

# Control de Sistemas Remotos mediante tecnología Android : REMSYS

Miguel Garcia Ponce

5 de noviembre de 2012

# Índice general

<b>1. Introducción</b>	<b>7</b>
1.1. Proyecto propuesto . . . . .	7
1.2. ¿Qué es Java? . . . . .	8
1.2.1. JRE . . . . .	8
1.2.1.1. Componentes . . . . .	9
1.2.2. APIs . . . . .	10
1.3. ¿Qué es Android? . . . . .	10
1.3.1. Diseño y desarrollo . . . . .	10
1.3.2. Características . . . . .	11
1.3.2.1. Diseño de dispositivo . . . . .	11
1.3.2.2. Almacenamiento . . . . .	11
1.3.2.3. Mensajería . . . . .	11
1.3.2.4. Navegador web . . . . .	11
1.3.2.5. Soporte de Java . . . . .	11
1.3.2.6. Soporte multimedia . . . . .	12
1.3.2.7. Soporte para streaming . . . . .	12
1.3.2.8. Soporte para hardware adicional . . . . .	12
1.3.2.9. Entorno de desarrollo . . . . .	12
1.3.2.10. Google Play . . . . .	12
1.3.2.11. Multi-táctil . . . . .	12
1.3.2.12. Bluetooth . . . . .	12
1.3.2.13. Videollamada . . . . .	13
1.3.2.14. Multitarea . . . . .	13
1.3.2.15. Características basadas en voz . . . . .	13
1.3.2.16. Tethering . . . . .	13
1.3.3. Arquitectura del Sistema Android . . . . .	14
1.3.4. Actualizaciones . . . . .	17
1.4. Conceptos . . . . .	18
1.5. Aplicaciones Cliente-Servidor . . . . .	18
1.5.1. Características . . . . .	19
1.5.2. Middleware . . . . .	20
<b>2. Desarrollo del calendario</b>	<b>21</b>

<b>3. Descripción general del proyecto</b>	<b>24</b>
3.1. Objetivos . . . . .	24
3.2. Alcance . . . . .	25
3.3. Interfaces . . . . .	26
3.3.1. Interfaz del cliente . . . . .	26
3.3.2. Interfaz del servidor . . . . .	27
3.4. Restricciones generales . . . . .	28
3.5. Herramientas utilizadas . . . . .	28
<b>4. Desarrollo del proyecto</b>	<b>30</b>
4.1. Metodología de desarrollo . . . . .	30
4.1.1. Ciclo de vida . . . . .	31
4.2. Requisitos del sistema . . . . .	32
<b>5. Análisis del sistema</b>	<b>34</b>
5.1. Modelo de casos de uso . . . . .	34
5.1.1. Caso de Uso Gestión Servidor . . . . .	35
5.1.1.1. Descripción caso de uso: Cambiar puerto de escucha	36
5.1.1.2. Descripción caso de uso: Cambiar puerto de trans-	
ferencia . . . . .	36
5.1.1.3. Descripción caso de uso: Obtener información del	
sistema . . . . .	37
5.1.1.4. Descripción caso de uso: Pruebas . . . . .	37
5.1.1.5. Descripción caso de uso: Actualizar servidor . . .	38
5.1.2. Caso de Uso Operaciones del cliente . . . . .	38
5.1.2.1. Descripción caso de uso: Apagar. . . . .	40
5.1.2.2. Descripción caso de uso: Reiniciar. . . . .	40
5.1.2.3. Descripción caso de uso: Encender . . . . .	41
5.1.2.4. Descripción caso de uso: Conectar host . . . . .	41
5.1.2.5. Descripción caso de uso: Crear host . . . . .	41
5.1.2.6. Descripción caso de uso: Autenticación . . . . .	42
5.1.3. Caso de Uso Funcionalidades del sistema . . . . .	43
5.1.3.1. Descripción caso de uso: Orden Libre . . . . .	44
5.1.4. Caso de Uso Información del sistema . . . . .	44
5.1.4.1. Descripción caso de uso: Información de discos. .	45
5.1.4.2. Descripción caso de uso: Información de red . .	46
5.1.4.3. Descripción caso de uso: Información de sistema	
operativo. . . . .	46
5.1.4.4. Descripción caso de uso: Información de memoria.	47
5.1.5. Caso de Uso Procesos . . . . .	47
5.1.5.1. Descripción caso de uso: Listar procesos. . . . .	48
5.1.5.2. Descripción caso de uso: Eliminar proceso. . . .	49
5.1.6. Caso de Uso Navegación . . . . .	49
5.1.6.1. Descripción caso de uso: Crear fichero . . . . .	50
5.1.6.2. Descripción caso de uso: Cortar/Copiar fichero .	51
5.1.6.3. Descripción caso de uso: Pegar fichero . . . . .	51

5.1.6.4. Descipción caso de uso: Renombrar fichero . . . . .	52
5.1.6.5. Descipción caso de uso: Eliminar fichero . . . . .	52
5.1.6.6. Descipción caso de uso: Mostrar directorio atrás . . . . .	53
5.1.6.7. Descipción caso de uso: Mostrar directorio adelante . . . . .	53
5.1.6.8. Descipción caso de uso: Transferir fichero . . . . .	54
5.1.7. Caso de Uso Scripts . . . . .	54
5.1.7.1. Descipción caso de uso: Listar Scripts . . . . .	55
5.1.7.2. Descipción caso de uso: Ver Script . . . . .	56
5.1.7.3. Descipción caso de uso: Ejecutar Script . . . . .	56
5.1.7.4. Descipción caso de uso: Transferir Script . . . . .	56
5.1.7.5. Descipción caso de uso: Eliminar Scripts . . . . .	57
5.2. Modelo conceptual de datos . . . . .	57
5.3. Modelo de comportamiento . . . . .	58
5.3.1. Caso de uso Cambiar puerto de escucha . . . . .	59
5.3.2. Caso de uso Cambiar puerto de transferencia . . . . .	59
5.3.3. Caso de uso Obtener información del sistema . . . . .	60
5.3.4. Caso de uso Pruebas . . . . .	61
5.3.5. Caso de uso Actualizar Servidor . . . . .	62
5.3.6. Caso de uso Apagar . . . . .	62
5.3.7. Caso de uso Reiniciar . . . . .	63
5.3.8. Caso de uso Encender . . . . .	64
5.3.9. Caso de uso Conectar host . . . . .	65
5.3.10. Caso de uso Crearhost . . . . .	66
5.3.11. Caso de uso Autenticación . . . . .	68
5.3.12. Caso de uso Orden Libre . . . . .	69
5.3.13. Caso de uso Información del sistema . . . . .	70
5.3.14. Caso de uso Listar procesos . . . . .	71
5.3.15. Caso de uso Eliminar proceso . . . . .	73
5.3.16. Caso de uso Crear fichero . . . . .	74
5.3.17. Caso de uso Cortar/Copiar fichero . . . . .	75
5.3.18. Caso de uso Pegar fichero . . . . .	76
5.3.19. Caso de uso Renombrar fichero . . . . .	77
5.3.20. Caso de uso Eliminar fichero . . . . .	79
5.3.21. Caso de uso Mostrar directorio atrás . . . . .	80
5.3.22. Caso de uso Mostrar directorio adelante . . . . .	81
5.3.23. Caso de uso Transferir fichero . . . . .	82
5.3.24. Caso de uso Listar Scripts . . . . .	83
5.3.25. Caso de uso Ver Script . . . . .	85
5.3.26. Caso de uso Ejecutar Script . . . . .	85
5.3.27. Caso de uso Transferir Script . . . . .	86
5.3.28. Caso de uso Eliminar Script . . . . .	88

<b>6. Diseño del sistema</b>	<b>90</b>
6.1. Arquitectura en tres capas . . . . .	90
6.2. Diagramas de interacción . . . . .	91
6.2.1. Menú Servidor . . . . .	91
6.2.1.1. Cambiar Puerto de escucha/transferencia . . . . .	92
6.2.1.2. Pruebas . . . . .	93
6.2.1.3. Actualizar Servidor . . . . .	94
6.2.2. Menu cliente . . . . .	95
6.2.2.1. Apagar . . . . .	97
6.2.2.2. Reiniciar . . . . .	97
6.2.2.3. Encender . . . . .	98
6.2.2.4. Informacion del sistema . . . . .	98
6.2.2.5. Listar Procesos . . . . .	99
6.2.2.6. Eliminar Proceso . . . . .	100
6.2.2.7. Navegación . . . . .	100
6.2.2.8. Crear fichero . . . . .	101
6.2.2.9. Cortar_Copiar fichero . . . . .	102
6.2.2.10. Pegar fichero . . . . .	102
6.2.2.11. Renombrar fichero . . . . .	103
6.2.2.12. Eliminar fichero . . . . .	103
6.2.2.13. Mostrar directorio atrás . . . . .	104
6.2.2.14. Mostrar directorio adelante . . . . .	105
6.2.2.15. Transferir fichero . . . . .	106
6.2.2.16. Listar Scripts . . . . .	107
6.2.2.17. Ver Script . . . . .	107
6.2.2.18. Ejecutar Script . . . . .	108
6.2.2.19. Transferir Script . . . . .	108
6.2.2.20. Eliminar Script . . . . .	109
6.3. Clases de diseño . . . . .	109
<b>7. Implementación del servidor bajo JDK</b>	<b>113</b>
7.1. Funcionalidades del sistema . . . . .	113
7.1.1. Información de discos . . . . .	113
7.1.1.1. Sistema Windows . . . . .	113
7.1.1.2. Sistema Linux . . . . .	116
7.1.2. Información de Red . . . . .	116
7.1.2.1. Sistema Windows . . . . .	116
7.1.2.2. Sistema Linux . . . . .	121
7.1.3. Información del Sistema Operativo . . . . .	122
7.1.4. Información de memoria . . . . .	124
7.1.4.1. Sistema Windows . . . . .	124
7.1.4.2. Sistema Linux . . . . .	126
7.1.5. Procesos . . . . .	126
7.1.5.1. Sistema Windows . . . . .	126
7.1.5.2. Sistema Linux . . . . .	127
7.1.6. Navegación . . . . .	128

7.1.6.1. MostrarMiPc() . . . . .	130
7.1.6.2. ActualizarGridView() . . . . .	132
7.1.6.3. EjecutarFichero() . . . . .	133
7.1.6.4. TransferirFichero() . . . . .	135
7.1.6.5. CopiarFichero() . . . . .	137
7.1.6.6. CorteFichero() . . . . .	137
7.1.6.7. EliminarFichero() . . . . .	138
7.1.6.8. RenombrarFichero() . . . . .	139
7.1.6.9. CrearFichero() . . . . .	140
7.1.7. Orden libre . . . . .	141
7.1.8. Scripts . . . . .	142
7.1.8.1. EjecutarScript() . . . . .	143
7.1.8.2. EliminarScript() . . . . .	143
7.1.8.3. TransferirScript() . . . . .	144
7.1.8.4. VerScript() . . . . .	145
7.1.9. Apagar Sistema . . . . .	146
7.1.10. Reiniciar Sistema . . . . .	146
7.2. Integración del sistema . . . . .	146
7.3. Interfaz gráfica utilizando JavaSwing . . . . .	151
<b>8. Implementación del cliente usando Android</b>	<b>161</b>
8.1. Interfaz gráfica . . . . .	161
8.1.1. Crear Usuario . . . . .	168
8.1.2. Hosts . . . . .	171
8.1.2.1. Encendido Remoto . . . . .	179
8.1.3. Menú Principal . . . . .	184
8.1.3.1. Reiniciar . . . . .	189
8.1.3.2. Apagar . . . . .	190
8.1.4. Menú Información del sistema . . . . .	191
8.1.4.1. Información de discos . . . . .	193
8.1.4.2. Información de red . . . . .	196
8.1.4.3. Información del Sistema Operativo . . . . .	201
8.1.4.4. Información de Memoria . . . . .	204
8.1.5. Procesos . . . . .	207
8.1.5.1. Eliminar Proceso . . . . .	213
8.1.6. Navegación . . . . .	214
8.1.6.1. Cortar/Copiar Pegar Fichero . . . . .	231
8.1.7. Orden Libre . . . . .	232
8.1.8. Scripts . . . . .	235
<b>9. Pruebas</b>	<b>243</b>
9.1. Pruebas realizadas . . . . .	244

<b>10. Manuales de usuario</b>	<b>246</b>
10.1. Instalación del cliente . . . . .	246
10.2. Autenticación y conexión a un host . . . . .	247
10.3. Acceso a funcionalidades del sistema desde el dispositivo móvil	
Android . . . . .	251
10.3.1. Encendido Remoto . . . . .	251
10.3.2. Información del sistema . . . . .	252
10.3.3. Procesos . . . . .	254
10.3.4. Navegación . . . . .	255
10.3.4.1. Eliminar y Renombrar fichero . . . . .	256
10.3.4.2. Ejecutar fichero . . . . .	256
10.3.4.3. Cortar-Copiar y Pegar fichero . . . . .	257
10.3.4.4. Transferir fichero a SD-Card . . . . .	258
10.3.4.5. Crear fichero . . . . .	259
10.3.5. Orden Libre . . . . .	259
10.3.6. Scripts . . . . .	261
10.3.7. Apagar y Reiniciar . . . . .	262
10.4. Instalación del servidor . . . . .	263
10.4.1. Configuración del Servidor . . . . .	264
10.4.1.1. Configuración del los puertos (escucha y trans-	
ferencia) . . . . .	264
10.4.1.2. Acceso a la información del sistema . . . . .	265
10.4.1.3. Acceso a las pruebas . . . . .	265
10.4.1.4. Abrir los puertos del router . . . . .	266
10.4.1.5. Configuración para encendido remoto (Wake On	
Lan) . . . . .	267
10.4.1.6. Ejecución al inicio del Sistema Operativo . . . . .	270
10.4.1.7. Permisos . . . . .	272
<b>11. Conclusiones</b>	<b>274</b>
11.1. Troubleshooting . . . . .	274
11.2. Versiones posteriores . . . . .	275
<b>12. Apéndice</b>	<b>276</b>
12.1. Código Servidor RemSys . . . . .	276
12.2. Código cliente Remsys . . . . .	321

# Capítulo 1

## Introducción

### 1.1. Proyecto propuesto

El proyecto propuesto, tal y como indica el título se basa en el control de sistemas remotos mediante tecnologías Android. La idea del proyecto surgió, además de por la proposición del tutor, por la necesidad que vi en la realización de mis prácticas de empresa, de obtener información de servidores en todo momento para saber el estado del mismo.

El proyecto consistirá en la construcción de un software cliente y otro servidor, el cual el cliente se conectará mediante sockets a éste segundo, y mediante un protocolo de comunicación obtener distinta información y poder actuar sobre ello desde un terminal móvil. Las funcionalidades que vamos a tener, serán los siguientes: información de red, información de los discos duros, información del sistema operativo, información de la memoria, listado de procesos, navegación por el árbol de directorios, listado y ejecución de scripts localizados en un directorio del servidor, dar orden libre, encendido remoto, reinicio y apagado. Además tendremos funcionalidades añadidas a éstas principales, como por ejemplo creación de ficheros, renombrar ficheros, copiarlos, moverlos... Llegados al punto de descripción de dichas funcionalidades entraremos en más detalles sobre ellas. La condición fundamental para el funcionamiento del sistema es la disposición de red en los dispositivos y la correcta configuración de los componentes que intervienen en ella, principalmente un router que tendrá abierto un puerto determinado (puerto de escucha del servidor) donde se procederá a la transacción de información entre los dispositivos.

El lenguaje de programación utilizado para el desarrollo de dicho proyecto es el lenguaje Java SE (conocido anteriormente hasta la versión 5.0 como Plataforma Java 2, Standard Edition o J2SE), tanto para el servidor como para el cliente. Además se utiliza lenguaje XML para la definición de la interfaz gráfica del cliente y Java Swing/AWT para la interfaz gráfica del servidor.

## 1.2. ¿Qué es Java?

Java es un lenguaje de programación de alto nivel orientado a objetos, desarrollado por James Gosling en 1995. El lenguaje en sí mismo toma mucha de su sintaxis de C, Cobol y Visual Basic, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria. La memoria es gestionada mediante un recolector de basura.

Las aplicaciones Java están típicamente compiladas en un bytecode, aunque la compilación en código máquina nativo también es posible. En el tiempo de ejecución, el bytecode es normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución directa por hardware del bytecode por un procesador Java también es posible. Podemos por lo tanto decir que es independiente de la plataforma, ya que se ejecuta en la máquina virtual de java (JVM).

La implementación original y de referencia del compilador, la máquina virtual y las bibliotecas de clases de Java fueron desarrollados por Sun Microsystems en 1995. Desde entonces, Sun ha controlado las especificaciones, el desarrollo y evolución del lenguaje a través del Java Community Process, si bien otros han desarrollado también implementaciones alternativas de estas tecnologías de Sun, algunas incluso bajo licencias de software libre.

El lenguaje Java se creó con cinco objetivos principales:

1. Debería usar el paradigma de la programación orientada a objetos.
2. Debería permitir la ejecución de un mismo programa en múltiples sistemas operativos.
3. Debería incluir por defecto soporte para trabajo en red.
4. Debería diseñarse para ejecutar código en sistemas remotos de forma segura.
5. Debería ser fácil de usar y tomar lo mejor de otros lenguajes orientados a objetos, como C++.

### 1.2.1. JRE

El JRE (Java Runtime Environment, o Entorno en Tiempo de Ejecución de Java) es el software necesario para ejecutar cualquier aplicación desarrollada para la plataforma Java. El usuario final usa el JRE como parte de paquetes software o plugins (o conectores) en un navegador Web. Sun ofrece también el SDK de Java 2, o JDK (Java Development Kit) en cuyo seno reside el JRE, e incluye herramientas como el compilador de Java, Javadoc para generar documentación o el depurador. Puede también obtenerse como un paquete independiente, y

puede considerarse como el entorno necesario para ejecutar una aplicación Java, mientras que un desarrollador debe además contar con otras facilidades que ofrece el JDK.

### 1.2.1.1. Componentes

- Bibliotecas de Java, que son el resultado de compilar el código fuente desarrollado por quien implementa la JRE, y que ofrecen apoyo para el desarrollo en Java. Algunos ejemplos de estas bibliotecas son:
  - Las bibliotecas centrales, que incluyen: Una colección de bibliotecas para implementar estructuras de datos como listas, arrays, árboles y conjuntos.
  - Bibliotecas para análisis de XML.
  - Seguridad.
  - Bibliotecas de internacionalización y localización.
  - Bibliotecas de integración, que permiten la comunicación con sistemas externos. Estas bibliotecas incluyen:
    - La API para acceso a bases de datos JDBC (Java DataBase Connectivity).
    - La interfaz JNDI (Java Naming and Directory Interface) para servicios de directorio.
    - RMI (Remote Method Invocation) y CORBA para el desarrollo de aplicaciones distribuidas.
  - Bibliotecas para la interfaz de usuario, que incluyen:
    - El conjunto de herramientas nativas AWT (Abstract Windowing Toolkit), que ofrece componentes GUI (Graphical User Interface), mecanismos para usarlos y manejar sus eventos asociados.
    - Las Bibliotecas de Swing, construidas sobre AWT pero ofrecen implementaciones no nativas de los componentes de AWT.
    - APIs para la captura, procesamiento y reproducción de audio.
  - Una implementación dependiente de la plataforma en que se ejecuta de la máquina virtual de Java (JVM), que es la encargada de la ejecución del código de las bibliotecas y las aplicaciones externas.
  - Plugins o conectores que permiten ejecutar applets en los navegadores Web.
  - Java Web Start, para la distribución de aplicaciones Java a través de Internet. Documentación y licencia.

### 1.2.2. APIs

Sun define tres plataformas en un intento por cubrir distintos entornos de aplicación. Así, ha distribuido muchas de sus APIs (Application Program Interface) de forma que pertenezcan a cada una de las plataformas:

- Java ME (Java Platform, Micro Edition) o J2ME — orientada a entornos de limitados recursos, como teléfonos móviles, PDAs (Personal Digital Assistant), etc.
- Java SE (Java Platform, Standard Edition) o J2SE — para entornos de gama media y estaciones de trabajo. Aquí se sitúa al usuario medio en un PC de escritorio.
- Java EE (Java Platform, Enterprise Edition) o J2EE — orientada a entornos distribuidos empresariales o de Internet.

Las clases en las APIs de Java se organizan en grupos disjuntos llamados paquetes. Cada paquete contiene un conjunto de interfaces, clases y excepciones relacionadas. La información sobre los paquetes que ofrece cada plataforma puede encontrarse en la documentación de ésta.

El conjunto de las APIs es controlado por Sun Microsystems junto con otras entidades o personas a través del programa JCP (Java Community Process). Las compañías o individuos participantes del JCP pueden influir de forma activa en el diseño y desarrollo de las APIs, algo que ha sido motivo de controversia.

## 1.3. ¿Qué es Android?

Android es una plataforma de software y un sistema operativo para dispositivos móviles basada en un kernel Linux, desarrollada por Google y más tarde por la Open Handset Alliance. Esta plataforma permite a los desarrolladores escribir código en Java que se ejecuten en móviles mediante las librerías Java desarrolladas por Google. También se pueden escribir aplicaciones en otros lenguajes, como por ejemplo C, para posteriormente ser compiladas en código nativo ARM y ejecutarlas, aunque este proceso de desarrollo no está soportado oficialmente por Google. La mayor parte de la plataforma de Android está disponible bajo licencia de software libre de Apache y otras licencias de código abierto.

### 1.3.1. Diseño y desarrollo

Android, al contrario que otros sistemas operativos para dispositivos móviles como iOS o Windows Phone, se desarrolla de forma abierta y se puede acceder tanto al código fuente como al listado de incidencias donde se pueden ver problemas aún no resueltos y reportar problemas nuevos.

El que se tenga acceso al código fuente no significa que se pueda tener siempre la última versión de Android en un determinado móvil, ya que el código para

soportar el hardware (controladores) de cada fabricante normalmente no es público, así que faltaría un trozo básico del firmware para poder hacerlo funcionar en dicho terminal, y porque las nuevas versiones de Android suelen requerir más recursos, por lo que los modelos más antiguos quedan descartados por razones de memoria (RAM), velocidad de procesador, etc.

### 1.3.2. Características

#### 1.3.2.1. Diseño de dispositivo

La plataforma es adaptable a pantallas de mayor resolución, VGA, biblioteca de gráficos 2D, biblioteca de gráficos 3D basada en las especificaciones de la OpenGL ES 2.0 y diseño de teléfonos tradicionales.

#### 1.3.2.2. Almacenamiento

SQLite, una base de datos liviana, que es usada para propósitos de almacenamiento de datos.

Conectividad Android soporta las siguientes tecnologías de conectividad: GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, HSDPA, HSPA+ y WiMAX.

#### 1.3.2.3. Mensajería

SMS y MMS son formas de mensajería, incluyendo mensajería de texto y ahora la Android Cloud to Device Messaging Framework (C2DM) es parte del servicio de Push Messaging de Android.

#### 1.3.2.4. Navegador web

El navegador web incluido en Android está basado en el motor de renderizado de código abierto WebKit, emparejado con el motor JavaScript V8 de Google Chrome. El navegador por defecto de Ice Cream Sandwich obtiene una puntuación de 100/100 en el test Acid3.

#### 1.3.2.5. Soporte de Java

Aunque la mayoría de las aplicaciones están escritas en Java, no hay una máquina virtual Java en la plataforma. El bytecode Java no es ejecutado, sino que primero se compila en un ejecutable Dalvik y corre en la Máquina Virtual Dalvik. Dalvik es una máquina virtual especializada, diseñada específicamente para Android y optimizada para dispositivos móviles que funcionan con batería y que tienen memoria y procesador limitados. El soporte para J2ME puede ser agregado mediante aplicaciones de terceros como el J2ME MIDP Runner.

### **1.3.2.6. Soporte multimedia**

Android soporta los siguientes formatos multimedia: WebM, H.263, H.264 (en 3GP o MP4), MPEG-4 SP, AMR, AMR-WB (en un contenedor 3GP), AAC, HE-AAC (en contenedores MP4 o 3GP), MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF y BMP.<sup>68</sup>

### **1.3.2.7. Soporte para streaming**

Streaming RTP/RTSP (3GPP PSS, ISMA), descarga progresiva de HTML (HTML5 <video> tag). Adobe Flash Streaming (RTMP) es soportado mediante el Adobe Flash Player. Se planea el soporte de Microsoft Smooth Streaming con el port de Silverlight a Android. Adobe Flash HTTP Dynamic Streaming estará disponible mediante una actualización de Adobe Flash Player.

### **1.3.2.8. Soporte para hardware adicional**

Android soporta cámaras de fotos, de vídeo, pantallas táctiles, GPS, acelerómetros, giroscopios, magnetómetros, sensores de proximidad y de presión, sensores de luz, gamepad, termómetro, aceleración por GPU 2D y 3D.

### **1.3.2.9. Entorno de desarrollo**

Incluye un emulador de dispositivos, herramientas para depuración de memoria y análisis del rendimiento del software. El entorno de desarrollo integrado es Eclipse (actualmente 3.4, 3.5 o 3.6) usando el plugin de Herramientas de Desarrollo de Android.

### **1.3.2.10. Google Play**

Google Play es un catálogo de aplicaciones gratuitas o de pago en el que pueden ser descargadas e instaladas en dispositivos Android sin la necesidad de un PC.

### **1.3.2.11. Multi-táctil**

Android tiene soporte nativo para pantallas capacitivas con soporte multi-táctil que inicialmente hicieron su aparición en dispositivos como el HTC Hero. La funcionalidad fue originalmente desactivada a nivel de kernel (posiblemente para evitar infringir patentes de otras compañías).<sup>70</sup> Más tarde, Google publicó una actualización para el Nexus One y el Motorola Droid que activa el soporte multi-táctil de forma nativa.

### **1.3.2.12. Bluetooth**

El soporte para A2DF y AVRCP fue agregado en la versión 1.5; el envío de archivos (OPP) y la exploración del directorio telefónico fueron agregados en la

versión 2.0; y el marcado por voz junto con el envío de contactos entre teléfonos lo fueron en la versión 2.2.

#### **1.3.2.13. Videollamada**

Android soporta videollamada a través de Google Talk desde su versión HoneyComb.

#### **1.3.2.14. Multitarea**

Multitarea real de aplicaciones está disponible, es decir, las aplicaciones que no estén ejecutándose en primer plano reciben ciclos de reloj, a diferencia de otros sistemas de la competencia en la que la multitarea es congelada (Como por ejemplo iOS, en el que la multitarea se limita a servicios internos del sistema y no a aplicaciones externas)

#### **1.3.2.15. Características basadas en voz**

La búsqueda en Google a través de voz está disponible como "Entrada de Búsqueda" desde la versión inicial del sistema.

#### **1.3.2.16. Tethering**

Android soporta tethering, que permite al teléfono ser usado como un punto de acceso alámbrico o inalámbrico (todos los teléfonos desde la versión 2.2, no oficial en teléfonos con versión 1.6 o inferiores mediante aplicaciones disponibles en Google Play (por ejemplo PdaNet). Para permitir a un PC usar la conexión de datos del móvil android se podría requerir la instalación de software adicional.

### 1.3.3. Arquitectura del Sistema Android

Para empezar con el desarrollo de aplicaciones en Android es importante conocer cómo está estructurado dicho sistema. La arquitectura Android está formada por varias capas. Dicha distribución de capas, permite acceder a las capas más bajas mediante el uso de librerías para que así el desarrollador no tenga que programar a bajo nivel las funcionalidades necesarias para que una aplicación haga uso de los componentes de hardware de los teléfonos.

Cada una de las capas utiliza elementos de la capa inferior para realizar sus funciones, conociéndose este tipo de arquitecturas como “pila”. Para entender mejor, a continuación cito el diagrama de la arquitectura Android:

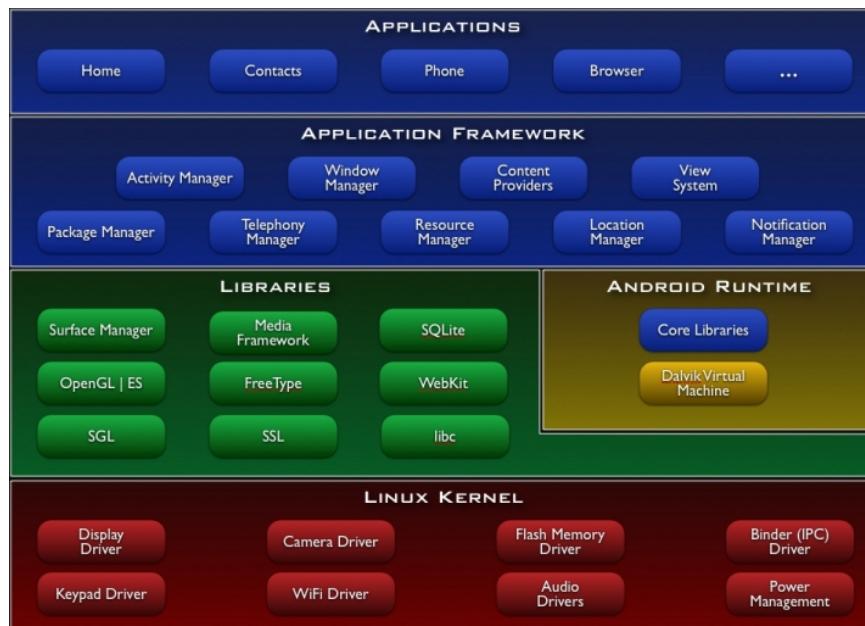


Figura 1.1: Arquitectura del Sistema Android

- Kernel de Linux.

El núcleo actúa como una capa de abstracción entre el hardware y el resto de las capas de la arquitectura. El desarrollador no accede directamente a esta capa, sino que debe utilizar las librerías disponibles en capas superiores. De esta forma también nos evitamos el hecho conocer las características precisas de cada

teléfono es decir, si necesitamos hacer uso de la cámara, el sistema operativo se encarga de utilizar la que incluya el teléfono, sea cual sea. Para cada elemento de hardware del teléfono existe un controlador (o driver) dentro del kernel que permite utilizarlo desde el software.

El kernel también se encarga de gestionar los diferentes recursos del teléfono (energía, memoria, etc.) y del sistema operativo en sí: procesos, elementos de comunicación (networking), etc.

- Librerías.

La siguiente capa que se sitúa justo sobre el kernel la componen las bibliotecas nativas de Android, también llamadas librerías. Están escritas en C o C++ y compiladas para la arquitectura hardware específica del teléfono. Estas normalmente están hechas por el fabricante, quien también se encarga de instalarlas en el dispositivo antes de ponerlo a la venta. El objetivo de las librerías es proporcionar funcionalidad a las aplicaciones para tareas que se repiten con frecuencia, evitando tener que codificarlas cada vez y garantizando que se llevan a cabo de la forma “más eficiente”.

Entre las librerías incluidas habitualmente encontramos OpenGL (motor gráfico), Bibliotecas multimedia (formatos de audio, imagen y video), Webkit (navegador), SSL (cifrado de comunicaciones), FreeType (fuentes de texto), SQLite (base de datos), entre otras.

- Entorno de ejecución.

Como podemos apreciar en el diagrama, el entorno de ejecución de Android no se considera una capa en sí mismo, dado que también está formado por librerías. Aquí encontramos las librerías con la funcionalidades habituales de Java así como otras específicas de Android.

El componente principal del entorno de ejecución de Android es la máquina virtual Dalvik. Las aplicaciones se codifican en Java y son compiladas en un formato específico para que esta máquina virtual las ejecute. La ventaja de esto es que las aplicaciones se compilan una única vez y de esta forma estarán listas para distribuirse con la total garantía de que podrán ejecutarse en cualquier dispositivo Android que disponga de la versión mínima del sistema operativo que requiera la aplicación.

Cabe aclarar que Dalvik es una variación de la máquina virtual de Java, por lo que no es compatible con el bytecode Java. Java se usa únicamente como lenguaje de programación, y los ejecutables que se generan con el SDK de Android tienen la extensión .dex que es específico para Dalvik, y por ello no podemos correr aplicaciones Java en Android ni viceversa.

- Framework de aplicaciones.

La siguiente capa está formada por todas las clases y servicios que utilizan directamente las aplicaciones para realizar sus funciones. La mayoría de los componentes de esta capa son librerías Java que acceden a los recursos de las capas anteriores a través de la máquina virtual Dalvik. Siguiendo el diagrama encontramos:

1. Activity Manager. Se encarga de administrar la pila de actividades de nuestra aplicación así como su ciclo de vida.
2. Windows Manager. Se encarga de organizar lo que se mostrará en pantalla. Básicamente crea las superficies en la pantalla que posteriormente pasarán a ser ocupadas por las actividades.
3. Content Provider. Esta librería es muy interesante porque crea una capa que encapsula los datos que se compartirán entre aplicaciones para tener control sobre cómo se accede a la información.
4. Views. En Android, las vistas los elementos que nos ayudarán a construir las interfaces de usuario: botones, cuadros de texto, listas y hasta elementos más avanzados como un navegador web o un visor de Google Maps.
5. Notification Manager. Engloba los servicios para notificar al usuario cuando algo requiera su atención mostrando alertas en la barra de estado. Un dato importante es que esta biblioteca también permite jugar con sonidos, activar el vibrador o utilizar los LEDs del teléfono en caso de tenerlos.
6. Package Manager. Esta biblioteca permite obtener información sobre los paquetes instalados en el dispositivo Android, además de gestionar la instalación de nuevos paquetes. Con paquete nos referimos a la forma en que se distribuyen las aplicaciones Android, estos contienen el archivo .apk, que a su vez incluyen los archivos .dex con todos los recursos y archivos adicionales que necesite la aplicación, para facilitar su descarga e instalación.
7. Telephony Manager. Con esta librería podremos realizar llamadas o enviar y recibir SMS/MMS, aunque no permite reemplazar o eliminar la actividad que se muestra cuando una llamada está en curso.
8. Resource Manager. Con esta librería podremos gestionar todos los elementos que forman parte de la aplicación y que están fuera del código, es decir, cadenas de texto traducidas a diferentes idiomas, imágenes, sonidos o layouts.
9. Location Manager. Permite determinar la posición geográfica del dispositivo Android mediante GPS o redes disponibles y trabajar con mapas.
10. Sensor Manager. Nos permite manipular los elementos de hardware del teléfono como el acelerómetro, giroscopio, sensor de luminosidad, sensor de campo magnético, brújula, sensor de presión, sensor de proximidad, sensor de temperatura, etc.
11. Cámara: Con esta librería podemos hacer uso de la(s) cámara(s) del dispositivo para tomar fotografías o para grabar vídeo.
12. Multimedia. Permiten reproducir y visualizar audio, vídeo e imágenes en el dispositivo.

- Aplicaciones. En la última capa se incluyen todas las aplicaciones del dispositivo, tanto las que tienen interfaz de usuario como las que no, las nativas (programadas en C o C++) y las administradas (programadas en Java), las que vienen preinstaladas en el dispositivo y aquellas que el usuario ha instalado. En esta capa encontramos también la aplicación principal del sistema: Inicio (Home) o lanzador (launcher), porque es la que permite ejecutar otras aplicaciones mediante una lista y mostrando diferentes escritorios donde se pueden colocar accesos directos a aplicaciones o incluso widgets, que son también aplicaciones de esta capa.

Como podemos ver, Android nos proporciona un entorno con un gran potencial para poder desarrollar aplicaciones que realicen cualquier tipo de tarea.

#### 1.3.4. Actualizaciones

Es una pieza fundamental en el éxito del sistema Android. Las actualizaciones permiten que el sistema esté siempre en continua evolución, solventando los principales errores (bugs) encontrados en versiones anteriores y optimizando otros muchos aspectos para hacer que el sistema sea mucho más flexible y eficiente. Paso a comentar las últimas características añadidas en la última versión de Android (4.1 Jelly Bean):

- Mejora de la fluidez y de la estabilidad gracias al proyecto "Project Butter".
- Ajuste automático de widgets cuando se añaden al escritorio, cambiando su tamaño y lugar para permitir que los nuevos elementos se puedan colocar.
- Dictado por voz mejorado con posibilidad de utilizarlo sin conexión a Internet.
- Nuevas lenguas no occidentales.
- Android Beam mejorado con posibilidad de transmitir vídeo por NFC.
- Nuevo modo de acceso rápido al álbum en la cámara, llamado Quick View.
- Notificaciones mejoradas, con acceso más rápido a más información en la propia barra de notificaciones.
- Nueva función Google Now, que nos permite tener una serie de "tarjetas inteligentes" que nos muestran información importante como el tiempo, el tráfico, si nuestros vuelos se han cancelado y muchas otras funciones.
- Búsqueda por voz mejorada, ahora contesta preguntas formuladas como si estuviéramos hablando con otra persona y nos da información en la misma aplicación, actualmente solo en inglés, pero han prometido más idiomas, en las comparativas se demuestra la velocidad de este servicio frente a otros como "Siri" de Apple.

- Cifrado de aplicaciones. En las actualizaciones de aplicaciones solo se descarga la parte de ésta que ha sido cambiada.
- Google Chrome se convierte en el navegador por defecto de Android. Se pone fin al soporte de Flash Player para Android a partir de esta versión.
- Nueva función "Sound Search", que permite saber qué música estás escuchando.
- Gestual Mode para las personas discapacitadas visualmente.
- Pequeños cambios en la interfaz, como la nueva barra de búsquedas.

Como vemos siempre va a haber mejoras en los diferentes aspectos del sistema, haciendo que sea un sistema en continua evolución, adaptándose a las necesidades del usuario a medida que van surgiendo éstas.

## 1.4. Conceptos

Vamos a proceder a dar una definición de los principales términos que nos vamos a encontrar en un entorno cliente-servidor:

- Cliente: El cliente es el proceso que permite al usuario formular los requerimientos y pasarlo al servidor, se le conoce con el término front-end.
- Servidor: Es el proceso encargado de atender a múltiples clientes que hacen peticiones de algún recurso administrado por él. Al proceso servidor se le conoce con el término back-end.
- Middleware: Un conjunto de controladores, API u otro software que mejora la conectividad entre las aplicaciones de cliente y un servidor.
- Base de Datos Relacional: Una base de datos en donde el acceso a la información está limitado por la selección de filas que satisfacen todos los criterios de búsqueda.
- Lenguaje de Consulta Estructurado: Un lenguaje desarrollado por IBM y estandarizado por ANSI para direccionar, crear, actualizar o consultar bases de datos relacionales.

## 1.5. Aplicaciones Cliente-Servidor

La arquitectura cliente-servidor es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. En la actualidad, las aplicaciones cliente-servidor forman parte de nuestra vida diaria, y han alcanzado gran importancia dentro del entorno empresarial. La necesidad de tener la cierta información centralizada y disponible para la mayoría de componentes

de la empresa. Dichas aplicaciones, permiten, a su vez, el control de acceso a dicha información, permitiendo la creación de grupos y la implementación de un sistema de autenticación que hace de estos sistemas más seguros.

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa. Los tipos específicos de servidores incluyen los servidores web, los servidores de archivo, los servidores del correo, etc. Mientras que sus propósitos varían de unos servicios a otros, la arquitectura básica seguirá siendo la misma.

Una disposición muy común son los sistemas multicapa en los que el servidor se descompone en diferentes programas que pueden ser ejecutados por diferentes computadoras aumentando así el grado de distribución del sistema.

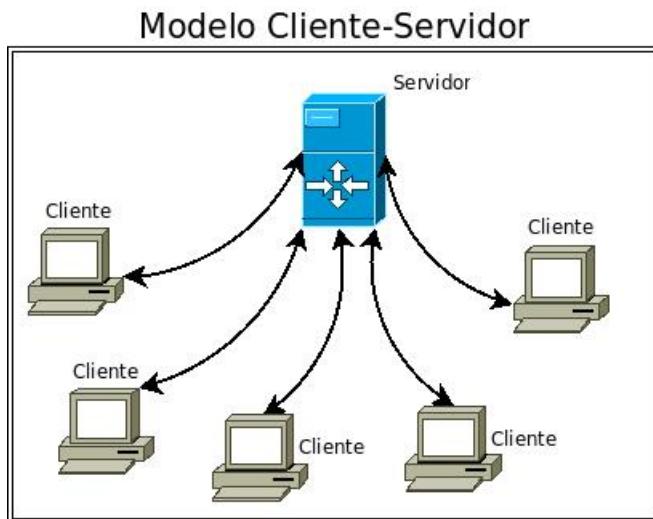


Figura 1.2: Diagrama modelo Cliente - Servidor

### 1.5.1. Características

Las características básicas de las arquitecturas cliente-servidor son las siguientes:

- Combinación de un cliente que interactúa con el usuario, y un servidor que interactúa con los recursos compartidos. El proceso del cliente proporciona la interfaz entre el usuario y el resto del sistema. El proceso del servidor maneja recursos compartidos y las sirve al servidor.

- Cliente y servidor con diferentes requerimientos de cómputo.
- Relación entre diferentes procesos.
- Distinción de funciones de servicios entre clientes y servidores.
- La relación de servicio puede ser de uno a muchos.
- Ambiente heterogéneo. Los sistemas del cliente y servidor no son siempre los mismos.
- Los sistemas cliente-servidor son ampliamente escalables.

### 1.5.2. Middleware

Como ya hemos comentado, un middleware es un conjunto de controladores, API u otro software que mejora la conectividad entre las aplicaciones de cliente y un servidor. Funciona como una capa de abstracción de software distribuida, que se sitúa entre las capas de aplicaciones y las capas inferiores (sistema operativo y red).

El middleware abstracta de la complejidad y heterogeneidad de las redes de comunicaciones subyacentes, así como de los sistemas operativos y lenguajes de programación, proporcionando una API para la fácil programación y manejo de aplicaciones distribuidas. Dependiendo del problema a resolver y de las funciones necesarias, serán útiles diferentes tipos de servicios de middleware. Por lo general el middleware del lado cliente está implementado por el Sistema Operativo, el cual posee las bibliotecas que ejecutan todas las funcionalidades para la comunicación a través de la red. Un esquema básico de su funcionamiento lo podemos ver en la siguiente figura:

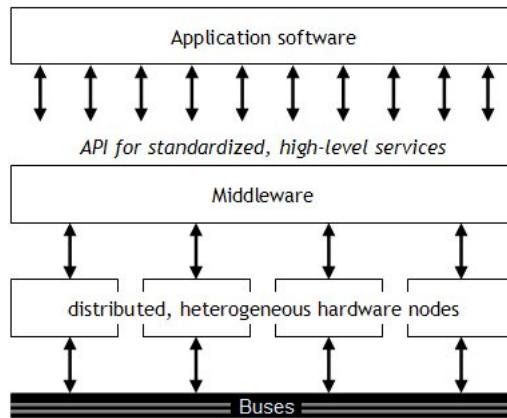


Figura 1.3: Diagrama Middleware

## Capítulo 2

# Desarrollo del calendario

Se ha desarrollado el calendario usado para el desarrollo del proyecto. El tiempo de duración total se estima en algo más de 6 meses, en el cual he dividido dicho proyecto en varias etapas, las cuales paso a indicar y resumir con la siguiente figura:

1. Acuerdo de Proyecto: El primer punto fué formalizar el proyecto, se sentó las bases de en qué iba a consistir dicho trabajo y se dió un plazo para pensarlo y decidirse.
2. Aprendizaje Java y Android: Esta estapa dura casi todo el tiempo que duró la realización del proyecto, puesto que siempre vas a estar aprendiendo cosas de ambas plataformas, tanto para el comienzo y bases de dichos lenguajes como para la resolución de problemas y en la etapa de pruebas, en la cual van surgiendo diferentes aspectos que van a necesitar corregirse y esta etapa es la que va a formarme en el conocimiento de la resolución de dichos aspectos.
3. Estudio de la conectividad software: Se comenzó con pequeños ejemplos de conexión de arquitecturas cliente-servidor, para luego transpasarlo al ámbito de dispositivos android - pcs. Se investigó sobre la opción de RMI (Java Remote Method Invocation, que es un mecanismo ofrecido por Java para invocar un método de manera remota), que forma parte del entorno estándar de ejecución de Java y proporciona un mecanismo simple para la comunicación de servidores en aplicaciones distribuidas basadas exclusivamente en Java, y por lo tanto no estaba disponible con la plataforma Dalvik VM, utilizada por Android.
4. Especificación de funcionalidades: Se pensó en cuales iban a ser las funcionalidades que iba a permitir el sistema, conociendo la viabilidad de dichas funciones.
5. Diseño del cliente: Se especifican el sistema y técnicas que se van a utilizar para cumplir los requisitos del cliente.

6. Diseño del servidor: Se especifican el sistema y técnicas que se van a utilizar para cumplir los requisitos del servidor.
7. Implementación: Se procede a la implementación de las funcionalidades para posteriormente integrarlas tanto en el cliente como en el servidor.
8. Pruebas: Se procede a las pruebas para el control de fallos del sistema.
9. Documentación: En esta etapa se procede a la realización de la documentación. Tanto la guía de usuario como memoria y transparencias para su posterior exposición.

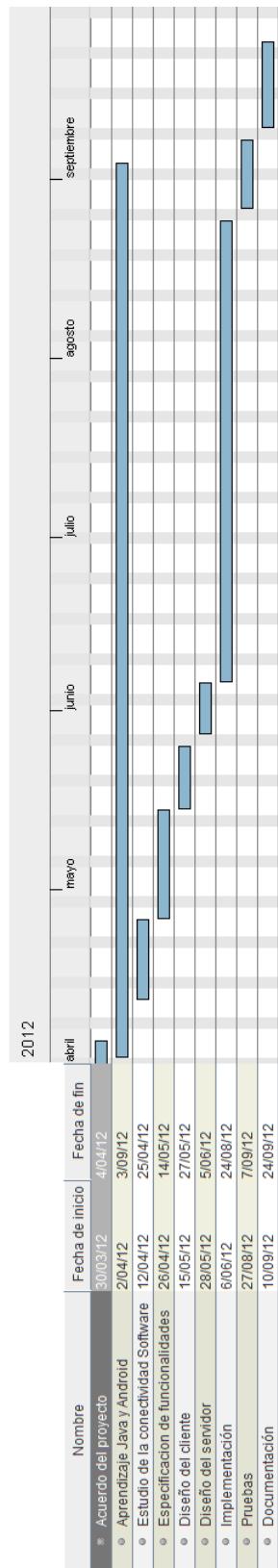


Figura 2.1: Diagrama de Gantt

## Capítulo 3

# Descripción general del proyecto

Remsys es un proyecto multiplataforma, el cual permite monitorizar el sistema mediante el uso de la red. Se compone de un dispositivo móvil con sistema Android y un servidor, con los sistemas Linux o Windows. La información obtenida se obtendrá mediante el uso de comandos del sistema. El proyecto es altamente configurable, podremos realizar scripts, y mediante ellos ejecutarlos cuando queramos desde el dispositivo móvil, dotando al sistema de gran utilidad.



Figura 3.1: Logo “RemSys”

### 3.1. Objetivos

La idea principal del proyecto es la realización de una aplicación de los entornos cliente-servidor, el cual va a consistir en el control y obtención de información de un sistema remoto y la presentación de esta información en el dispositivo móvil. Para ello se ha construido un protocolo de intercambio de información para empezar las distintas acciones que el servidor deberá realizar. Dicho intercambio se realiza mediante Sockets que utilizará la Internet para dicho intercambio.

Se ha creado una aplicación servidor, el cual estará a la espera de una conexión entrante, y recibirá órdenes desde este dispositivo móvil. Una vez realizada

la acción pedida, se le mandará el resultado de nuevo al dispositivo móvil, presentándola en pantalla la información correspondiente y pudiendo en algunos casos, realizar acciones adicionales. Todas las acciones que se manden desde el dispositivo móvil, quedará registrada en la pantalla del servidor, viéndose en tiempo real.

Por lo tanto el proyecto se ha basado principalmente en:

- Contrucción del cliente en dispositivo móvil
- Contrucción del servidor y sus funcionalidades.
- Interconexión entre los dos anteriores, enviando y recibiendo información entre ellos.

### 3.2. Alcance

La realización de este proyecto como ya he mencionado está orientado al control de sistemas remotos (servidores principalmente), donde obtendremos diversa información y podremos realizar distintas acciones. A continuación, muestro una lista de las distintas acciones/información que podemos realizar con estas aplicaciones, recordando que se podrá utilizar con sistemas Android-Windows y Android-Linux:

- Información de discos: Podremos obtener, el espacio de las particiones que tiene el sistema, así como su porcentaje de uso. Dicha información nos podrá servir para borrar archivos si hiciera falta espacio.
- Información de red: Obtendremos las distintas interfaces de red, sus direcciones ips y macs, máscaras de subred, si está vinculado a un servidor DHCP...
- Información del sistema operativo: Información sobre varios directorios importantes, configuración de java, directorio actual...
- Información sobre memoria: Memoria física y virtual, donde podremos tomar acciones para eliminar procesos que obtienen más recursos.
- Listado de procesos.
- Eliminar proceso.
- Navegación por el árbol de directorio: Podremos navegar por el sistema de ficheros, y podremos realizar diferentes acciones como:
  - Crear fichero.
  - Cortar/Pegar.
  - Copiar/Pegar.

- Renombrar fichero.
  - Ejecutar ficheros con extensiones java, exe, bat, py, bin, run, sh.
  - Eliminar ficheros.
  - Transferir ficheros a la tarjeta de memoria del dispositivo móvil
- Dar orden libre: esta funcionalidad nos permitirá obtener mayor flexibilidad y obtener diferente información de la preestablecida en la aplicación.
  - Ejecución de Scripts.
  - Ver Script.
  - Elimiar Script.
  - Transferir Script.

### 3.3. Interfaces

Las interfaces que vamos a poder observar en este proyecto van a ser las del cliente y otra la del servidor. La primera de ellas la podemos ver en el dispositivo móvil y la segunda la veremos en el sistema remoto a controlar.

#### 3.3.1. Interfaz del cliente

La interfaz del cliente se basa en Android, y se basa en los layouts que son los contenedores de los distintos componentes del sistema. En ellas podemos agregar botones, etiquetas, cajas de edición de textos, imágenes... Dichos layouts están definidos mediante ficheros XML y controlado en general por el documento `AndroidManifest.xml` el cual controla las distintas actividades del sistema. A cada layout se le asigna un comportamiento, el cual viene definido en un fichero java, el cual permitirá la navegación entre los distintos layouts para poder así desplazarse por el y obteniendo la distinta información del servidor. En la siguiente figura como ejemplo, se muestra la pantalla de login del dispositivo móvil:



Figura 3.2: Interfaz cliente

### 3.3.2. Interfaz del servidor

Por otra parte, el servidor tendrá una pantalla de información que nos alertará de las diferentes acciones que se están realizando, desde la conexión hasta la obtención de información. Tendrá también un botón para el reinicio del servidor en el puerto de escucha especificado. Además contará con información directa como IP local, IP pública y dirección MAC. Esta información también podrá ser obtenida, aún más ampliada (de todas sus interfaces de red) mediante el acceso a un menú de Información del sistema. Además de este menú, tendremos otro para el cambio del puerto, tanto de escucha general de conexión como el puerto de transferencia de ficheros y otro menú de pruebas para comprobar que lo que se va a mandar es correcto. En la siguiente figura se observa la ventana principal del servidor, construida con Java Swing (es la herramienta de construcción de GUI principal de Java, siendo parte de las Fundación de clases java de Oracle, JFC).



Figura 3.3: Interfaz del servidor RemSys

### 3.4. Restricciones generales

Por parte del servidor la única restricción que tenemos es que no está disponible para la plataforma Mac OS, aunque si quisieramos portarlo a dicha plataforma tendríamos que determinar las órdenes del sistema que implementan dichas funcionalidades y manipular los datos para obtener la información que se desea. Por parte del cliente la restricción es que disponga de sistema Android, está probado desde la versión 2.2, y que tenga acceso a internet.

Una restricción, aunque no tan importante, viene dada por el tamaño de la pantalla del dispositivo móvil. El proyecto está realizado para dispositivos con pantalla MDPI, o pantallas con puntos de densidad media que es la pantalla base de los dispositivos Android, aunque la aplicación se adaptará a cualquier tamaño de pantalla donde se ejecute.

### 3.5. Herramientas utilizadas

Las herramientas utilizadas para la realización del proyecto son las siguientes:

- Microsoft Windows 7 : Para la implementación del servidor en dicho entorno.
- Ubuntu 12.10: Para la implementación del servidor en dicho entorno.
- Java version 1.7.0\_05 con Java(TM) SE Runtime Environment y Java HotSpot(TM) 64-Bit Server VM : Máquina gratuita de Java.

- Eclipse Juno for Java EE Developers: entorno de desarrollo integrado de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido".
- Android SDK: Android Software Development Kit o Kit de desarrollo de software Android, herramienta básica que se integra con Eclipse para la realización de proyectos basados en Android.
- Android Development Tools (ADT): es un plugin para el entorno Eclipse que está diseñado para dar un entorno importante e integrado para construir aplicaciones Android.
- Lenguajes de Scripts para ejemplos como pueden ser los ".bat" en Windows y ".py" o los ".sh" , ".run" en Linux entre otros. También se ha probado la instalación de un intérprete Python para la ejecución de dicho tipos de Scripts en Windows.
- LyX: es un programa gráfico multiplataforma creado por Matthias Ettrich que permite la edición de texto usando LaTeX, por lo que hereda todas sus capacidades (notación científica, edición de ecuaciones, creación de índices, etcétera).
- StarUML: programa para realizar tanto el análisis como el diseño del proyecto. Diagramas de casos de uso, modelo conceptual, de comportamiento... es capaz de realizar dicho software.
- GanttProject: Software para realizar diagramas de Gantt para realización del calendario.

# **Capítulo 4**

# **Desarrollo del proyecto**

En este capítulo veremos la metodología de desarrollo utilizada para realizar el proyecto y las especificaciones de los requisitos del sistema.

## **4.1. Metodología de desarrollo**

La metodología de desarrollo utilizada es la más ampliamente extendida y estandarizada llamada “Proceso Unificado de Rational” (Rational Unified Process en inglés, habitualmente resumido como RUP) es un proceso de desarrollo de software desarrollado por la empresa Rational Software, actualmente propiedad de IBM. Junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, diseño, implementación y documentación de sistemas orientados a objetos. El RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización.

El RUP está basado en 6 principios clave que son los siguientes:

- Adaptar el proceso: El proceso deberá adaptarse a las necesidades del cliente ya que es muy importante interactuar con él. Las características propias del proyecto u organización, el tamaño del mismo, así como su tipo o las regulaciones que lo condicione, influirán en su diseño específico. También se deberá tener en cuenta el alcance del proyecto en un área sub formal para hacer un proceso de satisfacción del software.
- Equilibrar prioridades: Los requisitos de los diversos participantes pueden ser diferentes, contradictorios o disputarse recursos limitados. Debe encontrarse un equilibrio que satisfaga los deseos de todos. Gracias a este equilibrio se podrán corregir desacuerdos que surjan en el futuro.
- Demostrar valor iterativamente: Los proyectos se entregan, aunque sea de un modo interno, en etapas iteradas. En cada iteración se analiza la opinión de los inversores, la estabilidad y calidad del producto, y se refina la dirección del proyecto así como también los riesgos involucrados.

- Colaboración entre equipos: El desarrollo de software no lo hace una única persona sino múltiples equipos. Debe haber una comunicación fluida para coordinar requisitos, desarrollo, evaluaciones, planes, resultados, etc.
- Elevar el nivel de abstracción: Este principio dominante motiva el uso de conceptos reutilizables tales como patrón del software, lenguajes 4GL o marcos de referencia (frameworks) por nombrar algunos. Esto evita que los ingenieros de software vayan directamente de los requisitos a la codificación de software a la medida del cliente, sin saber con certeza qué codificar para satisfacer de la mejor manera los requisitos y sin comenzar desde un principio pensando en la reutilización del código. Un alto nivel de abstracción también permite discusiones sobre diversos niveles y soluciones arquitectónicas. Estas se pueden acompañar por las representaciones visuales de la arquitectura, por ejemplo con el lenguaje UML.
- Enfocarse en la calidad: El control de calidad no debe realizarse al final de cada iteración, sino en todos los aspectos de la producción. El aseguramiento de la calidad forma parte del proceso de desarrollo y no de un grupo independiente.

#### 4.1.1. Ciclo de vida

El ciclo de vida RUP es una implementación del Desarrollo en espiral. Fue creado ensamblando los elementos en secuencias semi-ordenadas. El ciclo de vida organiza las tareas en fases e iteraciones. RUP divide el proceso en cuatro fases, dentro de las cuales se realizan varias iteraciones en número variable según el proyecto y en las que se hace un mayor o menor hincapié en las distintas actividades.

Las primeras iteraciones (en las fases de Planificación e Ingeniería) se enfocan hacia la comprensión del problema y la tecnología, la delimitación del ámbito del proyecto, la eliminación de los riesgos críticos, y al establecimiento de una línea base de la arquitectura.

Durante la fase de planificación las iteraciones hacen mayor énfasis en actividades de modelado del negocio y de requisitos. En la fase de ingeniería, las iteraciones se orientan al desarrollo de la baseline de la arquitectura, abarcan más los flujos de trabajo de requisitos, modelo de negocios (refinamiento), análisis, diseño y una parte de implementación orientado a la baseline de la arquitectura. En la fase de construcción, se lleva a cabo la construcción del producto por medio de una serie de iteraciones.

Para cada iteración se seleccionan algunos Casos de Uso, se refinan su análisis y diseño y se procede a su implementación y pruebas. Se realiza una pequeña cascada para cada ciclo. Se realizan iteraciones hasta que se termine la implementación de la nueva versión del producto. En la fase de transición se pretende garantizar que se tiene un producto preparado para su entrega a la comunidad de usuarios.

