UNIVERSIDAD DE JAÉN

ESCUELA POLITÉCNICA SUPERIOR DE LINARES

Departamento de Ingeniería de Telecomunicación
Área de Telemática

CURSO 2010/2011

PROYECTO FIN DE CARRERA

DESARROLLO DE UN SERVIDOR PARA EL ENVÍO DE GUÍAS TURÍSTICAS A TRAVÉS DE BLUETOOTH

TITULACION: Ingeniería Técnica de Telecomunicación
ESPECIALIDAD: Telemática
AUTOR: José Fernando Aranda Jiménez
TUTOR: José Ángel Fernández Prieto

Linares, Septiembre, 2011

ÍNDICE

MEMORIA

Página

ESTUDIO ECONÓMICO

1. INTRODUCCIÓN

Cuando un turista llega a una ciudad, está interesado en conocer los monumentos, museos e información de interés turístico del lugar. Existen multitud de guías turísticas disponibles al público, sin embargo puede resultar engorroso ir cargado con tanto papel. En el presente documento se propone una alternativa que resultará más cómoda al usuario.

En la actualidad todo el mundo dispone de un terminal móvil, ya sea un teléfono o una PDA, y existen multitud de aplicaciones para estos terminales. Se propone el envío de guías turísticas a través de Bluetooth a dichos terminales. El usuario final tan sólo tendrá que aceptar la transferencia del archivo de guía turística e instalarla. La instalación de estas pequeñas aplicaciones es muy sencilla, pues suele ser automática con sólo aceptarla en el terminal. De este modo el turista dispondrá una guía turística en su teléfono móvil, lo cual resulta muy cómodo.

Pero, ¿cómo recibe el turista dicha guía? Se instalarán servidores Bluetooth en las estaciones de autobuses o trenes, aeropuertos o puertos marítimos en su caso. De este modo recibirán la guía turística de forma gratuita, pues una conexión Bluetooth no requiere coste alguno. Además hoy día todos los terminales disponen de la tecnología Bluetooth. El servidor estará permanentemente buscando nuevos dispositivos Bluetooth para enviarles la guía turística, evitando reenviarla dos veces al mismo dispositivo.

Es por todo ello, que se ha decidido diseñar una aplicación que funcione como guía turística para terminales móviles, y un servidor basado en la tecnología Bluetooth para el envío de dicha aplicación.

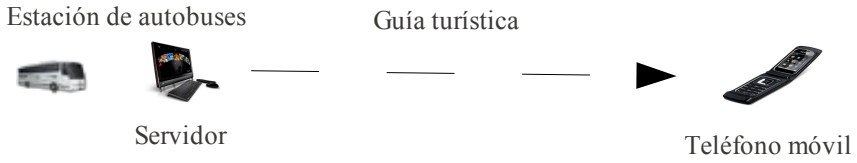


Figura 1. Esquema del proyecto

2. OBJETIVOS

El principal objetivo de este proyecto es desarrollar un servidor basado en la tecnología Bluetooth para el envío de guías turísticas a dispositivos móviles, tales cómo teléfonos móviles o PDA. La principal ventaja de esta tecnología es que no tiene coste alguno para el usuario final, siendo apropiado para administraciones públicas que deseen promover el turismo en una zona determinada, cómo por ejemplo una ciudad. Además, tan sólo es necesario que el terminal móvil disponga de las tecnologías Bluetooth y Java, algo muy extendido en la actualidad. Se pretende que el servidor envíe la aplicación de guía turística a todos los dispositivos móviles que encuentre, evitando reenviarla más de una vez al mismo dispositivo. El usuario tan sólo tendrá que aceptar la conexión y el envío de la aplicación Java (J2ME), para posteriormente instalarla en su terminal.

El segundo objetivo es implementar una guía turística para los anteriormente mencionados terminales móviles (teléfonos o PDA). Dicha guía será sobre la ciudad de Linares y mostrará los lugares de interés turístico así como los principales servicios de la ciudad.

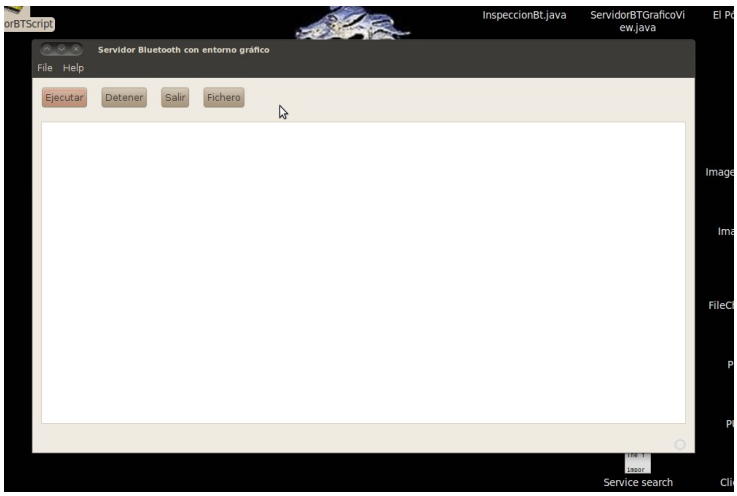


Figura 2. Servidor Bluetooth

Figura 1. Menú Principal de la guía turística

3. ANTECEDENTES

Es frecuente ver servidores web que ofrecen aplicaciones J2ME que requieren una conexión a Internet. En cambio, para conectarse a un servidor Bluetooth tan sólo se requiere un dispositivo provisto de tal tecnología, tan extendida en la actualidad. En los tiempos que corren están surgiendo multitud de servidores de muchos tipos y la tecnología Bluetooth está creciendo a pasos agigantados.

Hoy día Internet lo es todo y la telefonía móvil no podía quedarse atrás en un mundo sin cables. Desde la aparición de la tecnología Bluetooth, todos quieren estar conectados a cosas cercanas como móviles, PDA, ordenadores, impresoras, etc. Bluetooth posibilita otra forma de interconectarse y comunicarse en distancias cortas vía radiofrecuencia. Esta, ya no tan nueva, tecnología permite crear Redes de Área Personal (PAN o piconet) entre dispositivos cercanos entre sí.

Técnicamente, una red Bluetooth se denomina piconet y puede consistir en un dispositivo maestro y hasta un máximo de siete dispositivos esclavos. La red se puede extender hasta ocho si el dispositivo maestro actúa como esclavo en otra red. Sin embargo, todo ese hardware y detalles de los protocolos vía radio son transparentes al desarrollador usando el API de Java para Bluetooth.

El API de Java para la Tecnología inalámbrica Bluetooth es JSR-82. Esta especificación ofrece dos opciones completas y separadas. Estas API (que veremos más adelante) son:

- API de Java para Bluetooth
- API de Java para OBEX

Mientras el primero cubre exclusivamente comunicaciones vía Bluetooth, el segundo se centra en el protocolo OBEX. Este protocolo se puede implementar sobre una PAN (Red de

Área Personal) de Bluetooth, redes IP y comunicaciones por infrarrojos.

4. Estado del arte

La tecnología Bluetooth está en creciente auge en la actualidad, y se prevé un amplio desarrollo en el futuro. Además para establecer la conexión y realizar transferencias de ficheros no requiere coste económico alguno por parte del usuario final. Tan sólo es requisito indispensable tener activado el dispositivo Bluetooth y estar en el estado visible para poder ser descubierto. Por todo ello se ha decidido diseñar un servidor basado en la tecnología Bluetooth.

