribir aplicaciones a bajo nivel, que puede mostrar círculos, cuadrados, imágenes, etc…por la pantalla del móvil.
Graphics	Ofrece métodos para crear textos, rectángulos, círculos, imágenes y arcos.
Image	Esta clase sirve para tener la información de una imagen almacenada en memoria. Los objetos de esta clase pueden ser mutables o inmutables.
Paquete javax.microedition.io	
Interface	**Descripción**
HttpConnection	Este interface define los métodos necesarios para la creación de una conexión HTTP.

Tabla 12. Descripción de las clases, excepciones e interfaces más importantes en el desarrollo del MIDlet

3.5.4. Estructura y comportamiento

En el siguiente diagrama se puede observar la estructura diseñada para el desarrollo de esta alternativa de visualización.

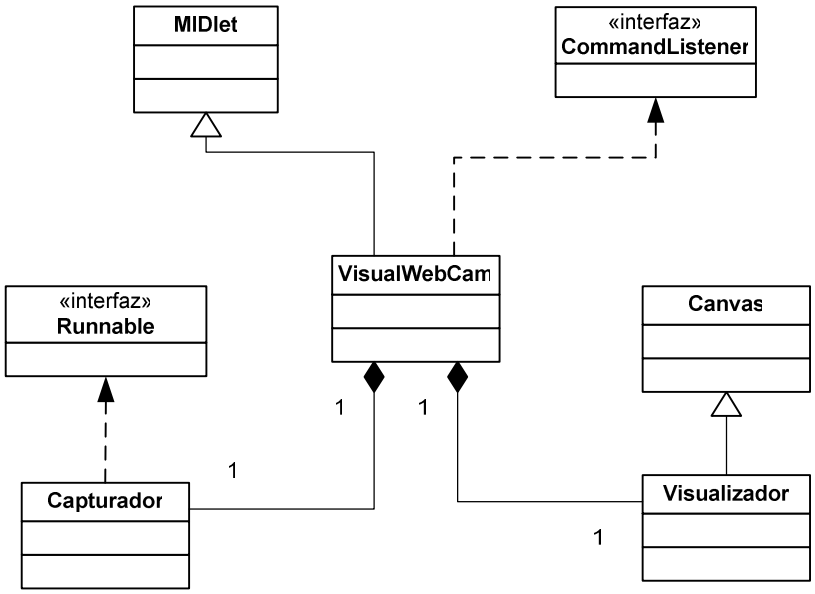


Figura 32. Estructura de clases del MIDlet

Como se puede observar la clase `VisualWebCam` extiende a la clase MIDlet, lo que significa que esta clase se podría ejecutar en un móvil. Esta clase implementa la interface `CommandListener` para poder definir las acciones surgidas de eventos de comandos. Además, esta clase contiene un `Capturador` y un `Visualizador`.

La función del `Capturador`, como su nombre indica, es la de capturar la imágenes. Es decir, se encarga de establecer la conexión con el servidor HTTP, solicitar la imagen, recibirla y pasársela al `Visualizador`. Como se puede observar también del anterior diagrama, implementa la interface `Runnable` para poder llevar a cabo sus funciones desde un nuevo hilo de ejecución y evitar que el móvil se quede bloqueado cuando se establece la comunicación.

La función del `Visualizador` es la de mostrar las imágenes que el `Capturador` le suministra. De este modo, cada vez que el `Capturador` le suministra una nueva imagen, el `Visualizador` la refresca en la pantalla. Para poder ofrecer esta imagen por la pantalla, es necesario que el `Visualizador` extienda a la clase `Canvas`. Ya que desde esta clase se pueden visualizar imágenes por la pantalla del móvil.

3.5.5. Comportamiento. Interacción entre objetos.

En el diagrama de la Figura 33 se puede observar como cada componente interviene en esta alternativa de visualización de imágenes. Como se puede observar, el MIDlet (aplicación que se ejecuta en el móvil) se llama `VisualWebCam` y se encarga de crear dos de los componentes que intervienen en esta alternativa de visualización: el `Visualizador` y el `Capturador`. El `Visualizador` simplemente espera a que el `Capturador` le suministre imágenes para posteriormente mostrarlas por la pantalla del dispositivo. El `Capturador` se encarga de solicitar y recibir las imágenes procedentes del servidor HTTP, para luego ofrecérselas al `Visualizador`.

La aplicación capturadora *Willing Webcam*, se encarga de capturar imágenes captadas por una cámara WEB, y copiarla en formato JPG en el directorio raíz del servidor HTTP.

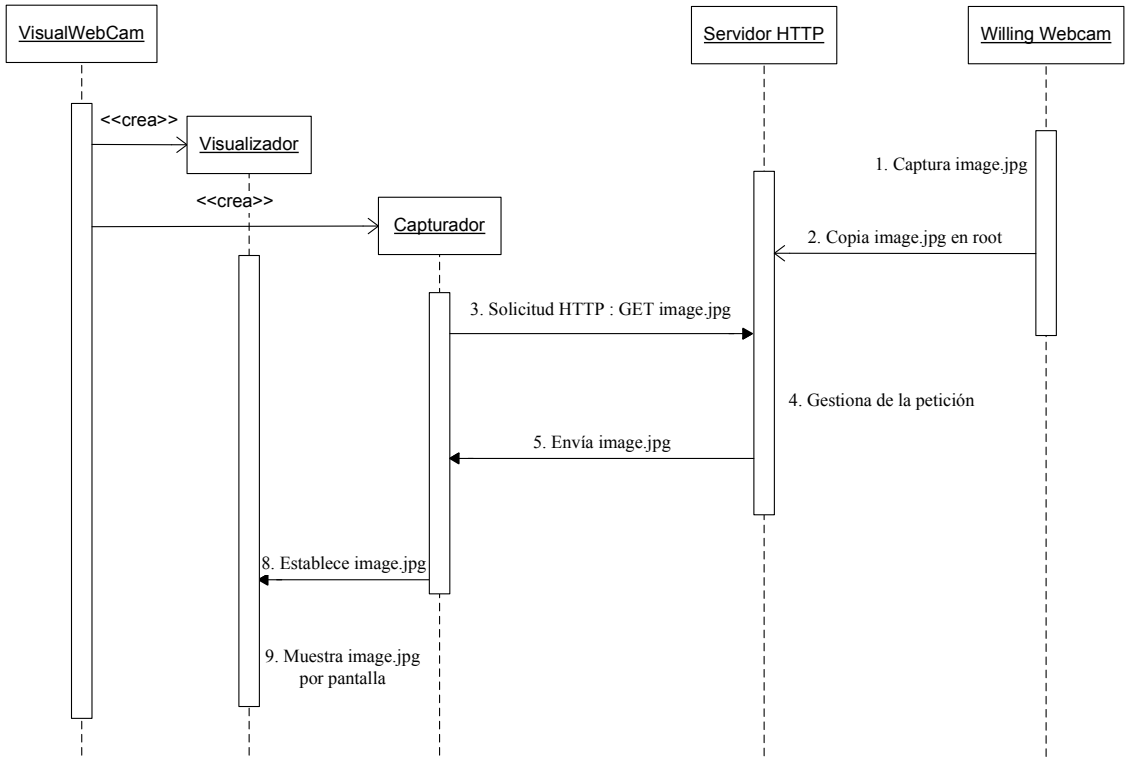


Figura 33. Diagrama de secuencia de la visualización utilizando el protocolo TCP

3.5.6. La aplicación. Instalación, ejecución y manejo.

3.5.6.1. Nodo móvil

La aplicación que se ha creado para ser ejecutada en el nodo móvil se llama `VisualWebCam` (Véase el aparatado 2.5.6.1. para completar información o Apéndice B si se desea más información de cómo desarrollar una aplicación MIDP para móviles).

En los siguientes apartados se explican los pasos necesarios para la instalación, la ejecución y el manejo de `VisualWebCam` utilizando *Java Wireless Toolkit 2.5 for CLDC* (esta herramienta puede encontrarla de forma gratuita en http://java.sun.com).

a. Instalación

Para instalar la aplicación `VisualWebCam` son necesarios los ficheros .jad y .jar. El fichero .jar contiene de forma comprimida, el conjunto de clases, librerías y recursos que utiliza la aplicación MIDP. La utilidad del fichero .jad radica en que es usado por el instalador del teléfono para reconocer / instalar / desinstalar la aplicación en cuestión. Estos dos archivos se pueden generar desde el IDE en el que se está desarrollando la aplicación.

La instalación de la aplicación en el móvil, varía dependiendo de la marca y el modelo del mismo, pero como se comentó anteriormente, se puede ejecutar la aplicación desde un simulador que se encuentra en *Java Wireless Toolkit for CLDC 2.5*.

b. Ejecución

Como se ha comentado en el apartado anterior (Instalación), la ejecución se llevará a cabo desde un simulador que se encuentra en el conjunto de herramientas *Java Wireless Toolkit for CDLC 2.5*.

Para ejecutar la aplicación en el simulador habrá que seguir los siguientes pasos:

1º) Ejecutar el simulador. Menú de Inicio→ Programas→Sun Java™ Wireless Toolkit 2.5→Run MIDP Aplication. Aparecerá la siguiente pantalla:

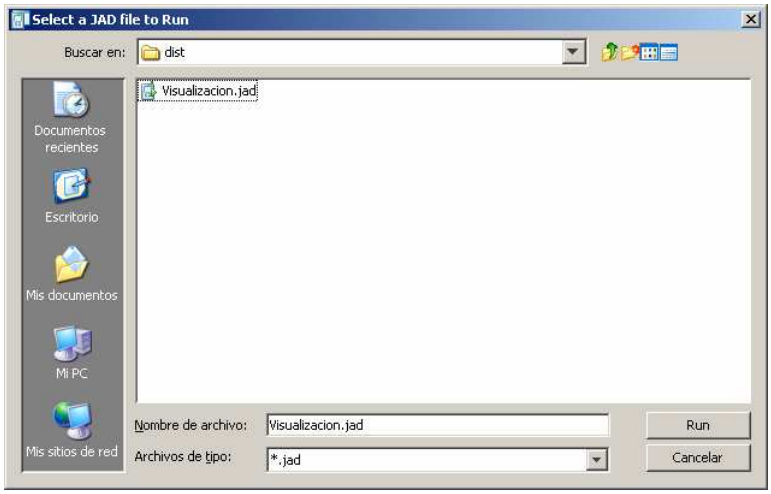


Figura 34. Explorador con la ruta y el jad del MIDlet

2º) Buscar el fichero .jad de la aplicación para el móvil que se encuentra en la carpeta "dist" del proyecto y ejecutarlo. Aparecerá un móvil por la pantalla del PC que hará las funciones de nodo móvil (Ver Figura 35).

3º) Seleccionar la aplicación `VisualWebCam`. Recuérdese que así es como se llama la aplicación MIDP que hemos desarrollado para realizar la visualización de imágenes.

Ahora ya se está ejecutando nuestra aplicación en el nodo móvil (Véase Figura 36).

Figura 35. Pantalla para seleccionar el MIDlet

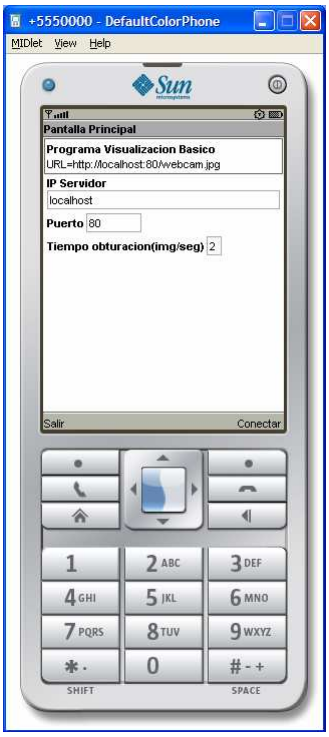


Figura 36. Pantalla inicial del MIDlet

c. Manejo

Como ya se ha comentado anteriormente, en esta alternativa de visualización van a ser necesarios 3 nodos: el nodo móvil, el servidor WEB y la aplicación capturadora (Véase Figura 3 y Figura 11). La aplicación del móvil esta preparada para conectarse con cualquier servidor HTTP disponible en Internet, para solicitarle el archivo "webcam.jpg" mediante el método GET del protocolo HTTP. Esto significa que aunque nosotros utilicemos nuestro propio servidor, no haría falta en realidad, ya que podría capturar la imagen y ponerla en cualquier servidor WEB disponible en Internet.

Los pasos que hay que realizar para visualizar las imágenes en el móvil son:

1º) Establecer la dirección IP del servidor HTTP donde se encuentra el archivo "webcam.jpg".

2º) Establecer puerto por donde ofrece el servicio el servidor HTTP.

3º) Establecer el intervalo de tiempo entre peticiones de imágenes.

4º) Una vez estén los parámetros de la visualización definidos pulsar el botón de Conectar del móvil. Aparecerá la pantalla de visualización (Véase Figura 37)

5º) Para comenzar la visualización pulsar Capturar, se pedirá autorización para realizar la comunicación (Véase Figura 38)

6º) Comienza la visualización de imágenes (Véase Figura 39).

La aplicación dispone de un botón "Salir" en la pantalla inicial (Véase Figura 36) para finalizar la ejecución de la misma. Además, cuando nos encontramos visualizando imágenes (Véase Figura39), podemos parar dicha visualización en el momento en el que queramos pulsando el botón "Parar". También tenemos la opción de volver a la pantalla inicial pulsando el

botón "Volver", pudiendo realizar una nueva configuración para la visualización de las imágenes.

Figura 37. Vista del Visualizador

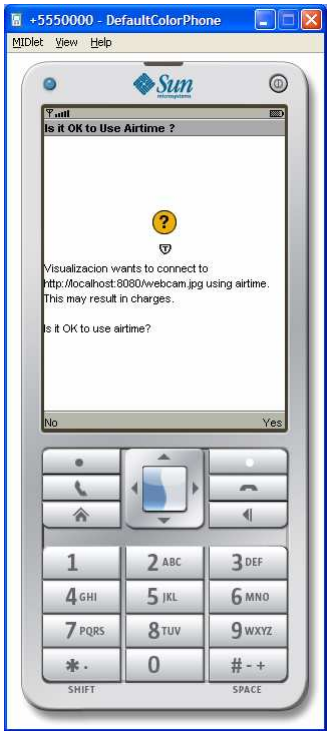


Figura 38. Autorización de conexión

Figura 39. Visualización de imágenes.

3.5.6.2. Nodo servidor WEB

Para desarrollar esta alternativa de comunicación no se precisa la utilización de un servidor HTTP propio, hubiera bastado con utilizar cualquier servidor WEB disponible en Internet. Pero por tener un mayor control sobre los posibles fallos que puedan aparecer en la visualización cuando se esté desarrollando la aplicación del móvil, se hace necesario tener nuestro propio servidor para depurar mejor los errores.

Se ha elegido el *servidor HTTP Apache* para Windows que puede encontrarlo y descargárselo de forma gratuita desde www.apache.org.

a. Instalación

En los siguientes manuales disponibles en Internet se pueden encontrar información detallada sobre la instalación y configuración de *servidor HTTP Apache*:

→ http://quark.fe.up.pt/ApachES/manual-es/windows.html (Recomendable)

→ http://wiki.gleducar.org.ar/wiki/Manual_Instalar_Apache (Más sencillo)

b. Ejecución

Este servidor en el proceso de instalación, da la opción de ejecutar cuando Windows se inicia. Si se marca esta opción el servidor se ejecutará automáticamente cada vez que Windows arranca. En caso de no haberla marcado, tendremos que ejecutar:

Menú Inicio → Programas → Apache HTTP Server → Control Apache Server → Start

En este momento el servidor debe estar ofreciendo servicio en el ordenador en el que fue instalado.

c. Manejo

Para la publicación de documentos tendremos que ir a la carpeta "htdocs" en donde se encuentra instalado el servidor, y copiar aquí dichos documentos. Es decir, la carpeta "htdocs" es la raíz del servidor y donde se debe copiar la imagen capturada de la cámara WEB.

3.5.6.3. Nodo Aplicación capturadora

La aplicación capturadora que se ha utilizado para el desarrollo de esta alternativa de visualización es *Willing Webcam*. Esta aplicación ofrece diversos servicios para la visualización de imágenes a través de Internet, como por ejemplo, enviar imágenes capturadas de la cámara WEB por FTP, publicar dichas imágenes a través de un servidor WEB que esta aplicación posee, etc. Pero el servicio que esta aplicación ofrece y que a nosotros nos interesa más, es el de capturar imágenes en formato JPG y poder copiarlas en el directorio raíz del servidor WEB que vamos a utilizar.

Este programa se puede adquirir en www.willingsoftware.com por unos 30 dólares aproximadamente.

En caso de dudas o problemas sobre la instalación o configuración de este programa puede acceder a la siguiente dirección WEB http://www.willingsoftware.com/help/howto.shtml.

Configuración

Los pasos necesarios para configurar la aplicación *Willing Webcam* para adaptarla a las necesidades que esta alternativa de visualización requiere son los siguientes:

1º) Tener conectada y funcionando la cámara WEB.

2º) Ejecutar la aplicación *Willing Webcam*. El aspecto del programa se puede observar en la Figura 40.

3º) En la tabla de opciones desplegar la opción "Imagen". En la figura 40 es la opción que está desplegada.

4º) Configurar la opción "Carpeta de destino" con la capeta "htdocs" de nuestro servidor HTTP Apache. Recuérdese que esta carpeta es el directorio raíz de nuestro servidor.

5º) En la opción "Fichero" ponemos el nombre "webcam.jpg". Así es como hemos definido la peticiones en la aplicación `VisualWebCam`, que recuérdese es la aplicación que se ejecuta en el móvil.

6º) En la opción "Ciclo de repetición" se establecerá un segundo. Este va a ser el intervalo de tiempo que la aplicación capturadora va a utilizar para refrescar las imágenes que se copian en el directorio raíz del servidor WEB.

7) Pulsar sobre la pestaña de la opción "Activar". De este modo la aplicación empezará a realizar las capturas de imágenes que esta alternativa precisa.

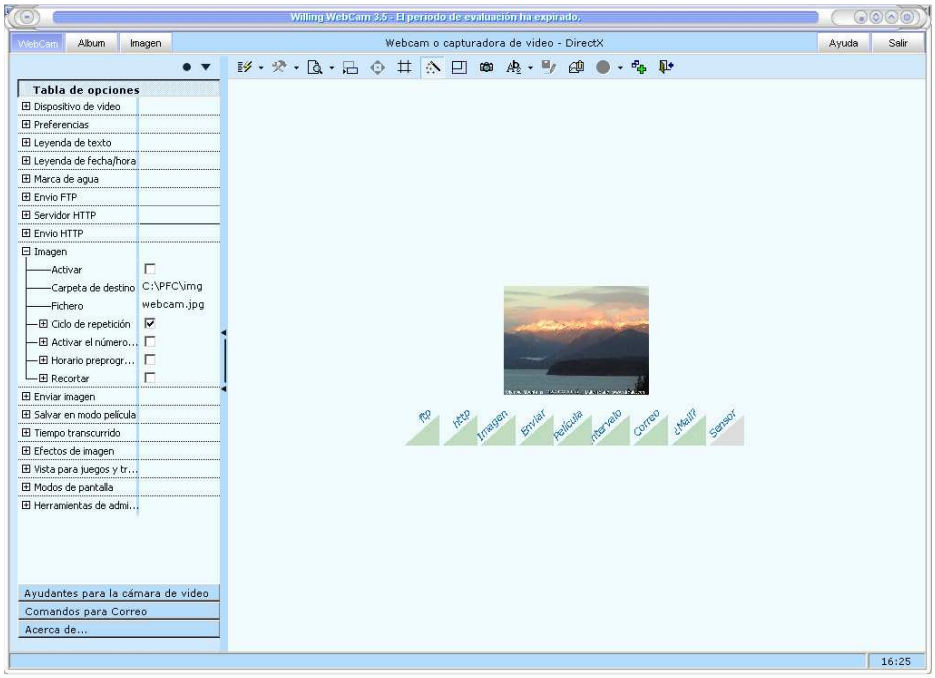


Figura 40. Pantalla principal de la aplicación capturadora Willing Webcam

3.6. Visualización utilizando el protocolo UDP

3.6.1. Infraestructura

En el siguiente diagrama se pueden observar los nodos y sus componentes que intervienen en esta alternativa de visualización. Se precisa un móvil, una aplicación java que será la que ofrezca las imágenes y una aplicación capturadora que se encargara de capturar las imágenes de la cámara WEB.