Como se puede observar en cada fase participan todas las disciplinas, pero dependiendo de la fase el esfuerzo dedicado a una disciplina varía. En la siguiente figura podemos observar el gráfico del desarrollo por el cual se ha optado:

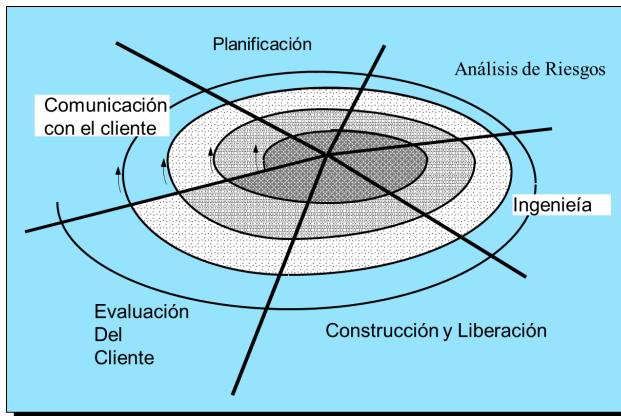


Figura 4.1: Desarrollo en espiral

## 4.2. Requisitos del sistema

Los requisitos del sistema del proyecto consistirán en:

- El servidor estará activado, pudiéndose cambiar el puerto de escucha en cualquier momento.
- Se establecerá una conexión para la comunicación entre los dispositivos, utilizando Sockets. El servidor aceptará conexiones mediante la construcción de un ServerSocket y el cliente creará el Socket para su conexión.
- Control de flujos: Se obtendrán los flujos para la comunicación, con clases ObjectInputStream y ObjectOutputStream.
- El servidor se podrá reiniciar en cualquier momento, volviendo al punto de espera de alguna conexión entrante.
- El servidor, para obtener información del sistema, procederá mediante la clase Runtime. Cada aplicación Java tiene una instancia simple de la clase Runtime que permite tener una interfaz con el entorno en el cual la aplicación está ejecutándose. El runtime actual puede obtenerse con el método getRuntime.
- El cliente, dispondrá principalmente de dos tipos de componentes. Uno de ellos consistirá principalmente en botones, el cual procederá a mandar un mensaje al servidor. En el servidor, una vez interpretado el mensaje,

y mediante la construcción anterior, obtendrá la información y la mandará al cliente donde la presentará en un segundo tipo de componente al cual podemos definir como de presentación de información. Estos últimos componentes serán GridViews, ListViews o un simple EditText.

- El servidor dispondrá en modo local, un menú con opciones de prueba de las funcionalidades implementadas para su comprobación.
- El servidor dispondrá de un menú para la información de las interfaces de red, el cual es útil para saber la ip local, si por ejemplo estamos conectados en nuestro terminar mediante VPN y queremos obtener información desde el dispositivo y sus direcciones MACs principalmente para la función “Wake On Lan” o encendido remoto.

# Capítulo 5

## Análisis del sistema

En este capítulo nos centraremos en hacer un modelado del análisis y diseño el cual nos permita establecer los requisitos del sistema, además de sus componentes estructurales y el comportamiento que va a tener el sistema ante las acciones realizadas por el cliente. A continuación, pasaremos a realizar el modelo de casos de uso (con sus descripciones correspondientes), pasando posteriormente a realizar el modelo conceptual, modelo de comportamiento y por último el diagrama de estado de sistema.

### 5.1. Modelo de casos de uso

Comenzamos el modelado del sistema, presentando el modelo de casos de uso. En este apartado podemos encontrar un modelo de caso de uso general, donde establecemos los roles que van a representar al cliente de las aplicaciones, el cual podemos ver en la siguiente imagen:



Figura 5.1: Caso de Uso general

En la figura podemos observar los distintos roles que el cliente va a tener. Uno de ellos (Cliente), se refiere a cuando se está tratando con el dispositivo

móvil el cual va a tener la opción de realizar distintas operaciones, representadas como el caso de uso general Operaciones del cliente. Por otra parte, el rol de Administrador del servidor, tenemos otro caso de uso general llamado Gestionar Servidor, el cual contendrá otros casos de uso que contendrán las operaciones que podrá realizar el cliente representado por ese rol.

### 5.1.1. Caso de Uso Gestionar Servidor

Empezaremos a entrar en más detalle en el modelo de casos de uso con el caso de uso “Gestionar Servidor”. Aquí podremos encontrar las diferentes operaciones que el usuario podrá realizar con la aplicación servidor. En la siguiente figura podemos observar como quedaría dicho caso de uso:

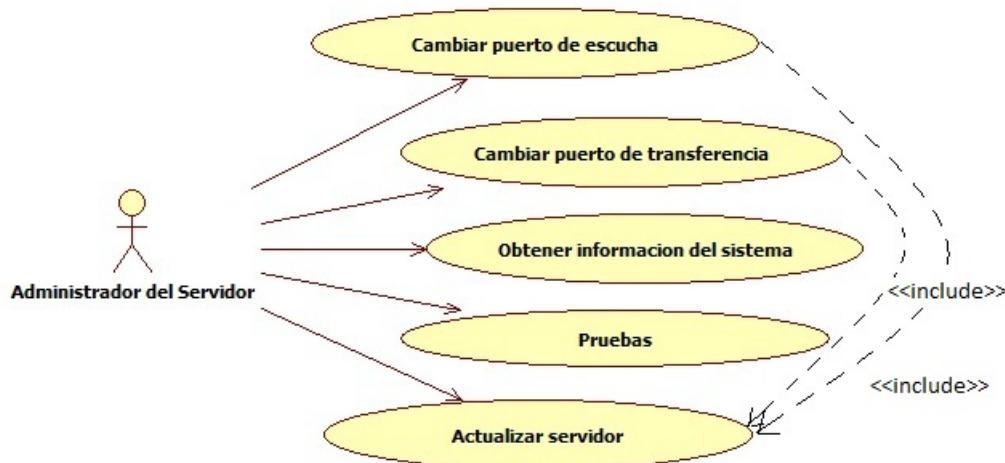


Figura 5.2: Caso de Uso Gestionar Servidor

Como podemos observar, el cliente, representado por el rol Administrador del Servidor, puede realizar las siguientes acciones:

- Cambiar puerto de escucha
- Cambiar puerto de transferencia
- Obtener información del sistema
- Pruebas
- Actualizar servidor.

Una vez que se cambie cualquiera de los puertos, se tendrá que actualizar el servidor, por ello la cláusula <<include>>. Procederemos a las descripciones de los casos de uso contenidos en este caso de uso “Gestionar Servidor”

#### 5.1.1.1. Descripción caso de uso: Cambiar puerto de escucha

*Descripción:* Se pide al sistema cambiar el puerto de escucha de conexiones entrantes.

*Resumen:* El Administrador del Servidor pide cambiar el puerto de escucha, el sistema procede a su petición.

*Actores:* Administrador del Servidor (Principal)

*Escenario principal:*

1. El Administrador del Servidor accede a la configuración de los puertos del sistema
2. El sistema muestra la configuración de los puertos del mismo
3. El Administrador del Servidor cambia el puerto de escucha y acepta los cambios.
4. El sistema registra el puerto de escucha.
5. El sistema actualiza el servidor con el nuevo puerto de escucha.

*Escenario alternativo:*

- \*a. El administrador cancela la petición de cambiar el puerto de escucha.

#### 5.1.1.2. Descripción caso de uso: Cambiar puerto de transferencia

*Descripción:* Se pide al sistema cambiar el puerto de transferencia para la transferencia de ficheros.

*Resumen:* El Administrador del Servidor pide cambiar el puerto de transferencia, el sistema procede a su petición.

*Actores:* Administrador del Servidor (Principal)

*Escenario principal:*

1. El Administrador del Servidor accede a la configuración de los puertos del sistema
2. El sistema muestra la configuración de los puertos del mismo
3. El Administrador del Servidor cambia el puerto de transferencia y acepta los cambios.
4. El sistema registra el puerto de transferencia.
5. El sistema actualiza el servidor con el nuevo puerto de transferencia.

*Escenario alternativo:*

- \*a. El administrador cancela la petición de cambiar el puerto de transferencia.

**5.1.1.3. Descripción caso de uso: Obtener información del sistema**

*Descripción:* Se pide al sistema la información de las interfaces de red (ips y macs).

*Resumen:* El Administrador del Servidor pide información sobre las interfaces de red del servidor.

*Actores:* Administrador del Servidor (Principal)

*Escenario principal:*

1. El Administrador del Servidor pide acceder a la información de las interfaces de red del sistema.
2. El sistema registra la petición y la acepta.
3. El sistema obtiene la información pedida.
4. El sistema muestra la información al Administrador del sistema.

*Escenario alternativo:*

\*a. El administrador cancela la petición de cambiar el puerto de transferencia.

4.a. El administrador selecciona una dirección ip distinta a la que el sistema predetermina.

4.a.1. El sistema actualiza la información ip que el administrador ha seleccionado.

4.b. El administrador selecciona una dirección mac distinta a la que el sistema predetermina.

4.b.1. El sistema actualiza la información mac que el administrador ha seleccionado.

**5.1.1.4. Descripción caso de uso: Pruebas**

*Descripción:* El administrador del Servidor pide la realización de distintas pruebas, las cuales pueden ser: discos, red, memoria, procesos, sistema operativo, orden libre y scripts.

*Resumen:* El Administrador del Servidor pide realizar una determinada prueba del sistema.

*Actores:* Administrador del Servidor (Principal)

*Escenario principal:*

1. El Administrador del Servidor pide realizar la prueba de disco.
2. El sistema registra la petición y la acepta.
3. El sistema obtiene la información pedida.
4. El sistema muestra la información obtenida al Administrador del sistema.

*Escenario alternativo:*

- \*a. El administrador cancela la petición de cambiar el puerto de transferencia.
  - 1.a.El Administrador del Servidor pide realizar la prueba de red.
  - 1.b.El Administrador del Servidor pide realizar la prueba de memoria.
  - 1.c.El Administrador del Servidor pide realizar la prueba de procesos.
  - 1.d.El Administrador del Servidor pide realizar la prueba de sistema operativo.
  - 1.e.El Administrador del Servidor pide realizar la prueba de orden libre.
    - 1.e.1 El sistema pide la orden libre a ejecutar
    - 1.e.2 El Administrador introduce la orden libre.
  - 1.f.El Administrador del Servidor pide realizar la prueba de scripts.
    - 1.f.1 El Sistema muestra la opcion de ver script.
    - 1.f.2 El Administrador del Servidor elige ver script.
    - 1.f.3 El sistema muestra los scripts.
    - 1.f.4 El Administrador del Servidor elige un script para ver.
  - 1.g.1 El Sistema muestra la opcion de ejecutar script.
  - 1.g.2 El Administrador del Servidor elige ejecutar script.
  - 1.g.3 El sistema muestra los scripts.
  - 1.g.4 El Administrador del Servidor elige un script para ejecutar.

**5.1.1.5. Descripción caso de uso: Actualizar servidor**

*Descripción:* Se pide al sistema la actualización del servidor.

*Resumen:* El Administrador del Servidor pide actualizar el servicio de espera de conexiones.

*Actores:* Administrador del Servidor (Principal)

*Escenario principal:*

1. El Administrador del Servidor pide actualizar el servidor (espera de conexiones).
2. El sistema se relanza con los parámetros actuales que posee.

*Escenario alternativo:*

- \*a. El administrador cancela la petición de actualizar servidor.

**5.1.2. Caso de Uso Operaciones del cliente**

El caso de uso “Operaciones del cliente” trata sobre las posibles acciones que la persona del dispositivo móvil, tomando parte como rol de cliente, puede realizar en el. En la siguiente figura podemos observar dicho diagrama:

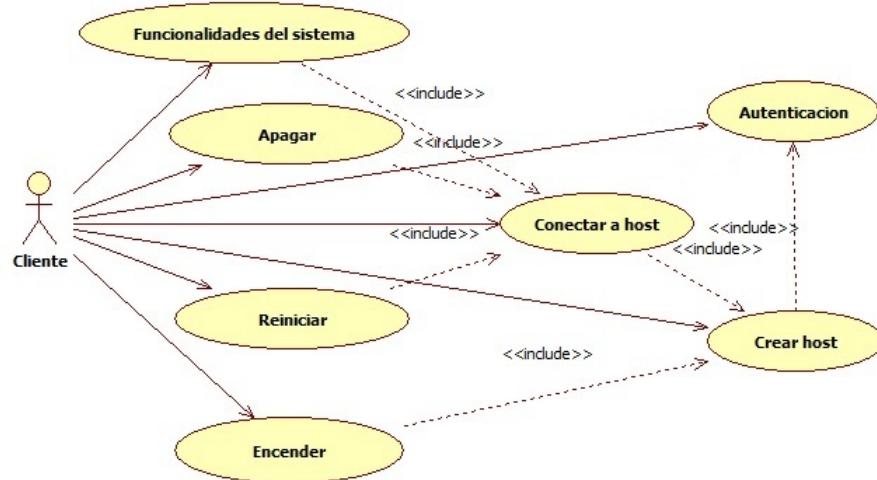


Figura 5.3: Caso de uso: Operaciones del cliente.

En ella podemos observar los diferentes casos de uso que engloba el caso de uso general “Operaciones del cliente”. Los casos de uso más específicos son:

- Funcionalidades del sistema: Este es un caso de uso general, donde englobará más operaciones del cliente que veremos a continuación.
- Apagar.
- Reiniciar.
- Encender.
- Conectar a un host.
- Autenticación.
- Crear un host.

Como podemos observar, para poder realizar una funcionalidad del sistema en la operación del cliente, reiniciar o apagar, se deberá anteriormente conectar a un host, por ello la cláusula <<include>>, es decir, para realizar cualquiera de los primeros es necesario estar conectado anteriormente a un host. Respectivamente, para conectarse a un host, se deberá de haber creado un host previamente, sin ello es imposible poderse conectar a un host, de ahí la cláusula <<include>> de conectar host hacia crear host. En consecuencia con ello para crear un host, antes es preciso logarse o autenticarse en el sistema con la finalidad de que los datos de conexión de un host sean privados en caso de pérdida del dispositivo

móvil, por ejemplo. La única diferencia es el caso de uso de encendido, el cual, como el host servidor está apagado y hay que encenderlo no estamos conectados a él. Procedemos a describir los casos de uso finales, es decir, aquellos que no se puede bajar más en su nivel de abstracción, más específicos.

#### **5.1.2.1. Descripción caso de uso: Apagar.**

*Descripción:* El cliente pide al sistema apagarlo.

*Resumen:* El cliente manda la orden de apagado al servidor y este se apaga.

*Actores:* Cliente (Principal)

*Escenario principal:*

1. El cliente se autentica.
2. El cliente se conecta al host (servidor).
3. El sistema registra la conexión.
4. El cliente envía la orden de apagado hacia el servidor.
5. El sistema recibe la orden de apagado.
6. El sistema se ejecuta la orden de apagado y se apaga.

*Escenario alternativo:*

\*a. El cliente cancela la petición de apagado.

\*b. Se pierde la conexión con el servidor.

#### **5.1.2.2. Descripción caso de uso: Reiniciar.**

*Descripción:* El cliente pide al sistema reiniciarse.

*Resumen:* El cliente manda la orden de reiniciar el servidor y este se reinicia.

*Actores:* Cliente (Principal)

*Escenario principal:*

1. El cliente se autentica.
2. El cliente se conecta al host (servidor).
3. El sistema registra la conexión.
4. El cliente envía la orden de reinicio hacia el servidor.
5. El sistema recibe la orden de reinicio.
6. El sistema se ejecuta la orden de reinicio y se reinicia.

*Escenario alternativo:*

\*a. El cliente cancela la petición de reinicio.

\*b. Se pierde la conexión con el servidor.

### 5.1.2.3. Descripción caso de uso: Encender

*Descripción:* El cliente pide al sistema encenderse.

*Resumen:* El cliente manda la orden de encender el servidor y éste se enciende.

*Actores:* Cliente (Principal)

*Escenario principal:*

1. El cliente se autentica.
2. El cliente crea un host con sus datos de conexión.
3. El sistema registra dicho host.
4. El cliente envía la orden de encendido remoto en la red del host servidor.
5. El sistema recibe la trama de red y se enciende.

*Escenario alternativo:*

- \*a. El cliente cancela la petición de encendido.

### 5.1.2.4. Descripción caso de uso: Conectar host

*Descripción:* El cliente pide conectarse a un determinado host.

*Resumen:* El cliente se conecta a un host el cual está esperando conexiones entrantes.

*Actores:* Cliente (Principal)

*Escenario principal:*

1. El cliente se autentica.
2. El cliente crea un host con los datos del él.
3. El sistema registra el host creado.
4. El cliente pide conectarse al host creado.
5. El sistema recibe la petición de conexión y la acepta.

*Escenario alternativo:*

- \*a. El cliente cancela la petición de conexión.

### 5.1.2.5. Descripción caso de uso: Crear host

*Descripción:* El cliente crea un nuevo host.

*Resumen:* El cliente procede a crear un nuevo host, registrándose en el sistema.

*Actores:* Cliente (Principal)

*Escenario principal:*

1. El cliente se autentica.
2. El cliente introduce el nombre del host.

3. El sistema registra el nombre del host.
4. El cliente introduce la ip del host.
5. El sistema registra la ip del host.
6. El cliente introduce el puerto del host.
7. El sistema registra el puerto del host.
8. El cliente introduce la dirección MAC del host.
9. El sistema registra la dirección MAC del host
10. El sistema crea el host con los datos obtenidos.

*Escenario alternativo:*

- \*a. El cliente cancela la petición de conexión.

#### **5.1.2.6. Descripción caso de uso: Autenticación**

*Descripción:* El cliente se autentica en el sistema

*Resumen:* El cliente crea un usuario y contraseña y se autentica en el sistema.

*Actores:* Cliente (Principal)

*Escenario principal:*

1. El cliente pide crear un usuario.
2. El sistema registra la petición de creación de usuario y la acepta.
3. El cliente introduce un nombre usuario.
4. El sistema registra el nombre de usuario.
5. El cliente introduce una contraseña.
6. El sistema registra la contraseña.
7. El cliente accede al sistema con el usuario y contraseña creadas.
8. El sistema acepta el acceso al sistema.

*Escenario alternativo:*

- \*a. El cliente cancela la petición autenticación.
- 4.a. El sistema ve que el nombre de usuario está repetido y lo comunica.
- 8.a. El usuario y la contraseña introducidas por el cliente no son correctas.
- 8.b. El sistema muestra el error.

### 5.1.3. Caso de Uso Funcionalidades del sistema

El caso de uso “Funcionalidades del sistema” es un caso de uso incluido en “Operaciones del cliente”, en el cual incluye estar conectados a un host. Estas funcionalidades van a ser las principales que vamos a poder realizar desde el cliente, aparte de las de reiniciar, apagar y encender, comentadas anteriormente. Procedemos a ver su diagrama y las descripciones de los casos de uso que sean finales.

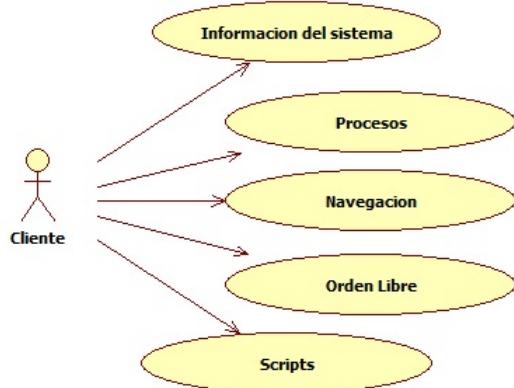


Figura 5.4: Caso de uso: Funcionalidades del sistema.

Aquí podemos observar que el caso de uso general “Funcionalidades del sistema” incluye otros casos de uso, también generales y otros más específicos. Los cuales son:

- Información del sistema.
- Procesos.
- Navegación.
- Orden Libre.
- Scripts.

De todos ellos el que pasamos a describir que es un caso de uso final o específico es el del orden libre.

### 5.1.3.1. Descripción caso de uso: Orden Libre

*Descripción:* El cliente manda una orden libre a ejecutar.

*Resumen:* El cliente procede enviar una orden libre al sistema, ejecutandolo y enviando el resultado al cliente.

*Actores:* Cliente (Principal)

*Escenario principal:*

1. El cliente se autentica.
2. El cliente se conecta al host.
3. El cliente introduce la orden a ejecutar.
4. El sistema registra la orden.
5. El cliente envía la orden al host.
6. El sistema registra la petición y la envía.
7. El host de recepción ejecuta la orden y registra la salida.
8. El host de recepción envía la salida al cliente.
9. El cliente recibe la salida de la orden.

*Escenario alternativo:*

- \*a. El cliente cancela la orden.
- \*b. Se pierde la conexión.

### 5.1.4. Caso de Uso Información del sistema

El caso de uso “Información el sistema” es un caso de uso general, en el cual podemos encontrar los casos de uso específicos que podemos ver en la siguiente figura:

Como podemos ver el caso de uso Información del sistema podemos encontrar los siguientes casos de uso más específicos:

- Información de discos.
- Información de red.
- Información del sistema operativo.
- Información de memoria.

Pasamos a ver las descripciones de ellos.

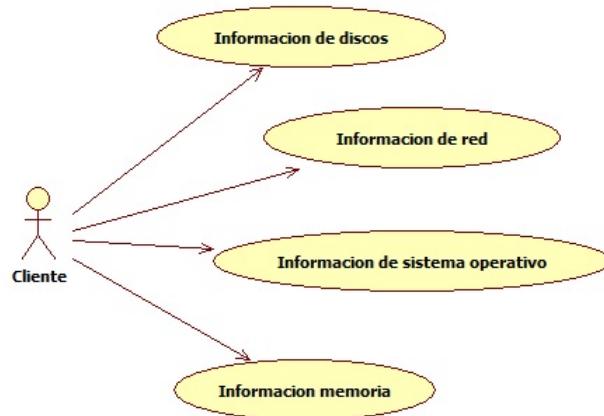


Figura 5.5: Caso de uso: Información del sistema.

#### 5.1.4.1. Descripción caso de uso: Información de discos.

*Descripción:* El cliente manda la orden para obtener la información de los discos.

*Resumen:* El cliente procede enviar una orden para obtener la información de los discos, recibiendo la respuesta del host servidor.

*Actores:* Cliente (Principal)

*Escenario principal:*

1. El cliente se autentica.
2. El cliente se conecta al host.
3. El cliente manda la orden de información de discos.
4. El sistema registra la orden.
5. El cliente envía la orden al host.
6. El sistema registra la petición y la envía.
7. El host de recepción ejecuta la orden y obtiene la información pedida.
8. El host de recepción envía la salida al cliente.
9. El cliente recibe la salida de la orden.

*Escenario alternativo:*

- \*a. El cliente cancela la orden.
- \*b. Se pierde la conexión.

**5.1.4.2. Descripción caso de uso: Información de red**

*Descripción:* El cliente manda la orden para obtener la información de las interfaces de red.

*Resumen:* El cliente procede enviar una orden para obtener la información de las interfaces de la red , enviándose el resultado de vuelta al cliente.

*Actores:* Cliente (Principal)

*Escenario principal:*

1. El cliente se autentica.
2. El cliente se conecta al host.
3. El cliente manda la orden de información de red.
4. El sistema registra la orden.
5. El cliente envía la orden al host.
6. El sistema registra la petición y la envía.
7. El host de recepción ejecuta la orden y obtiene la información pedida.
8. El host de recepción envía la salida al cliente.
9. El cliente recibe la salida de la orden.

*Escenario alternativo:*

- \*a. El cliente cancela la orden.
- \*b. Se pierde la conexión.

**5.1.4.3. Descripción caso de uso: Información de sistema operativo.**

*Descripción:* El cliente manda la orden para obtener la información del sistema operativo.

*Resumen:* El cliente procede enviar una orden para obtener la información del sistema operativo, enviándose el resultado de vuelta al cliente.

*Actores:* Cliente (Principal)

*Escenario principal:*

1. El cliente se autentica.
2. El cliente se conecta al host.
3. El cliente manda la orden de información del sistema operativo.
4. El sistema registra la orden.
5. El cliente envía la orden al host.
6. El sistema registra la petición y la envía.
7. El host de recepción ejecuta la orden y obtiene la información pedida.

8. El host de recepción envía la salida al cliente.
9. El cliente recibe la salida de la orden.

*Escenario alternativo:*

- \*a. El cliente cancela la orden.
- \*b. Se pierde la conexión.

#### 5.1.4.4. Descipción caso de uso: Información de memoria.

*Descripción:* El cliente manda la orden para obtener la información de la memoria del host servidor.

*Resumen:* El cliente procede enviar una orden para obtener la información de la memoria, enviándose el resultado de vuelta al cliente.

*Actores:* Cliente (Principal)

*Escenario principal:*

1. El cliente se autentica.
2. El cliente se conecta al host.
3. El cliente manda la orden de información de la memoria.
4. El sistema registra la orden.
5. El cliente envía la orden al host.
6. El sistema registra la petición y la envía.
7. El host de recepción ejecuta la orden y obtiene la información pedida.
8. El host de recepción envía la salida al cliente.
9. El cliente recibe la salida de la orden.

*Escenario alternativo:*

- \*a. El cliente cancela la orden.
- \*b. Se pierde la conexión.

#### 5.1.5. Caso de Uso Procesos

El caso de uso “Procesos” es un caso de uso general el cual puede verse en el siguiente diagrama:

En este diagrama podemos observar los diferentes casos de uso que componen el caso de uso general “Procesos”. Dicho diagrama está compuesto por:

- Listar procesos.
- Eliminar proceso.

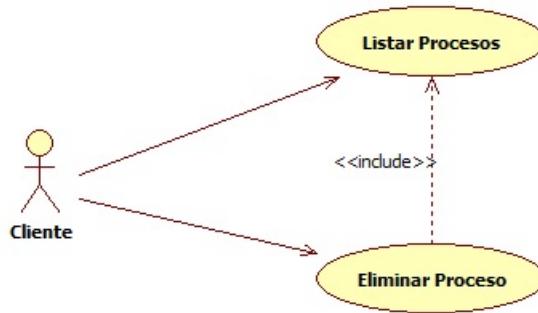


Figura 5.6: Caso de uso: Procesos.

El caso de uso “Eliminar proceso” incluye al caso de uso “Listar procesos”, puesto que para eliminar un proceso determinado, debemos conocer la lista de procesos que actualmente están ejecutándose en el sistema. Pasamos a describir dichos casos de uso:

#### 5.1.5.1. Descripción caso de uso: Listar procesos.

*Descripción:* El cliente manda la orden para obtener la información del listado de procesos del host servidor.

*Resumen:* El cliente procede enviar una orden para obtener el listado de procesos del host servidor, enviándose el resultado de vuelta al cliente.

*Actores:* Cliente (Principal)

*Escenario principal:*

1. El cliente se autentica.
2. El cliente se conecta al host.
3. El cliente manda la orden de listado de procesos.
4. El sistema registra la orden.
5. El host de recepción ejecuta la orden y obtiene la información pedida.
6. El host de recepción envía la salida al cliente.
7. El cliente recibe la salida de la orden.

*Escenario alternativo:*

- \*a. El cliente cancela la orden.
- \*b. Se pierde la conexión.

### 5.1.5.2. Descripción caso de uso: Eliminar proceso.

*Descripción:* El cliente manda la orden para obtener la información de la memoria del host servidor.

*Resumen:* El cliente procede enviar una orden para obtener la información de la memoria, enviándose el resultado de vuelta al cliente.

*Actores:* Cliente (Principal)

*Escenario principal:*

1. Se lista los procesos del sistema host remoto (<<include>> Listar Procesos).
2. El cliente selecciona un proceso a eliminar.
3. El sistema registra la selección del proceso.
4. El cliente confirma la eliminación del proceso.
5. El sistema registra la confirmación y manda la orden para la eliminación.
6. Se lista de nuevo los procesos del sistema host remoto actualizado.

*Escenario alternativo:*

\*a. El cliente cancela la eliminación del proceso.

\*b. Se pierde la conexión.

### 5.1.6. Caso de Uso Navegación

El caso de uso “Navegación” es un caso de uso general en donde encontramos casos de uso más específicos, siendo estos los siguientes:

- Crear Fichero.
- Cortar/Copiar Fichero.
- Pegar Fichero.
- Renombrar Fichero.
- Eliminar Fichero.
- Mostrar directorio atrás.
- Mostrar directorio adelante.
- Transferir fichero.

El diagrama del caso de uso Navegación quedaría como sigue:

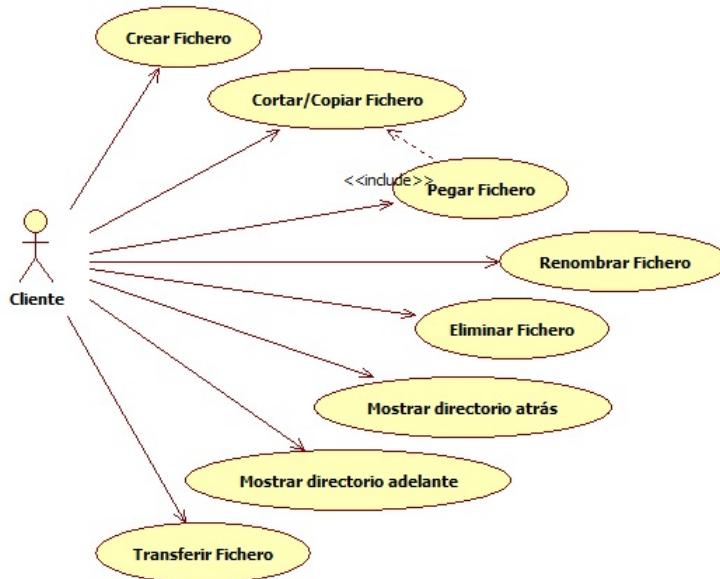


Figura 5.7: Diagrama casos de uso Navegación

A continuación paso a describir los distintos casos de uso del caso de uso más general “Navegación”:

#### 5.1.6.1. Descripción caso de uso: Crear fichero

*Descripción:* El cliente manda la orden para crear al fichero al host servidor y éste lo crea.

*Resumen:* El cliente procede enviar una orden para crear un fichero con su contenido, el host servidor lo recibe y lo crea.

*Actores:* Cliente (Principal)

*Escenario principal:*

1. El cliente se conecta al host (servidor).
2. El sistema acepta la conexión y la registra.
3. El cliente pide iniciar la navegación por el árbol de directorios.
4. El sistema registra la petición y la acepta.
5. El cliente pide crear un fichero.
6. El sistema registra la petición.

7. El cliente introduce el nombre del fichero.
8. El sistema registra el nombre del fichero.
9. El cliente introduce el contenido del fichero.
10. El sistema registra el nombre del fichero.
11. El cliente acepta y envía los datos.
12. El sistema registra los datos y crea el fichero con el nombre y el contenido.

*Escenario alternativo:*

- \*a. El cliente cancela la creación del fichero.
- \*b. Se pierde la conexión.

#### **5.1.6.2. Descipción caso de uso: Cortar/Copiar fichero**

*Descripción:* El cliente manda la orden para cortar/copiar fichero.

*Resumen:* El cliente procede enviar una orden para cortar/copiar un fichero.

*Actores:* Cliente (Principal)

*Escenario principal:*

1. El cliente se conecta al host (servidor).
2. El sistema acepta la conexión y la registra.
3. El cliente pide iniciar la navegación por el árbol de directorios.
4. El sistema registra la petición y la acepta.
5. El cliente pide cortar un fichero.
6. El sistema registra la ruta y el fichero cortado.

*Escenario alternativo:*

- \*a. El cliente cancela el copiado o cortado del fichero.
- \*b. Se pierde la conexión.
- 5.a. El cliente pide copiar un fichero.
- 5.a.1. El sistema registra la ruta y el fichero copiado.

#### **5.1.6.3. Descipción caso de uso: Pegar fichero**

*Descripción:* El cliente manda la orden para pegar fichero.

*Resumen:* El cliente procede enviar una orden para pegar un fichero.

*Actores:* Cliente (Principal)

*Escenario principal:*

1. El cliente ha cortado un fichero. ( <<include>> Copiar/Pegar fichero)
2. El cliente pide pegar el fichero anteriormente cortado.

3. El sistema registra la ruta donde se quiere pegar.
4. El cliente envía orden para pegar el fichero en la ruta indicada.

*Escenario alternativo:*

- \*a. El cliente cancela el copiado o cortado del fichero.
  - \*b. Se pierde la conexión.
- 1.a. El cliente ha copiado un fichero.
  - 5.a.1. El cliente pide pegar el fichero anteriormente copiado.

#### **5.1.6.4. Descripción caso de uso: Renombrar fichero**

*Descripción:* El cliente manda la orden para renombrar fichero.

*Resumen:* El cliente procede enviar una orden para renombrar un fichero.

*Actores:* Cliente (Principal)

*Escenario principal:*

1. El cliente se conecta al host (servidor).
2. El sistema acepta la conexión y la registra.
3. El cliente pide iniciar la navegación por el árbol de directorios.
4. El sistema registra la petición y la acepta.
5. El cliente selecciona un fichero a renombrar.
6. El sistema registra el fichero a renombrar.
7. El cliente introduce el nombre nuevo de fichero.
8. El sistema registra el nombre nuevo del fichero.
9. El cliente manda la orden al servidor.
10. El sistema remoto registra la orden y renombra el fichero.

*Escenario alternativo:*

- \*a. El cliente cancela el copiado o cortado del fichero.
- \*b. Se pierde la conexión.

#### **5.1.6.5. Descripción caso de uso: Eliminar fichero**

*Descripción:* El cliente manda la orden para eliminar fichero.

*Resumen:* El cliente procede enviar una orden para eliminar un fichero.

*Actores:* Cliente (Principal)

*Escenario principal:*

1. El cliente se conecta al host (servidor).
2. El sistema acepta la conexión y la registra.

3. El cliente pide iniciar la navegación por el árbol de directorios.
4. El sistema registra la petición y la acepta.
5. El cliente selecciona un fichero a eliminar.
6. El sistema registra el fichero a eliminar.
7. El cliente manda la orden al servidor.
8. El sistema registra la orden y elimina el fichero.

*Escenario alternativo:*

- \*a. El cliente cancela el copiado o cortado del fichero.
- \*b. Se pierde la conexión.

#### **5.1.6.6. Descipción caso de uso: Mostrar directorio atrás**

*Descripción:* El cliente manda la orden para mostrar el directorio atrás.

*Resumen:* El cliente procede enviar una orden para mostrar el directorio atrás.

*Actores:* Cliente (Principal)

*Escenario principal:*

1. El cliente se conecta al host (servidor).
2. El sistema acepta la conexión y la registra.
3. El cliente pide iniciar la navegación por el árbol de directorios.
4. El sistema registra la petición y la acepta.
5. El cliente pide mostrar el directorio atrás.
6. El sistema registra la petición, actualizando la ruta y el contenido del directorio.

*Escenario alternativo:*

- \*a. El cliente cancela el mostrar el directorio atrás.
- \*b. Se pierde la conexión.

#### **5.1.6.7. Descipción caso de uso: Mostrar directorio adelante**

*Descripción:* El cliente manda la orden para mostrar el directorio adelante.

*Resumen:* El cliente procede enviar una orden para mostrar el directorio adelante.

*Actores:* Cliente (Principal)

*Escenario principal:*

1. El cliente se conecta al host (servidor).
2. El sistema acepta la conexión y la registra.

3. El cliente pide iniciar la navegación por el árbol de directorios.
4. El sistema registra la petición y la acepta.
5. El cliente pide mostrar el directorio adelante.
6. El sistema registra la petición, actualizando la ruta y el contenido del directorio.

*Escenario alternativo:*

- \*a. El cliente cancela el mostrar el directorio adelante.
- \*b. Se pierde la conexión.

#### **5.1.6.8. Descripción caso de uso: Transferir fichero**

*Descripción:* El cliente manda la orden para transferir un fichero.

*Resumen:* El cliente procede enviar una orden para transferir un fichero.

*Actores:* Cliente (Principal)

*Escenario principal:*

1. El cliente se conecta al host (servidor).
2. El sistema acepta la conexión y la registra.
3. El cliente pide iniciar la navegación por el árbol de directorios.
4. El sistema registra la petición y la acepta.
5. El cliente selecciona un fichero a transferir.
6. El sistema registra la ruta y el nombre del fichero y transfiere el fichero al cliente.

*Escenario alternativo:*

- \*a. El cliente cancela el mostrar el directorio adelante.
- \*b. Se pierde la conexión.

#### **5.1.7. Caso de Uso Scripts**

El diagrama del caso de uso, lo podemos ver en la siguiente figura:

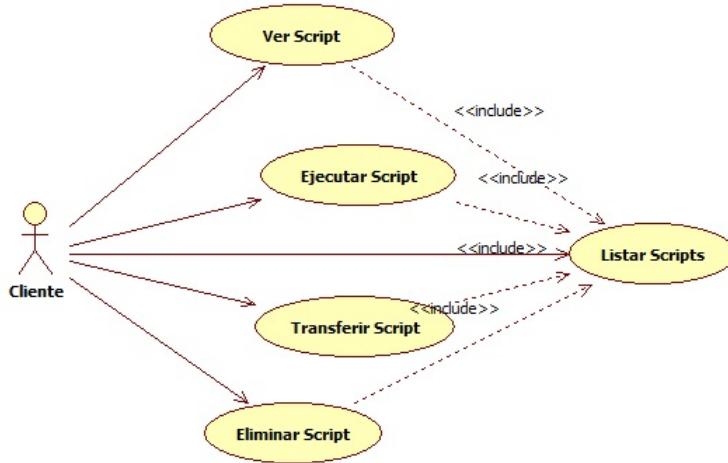


Figura 5.8: Diagrama de caso de uso Scripts

En el podemos encontrar los casos de uso específicos como son : “Ver Script”, “Ejecutar Script”, “Transferir Script”, “Eliminar Script” y “Listar Scripts”, incluyendo los primeros a éste último, ya que para realizarlos deben de haberse listado antes para proceder a su selección. Pasamos a las descripciones de los casos de uso más específicos.

#### 5.1.7.1. Descripción caso de uso: Listar Scripts

*Descripción:* El cliente manda la orden para listar los scripts.

*Resumen:* El cliente procede enviar una orden para listar los scripts y el sistema registra la petición y los lista.

*Actores:* Cliente (Principal)

*Escenario principal:*

1. El cliente se conecta al host (servidor).
2. El sistema acepta la conexión y la registra.
3. El cliente pide listar los scripts contenidos en el directorio.
4. El sistema registra la petición y ejecuta la orden de listado de scripts.
5. El sistema devuelve la salida de la orden al cliente.

*Escenario alternativo:*

- \*a. El cliente cancela el listado de scripts.
- \*b. Se pierde la conexión.

### 5.1.7.2. Descripción caso de uso: Ver Script

*Descripción:* El cliente manda la orden para ver un determinado script.

*Resumen:* El cliente procede enviar una orden para ver un script, el sistema lo registra y manda el contenido de dicho script.

*Actores:* Cliente (Principal)

*Escenario principal:*

1. El cliente ha listado los scripts que contienen la carpeta del servidor.  
(<<include>> Listar Scripts).
2. El cliente selecciona un script para ver.
3. El sistema registra el script seleccionado y manda la orden para ver su contenido.
4. El sistema devuelve el contenido del script seleccionado al cliente.

*Escenario alternativo:*

\*a. El cliente cancela el listado de scripts.

\*b. Se pierde la conexión.

### 5.1.7.3. Descripción caso de uso: Ejecutar Script

*Descripción:* El cliente manda la orden para ejecutar un script.

*Resumen:* El cliente procede enviar una orden para ejecutar un script, el sistema registra la petición y la ejecuta.

*Actores:* Cliente (Principal)

*Escenario principal:*

1. El cliente ha listado los scripts que contienen la carpeta del servidor.  
(<<include>> Listar Scripts).
2. El cliente selecciona un script para ejecutar.
3. El sistema registra el script seleccionado y manda la orden para ejecutarlo.

*Escenario alternativo:*

\*a. El cliente cancela la ejecución del script.

\*b. Se pierde la conexión.

### 5.1.7.4. Descripción caso de uso: Transferir Script

*Descripción:* El cliente manda la orden para transferir un script.

*Resumen:* El cliente procede enviar una orden para transferir un script, el sistema registra la petición y pide nombre y contenido, creándose el script en el servidor.

*Actores:* Cliente (Principal)

*Escenario principal:*

1. El cliente ha listado los scripts que contienen la carpeta del servidor.  
(<<include>> Listar Scripts).
2. El cliente pide transferir un script.
3. El sistema registra la petición.
4. El cliente introduce nombre del script.
5. El sistema registra el nombre del script.
6. El cliente introduce el contenido del script que se va a transferir.
7. El sistema registra el contenido del script a transferir.
8. El sistema envía la orden de creación del script junto con su nombre y contenido, creándose en el servidor.

*Escenario alternativo:*

- \*a. El cliente cancela la transferencia del script.
- \*b. Se pierde la conexión.

#### 5.1.7.5. Descripción caso de uso: Eliminar Scripts

*Descripción:* El cliente manda la orden para eliminar un script.

*Resumen:* El cliente procede enviar una orden para eliminar un script, el sistema registra la petición y la ejecuta.

*Actores:* Cliente (Principal)

*Escenario principal:*

1. El cliente ha listado los scripts que contienen la carpeta del servidor.  
(<<include>> Listar Scripts).
2. El cliente selecciona un script para ejecutar.
3. El sistema registra el script seleccionado y manda la orden para eliminarlo.

*Escenario alternativo:*

- \*a. El cliente cancela la eliminación del script.
- \*b. Se pierde la conexión.

## 5.2. Modelo conceptual de datos

El modelado conceptual de los datos nos va a permitir modelar los requisitos descritos usando las estructuras de datos necesarias y las relaciones entre ellos. La técnica que utilizaremos para realizar el modelado es un diagrama de clases. Este modelado incluye: clases, asociaciones, atributos, y las restricciones que se acuerdan relacionado con dicho modelo.

Pasamos a dar una breve definición de asociación que es la parte fundamental de dicho modelado. Una asociación es una relación entre dos conceptos que indica

alguna conexión significativa entre ellos. En UML se describen como relaciones estructurales entre clases de objetos. Es necesario identificar las asociaciones de los conceptos que se requieren para satisfacer los requerimientos de información de los casos de uso en cuestión y los que contribuyen a entender el modelo conceptual. Dicho modelo conceptual lo podemos ver en la siguiente figura:

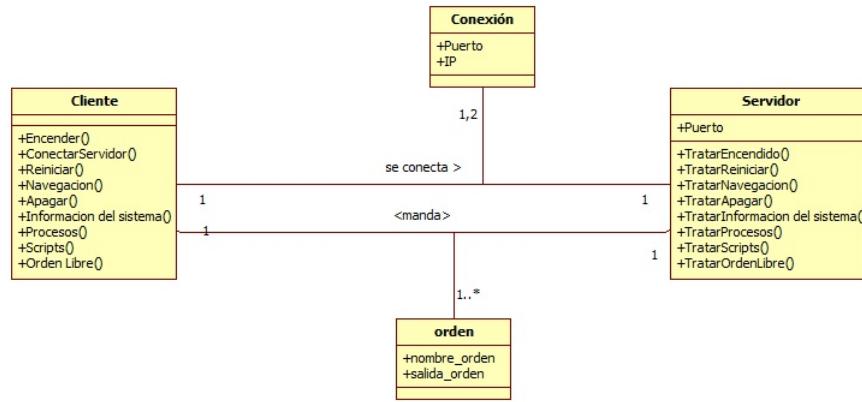


Figura 5.9: Modelo Conceptual de datos

La única restricción sobre este modelo es que el cliente y el servidor, se podrán comunicar (mandar órdenes), cuando previamente se haya producido una conexión entre ellos.

Paso a describir brevemente dicho modelo conceptual, sus asociaciones y multiplicidades. Las 4 clases principales van a ser el cliente, el servidor, la conexión y las órdenes. La clase cliente, tendrá los métodos que hemos podido ver en los casos de uso, estos son: encender, conectar a servidor, reiniciar, apagar, informacion del sistema, procesos, scripts y orden libre. Un cliente se conecta a un servidor, mediante 1 o 2 conexiones (una para el control de la aplicación y funcionalidades esenciales y otra para la transferencia de ficheros). El cliente manda 1 o más órdenes al servidor , y , recíprocamente el servidor manda órdenes al cliente, en este caso será la información obtenida mediante la orden, es decir, la salida de la orden. El servidor, una vez recibida la orden, la tratará y dependiendo de ella, ejecutará unos procedimientos u otros.

### 5.3. Modelo de comportamiento

El modelo de comportamiento indica la forma en que el software responderá a los eventos o estímulos externos. Las técnicas que aquí se utilizan son

los diagramas de secuencia y el contrato de las operaciones. Los diagramas de secuencia nos muestran la interacción entre los actores y el sistema y identifican las operaciones. Los contratos permiten describir los efectos que tienen las operaciones. Procedemos a dar el diagrama de secuencia del sistema junto con sus contratos de operaciones.

### 5.3.1. Caso de uso Cambiar puerto de escucha

Diagrama de secuencia

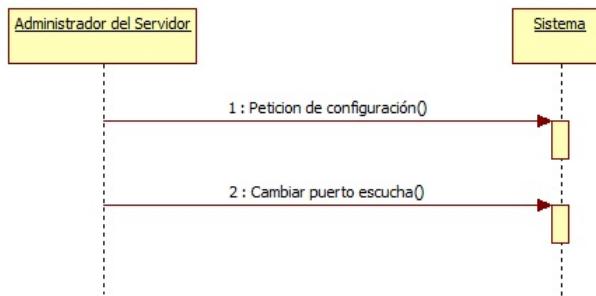


Figura 5.10: Diagrama de secuencia “Cambiar puerto de escucha”

Operación: Peticion\_de\_configuracion()

Responsabilidad: Dar acceso al cliente a la configuración del servidor.

Precondiciones: Se debe haber iniciado la aplicación servidor.

Postcondiciones: Se accede a la configuración del puerto de escucha del servidor.

Operación: Cambiar\_puerto\_escucha(int Puerto)

Responsabilidad: Cambiar el puerto de escucha del servidor.

Precondiciones: Debe de haberse accedido a la configuración del servidor.

Postcondiciones: Se actualiza el valor del puerto de escucha al de int\_Puerto.

### 5.3.2. Caso de uso Cambiar puerto de transferencia

Diagrama de secuencia

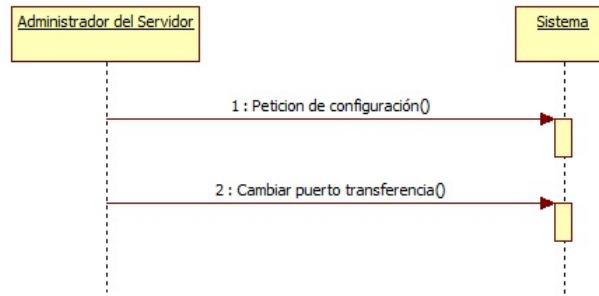


Figura 5.11: Diagrama de secuencia “Cambiar puerto de transferencia”

Operación: Peticion\_de\_configuracion()

Responsabilidad: Dar acceso al cliente a la configuración del servidor.

Precondiciones: Se debe haber iniciado la aplicación servidor.

Postcondiciones: Se accede a la configuración del puerto de escucha del servidor.

Operación: Cambiar\_puerto\_transferencia(int Puerto)

Responsabilidad: Cambiar el puerto de transferencia del servidor.

Precondiciones: Debe de haberse accedido a la configuración del servidor.

Postcondiciones: Se actualiza el valor del puerto de transferencia al de int\_Puerto.

### 5.3.3. Caso de uso Obtener información del sistema

Diagrama de secuencia

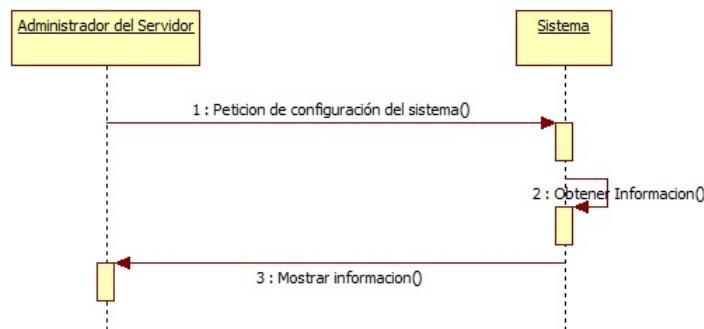


Figura 5.12: Diagrama de secuencia “Informacion del sistema”

Operación: Peticion\_de\_configuracion\_del\_sistema()

Responsabilidad: Dar acceso al cliente a la configuración de las interfaces de red.

Precondiciones: Se debe haber iniciado la aplicación servidor.

Postcondiciones: Se accede a la configuración de las interfaces de red del servidor.

Operación: Obtener\_información()

Responsabilidad: Obtener información de las interfaces de red del sistema (IPs y MACs)

Precondiciones: Debe de haberse accedido a la configuración del servidor.

Postcondiciones: Se obtiene la información de las ips y macs del sistema.

Operación: Mostrar\_información()

Responsabilidad: Muestra la información de las interfaces de red del servidor al administrador del servidor.

Precondiciones: Debe de haberse accedido a la configuración del servidor.

Postcondiciones: Se visualiza la información de las interfaces de red.

### 5.3.4. Caso de uso Pruebas

Diagrama de secuencia

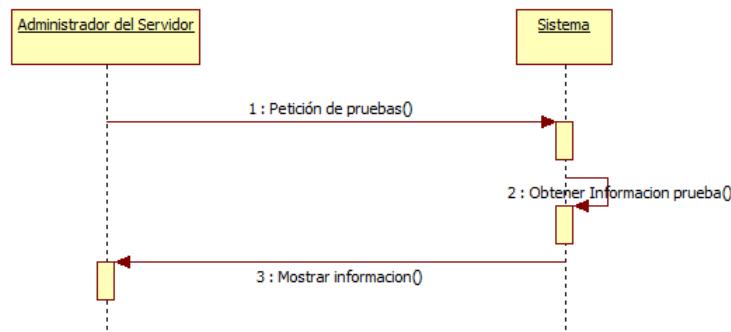


Figura 5.13: Diagrama de secuencia “Pruebas”

Operación: Peticion\_de\_configuracion\_del\_sistema()

Responsabilidad: Dar acceso al administrador del servidor a las pruebas.

Precondiciones: Se debe haber iniciado la aplicación servidor.

Postcondiciones: Se accede a las pruebas del servidor.

Operación: Obtener\_información\_prueba(id\_prueba)

Responsabilidad: Obtener información de la prueba seleccionada.

Precondiciones: Debe de haberse accedido al menu pruebas del servidor.

Postcondiciones: Se obtiene la información de la prueba con número identificativo id\_prueba.

Operación: Mostrar\_información()

Responsabilidad: Muestra la información de la prueba.

Precondiciones: Debe de haberse accedido al menu pruebas.

Postcondiciones: Se visualiza la información de la prueba del servidor.

### 5.3.5. Caso de uso Actualizar Servidor

Diagrama de secuencia

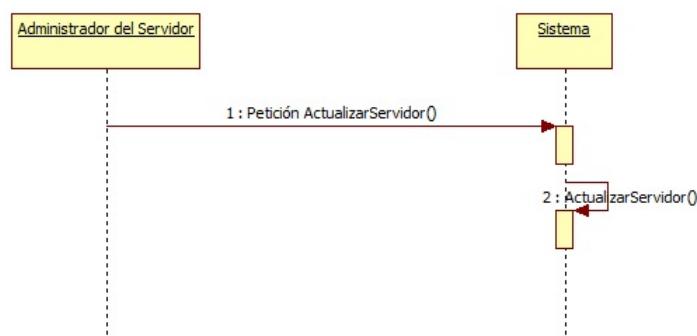


Figura 5.14: Diagrama de secuencia “Actualizar Servidor”

Operación: Peticion\_Actualizar\_Servidor(puerto\_escucha, puerto\_transferencia, ip, mac, ip\_local)

Responsabilidad: Mandar petición para actualizar el servidor con nuevos datos

Precondiciones: Se debe haber iniciado la aplicación servidor.

Postcondiciones: Se manda al sistema los valores para actualizar el servidor.

Operación: ActualizarServidor(puerto\_escucha, puerto\_transferencia, ip, mac, ip\_local)

Responsabilidad: Actualizar el servidor con los nuevos datos.

Precondiciones: Debe de haberse realizado una petición de actualización por parte del administrador del sistema.

Postcondiciones: Se actualiza el servidor con los valores de puerto\_escucha, puerto\_transferencia, ip, mac e ip\_local.

### 5.3.6. Caso de uso Apagar

Diagrama de secuencia

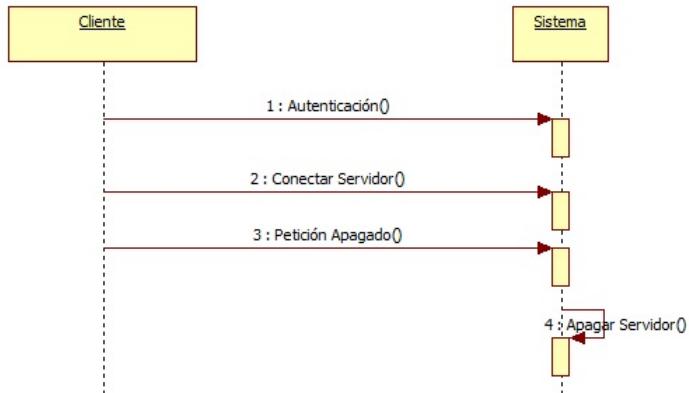


Figura 5.15: Diagrama de secuencia “Apagar”

Operación: Autenticación(nombre,ctsn)

Responsabilidad: Acceder al sistema

Precondiciones: Debe existir un usuario con nombre (nombre) y contraseña (ctsn).

Postcondiciones: Se accede al sistema con las credenciales nombre y ctsn.

Operación: Conectar\_Servidor(ip,puerto)

Responsabilidad: Establecer la conexión con el servidor.

Precondiciones: Debe de haber un cliente autenticado en el sistema.

Postcondiciones: Se inicia la conexión, creándose una instancia de conectar(ip,puerto).

Operación: PeticionApagado(cod\_orden)

Responsabilidad: Manda una orden para pedir al sistema que se apague.

Precondiciones: El cliente debe estar conectado al sistema.

Postcondiciones: El sistema gestiona la orden y manda la orden(cod\_orden) al sistema.

Operación: Apagar\_Servidor()

Responsabilidad: Manda una orden para pedir al sistema que se apague.

Precondiciones: El cliente debe estar conectado al sistema y haberse mandado la orden desde el cliente con código(cod\_orden).

Postcondiciones: El sistema gestiona la orden con cod\_orden y la ejecuta.

### 5.3.7. Caso de uso Reiniciar

Diagrama de secuencia

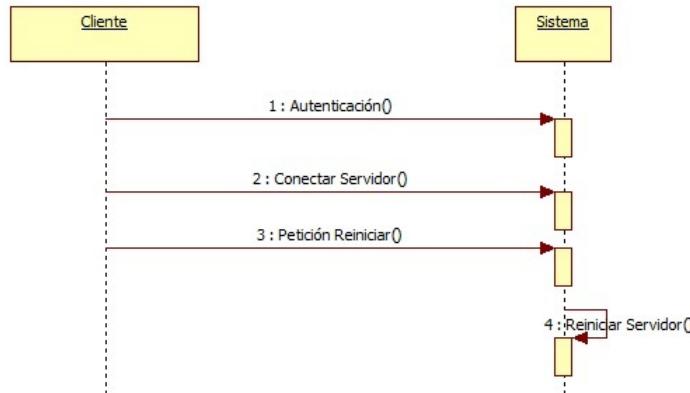


Figura 5.16: Diagrama de secuencia Reiniciar

Operación: Autenticación(nombre,ctsn)

Responsabilidad: Acceder al sistema

Precondiciones: Debe existir un usuario con nombre (nombre) y contraseña (ctsn).

Postcondiciones: Se accede al sistema con las credenciales nombre y ctsn.

Operación: Conectar\_Servidor(ip,puerto)

Responsabilidad: Establecer la conexión con el servidor.

Precondiciones: Debe de haber un cliente autenticado en el sistema.

Postcondiciones: Se inicia la conexión, creándose una instancia de conectar(ip,puerto).

Operación: Peticion\_Reiniciar(cod\_orden)

Responsabilidad: Manda una orden para pedir al sistema que se reinicie.

Precondiciones: El cliente debe estar conectado al sistema.

Postcondiciones: El sistema gestiona la orden y manda la orden(cod\_orden) al sistema.

Operación: Reiniciar\_Servidor()

Responsabilidad: Manda una orden para pedir al sistema que sereinicie..

Precondiciones: El cliente debe estar conectado al sistema y haberse mandado la orden desde el cliente con código(cod\_orden).

Postcondiciones: El sistema gestiona la orden con cod\_orden, resultando que es reiniciar y la ejecuta.

### 5.3.8. Caso de uso Encender

Diagrama de secuencia

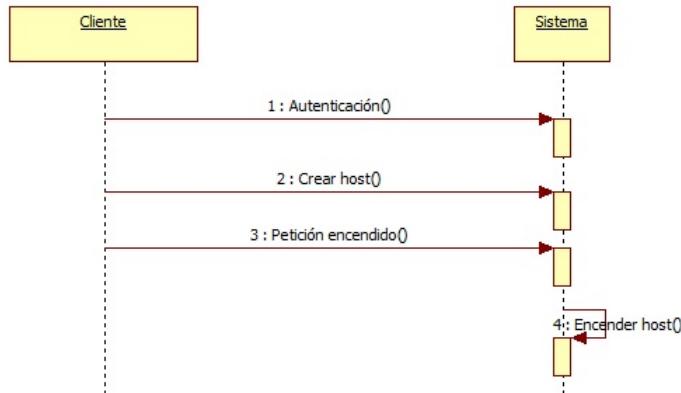


Figura 5.17: Diagrama de secuencia “Encender”

Operación: Autenticación(nombre,ctsn)

Responsabilidad: Acceder al sistema

Precondiciones: Debe existir un usuario con nombre (nombre) y contraseña (ctsn).

Postcondiciones: Se accede al sistema con las credenciales nombre y ctsn.

Operación: Crear\_host(nombre,ip,puerto,mac)

Responsabilidad: Crear un host.

Precondiciones: Debe de haber un cliente autenticado en el sistema.

Postcondiciones: Se inicia crea una instancia host con los datos de(nombre,ip,puerto,mac).