Para desarrollar esta tecnología en el servidor Bluetooth se ha escogido el lenguaje de programación Java en su versión J2SE, y en concreto hemos utilizado el API JSR-82 para la tecnología Bluetooth junto con la librería BlueCove de libre distribución. La elección ha sido Java porque está ampliamente desarrollado en la comunidad del software libre, no siendo necesaria por tanto licencia alguna. El lenguaje C# sería otra posibilidad pues ofrece ventajas similares cómo puede ser la portabilidad, aunque no está tan desarrollado como Java en su versión libre.

En cuanto a la aplicación de la guía turística, se ha elegido Java por las mismas razones. Para terminales móviles, Java dispone de su versión J2ME (Java 2 Micro Edition). La mayoría de los terminales móviles soportan esta tecnología.

5. DESCRIPCIÓN DEL PROYECTO

Se ha desarrollado un servidor para el envío de ficheros a través de Bluetooth. Este servidor se ha implementado haciendo uso del lenguaje de programación Java en su versión J2SE. Adicionalmente se ha implementado una aplicación J2ME de guía turística. A continuación se describe tanto la aplicación servidora como la guía turística.

5.1. Descripción del servidor Bluetooth

El servidor se ha desarrollado con la tecnología J2SE (Java 2 Standard Edition). Consiste en una colección de APIs del lenguaje de programación Java. Para poder implementar las API de la especificación JSR-82, se ha utilizado la librería BlueCove de libre distribución.

El servidor está compuesto de cuatro botones:

- Ejecutar: Sirve para lanzar la ejecución del servidor.
- Detener: Detiene la ejecución del servidor.
- Salir: Cierra el programa.
- Fichero: Se utiliza para seleccionar el fichero a enviar.

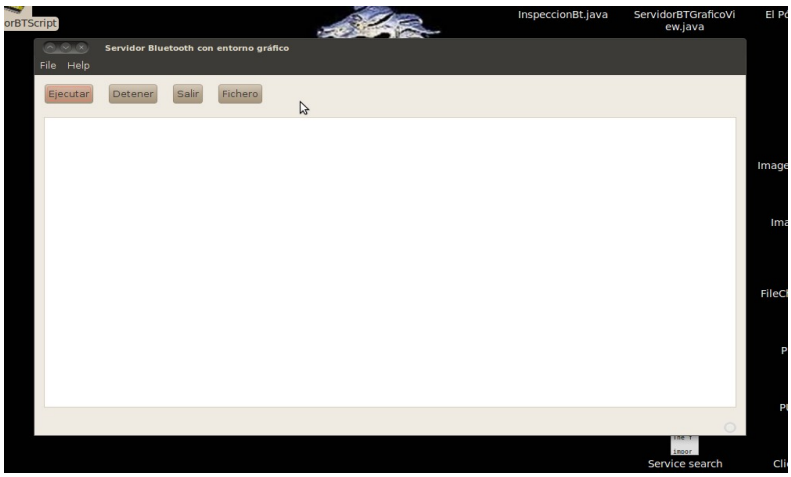


Figura 3. Servidor Bluetooth

El programa se estructura en siete ficheros. La funcionalidad del proyecto está implementada en sólo tres de ellos (ServidorBTurista.java, InspeccionBt.java, SesionObex.java). Los otros cuatro sirven para conformar el entorno gráfico del programa.

ServidorBTurista.java es el fichero central, el cual hace llamadas al resto. En este fichero se implementa la inicialización del dispositivo Bluetooth creando el manejador del mismo. En el fichero *InspeccionBt.java* se encuentra implementado todo el proceso de búsqueda de dispositivos así como la búsqueda de servicios Bluetooth. Por último, el fichero *SesionObex.java* implementa la transferencia de ficheros a través del protocolo obex. Cabe mencionar que el fichero *ServidorBTGraficoApp.java* implementa la funcionalidad de los cuatro botones del servidor.

Los problemas que se han encontrado han sido bajo el sistema operativo *Windows 7*. Existe un problema con la torre de protocolos de Bluetooth, no permitiendo el envío de ficheros a algunos terminales. Bajo el entorno *Linux* o en *Windows XP* funciona correctamente.

5.1.1. OBEX

El IrOBEX (Infrared Object Exchange protocol) está definido como una alternativa al Protocolo de Transporte de Hipertexto (HTTP) para dispositivos con restricciones. IrOBEX se ha hecho más popular desde que el SIG Bluetooth adquirió el protocolo para sus dispositivos. Cuando el protocolo es usado para dispositivos Bluetooth, el nombre se reduce a OBEX. El Bluetooth SIG definió OBEX como uno de los protocolos de la pila de protocolos Bluetooth. Para facilitar la construcción de nuevos perfiles, el Bluetooth SIG definió GOEP para ser el perfil que define como trabaja OBEX sin el entorno Bluetooth.

Se hablará de todo ello más detenidamente en secciones posteriores.

5.2. Descripción de la aplicación GuiaTurismo.jar

Se ha implementado una guía turística para los anteriormente mencionados terminales móviles (teléfonos o PDA). Dicha guía será sobre la ciudad de Linares y mostrará los lugares de interés turístico así como los principales servicios de la ciudad. Esta aplicación se ha desarrollado usando la clase "List". Esta clase admite dos tipos de listas: MÚLTIPLE (selección mútiple) o EXPLÍCITAS (selección de un solo objeto); y dentro de las segundas se pueden definir las denominadas listas IMPLÍCITAS, las cuales se usan para definir los menús. Consta de un menú principal:

- Museos
- Transportes
- Alojamiento
- Monumentos
- Ciudad de Linares

Figura 4. Menú Principal

Seleccionando cada uno de los menús, se despliegan los submenús correspondientes. En la opción seleccionada se muestra una imagen e información referente a dicha opción. Esto se ha realizado con la clase "Alert". Ésta clase es usada para mostrar información, en forma de texto y/o imágenes.

Están disponibles cuatro comandos:

- Inicio: Aparece en la pantalla de bienvenida y sirve para comenzar a usar la guía turística.
- Salir: Aparece en la pantalla del menu principal y sirve para cerrar la aplicación.
- Seleccionar: Permite acceder a una de las opciones que ofrecen los menús.

- Volver: Sirve para retroceder a la pantalla anterior.

Esta aplicación se compone de dos ficheros:

- guiaTurismoLinares.java: Es un MIDlet donde se implementan todos los menús junto con la información referente a la opción que se seleccione.
- Imagen.java: Hereda de la clase Canvas para ofrecer un mapa de Linares al usuario.

6. Descripción de la tecnología Bluetooth

La tecnología Bluetooth® es la especificación que define un estándar global de comunicaciones inalámbricas vía radiofrecuencia de corto alcance recogida por el grupo de trabajo 802.15.1 del IEEE (redes inalámbricas de área personal, grupo 1 – WPAN/Bluetooth).

El nombre Bluetooth es de origen danés, referido al rey del siglo X Harald Blatand, traducido como Harold Bluetooth. Fue un rey conocido por unificar las tribus en guerra de Noruega, Suecia y Dinamarca. La elección de este nombre es debido a que del mismo modo, esta tecnología pretende interconectar diferentes tipos de dispositivos (ordenadores, teléfonos móviles, etc). El logo de Bluetooth combina la representación de las runas nórdicas Hagalaz (transcrito por 'H') y Berkana (transcrito por 'B'):

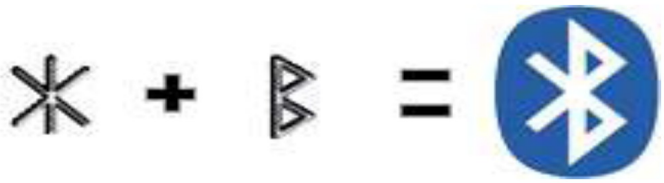


Figura 5. Logo Bluetooth

El comienzo de está tecnología se debió a un estudio de viabilidad de una interfaz de radio de baja potencia y bajo coste entre teléfonos móviles y otros accesorios con el objetivo de eliminar los cables. Este estudio fue llevado a cabo en el año 1994 por la compañía global de telecomunicaciones Ericsson Mobile Communications con sede en Suecia.