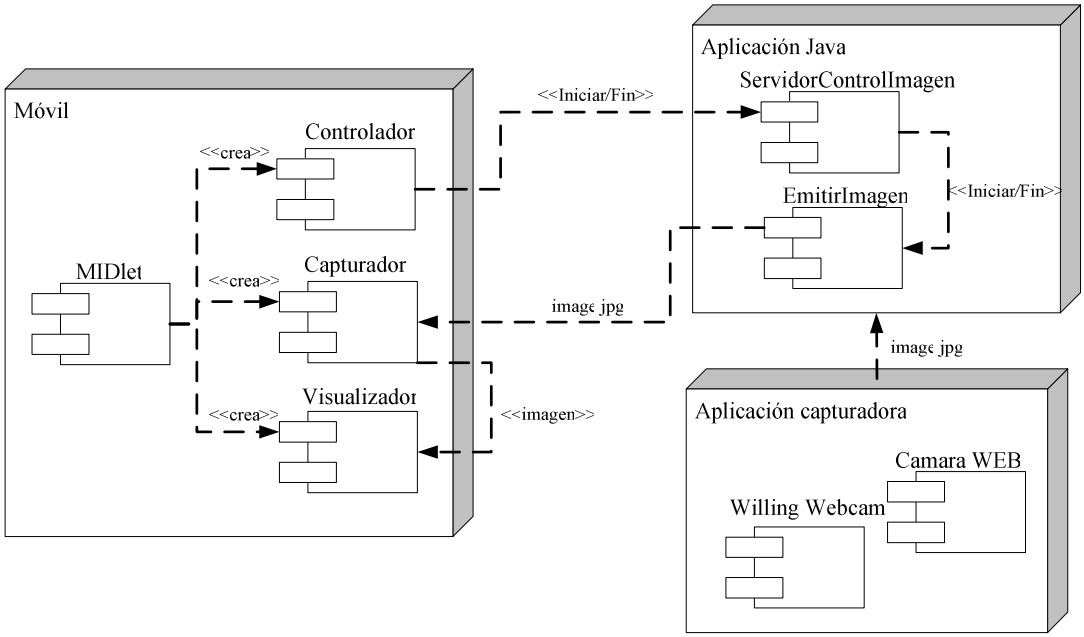


Figura 41. Nodos y componentes de la alternativa de visualización utilizando el protocolo UDP

a) Nodo móvil

Se precisa de un dispositivo móvil compatible con las API's MIDP 2.0 y CLDC 1.1 de Java ME (Véase Apéndice A). En caso de no disponer de un móvil que sea compatible con estas API's será imposible desarrollar esta alternativa de visualización.

Para el desarrollo práctico no hemos utilizado ningún móvil, sino un simulador de móvil ejecutándose sobre un PC. Este simulador se encuentra dentro del *Java Wireless Toolkit 2.5 for CLDC*, que no es más que un conjunto de herramientas para el desarrollo de aplicaciones en Java ME.

La aplicación que se ejecuta en el móvil se encarga de solicitar el inicio de la emisión de imágenes a la Aplicación Java. Una vez recibida esta solicitud, la aplicación Java comienza la mandarle imágenes al móvil, que cuando las recibe, las muestra por la pantalla.

b) Nodo Aplicación Java

Como el nombre indica, consiste en una aplicación desarrollada en lenguaje Java que prestará los servicios necesarios para realizar la visualización en el nodo móvil.

La aplicación Java se encargará de recibir las solicitudes para iniciar o finalizar la visualización, y de transmitir dichas imágenes a la aplicación que se ejecuta en el móvil.

Por lo tanto las clases o componentes que componen esta aplicación serán implementadas para que realicen estas funciones ya descritas.

c) Nodo Aplicación capturadora

Para realizar las capturas para la visualización en tiempo real, se utilizará una aplicación, que realizará dichas capturas y las almacenará en el directorio habilitado dentro de la aplicación Java. Es decir, se requiere una aplicación capaz de capturar imágenes de una cámara WEB en un formato legible por el móvil y que lo haga cada cierto intervalo de tiempo. Una aplicación que cumple con estos requisitos es *Willing Webcam* que se puede adquirir en www.willingsoftware.com por unos 30 dólares aproximadamente.

Esta aplicación se encargará de capturar cada segundo una imagen en formato JPG, y la copiará en el directorio creado para tal efecto en la aplicación Java.

3.6.2. Protocolo

Para el desarrollo de esta alternativa se tiene que diseñar un protocolo para iniciar la visualización (Véase Figura 42) y otro para poder finalizarla (Véase Figura 43).

Protocolo para iniciar la visualización

En el diagrama de la Figura 42 se puede observar que es la aplicación que se ejecuta en el móvil la que realiza la petición para iniciar el proceso de visualización. Esta petición se gestiona en la aplicación Java, que se pone en estado de emisión de imágenes. La aplicación recibe en su directorio "img" la imagen en formato JPG capturada por la aplicación capturadora. Como se encuentra en estado de emisión de imágenes, está imagen recibida se la envía al MIDlet, que se encargará de mostrarla por la pantalla del móvil.

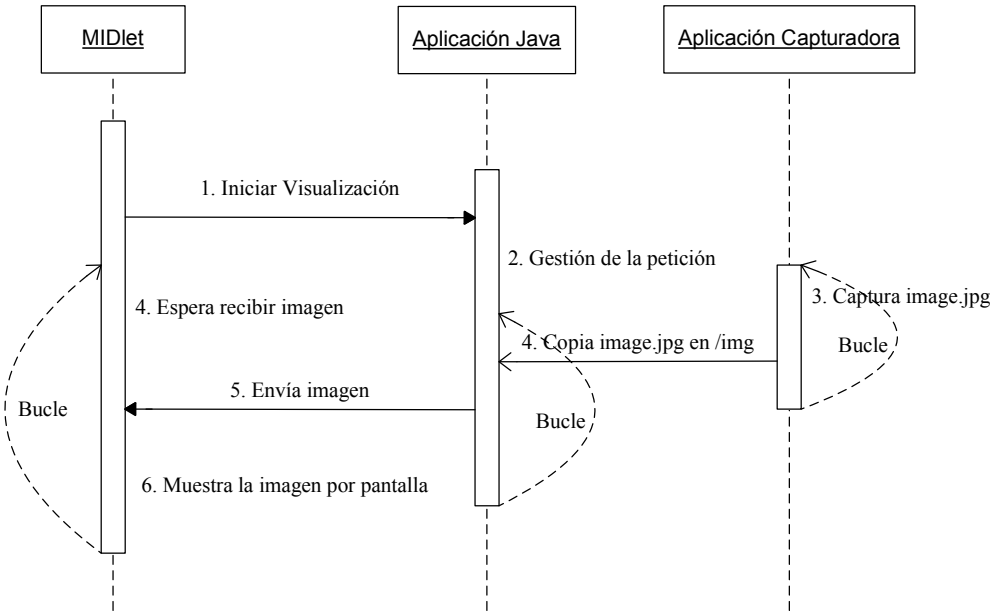


Figura 42. Diagrama de secuencia del protocolo para iniciar la visualización usando el protocolo UDP

Protocolo para finalizar la visualización

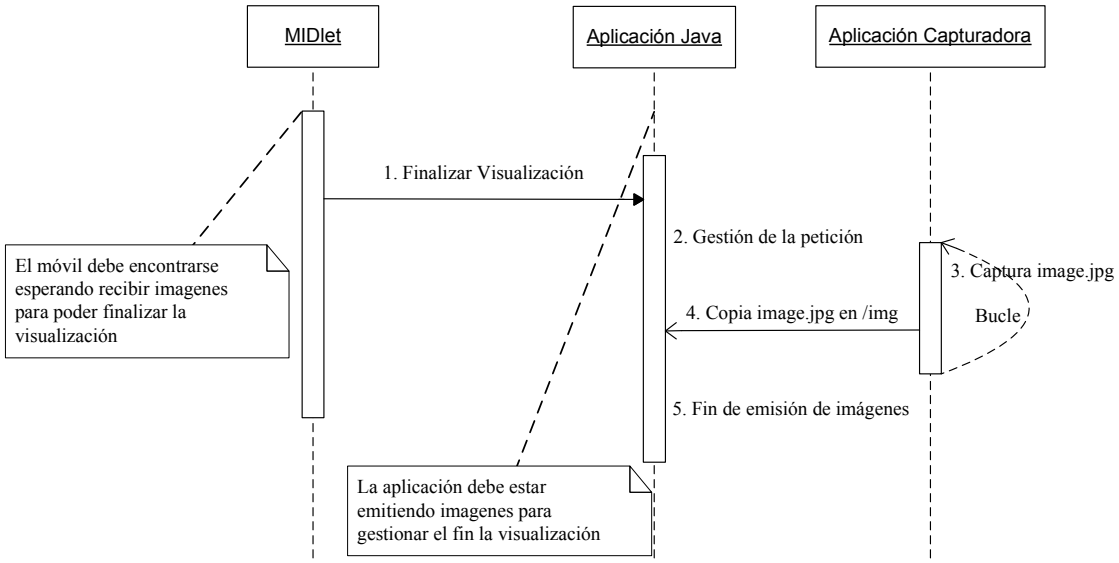


Figura 43. Diagrama de secuencia del protocolo para finalizar la visualización usando el protocolo UDP

Para poder finalizar el proceso de visualización el MIDlet y la aplicación Java deben de encontrarse esperando y emitiendo imágenes, respectivamente. Si se da esta condición, el MIDlet puede mandar la petición de finalización de la visualización. Esta petición la gestiona la aplicación Java que para la emisión. Obsérvese que aunque no se esta emitiendo imágenes, la aplicación capturadora seguirá capturando imágenes y copiándolas en el directorio "img" de la aplicación Java.

3.6.3. Paquetes y clases Java utilizadas

A continuación se mostrarán los paquetes y clases utilizadas en los componentes programados en lenguaje Java: MIDlet y Aplicación Java.

• MIDLET

En el diagrama de la Figura 44 se muestra los paquetes que se han utilizados para desarrollar el MIDlet, que recuérdese es la aplicación que se ejecuta en el móvil (Véase Figura 41):

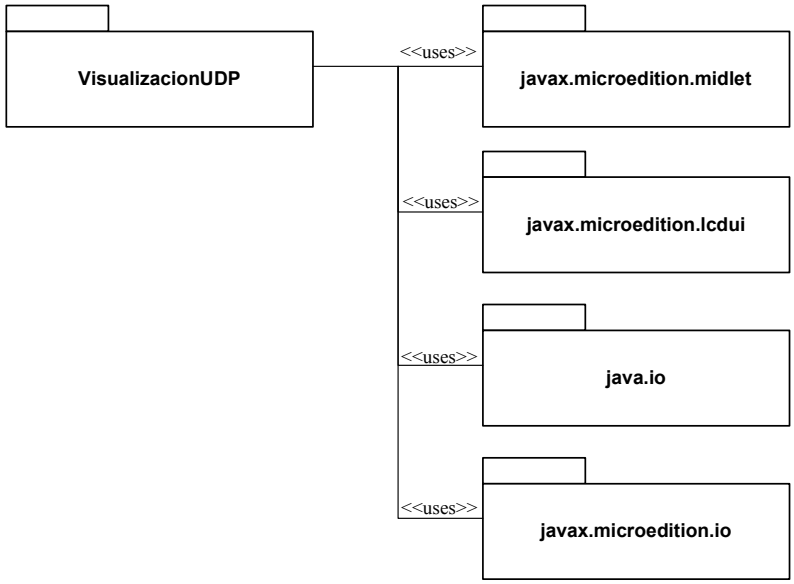


Figura 44. Paquetes Java necesarios para desarrollar el MIDlet

En la Tabla 13 se muestran las clases, interfaces y excepciones utilizadas de cada uno de los paquetes para el desarrollo del MIDlet:

Paquete	***Clases***	***Excepciones***	***Interfaces***
javax.microedition.midlet	MIDlet		
javax.microedition.lcdui	Display Form StringItem TextField Command Image Canvas Graphics		CommandListener
javax.microedition.io	Connector		SocketConnection UDPDatagramConnection Datagram
java.io	OutputStream	IOException	

Tabla 13. Clases, excepciones e interfaces necesarios para desarrollar el MIDlet

En la Tabla 14 se destacan algunas clases e interfaces, pero se recuerda que puede encontrar más información sobre el resto de clases en las Tablas 3, 8 y 12:

Paquete javax.microedition.io	
Interface	**Descripción**
SocketConnection	Este interface extiende al interface StreamConnection y define una conexión Socket.
UDPDatagramConnection	Este interface define una conexión por datagramas en la que son conocidas tanto la dirección local como la remota. El protocolo utilizado no está orientado a conexión, esto quiere decir que no se garantiza la entrega de los datagramas, ni que la llegada de estos sea en el mismo orden en que se transmitieron.
Datagram	Esta clase define un interface para la creación de datagramas. Sirve como contenedor de la información que se pretende recibir o enviar por una conexión de tipo UDPDatagramConnection o DatagramConnection.

Tabla 14. Descripción de las clases, excepciones e interfaces más importantes en el desarrollo del MIDlet

• Aplicación Java

Para la realización de la aplicación Java se han utilizado los paquetes que se muestran en el diagrama de la Figura 45.

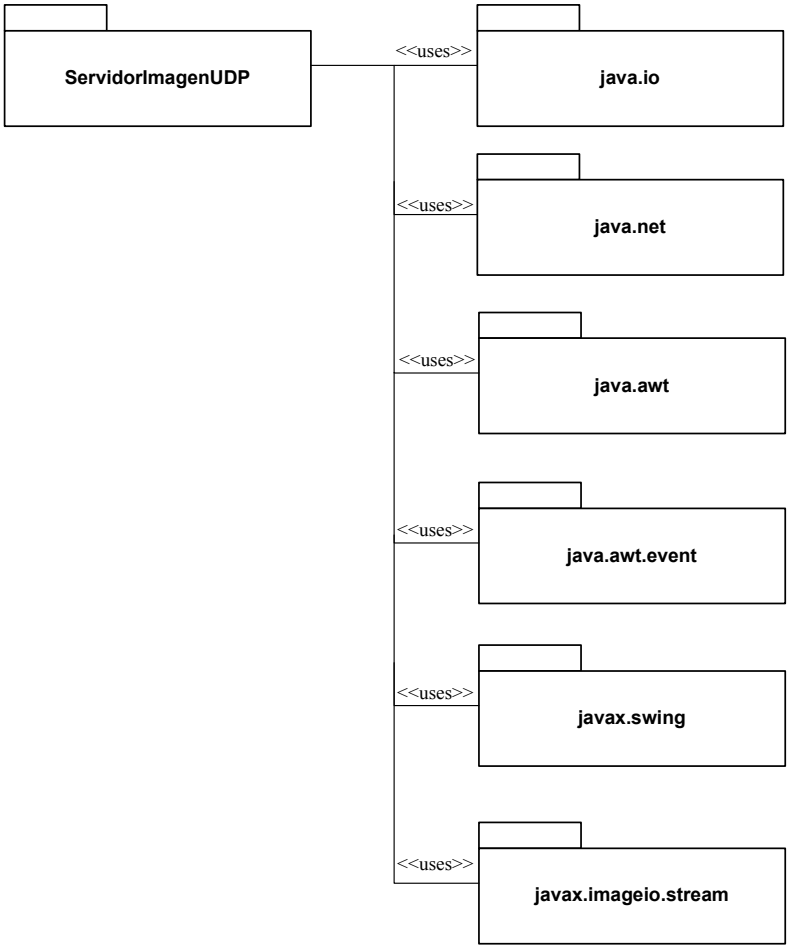


Figura 45. Paquetes Java necesarios para desarrollar la aplicación Java

En la Tabla 15 se muestran las clases, interfaces y excepciones utilizadas de cada uno de los paquetes anteriormente mostrados:

Paquete	***Clases***	***Excepciones***	***Interfaces***
java.io	InputStream File	IOException	
java.net	ServerSocket Socket InetAddress DatagramSocket DatagramPacket	SocketException	
java.awt	BorderLayout		
java.awt.event	ActionEvent		ActionListener
javax.swing	JFrame JScrollPane SwingUtilities JTextArea JButton		
javax.imageio.stream	FileImageInputStream		

Tabla 15. Clases, excepciones e interfaces necesarios para desarrollar la aplicación Java

De las clases, excepciones e interfaces mostrados en la tabla anterior se destacan:

Paquete java.net	
Clase	**Descripción**
DatagramSocket	Sirve para transmitir datagramas. La transmisión de la información no es orientada a conexión, es decir, los datagramas son enviados de una máquina hacia el remoto y no se garantiza la llegada, ni que lleguen con el mismo orden con el que fueron transmitidos.
DatagramPacket	Esta clase sirve para construir datagramas que pueden ser mandados usando un objeto DatagramSocket.
Paquete java.swing	
Clase	**Descripción**
SwingUtilities	Es un conjunto de métodos de utilidad que ofrece la clase swing. Se utilizará en concreto el método “InvokeLater” para mostrar mensajes sobre un entorno gráfico Java.
Paquete javax.imageio.stream	
Clase	**Descripción**
FileImageInputStream	Sirve como contenedor de la información de un archivo de imágenes dentro de una aplicación Java.

Tabla 16. Descripción de las clases, excepciones e interfaces más importantes en el desarrollo de la aplicación Java

3.6.4. Estructura y comportamiento

a. MIDlet

En el diagrama de la Figura 46 se puede observar la estructura de la aplicación que se ejecuta en el nodo móvil.