Operación: Peticion\_Encendido(cod\_orden)

Responsabilidad: Manda una orden para pedir al sistema que se inicie..

Precondiciones: Debe de existir un host el cual encender.

Postcondiciones: El sistema gestiona la orden y manda la orden(cod\_orden) al sistema.

Operación: Encender\_host()

Responsabilidad: Manda una orden para pedir al sistema que se inicie..

Precondiciones: Debe de existir un host y haberse mandado la orden desde el cliente con codigo(cod\_orden).

Postcondiciones: El sistema gestiona la orden con cod\_orden, resultando que es iniciar y lo inicia.

### 5.3.9. Caso de uso Conectar host

Diagrama de secuencia

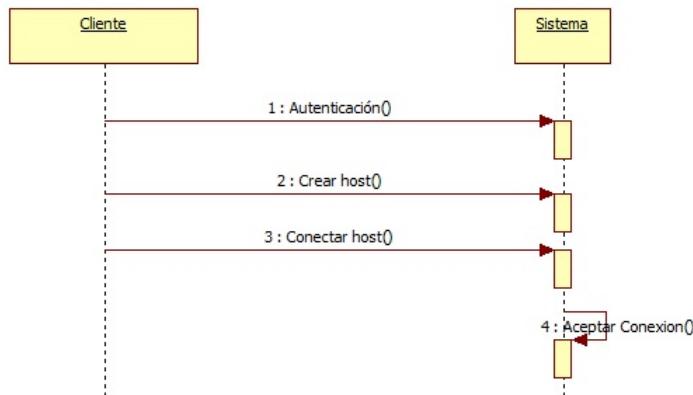


Figura 5.18: Diagrama de secuencia “Coneectar host”

Operación: Autenticación(nombre,ctsn)

Responsabilidad: Acceder al sistema

Precondiciones: Debe existir un usuario con nombre (nombre) y contraseña (ctsn).

Postcondiciones: Se accede al sistema con las credenciales nombre y ctsn.

Operación: Crear\_host(nombre,ip,puerto,mac)

Responsabilidad: Crear un host.

Precondiciones: Debe de haber un cliente autenticado en el sistema.

Postcondiciones: Se inicia crea una instancia host con los datos de(nombre,ip,puerto,mac).

Operación: Coneectar\_host(ip,puerto)

Responsabilidad: Establecer la conexión con el servidor.

Precondiciones: Debe de haber un cliente autenticado en el sistema.

Postcondiciones: Se inicia la conexión, creándose una instancia de conectar(ip,puerto).

Operación: Aceptar\_Conexión()

Responsabilidad: Aceptar conexión.

Precondiciones: Debe de existir una petición de conexión en la ip y puerto del servidor.

Postcondiciones: El sistema recibe la petición de conexión y la acepta. Se crea el una instancia de la clase de asociación Conexión entre el cliente y servidor con ip y puerto respectivos.

### 5.3.10. Caso de uso Crearhost

Diagrama de secuencia

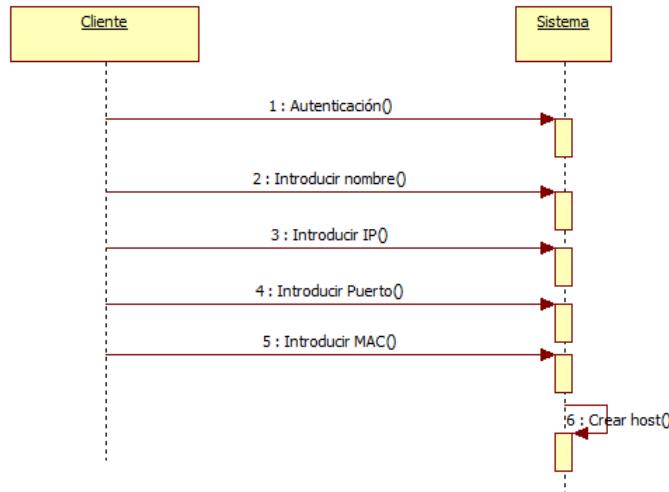


Figura 5.19: Diagrama de secuencia “Crearhost”

Operación: Autenticación(nombre,ctsn)

Responsabilidad: Acceder al sistema

Precondiciones: Debe existir un usuario con nombre (nombre) y contraseña (ctsn).

Postcondiciones: Se accede al sistema con las credenciales nombre y ctsn.

Operación: Introducir\_nombre(nombre)

Responsabilidad: Especificar nombre de host a crear.

Precondiciones: Se determina el nombre del host a crear.

Postcondiciones: -.

Operación: Introducir\_IP(ip)

Responsabilidad: Especificar la IP del host a crear.

Precondiciones: Se determina la IP del host a crear.

Postcondiciones: -.

Operación: Introducir\_Puerto(puerto)

Responsabilidad: Especificar el puerto del host a crear.

Precondiciones: Se determina el puerto del host a crear.

Postcondiciones: -.

Operación: Introducir\_MAC(mac)

Responsabilidad: Especificar la mac del host a crear.

Precondiciones: Se determina la mac del host a crear.

Postcondiciones: -.

Operación: Crear\_host(nombre,ip,puerto,mac)

Responsabilidad: Crear un host con los datos introducidos.

Precondiciones: Debe de haber un cliente autenticado en el sistema.

Postcondiciones: Se crea una instancia host con los datos(nombre,ip,puerto,mac).

### 5.3.11. Caso de uso Autenticación

Diagrama de secuencia

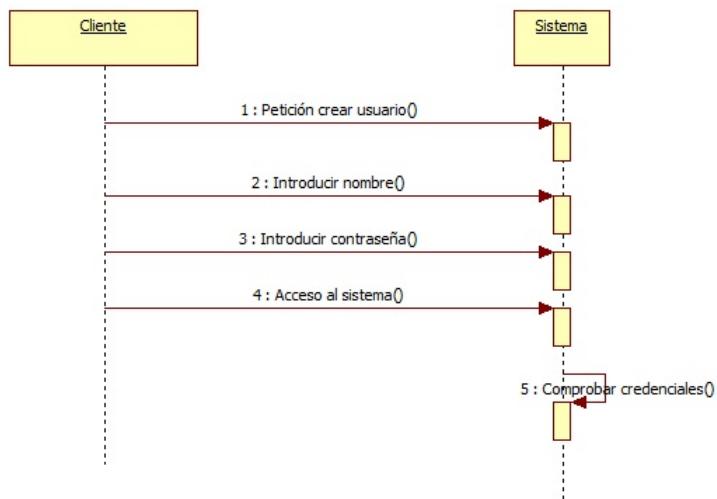


Figura 5.20: Diagrama de secuencia “Autenticación”

Operación: Peticion\_crear\_usuario(nombre,ctsn)

Responsabilidad: Crear usuario.

Precondiciones: -.

Postcondiciones: Se crea una instancia usuario con nombre = nombre y contraseña = ctsn.

Operación: Introducir\_nombre(nombre)

Responsabilidad: Especificar nombre de usuario a crear.

Precondiciones: Se determina el nombre del usuario a crear.

Postcondiciones: -.

Operación: Introducir\_contraseña(ctsn)

Responsabilidad: Especificar la contraseña del usuario a crear.

Precondiciones: Se determina la contraseña del usuario a crear.

Postcondiciones: -.

Operación: Acceso\_al\_sistema(nombre,ctsn)

Responsabilidad: El cliente pide el acceso al sistema con las credenciales nombre y ctsn.

Precondiciones: -.

Postcondiciones: -.

Operación: Comprobar\_credenciales(nombre,ctsn)

Responsabilidad: Comprobar si existe un usuario con las credenciales.

Precondiciones: -.

Postcondiciones: Se comprueba que existe una instancia de usuario con nombre=nombre y contraseña=ctsn

### 5.3.12. Caso de uso Orden Libre

Diagrama de secuencia

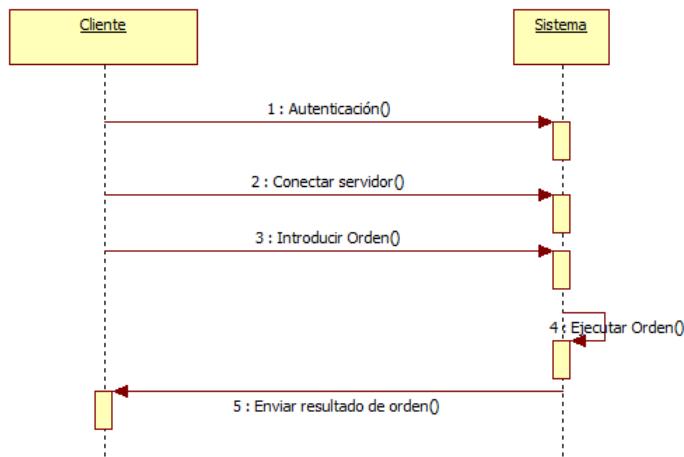


Figura 5.21: Diagrama de secuencia “Autenticación”

Operación: Autenticación(nombre,ctsn)

Responsabilidad: Acceder al sistema

Precondiciones: Debe existir un usuario con nombre (nombre) y contraseña (ctsn).

Postcondiciones: Se accede al sistema con las credenciales nombre y ctsn.

Operación: Conectar\_host(ip,puerto)

Responsabilidad: Establecer la conexión con el servidor.

Precondiciones: Debe de haber un cliente autenticado en el sistema.

Postcondiciones: Se inicia la conexión, creándose una instancia de conectar(ip,puerto).

Operación: Introducir\_Orden(w\_orden)

Responsabilidad: El cliente introduce una orden a ejecutar.

Precondiciones: Debe de estar conectado a un host.

Postcondiciones: Se manda una orden al host servidor para ejecutar la orden w\_orden.

Operación: Enviar\_resultado\_de\_orden(w\_resultado)

Responsabilidad: El sistema devuelve el resultado de la orden anteriormente mandada (w\_orden).

Precondiciones: Debe de estar conectado al host servidor y haber mandado una orden (w\_orden).

Postcondiciones: Se envía en una salida\_orden , el resultado (w\_resultado) de la orden w\_orden.

### 5.3.13. Caso de uso Información del sistema

Diagrama de secuencia

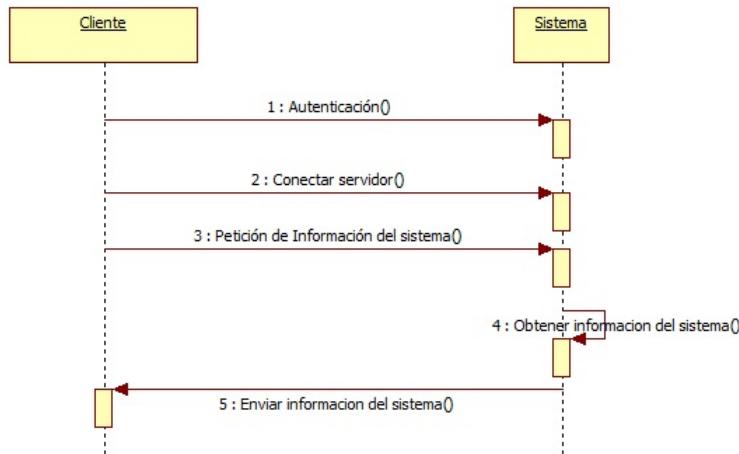


Figura 5.22: Diagrama de secuencia “Autenticación”

Operación: Autenticación(nombre,ctsn)

Responsabilidad: Acceder al sistema

Precondiciones: Debe existir un usuario con nombre (nombre) y contraseña (ctsn).

Postcondiciones: Se accede al sistema con las credenciales nombre y ctsn.

Operación: Conectar\_host(ip,puerto)

Responsabilidad: Establecer la conexión con el servidor.

Precondiciones: Debe de haber un cliente autenticado en el sistema.

Postcondiciones: Se inicia la conexión, creándose una instancia de conectar(ip,puerto).

Operación: Petición\_Information\_del\_sistema(w\_opcion)

Responsabilidad: El cliente pide la información del sistema correspondiente a la opción w\_opcion.

Precondiciones: Debe de estar conectado a un host.

Postcondiciones: Se manda una orden al host servidor para obtener la información cuya opción es w\_opcion.

Operación: Obtener\_Information\_del\_sistema(w\_opcion)

Responsabilidad: El sistema obtiene la información del sistema correspondiente a la opción w\_opcion.

Precondiciones: Debe de estar conectado al host servidor y haber mandado una orden de petición de información del sistema con su opción correspondiente (w\_opcion).

Postcondiciones: Se obtiene la información del sistema correspondiente a la información del sistema con opción (w\_opcion).

Operación: Enviar\_información\_del\_sistema(w\_resultado)

Responsabilidad: El sistema devuelve el resultado de la información anteriormente obtenida (w\_opcion).

Precondiciones: Debe de estar conectado al host servidor y haber mandado petición de información del sistema(w\_opción).

Postcondiciones: Se envía en una salida\_orden , el resultado (w\_resultado) de la información del sistema que se ha pedido con la opción w\_opcion.

### 5.3.14. Caso de uso Listar procesos

Diagrama de secuencia

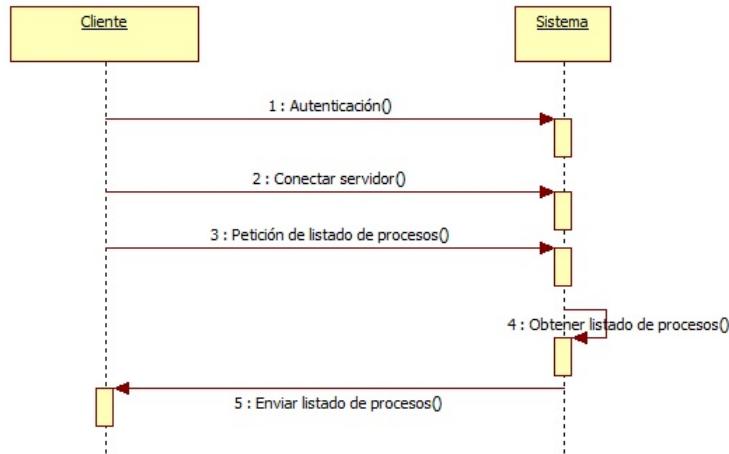


Figura 5.23: Diagrama de secuencia “Listar procesos”

Operación: Autenticación(nombre,ctsn)

Responsabilidad: Acceder al sistema

Precondiciones: Debe existir un usuario con nombre (nombre) y contraseña (ctsn).

Postcondiciones: Se accede al sistema con las credenciales nombre y ctsn.

Operación: Conectar\_host(ip,puerto)

Responsabilidad: Establecer la conexión con el servidor.

Precondiciones: Debe de haber un cliente autenticado en el sistema.

Postcondiciones: Se inicia la conexión, creándose una instancia de conectar(ip,puerto).

Operación: Petición\_Listado\_de\_procesos()

Responsabilidad: El manda una orden para listar los procesos del sistema.

Precondiciones: Debe de estar conectado a un host.

Postcondiciones: Se manda una orden al host servidor para listar los procesos en ejecución.

Operación: Obtener\_Listado\_de\_procesos()

Responsabilidad: El sistema obtiene el listado de procesos activos del sistema.

Precondiciones: Debe de estar conectado al host servidor y haber mandado una orden de petición de listado de procesos.

Postcondiciones: Se obtiene el listado de los procesos en ejecución del sistema.

Operación: Enviar\_Listado\_de\_procesos(w\_listado)

Responsabilidad: El sistema devuelve el listado de los procesos en ejecución anteriormente obtenido.

Precondiciones: Debe de estar conectado al host servidor, haber mandado petición de listado de procesos y haber obtenido dicha información.

Postcondiciones: Se envía en una salida\_orden , el resultado (w\_resultado) del listado de procesos anteriormente obtenido.

### 5.3.15. Caso de uso Eliminar proceso

Diagrama de secuencia

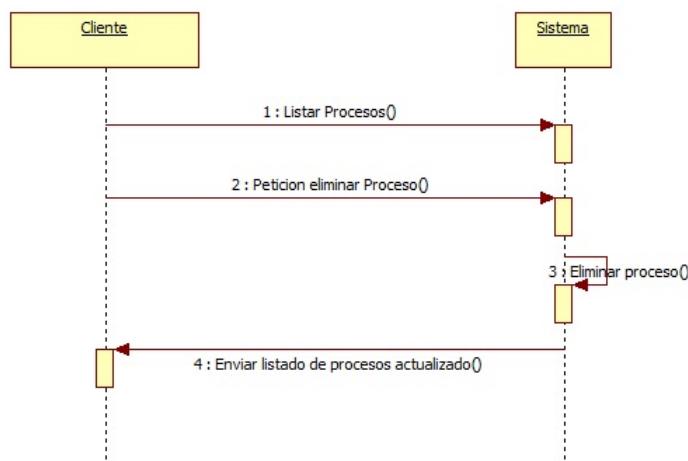


Figura 5.24: Diagrama de secuencia “Eliminar proceso”

Operación: Listar\_Procesos()

Responsabilidad: El cliente obtiene una lista de procesos en ejecución.

Precondiciones: Cliente conectado.

Postcondiciones: Cliente obtiene listado de procesos.

Operación: Petición\_eliminar\_proceso(w\_proceso)

Responsabilidad: El cliente manda la orden para eliminar el proceso con pid w\_proceso.

Precondiciones: Debe de haber un cliente autenticado en el sistema, conectado al host y haberse listado los procesos en ejecución del host servidor.

Postcondiciones: Se manda la orden para eliminar el proceso con pid w\_proceso.

Operación: Eliminar\_proceso(w\_proceso)

Responsabilidad: El sistema elimina el proceso con pid w\_proceso.

Precondiciones: Debe de haberse mandado una orden para la eliminación del proceso con pid w\_proceso.

Postcondiciones: El sistema ejecuta la orden necesaria para eliminar el proceso con pid w\_proceso.

Operación: Enviar\_listado\_de\_procesos\_actualizado()

Responsabilidad: El sistema devuelve el resultado de listar\_procesos tras haber eliminado uno de ellos.

Precondiciones: Debe de estar conectado al host servidor y haberse eliminado un proceso.

Postcondiciones: Se obtiene el listado de los procesos en ejecución del sistema y se manda en una salida\_orden al cliente.

### 5.3.16. Caso de uso Crear fichero

Diagrama de secuencia

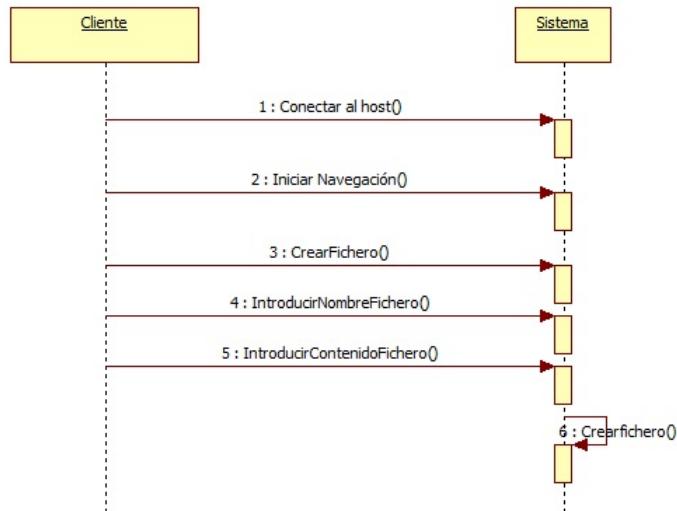


Figura 5.25: Diagrama de secuencia “Crear fichero”

Operación: Conectar\_host(ip,puerto)

Responsabilidad: Establecer la conexión con el servidor.

Precondiciones: Debe de haber un cliente autenticado en el sistema.

Postcondiciones: Se inicia la conexión, creándose una instancia de conectar(ip,puerto).

Operación: Iniciar\_navegación()

Responsabilidad: El cliente manda una orden para iniciar la navegación por el árbol de directorios.

Precondiciones: Debe de estar conectado a un host.

Postcondiciones: Se manda una orden al host servidor para iniciar la navegación por el árbol de directorios.

Operación: Crear\_fichero()

Responsabilidad: El cliente manda una orden para crear un fichero.

Precondiciones: Debe de estar conectado al host servidor y haber mandado una orden para iniciar la navegación.

Postcondiciones: Se envía una orden para crear un fichero.

Operación: IntroducirNombreFichero(w\_nombre)

Responsabilidad: El cliente manda el nombre del fichero a crear.

Precondiciones: Debe de estar conectado al host servidor, haber mandado petición de crear fichero.

Postcondiciones: Se envía el nombre del fichero al host servidor (w\_nombre).

Operación: IntroducirContenidoFichero(w\_contenido)

Responsabilidad: El cliente manda el contenido del fichero a crear.

Precondiciones: Debe de estar conectado al host servidor, haber mandado petición de crear fichero.

Postcondiciones: Se envía el contenido del fichero al host servidor (w\_contenido).

Operación: CrearFichero(w\_nombre,w\_contenido)

Responsabilidad: El sistema ejecuta la orden para crear el fichero con nombre w\_nombre y contenido w\_contenido.

Precondiciones: Debe de haberse enviado los parámetros w\_nombre y w\_contenido previamente.

Postcondiciones: Se ejecuta una orden para crear al fichero con nombre w\_nombre y contenido w\_contenido.

### 5.3.17. Caso de uso Cortar/Copiar fichero

Diagrama de secuencia

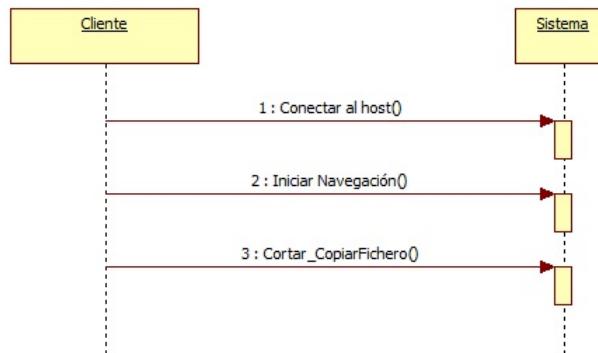


Figura 5.26: Diagrama de secuencia “Cortar/Copiar fichero”

Operación: Conectar\_host(ip,puerto)

Responsabilidad: Establecer la conexión con el servidor.

Precondiciones: Debe de haber un cliente autenticado en el sistema.

Postcondiciones: Se inicia la conexión, creándose una instancia de conectar(ip,puerto).

Operación: Iniciar\_navegación()

Responsabilidad: El cliente manda una orden para iniciar la navegación por el árbol de directorios.

Precondiciones: Debe de estar conectado a un host.

Postcondiciones: Se manda una orden al host servidor para iniciar la navegación por el árbol de directorios.

Operación: Cortar\_CopiarFichero(w\_nombre,w\_ruta,w\_orden)

Responsabilidad: El cliente manda la orden para copiar o pegar según w\_orden, un fichero con nombre w\_nombre en la ruta w\_ruta.

Precondiciones: Debe de estar conectado a un host e iniciado una navegación por el árbol de directorios.

Postcondiciones: Se manda una orden para cortar o copiar(según w\_orden) el fichero con nombre w\_nombre en la ruta w\_ruta.

### 5.3.18. Caso de uso Pegar fichero

Diagrama de secuencia

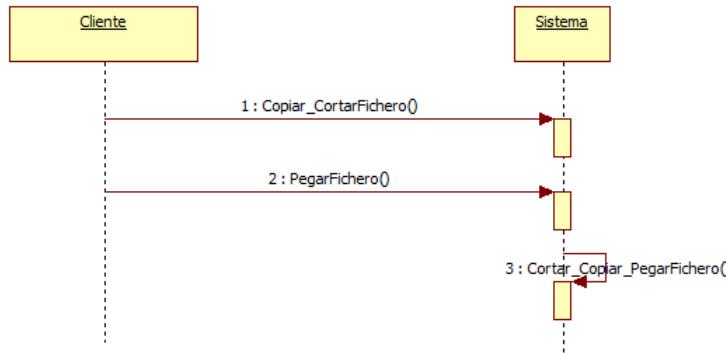


Figura 5.27: Diagrama de secuencia “Pegar fichero”

Operación: Cortar\_CopiarFichero(w\_nombre,w\_ruta,w\_orden)

Responsabilidad: El cliente manda la orden para copiar o pegar según w\_orden, un fichero con nombre w\_nombre en la ruta w\_ruta.

Precondiciones: Debe de estar conectado a un host e iniciado una navegación por el árbol de directorios.

Postcondiciones: Se manda una orden para cortar o copiar(según w\_orden) el fichero con nombre w\_nombre en la ruta w\_ruta.

Operación: PegarFichero(w\_nombre,w\_ruta,w\_orden);

Responsabilidad: El cliente manda una orden para pegar un fichero

Precondiciones: Debe de estar conectado a un host, haber iniciado la navegación y haber copiado o cortado un fichero.

Postcondiciones: Se manda una orden al host servidor para pegar el fichero w\_nombre en la ruta w\_ruta.

Operación: Cortar\_Copiar\_pagarFichero(w\_nombre,w\_ruta,w\_orden)

Responsabilidad: El sistema ejecuta la orden que según w\_orden será una archivo copiado o cortado, en la ruta w\_ruta el fichero w\_nombre.

Precondiciones: Debe de estar conectado a un host , iniciado una navegación por el árbol de directorios y haber copiado o cortado un fichero.

Postcondiciones: Se manda una orden para pegar el fichero copiado o cortado según w\_orden, con nombre w\_nombre en la ruta w\_ruta.

### 5.3.19. Caso de uso Renombrar fichero

Diagrama de secuencia

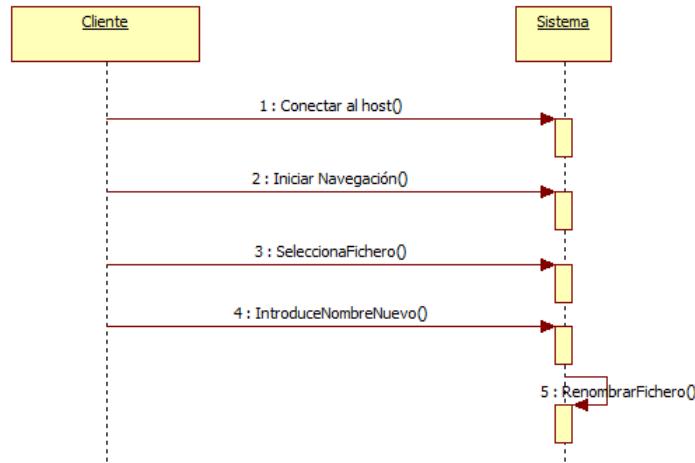


Figura 5.28: Diagrama de secuencia “Renombrar fichero”

Operación: Conectar\_host(ip,puerto)

Responsabilidad: Establecer la conexión con el servidor.

Precondiciones: Debe de haber un cliente autenticado en el sistema.

Postcondiciones: Se inicia la conexión, creándose una instancia de conectar(ip,puerto).

Operación: Iniciar\_navegación()

Responsabilidad: El cliente manda una orden para iniciar la navegación por el árbol de directorios.

Precondiciones: Debe de estar conectado a un host.

Postcondiciones: Se manda una orden al host servidor para iniciar la navegación por el árbol de directorios.

Operación: Seleccionar\_fichero(w\_nombre)

Responsabilidad: El cliente manda una orden para indicar que el fichero a renombrar es el fichero cuyo nombre es w\_nombre.

Precondiciones: Debe de estar conectado a un host y haber iniciado navegación.

Postcondiciones: Se manda una orden con el nombre del fichero seleccionado.

Operación: IntroducirNombreNuevo(w\_nombre\_nuevo)

Responsabilidad: El cliente manda el nombre nuevo al sistema.

Precondiciones: Debe de estar conectado a un host, iniciado una navegación por el árbol de directorios y haber y haber seleccionado un fichero.

Postcondiciones: Se manda una orden con el nuevo valor del nombre del fichero previamente seleccionado.

Operación: RenombrarFichero(w\_nombre,w\_nombre\_nuevo);

Responsabilidad: El sistema ejecuta la orden para renombrar el fichero con w\_nombre a w\_nombre\_nuevo.

Precondiciones: Debe de haberse seleccionado un fichero y haberse mandado la orden con el nuevo nombre del fichero.

Postcondiciones: Se ejecuta la orden en el servidor para renombrar el fichero con nombre w\_nombre a w\_nombre\_nuevo.

### 5.3.20. Caso de uso Eliminar fichero

Diagrama de secuencia

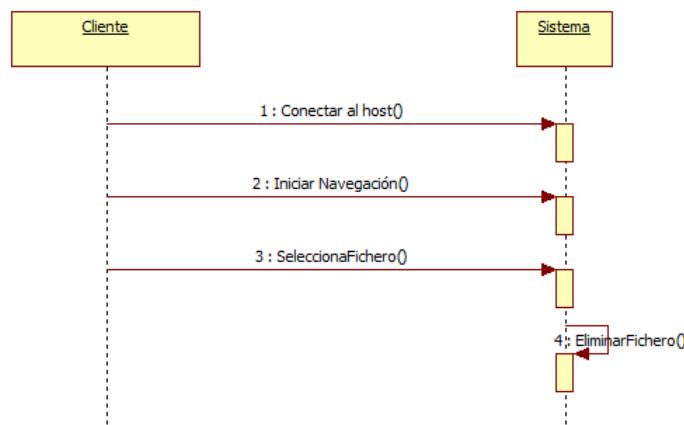


Figura 5.29: Diagrama de secuencia “Eliminar fichero”

Operación: Conectar\_host(ip,puerto)

Responsabilidad: Establecer la conexión con el servidor.

Precondiciones: Debe de haber un cliente autenticado en el sistema.

Postcondiciones: Se inicia la conexión, creándose una instancia de conectar(ip,puerto).

Operación: Iniciar\_navegación()

Responsabilidad: El cliente manda una orden para iniciar la navegación por el árbol de directorios.

Precondiciones: Debe de estar conectado a un host.

Postcondiciones: Se manda una orden al host servidor para iniciar la navegación por el árbol de directorios.

Operación: Seleccionar\_fichero(w\_nombre)

Responsabilidad: El cliente manda una orden para indicar que el fichero a renombrar es el fichero cuyo nombre es w\_nombre.

Precondiciones: Debe de estar conectado a un host y haber iniciado navegación.

Postcondiciones: Se manda una orden con el nombre del fichero seleccionado.

Operación: Eliminar\_fichero(w\_nombre)

Responsabilidad: El sistema elimina el fichero con nombre w\_nombre.

Precondiciones: Debe de estar conectado a un host, iniciado una navegación por el árbol de directorios y haber seleccionado un fichero.

Postcondiciones: El sistema ejecuta la orden para eliminar el fichero con nombre w\_nombre.

### 5.3.21. Caso de uso Mostrar directorio atrás

Diagrama de secuencia

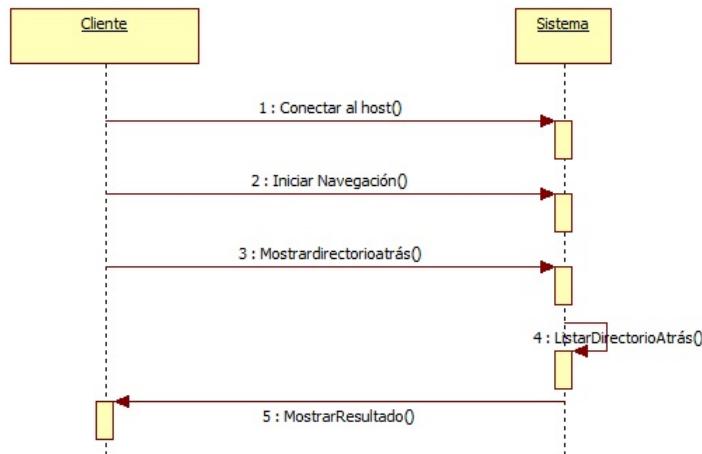


Figura 5.30: Diagrama de secuencia “Mostrar directorio atrás”

Operación: Conectar\_host(ip,puerto)

Responsabilidad: Establecer la conexión con el servidor.

Precondiciones: Debe de haber un cliente autenticado en el sistema.

Postcondiciones: Se inicia la conexión, creándose una instancia de conectar(ip,puerto).

Operación: Iniciar\_navegación()

Responsabilidad: El cliente manda una orden para iniciar la navegación por el árbol de directorios.

Precondiciones: Debe de estar conectado a un host.

Postcondiciones: Se manda una orden al host servidor para iniciar la navegación por el árbol de directorios.

Operación: Mostrar\_directorio\_atrás(w\_ruta)

Responsabilidad: El cliente manda una orden para mostrar el directorio atrás en el navegador.

Precondiciones: Debe de estar conectado a un host y haber iniciado navegación.

Postcondiciones: Se manda una orden para mostrar el directorio atrás del directorio con ruta w\_ruta.

Operación: ListarDirectorioAtrás(w\_ruta)

Responsabilidad: El sistema lista los ficheros y directorios de la ruta atrás de w\_ruta.

Precondiciones: Debe de estar conectado a un host, iniciado una navegación por el árbol de directorios y haberse mandado la orden de “Mostrar\_directorio\_atrás”.

Postcondiciones: Se lista los ficheros y directorios pertenecientes al directorio atrás de w\_ruta.

### 5.3.22. Caso de uso Mostrar directorio adelante

Diagrama de secuencia

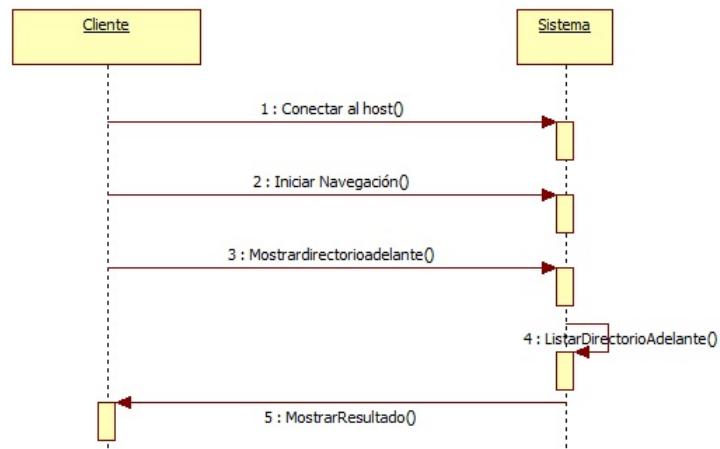


Figura 5.31: Diagrama de secuencia “Mostrar directorio adelante”

Operación: Conectar\_host(ip,puerto)

Responsabilidad: Establecer la conexión con el servidor.

Precondiciones: Debe de haber un cliente autenticado en el sistema.

Postcondiciones: Se inicia la conexión, creándose una instancia de conectar(ip,puerto).

Operación: Iniciar\_navegación()

Responsabilidad: El cliente manda una orden para iniciar la navegación por el árbol de directorios.

Precondiciones: Debe de estar conectado a un host.

Postcondiciones: Se manda una orden al host servidor para iniciar la navegación por el árbol de directorios.

Operación: Mostrar\_directorio\_adelante(w\_ruta)

Responsabilidad: El cliente manda una orden para mostrar el directorio adelante en el navegador.

Precondiciones: Debe de estar conectado a un host y haber iniciado navegación.

Postcondiciones: Se manda una orden para mostrar el directorio adelante del directorio con ruta w\_ruta.

Operación: ListarDirectorioAtrás(w\_ruta)

Responsabilidad: El sistema lista los ficheros y directorios de la ruta adelante de w\_ruta.

Precondiciones: Debe de estar conectado a un host, iniciado una navegación por el árbol de directorios y haberse mandado la orden de “Mostrar\_directorio\_adelante”.

Postcondiciones: Se lista los ficheros y directorios pertenecientes al directorio adelante de w\_ruta.

### 5.3.23. Caso de uso Transferir fichero

Diagrama de secuencia

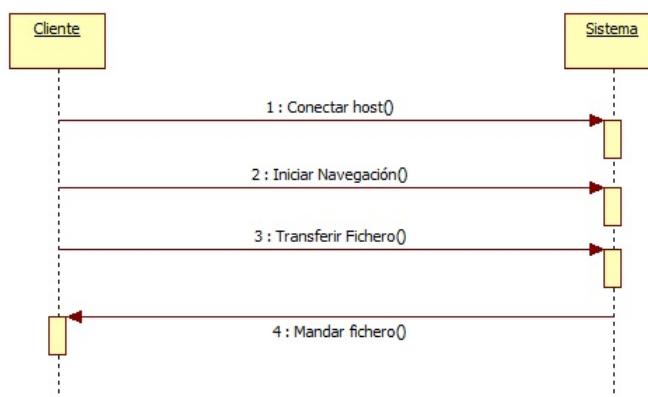


Figura 5.32: Diagrama de secuencia “Transferir fichero”

Operación: Conectar\_host(ip,puerto)

Responsabilidad: Establecer la conexión con el servidor.

Precondiciones: Debe de haber un cliente autenticado en el sistema.

Postcondiciones: Se inicia la conexión, creándose una instancia de conectar(ip,puerto).

Operación: Iniciar\_navegación()

Responsabilidad: El cliente manda una orden para iniciar la navegación por el árbol de directorios.

Precondiciones: Debe de estar conectado a un host.

Postcondiciones: Se manda una orden al host servidor para iniciar la navegación por el árbol de directorios.

Operación: TransferirFichero(w\_nombre)

Responsabilidad: El cliente manda una orden para transferir el fichero con nombre w\_nombre.

Precondiciones: Debe de estar conectado a un host y haber iniciado navegación.

Postcondiciones: Se manda una orden para transferir el fichero con nombre w\_nombre al sistema.

Operación: MandarFichero(w\_nombre)

Responsabilidad: El sistema transfiere al cliente el fichero con nombre w\_nombre.

Precondiciones: Debe de estar conectado a un host, iniciado una navegación por el árbol de directorios y haberse mandado la orden de “Transferir Fichero”.

Postcondiciones: Se transfiere el fichero con nombre w\_nombre al cliente.

### 5.3.24. Caso de uso Listar Scripts

Diagrama de secuencia

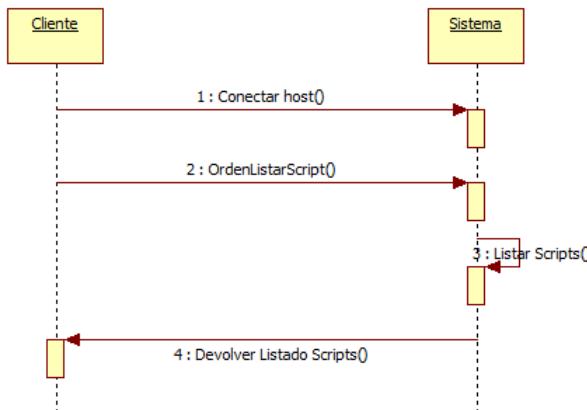


Figura 5.33: Diagrama de secuencia “Listar Scripts”

Operación: Conectar\_host(ip,puerto)

Responsabilidad: Establecer la conexión con el servidor.

Precondiciones: Debe de haber un cliente autenticado en el sistema.

Postcondiciones: Se inicia la conexión, creándose una instancia de conectar(ip,puerto).

Operación: OrdenListarScripts()

Responsabilidad: El cliente manda una orden para listar los scripts.

Precondiciones: Debe de estar conectado a un host.

Postcondiciones: Se manda una orden al host servidor para listar los scripts contenidos en el directorio Scripts del servidor.

Operación: ListarScripts()

Responsabilidad: El sistema obtiene el listado de los scripts contenidos en el directorio.

Precondiciones: Debe de estar conectado a un host y haberse mandado por parte del cliente la OrdenListarScripts.

Postcondiciones: Se ejecuta la orden para obtener el listado de scripts del directorio Scripts del servidor.

Operación: DevolverListadoScripts(w\_listado)

Responsabilidad: El sistema manda en la salida de una orden el resultado del “ListarScripts” w\_listado.

Precondiciones: Debe de estar conectado a un host y haberse ejecutado la operación ListarScripts, obteniendo su resultado.

Postcondiciones: Se manda en la salida de una orden, el resultado de “ListarScripts” (salida\_orden = w\_listado).

### 5.3.25. Caso de uso Ver Script

Diagrama de secuencia

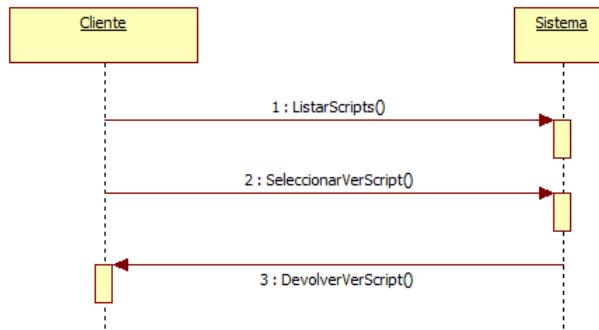


Figura 5.34: Diagrama de secuencia “Ver Script”

Operación: ListarScripts()

Responsabilidad: Realiza el anterior caso de uso “ListarScripts”.

Precondiciones: Debe de haber un cliente autenticado en el sistema.

Postcondiciones: Se manda el resultado del listado de scripts.

Operación: SeleccionarVerScript(w\_nombre)

Responsabilidad: El cliente manda selecciona un script en el listado para ver.

Precondiciones: Debe de estar conectado a un host y haber listado los scripts contenidos.

Postcondiciones: Se manda orden para ver el script con nombre w\_nombre.

Operación: DevolverScript(w\_contenido)

Responsabilidad: El sistema obtiene el contenido del script seleccionado y lo manda al cliente.

Precondiciones: Debe de estar conectado a un host y haberse mandado por parte del cliente la orden para ver un script.

Postcondiciones: Se devuelve en una orden el contenido del script seleccionado w\_contenido.

### 5.3.26. Caso de uso Ejecutar Script

Diagrama de secuencia

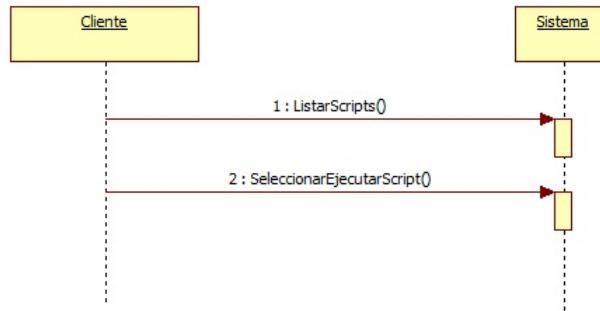


Figura 5.35: Diagrama de secuencia “EjecutarScript”

Operación: ListarScripts()

Responsabilidad: Realiza el anterior caso de uso “ListarScripts”.

Precondiciones: Debe de haber un cliente autenticado en el sistema.

Postcondiciones: Se manda el resultado del listado de scripts.

Operación: SeleccionarEjecutarScript(w\_nombre)

Responsabilidad: El cliente selecciona un script para ejecutar y el sistema lo ejecuta.

Precondiciones: Debe de estar conectado a un host y haber listado los scripts contenidos.

Postcondiciones: Se manda orden para ejecutar el script con nombre w\_nombre y el sistema realiza la acción pedida.

### 5.3.27. Caso de uso Transferir Script

Diagrama de secuencia

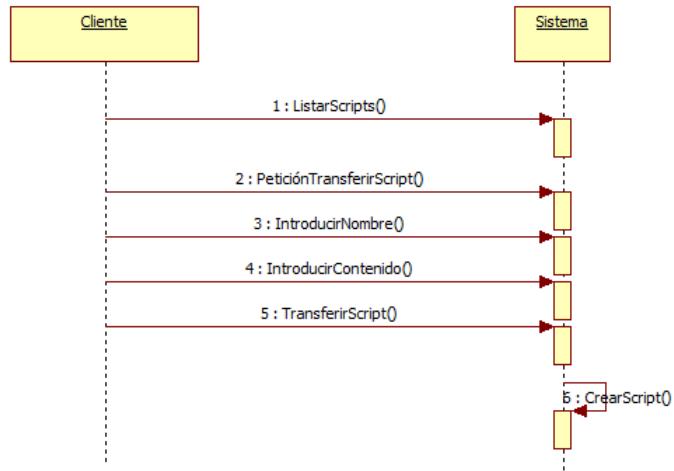


Figura 5.36: Diagrama de secuencia “EjecutarScript”

Operación: ListarScripts()

Responsabilidad: Realiza el anterior caso de uso “ListarScripts”.

Precondiciones: Debe de haber un cliente autenticado en el sistema.

Postcondiciones: Se manda el resultado del listado de scripts.

Operación: PeticiónTransferirScript()

Responsabilidad: El cliente pide transferir un script al sistema.

Precondiciones: Debe de estar conectado a un host y haber listado los scripts contenidos.

Postcondiciones: Se manda orden para transferir un script al sistema remoto.

Operación: IntroducirNombre(w\_nombre)

Responsabilidad: Introduce el nombre del script a transferir.

Precondiciones: Debe de haber un cliente autenticado en el sistema, listado los scripts existentes y haberse mandado una orden de petición transferir script.

Postcondiciones: Se manda en una orden el nombre w\_nombre del script a transferir.

Operación: IntroducirContenido(w\_contenido)

Responsabilidad: Introducir el contenido del script a transferir.

Precondiciones: Debe de haber un cliente autenticado en el sistema, listado los scripts existentes y haberse mandado una orden de petición transferir script.

Postcondiciones: Se manda en una orden el contenido w\_contenido del script a transferir.

Operación: TransferirScript(w\_nombre,w\_contenido)

Responsabilidad: Transferir el script al sistema remoto.

Precondiciones: Debe de haber un cliente autenticado en el sistema, listado los scripts existentes y haberse mandado una orden de petición transferir script, junto con su nombre w\_nombre, y contenido w\_contenido.

Postcondiciones: Se transfiere el script con nombre w\_nombre y contenido w\_contenido.

Operación: CrearScript(w\_nombre,w\_contenido)

Responsabilidad: El sistema crea el script con nombre w\_nombre y contenido w\_contenido.

Precondiciones: Debe haberse mandado una orden de transferencia de un script.

Postcondiciones: Se ejecuta la orden para crear un script con nombre w\_nombre y contenido w\_contenido.

### 5.3.28. Caso de uso Eliminar Script

Diagrama de secuencia

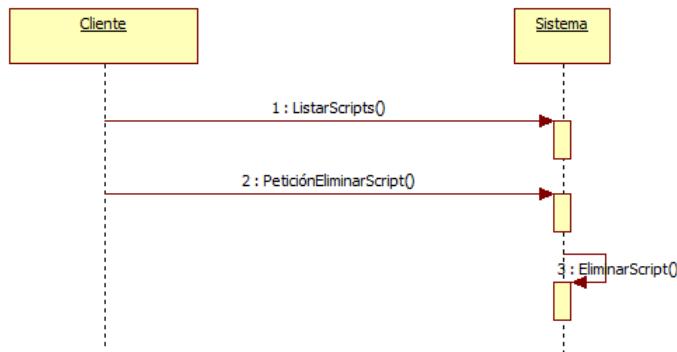


Figura 5.37: Diagrama de secuencia “Eliminar Script”

Operación: ListarScripts()

Responsabilidad: Realiza el anterior caso de uso “ListarScripts”.

Precondiciones: Debe de haber un cliente autenticado en el sistema.

Postcondiciones: Se manda el resultado del listado de scripts.

Operación: PeticiónEliminarScript(w\_nombre)

Responsabilidad: El cliente pide eliminar un scripts al sistema.

Precondiciones: Debe de estar conectado a un host y haber listado los scripts contenidos.

Postcondiciones: Se manda orden para eliminar un script con nombre w\_nombre.

Operación: EliminarScript(w\_nombre)

Responsabilidad: Eliminar el script con nombre w\_nombre.

Precondiciones: Debe de haber un cliente autenticado en el sistema, listado los scripts existentes y haberse mandado una orden de petición eliminar script.

Postcondiciones: Se ejecuta una orden para eliminar el script con nombre w\_nombre.

# Capítulo 6

## Diseño del sistema

La siguiente etapa es realizar el diseño del proyecto. Para ello pasaremos de la especificación del sistema a un modelo más claro para la implementación de el sistema. Habiéndo obtenido en el capítulo anterior la interacción, modelo conceptual, operaciones y respuestas del sistema, obtenemos en éste la arquitectura, clases y operaciones de cada clase, utilizando un enfoque orientado a objetos como es el lenguaje Java.

### 6.1. Arquitectura en tres capas

En este apartado presentamos la arquitectura del sistema software del proyecto, en el cual, lo dividiremos en tres capas las cuales son:

- Capa de presentación: Responsable de la interacción con el usuario.
- Capa del Dominio o de Negocio: Responsable de la implementación de las funcionalidades del sistema.
- Capa de Gestión de datos: Responsable de la interacción con el sistema de gestión de la base de datos o con los ficheros.

La primera de ellas estará manejada por unos eventos de presentación los cuales estarán compuestos por menús seleccionados, botones pulsados, muestras de ventanas... En la siguiente imágen podemos observar dicha arquitectura.

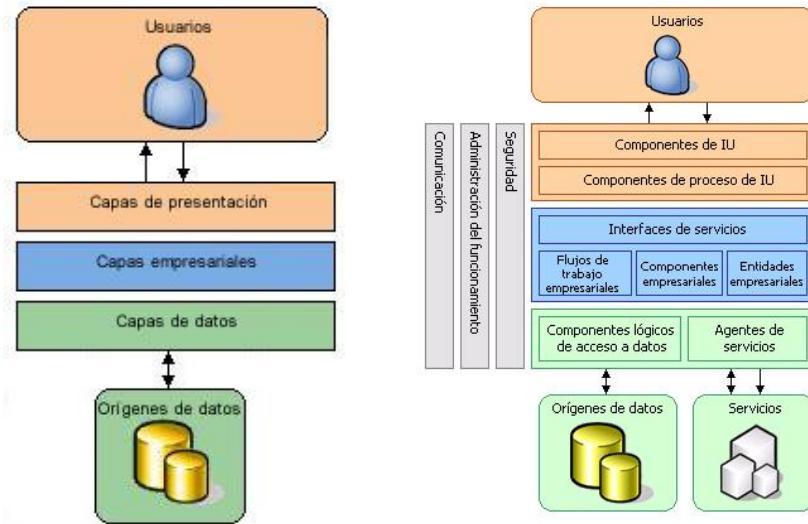


Figura 6.1: Arquitectura en 3 capas

Procederemos ahora a realizar los diferentes diagramas de interacción del sistema para poder entender las operaciones, sus resultados y cómo interactúan éstas con el sistema.

## 6.2. Diagramas de interacción

Los diagramas de secuencia muestran la interacción de un conjunto de objetos a través del tiempo. Esta descripción es importante porque puede dar detalle a los casos de uso, aclarándolos al nivel de mensajes de los objetos existentes. Es decir, nos proporciona la interacción entre los objetos, que se sucede en el tiempo, para un escenario específico durante la ejecución del sistema (por ejemplo, cuando se utiliza un requisito funcional concreto).

Los nombres de las funciones y su situación podrán cambiar a la hora de la implementación, ya que podría añadirse funcionalidades o añadir requerimientos del sistema que a ésta hora no sea contemplado.

### 6.2.1. Menú Servidor

En este primer diagrama podemos observar el menú que va a tener el servidor. El usuario elegirá una opción y dependiendo de esa opción se crearán las pantallas o acciones correspondientes.

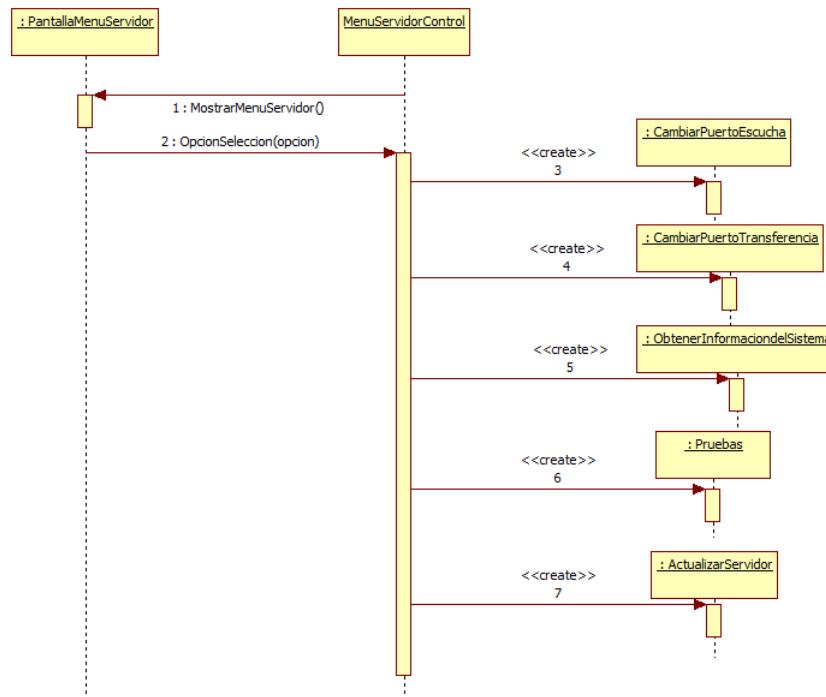


Figura 6.2: Diagrama de interacción del Servidor

CambiarPuertoEscucha
CambiarPuertoTransferencia
ObtenerInformacionSistema
Pruebas
ActualizarServidor

Cuadro 6.1: Menú del Servidor

### 6.2.1.1. Cambiar Puerto de escucha/transferencia

Al ser casos completamente idénticos, se realizará un diagrama de transferencia, quedando como sigue:

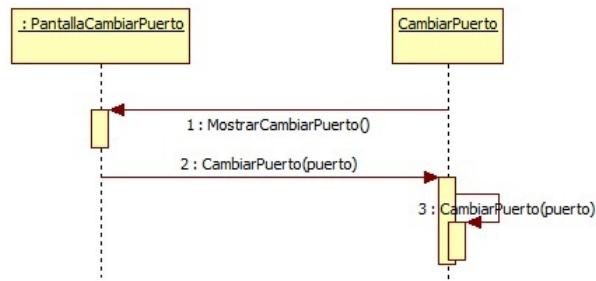


Figura 6.3: Diagrama de interacción “Cambiar puerto”

#### 6.2.1.2. Pruebas

En este diagrama tenemos un discriminante que va a ser la opción de pruebas que queremos obtener. Según éste, tendremos pruebas de discos, de red, del sistema operativo, de memoria, de procesos, de orden libre o de scripts.

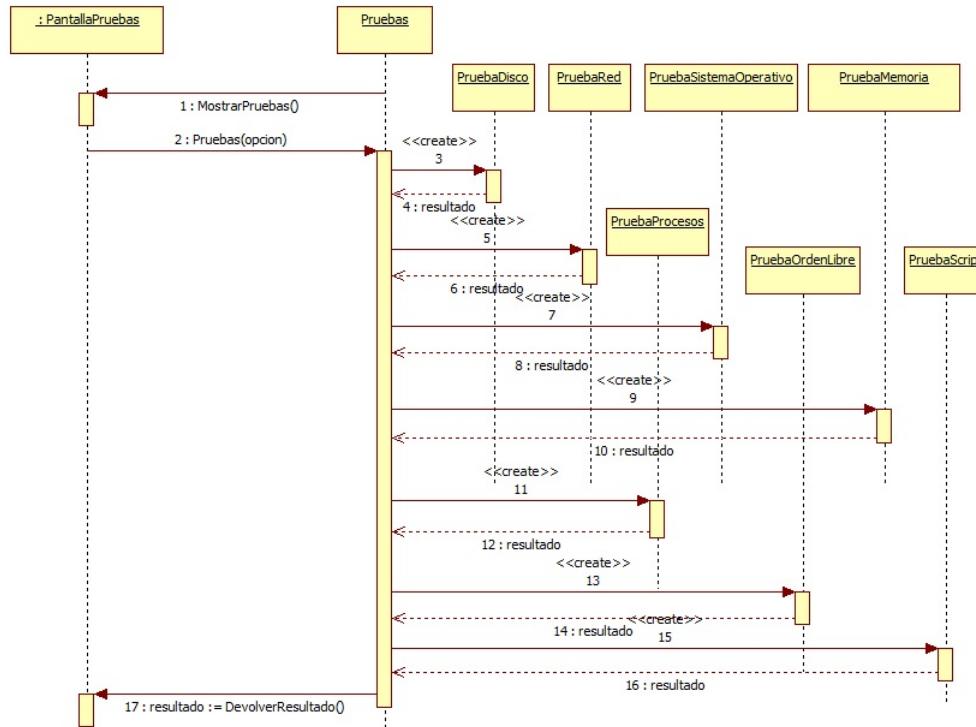


Figura 6.4: Diagrama de interacción “Pruebas”

Informacion de discos
Informacion de red
Informacion del Sistema Operativo
Informacion de Memoria
Procesos
Orden Libre
Scripts
DevolverResultado

Cuadro 6.2: Pruebas

### 6.2.1.3. Actualizar Servidor

En este diagrama mandamos al sistema los parámetros para actualizar el servidor, con los nuevos datos del puerto de escucha y puerto de transferencia.

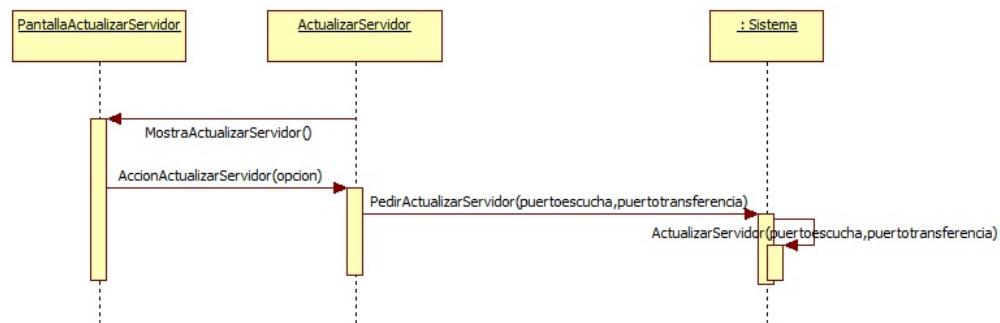


Figura 6.5: Diagrama de interacción “ActualizarServidor”

### 6.2.2. Menú cliente

En este apartado vamos a realizar el diagrama del menú cliente, en el cual podemos observar que se crean los diversos casos en los cuales el cliente puede interaccionar con el sistema. El encendido es el único que cambia, ya que no necesita estar conectado para crearse. Las demás operaciones del cliente, necesitan previamente estar conectados al host remoto.