El trabajo de Ericsson en este área atrajo la atención de otras compañías. Las empresas IBM, Intel, Nokia y Toshiba, junto con Ericsson, decidieron formar en Febrero de 1998 un grupo especial de investigación denominado SIG (Special Interest Group) Bluetooth. A finales de ese mismo año el SIG incluyó a su cuadrigentésimo miembro. En la actualidad el grupo tiene más de 14.000 socios.

Hoy día los teléfonos móviles están muy extendidos, no sólo en España, sino en todo el

mundo. Es por ello que han surgido multitud de aplicaciones para terminales móviles, ya sean teléfonos, PDA u otros. También se siguen desarrollando multitud de sistemas operativos para estos terminales, tanto propietarios como de código abierto (Symbian, Android, Windows Mobile, etc). Sin embargo la mayoría de ellos utilizan la tecnología Java para desarrollar sus aplicaciones. Esto se debe a la portabilidad que ofrece Java, a la accesibilidad de sus librerías, a que es orientado a objetos y a lo extendida que está esta tecnología.

Es por todo esto por lo que se ha decidido realizar una aplicación turística basada en Java (más concretamente en J2ME – Java 2 Mobile Edition) para terminales móviles. Pero el proyecto quedaría incompleto si no se desarrollara un servidor para el envío de la misma. Un servidor construido utilizando la tecnología Bluetooth permite a cualquier usuario con un terminal móvil que incorpore dicha tecnología, la posibilidad de recibir cualquier tipo de ficheros en su terminal con un bajo consumo de energía y sin coste alguno para establecer la conexión.

Existe, sin embargo, la limitación de la distancia de cobertura, restringida al alcance máximo del dispositivo Bluetooth utilizado en el servidor (normalmente de unos 200m). Aunque ello se solucionaría situando dicho servidor en lugares estratégicos, tales como en los accesos a estaciones, aeropuertos o puertos marítimos.

6.1. Principales protocolos de Bluetooth

Para iniciar una conexión saliente en un dispositivo Bluetooth es necesario especificar un protocolo de transporte (TCP/IP en el caso del acceso a internet, p. e.) así como elegir un dispositivo destinatario y un número de puerto. Cada dispositivo Bluetooth se identifica de forma unívoca mediante un número de 48 bits, denominado *dirección Bluetooth,* similar al número MAC de Ethernet. Para establecer una conexión Bluetooth es necesario conocer la dirección del otro dispositivo. Para referirnos a una dirección determinada usamos los *nombres amigables* asociados a cada dispositivo.

Bluetooth se compone de muchos protocolos de transporte, aunque consideraremos sólo dos. Los más usados son RFCOMM y L2CAP:

- RFCOMM (Radio Frecuency Communications)

Es un protocolo de transporte de propósito general que se usa para emular una conexión mediante puertos serie (RS-232). Ofrece una fiabilidad similar a la de TCP, pero a diferencia del protocolo TCP (soporta 65535 puertos abiertos en una sola máquina), RFCOMM sólo permite elegir entre 30 puertos.

Un dispositivo Bluetooth puede soportar uno o más perfiles, aunque nos centraremos en los dos más usados. El perfil SPP define los requerimientos necesarios en los dispositivos Bluetooth para emular las conexiones serie por cable usando RFCOMM entre dos dispositivos. El *Generic Object Profile* (GOEP) es un perfil abstracto con el cual se concreta el uso en cada caso de los perfiles que se pueden construir. Esos perfiles son los que usan OBEX[1] (e.g. btgoep://XXXXXXXXXXXX:XX).

- L2CAP

1. Hablaremos de OBEX más adelante

El Logical Link Control And Adaption Protocol es un protocolo basado en datagrama que puede ser configurado en diferentes niveles de fiabilidad. Por defecto el tamaño máximo de los paquetes es de 672 bytes pudiendo llegar hasta los 65.535 bytes después de realizar la conexión. Este protocolo es similar a UDP, aunque la principal diferencia estriba en que los paquetes llegan siempre en orden. Además permite varios protocolos de alto nivel o aplicaciones para usar comunicaciones por Bluetooth. Los protocolos SDP (Service Discovery Protocol) y RFCOMM son dos de esos protocolos de alto nivel. El SDP define el servicio de descripción (ServiceDescription).

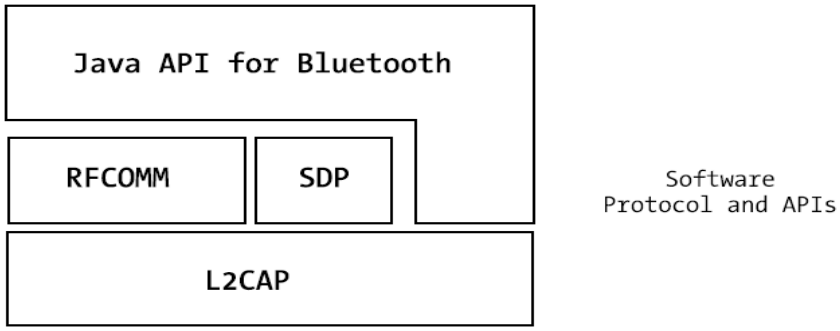


Figura 6. Protocolos software y API

7. Descripción de la tecnología Java

Para el desarrollo del servidor y de la aplicación de guía turística se utilizado el lenguaje Java en sus versiones J2SE y J2ME. Y se ha utilizado el entorno de desarrollo Netbeans IDE 6.9.1. Para implementar las API de Bluetooth se ha utilizado la librería bluecove de libre distribución.

A continuación se verá en más detalle la estructura de Java así como las diferentes clases y métodos que se han utilizado.

7.1. Estructuración de Java (J2ME y JABWT)

La comunidad de Java ha desarrollado una estandarización de un API, el denominado JSR-82 (Java Specification Requests) para Bluetooth. La especificación JSR-82 también es conocida como "Java API for Bluetooth wireless technology" (JABWT). Fue desarrollado por un grupo cuyos miembros representaban a veintiuna compañías, en Diciembre del 2000.

Cuando se usa por primera vez el lenguaje de programación Java, probablemente se comience por la versión J2SE (Java 2 Standard Edition). Ésta edición es el núcleo de herramientas y API usadas para construir las applets de Java, aplicaciones web y otras aplicaciones. En el mundo de las empresas, aún más en las grandes empresas, la situación es diferente. Se suelen usar grandes redes locales y servidores distribuidos. En estos casos se usa la versión J2EE (Java 2 Enterprise Edition).