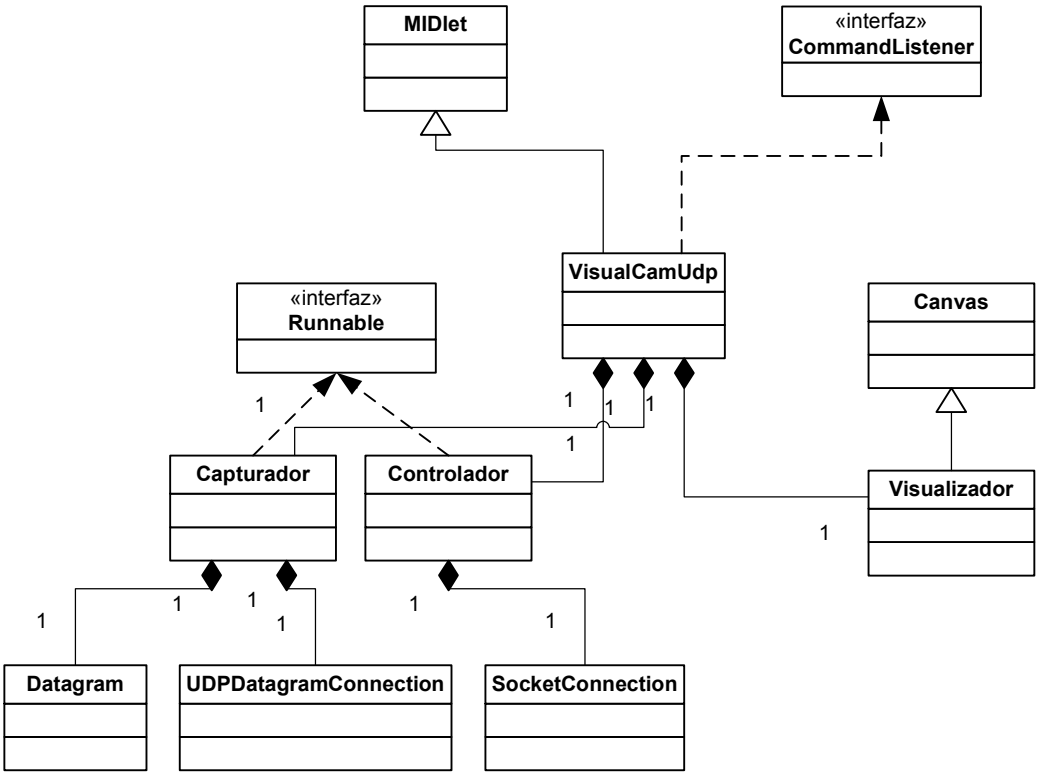


Figura 46. Estructura de clases del MIDlet

Obsérvese que la clase `VisualCamUdp` extiende a la clase MIDlet, que como ya se ha comentado anteriormente significa que dicha clase podrá ser ejecutada como una aplicación Java ME en cualquier dispositivo móvil. La clase `VisualCamUdp` implementa el interface `CommandListener` para definir las acciones surgidas de eventos de comandos.

En la figura se destacan la composición de los objetos de clase `VisualCamUdp` con objetos de las clases `Capturador`, `Controlador` y `Visualizador`. Cada una de estos objetos realiza un papel determinado en el desarrollo de la visualización en el móvil.

El `Controlador` se encarga de transmitir las órdenes de inicialización o finalización de la visualización. A su vez, también realiza las acciones necesarias para poner en marcha o parar al objeto de la clase `Capturador`. El proceso de control que realiza el `Controlador` se lleva a cabo desde un nuevo hilo de ejecución para evitar que la aplicación pueda quedar bloqueada, razón por la cual esta clase implementa el interface `Runnable`.

La tarea del `Capturador` es recibir los datagramas con imágenes que le manda la aplicación Java, para entregárselos al `Visualizador`. El estado de esta clase como ya se ha comentado antes, es dirigido por el `Controlador`, es decir, sólo cuando el `Controlador` lo manda, el `Capturador` recibe o deja de recibir imágenes. Para que la aplicación no se quede bloqueada cuando se está recibiendo las imágenes, este proceso se hace desde un nuevo hilo de ejecución, es por esta razón por lo que el `Capturador` implementa la clase `Runnable`. Se desea hacer notar que las imágenes viajan empaquetadas dentro de datagramas a través de una comunicación UDP.

El `Visualizador` se encarga de mostrar por la pantalla del móvil las imágenes que el `Capturador` recibe de la aplicación Java. Para poder mostrar imágenes por la pantalla del móvil es necesario hacer uso de la clase `Canvas`. Como la clase `Visualizador` hereda de la

clase `Canvas`, esto significa que la primera puede mostrar las imágenes por la pantalla del móvil usando los métodos apropiados.

b. Aplicación Java

En el diagrama de la Figura 47 se destacan los aspectos más relevantes de la aplicación Java:

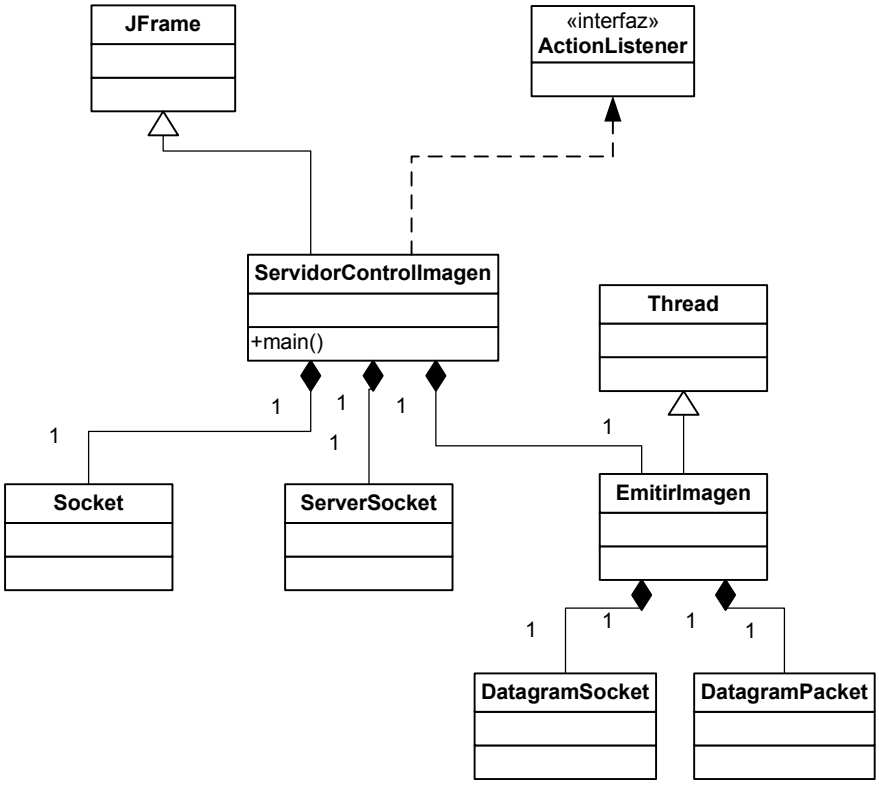


Figura 47. Estructura de clases de la aplicación Java

La clase principal como se puede observar es `ServidorControlImagen`. Contiene el método *main* que ejecuta la aplicación Java. Como se puede observar, esta clase extiende de la clase `JFrame` e implementa el interface `ActionListener`, lo que la convierte en una aplicación gráfica. Esta interface gráfica proporciona una vía para mostrar lo que está sucediendo, es decir, sirve para mostrar por la ventana de la aplicación la información que se está recibiendo de la aplicación móvil y el estado en el que se encuentra la aplicación.

La clase `ServidorControlImagen` se encarga de gestionar los comandos que se reciben desde la aplicación del móvil, de tal modo que si se recibe un comando "Iniciar", esta clase ordena a `EmitirImagen` que comience a mandar datagramas con imágenes al MIDlet que ha solicitado el inicio de la visualización. Del mismo modo, si se recibe un comando de "Fin", la clase `ServidorControlImagen` parará la emisión de imágenes. Como se puede observar de la Figura 47, los comandos llegan a través de una comunicación por Socket.

La clase `EmitirImagen` se encarga de enviar datagramas con imágenes al MIDlet a través de una comunicación UDP. Esta clase se encarga de leer las imágenes que le suministra la aplicación capturadora, empaquetarla dentro de un datagrama para luego enviarlo por una comunicación UDP hacia el móvil. Este proceso, se hace desde un nuevo hilo de ejecución para poder recibir la orden de fin de la visualización cuando sea preciso.

3.6.5. Comportamiento. Interacción entre objetos

En el diagrama de la Figura 48 se puede observar como cada componente interviene en esta alternativa de visualización.

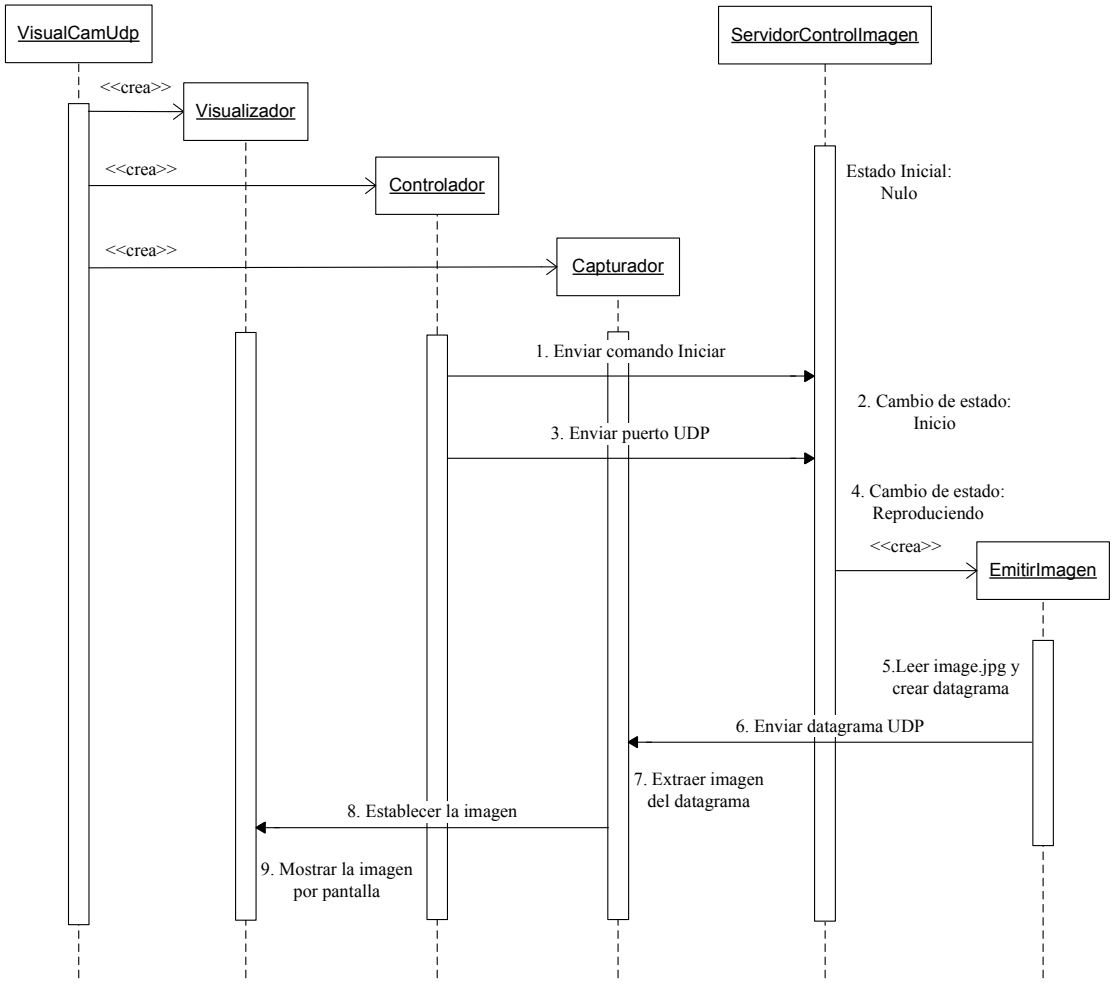


Figura 48. Diagrama de secuencia de la visualización utilizando el protocolo UDP

Este diagrama explica el proceso por el cual se establece la visualización de las imágenes en el móvil. A su vez, se obvia la participación de la aplicación Capturadora *Willing Webcam*, ya que el papel que desempeña es el mismo que en la alternativa de visualización utilizando el protocolo TCP.

Como ya se ha dicho, el diagrama refleja como se establece la visualización de imágenes por la pantalla del móvil. Como se puede observar, el MIDlet (aplicación que se ejecuta en el móvil) crea los componentes `Visualizador`, `Capturador` y `Controlador`, cuyos papeles en la comunicación ya se han explicado anteriormente (Véase Ap. 2.7.4.Estructura y comportamiento). Una vez creados estos componentes el `Controlador` envía a través de una comunicación por Sockets el comando “Iniciar”. La aplicación `ServidorControlImagen` (que recuérdese es la clase principal de la Aplicación Java), se encuentra en estado “Nulo” inicialmente, y cuando recibe el comando “Iniciar” cambia su estado al de “Inicio”. La aplicación en este estado, está esperando un nuevo mensaje procedente

del `Controlador`, que contenga el número de puerto UDP por el que pueda enviarle los datagramas con imágenes. Este mensaje el `Controlador` también lo manda a través de una comunicación por Sockets. Si el numero de puerto recibido es correcto (es un valor entre 1 y 65535) entonces `ServidorControlImagen` cambiará al estado "Reproduciendo". Con el fin de poder mandarles las imágenes al móvil a través del puerto UDP, `ServidorControlImagen` extrae de la comunicación por Socket (establecida para recibir el número de puerto UDP) la dirección IP remota (o del móvil).

Cuando la aplicación `ServidorControlImagen` se encuentra en estado "Reproduciendo", crea el componente `EmitirImagen` pasándole la IP del móvil y puerto UDP por el que espera los datagramas. Este componente es el que se encarga de leer las imágenes que captura la aplicación *Willing Webcam* de la cámara WEB, generar los datagramas con ellas y enviarlos al `Capturador`.

Como se puede observar, cuando el `Capturador` recibe los datagramas, extrae de ellos las imágenes y se las suministra al `Visualizador` para que las muestre por pantalla.

Al crear el componente `EmitirImagen` el proceso desde este punto es cíclico cada cierto intervalo de tiempo, más concretamente cada dos segundos se le enviarán datagramas con imágenes al móvil para que los muestre por pantalla.

El proceso para Finalizar la visualización de imágenes, se produce cuando el `Controlador` envía a través de una comunicación por Socket el comando "Fin". Si el componente `ServidorControlImagen` se encuentra en estado "Reproduciendo" recibe dicho comando, este componente pasará al estado "Nulo" y eliminará al componente `EmitirImagen` (Véase Figura 43). A su vez el `Controlador` mandará para de recibir datagramas al `Capturador`.

3.6.6. La aplicación. Instalación, ejecución y manejo.

3.6.6.1. Nodo móvil

La aplicación que se ha creado para ser ejecutada en el nodo móvil se llama `VisualCamUdp` (Véase el apartado "2.5.6.1.Nodo móvil" para completar información o Apéndice B si se desea más información de cómo desarrollar una aplicación MIDP para móviles).

En los siguientes apartados se explican los pasos necesarios para la instalación, la ejecución y el manejo de `VisualCamUdp` utilizando *Java Wireless Toolkit 2.5 for CLDC* (esta herramienta puede encontrarla de forma gratuita en http://java.sun.com).

a. Instalación

Véase el mismo punto del apartado 3.5.6.1.Nodo móvil.

b. Ejecución

Véase el mismo punto del apartado 3.5.6.1.Nodo móvil. Hay que tener en cuenta que ahora la aplicación es `VisualCamUdp`.

Una vez ejecutada la aplicación la pantalla inicial es la que muestra la siguiente figura:

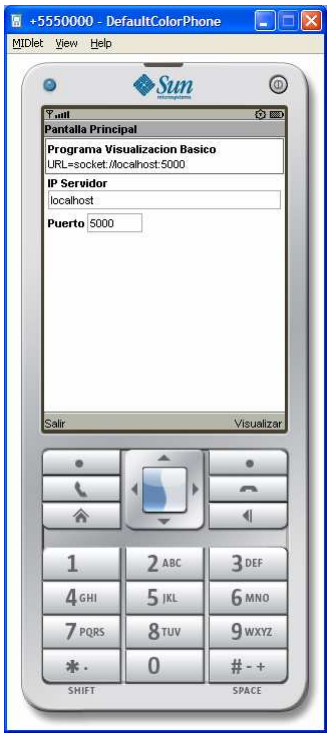


Figura 49. Pantalla inicial del MIDlet

c. Manejo

En esta alternativa de visualización utilizando el protocolo UDP va a requerir la participación de tres nodos en la comunicación: el móvil, la aplicación Java y la aplicación capturadora (Véase Figura 41). Como ya se ha hecho constar anteriormente, la aplicación Java es la que se encarga de gestionar las peticiones de inicio y fin requeridas por el móvil, es por esta razón por lo que es preciso que la aplicación Java se esté ejecutando cuando el móvil haga las solicitudes. Está claro que si esta aplicación no se ejecutase el proceso de visualización no se podría llevar a cabo.

Los pasos que hay que realizar para visualizar las imágenes en el móvil son:

1º) Establecer la dirección IP del PC donde se está ejecutando la aplicación `ServidorControlImagen` (Recuérdese que es así como se llama la aplicación Java).

2º) Establecer el puerto por donde la aplicación `ServidorControlImagen` ofrece el servicio. Este puerto se utiliza para mandar las solicitudes de inicio y fin de la visualización.

3º) Una vez estén los parámetros de la visualización definidos pulsar el botón de “Conectar” del móvil. Aparecerá la pantalla de visualización (Véase Figura 50)

4º) Para comenzar la visualización pulsar “Capturar”, se pedirá autorización para realizar la comunicación (Véase Figura 51)

5º) Comienza la visualización de imágenes (Véase Figura 52).

La aplicación dispone de un botón “Salir” en la pantalla inicial (Véase Figura 49) para finalizar la ejecución de la misma. Además, cuando nos encontramos visualizando imágenes (Véase Figura 52), podemos parar dicha visualización en el momento en el que queramos pulsando el botón “Parar”. También tenemos la opción de volver a la pantalla inicial pulsando el

botón “Volver”, pudiendo realizar una nueva configuración para la visualización de las imágenes.

Figura 50. Pantalla del visualizador

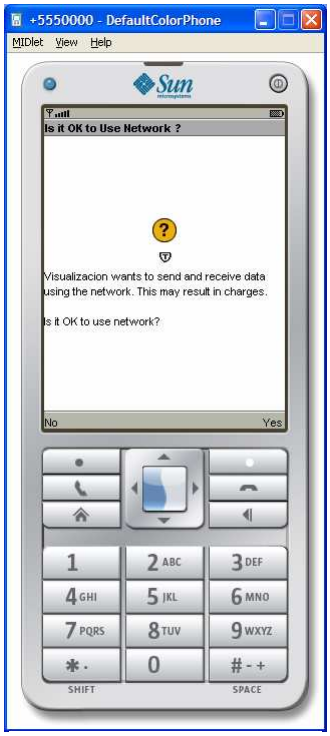


Figura 51. Autorización de conexión

Figura 52. Visualización de imágenes

3.6.6.2. Nodo Aplicación Java

La aplicación Java que se ejecuta en el nodo PC se llama `ServidorControlImagen`. Para desarrollarla es necesario tener instalado el *Java SE Development Kit* (JDK) y *Java Runtime Environment* (JRE) que se pueden descargar de la página de Sun (http://java.sun.com).

Para implementar la aplicación se puede realizar desde cualquier interfaz de desarrollo Java, como por ejemplo Eclipse. Este se puede descargar de forma gratuita desde www.eclipse.org.

a. Instalación

No requiere instalación.

b. Ejecución

Se ejecuta desde el interfaz de desarrollo, en este caso desde Eclipse, eligiendo la opción “Run as Aplication”.

c. Manejo

La aplicación `ServidorControlImagen` no requiere manejo alguno. Esta aplicación presenta un interface gráfico para mostrar el estado en que se encuentra, la dirección IP y el puerto del móvil que manda la solicitud, y la solicitud que se ha recibido (Véase Figura 53). La aplicación además tiene un botón para parar el servidor cerrando así adecuadamente las conexiones.