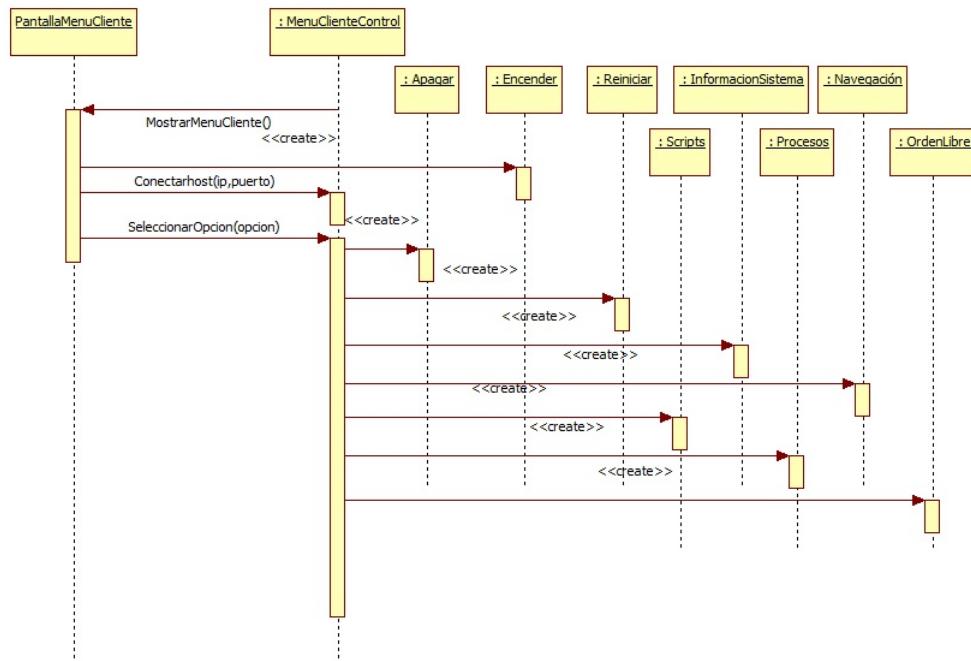


Figura 6.6: Diagrama de interacción “ActualizarServidor”

Información de Sistema
Procesos
Navegación
Scripts
Orden Libre
Apagar
Reiniciar
Encender

Cuadro 6.3: Menú cliente

### 6.2.2.1. Apagar

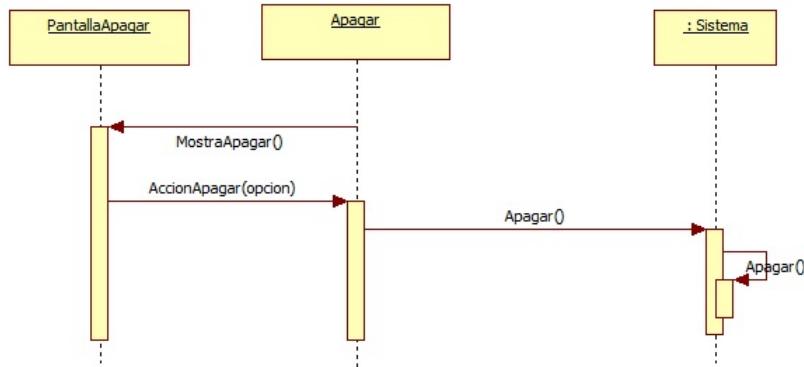


Figura 6.7: Diagrama de interacción “Apagar”

### 6.2.2.2. Reiniciar

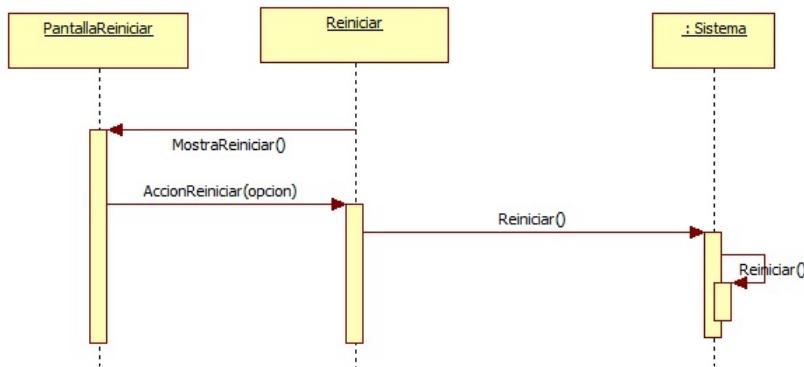


Figura 6.8: Diagrama de interacción “Reiniciar”

### 6.2.2.3. Encender

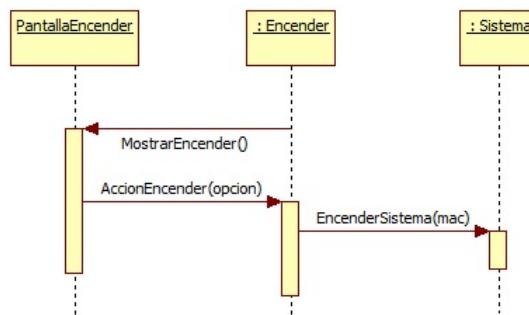


Figura 6.9: Diagrama de interacción “Encender”

### 6.2.2.4. Informacion del sistema

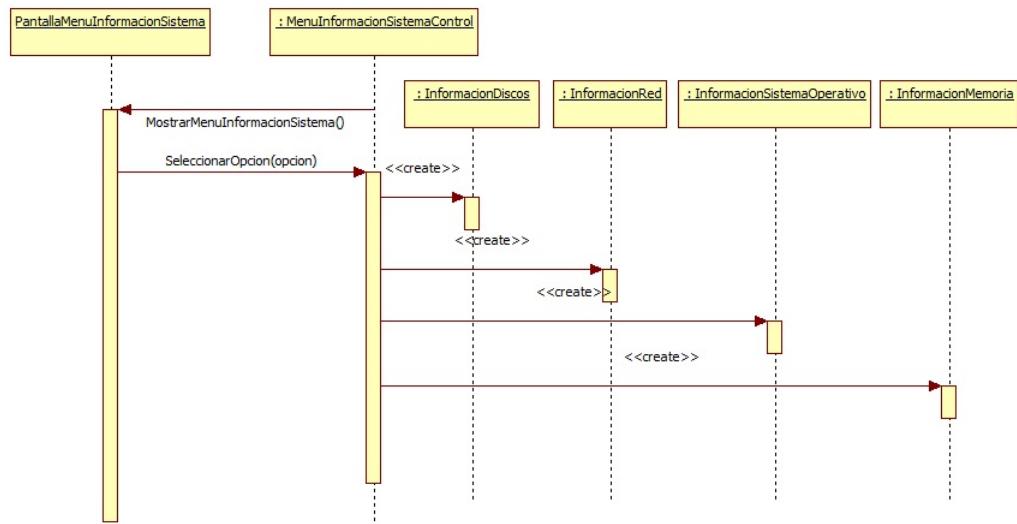


Figura 6.10: Diagrama de interacción “Informacion del sistema”

Informacion de discos
Informacion de Sistema Operativo
Informacion de Red
Informacion de Memoria

Cuadro 6.4: Menú “Informacion del sistema”

#### 6.2.2.5. Listar Procesos

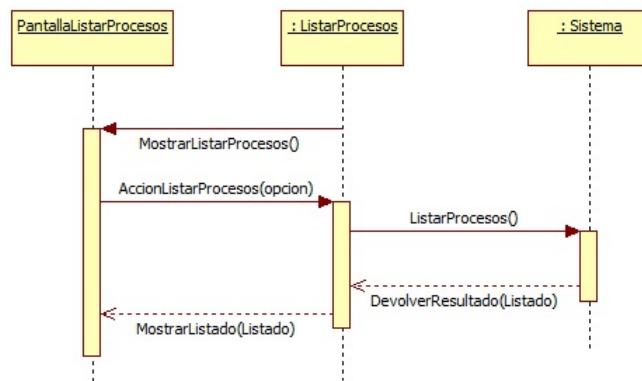


Figura 6.11: Diagrama de interacción “Listar Procesos”

### 6.2.2.6. Eliminar Proceso

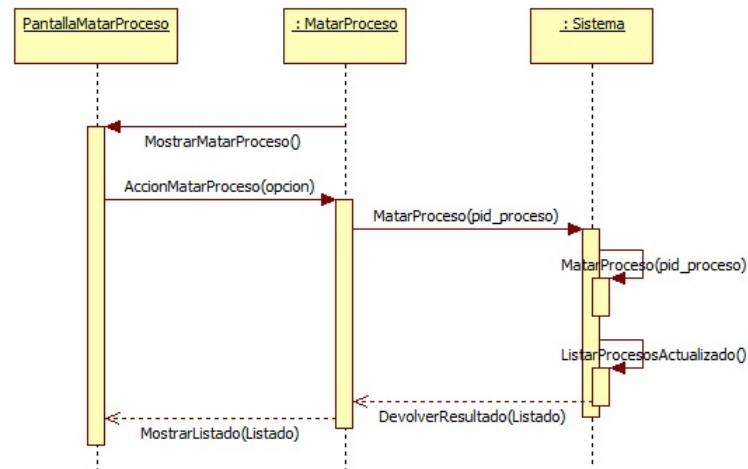


Figura 6.12: Diagrama de interacción “Eliminar Proceso”

### 6.2.2.7. Navegación

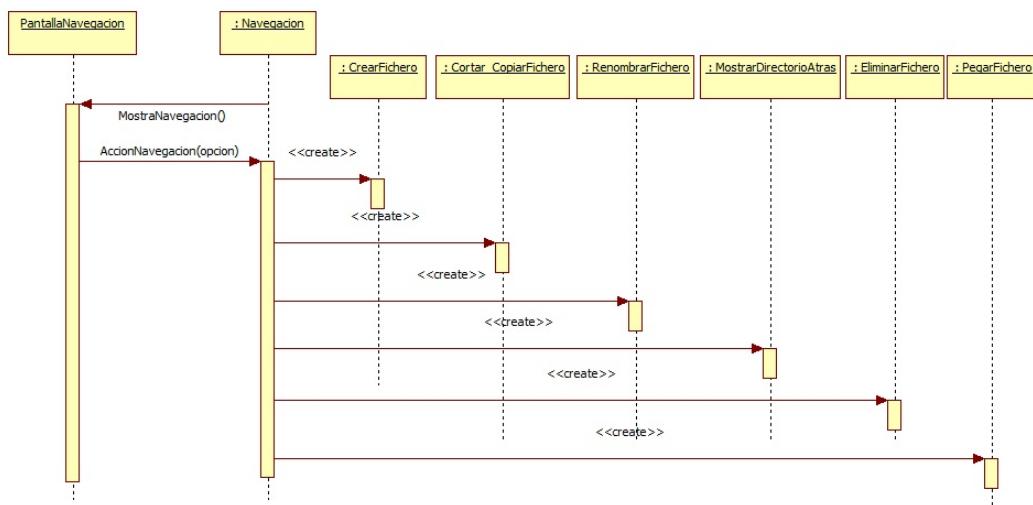


Figura 6.13: Diagrama de interacción “Navegación”

Crear Fichero
Eliminar Fichero
Renombrar Fichero
Copiar_Cortar Fichero
Pegar Fichero
Mostrar Directorio Atrás
Mostrar Directorio Adelante

Cuadro 6.5: Menú “Informacion del sistema”

#### 6.2.2.8. Crear fichero

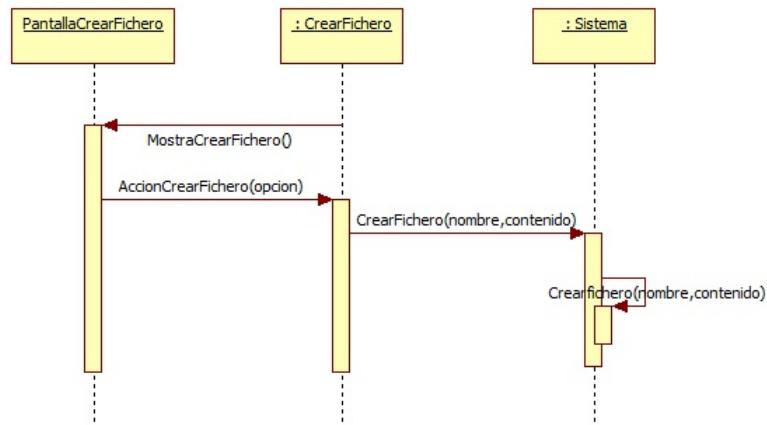


Figura 6.14: Diagrama de interacción “Crear Fichero”

### 6.2.2.9. Cortar\_Copiar fichero

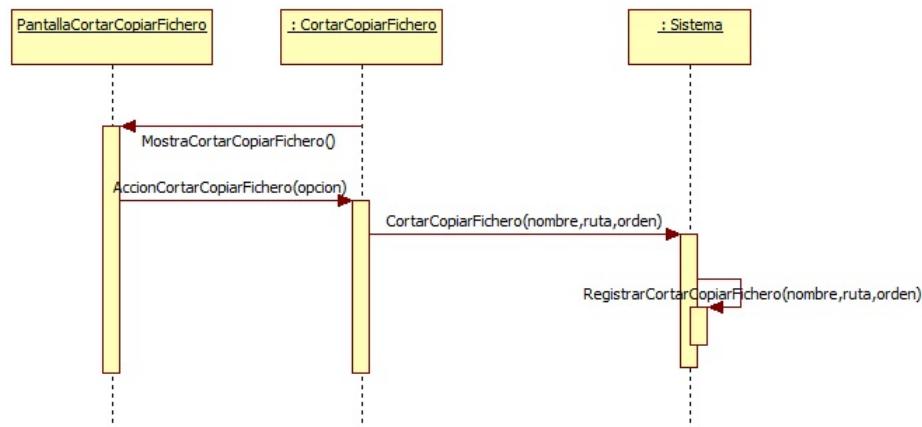


Figura 6.15: Diagrama de interacción “Cortar\_Copiar Fichero”

### 6.2.2.10. Pegar fichero

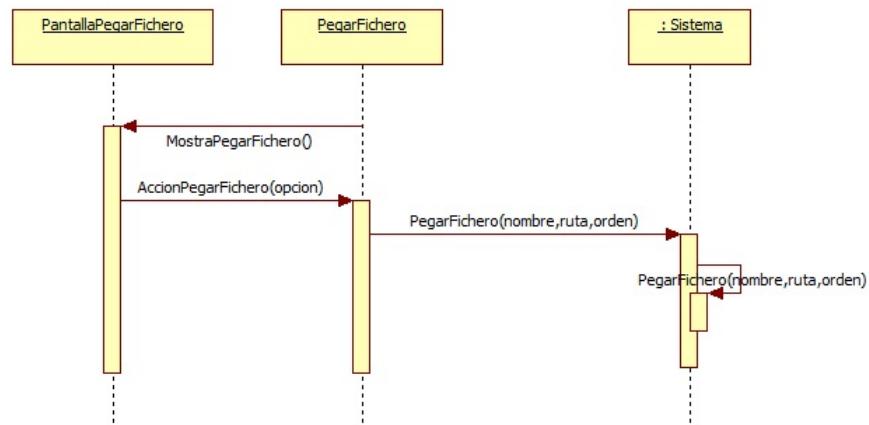


Figura 6.16: Diagrama de interacción “Pegar Fichero”

### 6.2.2.11. Renombrar fichero

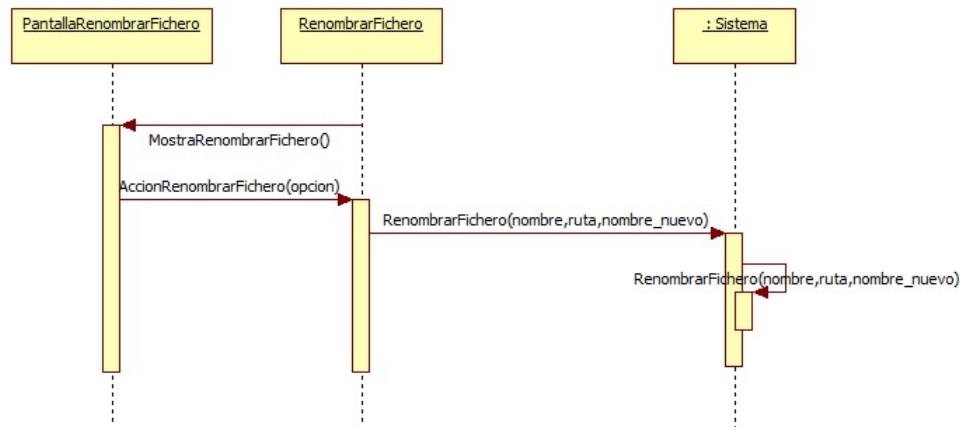


Figura 6.17: Diagrama de interacción “Renombrar Fichero”

### 6.2.2.12. Eliminar fichero

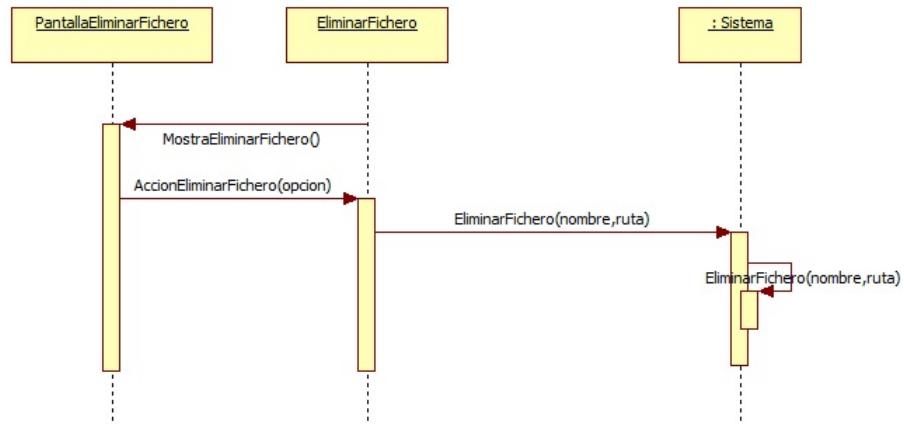


Figura 6.18: Diagrama de interacción “Eliminar Fichero”

### 6.2.2.13. Mostrar directorio atrás

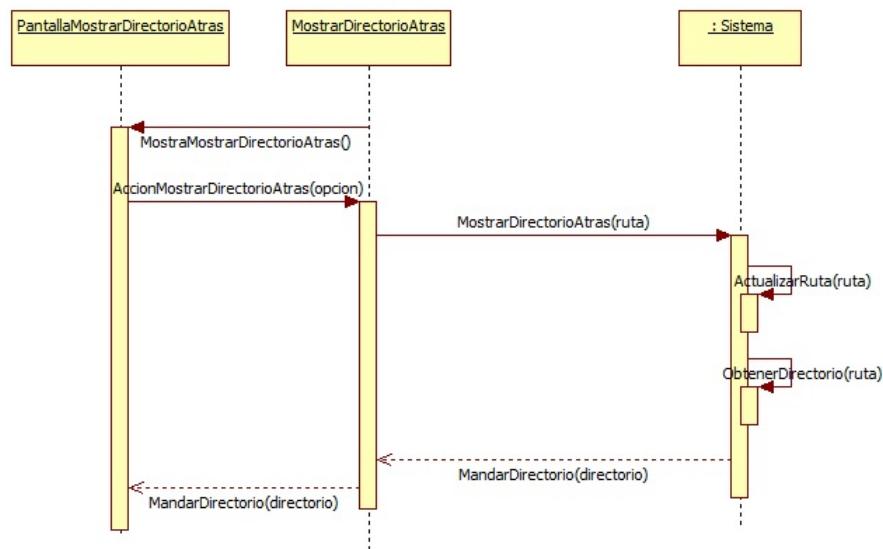


Figura 6.19: Diagrama de interacción “Mostrar directorio atrás”

### 6.2.2.14. Mostrar directorio adelante

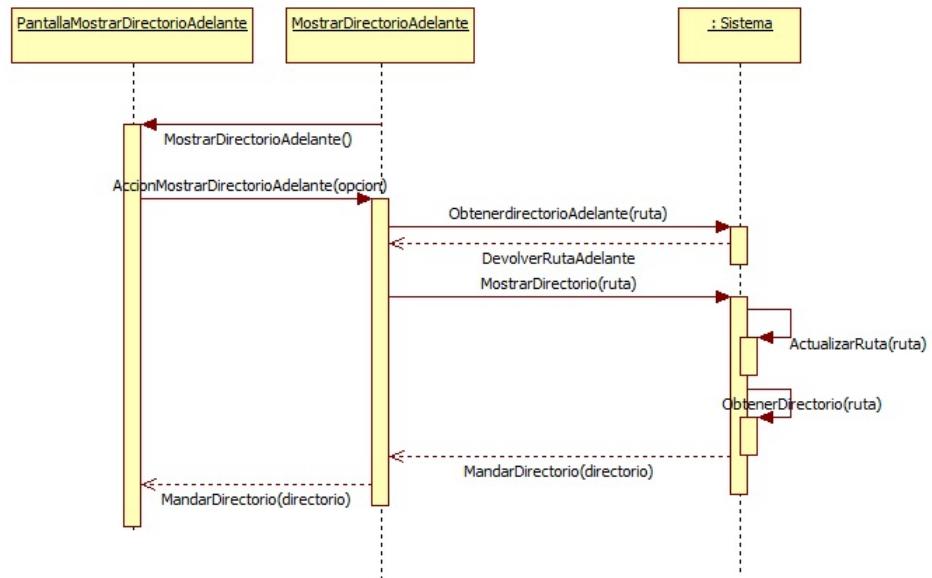


Figura 6.20: Diagrama de interacción “Mostrar directorio adelante”

### 6.2.2.15. Transferir fichero

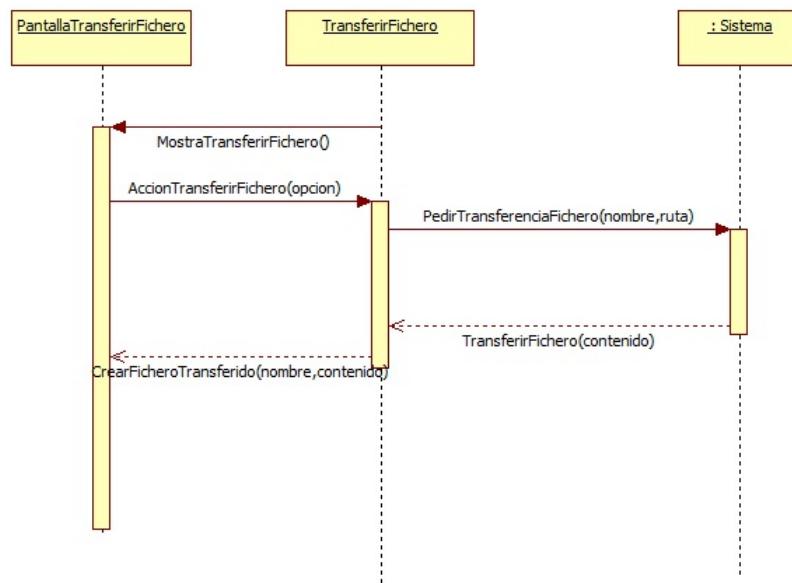


Figura 6.21: Diagrama de interacción “Transferir fichero”

### 6.2.2.16. Listar Scripts

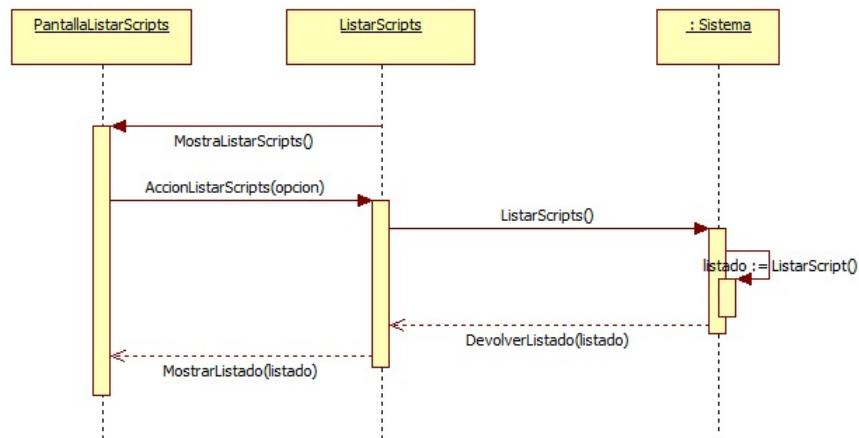


Figura 6.22: Diagrama de interacción “Listar Scripts”

### 6.2.2.17. Ver Script

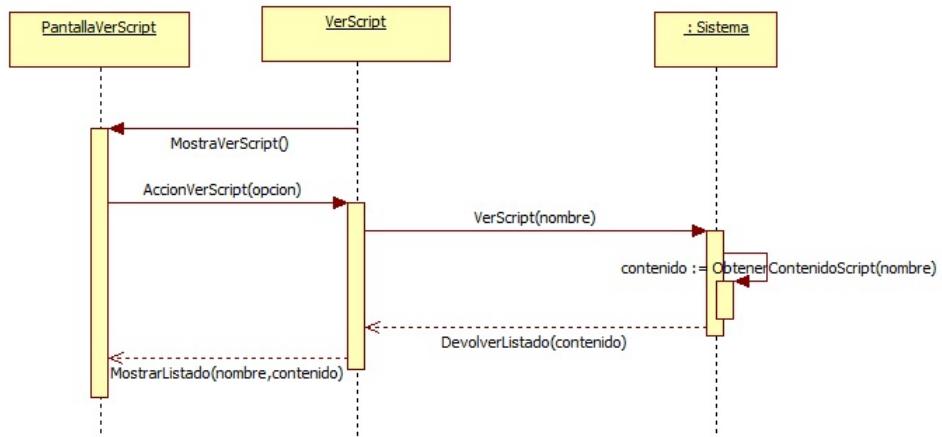


Figura 6.23: Diagrama de interacción “Ver Script”

### 6.2.2.18. Ejecutar Script

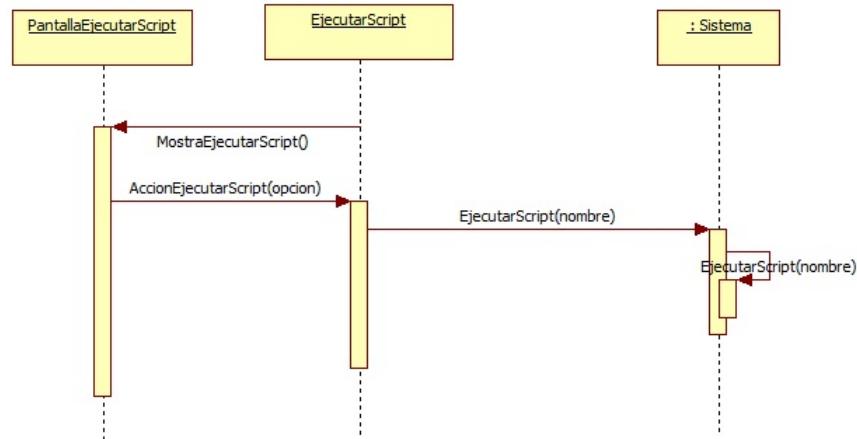


Figura 6.24: Diagrama de interacción “Ejecutar Script”

### 6.2.2.19. Transferir Script

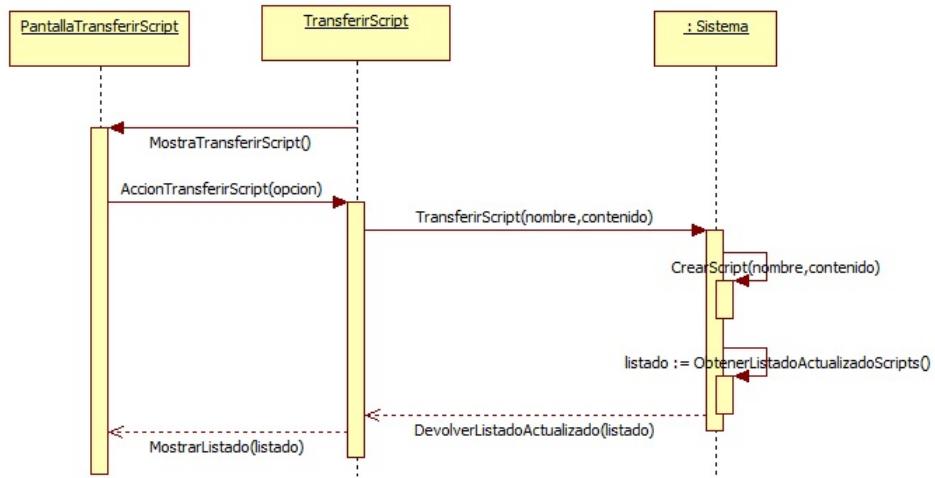


Figura 6.25: Diagrama de interacción “Transferir Script”

### 6.2.2.20. Eliminar Script

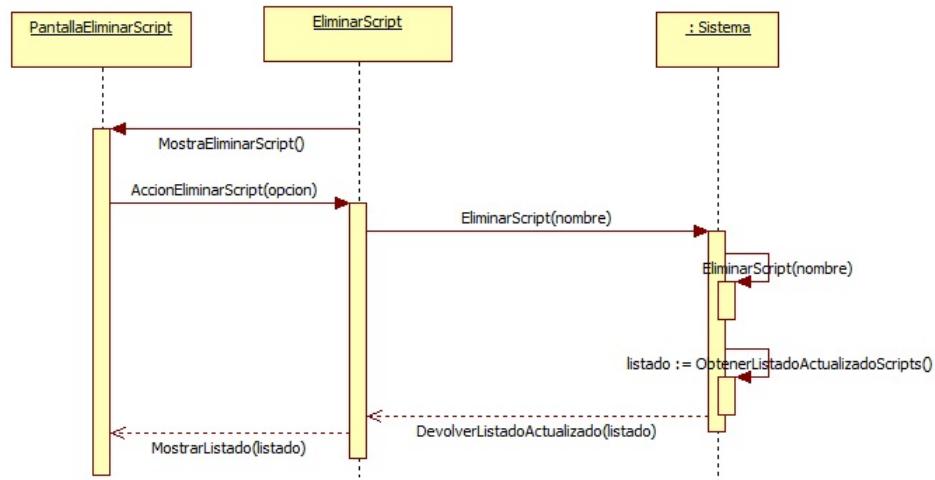


Figura 6.26: Diagrama de interacción “Eliminar Script”

## 6.3. Clases de diseño

Las clases utilizadas para realizar el diseño del proyecto son las siguientes:

1. Servidor.

a) Atributos:

- Puerto\_escucha;
- Puerto\_transferencia;
- Socket;
- DireccionIP;
- DireccionMAC;
- DireccionLocal;

b) Métodos:

- Pruebas(opcion);
- PruebaDisco();
- PruebaRed();

- PruebaProcesos() ;
- PruebaSistemaOperativo();
- PruebaMemoria();
- PruebaOrdenLibre();
- PruebaScripts();
- InformacionDiscos();
- InformaciónRed();
- InformacionSistemaOperativo();
- InformaciónMemoria();
- Scripts();
- Navegacion();
- OrdenLibre();
- ListarProcesos();
- EliminarProceso(int pid);
- ActualizarServidor(puertoescucha,puertotransferencia);
- CambiarPuertoEscucha(int puerto);
- CambiarPuertoTransferencia(int puerto);
- DevolverResultado();
- ListarProcesosActualizado() = ListarProcesos();
- CrearFichero(ruta,nombre,contenido);
- EliminarFichero(nombre,ruta);
- PegarFichero(nombre,ruta,orden);
- RenombrarFichero(nombre,ruta,nombre\_nuevo);
- MostrarDirectorioAtrás(ruta);
- MostrarDirectorioAdelante(ruta);
- ActualizarRuta(ruta);
- ObtenerDirectorio(ruta);
- MandarDirectorio() = DevolverResultado();
- TransferirFichero(contenido);
- ListarScripts();
- DevolverListado(string listado);
- ObtenerContenidoScript(nombre);
- EjecutarScript(nombre);
- CrearScript(nombre,contenido);
- ObtenerListadoActualizadoScripts();
- EliminarScript(nombre);

## 2. Cliente

### a) Atributos:

- IP.
- Puerto.
- ListaHost;

b) Métodos:

- EncenderHost();
- ConectarHost(string ip, int puerto);
- Apagar();
- Reiniciar();
- InformaciónSistema(int opcion);
- InformacionDiscos();
- InformacionRed();
- InformacionMemoria()
- InformacionSistemaOperativo();
- Navegacion(); CrearFichero();
- EjecutarScript(nombre) ;
- Cortar\_CopiarFichero();
- RenombrarFichero();
- VerScript(nombre);
- PegarFichero();
- MostrarDirectorioAtras();
- MostrarDirectorioAdelante(ruta);
- MostrarDirectorioadelante();
- Scripts(); ListarProcesos();
- EliminarProceso(int pid);
- Orden Libre();
- MostrarListado(string listado);
- MostrarDirectorioAtras(ruta);
- RegistrarCortarCopiarFichero(nombre,ruta,orden);
- RenombrarFichero(nombre,ruta,nombre\_nuevo);
- EliminarFichero(nombre,ruta);
- PegarFichero(nombre,ruta,orden);
- PedirTransferenciaFichero(nombre,ruta);
- ListarScripts();
- TransferirScript(nomnbre,contenido);
- EliminarScript(nombre);

3. Orden

a) Atributos:

- nombreOrden;

- Salida\_Orden;

b) Métodos:

- EjecutarOrden();

#### 4. Conexion

a) Atributos:

- IP.
- Puerto.
- FlujoEntrada;
- FlujoSalida;
- Socket;

b) Métodos:

- ConectarHost(ip,puerto);
- MandarApagar();
- MandarReiniciar();
- MandarInformacionSistema();
- MandarInformacionDiscos();
- MandarInformacionRed();
- MandarInformacionMemoria();
- MandarListarProcesos();
- MandarMatarProceso();
- MandarNavegación();
- MandarScript();
- MandarOrdenLibre();
- DevolverResultados();

Podemos observar que hay métodos los cuales será similares y a la hora de la implementación utilizarán la misma nomenclatura. También podemos solapar funciones, por ejemplo, en las pruebas de las funciones del servidor, tendremos un atributo, el cual nos indicará en el servidor si estamos ante una prueba o se debe de mandar la salida al cliente, lo que haría el número de funciones disminuir.

## Capítulo 7

# Implementación del servidor bajo JDK

En este capítulo vamos a describir cómo se obtiene las diferentes informaciones que desde el cliente se pide, es decir, sus funcionalidades. Para ello se utiliza las órdenes del sistema, que accederá a los datos solicitados. Dicha orden variará según el sistema operativo utilizado. Posteriormente comentaremos la utilización de Swing para la interfaz gráfica del servidor y por último, cómo se integró todo para obtener el finalmente el servidor. Comencemos a describir las funciones.

### 7.1. Funcionalidades del sistema

#### 7.1.1. Información de discos

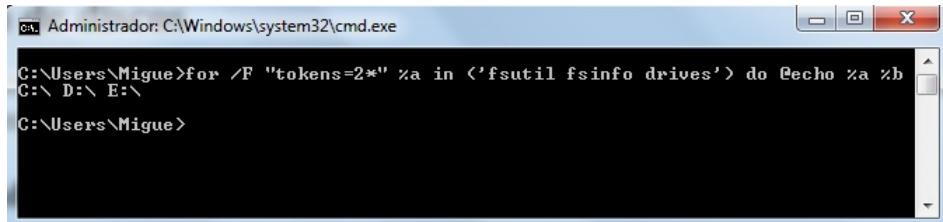
En esta sección vamos a ver cómo se han obtenido la información de las particiones de los discos duros. Las instrucciones utilizadas para la obtención de dicha información son las siguientes.

##### 7.1.1.1. Sistema Windows

Para un sistema Windows, primero tenemos esta instrucción:

```
Process p;
p = Runtime.getRuntime().exec ("cmd /c for /F \"tokens=2*\" %a
                                in ('fsutil fsinfo drives') do @echo %a %b");
```

aquí obtenemos las unidades (drives) que tenemos en el sistema. La salida de dicha orden sería algo como esto:



```
Administrator: C:\Windows\system32\cmd.exe
C:\Users\Migue>for /F "tokens=2*" %a in ('fsutil fsinfo drives') do @echo %a %b
C:\ D:\ E:\

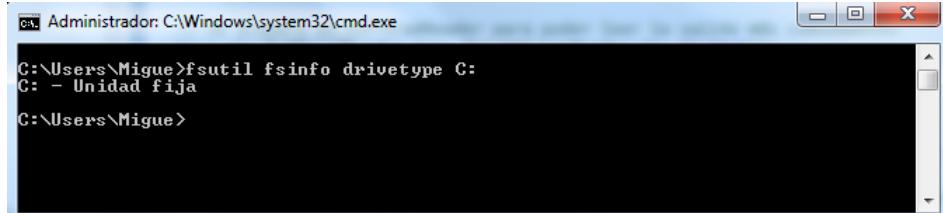
C:\Users\Migue>
```

Figura 7.1: Salida fsinfo drives

Una vez hecho ésto, tendremos que obtener el tipo de unidad que son, mediante la orden:

```
p = Runtime.getRuntime().exec ("cmd /c fsutil fsinfo
drivetype " + datos[i]);
```

cuya salida es:



```
Administrator: C:\Windows\system32\cmd.exe
C:\Users\Migue>fsutil fsinfo drivetype C:
C: - Unidad fija

C:\Users\Migue>
```

Figura 7.2: Salida drivetype

Una vez hecho ésto, registramos aquellas que tienen descripción de “Unidad fija”. Ahora, para cada partición obtenida, le aplicamos ésta orden:

```
Process p1 = Runtime.getRuntime().exec ("cmd /c fsutil fsinfo
volumeinfo " + datos[i]);
```

para obtener la información de la partición, obteniendo una lista como sigue:

```

Administrator: C:\Windows\system32\cmd.exe
Copyright <c> 2009 Microsoft Corporation. Reservados todos los derechos.

C:>fsutil volumeinfo C:
Nombre de volumen: OS
Número de serie de volumen: 0x8edbb0
Longitud de componente máxima: 255
Nombre del sistema de archivos: NTFS
Es compatible con nombres de archivos en mayúsculas y minúsculas
Conservar mayúsculas y minúsculas en los nombres de archivos
Es compatible con Unicode en nombres de archivos
Conserva y aplica ACL
Es compatible con la compresión basada en archivos
Es compatible con cuotas de discos
Es compatible con archivos dispersos
Es compatible con puntos de reprocesamiento
Es compatible con identificadores de objeto
Es compatible con el Sistema de cifrado de archivos
Es compatible con secuencias con nombre
Admite transacciones
Admite vínculos físicos
Admite atributos extendidos
Admite apertura por identificador de archivo
Admite diario USN

C:>
  
```

Figura 7.3: Salida volumeinfo

Posteriormente para ésta misma unidad, se obtienen los datos de espacio como sigue:

```
p1 = Runtime.getRuntime().exec ("cmd /c fsutil volume
diskfree " + datos [ i ]);
```

obteniendo un resultado como sigue:

```

Administrator: C:\Windows\system32\cmd.exe
C:>fsutil volume diskfree C:
Número total de bytes libres : 68272074752
Número total de bytes : 128029028352
Número total de bytes libres disponibles: 68272074752

C:>
  
```

Figura 7.4: Salida diskfree

donde seleccionamos los datos más relevantes de las distintas salidas de órdenes, continuando con todas las unidades.

Con esto obtendríamos, para cada unidad:

- Tipo de sistema de ficheros.
- Información de espacio.

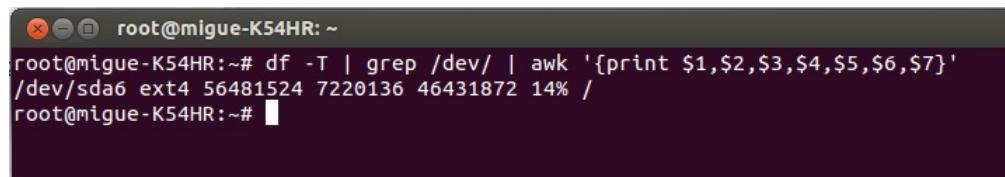
Con todo ello se creará un vector de una estructura de datos, llamada Espacio-Disco, el cual contendrá el nombre de la unidad, el tipo de sistema de ficheros, espacio libre, espacio ocupado y espacio total. Al crearse llamará a una función el cual calculará el porcentaje de espacio ocupado y libre.

### 7.1.1.2. Sistema Linux

Para un sistema Linx, he dado la siguiente orden:

```
String [] command = {"sh","-c","df -T | grep /dev/ | awk '{\n    print \$1,\$2,\$3,\$4,\$5,\$6,\$7}'"};\n\np = Runtime.getRuntime().exec (command);
```

dando como resultado una salida tal que así:



```
root@migue-K54HR:~# df -T | grep /dev/ | awk '{print $1,$2,$3,$4,$5,$6,$7}'\n/dev/sda6 ext4 56481524 7220136 46431872 14% /\nroot@migue-K54HR:~#
```

Figura 7.5: Salida “df -T”

El cual nos da toda la información que queremos obtener sobre la partición del ejemplo “/dev/sda6”.

### 7.1.2. Información de Red

#### 7.1.2.1. Sistema Windows

Para la información de la red utilizamos la orden:

```
Process p;\n\np = Runtime.getRuntime().exec ("cmd /c ipconfig /all");
```

con esta orden tendremos una salida como ésta:

Figura 7.6: Salida ipconfig /all

El siguiente paso sería manejar la salida para obtener los datos que nos interesan. Este procesamiento para obtener dicha información es complejo y se ha procedido de la siguiente manera:

## Código obtención de información de red (1)

```
p = Runtime.getRuntime().exec ("cmd /c ipconfig / all");
is = p.getInputStream();
br = new BufferedReader (new InputStreamReader (is , "Cp850"));

// Se lee la primera linea
aux = br.readLine();
```

```

// Mientras se haya leido alguna linea
while (aux!=null) {
    try{
        while(aux != null && (!aux.startsWith("Adaptador de
            LAN") & !aux.startsWith("Adaptador de Ethernet
            "))){
            aux=br.readLine();
        }

        if(res!=null)
            res=res+"\n";

        if(aux!=null){
            areaPantalla.append("\n\t" + aux + "\n");
            //Adaptador = aux;
            if(res!=null)
                res= res + aux + "\n";
            else
                res= aux + "\n";
            aux=br.readLine();
        }

        while(aux != null && (!aux.startsWith("Adaptador de LAN") &
            aux.startsWith("Adaptador de Ethernet"))){
            if(aux.startsWith("    Dirección física")){
                areaPantalla.append("\t" + aux + "\n");
                res= res + aux + "\n";
            }

            if(aux.startsWith("    Dirección IPv4")){
                areaPantalla.append("\t" + aux + "\n");
                res= res + aux + "\n";
            }

            if(aux.startsWith("    Descripción")){
                areaPantalla.append("\t" + aux + "\n");
                res= res + aux + "\n";
            }

            if(aux.startsWith("    Máscara")){
                areaPantalla.append("\t" + aux + "\n");
                res= res + aux + "\n";
            }

            if(aux.startsWith("    Estado")){
                areaPantalla.append("\t" + aux + "\n");
                res= res + aux + "\n";
            }

            if(aux.startsWith("    DHCP")){
                areaPantalla.append("\t" + aux + "\n");
                res= res + aux + "\n";
            }

            if(aux.startsWith("Adaptador de túnel"))

```

```

        aux=br.readLine();
    }
} catch(NullPointerException e){
    e.printStackTrace();
}
}

```

Lo que hace este código es guardar en una variable, llamada res, todas las interfaces que empiezan con “Adaptador de LAN” o “Adaptador de Ethernet” con las informaciones que queremos obtener de cada una de ellas, las cuales van a ser:

- Nombre del adaptador.
- Dirección IPv4.
- Dirección física.
- Descripción.
- Máscara de subred.
- Estado.
- DHCP.

Una vez tenemos la variable res obtenida, la informaciones de las distintas interfaces están separadas por un retorno de carro. Esta variable res tiene la siguiente forma:



Figura 7.7: Salida de la variable res

Al estar la distinta información desordenada de una adaptador a otro, procedemos a procesar la variable res. Cogemos el primer adaptador y obtenemos los datos, despues procedemos a procesar los adaptadores desde el segundo a el penúltimo y posteriormente procesamos el último. Todo ello ayudado por un vector donde guardaremos los números de líneas donde hay un retorno de carro (índices). El código queda de la siguiente manera:

---

#### Obtención de informacion de red (2)

---

```

Vector<Integer> indices = new Vector<Integer>();
String [] datos;
datos = res.split("\n");

for (int i=0;i<datos.length ; i++){
    if(datos[i].trim().length ()==0)
        indices.add(i);
}

Vector<Red> AdaptadoresRed = new Vector<Red>();

String Adaptador = null ,dirfis = null ,dirip = null ,descripcion =
    null ,mask=null ,estado=null ,dhcp=null ;
for (int i=0;i<indices.get(0) ; i++){
    if(datos[i].contains("Adaptador"))
        Adaptador=datos[i];
    if(datos[i].contains("Dirección física"))
        dirfis=datos[i];
    if(datos[i].contains("Dirección IPv4"))
        dirip=datos[i];
    if(datos[i].contains("Descripción"))
        descripcion=datos[i];
    if(datos[i].contains("Máscara"))
        mask=datos[i];
    if(datos[i].contains("Estado"))
        estado=datos[i];
    if(datos[i].contains("DHCP"))
        dhcp=datos[i];
}

AdaptadoresRed.add(new Red(Adaptador ,dirfis ,dirip ,descripcion ,mask ,
    estado ,dhcp));
Adaptador = dirfis = dirip = descripcion = mask= estado=dhcp=null;

if(indices.size ()>1){
    for (int i =0 ; i<indices.size ()-1;i++){
        for (int j=indices.get(i);j<indices.get(i+1);j++){
            if(datos[j].contains("Adaptador"))
                Adaptador=datos[j];
            if(datos[j].contains("Dirección física"))
                dirfis=datos[j];
            if(datos[j].contains("Dirección IPv4"))
                dirip=datos[j];
            if(datos[j].contains("Descripción"))
                descripcion=datos[j];
            if(datos[j].contains("Máscara"))

```

```

        mask=datos[j];
        if(datos[j].contains("Estado"))
            estado=datos[j];
        if(datos[j].contains("DHCP"))
            dhcp=datos[j];
    }
    AdaptadoresRed.add(new Red(Adaptador, dirfis, dirip,
                                descripcion, mask, estado, dhcp));
}

Adaptador = dirfis = dirip = descripcion = mask= estado=dhcp=null;

for(int i=indices.get(indices.size()-1);i<datos.length;i++){
    if(datos[i].contains("Adaptador"))
        Adaptador=datos[i];
    if(datos[i].contains("Dirección física"))
        dirfis=datos[i];
    if(datos[i].contains("Dirección IPv4"))
        dirip=datos[i];
    if(datos[i].contains("Descripción"))
        descripcion=datos[i];
    if(datos[i].contains("Máscara"))
        mask=datos[i];
    if(datos[i].contains("Estado"))
        estado=datos[i];
    if(datos[i].contains("DHCP"))
        dhcp=datos[i];
}
AdaptadoresRed.add(new Red(Adaptador, dirfis, dirip, descripcion, mask,
                           estado, dhcp));

```

Se utiliza un vector de Adaptadores de red el cual contendrá toda la información registrada de los diferentes adaptadores, para posteriormente enviarlo al cliente.

#### 7.1.2.2. Sistema Linux

Para un sistema Linx, he dado la siguiente orden:

```

String[] command = {"sh", "-c", "ifconfig | grep \"Link encap\""
                    | awk '{ print $1}'"};
p = Runtime.getRuntime().exec (command);

```

dando como resultado una salida tal que así:

```
root@migue-K54HR:~# ifconfig | grep "Link encap" | awk '{print $1}'
eth0
lo
wlan0
root@migue-K54HR:~#
```

Figura 7.8: Salida “ifconfig | grep \”Link encap\” | awk ’{print \$1}’”

Ahora, para cada una de las interfaces, ejecutamos un ifconfig, y seleccionamos los datos que queremos:

```
root@migue-K54HR:~# ifconfig eth0
eth0      Link encap:Ethernet direcciónHW c8:60:00:48:a8:3a
          ACTIVO DIFUSIÓN MULTICAST MTU:1500 Métrica:1
          Paquetes RX:0 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:0 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colatX:1000
          Bytes RX:0 (0.0 B) TX bytes:0 (0.0 B)
          Interrupción:52

root@migue-K54HR:~#
```

Figura 7.9: Salida “ifconfig”

Además para saber el estado de la interfaz de red utilizamos el comando “ifplugstatus”.

### 7.1.3. Información del Sistema Operativo

Esta información es obtenida mediante la API de java. Hay muchas cosas que podemos saber del sistema operativo desde dicha API. Se obtiene dentro de la clase System y dentro de ella, con el método getProperties(). En el código de obtención de dicha información se puede ver la descripción de las diferentes propiedades del sistema que podemos visualizar:

Obtención de informacion del sistema

---

```
private void InformacionSistemaOperativo(){
    Vector<Propiedad> Propiedades = new Vector<Propiedad>();
    Propiedades.add(new Propiedad("Version Java" , "java.version"));
```

```

Propiedades.add(new Propiedad("Fabricante","java.vendor"));
Propiedades.add(new Propiedad("URL Fabricante","java.vendor.url
"));
Propiedades.add(new Propiedad("Directorio de instalación","java
.home"));
Propiedades.add(new Propiedad("JVM versión","java.vm.
specification.version"));
Propiedades.add(new Propiedad("Fabricante JVM","java.vm.
specification.vendor"));
Propiedades.add(new Propiedad("Nombre JVM","java.vm.
specification.name"));
Propiedades.add(new Propiedad("Version implementación JVM","
java.vm.version"));
Propiedades.add(new Propiedad("Fabricante implementación JVM","
java.vm.vendor"));
Propiedades.add(new Propiedad("Version implementación Java","
java.vm.name"));
Propiedades.add(new Propiedad("Version JRE","java.specification
.version"));
Propiedades.add(new Propiedad("Fabricante JRE","java.
specification.vendor"));
Propiedades.add(new Propiedad("Nombre JRE","java.specification.
name"));
Propiedades.add(new Propiedad("Número de la versión de la clase
Java","java.class.version"));
Propiedades.add(new Propiedad("Java Path","java.class.path"));
Propiedades.add(new Propiedad("Path Librerías Java","java.
library.path"));
Propiedades.add(new Propiedad("Directorio temporal por defecto"
,"java.io.tmpdir"));
Propiedades.add(new Propiedad("Path del directorio de
extensiones","java.ext.dirs"));
Propiedades.add(new Propiedad("Nombre del Sistema Operativo","
os.name"));
Propiedades.add(new Propiedad("Arquitectura del Sistema
Operativo","os.arch"));
Propiedades.add(new Propiedad("Version del Sistema Operativo","
os.version"));
Propiedades.add(new Propiedad("Nombre de la cuenta de usuario",
"user.name"));
Propiedades.add(new Propiedad("Directorio de casa del usuario",
"user.home"));
Propiedades.add(new Propiedad("Directorio actual de trabajo",
"user.dir"));

for(int i=0;i<Propiedades.size();i++)
    Enviar(Propiedades.get(i).Nombre()+" "+System.
    getProperty(Propiedades.get(i).Orden()) );
}

}

```

siendo la estructura de datos para almacenar tanto la descripción como la orden que se da para obtenerla en un Vector de “Propiedades” quedando definida ésta como sigue:

---

Clase Propiedad

---

```

public class Propiedad {
    private String nombre;
    private String orden;

    Propiedad(String N, String O){
        nombre=N;
        orden=O;
    }
}

```

```

    }

public String Nombre(){
    return nombre;
}

public String Orden(){
    return orden;
}
}

```

#### 7.1.4. Información de memoria

##### 7.1.4.1. Sistema Windows

La obtención de ésta información la obtenemos de la siguiente manera:

---

###### Obtención de informacion de memoria en Windows

---

```

String memftotal=null , memfdis=null , memvirtamax=null , memvirdis=null
, memvirus=null;

p = Runtime.getRuntime() . exec ("cmd /c systeminfo");
is = p.getInputStream();

br = new BufferedReader (new InputStreamReader (is , "Cp850"));
// Se lee la primera linea
aux = br.readLine();

// Mientras se haya leido alguna linea
while (aux!=null) {
    if(SistemaOperativo.equals("Windows")){
        if(aux.startsWith("Cantidad total de memoria física
")){
            String datos [];
            datos=aux.split(":");
            memftotal=datos [1];
        }

        if(aux.startsWith("Memoria física disponible")){
            String datos [];
            datos=aux.split(":");
            memfdis=datos [1];
        }

        if(aux.startsWith("Memoria virtual: tamaño")){
            String datos [];
            datos=aux.split(":");
            memvirtamax=datos [2];
        }

        if(aux.startsWith("Memoria virtual: disponible")){
            String datos [];
            datos=aux.split(":");
            memvirdis=datos [2];
        }
    }
}

```

```

    }

    if(aux.startsWith("Memoria virtual: en uso")){
        String datos[];
        datos=aux.split(":");
        memvirus=datos[2];
    }
}

```

como podemos observar, la información de memoria que obtenemos mediante la orden “systeminfo” y que tomamos para enviarla al cliente es:

- Cantidad total de memoria física.
- Memoria física disponible.
- Tamaño de la memoria virtual.
- Memoria virtual disponible.
- Memoria virtual en uso.

En la siguiente figura podemos ver la salida que produce la orden, junto con la información que de ella queremos mandar al cliente:

```

C:\Users\Migue>systeminfo
Nombre de host: MIGUE-PC
Nombre del sistema operativo: Microsoft Windows 7 Home Premium
Versión del sistema operativo: 6.1.7601 Service Pack 1 Compilación 7601
Fabricante del sistema operativo: Microsoft Corporation
Configuración del sistema operativo: Estación de trabajo independiente
Tipo de compilación del sistema operativo: Multiprocessor Free
Propiedad de: Migue
Organización registrada:
Id. del producto: 00359-OEM-8992687-00007
Fecha de instalación original: 10/07/2012, 2:15:38
Tiempo de arranque del sistema: 04/10/2012, 18:37:07
Fabricante del sistema: ASUSTeK Computer Inc.
Modelo el sistema: K54HR
Tipo de sistema: x64-based PC
Procesador(es):
  1 Procesadores instalados.
    [0]: Intel64 Family 6 Model 42 Stepping 7 GenuineIntel ~2100 MHz
    Versión del BIOS: American Megatrends Inc. K54HR.203, 2/02/2012
Directorio de Windows: C:\Windows
Directorio de sistema: C:\Windows\system32
Dispositivo de arranque: \Device\HarddiskVolume2
Configuración regional del sistema: es:Español (internacional)
Idioma de entrada: es:Español (tradicional)
Zona horaria: (UTC+01:00) Bruselas, Copenhague, Madrid, París
Cantidad total de memoria física: 4.072 MB
Memoria física disponible: 2.413 MB
Memoria virtual: tamaño máximo: 8.142 MB
Memoria virtual: disponible: 6.310 MB
Memoria virtual: en uso: 1.832 MB
Opciones de archivo de paginación:
Dominio: WORKGROUP
Servidor de inicio de sesión: \\MIGUE-PC
Revisión(es): 136 revision(es) instaladas.
  [01]: KB982861
  [02]: KB982861
  [03]: KB982861

```

Figura 7.10: Salida de la orden “systeminfo”

#### 7.1.4.2. Sistema Linux

Para obtener la información de la memoria para un sistema Linux, hemos accedido a un fichero del sistema localizado en la ruta /proc/meminfo. El resultado de la orden cat de este fichero es el siguiente:

```

root@migue-K54HR:~# cat /proc/meminfo
MemTotal:        4097504 kB
MemFree:         3149504 kB
Buffers:          63808 kB
Cached:           396748 kB
SwapCached:       0 kB
Active:           526812 kB
Inactive:         341300 kB
Active(anon):    408272 kB
Inactive(anon):   680 kB
Active(file):    118540 kB
Inactive(file):  340620 kB
Unevictable:      0 kB
Mlocked:          0 kB
HighTotal:        3258900 kB
HighFree:         2422024 kB
LowTotal:         838604 kB
LowFree:          727480 kB
SwapTotal:        4168700 kB
SwapFree:         4168700 kB
Dirty:             172 kB
Writeback:         0 kB
AnonPages:        407552 kB
Mapped:            103852 kB
Shmem:             1400 kB
Slab:              33072 kB
SReclaimable:     16212 kB
SUnreclaim:       16860 kB
KernelStack:      2752 kB
PageTables:       6412 kB
NFS_Unstable:      0 kB
Bounce:             0 kB
WritebackTmp:      0 kB
Commitlimit:      6217452 kB
Committed_AS:    2037168 kB
/mallocTotal:     122880 kB
/mallocUsed:      14024 kB
/mallocChunk:     105332 kB
HardwareCorrupted: 0 kB

```

Figura 7.11: Salida de la orden “cat /proc/meminfo”

posteriormente se procedería a localizar los datos que interesan y obtenerlos.