Por último resta hablar de J2ME (Java 2 Micro Edition). J2ME es un subconjunto de J2SE que soporta un conjunto mínimo de características de Java orientadas a dispositivos con restricciones de memoria y de potencia de procesador. Esta relación se puede observar en la siguiente figura,

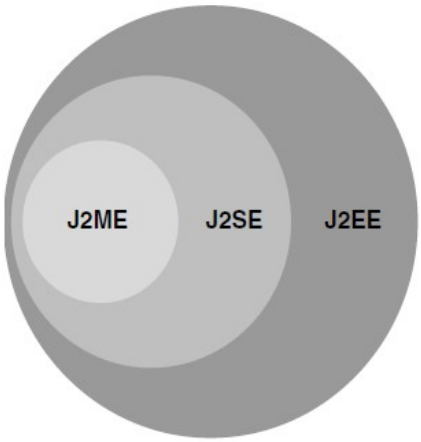


Figura 7. Versiones Java

7.2. Estructura de las aplicaciones J2ME

J2ME se sustenta en dos bloques principales: la configuración y el perfil. Las aplicaciones J2ME se estructuran en MIDP y CLDC. El Mobile Information Device Profile (MIDP) es un elemento clave de la Plataforma Java 2. Cuando se combina con el Connected Limited Device Configuration (CLDC), MIDP provee un entorno estándar de Java en tiempo de ejecución para los dispositivos móviles más populares de hoy día.

CLDC y MIDP proveen el núcleo de la funcionalidad de la aplicación requerida para teléfonos móviles, en forma de un entorno estándar de Java en tiempo de ejecución y una variada colección de API de Java.

J2ME presenta dos configuraciones: CLDC y CDC. La primera se usa para terminales con limitaciones de recursos. CDC se usa para dispositivos con más potencia. Parte de CLDC es un subconjunto de CDC, lo cual facilita la portabilidad.

Las aplicaciones MIDP se denominan MIDlets, las cuales pueden utilizar tanto las facilidades de MIDP como las API que hereda de CLDC, pero nunca acceden al sistema operativo subyacente. Un MIDlet consiste en una clase Java, como mínimo, derivada de la clase abstracta MIDP y que se ejecuta en un entorno de ejecución dentro de la Máquina Virtual de Java (JVM), la cual provee un ciclo de vida bien definido controlado mediante métodos de la clase MIDlet que se debe implementar.

Ciclo de vida de un MIDlet

El software encargado de gestionar los MIDlets es el Gestor de Aplicaciones o AMS (Application Management System). Nos referiremos a él como AMS. Se trata de un software de Java residente en el terminal y permite ejecutar, pausar o destruir una aplicación. Se encarga de dos funciones básicas:

- Gestionar el ciclo de vida de un MIDlet,
- Controlar los estados del MIDlet mientras éste está en ejecución.

El ciclo de vida de un MIDlet pasa por cinco fases: descubrimiento, instalación, ejecución, actualización y borrado. El AMS es el encargado de gestionar cada una de estas fases.

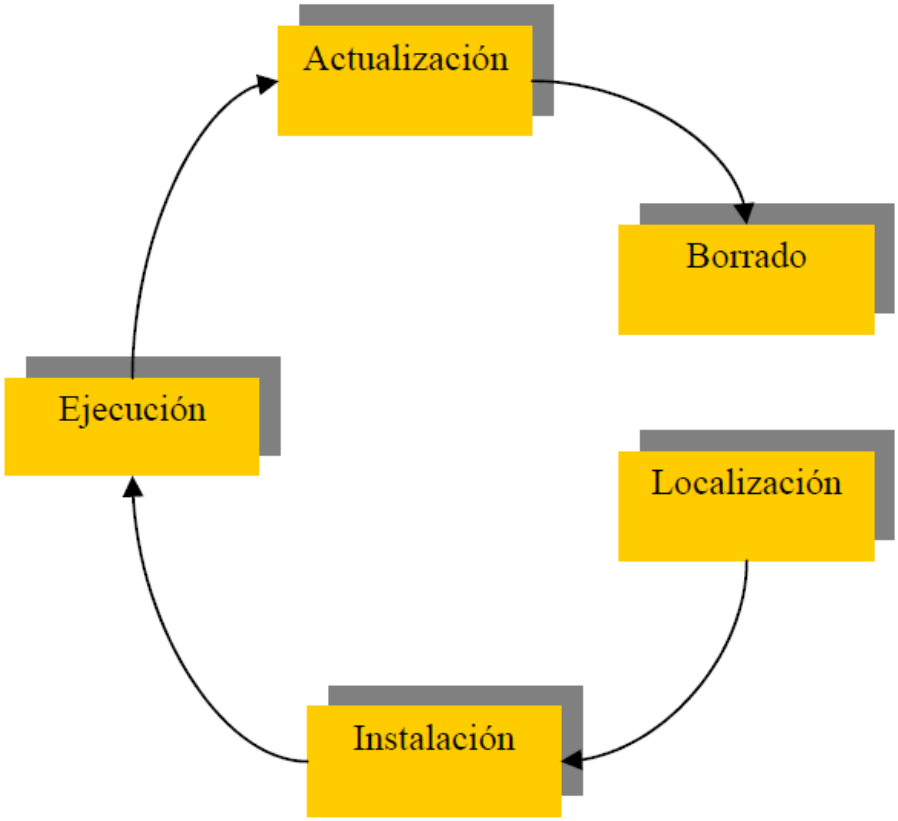


Figura 8. Ciclo de vida de un MIDlet

1. Descubrimiento: Esta es la fase previa a la instalación del MIDlet y es donde seleccionamos a través del gestor de aplicaciones la aplicación a descargar.
2. Instalación: Aquí comienza el proceso de instalación. El gestor de aplicaciones informa tanto de la evolución de la misma, como de los errores que se produzcan.

3. Ejecución: Mediante el gestor de aplicaciones será posible la ejecución de los MIDlets. El AMS tiene la función de gestionar los estados del MIDlet en función de los eventos que se produzcan.
4. Actualización: El AMS debe ser capaz de detectar si el MIDlet descargado es una actualización de un MIDlet ya instalado.
5. Borrado: Aquí el AMS es el encargado de eliminar el MIDlet seleccionado del dispositivo.

Estados de un MIDlet en fase de ejecución

El AMS es también el encargado de controlar los estados de un MIDlet durante su ejecución. Durante ésta, el MIDlet es cargado en la memoria del terminal y es aquí donde puede transitar entre tres estados diferentes: Activo, en pausa o destruido. Cuando un MIDlet inicia su ejecución está en el estado "Activo". Si se produce un evento externo a la aplicación (una llamada por ejemplo), el gestor de aplicaciones interrumpe la ejecución del MIDlet, sin que se vea afectada, y lo pasaría al estado de "Pausa" para atender al evento. Una vez se termine de trabajar con él y se decide salir de la aplicación, se pasa al estado de "Destruido", siendo eliminada de la memoria volátil del terminal.

7.3. Máquina Virtual de Java

La Máquina Virtual de Java (JVM) es un programa encargado de interpretar código intermedio (bytecode) de los programas Java precompilados a código máquina ejecutable por la plataforma, efectuar llamadas pertinentes al sistema operativo subyacente y observar las reglas de seguridad y corrección de código definidas para el lenguaje Java.

De esta forma la JVM proporciona a los programas independencia de la plataforma con respecto al hardware y al sistema operativo subyacentes. Las implementaciones tradicionales de JVM son muy pesadas en cuanto a memoria ocupada y requerimientos computacionales. Por ello, J2ME define varias JVM de referencia adecuadas a dispositivos electrónicos con limitaciones, como teléfonos móviles, consiguiendo una implementación menos exigente.

Sabemos que existen dos configuraciones, CLDC y CDC. Cada una requiere su propia máquina virtual. Tenemos por un lado la KVM para CLDC y por otro la CVM para CDC. Veamos una breve descripción de cada una de ellas:

- KVM (Kilobyte Virtual Machine):

Es la máquina virtual más pequeña desarrollada por Sun. Se trata de una implementación reducida y orientada a dispositivos con bajas prestaciones. Posee una carga de memoria de entre los 40Kb y 80Kb, es de alta portabilidad y modulable. Por su baja capacidad de memoria, ésta posee limitaciones; por ejemplo, no existen los tipos *double* y *float*. El verificador estándar de clases de Java es demasiado grande para la KVM. Por esta razón los dispositivos que usen CLDC introducen un algoritmo de verificación de clases.