La aplicación inicialmente está en estado “Nulo”, posteriormente, dependiendo de los comandos que se recibe desde el móvil, variará su estado entre “Inicio”, “Reproduciendo” o “Nulo”.

En estado Inicio se encuentra cuando recibe el comando “Inicio” desde el móvil. En este momento la aplicación espera recibir el número de puerto del móvil por donde mandarle los datagramas. En caso de recibir un numero no válido u otro comando, volverá al estado inicial de “Nulo”.

Cuando la aplicación está en estado “Reproduciendo” significa que se están emitiendo datagramas con imágenes hacia el móvil que solicitó la visualización. Los datagramas se crean con las imágenes recibidas de la aplicación capturadora. Para llegar a este estado, es preciso que previamente se hayan recibido el comando “Inicio” y un número de puerto válido disponible en el móvil por el que esta aplicación mandarle los datagramas.

Finalmente, la aplicación volverá al estado inicial “Nulo” cuando se encuentre en estado “Reproduciendo” y se reciba el comando “Fin”. En este momento la aplicación estará disponible para atender otras solicitudes que desean una visualización.

Figura 53. Consola del servidor de imágenes

3.6.6.3. Nodo Aplicación capturadora

La aplicación capturadora que se utiliza en esta alternativa de visualización utilizando el protocolo UDP, es la misma que para la alternativa que utiliza el protocolo TCP anteriormente mostrada, por lo que para obtener más información sobre la aplicación véase el apartado 3.5.6.3. Nodo Aplicación capturadora.

Capítulo 4.

Ejemplo de aplicación. Control de una maqueta de trenes.

4.1. Introducción

Hasta ahora se han mostrado diferentes alternativas para la comunicación entre un móvil y un PC y la visualización de imágenes a través del móvil, pero no hemos visto un ejemplo práctico que muestre realmente el potencial que ofrecen dichas alternativas. Este es el momento de plantearse un ejemplo práctico para demostrarlo, como por ejemplo desarrollar una infraestructura capaz de controlar una maqueta de trenes a través de un teléfono móvil.

4.2. Objetivos

Este capítulo describe el tercero de los objetivos de este proyecto: desarrollar la infraestructura necesaria para el control de una maqueta de trenes situada en la UPCT desde una aplicación que se ejecuta en un teléfono móvil. Dicha aplicación será capaz de iniciar los trenes (poner en marcha los trenes), realizar una parada de emergencia (parada inmediata de todos los trenes) o cambiar la velocidad de un tren que circule por la vías de la maqueta.

Entre las diferentes alternativas de comunicación mostradas anteriormente, se ha elegido la opción de comunicación directa a través de Socket para desarrollar la aplicación que controlará la maqueta. Para la visualización de la maqueta se podrá utilizar cualquiera de las dos opciones mostradas anteriormente en el capítulo de Visualización.

En la aplicación que se va a desarrollar no se integrará código para la visualización, ya que este cometido lo puede realizar cualquiera de las dos alternativas desarrolladas en el capítulo de Visualización.

En la Figura 54 se puede observar un diagrama con la infraestructura que se va a realizar para el control de la maqueta de trenes.

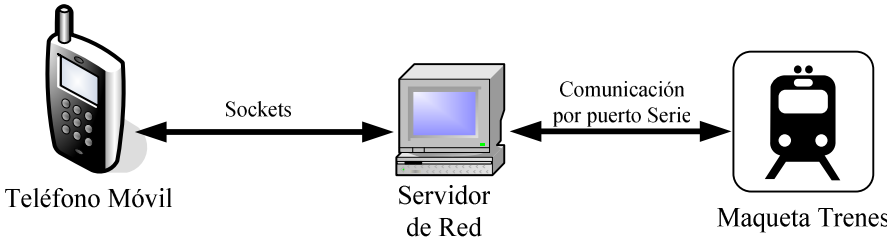


Figura 54. Nodos que intervienen en el control de la maqueta

4.3. Infraestructura

En el diagrama de la Figura 55 se puede observar la infraestructura y los componentes que intervienen en la misma, necesarios para realizar el control de la maqueta de trenes a través de un teléfono móvil.

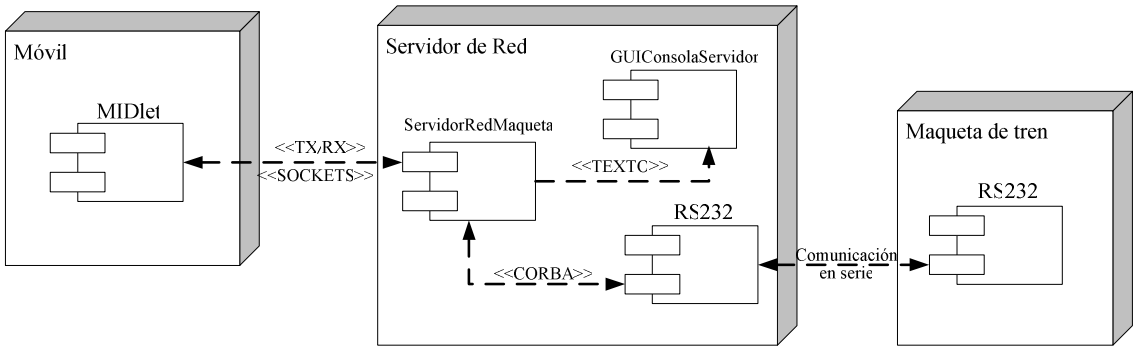


Figura 55. Nodos y componentes para realizar el control de la maqueta

4.3.1. Nodo Móvil

Se precisa de un dispositivo móvil compatible con las API's MIDP 2.0 y CLDC 1.1 de Java ME (Véase Apéndice A). Ya que son estas API's las que hacen posibles la comunicación a través de Socket, en concreto CLDC1.1. Si intentamos realizar la comunicación por Socket a través de un teléfono móvil que no es compatible con esta API será imposible realizar dicha comunicación.

Para probar la aplicación que se va a desarrollar para el control de la maqueta de trenes no se va a utilizar ningún móvil, sino un simulador de teléfono móvil ejecutándose sobre un PC. Este simulador se encuentra dentro del *Java Wireless Toolkit 2.5 for CLDC*, que no es más que un conjunto de herramientas para el desarrollo de aplicaciones en Java ME. Esta herramienta se puede descargar de forma gratuita de la página de Sun http://java.sun.com/javame/index.jsp.

Se va a desarrollar una aplicación para el móvil que podrá realizar tres funciones de control sobre la maqueta de trenes. Podrá activa la marcha de todos los trenes que se encuentren en la vías de la maqueta, hacer una parada de emergencia de todos los trenes, es decir, mandar parar inmediatamente a todos los trenes que se encuentren en las vías de la maqueta y cambiar la velocidad de cualquier tren que se encuentre en las vías de la maqueta.

Cada una de estas funciones, se transmitirán en forma de comando y parámetros a través de una comunicación por Socket hacia el Servidor de Red. Esperando recibir una respuesta afirmativa o negativa del mismo para mostrarla por pantalla.

4.3.2. Nodo Servidor de Red

Antes de empezar a explicar en que consiste el Servidor de Red, es necesario comentar unos aspectos relevantes sobre el control de la maqueta de trenes a través de un PC. Dicho control se puede hacer de forma local o remota desde un PC, gracias a las implementaciones desarrolladas en Java de los interfaces de Corba. Es decir, gracias a los métodos implementados de este interface se pueden mandar comandos que realizan acciones de control sobre la maqueta de trenes, ya que directamente desde Java resulta imposible realizar esta tarea.

Ahora, con el fin de que el teléfono móvil pueda mandar comandos de control a la maqueta, añadiremos a la implementación antes mencionada, un servidor de red. Este servidor se encargará de recibir los comandos de control a través de una conexión por Socket. Según el comando y los parámetros que se reciban, realizará las acciones de control sobre la maqueta de trenes.

Como ya se comento anteriormente, hay disponibles tres funciones para el control de la maqueta de trenes: iniciar la marcha de los trenes, realizar una parada de emergencia de todos los trenes y cambiar la velocidad de un tren específico que esté circulando por la maqueta. Para esta ultima opción, es necesario que se envíe el comando junto con dos parámetros, el primero de ellos será la identificación de la locomotora y el segundo la nueva velocidad a establecer. Por lo que el servidor de red deberá ser capaz de identificar tanto los comandos como los parámetros que vienen asociados a estos. Para tal fin, se ha diseñado un patrón de mensajes para poder identificar los comandos y sus parámetros. Los mensajes que reciba el servidor de red deberán seguir la siguiente estructura:

[comando]&[parámetro1]&[parámetro2]&…

Es decir, los comandos y los parámetros estarán separados por el símbolo “&”. La primera palabra del mensaje identifica el comando de control que se quiere ejecutar sobre la maqueta, el resto del mensaje contiene los parámetros asociados al comando de control.

El modo de funcionamiento del servidor de red es el siguiente:

1º) Comprobar que el servicio de red está activado y si no lo está se deberá activar para que se puedan realizar los siguientes pasos.

2º) El servidor se pone a la espera de recibir mensajes de control.

3º) Cuando un mensaje es recibido, se procesa realizando las acciones de control que describe este comando sobre la maqueta de trenes.

4º) Si la ejecución del comando sobre la maqueta se llevó con éxito, el servidor devolverá el mensaje **OK** al teléfono móvil, en caso negativo se devolverá el mensaje **NO_OK**.

4.3.3. Nodo Maqueta de trenes

La maqueta de trenes es el nodo final de la comunicación. Se encuentra conectada al PC donde se ejecuta el servidor de red por el puerto serie. Los comandos que se ejecutan desde el servidor de red o desde el la propia aplicación de control de la maqueta, se traducen en cambios físicos en la maqueta gracias a un interface Corba. Es decir, cuando el servidor recibe un mensaje con un comando y sus parámetros desde el teléfono móvil, lo ejecuta sobre la maqueta, produciendo un cambio de estado en la maqueta de trenes.

4.4. Protocolo

En el diagrama de la Figura 56 se muestra la secuencia de acciones que suceden desde que se envía un mensaje desde el teléfono móvil (la aplicación que se ejecuta en el móvil se llama MIDlet), pasando por la ejecución física en la maqueta y hasta que se genera una respuesta para ser mostrada en el teléfono móvil.

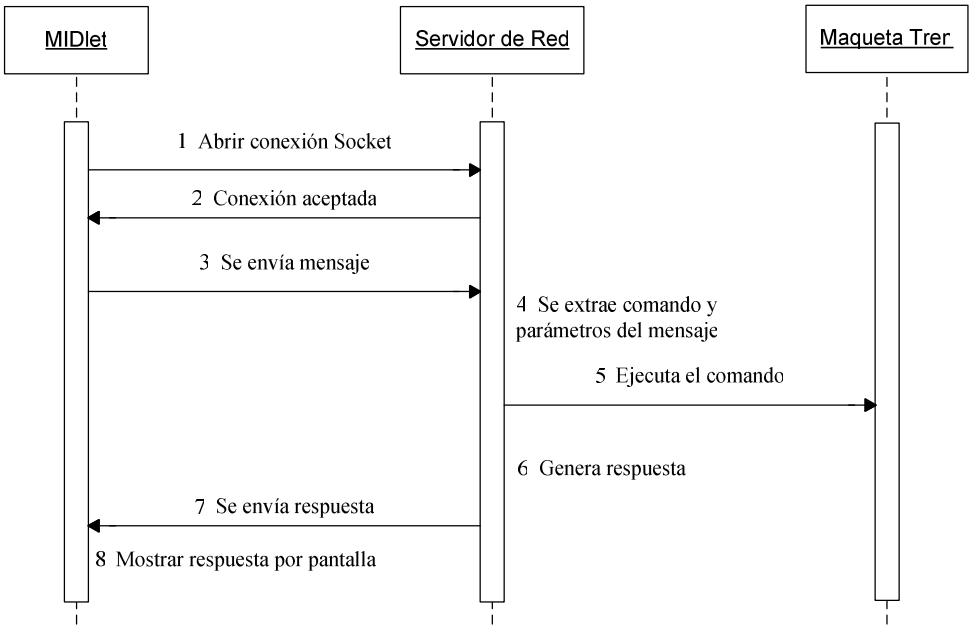


Figura 56. Diagrama de secuencia del protocolo para el control de la maqueta de trenes

4.5. Paquetes y clases Java utilizados

A continuación se mostrarán los paquetes y clases Java utilizadas para el desarrollo de las distintos componentes que forman la infraestructura de este ejemplo para el control de la maqueta (Véase Figura 55). Los componentes desarrollados son:

Nodo Móvil: MIDlet.

Nodo Servidor de Red: ServidorRedMaqueta y GUIConsolaServidor.

• Nodo Móvil: MIDLET

En el diagrama de la Figura 57 se muestra los paquetes que se han utilizados para desarrollar el MIDlet, que recuérdese es la aplicación que se ejecuta en el móvil (Véase Figura 55):

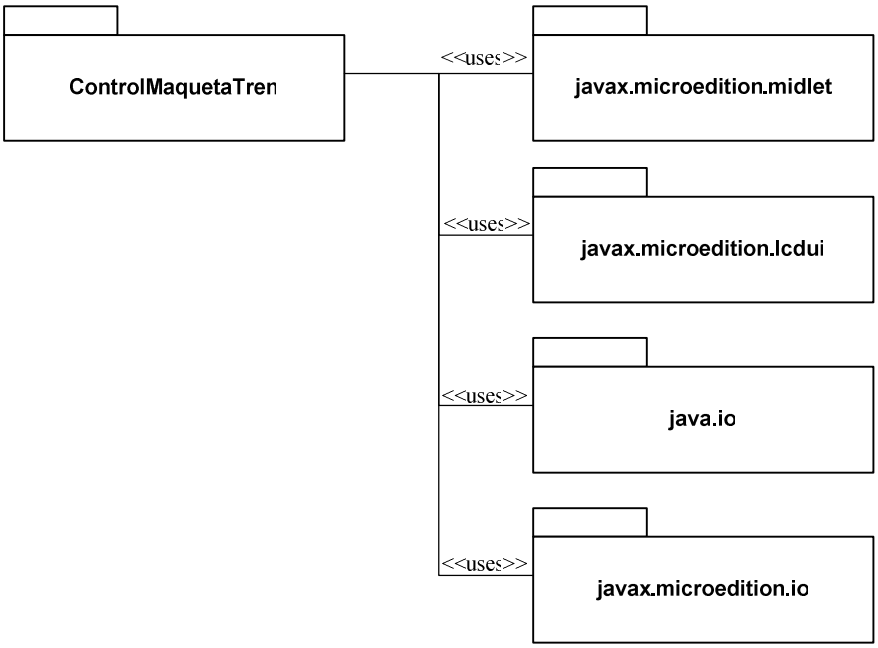


Figura 57. Paquetes Java necesarios para desarrollar el MIDlet

En la Tabla 17 se muestran las clases, interfaces y excepciones utilizadas de cada uno de los paquetes para el desarrollo del MIDlet que se encarga del control de la maqueta:

Paquete	***Clases***	***Excepciones***	***Interfaces***
javax.microedition.midlet	MIDlet		
javax.microedition.lcdui	Display Form TextBox StringItem TextField Command		CommandListener
javax.microedition.io	Connector		SocketConnection
Java.io	InputStream OutputStream	IOException	

Tabla 17. Clases, excepciones e interfaces necesarios para desarrollar el MIDlet

Se puede encontrar más información referente a estas clases, interfaces y excepciones en las Tablas 3 y 8, que se encuentran disponibles en el Capítulo 2 de este documento.

• Nodo Servidor de Red: ServidorRedMaqueta

En el diagrama de la Figura 58 se muestran los paquetes Java utilizados para la creación del componente `ServidorRedMaqueta`, que recuérdese se sitúa en el nodo Servidor de Red (Véase Figura 55).

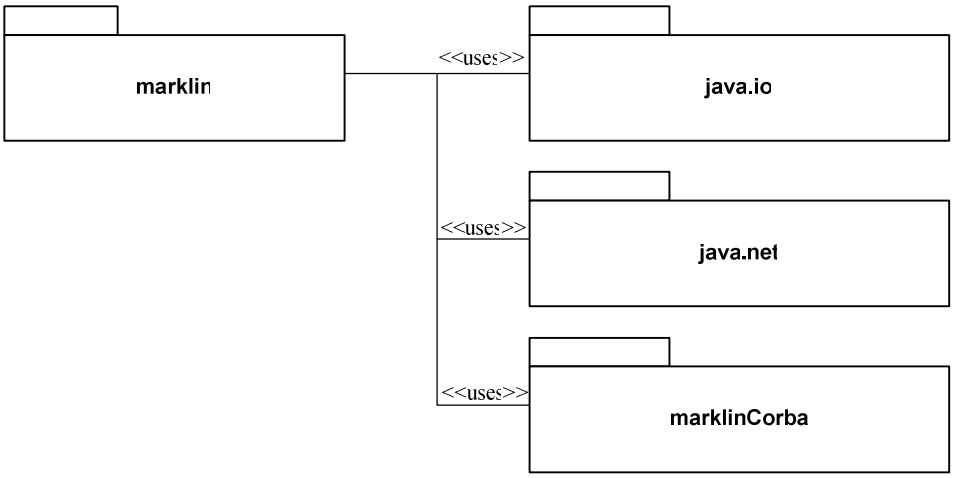


Figura 58. Paquetes Java necesarios para desarrollar el servidor de red

En la Tabla 18 que se muestra a continuación se pueden ver las clases, interfaces y excepciones usadas de cada paquete:

Paquete	***Clases***	***Excepciones***	***Interfaces***
java.io	InputStream OutputStream	IOException	
java.net	ServerSocket Socket		
marklinCorba			MarklinCorbaOperations

Tabla 18. Clases, excepciones e interfaces necesarios para desarrollar el servidor de red

• Nodo Servidor de Red: GUIConsolaServidor

En el siguiente diagrama se muestra los paquetes Java utilizados para el desarrollo del componente `GUIConsolaServidor`:

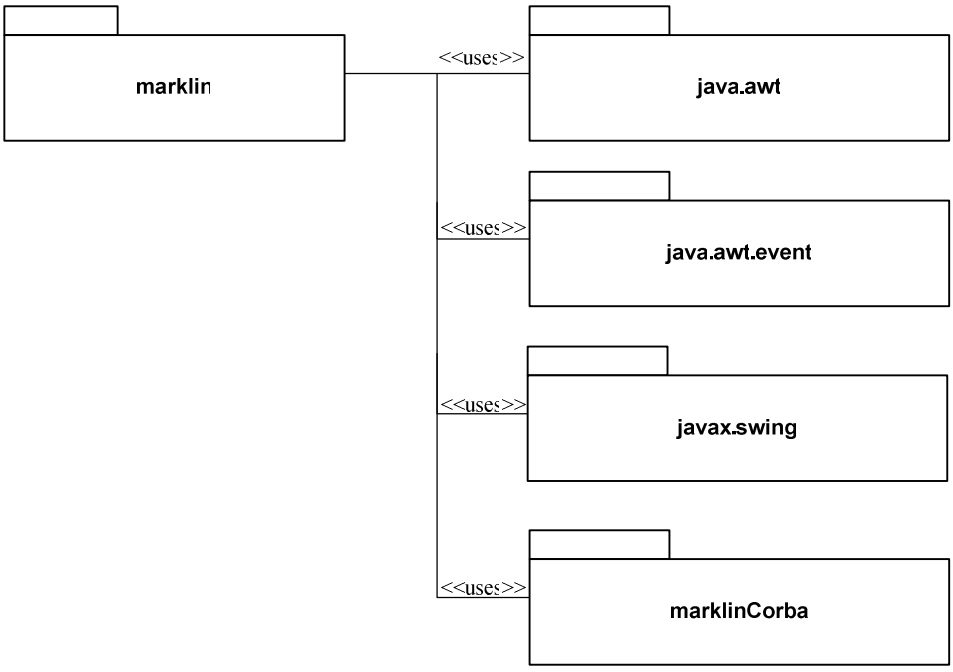


Figura 59. Paquetes Java necesarios para desarrollar la consola del servidor de red

Las clases, interfaces y excepciones utilizados de cada uno de estos paquetes se muestran en la siguiente tabla:

Paquete	***Clases***	***Excepciones***	***Interfaces***
java.awt	GridLayout BorderLayout		
java.awt.event	WindowAdapter		ActionListener
javax.swing	Júrame JTextArea JButton JTextField JLabel SwingUtilities		
marklinCorba			MarklinCorbaOperations
marklin			Publicador

Tabla 19. Clases, excepciones e interfaces necesarios para desarrollar la consola de servidor de red

Como se puede observar en la tabla anterior, se utilizan dos interfaces, de los cuales es conveniente una breve explicación para comprender que papel realizan dentro de la aplicación. Estos interfaces son `MarklinCorbaOperation` y `Publicador`.