#### 7.1.5. Procesos

##### 7.1.5.1. Sistema Windows

Para obtener la información de los procesos en ejecución del sistema en el sistema Windows utilizamos el siguiente código:

```
p = Runtime.getRuntime().exec ("cmd /c for /F \"tokens=1,2,5\"%a in ('tasklist /NH /FO TABLE ^| sort') do @echo %a %c");

```

es decir, obtenemos los tokens 1,2 y 5 de la salida de la orden “tasklist /NH /FO TABLE | sort”, dando una salida parecida a ésta:

Nombre Proceso	Tipo	PID	Tamaño	Estado
ACEngSvr.exe	Console	3096	5.800 KB	1
ACMON.exe	Console	2776	8.032 KB	1
AmicoSinglun64.exe	Console	2820	5.808 KB	1
armsvc.exe	Services	2008	3.848 KB	0
AsLdrSrv.exe	Services	1252	3.036 KB	0
AsScrPro.exe	Console	3164	5.040 KB	1
aticlxx.exe	Console	1404	6.336 KB	1
atiessrxx.exe	Services	928	4.140 KB	0
ATKOSD.exe	Console	4000	5.756 KB	1
ATKOSD2.exe	Console	1536	528 KB	1
BatteryLife.exe	Console	2988	528 KB	1
CLMISvc.exe	Console	3484	7.624 KB	1
cmd.exe	Console	4612	2.900 KB	1
conhost.exe	Services	1208	2.616 KB	0
conhost.exe	Services	2724	2.796 KB	0
conhost.exe	Console	4404	5.056 KB	1
csrss.exe	Services	456	4.428 KB	0
cssrss.exe	Console	564	11.148 KB	1
dllhost.exe	Services	4396	6.708 KB	0
DMedia.exe	Console	1172	4.820 KB	1
dum.exe	Console	1616	28.704 KB	1
eclipse.exe	Console	1612	3.232 KB	1
ETDCtr1.exe	Console	2808	12.248 KB	1
ETDCtr1Helper.exe	Console	3596	6.588 KB	1
explorer.exe	Console	1708	75.768 KB	1
FBAgent.exe	Services	1192	12.180 KB	0
firefox.exe	Console	3752	127.968 KB	1
FlashPlayerPlugin_11_4_40	Console	2612	9.832 KB	1
FlashPlayerPlugin_11_4_40	Console	3692	15.328 KB	1
GPNESSrv.exe	Services	1300	2.516 KB	0
HControl.exe	Console	1500	7.448 KB	1
HControlUser.exe	Console	1168	3.332 KB	1
InsOnSrv.exe	Services	1060	6.224 KB	0
InsOnWMI.exe	Console	3622	6.604 KB	1
javaw.exe	Console	4904	443.992 KB	1
jsched.exe	Console	2200	4.292 KB	1
KBFiltr.exe	Console	3028	3.916 KB	1
LMS.exe	Services	4924	4.816 KB	0

Figura 7.12: salida de la orden “tasklist /NH /FO TABLE | sort”

como podemos observar la salida está ordenada por orden alfabetico según el proceso y el procesamiento de la salida sería para obtener los datos del nombre del proceso, el pid y el tamaño que tiene, correspondientes a las columnas 1,2 y 5.

### 7.1.5.2. Sistema Linux

Para la obtención del listado de procesos activos en el sistema se realiza con el siguiente comando:

```
String [] command = {"sh","-c","ps -A -l --sort cmd | awk '{\n    print $4,$14,$10}'}";\n\np = Runtime.getRuntime().exec (command);
```

con la cual, obtenemos una salida como sigue:

```

root@migue-K54HR:~# ps -A -l --sort cmd | awk '{print $4,$14,$10}'
PID CMD SZ
789 NetworkManager 8174
1131 Xorg 18491
1157 accounts-daemon 3954
921 acpid 539
920 anacron 594
2187 apt 554
22 ata_sff 0
948 atd 613
822 avahi-daemon 859
823 avahi-daemon 859
2415 awk 601
1711 bamfdaemon 19557
2400 bash 1702
19 bdi-default 0
1674 bluetooth-apple 19932
766 bluetoothd 1181
1860 cat 1357
499 cfg80211 0
1437 colord 13370
1656 compiz 60138
1174 console-kit-dae 7529
13 cpuset 0
947 cron 650
33 crypto 0
819 cupsd 2765
740 dbus-daemon 1026
1630 dbus-daemon 1581
1629 dbus-launch 980
2136 dconf-service 8374
2155 deja-dup 29993
2046 deja-dup-monito 10974

```

Figura 7.13: salida de la orden “ps -A -l –sort cmd | awk '{print \$4,\$14,\$10}'”

al tener la utilidad awk, podemos elegir las columnas que queremos para nuestro resultado, como son el PID, CMD y SZ, es decir, el PID, el comando ejecutado para obtener ese proceso y el tamaño del mismo.

### 7.1.6. Navegación

En éste apartado, el servidor se encargará de ejecutar algunas acciones según el cliente pida, las cuales son:

- “MostrarMiPc()”: Esta función, mostrará la pantalla inicial de navegador. En Windows, mostrará las unidades de discos para poder empezar la navegación por ellos. En cambio, en linux, mostrará el contenido del directorio raíz.
- “ActualizarGridView()”: Esta función va a recibir del cliente un String que es la ruta donde el cliente está navegando, y manda una lista de los ficheros contenidos en esa ruta, ejecutando el comando dir en Windows o ls en Linux.

- “EjecutarFichero()”: Dicha función recibe del cliente los datos referentes a la ruta, nombre del fichero a ejecutar y argumentos. Los ficheros ejecutables en Windows son aquellos que tienen extensiones “.java”, “.bat”, “.exe”, “.py” y en Linux tenemos los ficheros con extensión “.java”, “.run”, “.bin”, “.sh” y “.py”.
- “TransferirFichero()”: función que recibe del cliente los parámetros de ruta y nombre del fichero y transfiere dicho fichero al cliente, creándolo en la SD-Card del dispositivo móvil.
- “CopiarFichero()”: función que recibe del cliente el fichero origen de la copia y el directorio de destino de la misma.
- “CorteFichero()”: función que recibe del cliente el fichero origen del corte y el directorio de destino de la misma.
- “EliminarFichero()”: esta función recibe del cliente la ruta y el nombre del fichero a eliminar, el servidor, se encargará de procesar la orden para la eliminación del mismo.
- “RenombrarFichero()”: función que recibe la ruta, el nombre del fichero a renombrar y el nombre nuevo del fichero a renombrar. El servidor se encargará de ejecutar la orden adecuada según el sistema para realizarla.
- “CrearFichero()”: función que recibe la ruta, el nombre y el contenido del fichero a crear, y el servidor procede a ejecutar la orden encargada para la creación del mismo.

La función principal que controla la ejecución de todas estas funciones viene implementada con el siguiente código:

---

Procesamiento de la Navegación

---

```

private void ProcesarNavegacion() throws IOException{
    areaPantalla.append( "\n      —> Navegación por árbol de
        directorios comenzada....\n" );
}

try {
    // Leemos entrada desde Navegador
    mensaje = (String) entrada.readObject();
    while(!mensaje.equals("Salir_Navegacion")){
        if(mensaje.equals("MostrarMiPc")){
            MostrarMiPc();
        }

        if(mensaje.equals("ActualizarGridView")){
            ActualizarGridView();
        }

        if(mensaje.equals("EjecutarFichero")){
            EjecutarFichero();
        }
    }
}

```

```

        if(mensaje.equals("TransferirFichero")){
            TransferirFichero();
        }

        if(mensaje.equals("Copia")){
            CopiarFichero();
        }

        if(mensaje.equals("Corte")){
            CorteFichero();
        }

        if(mensaje.equals("EliminarFichero")){
            EliminarFichero();
        }

        if(mensaje.equals("RenombrarFichero")){
            RenombrarFichero();
        }

        if(mensaje.equals("CrearFichero")){
            CrearFichero();
        }

        mensaje = (String) entrada.readObject();
    }

    areaPantalla.append("\n\n    —> Navegación por
                        árbol de directorios terminada.\n\n");
}

} catch (IOException e1) {
    // TODO Auto-generated catch block
    areaPantalla.append("\n    Se ha terminado la
                        conexión...");  

    start();
    e1.printStackTrace();
} catch (ClassNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
    areaPantalla.append("\nNavegacion ClassNotFoundException");
}
}

```

Ahora procedemos a comentar y visualizar el código de las distintas funciones antes mencionadas del apartado “Navegación”.

#### 7.1.6.1. MostrarMiPc()

Esta función se encarga de obtener los datos del primer contenido que tendrá el cliente al iniciar la navegación por el árbol de directorios. En un sistema Windows, obtendrá las diferentes unidades de disco que dispone el sistema. Para el caso de un sistema Linux, obtendrá el contenido del directorio raíz. El código de la función es el siguiente:

---

MostrarMiPc()

---

```

private void MostrarMiPc(){
    // Mostramos MiPC (unidades)

    String [] datos;
    try {
        areaPantalla.append("\n\t Mostrando MiPc... ");
        if(SistemaOperativo.equals("Windows")){
            p = Runtime.getRuntime().exec ("cmd /c for /F
                \tokens=2*%>in(\`fsutil fsinfo drives
                \`) do @echo %&%b");
        } else{
            String [] command = {"sh","-c","ls -1 / | awk '{
                print $1,$9}'"};
            p = Runtime.getRuntime().exec (command);
        }

        // Se obtiene el stream de salida del programa
        is = p.getInputStream();
        // Se prepara un bufferedReader para poder leer la
        // salida más comodamente.
        br = new BufferedReader (new InputStreamReader (is));
        ;
        // Se lee la primera linea
        aux = br.readLine();

        // Mientras se haya leido alguna linea
        while (aux!=null) {
            datos = aux.split(" ");
            if(SistemaOperativo.equals("Windows")){
                for(int i=0;i<datos.length ; i++){
                    p = Runtime.getRuntime().exec ("cmd /c
                        fsutil fsinfo drivetype " + datos[i]);
                    // Se obtiene el stream de salida del
                    // programa
                    is = p.getInputStream();

                    // Se prepara un bufferedReader para poder
                    // leer la salida más comodamente.
                    br = new BufferedReader (new
                        InputStreamReader (is , "Cp850"));

                    // Se lee la primera linea
                    aux = br.readLine();
                    if(aux.equals(datos[i] + " - Unidad fija"))
                    ){
                        salida.flush();
                        salida.writeObject(datos[i] +
                            Unidadfija);
                    }
                    if(aux.equals(datos[i] + " - Unidad de CD-

```

```

        ROM"")){
            salida.flush();
            salida.writeObject(datos[i] + " CD-
ROM");
        }
    }

} else{
    if (!datos[0].startsWith("total")){
        salida.flush();
        salida.writeObject(datos[1] + " " + datos
[0]);
    }
}

aux = br.readLine();
}

Enviar("Fin_MostrarMiPc");
} catch ( IOException excepcionES ) {
    excepcionES.printStackTrace();
}
}

```

Como podemos observar se envía en Windows el nombre de la unidad y si es de CD-ROM o una unidad fija, para poder distinguir a la hora de visualizarlo en el navegador del cliente. En el caso de un sistema Linux, obtenemos la información de los ficheros contenidos en el directorio raíz mediante un “ls -l” y posteriormente la enviamos al cliente.

#### 7.1.6.2. ActualizarGridView()

Esta función recibirá del cliente el directorio el cual el cliente quiere obtener el listado de ficheros contenido en el. Así pues, se procederá a listarlos y enviarlos de nuevo al cliente, quedando el código como sigue:

---

```

    ActualizarGridView()


---


private void ActualizarGridView(){
    try {
        mensaje = (String) entrada.readObject();
        areaPantalla.append("\n\t• (ActualizarGridView)
        Directorio: " + mensaje);
        if(SistemaOperativo.equals("Windows"))
            p = Runtime.getRuntime().exec ("cmd /c for
            /F "tokens=3*\"%a" in ('dir " + "\"
            + mensaje + "\" + "\") do @echo %a %b"
            );
        else{
            String [] command = {"sh", "-c", "ls -1 \"/" +
                mensaje + "\n" | awk '{ for (x=1; x<=NF
                ; x++) { if(x==1 || x>=9){ printf $x \
                \"\n; } }; print \"\\\" }'"};

```

```

        p = Runtime.getRuntime().exec (command);
    }

    // Se obtiene el stream de salida del programa
    is = p.getInputStream();
    // Se prepara un bufferedReader para poder leer la
    // salida más comodamente.
    if(SistemaOperativo.equals("Windows"))
        br = new BufferedReader (new
            InputStreamReader (is , "Cp850"));
    else{
        br = new BufferedReader (new
            InputStreamReader (is));
    }

    // Se lee la primera linea
    aux = br.readLine();
    // Mientras se haya leido alguna linea
    if(SistemaOperativo.equals("Windows")){
        while (aux!=null) {
            if(aux.endsWith("bytes") || aux.
                endsWith("bytes libres")){
            }else{
                salida.flush();
                salida.writeObject(aux);
            }
            aux = br.readLine();
        }
    }else{
        while (aux!=null) {
            if (!aux.startsWith("total")){
                salida.flush();
                salida.writeObject(aux);
            }
            aux = br.readLine();
        }
    }

    salida.flush();
    salida.writeObject("fin_listado_dir");

}catch (ClassNotFoundException | IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
}

```

Podemos observar que evitamos enviar las líneas que no corresponden exactamente a la información de ficheros.

#### 7.1.6.3. EjecutarFichero()

Esta función recibe tres datos por parte del cliente que son: la ruta, el fichero ejecutable y los parámetros. Luego se procede a identificar el sistema operativo y la extensión del fichero para poder saber la sintaxis de ejecución que tiene dicho fichero. A continuación inserto el código.

---

EjecutarFichero()

---

```

private void EjecutarFichero(){
    try {
        mensaje = (String) entrada.readObject();
        String Ruta=mensaje;

        mensaje = (String) entrada.readObject();
        String FicheroEjecutable=mensaje;

        mensaje = (String) entrada.readObject();
        String argumentos=mensaje;
        areaPantalla.append("\n\t Ejecutar Fichero:\tRuta: " +
            Ruta + "\tEjecutable: " + FicheroEjecutable + \
            tArgumentos: " + argumentos);

        // Compilamos y ejecutamos
        String eje [] = FicheroEjecutable.split("\\.");
        String Ejecutable = eje [0];
        String Extension = eje [1];

        if(SistemaOperativo.equals("Windows")){
            // Ejecución de Ficheros .java en Windows
            if(Extension.equals("java")){
                areaPantalla.append("\n\t ·Se ejecuta: cmd /c cd \""
                    " + Ruta + "\" && " + "javac \"\" + Ejecutable +
                    "." + Extension + "\" && java \"\" + Ejecutable +
                    "\" " + argumentos);
                p = Runtime.getRuntime().exec ("cmd /c cd \"\" +
                    Ruta + "\" && " + "javac \"\" + Ejecutable + "."
                    + Extension + "\" && java \"\" + Ejecutable + "
                    "\" " + argumentos);
            }
            // Ejecución de Ficheros .exe y .bat en Windows
            if(Extension.equals("exe") || Extension.equals("bat")){
                areaPantalla.append("\n\t ·Se ejecuta: cmd /c " +
                    "\" + Ruta + Ejecutable + "." + Extension + "\""
                    " + argumentos);
                p = Runtime.getRuntime().exec ("cmd /c " + "\" +
                    Ruta + Ejecutable + "." + Extension + "\" " +
                    argumentos);
            }
            // Ejecución de Ficheros .py en Windows
            if(Extension.equals("py")){
                areaPantalla.append("\n\t ·Se ejecuta: cmd /c " +
                    "cd \"\" + Ruta + "\" && python \"\" +
                    FicheroEjecutable + "\" " + argumentos);
                p = Runtime.getRuntime().exec ("cmd /c " + "cd \"\" +
                    Ruta + "\" && python \"\" + FicheroEjecutable +
                    "\" " + argumentos );
            }
        }else{
            // Ejecución de Ficheros .java en Linux
            if(Extension.equals("java")){
                areaPantalla.append("\n\t ·Se ejecuta: javac \"\" +
                    Ruta + "/" + Ejecutable + ".java\" && cd \"\" +
                    Ruta + "\" && " + "java \"\" + Ejecutable + \"\"");
            }
        }
    }
}

```

```

        " + argumentos);
    String [] command = {"sh", "-c", "javac \\" + Ruta +
        "/" + Ejecutable + ".java\\" && cd \\" + Ruta +
        "\\" && " + "java \\" + Ejecutable + "\\" +
        argumentos};
    p = Runtime.getRuntime().exec (command);
}
// Ejecución de Ficheros .bin, .run y .sh en Linux
if(Extension.equals("bin") || Extension.equals("run")
|| Extension.equals("sh"))
areaPantalla.append("\n\t ·Se ejecuta: cd \\" +
    Ruta + "\\" && "./" + FicheroEjecutable + "\\" +
    + argumentos);
String [] command = {"sh", "-c", "cd \\" + Ruta + "\\" +
    && "./" + FicheroEjecutable + "\\" +
    argumentos};
p = Runtime.getRuntime().exec (command);
}
// Ejecución de Ficheros .py en Linux
if(Extension.equals("py")){
    areaPantalla.append("\n\t ·Se ejecuta: cd \\" +
        Ruta + "\\" && python \\" + FicheroEjecutable +
        "\\" +
        argumentos);
    String [] command = {"sh", "-c", "cd \\" + Ruta + "\\" +
        && python \\" + FicheroEjecutable + "\\" +
        argumentos};
    p = Runtime.getRuntime().exec (command);
}
}
} catch (ClassNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
}
```

#### 7.1.6.4. TransferirFichero()

Para la transferencia de un fichero a la SD-Card, creamos una clase que implementa la interfaz Runnable, es decir, clase que utilizamos como hilo y que, para su creación, va a recibir como parámetros la ruta, el nombre del fichero y el puerto de transferencia. Una vez hecho esto se lanza el hilo, el cual espera a la conexión desde el cliente en ese puerto de transferencia. El siguiente paso, cuando se recibe la conexión desde el cliente es transferir el contenido del fichero y una vez hecho esto el hilo termina. El código quedaría como sigue:

TransferirFichero()

```
private class Transferirfichero implements Runnable{  
    private String Ruta;  
    private String Fichero;  
    private int PuertoT;
```

```

Transferirfichero(String r ,String f ,int pto){
    Ruta=r;
    Fichero=f;
    PuertoT=pto;
    try {
        servidortrans= new ServerSocket(PuertoT);
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    Thread t=new Thread (this);
    t.start();
}

@Override
public void run() {
    // TODO Auto-generated method stub
    areaPantalla.append( "\n\n    Esperando una conexión en
        el puerto " + String.valueOf(PuertoT) + "... \n\n");
    Enviar("ListoTransferencia:" + PuertoT);
    try{
        conexiontrans = servidortrans.accept();
        // Informamos de la Conexión recibida desde el
        terminal

        areaPantalla.append( "\n    Conexión trans" +
            " recibida de: " + conexiontrans.getInetAddress()
            .getHostName() + "\n");

        areaPantalla.append("\n\t Transferencia de fichero
            \tRuta: " + Ruta + "\tEjecutable: " + Fichero
            );

        final BufferedOutputStream outStream = new
            BufferedOutputStream(conexiontrans.
            getOutputStream());
        final BufferedInputStream inStream;
        if(SistemaOperativo.equals("Windows"))
            inStream = new BufferedInputStream(new
                FileInputStream(Ruta+Fichero));
        else
            inStream = new BufferedInputStream(new
                FileInputStream(Ruta+"/"+Fichero));

        final byte[] buffer = new byte[4096];
        for (int read = inStream.read(buffer); read >= 0;
            read = inStream.read(buffer))
            outStream.write(buffer, 0, read);

        inStream.close();
        outStream.close();
        servidortrans.close();
    } catch (IOException e) {
}
}

```

```
// TODO Auto-generated catch block  
e.printStackTrace();  
}  
}  
}
```

#### 7.1.6.5. CopiarFichero()

La función “CopiarFichero” va a recibir del cliente los datos de fichero origen a copiar y ruta de destino. El servidor, ejecutará la orden para copiarlo, según sea el sistema, si fuese Windows ejecutaríamos un “copy” y si fuese Linux ejecutariamos el comando “cp”. El código quedaría como sigue:

```
private void CopiarFichero(){
    try {
        mensaje = (String) entrada.readObject();
        String FicheroOrigen=mensaje;

        mensaje = (String) entrada.readObject();
        String RutaDestino=mensaje;
        areaPantalla.append("\n\t• CopiarFichero:\tFichero Origen:
                           " + FicheroOrigen + "\tRuta Destino: " + RutaDestino);

        if(SistemaOperativo.equals("Windows")){
            p = Runtime.getRuntime().exec ("cmd /c " +"copy /Y \""
                                         + FicheroOrigen + "\" \"\" " + RutaDestino + "\"");
        } else{
            String [] command = {"sh","-c","cp -f " + "\"" +
                FicheroOrigen + "\" \"\" " + RutaDestino + "/\""};
            p = Runtime.getRuntime().exec (command);
        }
        Enviar("Copia realizada con éxito");

    } catch (ClassNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

#### 7.1.6.6. CorteFichero()

La función “CopiarFichero” va a recibir del cliente los datos de fichero origen a cortar y ruta de destino. El servidor, ejecutará la orden para moverlo, según sea el sistema, si fuese Windows ejecutaríamos un “move” y si fuese Linux ejecutaríamos el comando “cp” y posteriormente borraríamos el fichero origen. El código quedaría como sigue:

```
CorteFichero()  
private void CorteFichero(){  
    try {  
        mensaje = (String) entrada.readObject();  
        String FicheroOrigen=mensaje;  
  
        mensaje = (String) entrada.readObject();  
        String RutaDestino=mensaje;  
        areaPantalla.append("\n\t- Cortar Fichero:\\" + Fichero_Origen:  
                           " + FicheroOrigen + "\\Ruta Destino: " + RutaDestino);  
  
        if(SistemaOperativo.equals("Windows")){  
            p = Runtime.getRuntime().exec ("cmd /c " +"move /Y \\\""  
                                         + FicheroOrigen + "\\\" \\\""+ RutaDestino + "\\\"");  
  
        } else{  
            String [] command = {"sh","-c","cp -f " + "\\" +  
                                FicheroOrigen + "\\\" \\\""+ RutaDestino + "\\\""};  
            p = Runtime.getRuntime().exec (command);  
            String [] command1 = {"sh","-c","rm -f " + "\\" +  
                                  FicheroOrigen + "\\\""};  
            p = Runtime.getRuntime().exec (command1);  
        }  
        Enviar("Pegado realizado con éxito");  
    } catch (ClassNotFoundException e) {  
        // TODO Auto-generated catch block  
        e.printStackTrace();  
    } catch (IOException e) {  
        // TODO Auto-generated catch block  
        e.printStackTrace();  
    }  
}
```

#### 7.1.6.7. EliminarFichero()

Esta función va a recibir del cliente los datos referentes a el fichero a eliminar junto a su ruta. El servidor mandará ejecutar la orden “del” en Windows o “rm” en Linux, la cual se encargará de eliminar dicho fichero. El código es el siguiente:

```
    EliminarFichero()  


---

private void EliminarFichero(){  
    try {  
        mensaje = (String) entrada.readObject();  
        String FicheroOrigen=mensaje;  
        areaPantalla.append("\n\t- Eliminar Fichero:\tFichero  
                           Origen: " + FicheroOrigen);  
  
        if(SistemaOperativo.equals("Windows")){  
            p = Runtime.getRuntime().exec ("cmd /c " +"del \\""+  
                                         FicheroOrigen + "\\"");  
        } else{  
            String [] command = {"sh","-c","rm -f " + "\\" +  
                                 FicheroOrigen + "\\"};  
        }  
    }  
}
```

```

        p = Runtime.getRuntime().exec (command);
    }

    Enviar("Fichero " + FicheroOrigen + " eliminado con exito")
    ;
} catch (IOException e1) {
// TODO Auto-generated catch block
e1.printStackTrace();
} catch (ClassNotFoundException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}

```

#### 7.1.6.8. RenombrarFichero()

Esta función va a recibir del cliente los datos de ruta del fichero a renombrar, nombre del fichero a renombrar y nombre nuevo que se le va a dar a dicho fichero. El servidor mandará ejecutar la orden “move” en Windows o “mv” en Linux, la cual se encargará de renombrar dicho fichero, manteniendo la misma ruta. El código es el siguiente:

---

RenombrarFichero()

---

```

private void RenombrarFichero(){
    try {
        mensaje = (String) entrada.readObject();
        String Ruta=mensaje;

        mensaje = (String) entrada.readObject();
        String Fichero=mensaje;

        mensaje = (String) entrada.readObject();
        String NombreNuevo=mensaje;
        areaPantalla.append("\n\t Renombrar Fichero:\tFichero
                           Origen: " + Ruta+Fichero + "\tNombre Nuevo: " +
                           NombreNuevo);

        if(SistemaOperativo.equals("Windows")){
            p = Runtime.getRuntime().exec ("cmd /c " +"move /Y \\
                                         + Ruta+Fichero + "\" \" + Ruta+NombreNuevo + "\\"
                                         );
        }else{
            String [] command = {"sh","-c","mv -f " + "\"" + Ruta+/
                               "+Fichero + "\"" + " " + "\"" + Ruta+/" +NombreNuevo +
                               "\""};
            p = Runtime.getRuntime().exec (command);
        }

        Enviar("Fichero " + Fichero + " renombrado con exito");
    } catch (IOException e1) {
// TODO Auto-generated catch block
e1.printStackTrace();
} catch (ClassNotFoundException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}

```

```

    }
}
```

#### 7.1.6.9. CrearFichero()

La función “CrearFichero” va a recibir del cliente los datos de ruta, nombre del fichero a crear y el contenido del mismo. El código utilizado va a ser el siguiente:

---

CrearFichero()

---

```

private void CrearFichero(){
    try {
        mensaje = (String) entrada.readObject();
        String Ruta=mensaje;

        mensaje = (String) entrada.readObject();
        String Fichero=mensaje;

        mensaje = (String) entrada.readObject();
        String Contenido=mensaje;
        areaPantalla.append("\n\t• CrearFichero \tRuta: " + Ruta +
                           "\tFichero: " + Fichero + "\n\t• Contenido:");

        FileWriter fichero = null;
        PrintWriter pw = null;
        try {
            if(SistemaOperativo.equals("Windows"))
                fichero = new FileWriter(Ruta+Fichero);
            else
                fichero = new FileWriter(Ruta+ "/" + Fichero);

            pw = new PrintWriter(fichero);

            String data[];
            data=Contenido.split("\n");
            for(int i=0;i<data.length;i++){
                areaPantalla.append("\n\t" + data[i]);
                pw.println(data[i]);
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            try {
                // Nuevamente aprovechamos el finally para
                 // asegurarnos que se cierra el fichero.
                if (null != fichero)
                    fichero.close();
            } catch (Exception e2) {
                e2.printStackTrace();
            }
        }
    } catch (ClassNotFoundException | IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
}
```

```
}
```

### 7.1.7. Orden libre

En este apartado veremos la implementación de la función “OrdenLibre()”, la cual simplemente va a recibir una orden del cliente y el servidor, segun su sistema operativo, va a ejecutar esa orden recibida y mandar el resultado de vuelta al cliente.

---

```
OrdenLibre()
```

---

```

private void OrdenLibre(){
    try {
        mensaje = (String) entrada.readObject();

        EjecutarOrdenLibre(mensaje);
    }catch ( IOException excepcionES ) {
        excepcionES.printStackTrace();
    } catch ( ClassNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    Enviar("Fin_OordenLibre");
}

private void EjecutarOrdenLibre(String mensaje){
    areaPantalla.append("\n      -----> Ejecutando orden libre: " +
        mensaje);

    try {
        if(SistemaOperativo.equals("Windows"))
            p = Runtime.getRuntime().exec ("cmd /c " + mensaje);
        else{
            String [] command = {"sh","-c",mensaje};
            p = Runtime.getRuntime().exec (command);
        }
        is = p.getInputStream();

        if(SistemaOperativo.equals("Windows"))
            br = new BufferedReader (new InputStreamReader (is , "Cp850"));
        else
            br = new BufferedReader (new InputStreamReader (is));
        // Se lee la primera linea
        aux = br.readLine();

        // Mientras se haya leido alguna linea
        while (aux!=null) {
            if (!IndicadorPrueba)
                Enviar(aux);
            else
                areaPantalla.append("\n" + aux);
            aux = br.readLine();
        }
    }
}
```

```

        }
    }catch ( IOException excepcionES ) {
        excepcionES.printStackTrace();
    }
}

```

### 7.1.8. Scripts

Esta sección va a tratar sobre las funcionalidades referidas a los Scripts. Cuando se llame a la función Script(), se mandará al cliente el listado de Scripts contenidos en la carpeta correspondiente, dado por la siguiente función:

---

```

ListadoScripts()
_____
private void ListadoScripts(){
    try {
        if( SistemaOperativo.equals( "Windows" )){
            p = Runtime.getRuntime().exec ( "cmd /c " + "dir /B \\""
                + DireccionScripts + "\\" );
        }else{
            String [] command = { "sh" , "-c" , "ls \\" + DireccionScripts + "\\" };
            p = Runtime.getRuntime().exec ( command );
        }

        is = p.getInputStream();

        if( SistemaOperativo.equals( "Windows" ))
            br = new BufferedReader (new InputStreamReader ( is , "Cp850" ));
        else
            br = new BufferedReader (new InputStreamReader ( is ));

        // Se lee la primera linea
        aux = br.readLine();

        // Mientras se haya leido alguna linea
        while (aux!=null) {
            if (!IndicadorPrueba && (aux.endsWith( ".py" ) || aux.endsWith( ".sh" ) || aux.endsWith( ".bat" ) || aux.endsWith( ".run" ) || aux.endsWith( ".bin" ))){
                Enviar(aux);
                areaPantalla.append( "\n\t" + aux );
            }

            aux = br.readLine();
        }
    }catch ( IOException excepcionES ) {
        excepcionES.printStackTrace();
    }
}

```

### 7.1.8.1. EjecutarScript()

La función “EjecutarScript()” recibe como parámetro el nombre del script a ejecutar. Dicha función se encarga de ejecutar la llamada al sistema para dicha ejecución, quedando el código como sigue:

---

```
EjecutarScript()
```

---

```

private void EjecutarScript(String fichero){
    try{
        areaPantalla.append("\n\t EjecutandoScript: " + fichero +
                            "\n");
        if(SistemaOperativo.equals("Windows")){
            if(fichero.endsWith("py")){
                p = Runtime.getRuntime().exec ("cmd /c " + "cd \\" +
                                              DireccionScripts + "\"" && python "\"" +
                                              fichero + "\"");
            }
            if(fichero.endsWith("bat")){
                p = Runtime.getRuntime().exec ("cmd /c " + "cd \\" +
                                              DireccionScripts + "\"" && "\"" + fichero + "\"");
            }
            else{
                p = Runtime.getRuntime().exec ("cmd /c \"\" + "
                                              DireccionScripts + "\\\" + fichero + "\\\"");
            }
        }
        else{
            if(fichero.endsWith("bin") || fichero.endsWith("run") ||
                fichero.endsWith("sh")){
                String [] command = {"sh", "-c", "cd \\" + DireccionScripts + "\"" && chmod 777 " + "\"" + fichero + "\"" && ./\" + fichero + "\""};
                p = Runtime.getRuntime().exec (command);
            }
            if(fichero.endsWith("py")){
                String [] command = {"sh", "-c", "cd \\" + DireccionScripts + "\"" && chmod 777 " + "\"" + fichero + "\"" && python "\"" + fichero + "\""};
                p = Runtime.getRuntime().exec (command);
            }
        }
    }catch (IOException excepcionES) {
        excepcionES.printStackTrace();
    }
}

```

---

### 7.1.8.2. EliminarScript()

Función que recibe como parámetro el script a eliminar, mandando el comando correspondiente para realizarlo, mediante el siguiente código:

---

```
EliminarScript()
```

---

```

private void EliminarScript(String fichero){

```

```

try{
    areaPantalla.append("\n\t EliminandoScript: " + fichero +
    "\n");

    if(SistemaOperativo.equals("Windows"))
        p = Runtime.getRuntime().exec ("cmd /c " +"del \\" +
        DireccionScripts + "\\\" + fichero + "\\\"");
    else{
        String [] command = {"sh", "-c", "rm -f " + "\\" +
        DireccionScripts + "/" + fichero + "\\\""};
        p = Runtime.getRuntime().exec (command);
    }
} catch ( IOException excepcionES ) {
    excepcionES.printStackTrace();
}
}

```

#### 7.1.8.3. TransferirScript()

Función que recibe como parámetro el nombre del script a crear y el contenido del mismo. Esta función utiliza una variable “DireccionScripts” que guarda la ruta de los Scripts, encargándose dicha función de mandar los comandos adecuados para la creación del fichero de la siguiente manera:

---

TransferirScript()

---

```

private void TransferirScript(String nombrefichero, String texto){
    areaPantalla.append("\n\t TransferirScript: " +
        nombrefichero + "\n\t Contenido: \n");

    FileWriter fichero = null;
    PrintWriter pw = null;

    try {
        if(SistemaOperativo.equals("Windows"))
            fichero = new FileWriter(DireccionScripts+"\\\"+
            nombrefichero);
        else
            fichero = new FileWriter(DireccionScripts+"/"+
            nombrefichero);

        pw = new PrintWriter(fichero);

        String data[];
        data=texto.split("\\n");
        for(int i=0;i<data.length; i++){
            areaPantalla.append("\n\t" + data[i]);
            pw.println(data[i]);
        }
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            // Nuevamente aprovechamos el finally para
            // asegurarnos que se cierra el fichero.
        }
    }
}

```

```
        if (null != fichero)
            fichero.close();
    } catch (Exception e2) {
        e2.printStackTrace();
    }
}
```

#### 7.1.8.4. VerScript()

Esta función recibe como parámetro el nombre del Script a leer. Utilizando los flujos correspondientes, vamos a ir leyendo el script e ir mandándolo al cliente para su posterior visualización.

VerScript()

```

private void VerScript(String fichero){
    File archivo = null;
    FileReader fr = null;
    BufferedReader br = null;
    areaPantalla.append("\n\tVerScript: " + fichero +"\n");

    try{
        if(SistemaOperativo.equals("Windows"))
            archivo = new File (DireccionScripts+"\\"+
                fichero);
        else
            archivo = new File (DireccionScripts+"/"+
                fichero);

        fr = new FileReader (archivo);
        br = new BufferedReader(fr);

        // Lectura del fichero
        String linea;
        while((linea=br.readLine())!=null){
            if(!IndicadorPrueba)
                Enviar(linea);
            areaPantalla.append("\t" + linea + "\n");
        }
        if(!IndicadorPrueba)
            Enviar("FinLecturaScript");
    }

} catch ( IOException excepcionES ) {
    excepcionES.printStackTrace();
} finally{
// En el finally cerramos el fichero, para asegurarnos
// que se cierra tanto si todo va bien como si salta
// una excepcion.
try{
    if( null != fr ){
        fr.close();
    }
} catch (Exception e2){
    e2.printStackTrace();
}
}

```

```
    }
}
```

### 7.1.9. Apagar Sistema

Para la dicha funcionalidad, tenemos el código siguiente:

---

```
Apagar()
```

---

```
private void Apagar() {
    areaPantalla.append("\nApagando... \n");
    try {
        if (SistemaOperativo.equals("Windows"))
            p = Runtime.getRuntime().exec ("cmd /c shutdown -s");
        else{
            String [] command = {"sh","-c","halt"};
            p = Runtime.getRuntime().exec (command);
        }
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

### 7.1.10. Reiniciar Sistema

Para la dicha funcionalidad, tenemos el código siguiente:

---

```
Reiniciar()
```

---

```
private void Reiniciar() {
    areaPantalla.append("\nReiniendo... \n");
    try {
        if (SistemaOperativo.equals("Windows"))
            p = Runtime.getRuntime().exec ("cmd /c shutdown -r");
        else{
            String [] command = {"sh","-c","reboot"};
            p = Runtime.getRuntime().exec (command);
        }
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

## 7.2. Integración del sistema

En esta sección vamos a ver qué código se ha utilizado para que el servidor vaya respondiendo a la peticiones que le llegan y cómo se ha integrado todo para según las peticiones del cliente, ejecutar las funcionalidades anteriormente descritas. Comenzamos con una función que es la principal del servidor y es la siguiente:

---

```

EjecutarServidor()
public void EjecutarServidor(int Puerto){
    try{
        servidor= new ServerSocket(Puerto);
        areaPantalla.append( "\n\n      Esperando una conexión en
                           el puerto " + String.valueOf(Puerto) + " . . . \n\n");
    }

    while(true){
        try {
            // Aceptamos la conexión
            conexion = servidor.accept();

            // Informamos de la Conexión recibida desde el
            // terminal
            areaPantalla.append( "\n      Conexión " +
                               " recibida de: " + conexion.getInetAddress()
                               .getHostName() + "\n");

            EstablecerFlujos();

            mensaje = (String) entrada.readObject();
            while(true){
                ProcesarEntrada(mensaje);
                mensaje = (String) entrada.readObject();
            }
        } catch ( EOFException excepcionEOF ) {

            areaPantalla.append(" \n\n      Se ha terminado la
                               conexión ... Reiniciando el servidor ... \n");

        } catch ( ClassNotFoundException e ) {
            areaPantalla.append("\n
                               ClassNotFoundException (EjecutarServidor)"
                               );
        } finally{
            ReiniciarServidor();
        }
    }
} catch ( IOException excepcionES ) {
    //excepcionES.printStackTrace();
}
}

```

Esta función es ejecutada mediante un hilo: “HiloEjecutarServidor” y también se controla mediante las funciones start() y ReiniciarServidor() de la siguiente manera:

```

private void ReiniciarServidor(){
    if(!servidor.isClosed()){
        try {
            servidor.close();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
    start();
}

public void start() {
    HiloEjecutarServidor = new HiloEjecutarServidor();
    HiloEjecutarServidor.start();
}

private class HiloEjecutarServidor extends Thread {
    public void run() {
        EjecutarServidor(Puerto);
    }
}

```

Pasamos a ver las últimas dos funciones para la integración del sistema, las cuales son “EstablecerFlujos()” y “ProcesarEntrada()”. La primera de ellas simplemente obtiene los flujos para la comunicación con el cliente. La segunda va a identificar la petición del cliente y ejecutar la acción que corresponde a esa petición:

---

EstablecerFlujos()

---

```

private void EstablecerFlujos(){
    // establecer flujo de salida para los objetos
    try {
        salida = new ObjectOutputStream( conexion.
            getOutputStream() );
        salida.flush(); // vaciar búffer de salida

        entrada = new ObjectInputStream( conexion.
            getInputStream() );
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

```

---

ProcesarEntrada()

---

```

private void ProcesarEntrada( String orden) throws IOException{

```

```
if(orden.matches("MainMenu")){
    Enviar(SistemaOperativo);

if(orden.matches("TerminarConexion")){
    areaPantalla.append("\n\n    Se ha terminado la
        conexión... Reiniciando el servidor...\n");
    salida.close();
    servidor.close();
}

if(orden.matches("Listado")){
    ListadoProcesos();
}

if(orden.startsWith("taskkill")){
    ProcesarTaskKillWindows(orden);
}

if(orden.matches("Navegacion")){
    ProcesarNavegacion();
}

if(orden.matches("InfDiscos")){
    InformacionDiscos();
}

if(orden.matches("Red")){
    InformacionRed();
}

if(orden.matches("Memoria")){
    InformacionMemoria();
}

if(orden.matches("SistemaOperativo")){
    InformacionSistemaOperativo();
}

if(orden.matches("OrdenLibre")){
    OrdenLibre();
}

if(orden.matches("Reiniciar")){
    Reiniciar();
}

if(orden.matches("Apagar")){
    Apagar();
}
```

```

if(orden.matches("Scripts")){
    Scripts();
}

if(orden.startsWith("EjecutarScript")){
    String fichero = mensaje.replace("EjecutarScript ", "");
    EjecutarScript(fichero);
}

if(orden.startsWith("EliminarScript")){
    String fichero = mensaje.replace("EliminarScript ", "");
    EliminarScript(fichero);
}

if(orden.equals("TransferirScript")){
    try {
        mensaje = (String) entrada.readObject();
        String nombrefichero = mensaje;
        mensaje = (String) entrada.readObject();
        String texto = mensaje;

        TransferirScript(nombrefichero, texto);
    } catch (ClassNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

if(orden.equals("VerScript")){
    try {
        mensaje = (String) entrada.readObject();
        String nombrefichero = mensaje;

        VerScript(nombrefichero);
    } catch (ClassNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

```

Con todo esto ya tenemos todo el sistema integrado, capaz de responder ante peticiones del cliente. El servidor comenzaría ejecutando un hilo “EjecutarServidor” el cual esperaría una conexión entrante. Una vez recibida, el servidor irá procesando las entradas que le llegen desde el cliente y ejecutando las funciones solicitadas desde el mismo.

### 7.3. Interfaz gráfica utilizando JavaSwing

La interfaz gráfica del servidor la podemos ver en la siguiente figura:

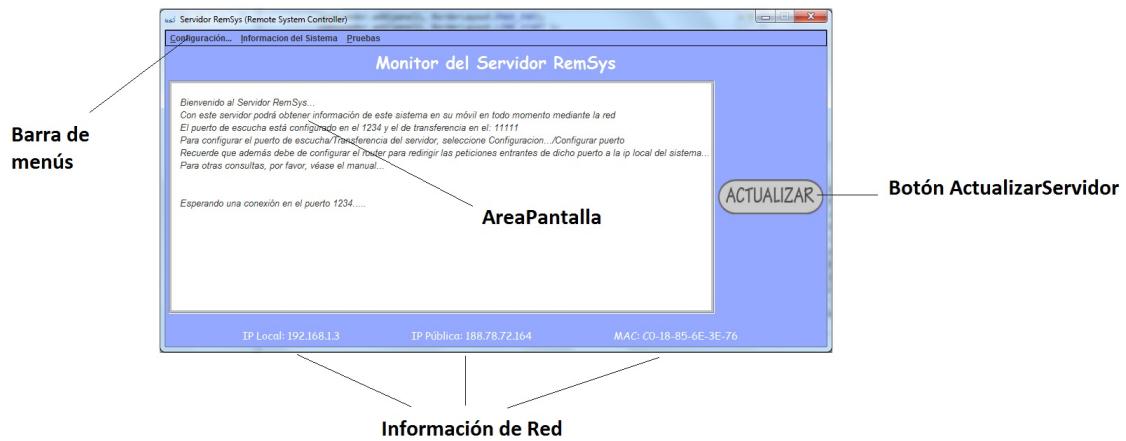


Figura 7.14: Interfaz Gráfica RemSys

Como podemos observar la interfaz gráfica se compone principalmente de:

- Barra de menús: La cual nos permitirá acceder a la ventana de configuración de los puertos, tanto de escucha como de transferencia, la ventana de información del sistema y las diversas pruebas que posee el servidor.
- Información de Red: Información que obtenemos a simple vista la cual es indispensable para el cliente poder acceder al servidor. Obtenemos de un vistazo la información de IP local, IP pública y la MAC. Toda esta información ampliada se puede obtener con el acceso al menú Información del sistema de la barra de menús, pudiendo modificar dicha información.
- AreaPantalla: Está compuesto por un JTextArea, el cual nos servirá como una especie de registro de todas las acciones y peticiones que llegan al servidor o que el servidor realiza.
- Botón ActualizarServidor: Solución para estados inconsistentes del sistema, pérdida de conexión o cualquier otro problema. Lo que hace es, con los nuevos datos, lanzar un nuevo hilo de ejecución del servidor, relanzándose el proceso de obtención de los datos de red.

Dicha interfaz está construida mediante código, el cual es el siguiente:

---

Constructor del Servidor

---

```

public class Servidor extends JFrame {
    private static final long serialVersionUID = 1L;
    private JTextArea areaPantalla;
    private JButton Activar_Desactivar;
    private JTextField direccionIP;
    private JTextField mac;
    private InetAddress addr;
    private JTextField direccionlocal;
    private MacAddress dirmac;
    private ServerSocket servidor;
    private Socket conexion;
    private ServerSocket servidortrans;
    private Socket conexiontrans;
    private ObjectOutputStream salida;
    private ObjectInputStream entrada;
    private String mensaje;
    private Process p ;
    private InputStream is ;
    private BufferedReader br;
    private String aux;
    private String nombreScript;
    private String DireccionScripts;
    private String SistemaOperativo;
    private boolean IndicadorPrueba;
    private volatile Thread HiloEjecutarServidor;
    private int Puerto =1234;
    private int PuertoTransferencia=1111;
    int cont=0;
    String excepcion="";
    // Fuentes
    Font boldUnderline = new Font("Comic Sans MS",Font.
        ROMAN_BASELINE, 15);
    Font FuenteAreaPantalla = new Font("Console",Font.
        HANGING_BASELINE, 13);
    Font FuenteTitulo = new Font("Comic Sans MS",Font.BOLD,
        24);
    Font FuenteTituloSubventana = new Font("Comic Sans MS",
        Font.BOLD, 18);

    // Colores
    Color color = new Color(149,168,255);

    // Frame Servidor
    public Servidor(){
        super( "Servidor RemSys (Remote System Controller)");
        // Establecemos el Sistema Operativo
    }
}
```

```
if( System.getProperty("os.name").startsWith("Windows"))
    SistemaOperativo = "Windows";
else
    SistemaOperativo = "Linux";

if( SistemaOperativo.equals("Windows"))
    setSize(950, 480);
else
    setSize(1085, 480);

// Establecemos el icono del programa Servidor
ImageIcon icon = new ImageIcon("./\Imagenes\remsyslogo.
png");
setIconImage(icon.getImage().getScaledInstance( 190, 130
, java.awt.Image.SCALEDEFAULT ));

setBackground( Color.CYAN );

// Establecemos la posicion de la ventana principal del
Servidor
Dimension screenSize = Toolkit.getDefaultToolkit().
getScreenSize();
Point middle = new Point(screenSize.width / 2,
screenSize.height / 2);
Point newLocation = new Point(middle.x - (getWidth() /
2), middle.y - (getHeight() / 2));
 setLocation(newLocation);

JTextField Titulo = new JTextField("Monitor del Servidor
RemSys");

areaPantalla = new JTextArea(18,70);
areaPantalla.setFont(FuenteAreaPantalla);
areaPantalla.setBorder(BorderFactory.createLineBorder(
Color.lightGray, 3));

DefaultCaret caret = (DefaultCaret)areaPantalla.getCaret
();
caret.setUpdatePolicy(DefaultCaret.ALWAYSUPDATE);

HiloEjecutarServidor = this.new HiloEjecutarServidor();

JPanel panel1 = new JPanel();
JPanel panel2 = new JPanel();
JPanel panel3 = new JPanel();
Container contenedor = getContentPane();
```

```
panel3.add(Titulo);

panel2.setBackground(color);
panel3.setBackground(color);
panel1.setBackground(color);

ImageIcon cup = new ImageIcon(this.getClass().getResource("btnactualizar.png"));

Image img = cup.getImage();
Image newimg = img.getScaledInstance(150, 50, java.awt.Image.SCALESMOOTH);
cup = new ImageIcon(newimg);

ManejadorBotones manejador = new ManejadorBotones();

Activar_Desactivar = new JButton(cup);
Activar_Desactivar.setPreferredSize(new Dimension(150, 50));
Activar_Desactivar.addActionListener(manejador);
Activar_Desactivar.setBorderPainted(false);
Activar_Desactivar.setContentAreaFilled(false);

direccionlocal = new JTextField(16);
direccionlocal.setEditable(false);
try {
    addr = InetAddress.getLocalHost();
    direccionlocal.setText("IP Local: " + addr.getHostName().trim());
} catch (UnknownHostException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

dirmac= new MacAddress();
mac = new JTextField(" MAC: ",18);
mac.setEditable(false);
mac.setFont(boldUnderline);
mac.setBackground(color);
mac.setForeground(Color.WHITE);
mac.setBorder(null);
mac.setEditable(false);

try{
```

```
mac.setText(" MAC: " + dirmac.MaC.toUpperCase().trim());  
  
}catch(NullPointerException e){  
    // No se tiene conexión y no se accede a InetAddress  
    // .getLocalHost() por lo tanto no obtiene la MAC.  
}  
  
direccionIP = new JTextField("IP Pública: ",23);  
direccionIP.setFont(boldUnderline);  
direccionIP.setBackground(color);  
direccionIP.setForeground(Color.WHITE);  
direccionIP.setBorder(null);  
direccionIP.setEditable(false);  
  
try{  
    GetIP ip = new GetIP();  
    direccionIP.setText(direccionIP.getText().toString()  
        concat(ip.IPaddress.toString().trim()));  
  
}catch(NullPointerException e){  
}  
  
Titulo.setFont(FuenteTitulo);  
Titulo.setBackground(color);  
Titulo.setForeground(Color.WHITE);  
Titulo.setBorder(null);  
Titulo.setEditable(false);  
  
direccionlocal.setFont(boldUnderline);  
direccionlocal.setBackground(color);  
direccionlocal.setForeground(Color.WHITE);  
direccionlocal.setBorder(null);  
direccionlocal.setEditable(false);  
  
panel2.add(new JScrollPane(areaPantalla));  
panel2.add(Activar_Desactivar);  
  
panel1.add(direccionlocal);  
JTextField Espacio1 = new JTextField(5);  
Espacio1.setBackground(color);  
Espacio1.setBorder(null);  
Espacio1.setEditable(false);  
  
panel1.add(Espacio1);  
try{  
    panel1.add(direccionIP);  
}catch(NullPointerException e){
```

```

        }
JTextField Espacio2 = new JTextField(2);
Espacio2.setBackground(color);
Espacio2.setBorder(null);
Espacio2.setEditable(false);
panel1.add(Espacio2);

try{
    panel1.add(mac);
}catch(NullPointerException e){

}

CrearMenus();

contenedor.add(panel1, BorderLayout.PAGE_END);
contenedor.add(panel2, BorderLayout.LINE_START );
contenedor.add(panel3, BorderLayout.BEFORE_FIRST_LINE );

setVisible( true );
setResizable(false);
areaPantalla.setEditable(false);

}

```

Como podemos observar los principales componentes gráficos del servidor están compuestos por JTextArea, JButton, JTextField, JPanel y los contenedores correspondientes para tener todos estos componentes. Además la función “CrearMenus()” es la encargada de, como su propio nombre indica, crear los menús mediante los componentes JMenu correspondientes, quedando su código como sigue:

#### Creación de menús

---

```

private void CrearMenus(){
    // Creación de menús
    JMenu menuConfigurarPuerto = new JMenu( "Configuración . . ." );
    menuConfigurarPuerto.setMnemonic( 'C' );

    // establecer elemento de menú Propiedades . . .
    JMenuItem ConfigurarPuerto = new JMenuItem( "Configurar
        puerto . . ." );
    ConfigurarPuerto.setMnemonic( 'P' );
    menuConfigurarPuerto.add( ConfigurarPuerto );

    JMenuItem LimpiarPantalla = new JMenuItem( "Limpiar
        pantalla" );
    LimpiarPantalla.setMnemonic( 'L' );
    menuConfigurarPuerto.add( LimpiarPantalla );

    ConfigurarPuerto.addActionListener( new ActionListener() {

```

```

public void actionPerformed( ActionEvent evento ) {
    // Ventana Propiedades
    CrearVentanaConfigurarPuerto();
}

}); // fin de la llamada a addActionListener

LimpiarPantalla.addActionListener( new ActionListener() {
    public void actionPerformed( ActionEvent evento ) {
        // Ventana Propiedades
        areaPantalla.setText("");
    }
});

}); // fin de la llamada a addActionListener

JMenu menuPruebas = new JMenu( "Pruebas" );
menuPruebas.setMnemonic( 'P' );

// establecer elemento de menú Propiedades...
JMenuItem PruebaDiscos = new JMenuItem( "Discos" );
PruebaDiscos.setMnemonic( 'P' );
menuPruebas.add( PruebaDiscos );

PruebaDiscos.addActionListener( new ActionListener() {
    public void actionPerformed( ActionEvent evento ) {
        // Ventana Propiedades
        IndicadorPrueba = true;
        InformacionDiscos();
        ReiniciarServidor();
    }
});

}); // fin de la llamada a addActionListener

JMenuItem PruebaRed = new JMenuItem( "Red" );
PruebaRed.setMnemonic( 'R' );
menuPruebas.add( PruebaRed );

PruebaRed.addActionListener( new ActionListener() {
    public void actionPerformed( ActionEvent evento ) {
        // Ventana Propiedades
        IndicadorPrueba = true;
        InformacionRed();
        ReiniciarServidor();
    }
});

}); // fin de la llamada a addActionListener

JMenuItem PruebaSistemaOperativo = new JMenuItem( "SistemaOperativo" );
PruebaSistemaOperativo.setMnemonic( 'S' );
menuPruebas.add( PruebaSistemaOperativo );

PruebaSistemaOperativo.addActionListener( new
    ActionListener() {
        public void actionPerformed( ActionEvent evento ) {
            // Ventana Propiedades

```

```

    IndicadorPrueba = true;
    InformacionSistemaOperativo();
    ReiniciarServidor();
}

}); // fin de la llamada a addActionListener

JMenuItem PruebaMemoria = new JMenuItem( "Memoria" );
PruebaMemoria.setMnemonic( 'P' );
menuPruebas.add( PruebaMemoria );

PruebaMemoria.addActionListener( new ActionListener() {
    public void actionPerformed( ActionEvent evento ) {
        // Ventana Propiedades
        IndicadorPrueba = true;
        InformacionMemoria();
        ReiniciarServidor();
    }
}); // fin de la llamada a addActionListener

JMenuItem PruebaProcesos = new JMenuItem( "Procesos" );
PruebaProcesos.setMnemonic( 'P' );
menuPruebas.add( PruebaProcesos );

PruebaProcesos.addActionListener( new ActionListener() {
    public void actionPerformed( ActionEvent evento ) {
        // Ventana Propiedades
        IndicadorPrueba = true;
        ListadoProcesos();
        ReiniciarServidor();
    }
}); // fin de la llamada a addActionListener

JMenuItem PruebaOrdenLibre = new JMenuItem( "Orden Libre" );
;
PruebaOrdenLibre.setMnemonic( 'O' );
menuPruebas.add( PruebaOrdenLibre );

PruebaOrdenLibre.addActionListener( new ActionListener() {
    public void actionPerformed( ActionEvent evento ) {
        // Ventana Propiedades
        IndicadorPrueba = true;
        CrearVentanaOrdenLibre();
    }
}); // fin de la llamada a addActionListener

final JMenu PruebaScripts = new JMenu( "Scripts" );
PruebaScripts.setMnemonic( 'S' );
menuPruebas.add( PruebaScripts );

JMenuItem PruebaVerScripts = new JMenuItem( "Ver Script" );

```

```

PruebaVerScripts.setMnemonic( 'S' );
PruebaScripts.add( PruebaVerScripts );

PruebaVerScripts.addActionListener( new ActionListener() {
    public void actionPerformed( ActionEvent evento ) {
        // Ventana Propiedades
        IndicadorPrueba = true;
        CrearVentanaScripts(1);
    }
}); // fin de la llamada a addActionListener

JMenuItem PruebaEjecutarScripts = new JMenuItem( "Ejecutar Script" );
PruebaEjecutarScripts.setMnemonic( 'S' );
PruebaScripts.add( PruebaEjecutarScripts );

PruebaEjecutarScripts.addActionListener( new ActionListener()
() {
    public void actionPerformed( ActionEvent evento ) {
        // Ventana Propiedades
        IndicadorPrueba = true;
        CrearVentanaScripts(2);
    }
}); // fin de la llamada a addActionListener

JMenu InformacionSistema = new JMenu( "Informacion del Sistema" );
InformacionSistema.setMnemonic( 'I' );

// establecer elemento de menú Propiedades...
JMenuItem Informacion = new JMenuItem( "Direcciones..." );
Informacion.setMnemonic( 'f' );
InformacionSistema.add( Informacion );

Informacion.addActionListener( new ActionListener() {
    public void actionPerformed( ActionEvent evento ) {
        // Ventana Propiedades
        CrearVentanaInformacionSistema();
    }
}); // fin de la llamada a addActionListener

JMenuBar BarraMenus = new JMenuBar();
setJMenuBar( BarraMenus );
BarraMenus.add( menuConfigurarPuerto );
BarraMenus.add( InformacionSistema );
BarraMenus.add( menuPruebas );
BarraMenus.setBackground( color );
BarraMenus.setBorder( BorderFactory.createLineBorder( Color.BLACK ) );
}

```

Al pulsar en cualquiera de las opciones del menú, se realizará alguna acción o en su mayor parte se nos llevará a una ventana de configuración o información

del sistema.

## **Capítulo 8**

# **Implementación del cliente usando Android**

En este capítulo nos centraremos en describir cómo es y cómo se ha implementado el programa cliente, correspondiente a un dispositivo con sistema Android. En una primera parte veremos la interfaz gráfica que se ha utilizado para que sea amigable con el usuario y bien vista estéticamente. A continuación veremos todas las pantallas a las que podemos acceder en el cliente, y las órdenes que manda al servidor según esta navegación. Por otra parte visualizaremos el código utilizado para tanto, mandar la orden para ejecutar alguna acción en el servidor, como su procesamiento posterior al obtener el resultado del mismo.

### **8.1. Interfaz gráfica**

Para la interfaz gráfica del cliente me he basado en la versión 4.0 (Ice Cream Sandwich) de Android y he intentado seguir un esquema de colores parecidos. El fondo elegido para todas las pantallas es el negro, con botones azules y letras en blanco. Podemos ver un ejemplo en la siguiente imagen:

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID162



Figura 8.1: Interfaz cliente

Esta es la primera pantalla que nos encontramos al iniciar la aplicación del cliente. El cliente tendrá primero que crear un usuario, que se guardará en una base de datos en el dispositivo móvil. Estos datos servirán para crear conexiones a host remotos donde se ejecutará la versión servidor de Remsys. Se ha realizado así por seguridad, ya que si, por ejemplo, se pierde el dispositivo móvil, se tendría la información de ips, puerto abierto y mac del host servidor.

Procedemos ahora a ver todas las pantallas que podemos ver en el dispositivo móvil. Éstas pantallas reciben el nombre de Layouts, los cuales se describen mediante ficheros .XML. Así pues, esta primera pantalla “Login” tendrá por código el siguiente:

Login.xml

---

```
<?xml version="1.0" encoding="utf-8"?>
```

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID163

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/
    android"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:layout_gravity="center"
        android:gravity="center_horizontal">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@drawable/fondotitulo"
        android:orientation="vertical" >

        <TextView
            android:id="@+id/textView1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="10dp"
            android:layout_marginLeft="20dp"
            android:layout_marginTop="10dp"
            android:text="Login"
            android:textAppearance="?android:attr/
                textAppearanceLarge" android:textSize="16dp"
            android:textStyle="bold"/>
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginLeft="20dp"
        android:layout_marginRight="20dp"
        android:layout_marginTop="60dp"
        android:orientation="vertical" >

        <TextView
            android:id="@+id/textUsuario"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginBottom="5dp"
            android:text="Usuario"
            android:textColor="#FFFFFF"
            android:textSize="16dp" />

        <EditText
            android:id="@+id/editTextUsuario"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:ems="10" />

        <TextView
```

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID164

```
        android:id="@+id/textContrasena"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginBottom="5dp"
        android:layout_marginTop="20dp"
        android:text="Contraseña"
        android:textColor="#FFFFFF"
        android:textSize="16dp" />

    <EditText
        android:id="@+id/editTextContrasena"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:ems="10"
        android:inputType="textPassword" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center"
    android:orientation="vertical" android:gravity="center">

    <ScrollView
        android:id="@+id/scrollView2"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:fillViewport="true" >

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:gravity="center">

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_margin="20dp"
                android:gravity="center"
                android:orientation="vertical" >

                <Button
                    android:id="@+id/BotonEntrar"
                    android:layout_width="130dp"
                    android:layout_height="46dp"
                    android:background="@drawable/botonentrar" />

                <ImageButton
```

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID165

```
        android:id="@+id/BotonAgregar"
        android:layout_width="146dp"
        android:layout_height="43dp"
        android:layout_marginTop="20dp"
        android:background="@drawable/botonagregar"
    />

    </LinearLayout>
</LinearLayout>
</ScrollView>
</LinearLayout>
</LinearLayout>
```

Cada componente visual tendrá una propiedad id, el cual servirá para localizar y referenciarlo en el fichero de comportamiento .java asociado a ese .xml. Por ejemplo, el layout “login.xml” tiene asociado el comportamiento dado por el código del fichero “Login.java” dado por el siguiente código:

### Login.java

---

```
package garciaponce.miguel;

import android.app.Activity;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.util.Log;
import android.view.KeyEvent;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.Toast;

public class Login extends Activity {
    private EditText Usuario;
    private EditText Contrasena;
    private ImageButton AgregarUsuario;
    private Button Entrar;
    private String NombreBD = "BDRemSys";
    private String NombreTabla = "Usuarios";
    private UsuariosSQL usdbh;
    private SQLiteDatabase db;
    private Cursor c;
    private Toast toast;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.login);

        // Obtenemos los componentes
        AgregarUsuario = (ImageButton) findViewById(R.id.BotonAgregar);
```

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID166

```
Entrar = (Button) findViewById(R.id.BotonEntrar);

// Si pulsamos sobre el botón agregar
AregarUsuario.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        Intent intent = new Intent(Login.this ,
            CrearUsuario.class);
        startActivity(intent);
    }
});

// Si pulsamos sobre el botón entrar
Entrar.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        Usuario = (EditText) findViewById(R.id.
            editTextUsuario);
        Contrasena = (EditText) findViewById(R.id.
            editTextContrasena);

        // Comprobamos si el usuario y contraseña se
encuentra en la base de datos.
if(ComprobarUsuario(Usuario.getText().toString().
    trim(),Contrasena.getText().toString().trim()))
{
    // Iniciamos el servicio para el control de
la sesión (Nombreusuario)
Intent sesion = new Intent(Login.this ,
    SesionLocal.class);
    if(startService(sesion)==null){
        Log.i("SesionLocal","No se ha podido
            iniciar el servicio");
    } else {
        Log.i("SesionLocal","Servicio iniciado
            correctamente");
    }
    SesionLocal.NombreUsuario=Usuario.getText().
        toString().trim();

    // Una vez comprobada las credenciales, pasamos
a la ListaHosts
Intent intent = new Intent(Login.this ,
    ListaHosts.class);
    startActivity(intent);
} else{
    // Ha habido algun error en la comprobación
de usuarios
toast = Toast.makeText(
    getApplicationContext(),"Credenciales
        incorrectas", Toast.LENGTHSHORT);
    toast.show();
}
});
}

private boolean ComprobarUsuario(String usu,String contra){
```

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID167

```

String Ctr = null;
// Accedemos a la BD
usdbh = new UsuariosSQL(Login.this, NombreBD, null, 1);
db = usdbh.getWritableDatabase();
// Si no existe la creamos
db.execSQL("CREATE TABLE IF NOT EXISTS Usuarios (nombre
        TEXT, contrasena TEXT , PRIMARY KEY(nombre))");
// Ejecutamos la sentencia SQL
c = db.rawQuery(" SELECT contrasena FROM " + NombreTabla +
        " WHERE nombre=\\'" + usu + "\\'", null);

//Nos aseguramos de que existe al menos un registro
if(c.getCount() == 0){
    return false;
} else{
    if (c.moveToFirst()) {
        //Recorremos el cursor hasta que no haya
        //más registros
        do {
            Ctr = c.getString(0);
            Log.i("Comprobar Usuario ", "Ctr: " + Ctr +
                    "contra: " + contra);
        } while(c.moveToNext());
    }

    db.close();

    // Si hemos encontrado alguna coincidencia (existe
    // usuario), devolvemos true
    if(Ctr.equals(contra)){
        return true;
    }
}
return false;
}

@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    if ((keyCode == KeyEvent.KEYCODE_BACK)) {
        moveTaskToBack(true);
        return true;
    }
    return super.onKeyDown(keyCode, event);
}
}

```

podemos observar que donde se produce la asociación entre el Layout y el código java donde se describe su comportamiento es la siguiente línea:

```
setContentView(R.layout.login);
```

además podemos observar el procesamiento para comprobar usuario y contraseña para acceder al sistema, comunicándose con la base de datos local y el método para referenciar objetos mediante su id (findViewById).