- CVM (Compact Virtual Machine)

Es la máquina virtual de referencia para CDC y soporta las mismas características que la Máquina Virtual de J2SE. Está orientada a dispositivos electrónicos con procesadores de 32

bits de gama alta y 2 Mb o más de memoria RAM.

Preverificación de clases

A continuación se habla de un nuevo paso en la construcción del programa a ejecutar: preverificación. Antes de empaquetar nuestro MIDlet es necesario realizar un proceso de preverificación de las clases Java. En esta fase se realiza un examen del código para ver que no viola ninguna restricción de seguridad de la plataforma J2ME.

Debido a la escasa memoria en los dispositivos pequeños, MIDP (en realidad CLDC) especifica que la verificación de los bytecode debe hacerse en dos partes. En algún momento, para apagar el dispositivo, se lleva a cabo una primera etapa de preverificación. En una segunda etapa, al encender el dispositivo, sólo es necesario hacer un proceso ligero de verificación antes de cargar las clases. En esta segunda etapa si un fichero de clases no ha sido preverificado, este se descarta.

En la implementación del MIDP, el J2ME Wireless Toolkit, que se ha usado en el entorno de desarrollo, existe una herramienta llamada *preverificación* que realiza el primer paso.

7.4. El API de Java para Bluetooth

Este API está definido en el paquete javax.bluetooth. Es posible comunicarse con el software de la pila local de los dispositivos, por ejemplo a través de los cuatro siguientes métodos:

- String getBluetoothAddress();
- DeviceClass getDeviceClass();
- DiscoveryAgent getDiscoveryAgent();
- String getFriendlyName();

El primer método (getBluetoothAdress()) devuelve la dirección Bluetooth del dispositivo. getDeviceClass() devuelve el tipo de dispositivo. getDiscoverable() nos dice si el dispositivo está visible para ser descubierto. getDiscoveryAgent() permite obtener un array de los dispositivos descubiertos. GetFriendlyName() devuelve un nombre fácilmente comprensible del dispositivo.

El API de Bluetooth recoge el concepto de agente de descubrimiento (DiscoveryAgent). Se denomina "agente" porque trabaja de forma independiente. Algún tiempo después de ejecutarlo podemos instanciarlo y comprobar su estado o solicitarle un nuevo proceso de descubrimiento. Para instanciarlo se puede escribir el siguiente código:

```
DiscoveryAgent myDa = LocalDevice.getInstance().getDiscoveryAgent();
```

Cuando se instancia un DiscoveryAgent, se puede indicar explícitamente el comienzo de la búsqueda de dispositivos con este método:

```
boolean startInquiry (int accessCode, DiscoveryListener listener);
```

El parámetro "accessCode" indica el tipo de búsqueda (o indagación: inquiry), y ésta puede ser DiscoveryAgent.GIAC (General Inquiry Access Code) o DiscoveryAgent.LIAC (Limited

Inquiry Access Code). Estos códigos se encuentran especificados por el "Bluetooth Assigned Numbers" en *http://www.bluetooth.org/Technical/AssignedNumbers/home.htm*. Se puede encontrar más información en el enlace:

http://www.palowireless.com/infotooth/tutorial/k1_gap.asp#Idle%20Mode%20Procedures

El método startInquiry() comenzará la búsqueda de dispositivos. Con cada dispositivo encontrado, el "listener" que se haya implementado recibirá una notificación. El listener deberá implementar el interfaz "DiscoveryListener". Este interfaz dispone de cuatro métodos, pero sólo dos son importantes para la búsqueda de dispositivos:

- void deviceDiscovered(RemoteDevice btDevice, DeviceClass cod);
- void inquiryCompleted(int discType);

La notificación del método deviceDiscovered() se llama cuando un dispositivo es descubierto durante la búsqueda. Al método inquiryCompleted() se llama cuando el proceso de búsqueda se ha completado. Esto puede deberse a que el periodo de espera (viene determinado por la implementación) se ha agotado o cuando el proceso de búsqueda se ha detenido.

También es posible detener el proceso de inspección de dispositivos con el método

```
public boolean cancelInquiry(DiscoveryListener listener);
```

Una vez se ha encontrado un dispositivo, se pueden buscar los servicios que soporta. Es el momento para usar el método searchServices() en el "DiscoveryAgent":

```
public int searchServices(int[] attrSet, UUID[] uuidSet, RemoteDevice btDev,
DiscoveryListener discListener) throws BluetoothStateException
```

El parámetro uuidSet indica el número UUID específico de cada servicio en el que se está interesado en descubrir. El UUID es un identificador único, universal e inmutable y representa

un valor de 128 bits. El attrSet indica la hoja de registros de los atributos que se pueden manejar en la búsqueda. “discListener” es un DiscoveryListener que el programador implementa.

Para cancelar la búsqueda de servicios se utiliza el siguiente método,

```
public boolean cancelServiceSearch(int transID);
```

El parámetro “transID” indica el identificador de la transacción que se desea cancelar.

7.5. El API de Java para OBEX

OBEX u Object Exchange, es un protocolo diseñado para el intercambio de información en formato vCalendar y vCard, para almacenar como contactos en el terminal correspondiente. Es un protocolo de pregunta/respuesta, heredado de HTTP. OBEX no es un protocolo asociado exclusivamente a Infrarrojos. Por ejemplo, OBEX puede trabajar sobre TCP/IP tan bien como sobre RFCOMM (Bluetooth).

Todos los API de Java para OBEX se encuentran en el paquete javax.obex y se encuentra especificado en el JSR-82. Una sesión OBEX consiste básicamente en una serie de clientes haciendo peticiones al servidor correspondiente. Al igual que en HTTP, cada solicitud y respuesta pueden contener un conjunto de cabeceras y un cuerpo. El interfaz ClientSession del API de OBEX tiene métodos que facilitan el envío de solicitudes del lado del cliente. Las operaciones GET y PUT posibilitan múltiples peticiones y son las más comúnmente usadas para transferir objetos binarios pesados. Dependiendo de los requerimientos de la aplicación hay al menos dos caminos para comunicarse mediante el API de OBEX:

- Mediante las cabeceras de OBEX,
- Mediante las operaciones PUT o GET.

Comunicaciones usando cabeceras

Las cabeceras de OBEX están predefinidas en el interfaz java.obex.HeaderSet (forma parte del JSR-82). Al usar este interfaz se pueden establecer valores definidos por el usuario. Para crear cabeceras se usa el siguiente método:

```
void setHeader(int headerID, java.lang.Object headerValue);
```

Para obtener el valor de una cabecera determinada se puede usar:

```
Object getHeader(int headerID);
```

Como ejemplo del servidor tratado en este documento, hemos usado la siguiente secuencia,

```
HeaderSet header;
header.setHeader(HeaderSet.TYPE, type);
```

Se ha creado la cabecera header y se ha establecido el tipo de dato type (="application/octet-stream").

Comunicaciones usando PUT o GET

Para comunicaciones que implican tipos de datos personalizados u objetos binarios pesados (ficheros, imágenes) se pueden usar las operaciones PUT y GET. En el servidor se ha utilizando del siguiente modo:

```
Operation op = conn.put(header);
OutputStream out = op.openOutputStream();
out.write(buffer);
```

Primero identificamos la operación PUT con el identificador “op”. A continuación abrimos un flujo de salida para transmitir un objeto, en este caso un fichero. Y en último lugar se lleva a cabo la operación de transferencia mediante el método write().