La interface `MarklinCorbaOperation` es la encargada de establecer los comandos Corba disponibles para el control de la maqueta. Una implementación de esta, es la encargada de realizar las acciones necesarias para el control de la maqueta de trenes.

La interface `Publicador` establece los métodos para poder publicar información. Una implementación de esta interface se encargara de publicar la información que el Servidor de Red genere.

4.6. Estructura y comportamiento

a. MIDlet

La estructura principal del MIDlet que se ejecuta en el móvil, se muestra en el siguiente diagrama:

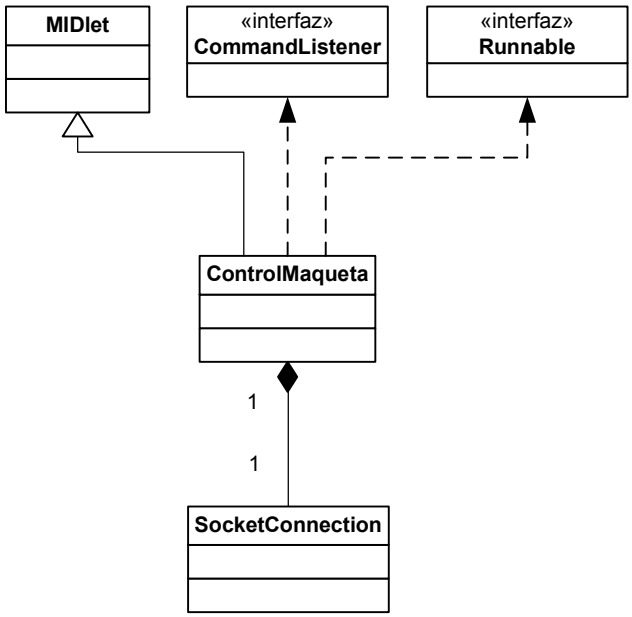


Figura 60. Estructura de clases del MIDlet

La clase `ControlMaqueta` extiende a la clase MIDlet, esto hace que un teléfono móvil sea capaz de ejecutarla. Además, como también se puede ver, `ControlMaqueta` implementa las interfaces `CommandListener` y `Runnable`. Gracias a la primera, el MIDlet es capaz de realizar acciones derivadas de la selección de uno de los comandos disponibles, es decir, la implementación de esta interface implica que se puede seleccionar una acción de control para enviar a la maqueta. La implementación de la interface `Runnable` se hace precisa para poder realizar una conexión de red por Socket sin que la aplicación pueda quedar bloqueada. Por este motivo, es imprescindible realizar las conexiones de red a través de un nuevo hilo de ejecución y evitar así que la aplicación se quede bloqueada. Por último, destacar la composición con un objeto de la interface `SocketConnection`, a través de este objeto se intercambia información entre el MIDlet y el Servidor de Red por medio de una comunicación por Socket.

b. Servidor de Red

La estructura principal que define la aplicación Servidor de Red y sus componentes se muestra en el siguiente diagrama:

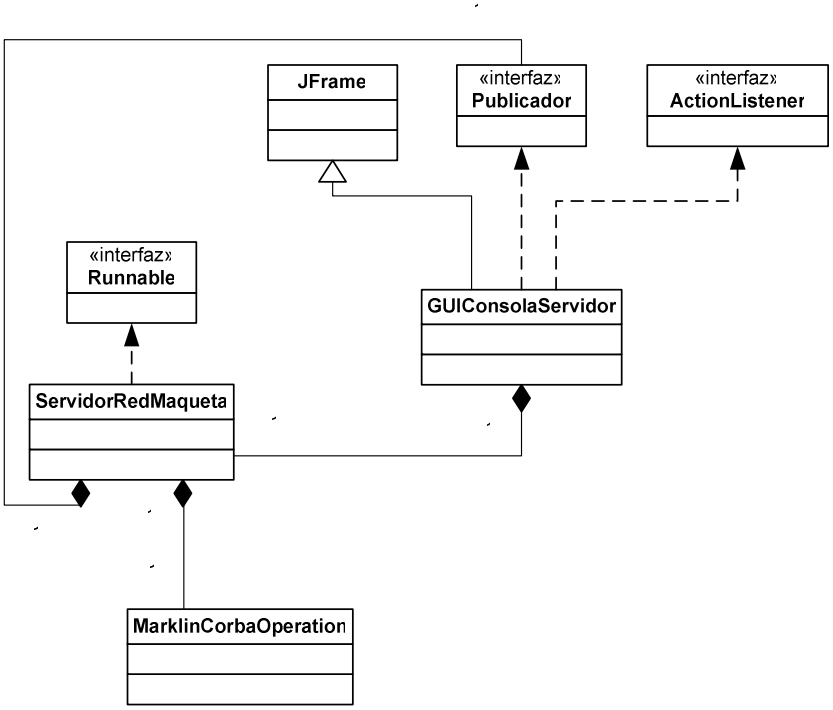


Figura 61. Estructura de clases del servidor de red

Como se puede observar el servidor de red esta compuesto principalmente por dos componentes, como ya se comentó anteriormente, dichos componentes son `GUIConsolaServidor` y `ServidorRedMaqueta` .

El componente `GUIConsolaServidor` hace que nuestro servidor de red presente una interface gráfica para el control y representación de la información, por este motivo este componente extiende a la clase `JFrame`. Como se puede observar implementa dos interface, la primera hace que este componente implemente los métodos necesarios para publicar la información a través de la pantalla. El segundo hace que se pueda iniciar o parar el Servidor de Red cuando se desee. Como se puede observar también, está compuesto de un objeto de la clase

`ServidorRedMaqueta`, es por este motivo por el que se puede mandar, parar o iniciar el Servidor de Red.

El `ServidorRedMaqueta` como su propio nombre indica es el Servidor de Red, básicamente es el que ofrece el servicio a través de la red. Se encarga de establecer las conexiones por Socket, recibir los comandos y sus parámetros asociados, y ejecutarlos sobre la maqueta de trenes. Para poder ejecutar los comandos que recibe por la red necesita la composición de una implementación de la interface `MarklinCorbaOperation`, ya que esta implementación es la que se encarga de la ejecución de los comandos sobre la maqueta de trenes. Todo esto lo hace desde su propio hilo de ejecución, por este motivo implementa la interface `Runnable`. Para representar la información de lo que sucede utiliza una implementación de la interface `Publicador`, en nuestro caso `GUIConsolaServidor`.

4.7. Comportamiento. Interacción entre objetos.

En el diagrama de secuencia de la Figura 62 se puede observar todos los pasos que se realizan en la comunicación para el control de la maqueta.

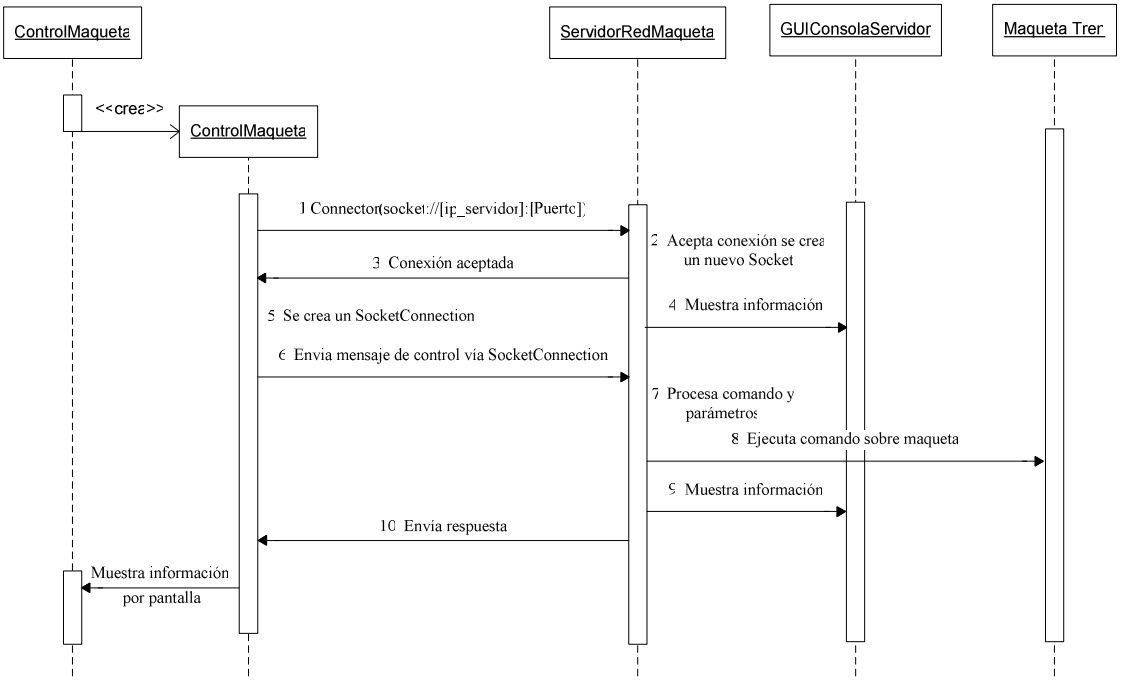


Figura 62. Diagrama de secuencia de la comunicación para el control de la maqueta

Como se puede observar la aplicación `ControlMaqueta` genera un nuevo hilo de ejecución para poder establecer una comunicación a través de Socket con el `ServidorRedMaqueta`. Una vez se ha establecido, la aplicación del móvil manda el mensaje de control sobre la maqueta hacia el servidor, este mensaje contendrá el comando y sus parámetros asociados. El servidor recibe dicho mensaje para posteriormente obtener el comando y sus parámetros. Cuando ha realizado esta operación, el `ServidorRedMaqueta` ya sabe que acción de control debe realizar sobre la maqueta y la ejecuta. Tanto si ha sido posible como si no realizar la acción sobre la maqueta, se enviará un mensaje para confirmar o denegar la ejecución de la acción de control a la aplicación `ControlMaqueta`. Mientras tanto,

`ServidorRedMaqueta` utilizará el componente `GUIConsolaServidor` para publicar toda la información referente a dicho servicio.

4.8. La aplicación. Instalación, ejecución y manejo.

4.8.1. Nodo Móvil

La aplicación desarrollada para el control de la maqueta de trenes desde el teléfono móvil se llama `ControlMaqueta`. Como ya se comento anteriormente, esta aplicación se encarga de mandar las acciones elegidas por el usuario al servidor que controla la maqueta de trenes (Véase el apartado "2.5.6.1.Nodo móvil" para completar información o Apéndice B si se desea más información de cómo desarrollar una aplicación MIDP para móviles).

En los siguientes apartados se explican los pasos necesarios para la instalación, la ejecución y el manejo de `ControlMaqueta` utilizando *Java Wireless Toolkit 2.5 for CLDC* (esta herramienta puede encontrarla de forma gratuita en http://java.sun.com).

a. Instalación

Para instalar la aplicación `ControlMaqueta` en un teléfono móvil son necesarios los ficheros .jad y .jar. El fichero .jar contiene de forma comprimida, el conjunto de clases, librerías y recursos que utiliza la aplicación MIDP. La utilidad del fichero .jad radica en que es usado por el instalador del teléfono para reconocer / instalar / desinstalar la aplicación en cuestión. Estos dos archivos se pueden generar desde el IDE en que se está desarrollando la aplicación.

La instalación de la aplicación en el móvil, varía dependiendo de la marca y el modelo del mismo, pero como se comentó anteriormente, se puede ejecutar la aplicación desde un simulador que se encuentra en *Java Wireless Toolkit for CLDC 2.5*.

b. Ejecución

Como se ha comentado en el apartado anterior (Instalación), la ejecución se llevará a cabo desde el simulador que se encuentra en el conjunto de herramientas *Java Wireless Toolkit for CDLC 2.5*. Hay dos formas de ejecutar la aplicación en el simulador. La primera de ellas es utilizar los archivos .jad y .jar creados por el IDE. La segunda, es ejecutar la aplicación desde el mismo entorno de desarrollo, ya que este utiliza también el simulador para ejecutar la aplicación que se ha creado sin necesidad de crear los archivos .jad y .jar.

En este caso, se va a utilizar la opción de ejecutar la aplicación a través de los archivos .jad y .jar creados desde el entorno de desarrollo. Hay que tener en cuenta que para realizar todos los pasos que se van a explicar a continuación se debe tener instalado el conjunto de herramientas *Java Wireless Toolkit*. Cuando ya se tiene dicha aplicación instalada los pasos a seguir para la ejecución del proyecto serán:

1º) Abrir el explorador y dirigirse a la carpeta "bin" del proyecto Comunicación.

2º) Abrimos el archivo .jad del proyecto, en este caso Comunicación.jad que se encuentra señalado en la Figura 63.

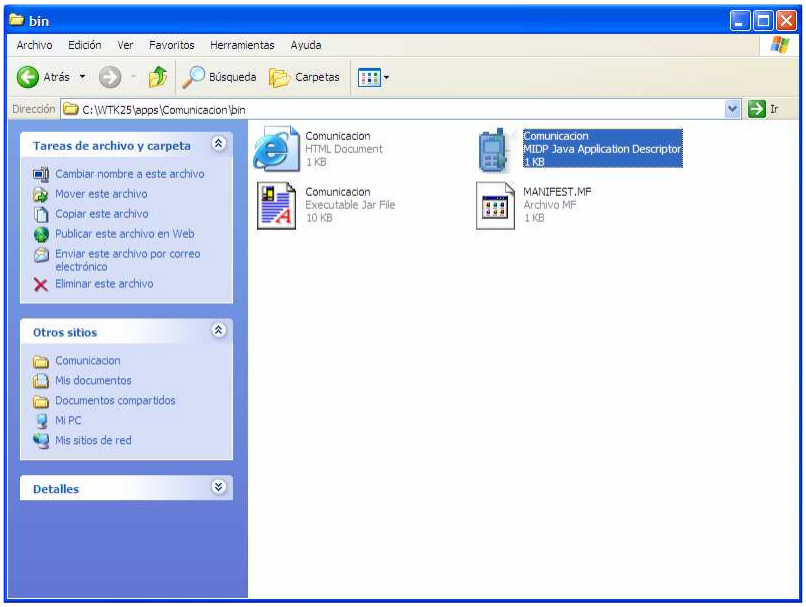


Figura 63. Explorador con la ruta y el jad del MIDlet

3º) Una vez esté abierta la aplicación del proyecto Comunicación seleccionar la opción `CtrlBasicServ` (Véase Figura 64). Recuérdese que así es como se llama la aplicación MIDP que hemos desarrollado para realizar esta comunicación.

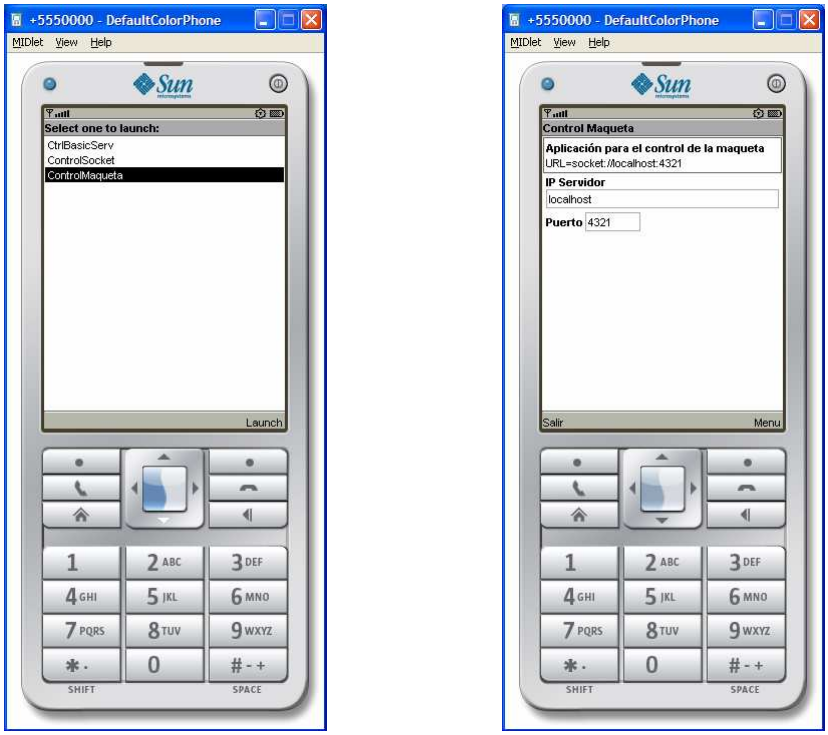


Figura 65. Pantalla inicial del MIDlet

Ahora ya se está ejecutando nuestra aplicación en el nodo móvil (Véase Figura 65).

c. Manejo

Como ya se ha comentado anteriormente, aparecen 3 nodos en la comunicación: el nodo móvil, el Servidor de Red y la maqueta de trenes. Cada uno de ellos, formado por unos componentes que intervienen directamente en la comunicación (Véase Figura 54 y Figura 55).

Si uno de ellos falla, no se está ejecutando o está mal configurado, la comunicación no se podrá realizar.

La aplicación `ControlMaqueta` es capaz de realizar tres funciones de control sobre la maqueta de trenes asociadas a tres comandos, los cuales se detallan en la siguiente tabla:

Comando	Descripción
Start	Inicia la marcha de las locomotoras que estén en la maqueta de trenes. Este comando no tiene asociado ningún parámetro.
Stop	Parada de emergencia. Para todas las locomotoras que se encuentren en la maqueta de trenes. Este comando no tiene asociado ningún parámetro.
Speed	Cambia la velocidad de una locomotora que se encuentra en la maqueta. Este comando tiene asociado dos parámetros: la identificación de la locomotora y la nueva velocidad que se le desea establecer.

Tabla 20. Comandos disponibles para el control de la maqueta

Los pasos que hay que realizar para controlar la maqueta de trenes desde el móvil son:

1º) Establecer la dirección IP del Servidor de Red. Recuérdese que este servidor se ejecuta desde el ordenador que está conectado a la maqueta de trenes a través de un cable del tipo RS-232 conectado al puerto serie del servidor.

2º) Establecer el puerto por donde ofrece el servicio el Servidor de Red.

3º) Seleccionar un comando de control. Para ello se debe pulsar la tecla correspondiente para la opción “Menu” (Véase Figura 65).