Hemos visto la primera pantalla (login.xml) y su funcionamiento es trivial. A partir de aquí vamos a visualizar los distintos layouts y expondré el código que utilizan dichos layouts para obtener los diversos datos que se procesarán para presentar al cliente.

### 8.1.1. Crear Usuario

El layout será el siguiente:



Figura 8.2: Layout “creausuario.xml”

estándo su comportamiento descrito por:

Login.java

---

```
package garciaponce.miguel;
```

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID169

```
import android.app.Activity;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class CrearUsuario extends Activity {
    private EditText Usuario;
    private EditText Contrasena;
    private EditText RepContrasena;
    private Button CrearUsu;
    private String NombreBD = "BDRemSys";
    private String NombreTabla = "Usuarios";
    private UsuariosSQL usdbh;
    private SQLiteDatabase db;
    private Cursor c;
    private Toast toast;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.creausuario);

        CrearUsu = (Button) findViewById(R.id.BotonCreaUsu);

        // Cuando le clickeemos en el boton crearUsuario
        CrearUsu.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                Usuario = (EditText) findViewById(R.id.editTextUsu)
                ;
                Contrasena = (EditText) findViewById(R.id.
                    editTextCont);
                RepContrasena = (EditText) findViewById(R.id.
                    editTextRepCont);

                // Comprobamos que el usuario no existe.
                usdbh = new UsuariosSQL(CrearUsuario.this, NombreBD
                    , null, 1);
                db = usdbh.getWritableDatabase();
                db.execSQL("CREATE TABLE IF NOT
                    EXISTS Usuarios (nombre TEXT,
                    contrasena TEXT , PRIMARY KEY(
                    nombre))");

                c = db.rawQuery(" SELECT nombre,
                    contrasena FROM Usuarios WHERE
                    nombre='" + Usuario.getText() .
```

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID170

```
        toString() + "\'", null);

switch(c.getCount()){
    case 1:
        Log.i("Fila BD ", "Ya existe
                usuario");
        toast = Toast.makeText(
                getApplicationContext(), "El
                usuario ya existe.", Toast.
                LENGTHSHORT);
        toast.show();
        break;
    case 0:
        Log.i("Fila BD ", "No existe
                usuario");
        if(Contrasena.getText().toString().
                equals(RepContrasena.getText().
                toString())){
            //Si hemos abierto
            //correctamente la base
            //de datos
            if(db != null)
            {
                db.execSQL("INSERT INTO " +
                        NombreTabla + "(nombre,
                        contrasena) VALUES ('" +
                        Usuario.getText().toString().
                        trim() + "','" + Contrasena.
                        getText().toString().trim() + "'");
            }
            toast = Toast.makeText(
                    getApplicationContext(), "Usuario
                    creado con éxito.", Toast.
                    LENGTHSHORT);
            toast.show();
        }
        Intent intent = new Intent(CrearUsuario
                .this , Login.class);
        startActivity(intent);
    }else{
        toast = Toast.makeText(
                getApplicationContext(), "Las
                contraseñas no coinciden",
                Toast.LENGTHSHORT);
        toast.show();
    }
    break;
default:
    Log.i("Fila BD ", "Usuario
                duplicado");
    break;
}
db.close();
```

```
    }  
});  
}  
}
```

El comportamiento de este layout es el de crear el usuario con la contraseña que el cliente ha introducido, comprobando si existe el usuario que se quiere crear previamente. El cliente solamente tendrá que llenar los campos “Nombre Usuario”, “Contraseña” y “Repita la contraseña” y pulsar el botón correspondiente para poner en marcha el procedimiento para la creación del usuario.

### 8.1.2. Hosts

El layout será el siguiente:



Figura 8.3: Layout “listahosts.xml”

estando su comportamiento descrito por:

ListaHosts.java

---

```

package garciaponce.miguel;

import java.util.Vector;

import android.app.Activity;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.util.Log;
import android.view.ContextMenu;
import android.view.ContextMenu.ContextMenuItemInfo;
import android.view.KeyEvent;
import android.view.LayoutInflater;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemLongClickListener;
import android.widget.ArrayAdapter;
import android.widget.ImageButton;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

public class ListaHosts extends Activity {

    private ListView ListaHosts;
    private Vector<Host> Hosts;
    private ImageButton BtnAgregar;
    private int pos;
    private String NombreBD = "BDRemSys";
    private String NombreTabla = "Hosts";
    private UsuariosSQL usdbh;
    private SQLiteDatabase db;
    private Cursor c;
    private Toast toast;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.listahosts);

        // Creamos el Vector de Hosts y obtenemos los componentes
        // del Layout
        Hosts = new Vector<Host>();
        ListaHosts = (ListView) findViewById(R.id.ListaHosts);
        BtnAgregar = (ImageButton) findViewById(R.id.
            BotonAgregarHost);
    }
}

```

```

// Acceder a los datos de la base de datos para
almacenarlos en el vector Hosts
usdbh = new UsuariosSQL(ListaHosts.this, NombreBD, null,
1);
db = usdbh.getWritableDatabase();

// Consulta para saber los host que pertenecen al usuario
ya logeado
// Creamos la table si no existe.
db.execSQL("CREATE TABLE IF NOT EXISTS Hosts (usuario TEXT
, nombrehost TEXT, dirip TEXT, puerto TEXT, dirmac TEXT,
PRIMARY KEY(usuario ,nombrehost ,dirip ,puerto))");
c = db.rawQuery("SELECT * FROM " + NombreTabla + " WHERE
usuario=\''" + SesionLocal.NombreUsuario + "\'', null);

if (c.moveToFirst()) {
    //Recorremos el cursor hasta que no haya más registros
    do {
        Host host = new Host(c.getString(1) ,c.getString
(2) ,c.getString(3) ,c.getString(4) );
        Hosts.add(host);
        Log.i("ListaHosts", c.getString(0) + " " +c.
getString(1) + " " +c.getString(2) + " " +c.
getString(3) + " " +c.getString(4) );
    } while(c.moveToNext());
}

// Creamos el Adaptador
AdaptadorHosts adaptador = new AdaptadorHosts(this);
ListaHosts.setAdapter(adaptador);
// Registraremos el menu Contextual
registerForContextMenu(ListaHosts);

ListaHosts.setOnItemLongClickListener(new
OnItemLongClickListener() {
    public boolean onItemLongClick(AdapterView<?>
parent, final View v, int position, long id)
    {
        // Cuando hacemos una pulsación larga,
guardamos la posicion.
pos=position;
        return false;
    }
});
}

// Si le damos al boton agregar, cambiamos de Layout para
crear un nuevo host.
BtnAgregar.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        Intent intent = new Intent(ListaHosts.this ,
CrearHost.class);
        startActivity(intent);
    }
});
}

```

```

    }

    @Override
    public void onCreateContextMenu(ContextMenu menu, View v,
        ContextMenuInfo menuInfo)
    {
        super.onCreateContextMenu(menu, v, menuInfo);

        MenuInflater inflater = getMenuInflater();

        if(v.getId() == R.id.ListaHosts)
        {
            inflater.inflate(R.menu.menuhosts, menu);
        }
    }

    @Override
    public boolean onContextItemSelected(MenuItem item) {
        // Menu contextual, Conectar, Eliminar y Encender.
        switch (item.getItemId()) {
            case R.id.menuhost1:
                Conectar();
                return true;
            case R.id.menuhost2:
                // Eliminar Host de la lista de
                // hosts de la Base de Datos y
                // volver a recargar el layout
                db.execSQL("DELETE FROM " +
                    NombreTabla + " WHERE usuario
                    ='\" + SesionLocal.
                    NombreUsuario + '\"' AND
                    nombrehost='\" + Hosts.get(pos)
                    .getNom() + '\"');");
                db.close();
                // Recargamos el Layout
                Intent intent = new Intent(
                    ListaHosts.this, ListaHosts.
                    class);
                startActivity(intent);
                return true;
            case R.id.menuhost3:
                // Si encendemos el sistema,
                // pasamos al siguiente intent los
                // datos necesarios con IP y MAC.
                Intent intent1 = new Intent(
                    ListaHosts.this, WOL.class);
                intent1.putExtra("DirIP", Hosts.get
                    (pos).getIP());
                intent1.putExtra("DirMac", Hosts.
                    get(pos).getMac());
                startActivity(intent1);
                return true;
            default:
                return super.onContextItemSelected(
                    item);
        }
    }
}

```

```

    }

class AdaptadorHosts extends ArrayAdapter {
    Activity context;

    AdaptadorHosts(Activity context) {
        super(context, R.layout.host, Hosts);
        this.context = context;
    }

    public View getView(int position, View convertView,
                        ViewGroup parent)
    {
        View item = convertView;
        ViewHolder holder;

        if(item == null)
        {
            LayoutInflater inflater = context.
                getLayoutInflater();
            item = inflater.inflate(R.layout.host, null
                );
            holder = new ViewHolder();
            holder.DatoHost1 = (TextView)item.
                findViewById(R.id.NombreHost);
            holder.DatoHost2 = (TextView)item.
                findViewById(R.id.IPHost);
            holder.DatoHost3 = (TextView)item.
                findViewById(R.id.PuertoHost);
            holder.DatoHost4 = (TextView)item.
                findViewById(R.id.MacHost);

            item.setTag(holder);
        }
        else
        {
            holder = (ViewHolder)item.getTag();
        }

        holder.DatoHost1.setText(Hosts.elementAt(
            position).getNom());
        holder.DatoHost2.setText(Hosts.elementAt(
            position).getIP());
        holder.DatoHost3.setText(Hosts.elementAt(
            position).getPuerto());
        holder.DatoHost4.setText(Hosts.elementAt(
            position).getMac());
    }

    return(item);
}

static class ViewHolder {
    public TextView DatoHost1;
    public TextView DatoHost2;
}

```

```

        public TextView DatoHost3;
        public TextView DatoHost4;
    }

    private void Conectar(){
        toast = Toast.makeText(getApplicationContext(),"Conectando
            al host " + Hosts.get(pos).getIP(), Toast.LENGTH_LONG);
        toast.show();

        // Establecemos los parametros del Servicio Conexión
        Conexion.establecerIP(Hosts.get(pos).getIP());
        Conexion.establecerSocket(Integer.parseInt(Hosts.get(pos).
            getPuerto()));

        // Iniciamos el servicio Conexión
        Intent servicio = new Intent(ListaHosts.this, Conexion.
            class);
        if(startService(servicio)==null){
            Log.i("Conectar","No se ha podido iniciar el
                servicio Conexion");
        } else {
            Log.i("Conectar","Servicio Conexion iniciado
                correctamente");
        }

        // Cambiamos el Layaout al menú principal, una vez
        // hemos iniciado el servicio Conexion.
        Intent intent = new Intent(ListaHosts.this ,
            Mainmenu.class);
        startActivity(intent);
    }

    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event) {
        if ((keyCode == KeyEvent.KEYCODE_BACK)) {
            Log.i("ListaHosts","Vuelve a login");
            Intent intent = new Intent(ListaHosts.this , Login.
                class);
            startActivity(intent);
        }
        return super.onKeyDown(keyCode, event);
    }
}

```

Lo que se encarga esta parte del cliente es ver en la base de datos, los host que tiene asociados el usuario logeado, presentándolos al cliente. El cliente realizará una pulsación larga sobre alguno de ellos, dando como resultado un menu contextual con las opciones: “Conectar”, “Eliminar” y “Encender”. Como podemos observar, si pulsamos sobre “Conectar”, se iniciará el servicio de conexión el cual está descrito por un servicio Android con código:

Conexion.java

---

```

package garciaponce.miguel;

import java.io.IOException;

```

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID177

```
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;

import android.app.Activity;
import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.util.Log;

public class Conexion extends Service
{
    public static Activity ACTIVIDAD;
    public static Socket conexion=null;
    public static ObjectOutputStream salida;
    public static ObjectInputStream entrada;
    private static String IP;
    private static int sckt;
    private static boolean conectado=false;

    public static void establecerIP(String ip){
        IP=ip;
    }

    public static String ObtenerIP(){
        return IP;
    }

    public static boolean ObtenerEstado(){
        return conectado;
    }

    public static int ObtenerPuerto(){
        return sckt;
    }
    public static void establecerSocket(int socket){
        sckt=socket;
    }

    public static void establecerActividadPrincipal(Activity actividad)
    {
        Conexion.ACTIVIDAD=actividad;
    }

    public void onCreate()
    {
        super.onCreate();

        // Iniciamos el servicio
        Conexion.iniciarServicio();

        Log.i(getClass().getSimpleName(), "Servicio iniciado");
    }
}
```

```

}

public void onDestroy()
{
    super.onDestroy();

    // Detenemos el servicio
    Conexion.finalizarServicio();

    Log.i(getClass().getSimpleName(), "Servicio detenido");
}

public IBinder onBind(Intent intent)
{
    // No usado de momento, sólo se usa si se va a utilizar IPC
    // (Inter-Process Communication) para comunicarse entre
    procesos
    return null;
}

public static void iniciarServicio()
{
    try
    {
        // Conectamos y obtenemos flujos.
        conexion = new Socket(IP, sckt);
        conectado=true;
        ObtenerFlujos();
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}

public static void ObtenerFlujos(){
    try {
        salida = new ObjectOutputStream(conexion.
            getOutputStream());
        salida.flush();
        entrada = new ObjectInputStream(conexion.
            getInputStream());
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

public static void finalizarServicio()
{
    try
    {
}

```

```
        connexion.close();  
    }  
    catch(Exception e)  
    {  
    }  
}  
}
```

Para el caso de “Eliminar” se procederá a eliminar el host seleccionado en la base de datos y actualizar el listado.

### **8.1.2.1. Encendido Remoto**

En el caso de “Encender” se navegará hacia otro layout que pedirá la ip (normalmente de broadcast) y predefinidamente tomará la dirección MAC del host para mandar el paquete mágico y poder encender el host remotamente. El código de esta función es el siguiente:

WOL.java

```
package garciaponce.miguel;

import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;

public class WOL extends Activity {
    private EditText IPDif;
    private EditText DirMacWOL;
    private Button btnEncender;

    private String IP;
    private String MAC;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.wakeonlan);

        Bundle extras = getIntent().getExtras();
        if(extras != null)
        {
            IP = extras.getString("DirIP");
            MAC = extras.getString("DirMac");
        }
    }
}
```

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID180

```
IPDif = (EditText) findViewById(R.id.IPDifusion);
DirMacWOL = (EditText) findViewById(R.id.DirMacWOL);
IPDif.setText(IP);
DirMacWOL.setText(MAC);

btnEncender = (Button) findViewById(R.id.BtnEncender);

btnEncender.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        WakeOnLan(IPDif.getText().toString(), DirMacWOL.
                getText().toString());

        Intent intent = new Intent(WOL.this, ListaHosts.
                getClass());
        startActivity(intent);
    }
});

public static final int PORT = 9;

protected void WakeOnLan(String ipStr, String macStr) {
    try {
        // Construcción del paquete mágico
        // El paquete mágico es una trama ethernet que
        // comienza con 6 bytes de cabecera FF FF FF FF FF FF
        // y sigue con 16 repeticiones de la dirección
        // física MAC
        byte[] macBytes = getMacBytes(macStr);
        byte[] bytes = new byte[6 + 16 * macBytes.length];
        for (int i = 0; i < 6; i++) {
            bytes[i] = (byte) 0xff;
        }
        for (int i = 6; i < bytes.length; i += macBytes.
                length) {
            System.arraycopy(macBytes, 0, bytes, i,
                    macBytes.length);
        }

        InetAddress address = InetAddress.getByName(ipStr);
        DatagramPacket packet = new DatagramPacket(bytes,
                bytes.length, address, PORT);
        DatagramSocket socket = new DatagramSocket();
        socket.send(packet);
        socket.close();

        Log.i("WOL", "Wake-on-LAN packet sent.");
    } catch (Exception e) {
        Log.i("WOL", "Failed to send Wake-on-LAN
                packet: " + e);
    }
}
```

```

private byte[] getMacBytes(String macStr) throws
    IllegalArgumentException {
    byte[] bytes = new byte[6];
    String [] hex = macStr.split(”(\\\\:|\\\\-)");
    // Si la longitud es distinta de 6 lanzamos excepción
    if (hex.length != 6) {
        throw new IllegalArgumentException(”Invalid MAC
            address.");
    }
    try {
        // Convertimos a un vector de byte
        for (int i = 0; i < 6; i++) {
            bytes[i] = (byte) Integer.parseInt(hex[i], 16);
        }
    }
    catch (NumberFormatException e) {
        throw new IllegalArgumentException(”Direccion MAC
            hexadecimal invalida");
    }
    return bytes;
}

```

Observamos que se construye el paquete mágico como se describe a continuación: “trama ethernet que comienza con 6 bytes de cabecera FF FF FF FF FF FF y sigue con 16 repeticiones de la dirección física MAC”. Una vez construido se manda a la ip proporcionada.

Podemos también crear un host al hacer click sobre el botón “Nuevo”, el cual provocará la navegación a la siguiente pantalla:

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID182



Figura 8.4: Layout “creahost.xml”

estando su comportamiento descrito por:

CrearHost.java

---

```
package garciaponce.miguel;

import android.app.Activity;
import android.content.Intent;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class CrearHost extends Activity {
    private EditText NombreMaquina;
```

```

private EditText DirIP;
private EditText Puerto;
private EditText DirMac;
private Button AgregaHost;
private String NombreBD = "BDRemSys";
private HostsSQL usdbh;
private SQLiteDatabase db;
private Toast toast;

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.creahost);

    // Accedemos a la BD.
    usdbh = new HostsSQL(CrearHost.this, NombreBD, null, 1);
    db = usdbh.getWritableDatabase();

    // Si pulsamos sobre agregarHost
    AgregaHost = (Button) findViewById(R.id.CreaHost);
    AgregaHost.setOnClickListener(new OnClickListener() {
        public void onClick(View v) {
            // Obtenemos los datos y componentes.
            String Usuario = SesionLocal.NombreUsuario;
            NombreMaquina = (EditText) findViewById(R.id.
                NombreMaquina);
            DirIP = (EditText) findViewById(R.id.DirIP);
            Puerto = (EditText) findViewById(R.id.Puerto);
            DirMac = (EditText) findViewById(R.id.DirMAC);

            // Ejecutamos la sentencia SQL para agregarla a la
            // base de datos a la tabla Host.
            db.execSQL("INSERT INTO Hosts (usuario,nombrehost,
                dirip,puerto,dirmac) VALUES ('" + Usuario.trim()
                () + "','" + NombreMaquina.getText().toString()
                () .trim() +
                "','" + DirIP.getText().toString()
                () .trim() + "','" + Puerto.
                getText().toString().trim() +
                "','" + DirMac.getText().
                toString().trim() + "')");

            // Visualizamos un toast para informar de la accion
            // anterior.
            toast = Toast.makeText(getApplicationContext(),"
                Host creado correctamente", Toast.LENGTH_SHORT)
            ;
            toast.show();

            // Cerramos la base de datos.
            db.close();

            // Volvemos al Layout anterior.
            Intent intent = new Intent(CrearHost.this,
                ListaHosts.class);
            startActivity(intent);
        }
    });
}

```

```
        }  
    }  
}
```

Pasamos a ver ahora el menu principal, obtenido tras pulsar “Conectar” en el menú contextual de la lista de hosts.

### 8.1.3. Menú Principal

El layout será el siguiente:



Figura 8.5: Layout “mainmenu.xml”

estándo su comportamiento descrito por:

Mainmenu.java


---

```

package garciaponce.miguel;

import java.io.IOException;
import java.io.OptionalDataException;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.KeyEvent;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;

public class Mainmenu extends Activity {
    private Button btnInfSys;
    private Button btnProcesos;
    private Button btnNavegacion;
    private Button btnScripts;
    private Button btnOrdenLibre;
    private Button btnReinicio;
    private Button btnApagado;
    private Toast toast;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.mainmenu);
        Conexion.ACTIVIDAD=this;
        if(Conexion.conexion == null){
            toast = Toast.makeText(getApplicationContext(),"No
                se ha conectado" , Toast.LENGTHLONG);
            toast.show();
            Intent intent = new Intent(Mainmenu.this , Login.
                class);
            startActivity(intent);
        }else{
            if(Conexion.conexion.isConnected()){
                toast = Toast.makeText(
                    getApplicationContext(),"Conectando al
                    host " + Conexion.ObtenerIP() , Toast.
                    LENGTHLONG);
                toast.show();
            }
        }
        // Mandamos mensaje para obtener el sistema operativo del
        // sistema remoto.
        try {
            Conexion.salida.flush();
            Conexion.salida.writeObject("MainMenu");
        }
    }
}

```

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID186

```
String mensaje = (String) Conexion.entrada.  
    readObject();  
    SesionLocal.establecerSistemaOperativo(  
        mensaje);  
} catch (OptionalDataException e1) {  
    // TODO Auto-generated catch block  
    e1.printStackTrace();  
} catch (ClassNotFoundException e1) {  
    // TODO Auto-generated catch block  
    e1.printStackTrace();  
} catch (IOException e1) {  
    // TODO Auto-generated catch block  
    e1.printStackTrace();  
}  
  
// Obtenemos componentes gráficos y actuamos según botones.  
final AlertDialog.Builder alert = new AlertDialog.Builder(  
    this);  
btnInfSys = (Button)findViewById(R.id.BotonInfSistema);  
btnInfSys.setOnClickListener(new OnClickListener() {  
    public void onClick(View v) {  
        Intent intent = new Intent(Mainmenu.this ,  
            Infssysmenu.class);  
        startActivity(intent);  
    }  
});  
  
btnProcesos = (Button)findViewById(R.id.BotonProcesos);  
btnProcesos.setOnClickListener(new OnClickListener() {  
    public void onClick(View v) {  
        Intent intent = new Intent(Mainmenu.this ,  
            InfProcesos.class);  
        startActivity(intent);  
    }  
});  
  
btnNavegacion = (Button)findViewById(R.id.BotonNavegacion);  
btnNavegacion.setOnClickListener(new OnClickListener() {  
    public void onClick(View v) {  
        Intent intent = new Intent(Mainmenu.this ,  
            Navegador.class);  
        startActivity(intent);  
    }  
});  
  
btnScripts = (Button)findViewById(R.id.BotonScripts);  
btnScripts.setOnClickListener(new OnClickListener() {  
    public void onClick(View v) {  
        Intent intent = new Intent(Mainmenu.this , Scripts.  
            class);  
        startActivity(intent);  
    }  
});  
  
btnOrdenLibre = (Button)findViewById(R.id.BotonOrdenLibre);  
btnOrdenLibre.setOnClickListener(new OnClickListener() {  
    public void onClick(View v) {
```

```

        Intent intent = new Intent(Mainmenu.this ,
                OrdenLibre.class);
                startActivity(intent);
            }
        });

btnReinicio = (Button)findViewById(R.id.BotonReiniciar);
btnReinicio.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        alert.setMessage("¿Seguro que desea reiniciar el
                        sistema?");
        alert.setPositiveButton("Reiniciar", new
                DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int
                            whichButton) {
                        try {
                            Conexion.salida.
                                flush();
                            Conexion.salida.
                                writeObject(""
                                    Reiniciar");
                        } catch (IOException e) {
                            // TODO Auto-
                            generated catch
                            block
                            e.printStackTrace();
                        }
                    }
                });
        alert.show();
    }
});

alert.setNegativeButton("Cancelar",
        new DialogInterface.OnClickListener
        () {
    public void onClick(DialogInterface dialog, int whichButton) {
        dialog.cancel();
    }
});
});

btnApagado = (Button)findViewById(R.id.BotonApagar);
btnApagado.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        alert.setMessage("¿Seguro que desea apagar el
                        sistema?");
        alert.setPositiveButton("Reiniciar", new
                DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int
                            whichButton) {
                        try {
                            Conexion.salida.
                                flush();
                            Conexion.salida.
                                writeObject(""

```

```

        Apagar");
    } catch (IOException e) {
        // TODO Auto-
        // generated catch
        // block
        e.printStackTrace();
    }
}

alert.setNegativeButton("Cancelar",
    new DialogInterface.OnClickListener
    () {
        public void onClick(DialogInterface dialog,
            int whichButton) {
            dialog.cancel();
        }
    });
alert.show();
}
}

@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    if ((keyCode == KeyEvent.KEYCODEBACK)) {
        // Si pulsamos el botón BACK, finalizamos el servicio
        // de conexión y volvemos al layout ListaHosts

        try {
            Conexion.salida.flush();
            Conexion.salida.writeObject("TerminarConexion");
        } catch (OptionalDataException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        } catch (IOException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }

        Intent servicio = new Intent(Mainmenu.this,
            Conexion.class);
        if(stopService(servicio))
        {
            Log.i("Conectar","Servicio finalizado correctamente
");
        }
    }
    else
    {
        Log.i("Conectar","No se ha podido finalizar
servicio Conexion");
    }
}

```

```

        Intent intent = new Intent(Mainmenu.this ,
            ListaHosts.class);
        startActivity(intent);
    }
    return super.onKeyDown(keyCode, event);
}
}

```

podemos observar como desde el menú principal se puede acceder a todas las funcionalidades descritas anteriormente. Lo primero que hace el sistema es comprobar si existe conexión, de lo contrario, lo comunica y vuelve a la pantalla anterior. A cada una de las funcionalidades se accede mediante su botón correspondiente. Las únicas funcionalidades que no son “directas” son las de “Información del sistema” que tienen un submenú el cual veremos a continuación. Si pulsamos sobre la tecla atrás del teléfono, se ejecutaría el método “OnKeyDown()”, el cual terminaría el servicio de conexión y volvería a la lista de hosts del usuario logeado.

#### 8.1.3.1. Reiniciar

La funcionalidad reiniciar está contenida en el layout “mainmenu.xml” y está descrito por:

##### Reiniciar en MainMenu.java

---

```

btnReinicio = (Button)findViewById(R.id.BotonReiniciar);
btnReinicio.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        alert.setMessage("¿Seguro que desea reiniciar el
                        sistema?");
        alert.setPositiveButton("Reiniciar", new
                        DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int
                            whichButton) {
                            try {
                                Conexion.salida.
                                    flush();
                                Conexion.salida.
                                    writeObject(""
                                            Reiniciar");
                            } catch (IOException e) {
                                // TODO Auto-
                                // generated catch
                                // block
                                e.printStackTrace();
                            }
                        }
                    });
        alert.setNegativeButton("Cancelar",
                        new DialogInterface.OnClickListener
                        () {
                        public void onClick(DialogInterface dialog, int
                            whichButton) {

```

```

        dialog.cancel();
    }
}
alert.show();
}
});

```

Simplemente lo que hace es mandar la orden para reiniciar al servidor. El servidor ya dependiendo del sistema operativo se encargará de dar la orden adecuada. Además tendremos una ventana de confirmación para el reinicio.

#### 8.1.3.2. Apagar

La funcionalidad reiniciar está contenida en el layout “mainmenu.xml” y está descrito por:

Funcion apagador en MainMenu.java

---

```

btnApagado = (Button) findViewById(R.id.BotonApagar);
btnApagado.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        alert.setMessage("¿Seguro que desea apagar el
                        sistema?");
        alert.setPositiveButton("Reiniciar", new
                DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int
                            whichButton) {
                        try {
                            Conexion.salida.
                                flush();
                            Conexion.salida.
                                writeObject(""
                                        + "Apagar");
                        } catch (IOException e) {
                            // TODO Auto-
                            // generated catch
                            // block
                            e.printStackTrace();
                        }
                    }
                });
        alert.setNegativeButton("Cancelar",
                new DialogInterface.OnClickListener
                () {
                    public void onClick(DialogInterface dialog, int
                            whichButton) {
                        dialog.cancel();
                    }
                });
        alert.show();
    }
});

```

#### 8.1.4. Menú Información del sistema

El layout será el siguiente:

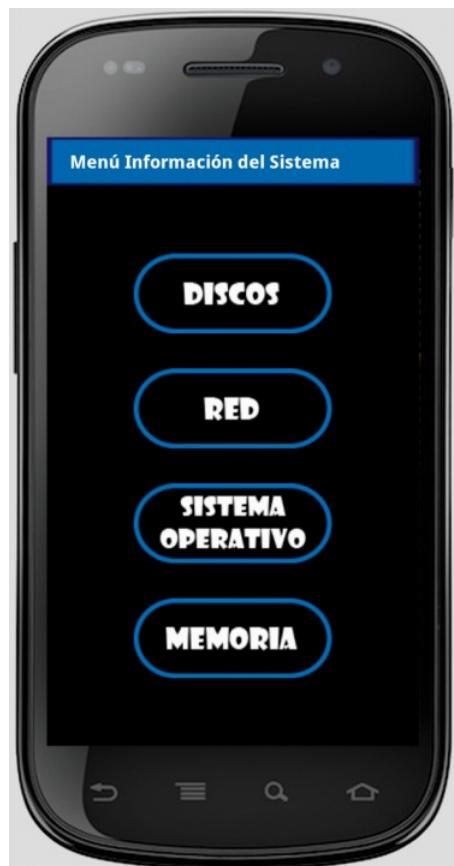


Figura 8.6: Layout “mainmenu.xml”

estándo su comportamiento descrito por:

Infssysmenu.java

```
package garciaponce.miguel;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
```

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID192

```
public class InfSysMenu extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.infsysmenu);
        Conexion.ACTIVIDAD=this;

        // Obtenemos componentes y pasamos al layout segun los
        // botones pulsados.
        Button btnDiscos = (Button)findViewById(R.id.button1);
        btnDiscos.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                Intent intent = new Intent(InfSysMenu.this ,
                    InfDiscos.class);
                startActivity(intent);
            }
        });

        Button btnRed = (Button)findViewById(R.id.button2);
        btnRed.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                Intent intent = new Intent(InfSysMenu.this ,
                    InfRed.class);
                startActivity(intent);
            }
        });

        Button btnSO = (Button)findViewById(R.id.Button01);
        btnSO.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                Intent intent = new Intent(InfSysMenu.this ,
                    InfSisOp.class);
                startActivity(intent);
            }
        });

        Button btnMem = (Button)findViewById(R.id.Button02);
        btnMem.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                Intent intent = new Intent(InfSysMenu.this ,
                    InfMem.class);
                startActivity(intent);
            }
        });
    }
}
```

La navegación está controlada por 4 botones, cada uno de los cuales nos dará paso a la diferente información que corresponde con las funcionalidades del sistema que son: Discos, Red, Sistema Operativo y Memoria. En las siguientes secciones, veremos como se ha implementado cada uno de ellos.

#### 8.1.4.1. Información de discos

El layout será el siguiente:

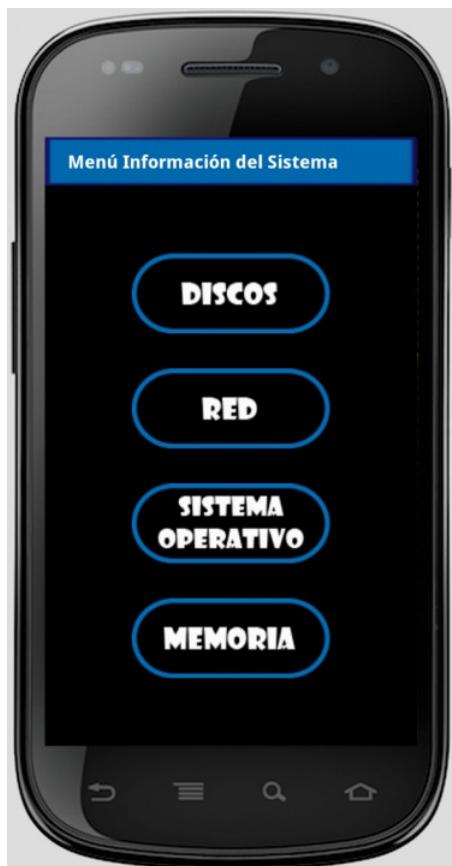


Figura 8.7: Layout “infdiscos.xml”

estándo su comportamiento descrito por:

---

#### InfDiscos.java

---

```
package garciaponce.miguel;

import java.io.IOException;
import java.util.Vector;

import android.app.Activity;
import android.os.Bundle;
import android.view.LayoutInflater;
```

```

import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;

public class InfDiscos extends Activity {
    private Vector<Disco> discos;
    private ListView lstDiscos;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.infdiscos);
        Conexion.ACTIVIDAD=this;

        discos = new Vector<Disco>();

        try {
            // Enviamos el mensaje Navegacion para obtener las
            // unidades (Mi PC) y empezar a visualizarlas.
            Conexion.salida.flush();
            Conexion.salida.writeObject("InfDiscos");

            String mensaje = (String) Conexion.entrada.
                readObject();
            // Mientras no recibamos "Fin_InfDiscos",
            // obtenemos los datos de nombre, tipo,
            // ocupado y tamñototal.
            while (!mensaje.equals("Fin_InfDiscos")){
                String nombre = mensaje;
                mensaje = (String) Conexion
                    .entrada.readObject();
                String Tipo = mensaje;
                mensaje = (String) Conexion
                    .entrada.readObject();
                int TamTotal = Integer.
                    parseInt(mensaje);
                mensaje = (String) Conexion
                    .entrada.readObject();
                int Ocupado = Integer.
                    parseInt(mensaje);
                Disco aux = new Disco(
                    nombre, Tipo, Ocupado,
                    TamTotal);
                discos.add(aux);
                mensaje = (String) Conexion
                    .entrada.readObject();
            }
            // Establecemos adaptador de Discos y
            // registramos MenuContextual.
            AdaptadorDisco adaptador = new
                AdaptadorDisco(this);
            lstDiscos = (ListView)findViewById(R.id.
                ListaDiscos);
            lstDiscos.setAdapter(adaptador);
        }
    }
}

```

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID195

```
        registerForContextMenu(lstDiscos);

    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (ClassNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

class AdaptadorDisco extends ArrayAdapter {

    Activity context;

    AdaptadorDisco(Activity context) {
        super(context, R.layout.disco, discos);
        this.context = context;
    }

    public View getView(int position, View convertView,
                       ViewGroup parent)
    {
        View item = convertView;
        ViewHolder holder;

        if(item == null)
        {
            LayoutInflater inflater = context.
                getLayoutInflater();
            item = inflater.inflate(R.layout.disco,
                                   null);

            holder = new ViewHolder();
            holder.Nombre = (TextView)item.findViewById(
                R.id.NombreDisco);
            holder.Tipo = (TextView)item.findViewById(R
                .id.TipoDisco);
            holder.PorOcupado = (TextView)item.
                findViewById(R.id.PorcentajeOcupado);
            holder.Tam = (TextView)item.findViewById(R.
                id.TamTotal);

            item.setTag(holder);
        }
        else
        {
            holder = (ViewHolder)item.getTag();
        }

        holder.Nombre.setText(discos.elementAt(
            position).NombreDisco());
    }
}
```

```
        holder.Tipo.setText(discos.elementAt(
            position).TipoSistemaDisco());
        holder.PorOcupado.setText(String.valueOf(
            discos.elementAt(position).PorUsado())
            + "%");
        holder.Tam.setText(String.valueOf(discos.
            elementAt(position).Tamanio())+ "Gb");
    return(item);
}
}

static class ViewHolder {
    public TextView Nombre;
    public TextView Tipo;
    public TextView PorOcupado;
    public TextView Tam;
}
}
```

Como podemos observar, el comportamiento de esta pantalla es el siguiente. Se va recibiendo los datos desde el servidor, como son **el nombre de la unidad de disco, el tipo, el tamaño total y el porcentaje usado** (estadísticas obtenidas desde el servidor). Esta información se almacena en un vector, que mediante un “adaptador” se visualizará en un componente de tipo ListView de Android.

#### 8.1.4.2. Información de red

El layout será el siguiente:



Figura 8.8: Layout “infred.xml”

estándo su comportamiento descrito por:

InfRed.java

---

```
package garciaponce.miguel;

import java.io.IOException;
import java.util.Vector;

import android.app.Activity;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.LinearLayout;
```

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID198

```
import android.widget.ListView;
import android.widget.TextView;

public class InfRed extends Activity {
    private Vector<Red> DispRed;
    private ListView lstDispRed;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.infred);
        Conexion.ACTIVIDAD=this;

        //Creamos la estructura de datos necesaria
        DispRed = new Vector<Red>();

        // Enviamos orden al servidor para obtener datos de red
        try {
            Conexion.salida.flush();
            Conexion.salida.writeObject("Red");

            // Obtenemos los datos del servidor.
            String mensaje = (String) Conexion.entrada.
                readObject();
            while(!mensaje.equals("Fin_Red")){
                // Obtenemos los valores desde el
                // servidor.
                String Adaptador = null, dirfis =
                    null, dirip = null, descripcion =
                    null, mask=null, estado=null,
                    dhcp=null;
                Adaptador=mensaje;
                mensaje = (String) Conexion.entrada
                    .readObject();
                dirfis=mensaje;
                mensaje = (String) Conexion.entrada
                    .readObject();
                dirip=mensaje;
                mensaje = (String) Conexion.entrada
                    .readObject();
                descripcion=mensaje;
                mensaje = (String) Conexion.entrada
                    .readObject();
                mask=mensaje;
                mensaje = (String) Conexion.entrada
                    .readObject();
                estado=mensaje;
                mensaje = (String) Conexion.entrada
                    .readObject();
                dhcp=mensaje;

                Red aux = new Red(Adaptador, dirfis,
                    dirip, descripcion, mask, estado,
                    dhcp);
                DispRed.add(aux);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```

        mensaje = ( String ) Conexion.entrada
                    .readObject();
    }

    for( int i=0;i<DispRed.size(); i++ ){
        if( DispRed.get(i).Adaptador().
            equals("") && DispRed.get(i).
            DirFis().equals("") && DispRed.
            get(i).DirIP().equals("") &&
            DispRed.get(i).Descripcion().
            equals("") && DispRed.get(i).
            Mascara().equals("") && DispRed.
            get(i).Estado().equals("") &&
            DispRed.get(i).DHCP().equals(""))
            DispRed.remove(i);
    }

    // Establecemos adaptador
    AdaptadorRed adaptador = new AdaptadorRed(
        this);
    lstDispRed = ( ListView )findViewById(R.id.
        ListaRed);
    lstDispRed.setAdapter(adaptador);

} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ClassNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
}

class AdaptadorRed extends ArrayAdapter {
    Activity context;

    AdaptadorRed(Activity context) {
        super(context, R.layout.red, DispRed);
        this.context = context;
    }

    public View getView(int position, View convertView,
        ViewGroup parent)
    {
        View item = convertView;
        ViewHolder holder;

        if(item == null)
        {
            LayoutInflator inflater = context.
                getLayoutInflater();
            item = inflater.inflate(R.layout.red, null)
            ;
        }
    }
}

```

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID200

```
holder = new ViewHolder();
holder.Adapter = (TextView) item.
    findViewById(R.id.Adapter);
holder.DirFis = (TextView) item.findViewById
    (R.id.DirFisica);
holder.DirIP = (TextView) item.findViewById(
    R.id.DirIP);
holder.Descripcion = (TextView) item.
    findViewById(R.id.Descripcion);
holder.Mascara = (TextView) item.
    findViewById(R.id.Mascara);
holder.Estado = (TextView) item.findViewById
    (R.id.Estado);
holder.DHCP = (TextView) item.findViewById(R
    .id.DHCP);
holder.TextoDHCP = (TextView) item.
    findViewById(R.id.textView12);
holder.ImagenRed= (ImageView) item.
    findViewById(R.id.ImagenRed);
LinearLayout.LayoutParams layoutParams =
    new LinearLayout.LayoutParams(60, 60);
holder.ImagenRed.setLayoutParams(
    layoutParams);

item.setTag(holder);
}
else
{
    holder = (ViewHolder) item.getTag();
}

holder.Adapter.setText(DispRed.elementAt(position
    ).Adapter());
holder.DirFis.setText(DispRed.elementAt(
    position).DirFis());
holder.DirIP.setText(DispRed.elementAt(
    position).DirIP());
holder.Descripcion.setText(DispRed.
    elementAt(position).Descripcion());
holder.Mascara.setText(DispRed.elementAt(
    position).Mascara());
if(holder.DirIP.getText().toString().trim()
    .length()>0){
    holder.Estado.setText("Conectado");
} else{
    holder.Estado.setText("Desconectado
        ");
}

holder.DHCP.setText(DispRed.elementAt(
    position).DHCP());

if(SesionLocal.ObtenerSistemaOperativo().
    equals("Linux")){
}
```

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID201

```
        holder.TextoDHCP.setVisibility(View
                .INVISIBLE);
    }

    if(holder.Adaptador.getText().toString().
            startsWith("Adaptador de Ethernets") ||
            holder.Adaptador.getText().toString().
            startsWith("eth") || holder.Adaptador.
            getText().toString().startsWith("lo")){
        holder.ImagenRed.
            setDrawable(R.
            drawable.iconoethernet);
    } else{
        holder.ImagenRed.
            setDrawable(R.
            drawable.iconowireless);
    }
    return(item);
}

static class ViewHolder {
    public TextView Adaptador;
    public TextView DirFis;
    public TextView DirIP;
    public TextView Descripcion;
    public TextView Mascara;
    public TextView Estado;
    public TextView DHCP;
    public ImageView ImagenRed;
    public TextView TextoDHCP;
}
}
```

El comportamiento de esta pantalla es el siguiente: se va recibiendo los datos desde el servidor, como son el **nombre del adaptador de red, la dirección física, la dirección IP, descripción, máscara de red, estado del adaptador y la utilización de DHCP**. Esta información se almacena en un vector, que mediante un “adaptador” se visualizará en un componente de tipo ListView de Android.

### 8.1.4.3. Información del Sistema Operativo

El layout será el siguiente:

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID202



Figura 8.9: Layout “infsisop.xml”

estándo su comportamiento descrito por:

InfSisOp.java

---

```
package garciaponce.miguel;

import java.io.IOException;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class InfSisOp extends Activity {
    private TextView PropiedadesSistemaOperativo;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
```

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID203

```
super.onCreate(savedInstanceState);
setContentView(R.layout.infsisop);
Conexion.ACTIVIDAD=this;

PropiedadesSistemaOperativo = (TextView) findViewById(R.id
    .PropiedadesSistemaOperativo);

// Mandamos orden para obtener la informacion
try {
    Conexion.salida.flush();
    Conexion.salida.writeObject("SistemaOperativo");

    // Recibimos datos del sistema operativo
    String mensaje = (String) Conexion.entrada.
        readObject();
    while (!mensaje.equals("Fin_SistemaOperativo")){
        if(mensaje.startsWith("Path
            Librerías Java")){
            String datos [];
            datos = mensaje.split(";");
            PropiedadesSistemaOperativo
                .append(datos[0] + "\n");
            for(int i = 1;i<datos.length
                ;i++){
                if(datos[i].length
                    ()>3)
                    PropiedadesSistemaOperativo
                        .append
                            ("\t" +
                                datos[
                                    i].trim
                                () + "\n");
            }
            PropiedadesSistemaOperativo
                .append("\n");
        }else{
            PropiedadesSistemaOperativo
                .append(mensaje + "\n\n");
        }
        mensaje = (String) Conexion.entrada
            .readObject();
    }
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ClassNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
}
```

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID204

}

En esta pantalla, su comportamiento sólo es de recibir tanto el nombre de la propiedad, como su valor y mostrarlo mediante un TextView al cliente.

### 8.1.4.4. Información de Memoria

El layout será el siguiente:



Figura 8.10: Layout “infmem.xml”

estándo su comportamiento descrito por:

InfMem.java

---

```
package garciaponce.miguel;
```

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID205

```
import java.io.IOException;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;

public class InfMem extends Activity {
    private String memftotal=null, memfdis=null, memvirtammax=null,
               memvirdis=null, memvirus=null;
    private TextView MemFisTot;
    private TextView MemFisDis;
    private TextView MemVirTot;
    private TextView MemVirDis;
    private TextView MemVirUs;
    private Button btnKb;
    private Button btnMb;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.infmem);
        Conexion.ACTIVIDAD=this;

        // Obtenemos los componentes
        MemFisTot = (TextView) findViewById(R.id.MemFisTot);
        MemFisDis =(TextView) findViewById(R.id.MemFisDis);
        MemVirTot = (TextView) findViewById(R.id.MemVirTot);
        MemVirDis = (TextView) findViewById(R.id.MemVirDis);
        MemVirUs = (TextView) findViewById(R.id.MemVirUs);

        btnKb = (Button) findViewById(R.id.btnKb);
        btnMb = (Button) findViewById(R.id.btnMb);

        // Mandamos la orden de MEmoria
        try {
            Conexion.salida.flush();
            Conexion.salida.writeObject("Memoria");

            // Obtenemos los datos del servidor
            String mensaje = (String) Conexion.entrada.readObject();
            while(!mensaje.equals("Fin_Memoria")){
                memftotal=mensaje;
                mensaje = (String) Conexion.entrada.readObject();
                memfdis=mensaje;
                mensaje = (String) Conexion.entrada.readObject();
                memvirtammax=mensaje;
                mensaje = (String) Conexion.entrada.readObject();
                memvirdis=mensaje;
                mensaje = (String) Conexion.entrada.readObject();
                memvirus=mensaje;
                mensaje = (String) Conexion.entrada.readObject();
            }
        }
```

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID206

```
// Dependiendo de el sistema operativo que tenga el
// sistema remoto, vamos a activar o desactivar KB o MB.
if(SesionLocal.ObtenerSistemaOperativo().equals("Linux"))
    btnKb.setEnabled(false);
else
    btnMb.setEnabled(false);

MemFisTot.setText(memftotal.trim());
MemFisDis.setText(memfdis.trim());
MemVirTot.setText(memvirtamax.trim());
MemVirDis.setText(memvirdis.trim());
MemVirUs.setText(memvirus.trim());

// Actuamos segun la pulsacion de los botones KB y Mb,
// segun linux o Windows.
btnMb.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        btnMb.setEnabled(false);
        btnKb.setEnabled(true);

        if(SesionLocal.ObtenerSistemaOperativo().equals("Linux")){
            int res= Integer.valueOf(memftotal.substring(
                0, memftotal.length()-3))/1024;
            MemFisTot.setText(String.valueOf(res)+" Mb");
            res= Integer.valueOf(memfdis.substring(0,
                memfdis.length()-3))/1024;
            MemFisDis.setText(String.valueOf(res)+" Mb");
            res= Integer.valueOf(memvirtamax.substring(0,
                memvirtamax.length()-3))/1024;
            MemVirTot.setText(String.valueOf(res)+" Mb");
            res= Integer.valueOf(memvirdis.substring(0,
                memvirdis.length()-3))/1024;
            MemVirDis.setText(String.valueOf(res)+" Mb");
            res= Integer.valueOf(memvirus.substring(0,
                memvirus.length()-3))/1024;
            MemVirUs.setText(String.valueOf(res)+" Mb");
        }else{
            MemFisTot.setText(memftotal.trim());
            MemFisDis.setText(memfdis.trim());
            MemVirTot.setText(memvirtamax.trim());
            MemVirDis.setText(memvirdis.trim());
            MemVirUs.setText(memvirus.trim());
        }
    }
});

btnKb.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        btnMb.setEnabled(true);
        btnKb.setEnabled(false);
```

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID207

```
    if(SesionLocal.ObtenerSistemaOperativo().equals("Linux")){
        MemFisTot.setText(memftotal.trim());
        MemFisDis.setText(memfdis.trim());
        MemVirTot.setText(memvirtamax.trim());
        MemVirDis.setText(memvirdis.trim());
        MemVirUs.setText(memvirus.trim());
    }else{
        int res= Integer.valueOf(memftotal.substring(0, memftotal.length()-3).trim().replace(".", ""))*1024;
        MemFisTot.setText(String.valueOf(res)+" Kb");
        res= Integer.valueOf(memfdis.substring(0, memfdis.length()-3).trim().replace(".", ""))*1024;
        MemFisDis.setText(String.valueOf(res)+" Kb");
        res= Integer.valueOf(memvirtamax.substring(0, memvirtamax.length()-3).trim().replace(".", ""))*1024;
        MemVirTot.setText(String.valueOf(res)+" Kb");
        res= Integer.valueOf(memvirdis.substring(0, memvirdis.length()-3).trim().replace(".", ""))*1024;
        MemVirDis.setText(String.valueOf(res)+" Kb");
        res= Integer.valueOf(memvirus.substring(0, memvirus.length()-3).trim().replace(".", ""))*1024;
        MemVirUs.setText(String.valueOf(res)+" Kb");
    }
}
}) ;

} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ClassNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
}
```

En esta pantalla presentamos los datos de memoria: **memoria física total, memoria física disponible, memoria virtual total, memoria virtual disponible y memoria virtual usada**. También tendremos dos botones complementarios para poder visualizar la información en diferentes unidades (Mb y Kb).