8. MANUALES

8.1. Manual de usuario

A) Aplicación J2ME "GuiaTurismo.jar"

Una vez descargada la aplicación Java, tan sólo debe instalarse en el terminal correspondiente. Al lanzar dicha aplicación, aparece una pantalla de bienvenida. Si se pulsa el botón correspondiente al comando "Inicio", aparece el menú principal. Para navegar por los diferentes menús, basta con pulsar los comandos "Seleccionar", "Salir" o "Volver".

B) Servidor J2SE "ServidorBTGrafico.jar"

Para lanzar la aplicación servidora habrá que considerar tres casos, según el sistema operativo del ordenador donde se ubique el servidor:

- Si el sistema operativo es Linux, se dispondrá de un script. Para que funcione correctamente es necesario disponer de una cuenta cuyo identificador sea "usuario". El script "ServidorBTScript" estará ubicado en el escritorio y la carpeta "ServidorBTGrafico" de la aplicación se ubicará en la ruta "/home/usuario/ServidorBTGrafico". La aplicación "GuiaTurismo.jar" se ubicará por defecto en la ruta "/home/usuario/GuiaTurismo.jar". Aunque se puede elegir otra ubicación con el botón "Fichero" del servidor. Para lanzar la aplicación basta con ejecutar el script (ServidorBTScript).

  Para evitar reenviar más de una vez al mismo dispositivo la guía turística, el servidor genera un archivo llamado "listado.txt". En este archivo se irán almacenando las direcciones Bluetooth de los diferentes dispositivos a los que se ha enviado la aplicación. En Linux estará almacenado en la ruta "/home/usuario/listado.txt".

- En el caso de Windows se dispondrá del acceso directo "ServidorBT.lnk" que deberá estar ubicado en el escritorio. En el directorio raíz de la unidad C estará ubicado el proyecto "ServidorBTGrafico" (C:\ServidorBTGrafico) con el fichero ejecutable "ServidorBTGrafico.jar" y las correspondientes librerías. También en el directorio raíz de la unidad C deberá estar ubicado el archivo ejecutable "Servidor.bat". El fichero a enviar estará en la ruta "C:\GuiaTurismo.jar". Para lanzar la aplicación bastará con ejecutar el acceso directo "ServidorBT".
  En Windows tendremos el fichero almacén en la ruta "C:\listado.txt".
  En Windows 7 debemos ejecutar el servidor como administrador, para tener acceso total al fichero ejecutable. Para ello clicamos con el botón derecho del ratón en el acceso directo y seleccionamos *"Ejecutar como administrador"*.

- Si es otro sistema operativo el que se use se pondrá el fichero y el directorio del fichero ejecutable donde se estime oportuno. Pero debemos seleccionar el fichero a enviar antes de ejecutar el servidor. Con la carpeta del fichero ejecutable colocada en la ruta que corresponda, desde un panel de línea de comandos tecleamos:
  java -jar $/ruta/ServidorBTGrafico/dist/ServidorBTGrafico.jar
  En otros sistemas operativos el fichero "listado.txt" se ubicará en la carpeta del servidor.

La aplicación servidora exige tener instalado Java en el ordenador en el que se va a ejecutar. De igual modo habrá de tenerse intalado el software de Java en el terminal correspondiente para ejecutar la aplicación "GuiaTurismo.jar".

Al lanzar el servidor, aparece un entorno gráfico con varios botones. El botón "Ejecutar", inicia la ejecución del servidor. Con el botón "Detener" éste se detiene, pudiendo ser ejecutado de nuevo en cualquier momento. Con el botón "Salir" se cierra el servidor. Por último, el botón "Fichero" sirve para seleccionar el fichero a enviar en una ubicación diferente a la preestablecida por defecto. Tras pulsar este botón aparece una ventana con dos opciones: "Abrir" y "Cerrar". Pulsamos "Abrir" y en el cuadro de diálogo que se muestra elegimos el fichero que deseamos enviar. Tras esto, pulsamos "Aceptar" y a continuación "Cerrar". El

fichero a enviar por defecto es “GuíaTurismo.jar”, salvo en el caso de un PC con sistema operativo distinto de Windows o Linux, donde habrá de elegirse el fichero antes de lanzar el servidor.

Como se mencionó anteriormente, el servidor crea un fichero donde se irán almacenando las direcciones Bluetooth de los terminales a los que se le han enviado la aplicación “GuiaTurismo.jar”. De este modo no se envía la aplicación al mismo terminal más de una vez.

8.1.1. Recomendaciones

Para un uso más eficiente se recomienda usar una distribución del sistema operativo Linux, más concretamente la distribución Ubuntu 10.04 o superior. En su defecto es conveniente usar Windows XP.

Si el sistema operativo es distinto a Linux o Windows, primero se debe elegir la ubicación del fichero a enviar (GuiaTurismo.jar por defecto).

8.2. Manual de mantenimiento

Para desarrollar las aplicaciones de este proyecto se ha usado la plataforma Netbeans IDE 6.9.1.

A) Aplicación GuiaTurismo

Consta de dos ficheros: GuiaTurismo.java e Imagen.java.

El primero es un MIDlet donde se desarrollan un menú principal con sus submenús. Para ello se ha recurrido a la clase **List** para crear listas *implícitas* que nos permiten navegar entre los diferentes elementos que componen un menú. Cada menú consta de los siguientes elementos: un array de cadenas de texto, un array de imágenes, las listas asociadas a cada menú y los comandos asociados. Para cargar las imágenes se ha implementado el método **loadImage(String name)**.

En cada menú utilizamos la clase **Alert** para mostrar a la vez el texto deseado y una imagen relacionada. Para poder responder a los eventos generados por los comandos necesitamos lanzar un **setCommandListener()**, en la clase principal (guiaTurismoLinares) de forma que permanezca escuchando en todo momento. Para lanzar los eventos debemos implementar un método del tipo **CommandAction(Command com,Displayable dis)** para cada menú. En algunos casos hemos utilizado la clase **Ticker** para mostrar cadenas de texto largas, de modo que el texto se desplaza de derecha a izquierda.

En el método que inicia la ejecución del MIDlet debemos indicar cual es la pantalla principal, que en nuestro caso es una imagen con un texto de bienvenida. Se trata de una implementación de una variable *Alert* en el método **startApp()**. Para indicar cual es la pantalla actual en cada momento usamos el método ***pantalla*.setCurrent(Entrada, MenuPrincipal)** donde *pantalla* es una variable del tipo **Display**, **Entrada** es una variable del tipo *Alert*, y *MenuPrincipal* es del tipo *List*.

El fichero Imagen.java se emplea para crear el mapa y que se pueda navegar por él con las teclas de desplazamiento de cada terminal. Para ello creamos una clase que hereda de **Canvas** e implementa un **CommandListener**. Creamos el método **volverMapa()** para regresar a la pantalla del menú *MenuMunicipio*. A través del método **Imagen(guiaTurismoLinares mid)** pasamos una referencia del midlet. De este modo el comando **finMapa** queda conectado con el MIDlet.

B) Aplicación ServidorBTGrafico

El proyecto "ServidorBTGrafico" ha sido creado como una aplicación de escritorio. Éste consta de siete archivos, aunque el funcionamiento principal del servidor está implementado en sólo tres de ellos. Los otros cuatro sirven para implementar el entorno gráfico del mismo. Por ello nos centraremos en los tres siguientes: "ServidorBTurista.java", "InspeccionBt.java" y "SesionObex.java".