4º) Del menú desplegado seleccionar una de la tres opciones disponibles: “Start”, “Stop” o “Speed” (Véase Figura 66).

Figura 66. Pantalla inicial con el menú de comandos desplegado

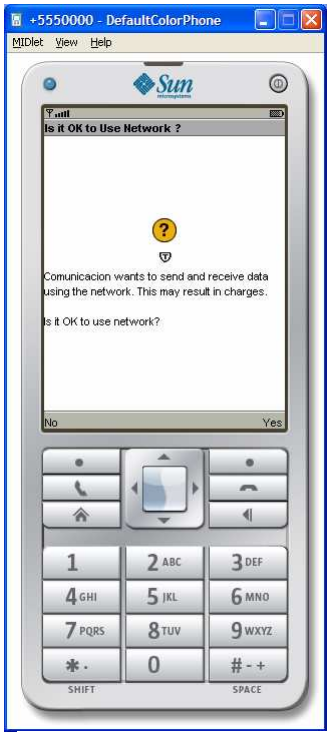


Figura 67. Autorización de conexión

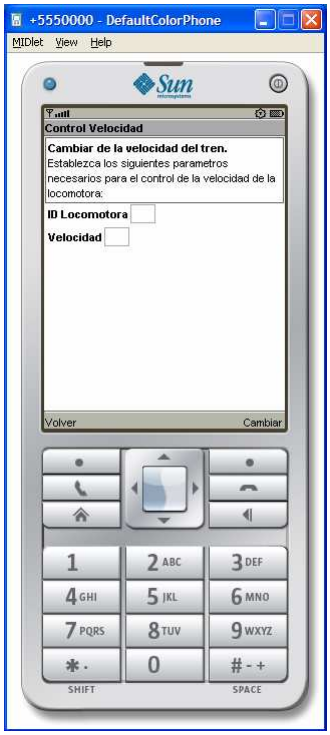


Figura 68. Pantalla para la configuración del comando “Speed”

Si la acción de control que se ha seleccionado es “Start” o “Stop”, se establece directamente la comunicación con el Servidor de Red, puesto que estos dos comandos no llevan ningún parámetro asociado. Para realizar la comunicación del comando se pide una autorización, a la que se debe responder de forma positiva (Véase Figura 67). Si se pulsa la opción “No” la comunicación no se llevará a cabo y por tanto la acción de control sobre la maqueta no tendrá efecto.

Si la acción de control seleccionada es “Speed”, entonces aparecerá una nueva pantalla para configurar los parámetros asociados a esta acción de control (Véase Figura 68). Los parámetros que hay que configurar para ser enviados al Servidor de Red son la identificación de la locomotora y la nueva velocidad que se le quiere asociar. Una vez configurados dichos parámetros, se debe seleccionar la opción de “Cambiar” para enviar dicha acción de control al Servidor de Red. Al igual que con las otras dos acciones de control, cuando se envía esta también se pide confirmación (Véase Figura 67), a la que se debe responder de forma positiva para que la acción pueda llegar al Servidor.

Esta aplicación, después de mandar la acción de control sobre la maqueta, espera recibir una respuesta sobre la ejecución del Servidor de Red. Según la respuesta del Servidor sea positiva o negativa, aparecerá un mensaje u otro (Véase Figura 69 y Figura 70).

Figura 69. Pantalla que confirma la realización de control sobre la maqueta.

Figura 70. Pantalla que indica que no ha sido posible realizar el control sobre la maqueta

4.8.2. Nodo Servidor de Red

El PC que realiza las funciones de control sobre la maqueta de trenes, también realizará las funciones de Servidor de Red. Este servicio se ofrece, como ya se ha comentado anteriormente, gracias a dos componentes: `ServidorRedMaqueta` y `GUIConsolaServidor` (Véase Figura 55). Para el desarrollo de estos dos componentes se ha utilizado Java SE, lo que hace preciso la instalación de un entorno para la programación en dicho lenguaje. Este entorno debe contar con el *Java Runtime Environment* y el *Java*

Development Kit para poder implementar los componentes. Estos programas se pueden descargar de forma gratuita de la página de Sun http://java.sun.com/javase/downloads/index.jsp.

a. Instalación

Estos dos componentes necesarios para el control de la maqueta de trenes, al igual que el conjunto de componentes que forman la aplicación para el control de la maqueta, no necesitan ser instalado puesto que se ejecutan desde el mismo entorno de desarrollo. El entorno de desarrollo elegido para implementación de los componentes ha sido *Eclipse*.

b. Ejecución

Antes de comentar como se ejecuta la aplicación, es necesario saber que la aplicación que controla la maqueta puede ser ejecutada de forma local o remota, esto es debido a la implementación del interface Corba disponible en la aplicación. En este caso, se ha elegido la opción de ejecutar la aplicación de forma local, es decir, que se va a ejecutar desde el mismo ordenador al que está conectada la maqueta a través del puerto serie.

Para lanzar la aplicación que controla la maqueta junto con el servicio de red, se ha creado un nuevo componente llamado `ClienteDeRed`. Este componente es el que hay que ejecutar añadiéndole el parámetro "local" para indicar que se va a ejecutar de forma local el control de la maqueta.

Una vez lanzada la aplicación en modo local, aparecerán dos ventanas (Véase Figuras 71 y72), en la primera de ellas se puede observar la aplicación encargada del control de la maqueta de modo local y en la segunda se encuentra el Servidor de Red implementado en este ejemplo práctico desarrollado.

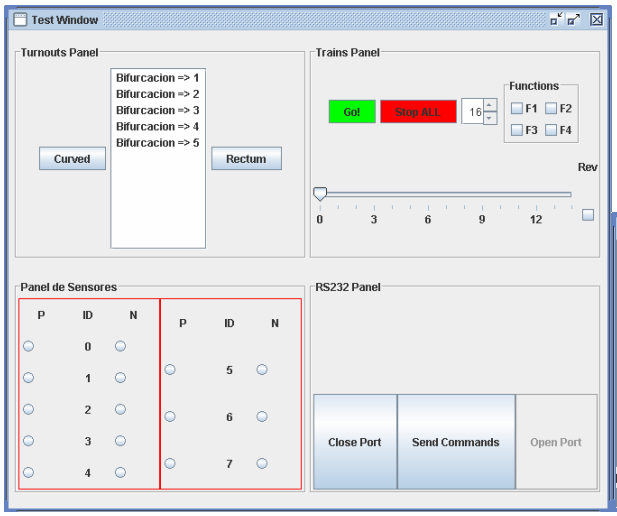


Figura 71. Pantalla para el control local de la maqueta

Figura 72. Pantalla de la consola del servidor de red

Para configurar el Servidor de Red (Véase Figura 72) habrá que seguir los siguientes pasos:

1º) Introducir el puerto de servicio para el servidor. Es recomendable que el número del puerto esté entre 1024 y 65535.

2º) Pulsar el botón para activar el servidor.

En este momento el Servidor de Red ya se está ejecutando y por lo tanto ofreciendo servicio para que desde la aplicación que se ejecuta desde el móvil se puedan enviar comandos de control sobre la maqueta.

c. Manejo

El Servidor de Red no requiere ningún tipo de manejo, puesto que una vez ejecutado, solo se encarga de recibir los comandos enviados desde el móvil para ejecutarlos directamente sobre la maqueta.

Capítulo 5.

Conclusiones.

Como ya se ha demostrado en los capítulos anteriores, se puede ver que la comunicación entre dos plataformas diferentes como son los teléfonos móviles y los PC es posible a través de un lenguaje como Java. Cada una de las alternativas mostradas, tanto para la comunicación como para la visualización, tienen sus ventajas y desventajas, además de requerimientos de Hardware y limitaciones. En este capítulo intentaremos dejar claro, cuales y como afectan las características a la comunicación que se pretende realizar.

Con el fin de que las conclusiones queden más claras se analizaran cada grupo de alternativas según sea de comunicación o de visualización. De esta manera el lector podrá tener más claras las diferencias que existen entre cada una de la alternativas presentadas.

Conclusiones de Comunicación

En el Capítulo 2, se presentan dos alternativas diferentes para el intercambio de información entre el teléfono móvil y el PC. La primera de ellas es la comunicación a través de un Servlet y la segunda es la comunicación directa a través de Sockets. Ahora se mostrará las diferencias existentes entre ambas alternativas teniendo en cuenta infraestructuras necesaria, limitaciones de comunicación, ventajas y desventajas.

Infraestructura

- Comunicación a través de Servlets.

  Hace falta la presencia de tres nodos en la comunicación:

  1) El teléfono móvil, que debe ser capaz de conectarse a Internet y ser compatible con las API's CLDC1.1 y MIDP2.0 de Java MicroEdition (Véase Apéndice A para más información).

  2) Un servidor WEB con capacidad de ejecutar el Servlet que hace de pasarela entre el móvil y un el PC.

3) Un PC con capacidad para conectarse con el servidor WEB donde se ejecuta el Servlet a través de Socket.

- Comunicación directa a través de Sockets.

  Se precisan solo dos nodos en la comunicación:

  1) El teléfono móvil, que debe ser capaz de conectarse a Internet y ser compatible con las API's CLDC1.1 y MIDP2.0 de Java MicroEdition (Véase Apéndice A para más información).

  2) Un PC en donde se ejecuta la aplicación Java y debe tener acceso a Internet para poder desarrollar esta comunicación.

Limitaciones en la comunicación

Las compañías telefónicas ofrecen conectividad a Internet a través de un servidor Proxy. Un servidor Proxy sirve caché de páginas WEB y como barrera de seguridad filtrando paquetes no deseados, además no permiten que haya tráfico directo entre las partes que realizan la comunicación. Este servidor es un problema a la hora de realizar la comunicación entre ambas plataformas, ya que el servidor Proxy de las compañías telefónicas solo deja pasar tráfico HTTP (Véase Figura 73). Por este motivo, la alternativa de comunicación directa a través de Socket no es posible en la actualidad, aunque las prestaciones de los teléfonos móviles y de Java MicroEdition lo hacen practicable.

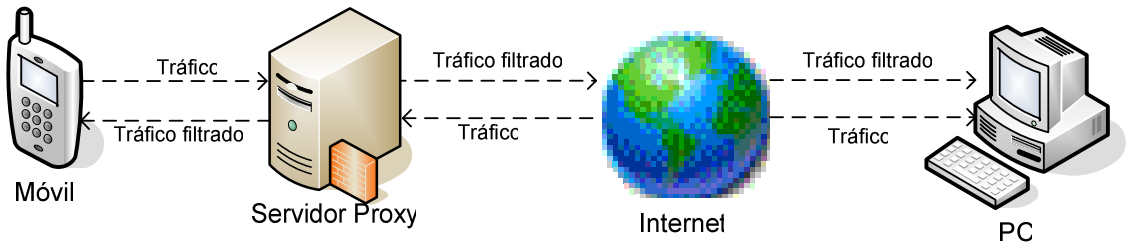


Figura 73. Limitaciones en la comunicación usando un servidor proxy

La implantación de este sistema no es un capricho de las compañías telefónicas, ya que de este modo se evitan posibles problemas de seguridad como virus o ataques malintencionados que pueden causar daños en los teléfonos móviles. Por el momento, a falta de que los teléfonos incorporen antivirus y firewalls, la única alternativa que se puede utilizar para implantarla en una aplicación para teléfonos móviles es la comunicación a través de Servlets.

Ventajas y desventajas

Con todo esto, las ventajas y desventajas de cada una de las alternativas son las siguientes:

	Ventajas	Desventajas
Comunicación a través de Servlets	- Compatibilidad con la mayoría de móviles disponibles en el mercado. Es suficiente con que el móvil pueda acceder a Internet. - Compatibilidad con diferentes plataformas. No sólo se podría mandar información desde un móvil, sino que también se podría desde un navegador WEB. - **Se puede realizar una aplicación para un móvil y este es capaz de realizar la comunicación.**	- Necesidad de intermediarios en la comunicación, que implica que si se cae el servidor WEB, aunque los demás nodos funcionen correctamente, la comunicación no se puede establecer. - La comunicación se hace menos segura, ya que el envío de la información se hace a través de la URL. - Comunicación más lenta. - Mayor infraestructura. Más costoso su mantenimiento
Comunicación directa a través de Sockets	- No son necesarios intermediarios, lo que hace que sea una comunicación mucho más ágil. - Puede realizarse una comunicación mucho más segura si fuese preciso. - Comunicaciones más rápidas.	- Incompatibilidad con muchos teléfonos móviles disponibles en el mercado. - **No se puede establecer una comunicación entre teléfono móvil y aplicación del PC debido a que el móvil utiliza un servidor Proxy para conectarse a Internet.**

Tabla 21. Ventajas y desventajas de las alternativas de comunicación

Conclusiones de Visualización

En el Capítulo 3 se presentan dos alternativas para la visualización de imágenes a través del teléfono móvil. La primera de estas alternativas ofrece la posibilidad de visualizar imágenes haciendo uso del protocolo TCP mediante peticiones HTTP, la segunda alternativa hace uso del protocolo UDP, siendo preciso un protocolo de control sobre la visualización que se hace a través de la segunda alternativa mostrada en el capítulo referente a comunicación. Ahora se mostrará las diferencias existentes entre ambas alternativas teniendo en cuenta infraestructuras necesarias, limitaciones de la visualización, ventajas y desventajas. Para después dar constancia de posibles mejoras sobre las alternativas existentes y nuevas alternativas que se pueden estudiar a partir de ahora.

Infraestructura

- Visualización utilizando protocolo TCP.

  Hace falta la presencia de tres nodos en la visualización:

  1) El teléfono móvil, que debe ser capaz de conectarse a Internet y ser compatible con las API's CLDC1.1 y MIDP2.0 de Java MicroEdition (Véase Apéndice A para más información).

  2) El servidor HTTP capaz de gestionar las peticiones de imágenes recibidas desde el móvil.

3) La aplicación encargada de capturar las imágenes de la webcam.

- Visualización utilizando protocolo UDP.

  Además de la creación de un protocolo de control capaz de iniciar o parar el mecanismo de visualización desde el móvil, hace falta la participación de tres elementos:

  1) El teléfono móvil, que debe ser capaz de conectarse a Internet y ser compatible con las API's CLDC1.1 y MIDP2.0 de Java MicroEdition (Véase Apéndice A para más información).

  2) La aplicación Java encargada de gestionar los comandos de control de la visualización y la emisión de imágenes hacia el teléfono móvil.

  3) La aplicación encargada de capturar las imágenes de la webcam.

Limitaciones de la visualización

Las alternativas de visualización que se han presentado tienen una finalidad didáctica y por este motivo también sus limitaciones: no comprime las imágenes que se transfieren y no hace uso de protocolos específicos como es RTP/RTCP.

- La transferencia de imágenes no está optimizada ya que las imágenes se transmiten sin comprimir, esto produce un mal aprovechamiento del ancho de banda disponible entre el móvil y el PC.

- No se hacen uso de protocolos específicos para la transmisión de imágenes en tiempo real como es RTP/RTCP. Este protocolo ofrece muchas más posibilidades que el implementado en el desarrollo de las alternativas de visualización expuestas en este documento.

Aunque también hay que tener en cuenta que para visualizar imágenes usando el protocolo RTP/RTCP es necesario que el teléfono móvil que se desea utilizar sea compatible con la API de Java Mobile Media API 1.2. además de las anteriormente mencionadas.

Por lo tanto el utilizar las alternativas mostradas anteriormente en este documento pueden acarrear mayores retardos que utilizando el protocolo RTP/RTCP.

Problemas de portabilidad

Una limitación que cabe destacar es la maquina virtual que se ejecuta en cada teléfono móvil compatible con Java MicroEdition, ya que cada marca o teléfono móvil lleva una maquina virtual diferente, que no funcionan igual en todos los casos. Esto depende de la implementación de dicha máquina virtual por cada fabricante. De este modo, algunos teléfonos móviles pueden ser capaces de realizar unas acciones a priori, que otros teléfonos con las mismas compatibilidades no son capaces de realizar. Un ejemplo de este caso es la visualización de imágenes en formato JPG en el teléfono móvil. En el desarrollo de las dos alternativas de visualización se ha utilizado el formato de imagen JPG, por lo que hay que tener en cuenta que para poder visualizar dicha imagen en el teléfono móvil debemos asegurarnos que es capaz de representar dicho formato de imágenes.

Ventajas y desventajas.

Las ventajas y desventajas que presenta la alternativa de visualización de imágenes en tiempo real usando el protocolo TCP son:

Ventajas	Desventajas
- Una infraestructura sencilla de implementar. - Permite gestionar el intervalo de tiempo entre imágenes. - El desarrollo de la aplicación para el móvil es más sencilla. - No es preciso generar ningún nuevo protocolo de nivel de aplicación para la visualización. Se utiliza HTTP.	- Una solicitud por cada imagen que se desea mostrar. Lo que implica la necesidad de mayor ancho de banda. - Se producen mayores retrasos. Cuando la comunicación falla, se produce retransmisión de imágenes. - Como consecuencia de lo anterior, se pierde el tiempo visualizando imágenes pasadas, cuando nos interesa ver las imágenes actuales.

Tabla 22. Ventajas y desventajas de la alternativa de visualización usando el protocolo TCP

La alternativa de visualización de imágenes en tiempo real usando el protocolo UDP presenta las siguientes ventajas y desventajas:

Ventajas	Desventajas
- La perdida de imágenes en la comunicación no implica retrasos en la visualización. - No se pierde el tiempo visualizando imágenes pasadas, siempre se verán las imágenes actuales. - Optimiza mejor el ancho de banda. Solo hace falta la solicitud de inicio y fin de la visualización.	- Precisa generar un nuevo protocolo de aplicación al que deben ajustarse la aplicación del móvil y la aplicación Java. - De lo anterior, se requiere una mayor complejidad a la hora de implementar las aplicaciones. - No se puede ajustar los tiempos entre imágenes.

Tabla 23. Ventajas y desventajas de la alternativa de visualización usando el protocolo UDP

Posibles mejoras de las alternativas

La posible mejora que se proponen para las alternativas de visualización es realizar una aplicación que se encargue de capturar imágenes de la webcam en puesto de utilizar una aplicación externa como se ha utilizado para el desarrollo de las dos alternativas. Esto se puede hacer con Java Media Framework y de esta forma poder utilizar el formato de imágenes PNG para visualización de imágenes. Este formato de imágenes si que es soportado por la mayoría de las máquinas virtuales implementadas por los fabricantes.

Nuevas alternativas a estudiar

La mejor utilización de la alternativa de visualización utilizando el protocolo TCP, no es la visualización de imágenes en tiempo real. Esta alternativa esta mejor ideada para tomar una instantánea o solicitar una imagen a un servidor WEB, no para visualizar imágenes en tiempo real. Como futuras aplicaciones que pueden hacer uso de esta alternativa se pueden pensar en aplicaciones de seguridad para el control a través del móvil de instantáneas de alguna zona determinada, o para ver una instantánea de la guardería en la que se encuentran nuestros hijos.

En cuanto a la utilización de la alternativa de visualización utilizando el protocolo UDP, no hay mucho que mejorar salvo lo anteriormente mencionado en el apartado de mejoras. Ya que los principios que se utilizan son los mismo que se utilizan para la visualización de

imágenes a través del protocolo RTP/RTCP. Pero este protocolo para la visualización en tiempo real ofrece muchas más opciones que el que se ha desarrollado en esta alternativa, es mucho más funcional, por lo que se debería realizar su estudio a la hora de visualizar imágenes en tiempo real a través del teléfono móvil.

Capítulo 6.