### 8.1.5. Procesos

El layout será el siguiente:

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID208

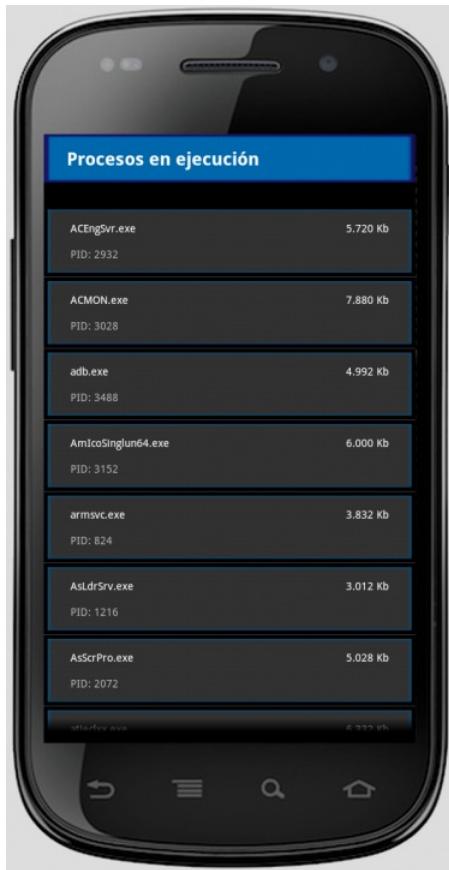


Figura 8.11: Layout “infprocesos.xml”

estando su comportamiento descrito por:

InfProcesos.java

---

```
package garciaponce.miguel;

import java.io.IOException;
import java.util.Vector;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.ContextMenu;
import android.view.ContextMenu.ContextMenuItemInfo;
import android.view.KeyEvent;
import android.view.LayoutInflater;
import android.view.MenuInflater;
```

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID209

```
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemLongClickListener;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

public class InfProcesos extends Activity {
    private ListView lstOpciones;
    Vector<Proceso> procesos;
    private int pos;
    Toast toast;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.infprocesos);

        // Creamos la estructura de datos.
        procesos = new Vector<Proceso>();

        // Mandamos orden de listar procesos
        try {
            Conexion.salida.flush();
            Conexion.salida.writeObject("Listado");
            String mensaje = (String) Conexion.entrada.readObject();
            ;
            String[] datos;
            // Obtenemos los datos, introduciendolos en el vector.
            while(!mensaje.equals("fin")){
                datos = mensaje.split(" ");
                if(!datos[0].equals("PID") && !datos[1].equals(
                    "CMD") && !datos[1].equals("SZ")){
                    Proceso aux = new Proceso(datos[0],datos
                        [1],datos[2]);
                    procesos.add(aux);
                }
                mensaje = (String) Conexion.entrada.readObject();
            }

            // Establecemos el adaptador de procesos y registramos
            // menu contextual
            AdaptadorProcesos adaptador = new AdaptadorProcesos(
                this);
            lstOpciones = (ListView)findViewById(R.id.ListaProcesos
                );
            lstOpciones.setAdapter(adaptador);
            registerForContextMenu(lstOpciones);

            // Si hacemos una pulsacion larga, registramos la
            // posicion.
        }
```

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID210

```
lstOpciones.setOnItemLongClickListener(new
    OnItemLongClickListener() {
        public boolean onItemLongClick(AdapterView<?>
            parent, final View v, int position, long id)
        {
            // record position/id/whatever here
            pos=position;
            return false;
        }
    });
}

} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ClassNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

}

@Override
public void onCreateContextMenu(ContextMenu menu, View v,
    ContextMenuItemInfo menuInfo)
{
    super.onCreateContextMenu(menu, v, menuInfo);

    MenuInflater inflater = getMenuInflater();

    if(v.getId() == R.id.ListaProcesos)
    {
        inflater.inflate(R.menu.menuprocesos, menu);
    }
}

// Actuamos segun el menu contextual: eliminar proceso.
@Override
public boolean onContextItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.menuprocesos1:
            Toast toast1 = Toast.makeText(getApplicationContext()
                () , "Se ha mandado la orden para eliminar el
                proceso " + procesos.get(pos).getNom() , Toast.
                LENGTHLONG);
            toast1.show();
            try {
                Conexion.salida.flush();
                Conexion.salida.writeObject("taskkill /F /PID "
                    + procesos.get(pos).getPID());
                String mensaje = (String) Conexion.entrada.
                    readObject();
                while(!mensaje.equals("fin_elimina_proceso")){
                    mensaje = (String) Conexion.entrada.
                        readObject();
                }
            }
    }
}
```

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID211

```
        toast1 = Toast.makeText(getApplicationContext()
            , "Se ha eliminado el proceso ", Toast.
            LENGTHLONG);
        toast1.show();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (ClassNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    // Cuando se elimina el proceso, recargamos el
    // layout con los nuevos datos.(sin el proceso ya
    // eliminado)
    Intent intent = new Intent(InfProcesos.this ,
        InfProcesos.class);
    startActivity(intent);
    return true;
default:
    return super.onContextItemSelected(item);
}
}

@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    if ((keyCode == KeyEvent.KEYCODE_BACK)) {
        Intent intent = new Intent(InfProcesos.this , Mainmenu.
            class);
        startActivity(intent);
    }
    return super.onKeyDown(keyCode, event);
}

class AdaptadorProcesos extends ArrayAdapter {
    Activity context;

    AdaptadorProcesos(Activity context) {
        super(context, R.layout.proc, procesos);
        this.context = context;
    }

    public View getView(int position, View convertView,
        ViewGroup parent)
    {
        View item = convertView;
        ViewHolder holder;

        if(item == null)
        {
            LayoutInflator inflater = context.getLayoutInflater
                ();
            item = inflater.inflate(R.layout.proc, null);

            holder = new ViewHolder();
            item.setTag(holder);
        }
        else
            holder = (ViewHolder)item.getTag();

        holder.txtProc.setText(procesos[position].proc);
        holder.txtEstado.setText(procesos[position].estado);
        holder.txtTiempo.setText(procesos[position].tiempo);
        holder.txtUso.setText(procesos[position].uso);
        holder.txtUso.setCompoundDrawablesWithIntrinsicBounds(R.drawable.
            uso, 0, 0, 0);
        holder.txtUso.setEllipsize(TextUtils.TruncateAt.END);
        holder.txtUso.setSingleLine(true);
        holder.txtUso.setTextColor(Color.parseColor("#000000"));

        return item;
    }
}
```

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID212

```
        holder.DatoProc1 = (TextView) item.findViewById(R.id
                .DatoProc1);
        holder.DatoProc2 = (TextView) item.findViewById(R.id
                .DatoProc2);
        holder.DatoProc3 = (TextView) item.findViewById(R.id
                .DatoProc3);

        item.setTag(holder);
    }
    else
    {
        holder = (ViewHolder) item.getTag();
    }

    holder.DatoProc1.setText(procesos.elementAt(position).
            getNom());
    holder.DatoProc2.setText(procesos.elementAt(position).
            getTam() + " Kb");
    holder.DatoProc3.setText("PID: " + procesos.elementAt(
            position).getPID());

    return(item);
}
}

static class ViewHolder {
    public TextView DatoProc1;
    public TextView DatoProc2;
    public TextView DatoProc3;
}
}
```

El código básicamente lo que utiliza es un Vector de Proceso, que utiliza para llenar un ListView con los datos de: nombre del proceso, tamaño del proceso y PID del proceso. La clase Proceso, está definida como:

Proceso.java

---

```
public class Proceso {
    private String PID;
    private String nomproc;
    private String tamproc;

    public Proceso(String pid, String nom, String tam){
        PID=pid;
        nomproc=nom;
        tamproc=tam;
    }

    public String getPID(){
        return PID;
    }

    public String getNom(){
        return nomproc;
    }

    public String getTam(){

```

```
        return tamproc;
    }
}
```

#### 8.1.5.1. Eliminar Proceso

Para eliminar un proceso, tendríamos que, en el anterior Layout, InfProcesos, hacer una pulsación larga sobre cuaquiera de ellos y, una vez que aparezca el menú contextual, seleccionar “Eliminar Proceso”.

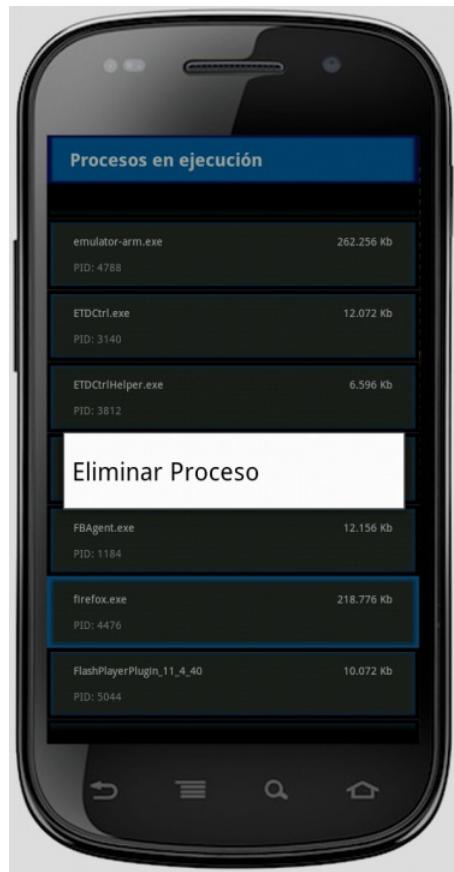


Figura 8.12: Layout “infprocesos.xml”

una vez elegida la opción de “Eliminar Proceso”, se enviará al servidor la orden para eliminar el proceso, junto con su PID. El código se puede ver en el apartado anterior.

### 8.1.6. Navegación

El layout será el siguiente:

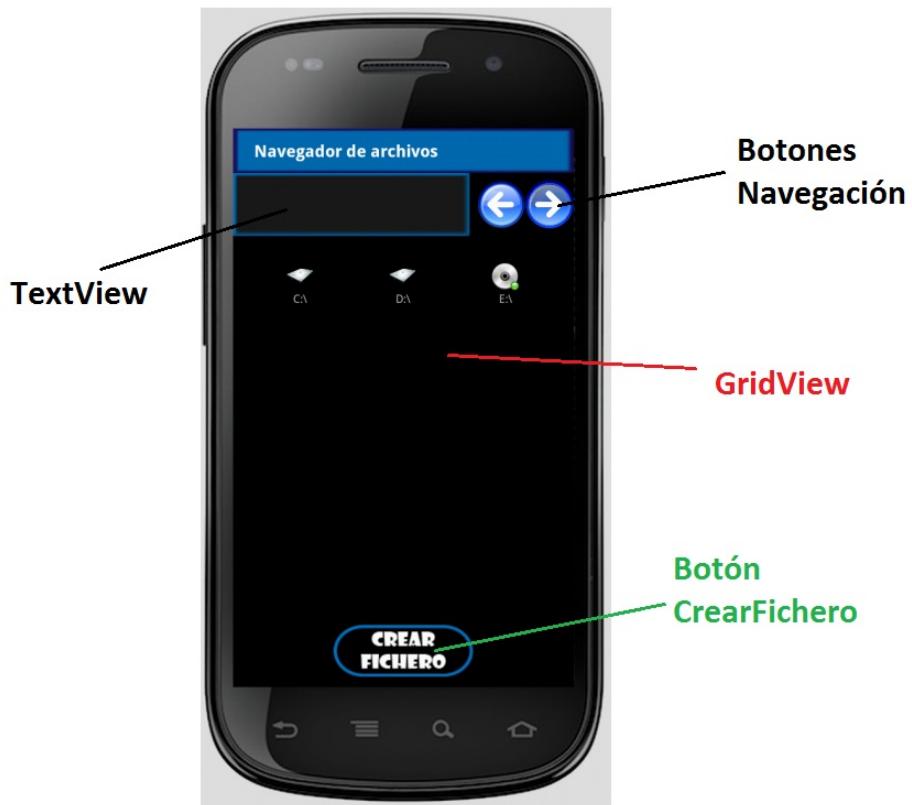


Figura 8.13: Layout “navegador.xml”

Podemos observar como consta de:

- TextView: Para ir mostrando la ruta.
- Botones de Navegación: Para mostrar directorio atrás y directorio adelante.
- GridView: Para mostrar gráficamente el contenido del directorio mostrado en el TextView de la ruta.

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID215

- Botón CrearFichero: Para dar la funcionalidad de crear un fichero en una ruta determinada.

Su código de comportamiento es el siguiente:

Navegador.java

```
package garciaponce.miguel;

import java.io.BufferedInputStream;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OptionalDataException;
import java.net.Socket;
import java.util.Vector;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.os.Environment;
import android.util.Log;
import android.view.ContextMenu;
import android.view.ContextMenu.ContextMenuItemInfo;
import android.view.KeyEvent;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.GridView;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;

public class Navegador extends Activity {
    private TextView Ruta;
    private String NuevaRuta;
    private GridView ListaFicheros;
    private Vector<Fichero> Ficheros;
    private Button btnCrearFichero;
    private Vector<String> RutaPrevias;
    private AdaptadorFicheros adaptador;
    private ImageButton ImagenFlechaIzq;
    private ImageButton ImagenFlechaDer;
    private String mensaje;
    private int pos;
    private boolean IndicadorOpcionPegar;
    private boolean IndicadorCopia;
    private boolean IndicadorCorte;
    private String FicheroOrigenPegado;
    private String argumentos;
    private boolean IndicadorCFichero=false;
    private String RutaCrearFichero;
    private Socket conexiontrans;
```

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID216

```
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.navegador);

    // Inicialización
    IndicadorOpcionPegar=false;
    RutaPrevias=new Vector<String>();
    Ficheros = new Vector<Fichero>();

    Ruta = (TextView) findViewById(R.id.TextViewRuta);

    // Comprobamos si hemos vuelto de la actividad CrearFichero
    Bundle b = this.getIntent().getExtras();
    if(b != null){
        IndicadorCFichero = b.getBoolean("IndicadorCrearFichero");
        RutaCrearFichero = b.getString("Ruta");
    }

    // Si hemos venido de la actividad anterior, actualizamos ruta y
    // GridView
    if(IndicadorCFichero){
        Ruta.setText(RutaCrearFichero);
        ActualizarGridViewAnterior();

        // Establecemos el adaptador.
        adaptador = new AdaptadorFicheros(this);
        ListaFicheros = (GridView) findViewById(R.id.GridViewFicheros)
        ;
        ListaFicheros.setAdapter(adaptador);

    }else{ // Sino, comenzamos una nueva navegación, estableciendo
        // la ruta y obteniendo MiPc

        Ruta.setText("");
        try {
            // Enviamos el mensaje Navegacion para obtener las
            // unidades (Mi PC) y empezar a visualizarlas.
            Conexion.salida.flush();
            Conexion.salida.writeObject("Navegacion");
            MostrarMiPc();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    // Registraremos menu contextual para el GridView
    registerForContextMenu(ListaFicheros);

    // Cuando pulsemos sobre algun fichero, actualizamos ruta y
    // GridView
    ListaFicheros.setOnItemClickListener(new android.widget.
        AdapterView.OnItemClickListener(){
        public void onItemClick(AdapterView<?> parent, final View v,
            int position, long id) {
            // record position/id/whatever here
            ActualizarRuta(parent, v, position,id);
            ActualizarGridView(parent,v,position,id);
        }
    });

    // Al pulsar largo sobre cualquier elemento, guardamos su
    // posicion
}
```

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID217

```
ListaFicheros.setOnItemLongClickListener(new
    OnItemLongClickListener() {
        public boolean onItemLongClick(AdapterView<?> parent, final
            View v, int position, long id) {
            // record position/id/whatever here
            pos=position;
            return false;
        }
    });

// Pulsación del ImageButton FlechaIzquierda
ImagenFlechaIzq = (ImageButton)findViewById(R.id.ImageFlechaIzq);
ImagenFlechaIzq.setOnClickListener(new Button.OnClickListener() {
    public void onClick(View v) {
        // Si no hay ruta especificada (principio), no hacemos
        // nada
        if(SesionLocal.ObtenerSistemaOperativo().equals("Windows"))
        {
            //Windows
            if(Ruta.getText().toString().length()!=0){
                //Guardamos la RutaPrevia para el
                //botónFlechaDerecha
                NuevaRuta="";
                int cont=RutaAnteriorGuardado();

                // Si estamos listando una unidad, para volver a MiPc
                // , recargamos el layout
                if(cont==0){
                    MostrarMiPc();
                    Ruta.setText("");
                }else{
                    Ruta.setText(NuevaRuta);
                    // Mandamos el directorio a listar.
                    ActualizarGridViewAnterior();
                }
            }
        }else{
            //Linux
            if(Ruta.getText().toString().length()>1){
                String NuevaRuta="";
                int cont=RutaAnteriorGuardado();

                // Separamos la ruta por el carácter "/"
                String datos[] = Ruta.getText().toString().split("/");
                for(int i=1;i<(datos.length-1);i++)
                    NuevaRuta=NuevaRuta + "/" + datos[i];
                NuevaRuta = NuevaRuta.trim();
                if(NuevaRuta.equals(""))
                    NuevaRuta="/";
                Ruta.setText(NuevaRuta);
                // Mandamos el directorio a listar.
                ActualizarGridViewAnterior();
            }
        }
    }
});

// Flecha Derecha
ImagenFlechaDer = (ImageButton)findViewById(R.id.ImageFlechaDer);
ImagenFlechaDer.setOnClickListener(new Button.OnClickListener() {
    public void onClick(View v) {
        if(RutaPrevias.size()!=0){
            if(RutaPrevias.lastElement().length()>Ruta.getText()
                ().length()){
                Ruta.setText(RutaPrevias.lastElement());
            }
        }
    }
});
```

CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID218

```

        // Eliminamos el ultimo elemento de las rutas
        // previas
        RutaPrevias.removeElementAt(RutaPrevias.size() - 1);
        // Obtenemos los resultados y almacenamos en un
        // vector Ficheros para visualizarlos en el
        // GridView
        ActualizarGridViewAnterior();
    }

});

// Boton Crear Fichero
btnCrearFichero = (Button) findViewById(R.id.BotonCrearFichero);
btnCrearFichero.setOnClickListener(new Button.OnClickListener() {
    public void onClick(View v) {
        // Nos cercioramos de que sea una ruta válida (no se
        // puede crear en una ruta vacía)
        if ((Ruta.getText().length() >= 3 && SesionLocal.
            ObtenerSistemaOperativo().equals("Windows")) || (
            Ruta.getText().length() >= 1 && SesionLocal.
            ObtenerSistemaOperativo().equals("Linux"))){
            Intent intent = new Intent(Navegador.this,
                CrearFichero.class);
            intent.putExtra("Ruta", Ruta.getText().toString());
            startActivity(intent);
        }
    }
});

private int RutaAnteriorGuardado(){
    int cont = 0;

    if (SesionLocal.ObtenerSistemaOperativo().equals("Windows")){
        // Añadimos la ruta actual a las previas
        RutaPrevias.add(Ruta.getText().toString());

        // Establecemos NuevaRuta a la RutaAnterior
        String ruta[];
        ruta=Ruta.getText().toString().split("\\\\\\");
        cont=0;
        for(int i=0;i<(ruta.length-1);i++){
            if(NuevaRuta.length()==0){
                NuevaRuta=ruta[i] + "\\";
            }else{
                NuevaRuta=NuevaRuta+ruta[i]+ "\\";
            }
            cont++;
        }
    }else{
        //Linux
        RutaPrevias.add(Ruta.getText().toString());
    }
    for(int i=0;i<RutaPrevias.size();i++)
        Log.i("RutaPrevias", RutaPrevias.get(i));
    return cont;
}

// Procedimiento para tratar con directorios . y ..
private void ActualizarRutaAnterior(){
    if(SesionLocal.ObtenerSistemaOperativo().equals("Windows")){
        String ruta[];
        String NuevaRuta="";

```

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID219

```
ruta=Ruta.getText().toString().split("\\\\");
for(int i=0;i<(ruta.length-1);i++){
    if(NuevaRuta.length()==0){
        NuevaRuta=ruta[i] + "\\";
    }else{
        NuevaRuta=NuevaRuta+ruta[i]+"\\";
    }
}
Ruta.setText(NuevaRuta);
}else{
    //Linux
}

}

private void ActualizarGridViewAnterior(){
try {
    Conexion.salida.flush();
    Conexion.salida.writeObject("ActualizarGridView");
    Conexion.salida.flush();
    Conexion.salida.writeObject(Ruta.getText());

    // Obtenemos los resultados y almacenamos en un vector
    // Ficheros para visualizarlos en el GridView
    mensaje = (String) Conexion.entrada.readObject();
    Ficheros.clear();
    int cont=0;
    String[] datos_recibidos;
    while(!mensaje.equals("fin_listado_dir")){
        if(SesionLocal.ObtenerSistemaOperativo().equals("Windows")){
            if(cont > 2){
                String NombreFichero="";
                datos_recibidos = mensaje.split(" ");
                for(int i=1;i<datos_recibidos.length; i++){
                    NombreFichero = NombreFichero+" "+datos_recibidos[i];
                }
                Fichero aux = new Fichero(datos_recibidos[0].toString()
                    .trim(),NombreFichero.trim());
                Ficheros.add(aux);
            }
        }else{
            String NombreFichero="";
            datos_recibidos = mensaje.split(" ");
            for(int i=1;i<datos_recibidos.length; i++){
                NombreFichero = NombreFichero+" "+datos_recibidos[i];
            }
            Fichero aux = new Fichero(datos_recibidos[0].toString()
                .trim(),NombreFichero.trim());
            Ficheros.add(aux);
        }
        cont++;
        mensaje = (String) Conexion.entrada.readObject();
    }

    // Establecemos el adaptador con el nuevo Vector<Fichero>
    // obtenido.
    ListaFicheros = (GridView)findViewById(R.id.GridViewFicheros);
    ListaFicheros.setAdapter(adapter);
}catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ClassNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
}
```

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID220

```
// Función encargada de la actualización de la Ruta.
private void ActualizarRuta(AdapterView<?> parent, final View v, int
    position, long id){
    // Actualizamos la ruta
    if(SesionLocal.ObtenerSistemaOperativo().equals("Windows")){
        if(Ruta.getText().length() == 0){
            Ruta.setText(Ficheros.get(position).getNombre());
        }else{
            // Si el fichero que pulsamos es .. , volvemos a la ruta
            // anterior
            if(Ficheros.get(position).getNombre().equals(".")){
                ActualizarRutaAnterior();
            }else{
                // Si el fichero el cual pulsamos es un punto, no hacemos
                // nada en el cambio de ruta
                if(Ficheros.get(position).getNombre().equals(".")){
                    }else{
                        Ruta.setText(Ruta.getText() + Ficheros.get(position).
                            getNombre() + "\\");
                    }
                }
            }
        }else{
            //Linux
            if(Ruta.getText().toString().length() > 1){
                Ruta.setText(Ruta.getText().toString() + "/" + Ficheros.get(
                    position).getNombre());
            }else{
                Ruta.setText(Ruta.getText().toString() + Ficheros.get(
                    position).getNombre());
            }
        }
    }

private void ActualizarGridView(AdapterView<?> parent, final View v,
    int position, long id){
    // Mandamos el directorio a listar.
    try {
        Conexion.salida.flush();
        Conexion.salida.writeObject("ActualizarGridView");
        Conexion.salida.flush();
        Conexion.salida.writeObject(Ruta.getText());

        // Obtenemos los resultados y almacenamos en un vector
        // Ficheros para visualizarlos en el GridView
        mensaje = (String) Conexion.entrada.readObject();
        Ficheros.clear();
        int cont=0;
        String[] datos_recibidos;
        while(!mensaje.equals("fin_listado_dir")){
            if(SesionLocal.ObtenerSistemaOperativo().equals("Windows")){
            }else{
                if(cont > 2 ){
                    String NombreFichero="";
                    datos_recibidos = mensaje.split(" ");
                    for(int i=1;i<datos_recibidos.length;i++){
                        NombreFichero = NombreFichero+" "+
                            datos_recibidos[i];
                    }
                    Fichero aux = new Fichero(datos_recibidos[0].
                        toString().trim(),NombreFichero.trim());
                    Ficheros.add(aux);
                }
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID221

```
String NombreFichero="";
datos_recibidos = mensaje.split(" ");
for(int i=1;i<datos_recibidos.length;i++){
    NombreFichero = NombreFichero+" "+
        datos_recibidos[i];
}
Fichero aux = new Fichero(datos_recibidos[0].
    toString().trim(),NombreFichero.trim());
Ficheros.add(aux);
}
cont++;
mensaje = (String) Conexion.entrada.readObject();
}

ListaFicheros = (GridView)findViewById(R.id.GridViewFicheros);
ListaFicheros.setAdapter(adaptador);
} catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
} catch (ClassNotFoundException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}

private void MostrarMiPc(){
try {
// Si es Linux
if(SesionLocal.ObtenerSistemaOperativo().equals("Linux"))
Ruta.setText("/");
Conexion.salida.flush();
Conexion.salida.writeObject("MostrarMiPc");
Ficheros.clear();

// Obtenemos los datos del servidor.
mensaje = (String) Conexion.entrada.readObject();
String [] datos;
while(!mensaje.equals("Fin_MostrarMiPc")){
    datos = mensaje.split(" ");
    Fichero aux = new Fichero(datos[1].toString().trim(),datos
        [0].toString().trim());
    Ficheros.add(aux);
    Log.i("Ficheros",aux.getIcono()+" "+aux.getNombre());
    mensaje = (String) Conexion.entrada.readObject();
}
// Establecemos el adaptador
adaptador = new AdaptadorFicheros(this);
ListaFicheros = (GridView)findViewById(R.id.GridViewFicheros);
;
ListaFicheros.setAdapter(adaptador);
} catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
} catch (ClassNotFoundException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}

// Adaptador del GridView
class AdaptadorFicheros extends ArrayAdapter {

    Activity context;
```

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID222

```
AdaptadorFicheros(Activity context) {
    super(context, R.layout.fichero, Ficheros);
    this.context = context;
}

public View getView(int position, View convertView, ViewGroup
parent)
{
    View item = convertView;
    ViewHolder holder;

    if(item == null)
    {
        LayoutInflator inflater = context.getLayoutInflater();
        item = inflater.inflate(R.layout.fichero, null);

        holder = new ViewHolder();
        holder.icono = (ImageView)item.findViewById(R.id.
ImagenFichero);
        holder.nombre = (TextView)item.findViewById(R.id.
TextoFichero);
        // Cambiamos el tamaño de la imagen del fichero.
        LinearLayout.LayoutParams layoutParams = new LinearLayout.
LayoutParams(40, 40);
        holder.icono.setLayoutParams(layoutParams);
        holder.nombre.setTextSize(11);

        item.setTag(holder);
    }
    else
    {
        holder = (ViewHolder)item.getTag();
    }

    holder.nombre.setText(Ficheros.elementAt(position).getNombre());
    // Procesamiento de iconos
    if(Ficheros.elementAt(position).getIcono().equals("Unidadfija"))
    ){
        holder.icono.setBackgroundResource(R.drawable.discoduro);
    }else{
        if(Ficheros.elementAt(position).getIcono().equals("CD-ROM"))
        {
            holder.icono.setBackgroundResource(R.drawable.cdrom);
        }else{
            if(Ficheros.elementAt(position).getIcono().equals("<DIR>")
|| Ficheros.elementAt(position).getIcono().
startsWith("d"))){
                holder.icono.setBackgroundResource(R.drawable.carpeta);
            };
        }else{
            if(Ficheros.elementAt(position).getNombre().endsWith(
".exe")){
                holder.icono.setBackgroundResource(R.drawable.
iconoexe);
            }else{
                if(Ficheros.elementAt(position).getNombre().
endsWith(".zip") || Ficheros.elementAt(
position).getNombre().endsWith(".rar")){
                    holder.icono.setBackgroundResource(R.drawable.
iconozip);
                }else{
                    if(Ficheros.elementAt(position).getNombre().
endsWith(".txt")){
                        holder.icono.setBackgroundResource(R.drawable.
.iconotxt);
                    }else{
                }
            }
        }
    }
}
```

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID223

```
if(Ficheros.elementAt(position).getNombre() .  
    endsWith(".jpg")){  
    holder.icono.setBackgroundDrawable(R.  
        drawable.icono_jpg);  
}  
else{  
    if(Ficheros.elementAt(position).getNombre()  
        () .endsWith(".mp3")){  
        holder.icono.setBackgroundDrawable(R.  
            drawable.iconomp3);  
    }  
    else{  
        if(Ficheros.elementAt(position) .  
            getNombre() .endsWith(".bat")){  
            holder.icono.setBackgroundDrawable(R.  
                drawable.icono_bat);  
        }  
        else{  
            if(Ficheros.elementAt(position) .  
                getNombre() .endsWith(".pdf")){  
                holder.icono .  
                    setBackgroundDrawable(R.  
                        drawable.iconopdf);  
            }  
            else{  
                if(Ficheros.elementAt(position) .  
                    getNombre() .endsWith(".dll")){  
                    holder.icono .  
                        setBackgroundDrawable(R.  
                            drawable.iconodll);  
                }  
                else{  
                    if(Ficheros.elementAt(position) .  
                        getNombre() .endsWith(".ini")){  
                        holder.icono .  
                            setBackgroundDrawable(   
                                R.drawable.iconoini);  
                    }  
                    else{  
                        if(Ficheros.elementAt(   
                            position) .getNombre() .  
                            endsWith(".png")){  
                            holder.icono .  
                                setBackgroundDrawable(   
                                    R.drawable.  
                                        iconopng);  
                        }  
                        else{  
                            if(Ficheros.elementAt(   
                                position) .getNombre()  
                                () .endsWith(".java" )){  
                                holder.icono .  
                                    setBackgroundDrawable(   
                                        R.drawable.  
                                            iconojava);  
                            }  
                            else{  
                                if(SesionLocal .  
                                    SistemaOperativo  
                                    .equals("Windows  
                                " ))  
                                holder.icono .  
                                    setBackgroundDrawable(   
                                        R.drawable.  
                                            iconoarchivowin  
                                    );  
                            }  
                            else  
                                holder.icono .  
                                    setBackgroundDrawable(   
                                        R.drawable.  
                                            iconolinux);  
                            }  
                        }  
                    }  
                }  
            }  
        }  
    }  
}
```



## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID225

```
Ficheros .get (pos) .getNombre () .endsWith (".png") ||  
Ficheros .get (pos) .getNombre () .endsWith (".pdf"))  
inflater .inflate (R.menu. menunavegadortransferir ,  
menu);  
else {  
    /* Demás ficheros (Cuya longitud de ruta sea mayor  
    de 0)  
    if (Ruta.length ()>0)  
        inflater .inflate (R.menu. menunavegador , menu);  
}  
}  
}  
  
// Tratamiento del menu contextual  
@Override  
public boolean onContextItemSelected (MenuItem item) {  
    switch (item.getItemId ()) {  
        // Si pulsamos sobre ejecutar.  
        case R.id. menunavegadorEjecutar:  
            Ejecutar ();  
            return true;  
  
        // Si pulsamos sobre Transferir a SD-Card  
        case R.id. menunavegadorTransferir:  
            TransferirFichero ();  
            return true;  
  
        // Si pulsamos sobre copiar  
        case R.id. menunavegadorCopiar:  
            IndicadorOpcionPegar=true;  
            IndicadorCopia=true;  
            IndicadorCorte=false;  
            if (SesionLocal. ObtenerSistemaOperativo () .equals ("Windows"))  
                FicheroOrigenPegado=Ruta.getText () .toString () +Ficheros .get  
(pos) .getNombre () .toString ();  
            else  
                FicheroOrigenPegado=Ruta.getText () .toString () +"/"+Ficheros .  
get (pos) .getNombre () .toString ();  
  
            return true;  
        // Si pulsamos sobre Cortar  
        case R.id. menunavegadorCortar:  
            IndicadorOpcionPegar=true;  
            IndicadorCorte=true;  
            IndicadorCopia=false;  
            if (SesionLocal. ObtenerSistemaOperativo () .equals ("Windows"))  
                FicheroOrigenPegado=Ruta.getText () .toString () +Ficheros .get  
(pos) .getNombre () .toString ();  
            else  
                FicheroOrigenPegado=Ruta.getText () .toString () +"/"+Ficheros .  
get (pos) .getNombre () .toString ();  
  
            return true;  
        // Si pulsamos sobre eliminar  
        case R.id. menunavegadorEliminar:  
            EliminarFichero ();  
            return true;  
        // Si pulsamos sobre renombrar  
        case R.id. menunavegadorRenombrar:  
            RenombrarFichero ();  
            return true;  
    default:  
        return super.onContextItemSelected (item);  
    }  
}  
  
// Renombrar Fichero
```

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID226

```
private void RenombrarFichero(){
    final AlertDialog.Builder alert = new AlertDialog.Builder(this);
    alert.setMessage("Nuevo nombre para el fichero " + Ficheros.get(
        pos).getNombre());
    final EditText input = new EditText(this);
    alert.setView(input);
    alert.setPositiveButton("Renombrar", new DialogInterface.
        OnClickListener() {
            public void onClick(DialogInterface dialog, int whichButton)
            {
                try {
                    Enviar("RenombrarFichero");
                    Enviar(Ruta.getText().toString());
                    Enviar(Ficheros.get(pos).getNombre().toString());
                    Enviar(input.getText().toString());
                    mensaje = (String) Conexion.entrada.readObject();
                    Toast toast1 = Toast.makeText(getApplicationContext(),
                        mensaje, Toast.LENGTHLONG);
                    toast1.show();
                } catch (OptionalDataException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                } catch (ClassNotFoundException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                } finally{
                    ActualizarGridViewAnterior();
                }
            }
        });
    alert.setNegativeButton("Cancelar",
        new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int
                whichButton) {
                    dialog.cancel();
                }
        });
    alert.show();
}

// EliminarFichero
private void EliminarFichero(){
    // Confirmacion
    final AlertDialog.Builder alert = new AlertDialog.Builder(this);
    alert.setMessage("Seguro que quieres eliminar el fichero " +
        Ficheros.get(pos).getNombre());
    alert.setPositiveButton("Eliminar", new DialogInterface.
        OnClickListener() {
            public void onClick(DialogInterface dialog, int whichButton)
            {
                try {
                    // Enviar datos al Servidor.
                    Enviar("EliminarFichero");
                    if(SesionLocal.ObtenerSistemaOperativo().equals("Windows"))
                        Enviar(Ruta.getText().toString()+Ficheros.get(pos).
                            getNombre().toString());
                    else
                        Enviar(Ruta.getText().toString() + "/" + Ficheros.get(
                            pos).getNombre().toString());
                    mensaje = (String) Conexion.entrada.readObject();
                }
            }
        });
}
```

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID227

```
        Toast toast1 = Toast.makeText(getApplicationContext() ,  
                mensaje , Toast.LENGTHLONG);  
        toast1.show();  
    } catch (OptionalDataException e) {  
        // TODO Auto-generated catch block  
        e.printStackTrace();  
    } catch (ClassNotFoundException e) {  
        // TODO Auto-generated catch block  
        e.printStackTrace();  
    } catch (IOException e) {  
        // TODO Auto-generated catch block  
        e.printStackTrace();  
    } finally{  
        // Actualizamos GridView  
        ActualizarGridViewAnterior();  
    }  
}  
}  
});  
  
alert.setNegativeButton("Cancelar",  
        new DialogInterface.OnClickListener() {  
            public void onClick(DialogInterface dialog, int  
                    whichButton) {  
                dialog.cancel();  
            }  
        } );  
alert.show();  
}  
  
// TransferirFichero => hilo que intenta conectar a un nuevo puerto  
abierto (1111)  
private class Transferirfichero implements Runnable{  
    private String ip;  
    private String nombre;  
    private int Puerto;  
  
    Transferirfichero(String i, String n, int puerto){  
        ip=i;  
        nombre=n;  
        Puerto=puerto;  
        Thread t=new Thread (this);  
        t.start();  
    }  
  
    public void run() {  
        // TODO Auto-generated method stub  
        try {  
            conexiontrans = new Socket(ip ,Puerto);  
            File folder = new File(Environment.  
                    getExternalStorageDirectory() + "/RemSys/");  
  
            // Si la carpeta no existe , se crea  
            if (!folder.exists()) {  
                folder.mkdir();  
            }  
  
            // Abrimos fichero para escritura y mediante los flujos  
            // adecuados escribimos lo que nos venga de el  
            File f = new File(folder.getAbsolutePath() , nombre);  
            byte[] b = new byte[1024];  
            int len = 0;  
            int bytcount = 1024;  
            FileOutputStream inFile = new FileOutputStream(f);  
            InputStream is = conexiontrans.getInputStream();  
            BufferedInputStream in2 = new BufferedInputStream(is , 1024);  
        }  
    }  
}
```

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID228

```
        while ((len = in2.read(b, 0, 1024)) != -1) {
            bytcount = bytcount + 1024;
            inFile.write(b, 0, len);
        }
        // Cerramos flujos y socket
        in2.close();
        inFile.close();
        conexiontrans.close();
    } catch (FileNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (OptionalDataException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

private void TransferirFichero(){
    // Comunicamos al servidor que queremos la transferencia de un
    // fichero y enviamos datos de localización
    Enviar("TransferirFichero");
    Enviar(Ruta.getText().toString());
    Enviar(Ficheros.get(pos).getNombre().toString());

    try {
        // Obtenemos mensaje de sincronizacion, en este momento el
        // servidor espera una conexion por el puerto indicado
        mensaje = (String) Conexion.entrada.readObject();
        String [] datos = mensaje.split(":");
        // Creamos el hilo que va a servir para la transferencia.
        if(datos[0].equals("ListoTransferencia")){
            Log.i("TransferirFichero", "new Transferirfichero(" +
                Conexion.ObtenerIP() + "," + Ficheros.get(pos).
                getNombre().toString() + "," + Integer.parseInt(datos
                [1]) + ")");
            new Transferirfichero(Conexion.ObtenerIP(),Ficheros.get(pos)
                .getNombre().toString(),Integer.parseInt(datos[1]));
        }
    } catch (OptionalDataException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (ClassNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

    Toast toast1 = Toast.makeText(getApplicationContext(),"
        Transferencia realizada", Toast.LENGTHLONG);
    toast1.show();
}

// Ejecutar un fichero con argumentos.
private void Ejecutar(){
    final AlertDialog.Builder alert = new AlertDialog.Builder(this);
    alert.setMessage("Introduce los parámetros para la ejecucion del
        fichero " + Ficheros.get(pos).getNombre());
    final EditText input = new EditText(this);
    alert.setView(input);
```

```

        alert.setPositiveButton("Ejecutar", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int whichButton) {
                argumentos = input.getText().toString().trim();
                EjecutarFichero();
            }
        });

        alert.setNegativeButton("Cancelar",
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int
                            whichButton) {
                        dialog.cancel();
                    }
                });
        alert.show();
    }

private void EjecutarFichero(){
    // Establece comunicacion y envia datos para la ejecución del
    // fichero.
    Enviar("EjecutarFichero");
    Enviar(Ruta.getText().toString());
    Enviar(Ficheros.get(pos).getNombre().toString());
    Enviar(argumentos);
}

private void Enviar(String n){
    try{
        Conexion.salida.flush();
        Conexion.salida.writeObject(n);

        }catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
}

// Boton menu del teléfono para la opcion de pegado de ficheros.
@Override
public boolean onPrepareOptionsMenu(Menu menu) {
    menu.clear();

    if(IndicadorOpcionPegar && Ruta.getText().toString().length()>0) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.menupegar, menu);
    }
    return super.onPrepareOptionsMenu(menu);
}

// Procesamiento de pegado de fichero, distinguiendo si se ha
// copiado o cortado.
private void PegarFichero(){
    if(IndicadorCopia){
        Enviar("Copia");
    }else{
        Enviar("Corte");
    }
    Enviar(FicheroOrigenPegado);
    Enviar(Ruta.getText().toString());
    try {
        String mensaje = (String) Conexion.entrada.readObject();
        Toast toast1 = Toast.makeText(getApplicationContext(),mensaje,
                Toast.LENGTHLONG);
    }
}

```

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID230

```
        toast1.show();
    } catch (OptionalDataException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (ClassNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } finally{
        // Actualizamos Gridview y indicadores.
        ActualizarGridViewAnterior();
        if(IndicadorCorte)
            IndicadorOpcionPegar=IndicadorCorte=false;
    }

    // Seleccion de la opcion de pegado.
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.menuPegar:
                PegarFichero();
        }
        return true;
    }
}
```

El código es largo y complejo, aunque pasare a comentar las principales características de dicho código:

La primera parte comprueba tanto si se viene de crear un fichero. En caso de que sea así, se tendría que actualizar el GridView. Si estamos ante una ruta no válida para crear un fichero (MiPc) no se nos da esa opción. A continuación, dependiendo del sistema operativo del servidor, se establece la ruta, y se actualiza el contenido del GridView: en el caso de Windows, se mostrarán las unidades de disco, en el caso de Linux, se mostrará el contenido del directorio raíz. Posteriormente definimos los eventos “setOnItemClickListener” y “setOnItemLongClickListener”, los cuales corresponden a un click y una pulsación larga. El primero de ellos, hará que se actualice la ruta y el GridView con el contenido del directorio pulsado. El segundo de ellos, hará que aparezca un determinado menú contextual. Dichos menús contextuales están descritos por ficheros .xml y dependiendo del tipo de fichero (extensión), aparecerá un menú u otro. Por ejemplo para un fichero .java, tendremos que nos aparecerá un menú con ejecución del fichero (que realmente hace una compilación y ejecución), mientras que por ejemplo un simple .txt no tendrá esta opción. Ésto se controla mediante la función “onCreateContextMenu”.

Una funcionalidad más a destacar, sería la de las flechas de navegación. Son botones los cuales tiene una particularidad y es que se debe de guardar las rutas, al hacer click en la flecha atrás, para que, posteriormente, al hacer click en su flecha complementaria, tenga acceso a las rutas en las cuales ha pasado el cliente. Para ello se utiliza un vector “RutaPrevias” que utiliza la función “RutaAnteriorGuardado()” de la cual se hace uso al pulsar la flecha atrás. A continuación vamos a ver algunas de las pantallas pertenecientes a dicho navegador:



Figura 8.14: Ejemplos de distintos menús contextuales

- Los ficheros “.exe”, “.bat”, “.java”, “.sh”, “.run”, “.py” y “.bin” tendrán las opciones comunes además de poderse ejecutar.
- Los ficheros “.txt”, “.jpg”, “.png” y “.pdf” se podrán transferir a la tarjeta SD del dispositivo móvil, además de las opciones comunes.

A continuación vamos a ver la opción más compleja que podemos tener en el sistema, copiar/cortar y pegar.

#### 8.1.6.1. Cortar/Copiar Pegar Fichero

Para cortar o copiar un fichero procederemos a establecer una pulsación larga sobre él. En el menú contextual, elegiremos si queremos cortar o copiar y posteriormente nos situaremos en la ruta donde queramos pegar el fichero. Para que aparezca la opción de pegar fichero, tendremos que tener un fichero previamente cortado o copiado y estando situado en un ruta válida, pulsar el botón menú del dispositivo móvil. A continuación se presentan algunas imágenes:

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID232



Figura 8.15: Proceso Cortar/Copiar y Pegar

En la imagen podemos ver el proceso a seguir para copiar o cortar y pegar el fichero key de la carpeta D:\ a la ruta o directorio D:\Proyecto\.

### 8.1.7. Orden Libre

El layout será el siguiente:

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID233



Figura 8.16: Layout “ordenlibre.xml”

Podemos observar como consta principalmente de:

- EditText: Para introducción de la orden a ejecutar.
- Boton “Ejecutar”: Para dar la orden para que se envíe a ejecución.
- TextView: Para la obtención de la salida de la orden.

Su código de comportamiento es el siguiente:

---

### OrdenLibre.java

---

```
package garciaponce.miguel;  
  
import java.io.IOException;  
  
import android.app.Activity;
```

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID234

```
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class OrdenLibre extends Activity {
    private TextView SalidaOrden;
    private EditText Orden;
    private Button BtnEjecutar;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.ordenlibre);
        Conexion.ACTIVIDAD=this;

        Orden = (EditText) findViewById(R.id.Orden);
        BtnEjecutar = (Button) findViewById(R.id.BotonEjecutar);

        BtnEjecutar.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                ObtenerSalida();
            }
        });
    }

    private void ObtenerSalida(){
        SalidaOrden = (TextView) findViewById(R.id.SalidaOrden);

        if(SalidaOrden.getText().length()>0)
            SalidaOrden.setText("");

        try{
            Conexion.salida.flush();
            Conexion.salida.writeObject("OrdenLibre");
            Conexion.salida.flush();
            Conexion.salida.writeObject(Orden.getText().toString().trim());

            String mensaje = (String) Conexion.entrada.readObject();
            ;
            while(!mensaje.equals("Fin_OordenLibre")){
                SalidaOrden.append(mensaje + "\n");
                mensaje = (String) Conexion.entrada.readObject();
            }
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

```
}
```

La que hace el código es mandar la palabra “OrdenLibre” al servidor y posteriormente el contenido del TextView con id “Orden”. El servidor ejecutará la orden y mandará la salida. El cliente leerá y mostrará en el TextView la salida, hasta que el servidor mande “Fin\_OrdenLibre”.

### 8.1.8. Scripts

Para la implementación de la funcionalidad Scripts, tenemos varios layouts, los cuales corresponderán a:

- Layout Principal: Donde se presentarán mediante un ListView la lista de los Scripts contenidos en la carpeta del servidor “Scripts”.
- Layout “Ver Script”: Donde se visualiza el contenido de un Script. Tendrá la parte del título donde se agregarán el nombre del script y otra parte el contenido del mismo.
- Layout “Transferir Script”: Encontraremos un EditText para establecer su contenido. Además tendrá un botón para la aceptación de los mismos, el cual hará aparecer un recuadro para introducir el nombre del mismo y mandar la orden para crear dicho script con su contenido.

En la siguiente figura podemos ver los tipos de layouts:



Figura 8.17: Layout “scripts” y “verscript”

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID236

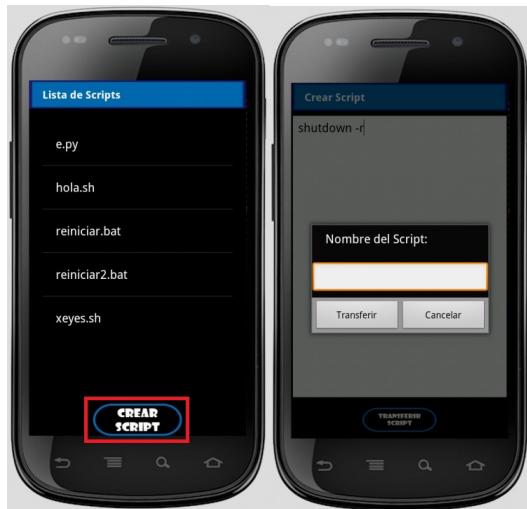


Figura 8.18: Layout “scripts” y “crearscript”

También tendremos la opción de “Eliminar Script”, el cual tomará el nombre del script y enviará al servidor tanto la orden como dicho nombre para ejecutar la orden que realice dicha acción. Los códigos que describen el comportamiento de los distintos layouts son:

- Layout “scripts”:

Scripts.java

```
package garciaponce.miguel;

import java.io.IOException;
import java.util.Vector;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.ContextMenu;
import android.view.ContextMenu.ContextMenuItemInfo;
import android.view.KeyEvent;
import android.view.LayoutInflater;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemLongClickListener;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.widget.TextView;

public class Scripts extends Activity {
    private Vector<String> Scripts;
```

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID237

```
private ListView lstScripts;
private int pos;
private Button CrearScript;
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.scripts);
    Conexion.ACTIVIDAD=this;

    Scripts = new Vector<String>();

    try {
        // Enviamos el mensaje Navegacion para obtener los Scripts del
        // directorio
        Conexion.salida.flush();
        Conexion.salida.writeObject("Scripts");

        String mensaje = (String) Conexion.entrada.readObject();
        while(!mensaje.equals("Fin_Scripts")){
            Scripts.add(mensaje);
            mensaje = (String) Conexion.entrada.readObject();
        }

        // Establecemos el adaptador y registramos menu contextual
        AdaptadorScripts adaptador = new AdaptadorScripts(this);
        lstScripts = (ListView)findViewById(R.id.ListaScripts);
        lstScripts.setAdapter(adaptador);
        registerForContextMenu(lstScripts);

        // Registramos posicion del elemento seleccionado
        lstScripts.setOnItemLongClickListener(new
            OnItemLongClickListener() {
                public boolean onItemLongClick(AdapterView<?> parent,
                    final View v, int position, long id) {
                    // record position/id/whatever here
                    pos=position;
                    return false;
                }
            });
    }

    // Boton CrearScript => nuevo layout
    CrearScript = (Button) findViewById(R.id.CrearScript);
    CrearScript.setOnClickListener(new OnClickListener() {
        public void onClick(View v) {
            Intent intent = new Intent(Scripts.this , CrearScript.
                getClass());
            startActivity(intent);
        }
    });
}

} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ClassNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (NullPointerException e){
    // TODO Auto-generated catch block
    e.printStackTrace();
}
}

@Override
public void onCreateContextMenu(ContextMenu menu, View v,
    ContextMenuItemInfo menuInfo)
```

```

{
    super.onCreateContextMenu(menu, v, menuInfo);
    MenuInflater inflater = getMenuInflater();
    if(v.getId() == R.id.ListaScripts){
        // Segun sea el sistema, obtenemos los menus contextuales de
        // los scripts a ejecutar
        if(SesionLocal.ObtenerSistemaOperativo().equals("Windows")){
            if(Scripts.get(pos).toString().endsWith(".exe") || Scripts.
                get(pos).toString().endsWith(".bat") || Scripts.get(pos).
                toString().endsWith(".cmd") || Scripts.get(pos).
                toString().endsWith(".py"))
                inflater.inflate(R.menu.menuscripts, menu);
        } else{
            if(Scripts.get(pos).toString().endsWith(".sh") || Scripts.
                get(pos).toString().endsWith(".py") || Scripts.get(pos).
                toString().endsWith(".run") || Scripts.get(pos).
                toString().endsWith(".bin"))
                inflater.inflate(R.menu.menuscripts, menu);
        }
    }
    // Menu contextual (Ejecutar, eliminar y ver script)
    public boolean onContextItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            // Si pulsamos sobre ejecutar.
            case R.id.menuscript1:
                EjecutarScript();
                return true;
            // Si pulsamos sobre Transferir a SD-Card
            case R.id.menuscript2:
                EliminarScript();
                return true;
            case R.id.menuscript3:
                VerScript();
                return true;
            default:
                return super.onContextItemSelected(item);
        }
    }

    private void VerScript(){
        Intent intent = new Intent(Scripts.this, VerScript.class);
        intent.putExtra("Fichero", Scripts.get(pos).toString());
        startActivity(intent);
    }

    private void EjecutarScript(){
        try {
            Conexion.salida.flush();
            Conexion.salida.writeObject("EjecutarScript " + Scripts.get(
                pos).toString());
            Intent intent = new Intent(Scripts.this, Scripts.class);
            startActivity(intent);
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}

```

```

}

// Boton BACK del teléfono
@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    if ((keyCode == KeyEvent.KEYCODEBACK)) {
        try {
            Conexion.salida.flush();
            Conexion.salida.writeObject("SalirScripts");
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        Intent intent = new Intent(Scripts.this, Mainmenu.class);
        startActivity(intent);
        return super.onKeyDown(keyCode, event);
    }

    private void EliminarScript() {
        try {
            Conexion.salida.flush();
            Conexion.salida.writeObject("EliminarScript " + Scripts.get(pos)
                .toString());

            String mensaje = (String) Conexion.entrada.readObject();
            if(mensaje.equals("FinEliminarScript")){
                Intent intent = new Intent(Scripts.this, Scripts.class);
                startActivity(intent);
            }
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    class AdaptadorScripts extends ArrayAdapter {
        Activity context;

        AdaptadorScripts(Activity context) {
            super(context, R.layout.script, Scripts);
            this.context = context;
        }

        public View getView(int position, View convertView, ViewGroup
            parent)
        {
            View item = convertView;
            ViewHolder holder;

            if(item == null)
            {
                LayoutInflator inflater = context.getLayoutInflater();
                item = inflater.inflate(R.layout.script, null);

                holder = new ViewHolder();
                holder.Nombre = (TextView)item.findViewById(R.id.
                    NombreScript);

                item.setTag(holder);
            }
        }
    }
}

```

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID240

```
        }
    else
    {
        holder = (ViewHolder)item.getTag();
    }

    holder.Nombre.setText(Scripts.get(position).toString());
    return(item);
}

static class ViewHolder {
    public TextView Nombre;
}
}
```

Podemos observar como dependiendo de la extensión de los ficheros de la carpeta Script y del sistema operativo, los va a permitir ejecutar o no. Igualmente tenemos descritas las opciones que va a tener al hacer una pulsación larga sobre cualquiera de las scripts válidos para cada sistema operativo. La lista de scripts queda presentada en un componente ListView.

- Layout “verscript”:

VerScript.java

---

```
package garciaponce.miguel;

import java.io.IOException;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.KeyEvent;
import android.widget.EditText;
import android.widget.TextView;

public class VerScript extends Activity {
    private EditText TextoVerScript;
    private TextView Titulo;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.verscript);
        Conexion.ACTIVIDAD=this;

        Intent intent = getIntent();
        String fichero = intent.getStringExtra("Fichero");

        TextoVerScript = (EditText) findViewById(R.id.TextoVerScript);
        Titulo = (TextView) findViewById(R.id.TituloVerScript);
        Titulo.append(": " + fichero);
        try {
            Conexion.salida.flush();
            Conexion.salida.writeObject("VerScript");
            Conexion.salida.flush();
            Conexion.salida.writeObject(fichero);

            String mensaje = (String) Conexion.entrada.readObject();

            while(!mensaje.equals("FinLecturaScript")){
                TextoVerScript.append(mensaje + "\n");
                mensaje = (String) Conexion.entrada.readObject();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID241

```
        }

    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (ClassNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    TextoVerScript.setOnKeyListener(null);
}

@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    if ((keyCode == KeyEvent.KEYCODE_BACK)) {
        Intent intent = new Intent(VerScript.this, Scripts.class);
        startActivity(intent);
    }
    return super.onKeyDown(keyCode, event);
}
}
```

El código lo que hace es mandar la orden VerScript junto con el nombre del script a visualizar. En el servidor se leerá el script, enviándose el contenido al cliente. Este lo leerá hasta que reciba “FinLecturaScript” e irá mostrandolo en el EditText llamado TextoVerScript.

- Layout “crearscript”:

CrearScript.java

---

```
package garciaponce.miguel;

import java.io.IOException;
import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;

public class CrearScript extends Activity {

    private Button TransferirScript;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.crearscript);
        Conexion.ACTIVIDAD=this;

        // Si pulsamos sobre transferirScript
        TransferirScript = (Button) findViewById(R.id.TransferirScript)
        ;
        TransferirScript.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                // Pedimos el nombre del script
                final AlertDialog.Builder alert = new AlertDialog.Builder(
                        CrearScript.this);
                alert.setMessage("Nombre del Script: ");

```

## CAPÍTULO 8. IMPLEMENTACIÓN DEL CLIENTE USANDO ANDROID242