- Fichero *ServidorBTurista.java*

Contiene el método **iniciaDisp()** para inicializar el dispositivo Bluetooth instalado en el ordenador. Este método supone la primera acción que se realiza al ejecutar el servidor. A continuación se realiza la búsqueda de dispositivos con el método **InspecciónBt.InspecciónBt()**. Una vez descubiertos los dispositivos, leemos el array **InspeccionBt.dispositivosDescubiertos** y comprobamos si se le ha enviado el fichero leyendo de "listado.txt"[2]. Si no se encuentra la dirección del dispositivo en el archivo "listado.txt", se procede al envío del archivo. Para dicho envío se recurre al método **SesionObex.SesionObex(int i)**. Si ha finalizado la comprobación y el envío del fichero, se lanza el método **contarTiempo(int segundos)** y se espera durante un minuto. Terminada la cuenta de tiempo se inicia de nuevo el proceso de búsqueda[3].

2. Si el sistema operativo es distinto a Linux o Windows, se recomienda especificar la ruta de ubicación del fichero en la variable "rutaListadOtro" (e.g. rutaListadOtro = /home/usuario/listado.txt) del fichero tratado
3. Véase el anexo B

– Fichero *InspeccionBt.java*[3]

Este fichero consta de dos eventos. Un evento se encarga de la búsqueda de dispositivos (**eventoBusquedaDispositivosCompletada**) y el otro de la búsqueda de servicios en el dispositivo (**eventoBusquedaServicioCompletada**). Ambos se encuentran sincronizados con el proceso de búsqueda. Se inicia la búsqueda de dispositivos con la sentencia:

```
LocalDevice.getLocalDevice().getDiscoveryAgent().startInquiry(DiscoveryAgent.GIAC,
listener);
```

Se espera a que se lance el evento de búsqueda de dispositivos completada y se inicia la búsqueda de servicios con la sentencia:

```
LocalDevice.getLocalDevice().getDiscoveryAgent().searchServices(attrIDs, searchUuidSet,
DispositivoBt, listener);
```

Si se produce algún error durante el proceso, se reintenta la búsqueda hasta un máximo de diez intentos.

– Fichero *SesionObex.java*[3]

En primer lugar se carga la ruta del fichero a enviar según el sistema operativo en el que se esté ejecutando el servidor. Para Linux la ruta por defecto es "/home/usuario/GuiaTurismo.jar". Para Windows es "C:\GuiaTurismo.jar". Si se trata de otro sistema operativo, antes de ejecutar el servidor se debe seleccionar el fichero a enviar, clicando en el botón "Fichero".

El siguiente paso es crear un flujo para escribir en el fichero "listado.txt", si es necesario, y un flujo para transferir el archivo deseado, si es pertinente. A continuación se crean las cabeceras

3. Véase el anexo B

para la conexión y la trasferencia del archivo. Por último se cierran todos los fujos y la conexión obex.

Si el sistema operativo no es ni Windows ni Linux, el fichero listado.txt se creará en la carpeta "ServidorBTGrafico". Para evitar posibles errores, se recomienda indicar una ruta fija en la variable "ServidorBTGraficoApp.rutaFicherOtro".

Adicionalmente, comentar que en el archivo ServidorBTGraficoApp.java se implementan todas las acciones del entorno gráfico del servidor, así como algunas variables globales que son utilizadas en las clases definidas anteriormente.

9. CONCLUSIONES

Concluyendo, se ha obtenido una aplicación de 1,7MB de peso, con la información más destacada y de interés turístico sobre la ciudad de Linares. Es una aplicación que puede ejecutarse en cualquier dispositivo que soporte la tecnología J2ME. El servidor completo tiene un peso de 1,5MB y puede ejecutarse en cualquier equipo que soporte Java, sea cual sea la plataforma empleada, cumpliéndose los objetivos propuestos.

Por otro lado, se puede decir que la tecnología Bluetooth es apropiada para transferir pequeñas aplicaciones que pueden servir de promoción de lugares turísticos o de otra índole, como por ejemplo enviar publicidad de forma gratuita. Además debido a la rápida evolución de esta tecnología no se tardará mucho en superar limitaciones como puede ser la distancia de conexión.

En ultimo lugar, cabe decir que Java proporciona una plataforma muy completa de clases y librerías para Bluetooth (JSR-82) y para terminales de bajas prestaciones (J2ME) como son los teléfonos móviles. Además, sistemas operativos como Android basan sus desarrollo en Java. Y aunque existen otras plataformas como es la plataforma de C# para el sistema operativo IOS de Apple, no están tan desarrolladas en la comunidad del software libre.

10. Líneas de futuro

La aplicación "GuiaTurismo" se ha diseñado para terminales que soporten configuraciones de MIDlets (MIDP-x.x, CLDC-x.x). En el futuro se podría crear una versión para Android. Restaría, por tanto, diseñarla para el sistema operativo IOS de iPhone (Apple). En el primer caso (Android), habría que crear una aplicación en lenguaje Java con el SDK (Software Development Kit) de Android. En el segundo caso debemos implementar la aplicación en lenguaje C#. De este modo quedarían cubiertos todas las posibilidades de los sistemas operativos de terminales móviles.

Otra posibilidad que quedaría para el futuro, sería diseñar la aplicación J2ME haciendo uso de la clase Canvas de Java. De este modo se podría obtener una aplicación de guía turística más amigable para el usuario.

En cuanto al servidor, se podría diseñar para poder usar dispositivos Bluetooth que soporten varias conexiones simultáneas. El presente servidor se ha elaborado con un dispositivo que permite una sola trasferencia simultánea.

11. BIBLIOGRAFÍA

[1] Timothy J. Thomson. Bluetooth Application Programming with the Java APIs Essentials Edition. Ed. Morgan Kaufmann, 2008.

[2] Albert S. Huang. Ed. Bluetooth Essentials for Programmers. Ed. Cambridge University Press, 2007.

[3] Sing Li. Beginning J2ME - From Novice to Professional, Third Edition. Ed. Apress, 2005.

[4] Sergio Gálvez Rojas, Java a tope: J2ME, Universidad de Málaga, 2003.

[5] Steven Holzner. El Lenguaje Java 2. Ed. Anaya, 2011.

[6] Jürgen Petri. Netbeans Platform 6.9. Developer's Guide. Ed. Packt Publishing (open source*), 2010.

[7] http://www.bluetooth.org

[8] http://www.bluetooth.com

[9] http://www.bluecove.es

ANEXOS

ANEXO A. REQUERIMIENTOS

1. Requisitos para el funcionamiento de la aplicación "GuiaTurismo.jar".

El terminal móvil debe disponer del software de Java necesario para la ejecución de los MIDlets. En concreto las versiones MIDP-2.1 y CLDC-1.1, o superior.