Bibliografía

Libros

[1] Java how to program
Autor: Deitel, H.M.
Editorial: Upper Saddle River: Prentice Hall

Recursos Web

[2] Java Technologic
Link: http://java.sun.com/

[3] Programación de juegos para móviles con J2ME
Link: http://www.programacion.net/java/tutorial/ags_j2me/

[4] The Java ME Plataform
Link: http://java.sun.com/javame/index.jsp

[5] NetBeans
Link: http://www.netbeans.org/index.html

[6] Eclipse
Link: http://www.eclipse.org/

[7] Willing Webcam
Link: http://www.willingsoftware.com/

[8] The Apache software
Link: http://www.apache.org/

[9] Introducción a J2ME
Link: http://www.todosymbian.com/secart36.html

[10] Experiments in Streaming Content in Java ME
Link: http://today.java.net/pub/a/today/2006/08/22/experiments-in-streaming-java-me.html

[11] Conexión a red en dispositivos móviles
Link: http://leo.ugr.es/J2ME/REDES/redes.htm

[12] CGI: Common Gateway Interface

Link: http://www.w3.org/CGI/

[13] Protocolo HTTP

Link: http://www.infor.uva.es/~jvegas/cursos/buendia/pordocente/node13.html

Apéndice A.

Introducción a Java MicroEdition

Se pone a disposición este apéndice sobre conceptos básicos de Java ME porque es interesante, aunque no necesario, conocer ciertos conceptos relacionados con java y esta versión microedtion (J2ME).

Configurations

Configuración es una especificación que cubre una gama de equipos con características similares, en ella se define las clases mínimas que esos equipos pueden usar. Define los requerimientos del VM (Virtual Machine: intérprete de java) y de las clases principales. Para J2ME CLDC y CDC. Sun ha desarrollado KVM y CVM para CLDC y CDC respectivamente, CLDC es la que nos interesa a nosotros.

CLDC

Connected Limited Device Configuration que cubre las necesidades de pequeños aparatos con limitadas posibilidades en cuanto a interfaz de usuario, poder de proceso, etc.

Esta configuración posee la K Virtual Machine, un intérprete de java preparado para microprocesadores de 16 y 32 bits RISC/CISC con tan solo unos pocos cientos de Kb de memoria. Debido a esto, CLDC no incluye ciertos tipos de clases, por ejemplo en la versión 1.0 no se pueden usar números float.

Las clases obtenidas de la versión de J2SE son:

- java.lang.*
- java.io.*
- java.util.*

Nuevas clases son:

- java.microedition.io.*

Actualmente Sun ofrece dos configuraciones posibles a los usuarios que quieran programar en este lenguaje:

- CLDC 1.0: Primera configuración creada para esta plataforma.
- CLDC 1.1: Última configuración a disposición de los usuarios. Incluye nuevas clase que hace más compatibles las aplicaciones para las conexiones de red.

Importante: El verificador de J2SE es demasiado grande para ser incluido con el CLDC, de hecho es mas grande que el KVM, por lo cual debemos verificar los archivos antes de mandarlos al equipo donde queremos que se ejecuten. Para esta tarea, los SDKs poseen herramientas que nos ayudaran en su proceso y no será necesario realizarlo manualmente.

Profiles

Un perfil es una especificación de las APIs de java, que funciona en conjunción de la capa de configuración. Añaden nuevas clases a las que venían con la configuración.

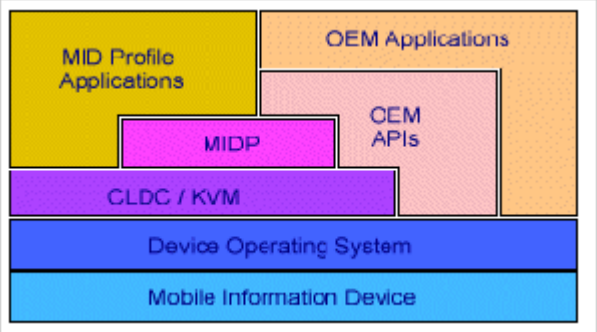
Hay muchos perfiles, pero los más importantes en referencia a J2ME son Personal Profile, Foundation Profile y MID Profile.

MIDP

Este perfil esta diseñado para funcionar especialmente con CLDC.

Las clases que contiene son:

- javax.microedition.midlet: se ocupa del ciclo de vida de la aplicación
- javax.microedition.lcdui: interfaz de usuario
- javax.microedition.rms: sistema de mantenimiento de registros (Record Management System) usado para guardar información
- javax.microedition.io: clases para usar redes
- java.lang: clases de lenguaje
- java.util: clases de utilidades



Arquitectura del MIDP

Actualmente Sun tiene dos perfiles disponibles para los usuarios que quieran hacer uso de ellos. Estos perfiles son:

- MIDP 1.0: Primer perfil desarrollado para Java ME.

- MIDP 2.0: Último perfil que Sun pone a disposición de los usuarios. Incluye nuevas clases y ofrece más compatibilidad principalmente con conexiones a la red.

Ciclo de vida del MIDP

El ciclo de vida de un MIDP esta muy bien definido ya que ayuda al MIDlet a coexistir con otros programas en el MIDP. Las fases del ciclo de vida son:

- Retrieval
- Installation
- Launching
- Version Management
- Removal

Retreival

El teléfono consigue la aplicación desde la fuente, puede ser vía IRDA, Bluetooth o Internet. En este momento el MIDlet y MID establecen una comunicación para intercambiar información sobre el sistema y la aplicación y decidir así si se procede a instalarlo o no.

Installation

La aplicación se instala en el MID. La implementación de MIDP verifica que el nuevo MIDlet no viola la seguridad de MID.

Launching

El usuario ejecuta la aplicación. En esta fase, el MIDlet se ejecuta en la KVM y los métodos de ciclos de vida del MIDlet son ejecutados.

- MIDlet Instance creation – Paused
- MIDlet initialization – Active
- MIDlet termination – Destroyed

Version management

El teléfono mantiene una base de datos sobre que programas han sido instalados y su versión. Así, usando la descripción (descriptor) del MIDlet puede ser actualizado si aparece una nueva versión

Renoval

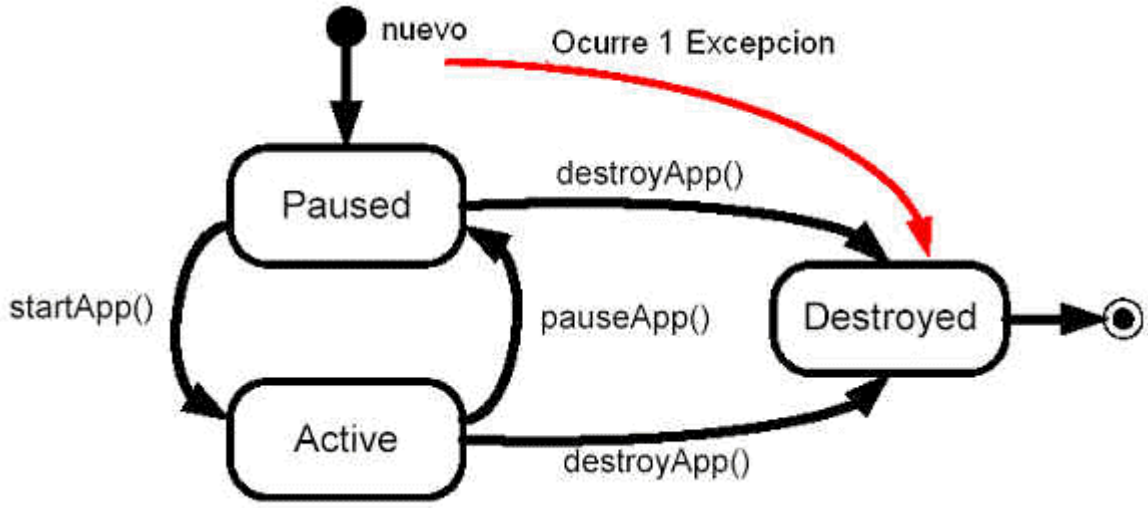
El MIDlet es borrado del teléfono.

Aplicaciones MIDP: MIDlets

Un MIDlet es una aplicación escrita especialmente para el perfil MIDP de J2ME.

Hay muchos conceptos básicos que serian importantes conocer para familiarizarse con los MIDlets, como son el ciclo de vida, interfaz de usuario, instalación, timers, redes, etc.

Ciclo de vida de un MIDlet



Ciclo de vida de un MIDlet

- **startApp()**
  - método setCurrent() debería ser llamado aquí si no fue llamado antes. setCurrent() define que display será visible al usuario, solo un display puede ser visible al mismo tiempo.
  - Puede ser ejecutado más de una vez, así que es mejor no poner ninguna inicialización aquí.
- **pauseApp()**
  - cuando una aplicación es reactivada, puede aparecer con otro display usando el método setCurrent()
- **destroyApp()**
  - libera y destruye todos los recursos usados por la aplicaciones, incluidos los componentes del interfaz de usuario

Estos tres métodos tienen que aparecer siempre.

Aplicación mínima

La estructura básica de un MIDlet es la que se muestra a continuación:

```
import javax.microedition.midlet.*;

public class HelloMIDlet extends MIDlet {
public void startApp() {
}

public void pauseApp() {
}

public void destroyApp(boolean unconditional) {
}
}
```

Apéndice B.

Manual básico de Servlet

Índice

1. Introducción

Los Servlets son la respuesta de la tecnología Java a la programación CGI. Son programas que se ejecutan en un servidor Web y que son capaces de construir páginas Web de forma dinámica. Construir páginas Web al vuelo es útil (y comúnmente usado) por un número de razones:

- La página Web está basada en datos enviados por el usuario.
- Los datos cambian frecuentemente.
- Las páginas Web que usan información desde bases de datos corporativas u otras fuentes.

Los Servlets Java son más eficientes, fáciles de usar, más poderosos y más portables que el CGI tradicional y otras muchas tecnologías del tipo CGI.

En este manual básico, se explica el uso de Servlets como puente en la comunicación, entre una aplicación ejecutada desde un móvil y otra ejecutada desde un ordenador.

2. Kits de desarrollo de Servlets

La creación de un Servlet es un tanto compleja, puesto que es necesario tener una herramienta de desarrollo en Java, que de soporte para la programación de dicho elemento. Los pasos necesarios para crear un servlet son los siguientes:

1. Desarrollar el código del Servlet.
2. Compilar dicho código java. Para evitar que haya errores en el código.
3. Ejecutarlo para depurar posibles errores en tiempo de ejecución
4. Construir el Servlet (Deployment).

Con la mayoría de los entornos de desarrollo para aplicaciones Java, se pueden llevar a cabo los dos primeros pasos descritos, pero no los dos últimos. Existen entornos capaces de realizar todos los pasos descritos como son Eclipse o NetBeans. En este manual se ha utilizado NetBeans para el desarrollo del ejemplo, puesto que esta aplicación ya viene preparada para poder ejecutar los servlets y poder depurar así posibles errores. Esta aplicación se puede descargar de forma gratuita desde http://www.netbeans.org.

Una vez que se ha construido el Servlet, para su posterior ejecución, es necesaria una herramienta que de soporte única y exclusivamente a la ejecución del mismo. Esta aplicación es el servidor Apache Tomcat. La instalación de esta herramienta es muy sencilla y su configuración se puede realizar a través de una página WEB. Cuando se ha terminado el Servlets solo hay que copiarlo a la carpeta de trabajo de éste servidor para que pueda ser ejecutado desde cualquier navegador. Esta herramienta se puede descargar de forma gratuita desde http://tomcat.apache.org.

3. Estructura básica de un Servlet

A continuación se muestra un Servlet básico que maneja peticiones GET. Las peticiones GET son peticiones hechas por el navegador cuando el usuario introduce una URL en la línea de direcciones, sigue un enlace desde una página Web, o rellena un formulario que no especifica

un METHOD. Los Servlets también pueden manejar peticiones POST muy fácilmente, que son generadas cuando alguien crea un formulario HTML que especifica METHOD="POST". Pero en definitiva, la estructura básica de cualquier servlet es como la que se describe abajo:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class SomeServlet extends HttpServlet {
  public void doGet(HttpServletRequest request,
         HttpServletResponse response)throws ServletException, IOException {

    // Use "request" to read incoming HTTP headers (e.g. cookies)
    // and HTML form data (e.g. data the user entered and submitted)

    // Use "response" to specify the HTTP response line and headers
    // (e.g. specifying the content type, setting cookies).

    PrintWriter out = response.getWriter();
    // Use "out" to send content to browser
  }
}
```

Para que se pueda entender el código como un Servlet, una clase debe extender a `HttpServlet` y sobrescribir los métodos *doGet* o *doPost* (o ambos), dependiendo de si los datos están siendo enviados mediante el método GET o POST del protocolo HTTP. Estos métodos toman dos argumentos: un `HttpServletRequest` y un `HttpServletResponse`.

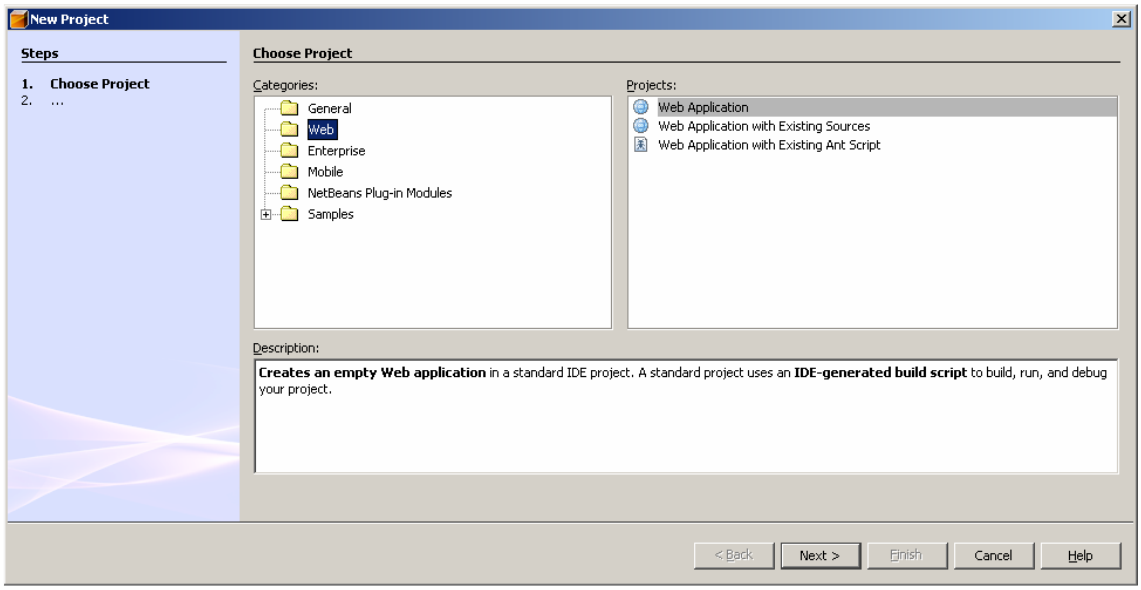
El `HttpServletRequest` tiene métodos que permiten encontrar información entrante como datos de un FORM, cabeceras de petición HTTP, etc. El `HttpServletResponse` tiene métodos que permiten especificar líneas de respuesta HTTP (200, 404, etc.), cabeceras de respuesta (Content-Type, Set-Cookie, etc.), y lo que todavía es más importante, permiten obtener un `PrintWriter` que se usa para enviar el código HTML (generalmente) de vuelta al cliente. Cuando se crean Servlets sencillos, la mayoría del esfuerzo se concentra en sentencias ***println*** que acaban generando la página o código HTML de respuesta.

Como se puede observar en el código mostrado anteriormente, los métodos *doGet* y *doPost* pueden lanzar dos excepciones que son `ServletException` e `IOException`, por este motivo se hace necesario incluirlas en la declaración de estos métodos. También se puede observar los paquetes que se tiene que importar para que el Servlet pueda realizar su función, paquetes como *java.io* (para *PrintWriter*, etc.), *javax.servlet* (para *HttpServlet*, etc.), y *javax.servlet.http* (para *HttpServletRequest* y *HttpServletResponse*).

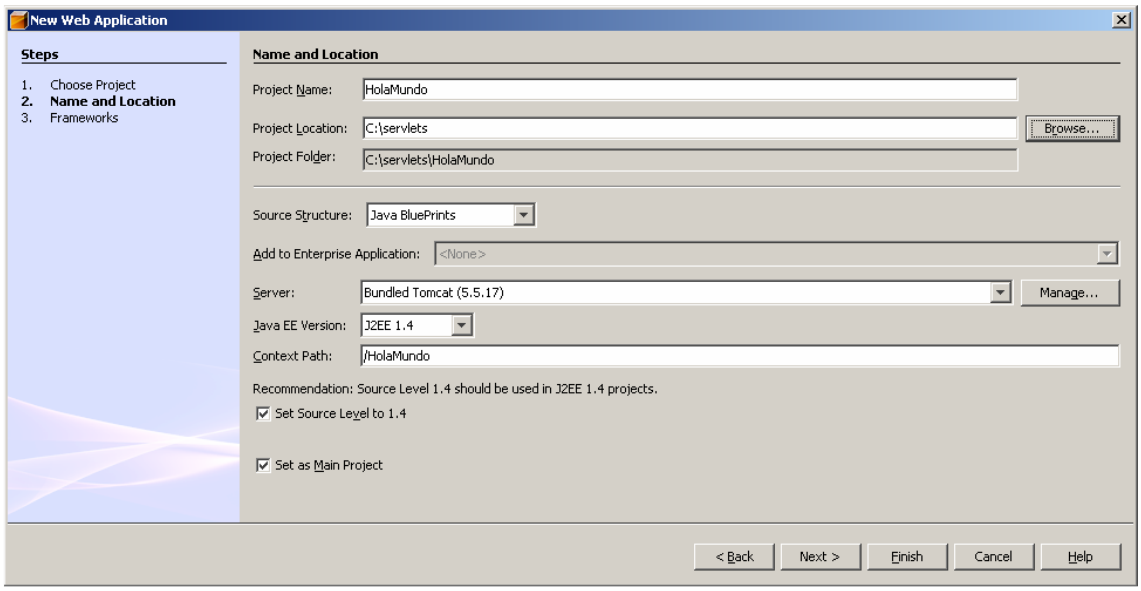
4. Primeros pasos. Un ejemplo sencillo: HolaMundo

El primer paso que se debe realizar es instalar el NetBeans IDE. Una vez instalado, se ejecutará ésta herramienta para crear el Servlet.

Una vez abierto, se selecciona File → New Project , aparecerá la ventaja que se muestra más abajo, y se debe escoger la opción Web → Web Aplication y continuar pulsando el botón Next.

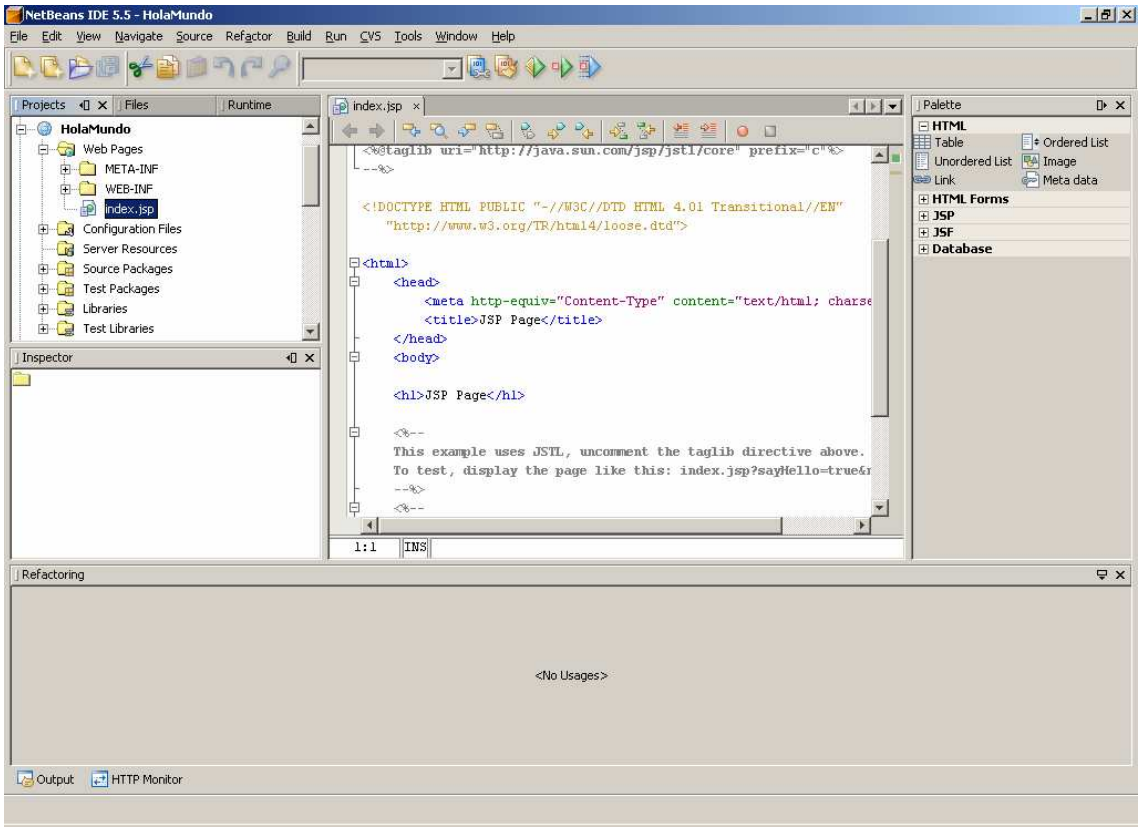


Entonces aparecerá la siguiente ventana:



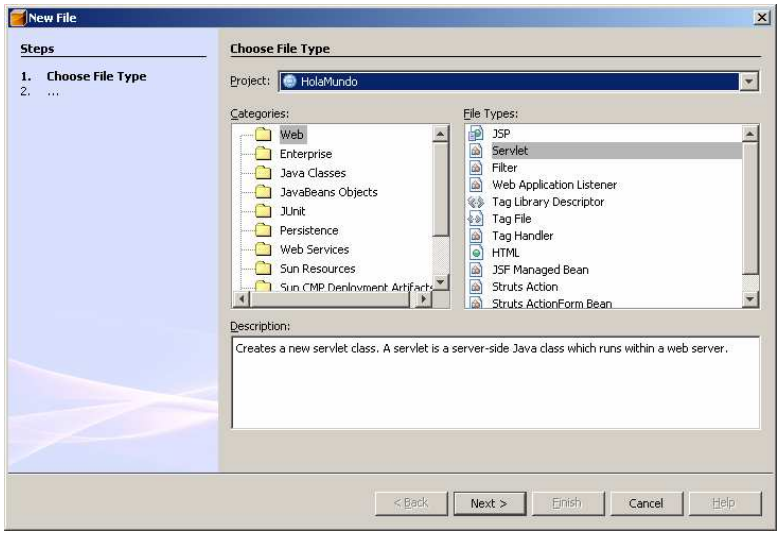
En esta ventana es en donde se tiene que poner el nombre al proyecto y elegir su ubicación. Es importante que en la opción de “Server” aparezca Bundled Tomcat, ya que va a ser este servidor el que sirva para depurar el código del Servlet. Si no apareciese, se tendrá que bajar el plugins correspondiente desde la página de NetBeans e instalarlo. Finalmente comentar la opción “Context Path”, hay que fijarse en su valor, puesto que va ser este el que posteriormente va a servirnos para ejecutar el Servlets una vez construido. Lo demás se puede dejar por defecto y pulsar el botón Finish.

Por pantalla se debe aparecer algo parecido a lo que se muestra en la siguiente imagen.



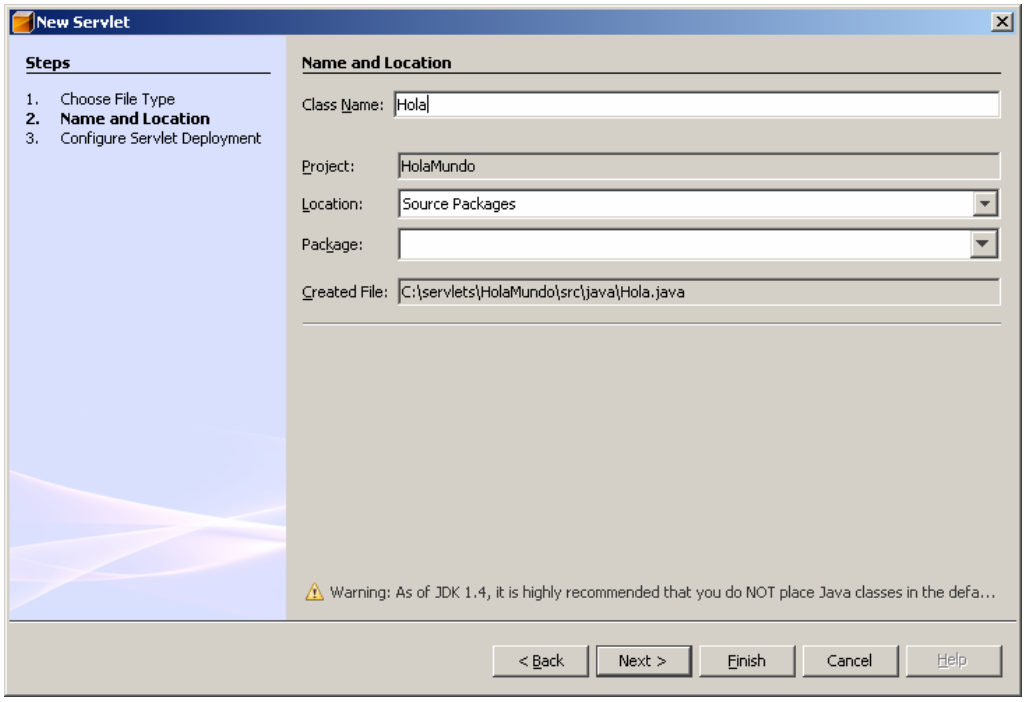
Cuando se crea el proyecto, por defecto viene con una página inicial ya creada en lenguaje JSP, la cual no interesa para realizar este ejemplo, ya que lo que se desea es realizar un Servlets. La manipulación de dicha página o la creación de alguna nueva, en estos momentos esta fuera de lo que se quiere realizar, pero es recomendable que se le eche un vistazo, dentro del proyecto "HolaMundo", a "Configuration Files"→ "Web XML", ya que desde aquí se pueden cambiar muchos detalles de la configuración del proyecto.

Ahora es cuando se puede crear el Servlet. Para ello se tiene que elegir la opción File → New Files y aparecerá la siguiente ventana:

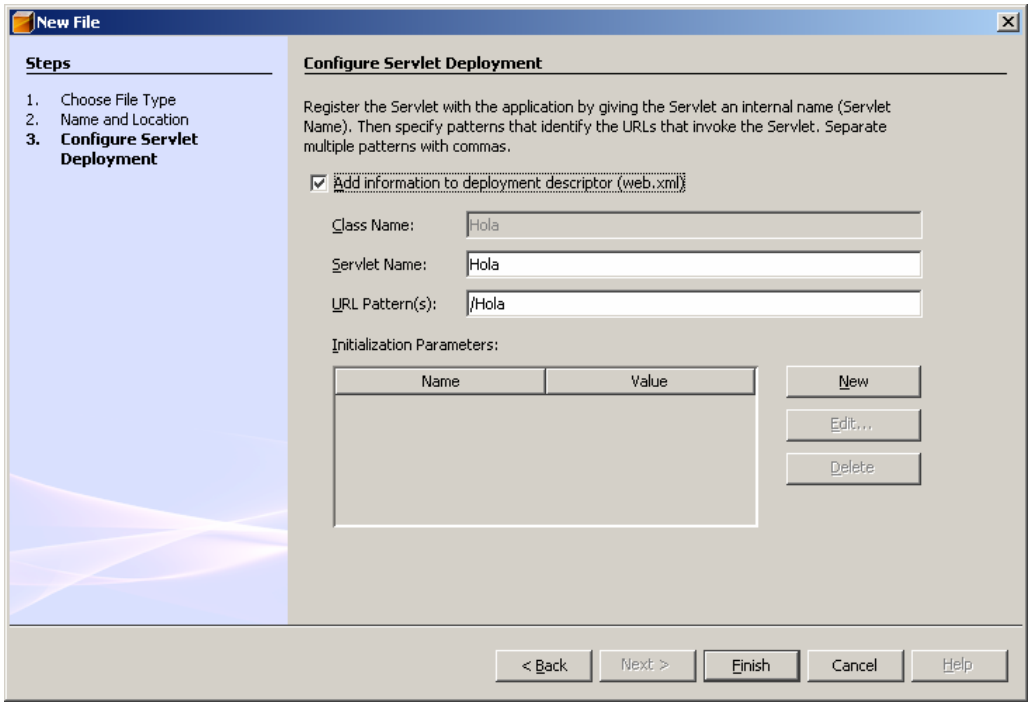


Elegir dentro de la categoría Web el tipo de archivo Servlet. Observar también que se pueden desarrollar páginas en JSP o HTML. Pulsar Next para ir a la siguiente pantalla.

En ésta pantalla, que se muestra abajo, se debe especificar el nombre del Servlet. En este ejemplo se ha llamado Hola.



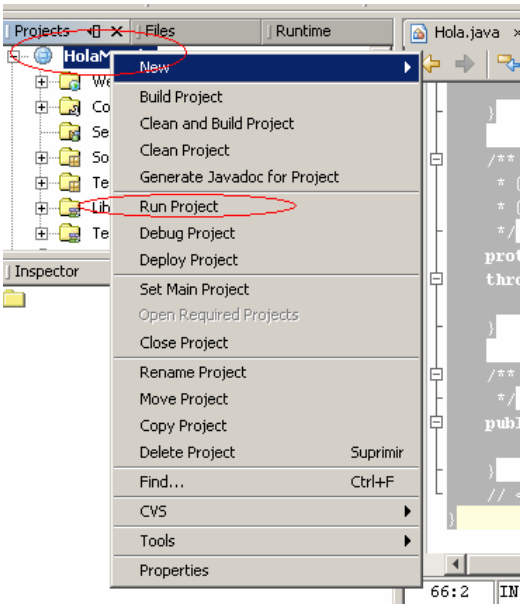
Finalmente se llega hasta ésta última pantalla que se muestra abajo, en la cual se tiene que seleccionar la opción “Add information to …” de esta forma cuando se construya el Servlet definitivo se podrá ejecutar desde cualquier servidor de Servlets como el Apache Tomcat.



Una vez se hayan realizado todos estos pasos antes expuestos, se puede empezar ya a añadir código al Servlet. El código que se debe añadir es el siguiente:

```
import java.io.*;
import java.net.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Hola extends HttpServlet {

   /** Processes requests for both HTTP <code>GET</code> and
   * <code>POST</code> methods.
   * @param request servlet request
   * @param response servlet response
   */
   protected void processRequest(HttpServletRequest request,
         HttpServletResponse response)throws ServletException,
         IOException{

      response.setContentType("text/html;charset=UTF-8");
      PrintWriter out = response.getWriter();

      // TODO output your page here
      out.println("<html>");
      out.println("<head>");
      out.println("<title>Hola mundo</title>");
      out.println("</head>");
      out.println("<body>");
      out.println("<h1>Hola Mundo desde mi servlet => " +
            request.getContextPath () + "</h1>");

      out.println("</body>");
      out.println("</html>");

      out.close();
   }

   // <editor-fold defaultstate="collapsed" desc="HttpServlet methods.
   // Click on the + sign on the left to edit the code.">

   /** Handles the HTTP <code>GET</code> method.
   * @param request servlet request
   * @param response servlet response
   */
   protected void doGet(HttpServletRequest request, HttpServletResponse
           response) throws ServletException, IOException {

      processRequest(request, response);
   }

   /** Handles the HTTP <code>POST</code> method.
   * @param request servlet request
   * @param response servlet response
   */
   protected void doPost(HttpServletRequest request, HttpServletResponse
        response) throws ServletException, IOException {

      processRequest(request, response);
   }

   /** Returns a short description of the servlet.
   */
   public String getServletInfo() {
      return "Short description";
   }
   // </editor-fold>
}
```

Una vez añadido el código y comprobado que no hay errores de compilación (los indica el NetBeans IDE mientras se está escribiendo el código), se tiene que comprobar que dicho código hace lo que se desea, para ello, ejecutar el proyecto desde el propio NetBeans, siguiendo estos pasos:



1º Desde la ventana de proyectos, pulsar con el botón derecho del ratón sobre el proyecto HolaMundo.

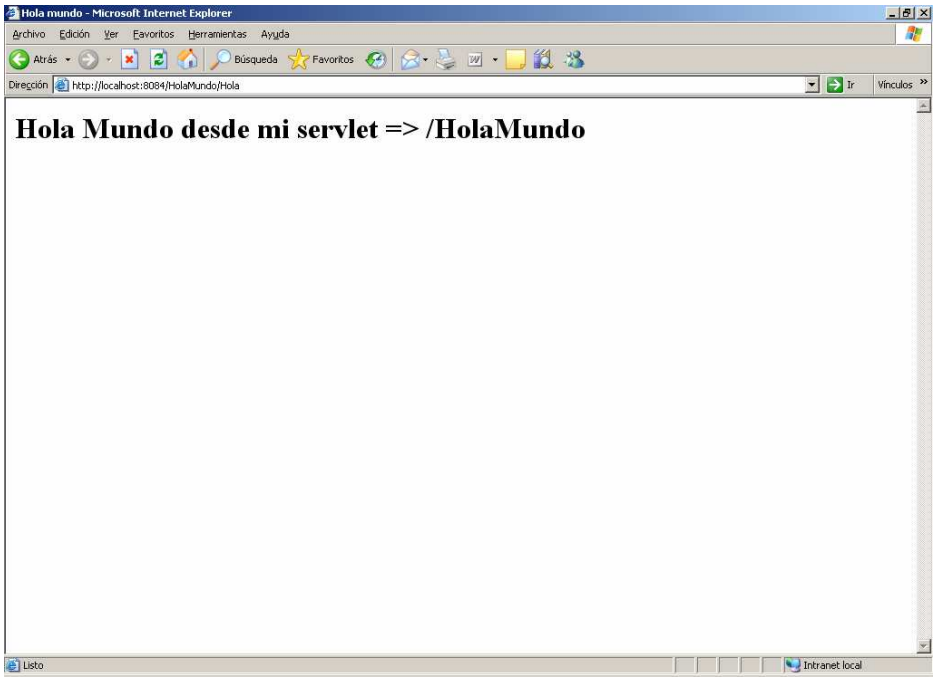
2º Seleccionar la opción “Run Project”.

3º Esperar a que se habrá la ventana del navegador con la página de inicio del proyecto (no se ha modificado luego será la que crea por defecto el NetBeans).

4º Probar el Servlets estableciendo la siguiente URL desde un navegador:

http://localhost:8084/HolaMundo/Hola

Y podremos comprobar que se obtiene el siguiente resultado:



Ahora ya se puede construir el proyecto para exportarlo al servidor de Servlet Apache Tomcat.

5. Aplicación avanzada.

a. Definición de la aplicación

Se desea crear una aplicación para el móvil desde la cual se pueda enviar ciertos comandos a un servidor que ofrece servicio en un PC a través de conexiones por Sockets. Para ello, se utilizará un Servlet como pasarela entre el móvil y el servidor del PC. El Servlet recibirá el comando enviado desde el móvil a través de una petición GET del protocolo http (el comando viene en la propia URL), para posteriormente conectarse al servidor del PC para enviarle dicho comando.

El proceso será como sigue:

1º) Una aplicación que se esta ejecutando en el móvil, manda el comando a través de una URL aprovechando el método GET de HTTP. Ejemplo:

http://[IP_Servidor_Servlets:puerto]/[context_path]/[nombre_servlet]?comando=”acción”

2º) Esta petición HTTP, es recibida por el servidor de Servlet, el cual ejecutará dicho Servlet obteniendo el comando para enviarlo hacia la aplicación servidora.

3º) Dicha aplicación se está ejecutando permanentemente, luego está esperando a recibir un comando desde el Servlet. Cuando este es recibido, cambia el estado del servidor según el comando.

4º) La aplicación envía una respuesta al Servlet con el nuevo estado.

5º) El Servlet le retransmite esa misma información a la aplicación del móvil.

b. Código java del Servlet

El código que se muestra a continuación es el del servlet que se especifica en la definición. Pero para su mejor entendimiento la respuesta del mismo se hace en formato HTML, lo que nos sirve para ejecutarlo desde cualquier navegador en vez desde un dispositivo móvil.

```
import java.io.*;
import java.net.*;

import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.ServletRequest;

public class Hola extends HttpServlet {

   private Socket s;
   private int i,c, longCad;
   private InputStream sIn;
   private OutputStream sOut;
   private String cadena = "Azul";
   private char []entrada;

  protected void processRequest(HttpServletRequest request,
        HttpServletResponse response)throws ServletException, IOException{

      String Comando = request.getParameter("Comando");

      // Abrimos una conexión con depserver en el puerto 4321
      s = new Socket( "localhost",4321 );
```

```
        //Obtenermos el controlador de salida del socket para mandar
        // información al servidor
        sOut = s.getOutputStream();
        longCad = Comando.length();

        for( int i=0; i < longCad; i++ ){
           sOut.write( (int)Comando.charAt( i ) );
        }

        sOut.write((int)-1);

        // Obtenemos un controlador de fichero de entrada del socket y
        // leemos esa entrada
        sIn = s.getInputStream();
        entrada = new char[120];
        i=0;
        while( ( c = sIn.read() ) != 255 ){
           entrada[i]=( (char)c );
           i++;
        }

        String datosRecibidos =new String(entrada);

        //Pagina a mostrar por el servlet
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();

        // TODO output your page here
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet Hola</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Servlet Hola at "+request.getContextPath()+
                    "</h1>");

        out.println("<h2> Realizado por Antonio Guillen Pascual</h2>");
        out.print("<h3>El comando selecionado es: "+Comando+"</h3>");
        out.println("<h3>Los datos recibidos del servidor
                    son:"+datosRecibidos+"</h3>");

        out.println("</body>");
        out.println("</html>");

        out.close();
    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click
    // on the + sign on the left to edit the code.">

    /** Handles the HTTP <code>GET</code> method.
    * @param request servlet request
    * @param response servlet response
    */
    protected void doGet(HttpServletRequest request, HttpServletResponse
              response)throws ServletException, IOException {

        processRequest(request, response);

    }
```

```
    /** Handles the HTTP <code>POST</code> method.
    * @param request servlet request
    * @param response servlet response
    */
    protected void doPost(HttpServletRequest request, HttpServletResponse
            response) throws ServletException, IOException {

         processRequest(request, response);

    }

    /** Returns a short description of the servlet.
    */
    public String getServletInfo() {
         return "Short description";
    }

    // </editor-fold>
  }
```