2. Requisitos para el funcionamiento del servidor "ServidorBTGrafico".

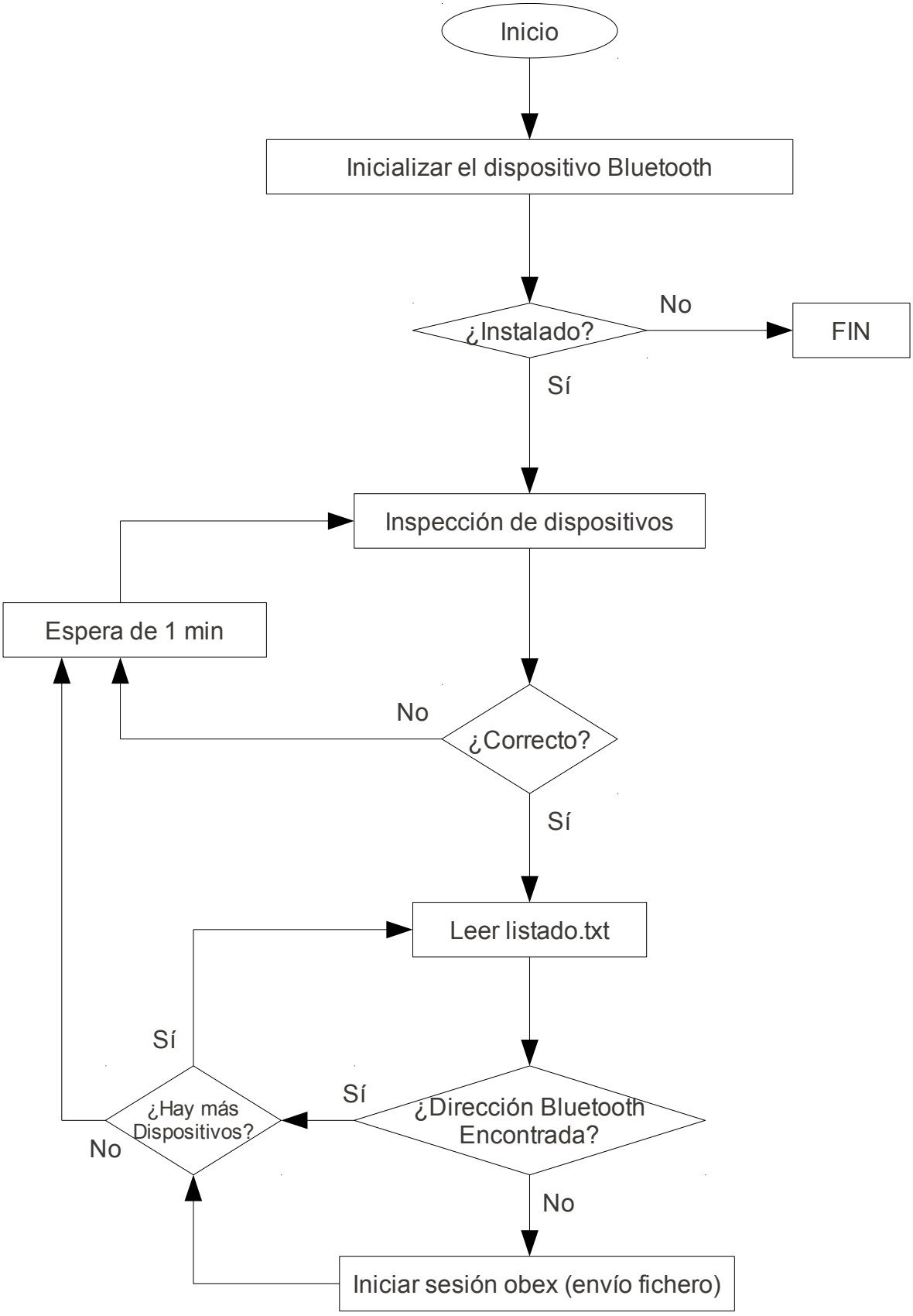
El ordenador en el que se quiera ejecutar el servidor debe tener instalado el software de Java. El enlace para la descarga gratuita de este software es,

"http://java.com/es/download/index.jsp".

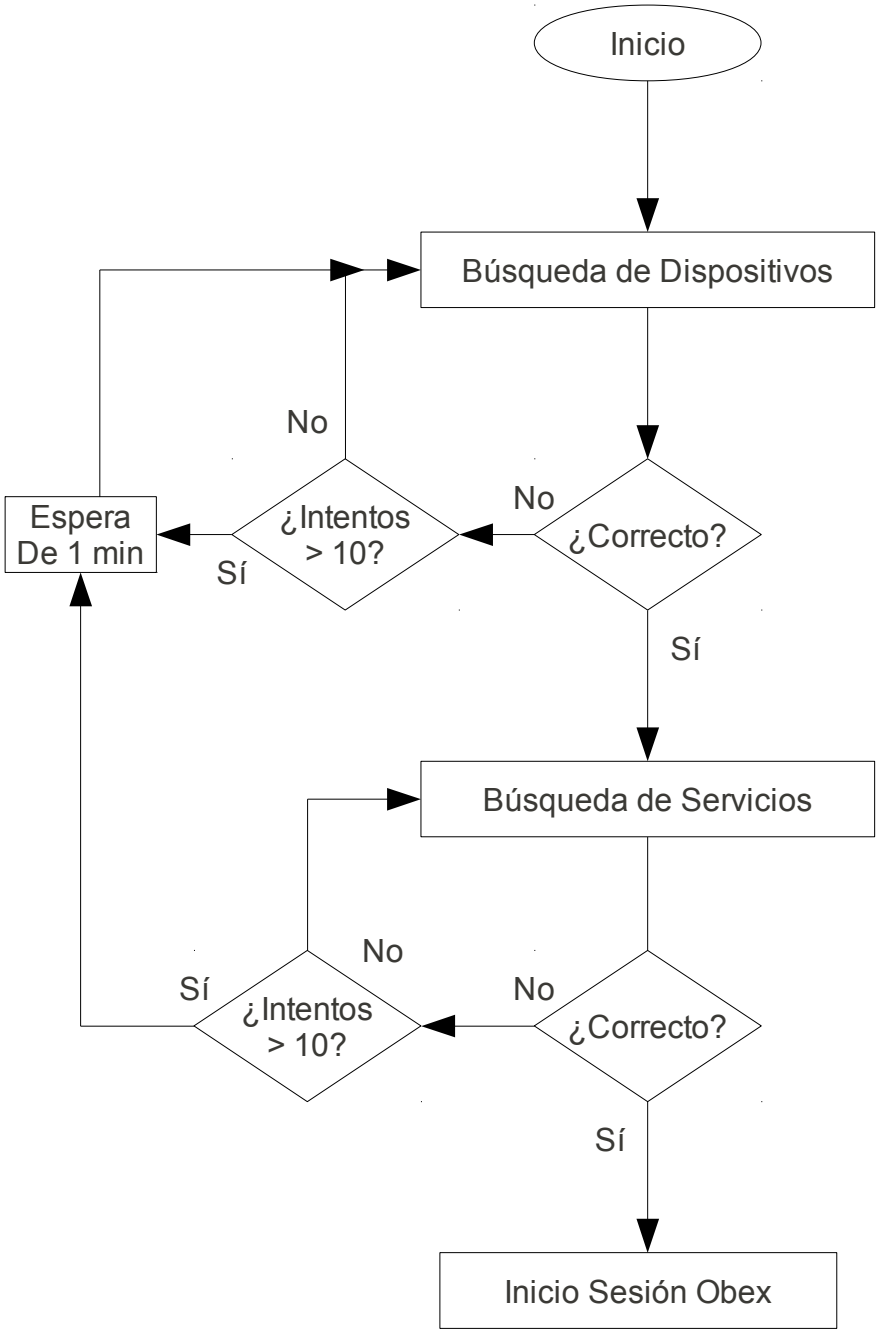
El Ordenador también debe disponer de un dispositivo Bluetooth

Anexo B. Diagramas de flujo.

Fichero “ServidorBTurista.java”.


Inicio
Inicializar el dispositivo Bluetooth
¿Instalado?
No
FIN
Sí
Inspección de dispositivos
Espera de 1 min
No
¿Correcto?
Sí
Leer listado.txt
Sí
¿Hay más Dispositivos?
Sí
¿Dirección Bluetooth Encontrada?
No
No
Iniciar sesión obex (envío fichero)

Fichero “InspecciónBt.java”.



Fichero “SesionObex.java”