```
final EditText input = new EditText(CrearScript.this);
alert.setView(input);
// Si pulsamos sobre Transferir con los datos
// introducidos.
alert.setPositiveButton("Transferir", new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int whichButton) {
        // Mandamos datos al servidor.
        try {
            EditText TextoScript = (EditText) findViewById(R.id.TextoScript);
            Conexion.salida.flush();
            Conexion.salida.writeObject("TransferirScript");
            Conexion.salida.flush();
            Conexion.salida.writeObject(input.getText().toString());
            Conexion.salida.flush();
            Conexion.salida.writeObject(TextoScript.getText().toString());
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } finally{
            // Al acabar, volvemos al layout Scripts
            Intent intent = new Intent(CrearScript.this,
                Scripts.class);
            startActivity(intent);
        }
    }
});
alert.setNegativeButton("Cancelar",
new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog,
        int whichButton) {
        dialog.cancel();
    }
});
alert.show();
}
}
}
```

El código lo único que hace es que cuando se pulsa en el botón con nombre TransferirScript, se abre una ventana de diálogo donde se pide el nombre del script a transferir. Junto con el EditText con id TextoScript se envía al servidor el cual se encargará con estos datos de la creación del script correspondiente con su nombre y contenido especificados desde el cliente.

# Capítulo 9

## Pruebas

Un plan de pruebas es fundamental para el desarrollo de un proyecto software. El cliente interacciona con el servidor y este a su vez con el sistema operativo. El plan de pruebas consiste en hacer un número suficiente de pruebas para verificar que el software hace lo que debe hacer, llevándolo a situaciones críticas donde podría fallar.

Para la mayoría de las funcionalidades, tenemos en el servidor una opción para verificar las salidas de dichas funciones, estando contenidas en el menú “Pruebas”. Podemos ver el acceso a dichas pruebas en la siguiente imagen:



Figura 9.1: Menú pruebas en el servidor Remsys

Por lo tanto las pruebas funcionales que podemos obtener directamente a través del servidor es:

- Discos.
- Red.
- Sistema Operativo.
- Memoria.
- Procesos.
- Orden Libre.
- Scripts.

Dichas pruebas se han realizado tanto con el sistema Windows como en Linux, a través del simulador proporcionado con la plataforma eclipse (Android SDK), como con un dispositivo móvil físico. También pruebas realizadas con ip pública e ip local. Además, las pruebas que realizamos al software son las siguientes:

- De contenido: Nos permiten detectar errores en la información enviada al cliente para su posterior visualización y permite asegurarnos que se recibe la información correcta.
- De interfaz: Estas pruebas consisten en que en cada momento y, de acuerdo a las distintas acciones que realice el cliente, se modifique la interfaz de acuerdo a dicha acción. Por ejemplo, tenemos el caso en el cual, al iniciar la navegación por el árbol de directorios, tenemos un botón, “crear fichero”, el cual si está en una ruta que no sea válida, por ejemplo, mostrando “MiPc”, el botón no estaría activo y su acción de hacer click no tendría consecuencias.
- De navegación: Pruebas consistentes en la navegación de la aplicación de cliente, comprobando que podemos acceder a todos los layouts y cerciorandonos que se mandan las órdenes para su correcta navegación. Por ejemplo, la comprobación de que cuando accedemos al menu principal del dispositivo móvil, previamente, en el layout anterior se ha iniciado un servicio de conexión. En el layout menu principal, se comprueba esa conexión y si hacemos click en el botón BACK del dispositivo móvil, dicho servicio es detenido, volviéndose al layout listahosts, pudiéndose repetir el proceso de conexión a otro host cualquiera.

### 9.1. Pruebas realizadas

- Validación del código Remsys del servidor: Se comprueba que el código escrito bajo Java es correcto y no contenga fallos, que está todo bien enlazado y no hay fallos en sus trazas. Se corrigen errores, borrado de variables no usadas, importaciones de paquetes no usados, simplificación de código...

- Validación del código Remsys del cliente: Se comprueba que el código escrito en Java es correcto y no contenga fallos. Se obtiene la navegación válida y se producen las acciones correspondientes a interacciones del cliente concretas. Además se comprueban que se obtienen todos los permisos que hacen que la aplicación funcione correctamente (fichero AndroidManifest.xml). Se corrijen errores, borrado de variables no usadas, importaciones de paquetes no usados, simplificación de código...)
- Pruebas de interfaz con el sistema operativo: Se comprueba que el servidor, que es el que interacciona con el sistema operativo, manda las órdenes correctas para obtener la información que queremos. Se visualiza por la pantalla las órdenes que se mandan a ejecutar al sistema operativo para comprobar que son válidas...
- Pruebas de seguridad: Se comprueba la integridad del sistema, dando lugar a un resultado exitoso.

# Capítulo 10

## Manuales de usuario

En este capítulo veremos como se instalan y se acceden a las funcionalidades de Remsys.

### 10.1. Instalación del cliente

Para la instalación del cliente, necesitaremos un espacio libre de aproximadamente 1Mb. Para la instalación de RemSys, lo primero que debemos de hacer es transferir el fichero “RemSys.apk” al teléfono móvil.

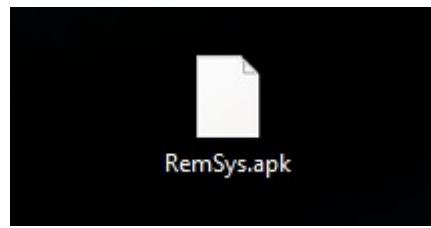


Figura 10.1: Fichero “RemSys.apk”

Posteriormente debemos de localizarlo mediante, por ejemplo un navegador en el dispositivo móvil abrirlo, dando una pantalla como sigue:

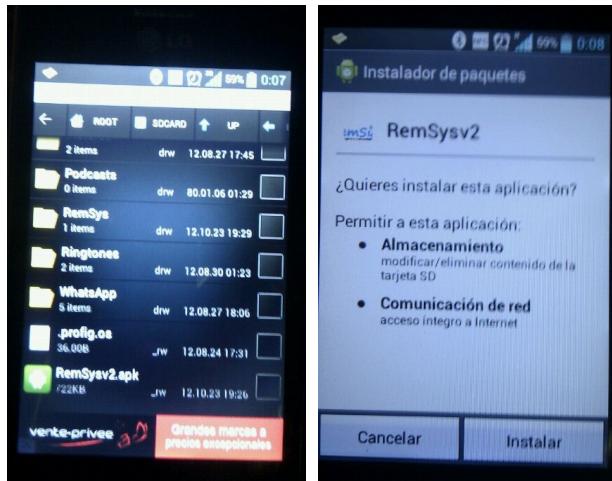


Figura 10.2: Instalación RemSys cliente

En esta pantalla de instalación se nos avisa de que el programa a instalar, RemSys, tendrá acceso íntegro a internet y acceso a la SD-Card del dispositivo móvil. Una vez aceptadas las condiciones de instalación, se procederá a su instalación en el dispositivo.

## 10.2. Autenticación y conexión a un host

Primeramente, lo que tenemos que hacer es logearnos en el sistema. Para ello, cuando nos aparezca la primera pantalla de Remsys, tendremos que pulsar sobre “Nuevo”. Posteriormente tendremos que llenar el formulario con los datos de nombre de usuario, contraseña y repetición de la contraseña. Finalmente solo tendremos que introducir las credenciales facilitadas anteriormente para acceder al sistema.



Figura 10.3: Acceso al sistema

- Crear Usuario: “Nuevo” -> Rellenar Formulario.
- Acceso al sistema: Completar Credenciales -> “Entrar”

Para conectar a un host, al acceder al sistema siguiendo los pasos anteriores, tendremos una pantalla como la que sigue:



Figura 10.4: Lista hosts

Si no tenemos ningún host creado al que acceder, se pulsará en el botón “Nuevo”, accediéndose a una pantalla como la que sigue:



Figura 10.5: Creación de un host

donde se tendrá que completar el formulario. El dato de MAC se podrá omitir, aunque es factible para realizar directamente el encendido remoto.

Una vez creado y ya en la pantalla de la lista de host, realizaremos una pulsación larga sobre el host a conectar, apareciéndonos las funciones de “Conectar”, “Eliminar” y “Encender”.



Figura 10.6: Funciones desde la lista de hosts

- Opción “Conectar” : Se accede al menú principal del host remoto.
- Opción “Eliminar” : Elimina el host seleccionado de la lista de host.
- Opción “Encender” : Se accede a la pantalla para mandar la orden necesaria para encender un host remotamente.

### 10.3. Acceso a funcionalidades del sistema desde el dispositivo móvil Android

#### 10.3.1. Encendido Remoto

Para acceder a las funcionalidades del sistema, desde la lista de hosts, tenemos la primera de ellas que es la de “Encender”, la cual dará paso a una pantalla como la siguiente:



Figura 10.7: Encendido Remoto

El formulario se autorellenará al acceder a esta pantalla con los datos que tenemos almacenados en la lista de hosts. Se tendrá que dar una IP de difusión o broadcast y la dirección MAC. Posteriormente se pulsará sobre el botón “Encender” para mandar la orden al host remoto.

### 10.3.2. Información del sistema

Para acceder a la información del sistema, tendremos que conectarnos a un host, accediendo a un menú principal como la que sigue y pulsando sobre el botón “Información del sistema” el cual dará acceso al siguiente menú mostrado:



Figura 10.8: Menú principal y menu “informacion del sistema”

Pulsando sobre cada uno de los botones se accederá a la funcionalidad correspondiente. En la siguiente imagen podemos ver el acceso a ellas:



Figura 10.9: Funcionalidades desde “Información del sistema”

### 10.3.3. Procesos

Para acceder a la información los procesos en ejecución, tendremos que conectarnos a un host, accediendo a un menú principal como la que sigue y pulsando sobre el botón “Procesos” el cual dará acceso al siguiente menú mostrado:



Figura 10.10: Menú principal y acceso a “Procesos”

Al realizar una pulsación larga sobre un determinado proceso, accederemos a la funcionalidad “Eliminar Proceso”.

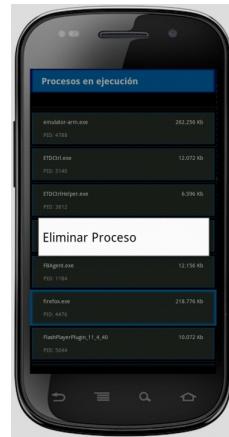


Figura 10.11: Funcionalidad “Eliminar Proceso”

#### 10.3.4. Navegación

Para acceder a la navegación por el árbol de directorios, tendremos que conectarnos a un host, accediendo a un menú principal como la que sigue y pulsando sobre el botón “Navegación” el cual dará acceso al siguiente menú mostrado:



Figura 10.12: Menú principal y acceso a “Navegación”

Una vez ahí, podremos navegar por el árbol de directorios pulsando sobre los iconos de los ficheros. Las funcionalidades de la parte de navegación podemos obtenerlas mediante una pulsación larga en los ficheros de la navegación, y son las siguientes:

- Eliminar fichero
- Cortar fichero
- Copiar fichero
- Renombrar fichero
- Ejecutar fichero
- Transferir fichero a SD-Card
- Crear fichero

#### 10.3.4.1. Eliminar y Renombrar fichero

Para eliminar y renombrar un fichero, accederemos a la navegación por el árbol de directorios anteriormente descrita, y pulsaremos mediante una pulsación larga el fichero que queramos eliminar o renombrar. Nos saldrá un menú contextual en donde elegiremos la opción que queramos. En el caso de renombrar, se nos pedirá un nuevo nombre para el fichero. Por otro lado, si elegimos eliminar, nos saldrá una ventana de confirmación.

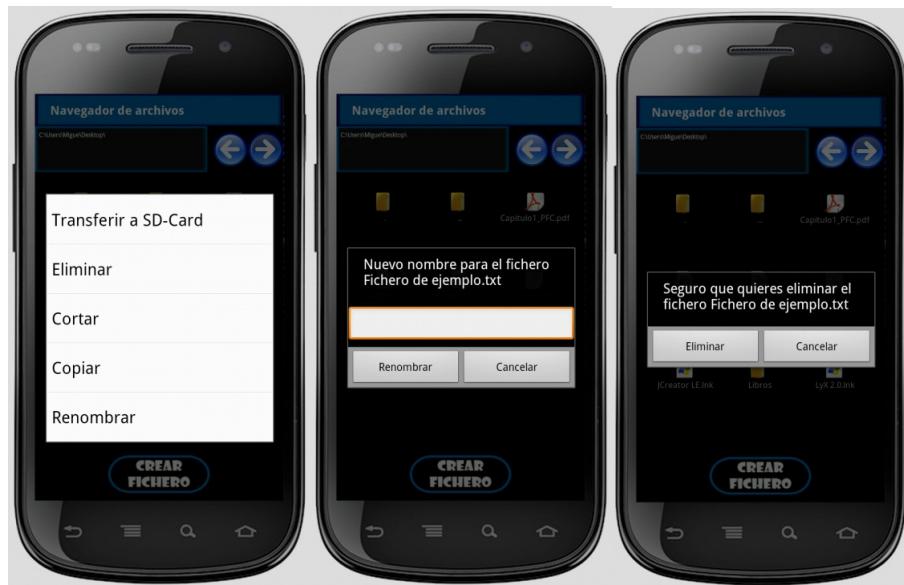


Figura 10.13: Funcionalidad “Eliminar fichero” y “Renombrar fichero”

#### 10.3.4.2. Ejecutar fichero

Se podrá ejecutar y en el caso de ficheros .java, compilar y ejecutar ficheros con extensiones : “.exe”, “.bat”, “.java”, “.sh”, “.run”, “.py” y “.bin” tendrán las opciones comunes además de poderse ejecutar. Una vez elegida la orden de ejecutar, se pedirá la introducción de argumentos adicionales para la ejecución del programa.



Figura 10.14: Funcionalidad “Ejecutar fichero”

#### 10.3.4.3. Cortar-Copiar y Pegar fichero

Para poder realizar la funcionalidad de cortar o copiar fichero, accederemos como anteriormente al menú contextual mediante una pulsación larga sobre el y elegiremos la opción que deseemos. Posteriormete, pulsaremos el botón menú del dispositivo móvil , donde nos aparecerá la opción para pegar, habiéndonos situado anteriormente en la ruta donde queremos pegar el fichero.



Figura 10.15: Funcionalidad “Copiar/Cortar y Pegar fichero”

#### 10.3.4.4. Transferir fichero a SD-Card

Los ficheros “.txt”, “.jpg”, “.png”, “.mp3” y “.pdf” se podrán transferir a la tarjeta SD del dispositivo móvil, además de las opciones comunes. El usuario tendrá que tener en cuenta que el tamaño del fichero a transferir, puede ser grande y de ahí, mayor tiempo de transferencia. Ahora vemos un ejemplo de la información que se obtiene al transferir un fichero:

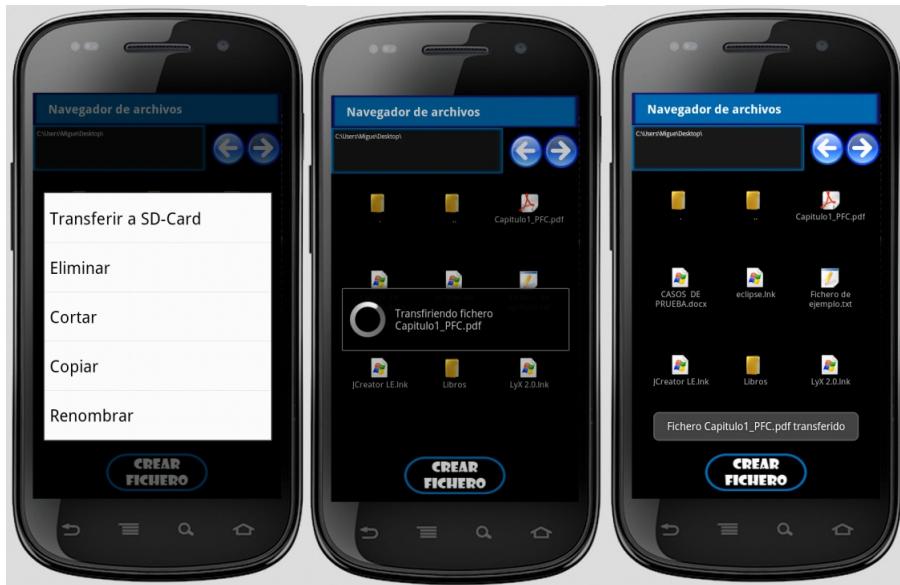


Figura 10.16: Funcionalidad “Transferir fichero”

#### 10.3.4.5. Crear fichero

Para crear un fichero, seguiremos los siguientes pasos:

1. Conectarse al hosts remoto.
2. Acceder a la Navegación por el árbol de directorios.
3. Situarse en la ruta donde se quiera crear el fichero.
4. Pulsar sobre el botón “Crear fichero”.
5. Rellenar el formulario con el nombre y contenido del fichero.
6. Pulsar el botón “Crear fichero” en la pantalla para aceptar tanto el nombre como el contenido y mandar la orden.

En el ejemplo vemos como se ha creado el fichero “ficheroejemplo.txt”.

#### 10.3.5. Orden Libre

Para acceder a la funcionalidad “Orden libre”, nos tendremos que conectar al host remoto y una vez en el menú principal, pulsar sobre el botón “Orden Libre”



Figura 10.17: Funcionalidad “Crear fichero”



Figura 10.18: Menú principal y acceso a “Orden Libre”

Para realizar esta funcionalidad, debe de completar el recuadro de “Introduzca orden a procesar”, con la orden que quiera ejecutar. Posteriormente se pulsará sobre el botón “Ejecutar”. La salida aparecerá en el recuadro inferior, con el resultado de la orden.

### 10.3.6. Scripts

Accedemos a la funcionalidad “Scripts”, pulsando sobre el botón con dicho nombre en el menú principal. Una vez allí, se visualizará una lista con los scripts contenidos en la carpeta de la aplicación servidor del host remoto.



Figura 10.19: Menú principal y acceso a “Scripts” y funcionalidad “Ver Script”

Una vez presentada la lista de scripts, se podrán realizar las siguientes acciones:

- Ver Script: El cual se accederá al realizar una pulsación larga sobre el script a ver y una vez en el menú contextual, pulsando en “Ver Script”.
- Eliminar Scripts: El cual se accederá al realizar una pulsación larga sobre el script a ver y una vez en el menú contextual, pulsando en “Eliminar Script”.
- Ejecutar Scripts: El cual se accederá al realizar una pulsación larga sobre el script a ver y una vez en el menú contextual, pulsando en “Ejecutar Script”.
- Transferir Script: El cual se accede pulsando el botón inferior “Crear Script”, accediendo a una pantalla como sigue:



Figura 10.20: Menú principal y acceso a “Crear Script”

se procederá a llenar el recuadro con las instrucciones del script, y pulsando el botón “Transferir Script” para aceptarlo, dando lugar a una ventana de confirmación donde pedirá el nombre del script a transferir. Una vez introducido, se pulsará sobre el botón transferir para mandar la orden al servidor.

#### 10.3.7. Apagar y Reiniciar

Para acceder a estas funcionalidades, habrá que acceder al menú principal y pulsar en los botones correspondientes. Una vez pulsado, se pedirá confirmación antes de realizar la acción deseada.



Figura 10.21: Menú principal y acceso a “Apagar” y “Reiniciar”

#### 10.4. Instalación del servidor

Para la instalación del servidor, tendremos solamente un archivo, el cual tendrá el nombre de “Servidor RemSys.jar” con un aspecto como sigue:

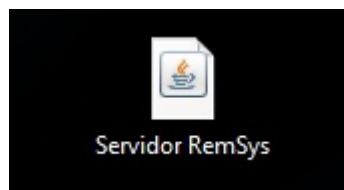


Figura 10.22: Fichero “Servidor RemSys.jar”

Dicho fichero es directamente ejecutable, en cuyo directorio de ejecución se creará también la carpeta de Scripts.

### 10.4.1. Configuración del Servidor

#### 10.4.1.1. Configuración de los puertos (escucha y transferencia)

Para configurar los puertos, tanto de escucha como de transferencia, haremos click en el menú “Configuración...” y luego pulsaremos en “Configurar puerto”, dando lugar a una pantalla como sigue:

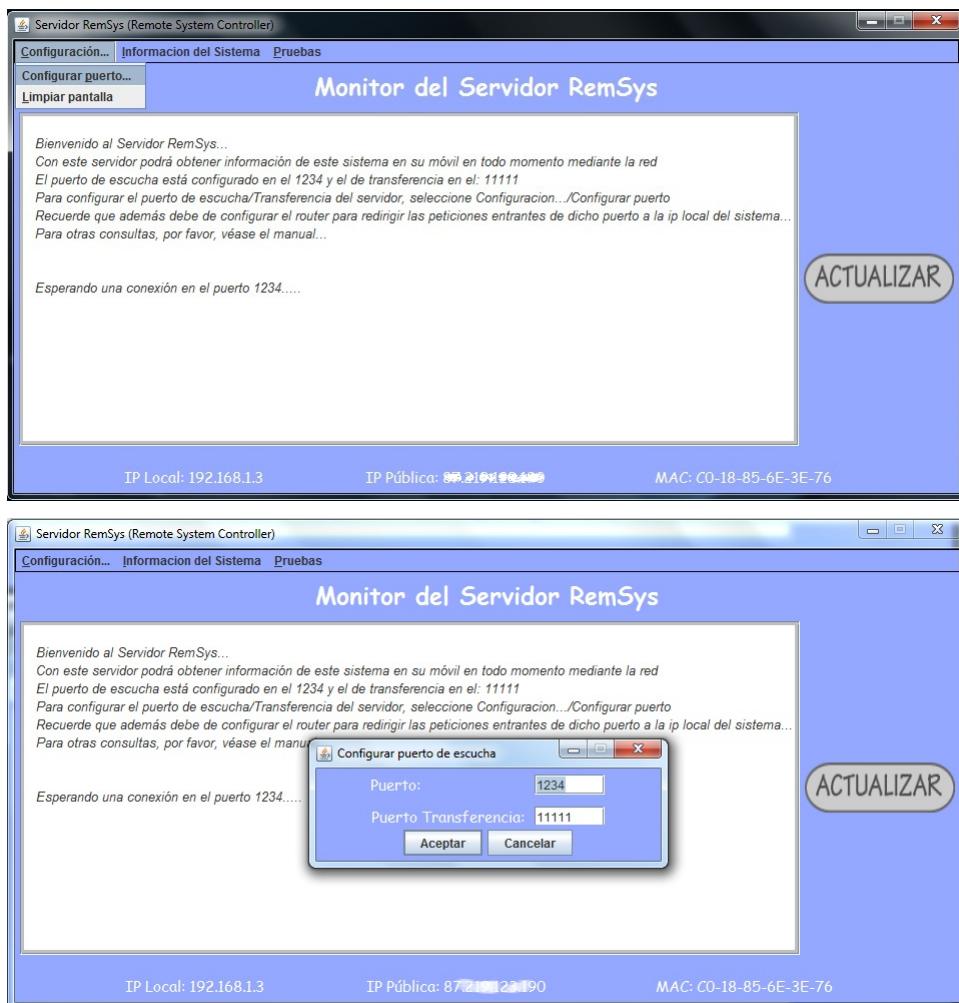


Figura 10.23: Acceso a la configuración de los puertos del servidor RemSys

donde se indicarán los puertos pertenecientes al de escucha y el de transferencia.

#### 10.4.1.2. Acceso a la información del sistema

Para acceder a la información del sistema, haremos click en “Información del sistema” y posteriormente pulsamos en “Direcciones”, dando lugar a una ventana como sigue:



Figura 10.24: Acceso a la información del sistema del servidor RemSys

donde se indicarán tanto las direcciones MAC de las interfaces de red, como las ip's locales de dichas interfaces en un listado que actualizará los datos mostrados en la ventana principal.

#### 10.4.1.3. Acceso a las pruebas

Para acceder a la información del sistema, haremos click en “Pruebas” y posteriormente pulsamos en la prueba que deseemos realizar:



Figura 10.25: Acceso a las pruebas del servidor RemSys

El resultado de la pulsación en dicho menú de pruebas, es la ejecución de la prueba y la salida por el monitor del servidor RemSys.

#### 10.4.1.4. Abrir los puertos del router

Para redirigir las peticiones entrantes por el puerto X, sea el puerto de transferencia X1 y la dirección IP Y del PC donde se está ejecutando nuestro Servidor RemSys, tendremos que acceder a la configuración del router. Normalmente se hace con los siguientes pasos:

1. Abrir Navegador de internet.
2. Introducir la dirección 192.168.1.1
3. Introducir credenciales: Normalmente son usuario admin, contraseña admin aunque también puede ser 1234 y combinaciones de éstas.
4. Una vez dentro del router, nos iremos al menú NAT y “Mapeo de puertos”
5. Introducimos los datos correspondiente a nuestra IP y el o los puertos (escucha y trasnferencia) pertenecientes al Servidor RemSys.

En la siguiente imagen vemos un ejemplo de la configuración de un router Huawei:

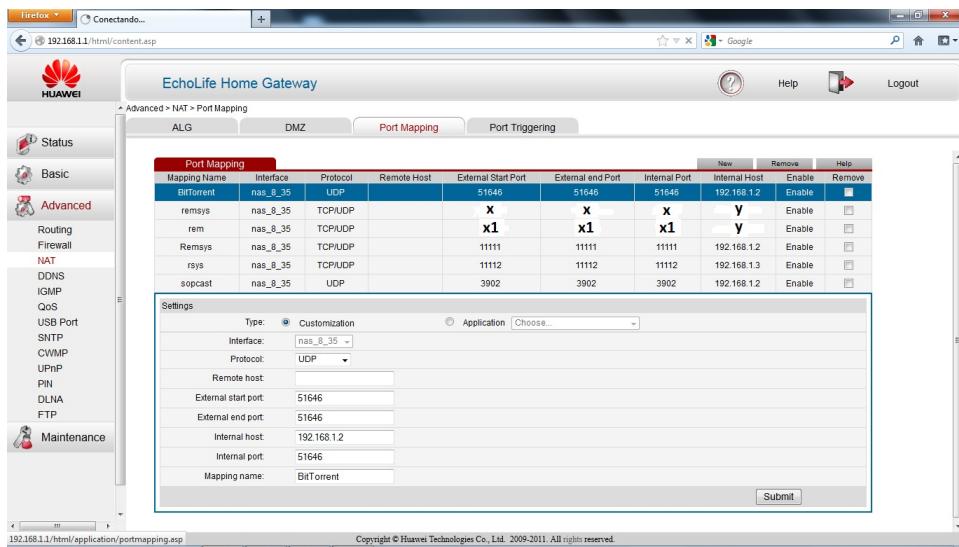


Figura 10.26: Configuración del router para el servidor RemSys

#### 10.4.1.5. Configuración para encendido remoto (Wake On Lan)

Hay por internet muchos manuales que nos permiten configurar nuestro equipo para el encendido remoto. Aquí veremos los pasos más importantes y básicos para dicha configuración, ya que depende del gestor de configuración de la BIOS y del sistema operativo utilizado. La configuración básica de la BIOS es:

- Acceder a la BIOS: Las maneras de acceder a la BIOS son diversas, pero las más comunes son pulsar el botón "Supr", o "Del", o "F1" o "F2" cuando iniciamos el sistema.
- Una vez dentro tenemos que ir a "Power management" y cambiar "Wake-On-LAN" a "Enabled".
- Cambiar el valor de "ACPI Suspend type" a "S3(SRT)".

Aquí vemos algunas imágenes de la configuración de la BIOS:

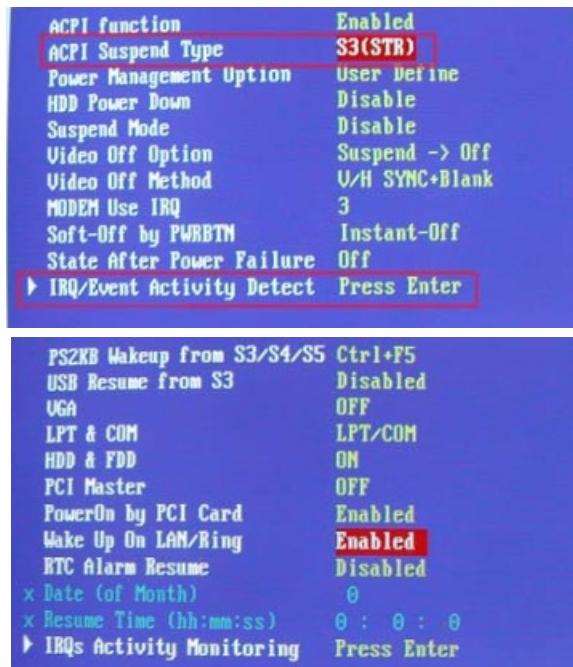


Figura 10.27: Configuración de la BIOS para la funcionalidad “Wake On Lan” (WOL)

Para configurar el adaptador de red para el encendido remoto, tendremos que:

- En Windows:
  - Acceder al administrador de dispositivos (Inicio, click botón derecho del ratón sobre “Equipo”, Propiedades y en la parte izquierda “Administrador de dispositivos”).
  - Buscar el dispositivo de red y botón derecho, propiedades.
  - En Opciones Avanzadas seleccionar :
    - Wake Up Capabilities : Magic Packet
  - En Administración de Energía, seleccionar :
    - Permitir sólo un Magic Packet para reactivar el equipo.

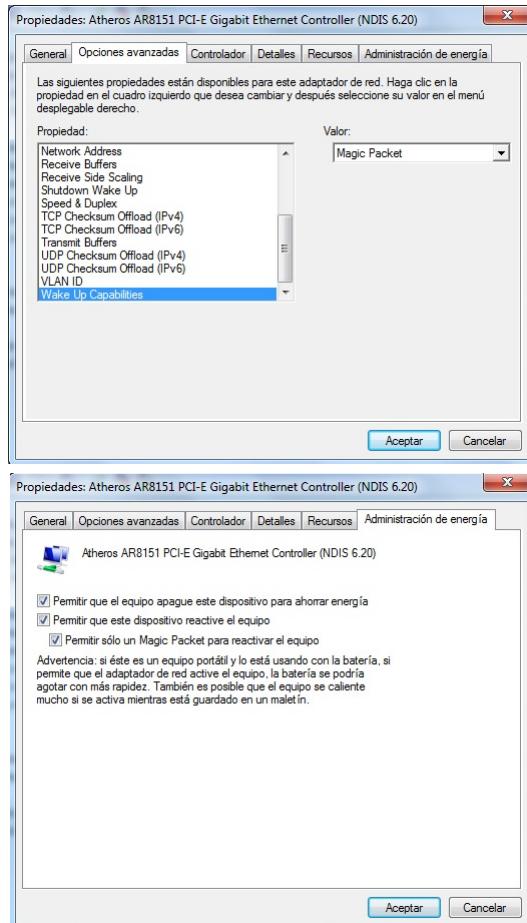
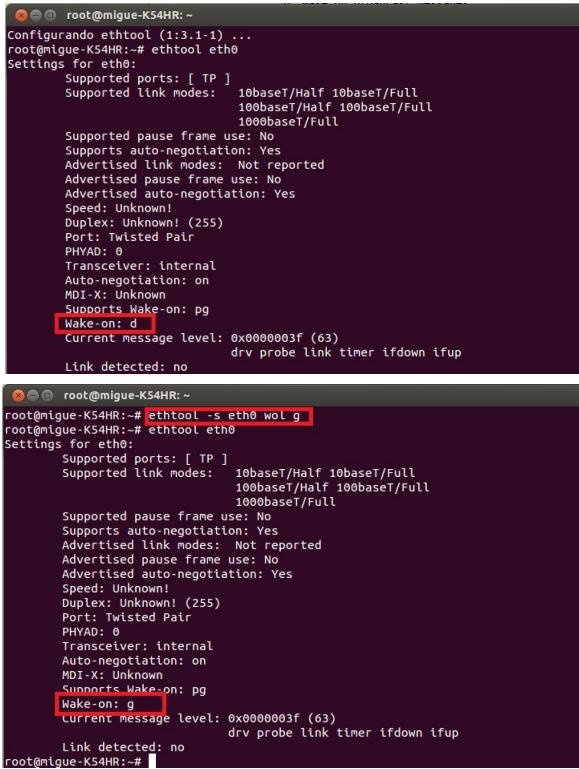


Figura 10.28: Configuración de las propiedades de los dispositivos de red en Windows

- En Linux utilizamos la herramienta ethtool y simplemente tendremos que ejecutar la orden “ethtool -s eth0 wol g”, correspondiendo eth0 con nuestra interfaz de red.



```

root@miguel-K54HR:~ root@miguel-K54HR:~ 
Configurando ethtool (1:3.1-1) ...
root@miguel-K54HR:~# ethtool eth0
Settings for eth0:
Supported ports: [ TP ]
Supported link modes: 10baseT/Half 10baseT/Full
100baseT/Half 100baseT/Full
1000baseT/Full
Supported pause frame use: No
Supports auto-negotiation: Yes
Advertised link modes: Not reported
Advertised pause frame use: No
Advertised auto-negotiation: Yes
Speed: Unknown!
Duplex: Unknown! (255)
Port: Twisted Pair
PHYAD: 0
Transceiver: internal
Auto-negotiation: on
MDI-X: Unknown
Supports Wake-on: pg
Wake-on: d
Current message level: 0x0000003f (63)
drv probe link timer ifdown ifup
Link detected: no

root@miguel-K54HR:~ root@miguel-K54HR:~ 
root@miguel-K54HR:~# ethtool -s eth0 wol g
root@miguel-K54HR:~# ethtool eth0
Settings for eth0:
Supported ports: [ TP ]
Supported link modes: 10baseT/Half 10baseT/Full
100baseT/Half 100baseT/Full
1000baseT/Full
Supported pause frame use: No
Supports auto-negotiation: Yes
Advertised link modes: Not reported
Advertised pause frame use: No
Advertised auto-negotiation: Yes
Speed: Unknown!
Duplex: Unknown! (255)
Port: Twisted Pair
PHYAD: 0
Transceiver: internal
Auto-negotiation: on
MDI-X: Unknown
Supports Wake-on: pg
Wake-on: g
Current message level: 0x0000003f (63)
drv probe link timer ifdown ifup
Link detected: no
root@miguel-K54HR:~#

```

Figura 10.29: Configuración de las propiedades de los dispositivos de red en Linux

#### 10.4.1.6. Ejecución al inicio del Sistema Operativo

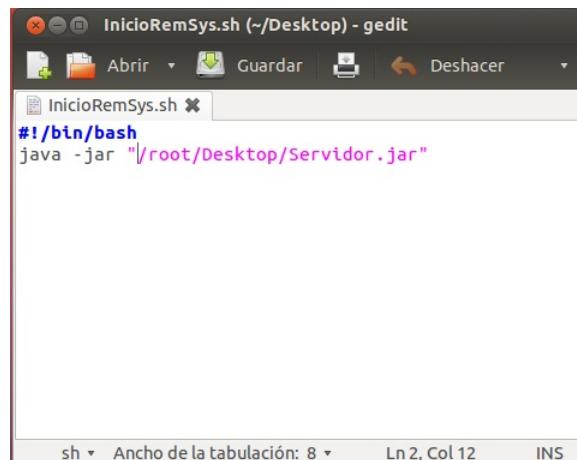
En esta sección vamos a ver como podemos añadir el programa servidor “Servidor RemSys.jar” al comienzo de la ejecución del sistema operativo.

- Windows

- En Windows simplemente, situados el fichero “Servidor RemSys.jar” en el directorio que queramos dejarlo instalado, creamos un acceso directo (botón derecho del ratón -> Crear acceso directo).
- Una vez creado el acceso directo, lo movemos a la carpeta “Inicio” del Windows (por ejemplo podría ser : C:\Users\”Usuario”\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup).
- Una vez hecho esto, reiniciamos y veremos como se inicia al comienzo de la sesión.

- Linux

- Crear un Script, que mediante java ejecute “Servidor RemSys.jar”, cerciorándonos que posee los permisos de ejecución oportunos (chmod). El script contendrá la orden “java -jar PATH\_A\_SERVIDOR\_REMSYS.jar”, como por ejemplo:



```
#!/bin/bash
java -jar "/root/Desktop/Servidor.jar"
```

Figura 10.30: Realización script bash para ejecución al inicio de RemSys

- Una vez creado el Script, nos vamos a “Aplicaciones al inicio” del menú del sistema del escritorio de Linux, en este caso, he utilizado Ubuntu.

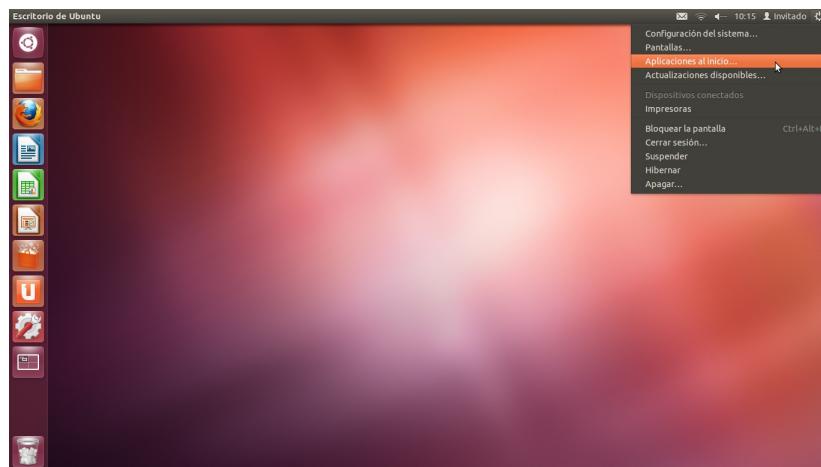


Figura 10.31: Acceso a aplicaciones de inicio en un sistema Linux (Ubuntu)

- Pulsamos sobre “Añadir”, rellenando el nombre, la orden que será la localización del script realizado y dando una descripción del mismo. Podemos ver el proceso en las siguientes imágenes, en este caso el fichero estará situado en “/root/Desktop/Servidor.jar”:

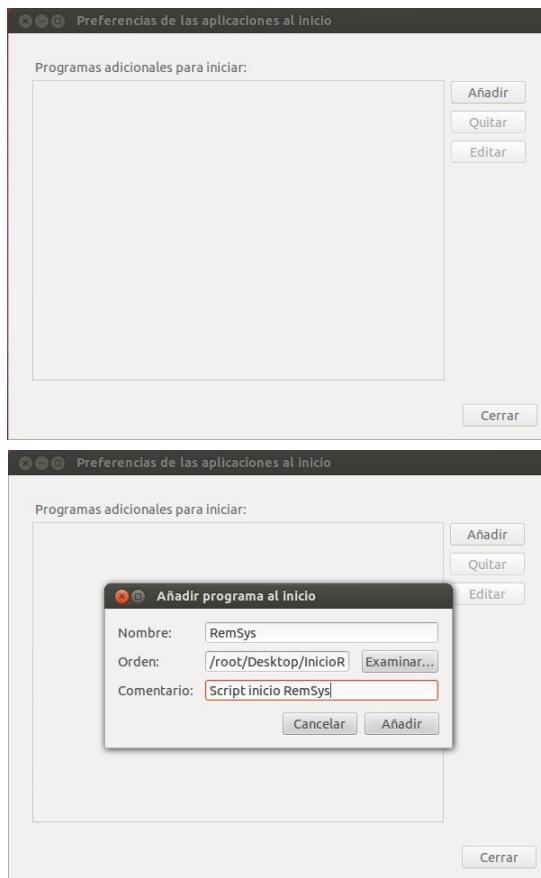


Figura 10.32: Configuración de Ubuntu para ejecución automática de inicio de RemSys

#### 10.4.1.7. Permisos

En esta sección vamos a ver como dar los principales permisos al usuario, principalmente en Linux, ya que en Windows basta con crear el usuario como administrador, y si no queremos recibir notificaciones desactivar éstas buscando

“Cambiar configuración de control de cuentas de usuario” en el “Centro de actividades”. Así pues, nos centraremos en los sistemas Linux, ya que es importante dar permisos de ejecución para el usuario, tanto a los ficheros de Scripts y el ejecutable del Servidor RemSys, como a los comandos que se utilizan en ellos.

A continuación mostraremos como dar permisos para apagar y reiniciar en un sistema Linux (Ubuntu). Primeramente localizamos los comandos que nos hacen falta para realizar las acciones, estas van a ser:

```
halt  
reboot
```

Una vez localizados, tenemos que logearnos como superusuario mediante el comando “su”.

Acto seguido, vamos a crear un grupo llamado shutdown, para permitirles apagar y reiniciar la máquina:

```
addgroup shutdown
```

Después, vamos a añadir al grupo creado, los comandos necesarios mediante las ordenes:

```
chgrp shutdown /sbin/shutdown /sbin/reboot /sbin/halt  
chmod u+s,o-rwx /sbin/shutdown /sbin/reboot /sbin/halt
```

Después a crear los enlaces simbólicos para los comandos, y disponibles para el usuario:

```
ln -s /sbin/shutdown /usr/bin/shutdown  
ln -s /sbin/reboot /usr/bin/reboot  
ln -s /sbin/halt /usr/bin/halt
```

Por último, agregaremos al usuario “X” al grupo “shutdown” para que pueda hacer uso de esos comandos:

```
adduser X shutdown
```

y procedemos a reiniciar el equipo. Con esto ya le habríamos dado al usuario X los permisos para apagar, reiniciar y suspender el equipo y de esta manera poder mandarlas desde el dispositivo móvil y proceder a su ejecución en el servidor.

# Capítulo 11

## Conclusiones

La elaboración de este proyecto, me ha servido para introducirme en el lenguaje Java principalmente. Así mismo, también me ha servido para implantar un protocolo para realización de acciones concretas según qué acciones, todo ello a través de la red, el cual me ha dado gran satisfacción personal ya que es una de mis grandes pasiones, la red, la comunicación a través de ella y las nuevas tecnologías. Al haberse contruido dos software, he aprendido los principios básicos tanto de el sistema Android, como de Java y la construcción de interfaces gráficas con Java Swing. RemSys ha sido diseñado para poderse instalar y controlar computadoras que presenten instaladas sistemas Windows o Linux, es decir es multiplataforma, quedando únicamente la integración de sistemas MACs. Incluso en versiones posteriores se podrá portar el programa cliente, tanto para iphone como para Windows mobile.

En conclusión, he aprendido mucho con este proyecto, tanto en su desarrollo como su análisis, diseño y realización de documentación... la organización de su realización es básica para cualquier tipo de proyecto. Me ha reportado una gran satisfacción personal ya que he podido solventar problemas que siempre, en cualquier proyecto, surgen inesperadamente con resultados satisfactorios.

El proyecto me parece bastante completo, ya que podemos ejecutar scripts, modificar el contenido de cualquier directorio, acceder a la información del sistema y más general, acceder al simbolo del sistema o terminal (consola), en fin, nos da muchas herramientas que pueden ser útiles para un administrador del sistema.

### 11.1. Troubleshooting

Varios problemas se han presentado a la hora de realizar el proyecto. Alguno de ellos vienen en cuanto a la obtención de la información, otras simplemente por la estructuración del programa. Vemos algunos ejemplo:

- En la obtención de la información de interfaces de red, la información en la plataforma Windows no viene ordenada. La solución dada es la reestructuración de la información obtenida y un nuevo procesamiento sobre ella para ordenar dicha información para cada interfaz de red.
- Decisión de realizar la conexión del sistema mediante un servicio, para de este modo poder mantener la conexión que el sistema reomoto en todo momento.
- No existencia del protocolo RMI para Android debido a la serialización, decidiéndose su realización mediante Sockets.
- No se pudo realizar una versión con sockets SSL, la cual puede ser realizada en versiones posteriores y con poca modificación de código. Debido a que la conexión está implementado como un servicio, sólo se tendría que modificar dicha fichero del servicio para aplicarle una característica SSL.
- Se modificó el proceso de “Transferencia de fichero a SD-Card”, en lugar de mediante lanzamiento de hilos, se procedió a realizarlo mediante la construcción de “AsyncTask” o tarea asíncrona, ya que, anteriormente, se podía comenzar una transferencia cuando no había terminado la anterior, dando lugar a ficheros corruptos. Con esta funcionalidad, introducida por Android, se podrá realizar tareas en segundo plano, mientras se realizan otras en primero.

## 11.2. Versiones posteriores

Para versiones posteriores se podrá por ejemplo:

- Utilizar SSL sockets, para la codificación de la información y mensajes que pasan a través de la red.
- Aumento de las funcionalidades del sistema.
- Cambios de la interfaz y mayor nivel de personalización.

# Capítulo 12

## Apéndice

### 12.1. Código Servidor RemSys

Servidor.java

---

```
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Component;
import java.awt.Container;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.Image;
import java.awt.Point;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.BufferedReader;
import java.io.EOFException;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.PrintWriter;
import java.net.InetAddress;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.UnknownHostException;
import java.util.Vector;

import javax.swing.AbstractAction;
import javax.swing.Action;
import javax.swing.BorderFactory;
import javax.swing.DefaultListCellRenderer;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JList;
import javax.swing.JMenu;
```

```

import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.ListSelectionModel;
import javax.swing.SwingConstants;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;
import javax.swing.text.DefaultCaret;

public class Servidor extends JFrame {
    private static final long serialVersionUID = 1L;
    private JTextArea areaPantalla;
    private JButton Activar_Desactivar;
    private JTextField direccionIP;
    private JTextField mac;
    private InetAddress addr;
    private JTextField direccionlocal;
    private MacAddress dirmac;
    private ServerSocket servidor;
    private Socket conexion;
    private ServerSocket servidortrans;
    private Socket conexiontrans;
    private ObjectOutputStream salida;
    private ObjectInputStream entrada;
    private String mensaje;
    private Process p ;
    private InputStream is ;
    private BufferedReader br;
    private String aux;
    private String nombreScript;
    private String DireccionScripts;
    private String SistemaOperativo;
    private boolean IndicadorPrueba;
    private volatile Thread HiloEjecutarServidor;
    private int Puerto =1234;
    private int PuertoTransferencia=11111;
    int cont=0;
    String excepcion="";
    // Fuentes
    Font boldUnderline = new Font("Comic Sans MS",Font.ROMAN_BASELINE,
        15);
    Font FuenteAreaPantalla = new Font("Console",Font.HANGING_BASELINE,
        13);
    Font FuenteTitulo = new Font("Comic Sans MS",Font.BOLD, 24);
    Font FuenteTituloSubventana = new Font("Comic Sans MS",Font.BOLD,
        18);

    // Colores
    Color color = new Color(149,168,255);

    // Frame Servidor
    public Servidor(){
        super( "Servidor RemSys (Remote System Controller)");

        // Establecemos el Sistema Operativo
        if(System.getProperty("os.name").startsWith("Windows"))
            SistemaOperativo = "Windows";
        else
            SistemaOperativo = "Linux";

        if(SistemaOperativo.equals("Windows"))
            setSize(950, 480);
        else

```

```
setSize(1085, 480);

// Establecemos el icono del programa Servidor
ImageIcon icon = new ImageIcon(".\\Imagenes\\remsyslogo.png");
setIconImage(icon.getImage().getScaledInstance(190, 130, java.awt.Image.SCALE_DEFAULT));

setBackground(Color.CYAN);

// Establecemos la posicion de la ventana principal del Servidor
Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize()
;
Point middle = new Point(screenSize.width / 2, screenSize.height /
2);
Point newLocation = new Point(middle.x - (getWidth() / 2), middle.y -
(getHeight() / 2));
setLocation(newLocation);

JTextField Titulo = new JTextField("Monitor del Servidor RemSys");

areaPantalla = new JTextArea(18,70);
areaPantalla.setFont(FuenteAreaPantalla);
areaPantalla.setBorderStyle(BorderFactory.createLineBorder(Color.lightGray, 3));

DefaultCaret caret = (DefaultCaret)areaPantalla.getCaret();
caret.setUpdatePolicy(DefaultCaret.ALWAYSUPDATE);

HiloEjecutarServidor = this.new HiloEjecutarServidor();

 JPanel panel1 = new JPanel();
 JPanel panel2 = new JPanel();
 JPanel panel3 = new JPanel();
 Container contenedor = getContentPane();
panel3.add(Titulo);

panel2.setBackground(color);
panel3.setBackground(color);
panel1.setBackground(color);

 ImageIcon cup = new ImageIcon(this.getClass().getResource("btactualizar.png"));

Image img = cup.getImage();
Image newimg = img.getScaledInstance(150, 50, java.awt.Image.SCALESMOOTH);
cup = new ImageIcon(newimg);

ManejadorBotones manejador = new ManejadorBotones();

Activar_Desactivar = new JButton(cup);
Activar_Desactivar.setPreferredSize(new Dimension(150, 50));
Activar_Desactivar.addActionListener(manejador);
Activar_Desactivar.setBorderPainted(false);
Activar_Desactivar.setContentAreaFilled(false);

direccionlocal = new JTextField(16);
direccionlocal.setEditable(false);
try {
addr = InetAddress.getLocalHost();
```

```

direccionlocal.setText("IP Local: " + addr.getHostAddress().trim()
);

} catch (UnknownHostException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}

dirmac= new MacAddress();
mac = new JTextField(" MAC: ",18);
mac.setEditable(false);
mac.setFont(boldUnderline);
mac.setBackground(color);
mac.setForeground(Color.WHITE);
mac.setBorder(null);
mac.setEditable(false);

try{
mac.setText(" MAC: " + dirmac.MaC.toUpperCase().trim());
}catch(NullPointerException e){
// No se tiene conexión y no se accede a InetAddress.
getLocalHost() por lo tanto no obtiene la MAC.
}

direccionIP = new JTextField("IP Pública: ",23);
direccionIP.setFont(boldUnderline);
direccionIP.setBackground(color);
direccionIP.setForeground(Color.WHITE);
direccionIP.setBorder(null);
direccionIP.setEditable(false);

try{
GetIP ip = new GetIP();
direccionIP.setText(direccionIP.getText().toString().concat(ip
.IPaddress.toString().trim()));
}catch(NullPointerException e){
}

Titulo.setFont(FuenteTitulo);
Titulo.setBackground(color);
Titulo.setForeground(Color.WHTE);
Titulo.setBorder(null);
Titulo.setEditable(false);

direccionlocal.setFont(boldUnderline);
direccionlocal.setBackground(color);
direccionlocal.setForeground(Color.WHITE);
direccionlocal.setBorder(null);
direccionlocal.setEditable(false);

panel2.add(new JScrollPane(areaPantalla));
panel2.add(Activar_Desactivar);

panel1.add( direccionlocal );
JTextField Espacio1 = new JTextField(5);
Espacio1.setBackground(color);
Espacio1.setBorder(null);
Espacio1.setEditable(false);

```

```

panel1.add(Espacio1);
try{
    panel1.add(direccionIP);
} catch(NullPointerException e){
}
JTextField Espacio2 = new JTextField(2);
Espacio2.setBackground(color);
Espacio2.setBorder(null);
Espacio2.setEditable(false);
panel1.add(Espacio2);

try{
    panel1.add(mac);
} catch(NullPointerException e){
}

CrearMenus();

contenedor.add(panel1, BorderLayout.PAGE_END);
contenedor.add(panel2, BorderLayout.LINE_START );
contenedor.add(panel3, BorderLayout.BEFORE_FIRST_LINE );

setVisible(true);
setResizable(false);
areaPantalla.setEditable(false);

} // fin del constructor de Servidor

private void CrearMenus(){
// Creación de menús
JMenu menuConfigurarPuerto = new JMenu("Configuración...");
menuConfigurarPuerto.setMnemonic('C');

// establecer elemento de menú Propiedades...
JMenuItem ConfigurarPuerto = new JMenuItem("Configurar puerto...");
ConfigurarPuerto.setMnemonic('P');
menuConfigurarPuerto.add(ConfigurarPuerto);

JMenuItem LimpiarPantalla = new JMenuItem("Limpiar pantalla");
LimpiarPantalla.setMnemonic('L');
menuConfigurarPuerto.add(LimpiarPantalla);

ConfigurarPuerto.addActionListener(new ActionListener() {
    public void actionPerformed( ActionEvent evento ) {
        // Ventana Propiedades
        CrearVentanaConfigurarPuerto();
    }
}); // fin de la llamada a addActionListener

LimpiarPantalla.addActionListener(new ActionListener() {
    public void actionPerformed( ActionEvent evento ) {
        // Ventana Propiedades
        areaPantalla.setText("");
    }
}); // fin de la llamada a addActionListener

JMenu menuPruebas = new JMenu("Pruebas");

```

```

menuPruebas.setMnemonic( 'P' );

// establecer elemento de menú Propiedades...
JMenuItem PruebaDiscos = new JMenuItem( "Discos" );
PruebaDiscos.setMnemonic( 'P' );
menuPruebas.add( PruebaDiscos );

PruebaDiscos.addActionListener( new ActionListener() {
    public void actionPerformed( ActionEvent evento ) {
        // Ventana Propiedades
        IndicadorPrueba = true;
        InformacionDiscos();
        ReiniciarServidor();
    }
}); // fin de la llamada a addActionListener

JMenuItem PruebaRed = new JMenuItem( "Red" );
PruebaRed.setMnemonic( 'R' );
menuPruebas.add( PruebaRed );

PruebaRed.addActionListener( new ActionListener() {
    public void actionPerformed( ActionEvent evento ) {
        // Ventana Propiedades
        IndicadorPrueba = true;
        InformacionRed();
        ReiniciarServidor();
    }
}); // fin de la llamada a addActionListener

JMenuItem PruebaSistemaOperativo = new JMenuItem( "SistemaOperativo" );
PruebaSistemaOperativo.setMnemonic( 'S' );
menuPruebas.add( PruebaSistemaOperativo );

PruebaSistemaOperativo.addActionListener( new ActionListener() {
    {
        public void actionPerformed( ActionEvent evento ) {
            // Ventana Propiedades
            IndicadorPrueba = true;
            InformacionSistemaOperativo();
            ReiniciarServidor();
        }
    }
}); // fin de la llamada a addActionListener

JMenuItem PruebaMemoria = new JMenuItem( "Memoria" );
PruebaMemoria.setMnemonic( 'P' );
menuPruebas.add( PruebaMemoria );

PruebaMemoria.addActionListener( new ActionListener() {
    public void actionPerformed( ActionEvent evento ) {
        // Ventana Propiedades
        IndicadorPrueba = true;
        InformacionMemoria();
        ReiniciarServidor();
    }
}); // fin de la llamada a addActionListener

JMenuItem PruebaProcesos = new JMenuItem( "Procesos" );
PruebaProcesos.setMnemonic( 'P' );
menuPruebas.add( PruebaProcesos );

```

```

PruebaProcesos.addActionListener( new ActionListener() {
    public void actionPerformed( ActionEvent evento ) {
        // Ventana Propiedades
        IndicadorPrueba = true;
        ListadoProcesos();
        ReiniciarServidor();
    }
}); // fin de la llamada a addActionListener

JMenuItem PruebaOrdenLibre = new JMenuItem( "Orden Libre" );
PruebaOrdenLibre.setMnemonic( 'O' );
menuPruebas.add( PruebaOrdenLibre );

PruebaOrdenLibre.addActionListener( new ActionListener() {
    public void actionPerformed( ActionEvent evento ) {
        // Ventana Propiedades
        IndicadorPrueba = true;
        CrearVentanaOrdenLibre();
    }
}); // fin de la llamada a addActionListener

final JMenu PruebaScripts = new JMenu( "Scripts" );
PruebaScripts.setMnemonic( 'S' );
menuPruebas.add( PruebaScripts );

JMenuItem PruebaVerScripts = new JMenuItem( "Ver Script" );
PruebaVerScripts.setMnemonic( 'S' );
PruebaScripts.add( PruebaVerScripts );

PruebaVerScripts.addActionListener( new ActionListener() {
    public void actionPerformed( ActionEvent evento ) {
        // Ventana Propiedades
        IndicadorPrueba = true;
        CrearVentanaScripts(1);
    }
}); // fin de la llamada a addActionListener

JMenuItem PruebaEjecutarScripts = new JMenuItem( "Ejecutar Script" );
PruebaEjecutarScripts.setMnemonic( 'S' );
PruebaScripts.add( PruebaEjecutarScripts );

PruebaEjecutarScripts.addActionListener( new ActionListener() {
    public void actionPerformed( ActionEvent evento ) {
        // Ventana Propiedades
        IndicadorPrueba = true;
        CrearVentanaScripts(2);
    }
}); // fin de la llamada a addActionListener

JMenu InformacionSistema = new JMenu( "Informacion del Sistema" );
InformacionSistema.setMnemonic( 'I' );

```

```

// establecer elemento de menú Propiedades...
JMenuItem Informacion = new JMenuItem( "Direcciones ..." );
Informacion.setMnemonic( 'f' );
InformacionSistema.add( Informacion );

Informacion.addActionListener( new ActionListener() {
    public void actionPerformed( ActionEvent evento ) {
        // Ventana Propiedades
        CrearVentanaInformacionSistema();
    }
}); // fin de la llamada a addActionListener

JMenuBar BarraMenus = new JMenuBar();
setJMenuBar( BarraMenus );
BarraMenus.add( menuConfigurarPuerto );
BarraMenus.add( InformacionSistema );
BarraMenus.add( menuPruebas );
BarraMenus.setBackground( color );
BarraMenus.setBorder( BorderFactory.createLineBorder( Color.BLACK
));
}

private void CrearVentanaConfigurarPuerto(){
    final JFrame VentanaPropiedades = new JFrame("Configurar puerto de
    escucha");
    VentanaPropiedades.setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
    VentanaPropiedades.setSize(350, 120);
    JPanel PanelPropiedades = new JPanel();
    JPanel PanelPropiedadesTransferencia = new JPanel();
    JPanel PanelBotonesPropiedades = new JPanel();

    Container contenedor = VentanaPropiedades.getContentPane();
    VentanaPropiedades.setLocationRelativeTo(null);

    // Creamos un componente; aquí hay polimorfismo...
    final JTextField CampoTextoPuerto = new JTextField(6); // un área
    de texto
    CampoTextoPuerto.setText( String.valueOf(Puerto) );
    CampoTextoPuerto.addFocusListener(new java.awt.event.FocusAdapter
    () {
        public void focusGained(java.awt.event.FocusEvent evt) {
            CampoTextoPuerto.selectAll();
        }
    });
}

@SuppressWarnings("serial")
Action nextFocusAction = new AbstractAction("Move Focus Forwards")
{
    public void actionPerformed(ActionEvent evt) {
        ((Component) evt.getSource()).transferFocus();
    }
};
@SuppressWarnings("serial")
Action prevFocusAction = new AbstractAction("Move Focus Backwards")
{
    public void actionPerformed(ActionEvent evt) {
        ((Component) evt.getSource()).transferFocusBackward();
    }
};

```

```

        CampoTextoPuerto.getActionMap().put(nextFocusAction.getValue(
            Action.NAME), nextFocusAction);
        CampoTextoPuerto.getActionMap().put(prevFocusAction.getValue(
            Action.NAME), prevFocusAction);

final JTextField CampoEtiquetaPuerto = new JTextField("Puerto :
"); // un área de texto
CampoEtiquetaPuerto.setEditable(false);
CampoEtiquetaPuerto.setBorder(null);
CampoEtiquetaPuerto.setBackground(color);
CampoEtiquetaPuerto.setForeground(Color.white);
CampoEtiquetaPuerto.setFont(boldUnderline);

final JTextField CampoTextoPuertoTransferencia = new JTextField
(6);
CampoTextoPuertoTransferencia.setText(String.valueOf(
    PuertoTransferencia));
final JTextField CampoEtiquetaPuertoTransferencia = new
    JTextField("Puerto Transferencia: "); // un área de texto
CampoEtiquetaPuertoTransferencia.setEditable(false);
CampoEtiquetaPuertoTransferencia.setBorder(null);
CampoEtiquetaPuertoTransferencia.setBackground(color);
CampoEtiquetaPuertoTransferencia.setForeground(Color.white);
CampoEtiquetaPuertoTransferencia.setFont(boldUnderline);
// Añadir el componente al panel de la ventana
PanelPropiedades.add(CampoEtiquetaPuerto);
PanelPropiedades.add(CampoTextoPuerto);
PanelPropiedadesTransferencia.add(
    CampoEtiquetaPuertoTransferencia);
PanelPropiedadesTransferencia.add(CampoTextoPuertoTransferencia);

// Botones Aceptar y Cancelar
 JButton btnAceptar = new JButton("Aceptar");
 JButton btnCancelar = new JButton("Cancelar");
 PanelBotonesPropiedades.add(btnAceptar);
 PanelBotonesPropiedades.add(btnCancelar);

// Acciones de los botones
btnAceptar.addActionListener( new ActionListener() {
    public void actionPerformed( ActionEvent evento ) {
        // Ventana Propiedades
        Puerto = Integer.parseInt(CampoTextoPuerto.getText() .
            toString());
        PuertoTransferencia = Integer.parseInt(
            CampoTextoPuertoTransferencia.getText() .toString());
        if(Puerto==PuertoTransferencia){
            JOptionPane.showMessageDialog(VentanaPropiedades , "
                Puerto de escucha y transferencia iguales , por
                favor , modifique alguno de ellos");
        }else{
            VentanaPropiedades.dispose();
            JOptionPane.showMessageDialog(VentanaPropiedades , "El
                puerto de escucha se ha cambiado a : " + Puerto +
                " y el de transferencia al: " +
                PuertoTransferencia );
            ReiniciarServidor();
        }
    }
}); // fin de la llamada a addActionListener

btnCancelar.addActionListener( new ActionListener() {
    public void actionPerformed( ActionEvent evento ) {

```

```

        // Ventana Propiedades
        VentanaPropiedades.dispose();
    }

}); // fin de la llamada a addActionListener

VentanaPropiedades.getRootPane().setDefaultButton(btnAceptar);

PanelPropiedades.setBackground(color);
PanelPropiedadesTransferencia.setBackground(color);
PanelBotonesPropiedades.setBackground(color);
contenedor.add(PanelPropiedades, BorderLayout.NORTH);
contenedor.add(PanelPropiedadesTransferencia, BorderLayout.CENTER);
contenedor.add(PanelBotonesPropiedades, BorderLayout.SOUTH);

// mostrar la ventana; igual que en AWT
VentanaPropiedades.setVisible(true);
VentanaPropiedades.setResizable(false);
CampoTextoPuerto.requestFocus();
}

private void CrearVentanaInformacionSistema() {
    String DirMac = null;
    Vector<String> DireccionesMAC = new Vector<String>();
    String DirIP = null;
    Vector<String> DireccionesIPs = new Vector<String>();
    final JFrame VentanaPropiedades = new JFrame("Informacion de las
        direcciones del Sistema (MAC/IP)");
    VentanaPropiedades.setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
    if(SistemaOperativo.equals("Windows"))
        VentanaPropiedades.setSize(570, 150);
    else
        VentanaPropiedades.setSize(650, 150);

    JPanel PanelPropiedades = new JPanel();
    JPanel PanelBotonesPropiedades = new JPanel();
    JPanel PanelTitulo = new JPanel();

    Container contenedor = VentanaPropiedades.getContentPane();
    VentanaPropiedades.setLocationRelativeTo(null);

    try {
        if(SistemaOperativo.equals("Windows")){
            p = Runtime.getRuntime().exec ("cmd /c " + "ipconfig /all");
        }else{
            String [] command = {"sh", "-c", "ifconfig"};
            p = Runtime.getRuntime().exec (command);
        }

        is = p.getInputStream();

        if(SistemaOperativo.equals("Windows"))
            br = new BufferedReader (new InputStreamReader (is , "Cp850"));
        else
            br = new BufferedReader (new InputStreamReader (is));

        // Se lee la primera linea
        aux = br.readLine();

        // Mientras se haya leido alguna linea
        while (aux!=null) {
            if(SistemaOperativo.equals("Windows")){

```

```

        if(aux.startsWith("    Dirección física")){
            String datos[];
            datos=aux.split(":");
            DirMac = datos[1];
            if(DirMac.trim().length()<=17)
                DireccionesMAC.add(DirMac.trim());
        }

        if(aux.startsWith("    Dirección IPv4")){
            String datos[];
            datos=aux.split(":");
            DirIP = datos[1];
            DireccionesIPs.add(DirIP.trim().substring(0, DirIP.length()-13));
        }

    }else{
        if(aux.contains("direcciónHW")){
            String datos[];
            datos=aux.split(" ");
            DirMac = datos[datos.length-1];
            if(DirMac.trim().length()<=17)
                DireccionesMAC.add(DirMac.trim());
        }
        if(aux.contains("Direc. inet")){
            String datos[];
            datos=aux.split(":");
            DirIP = datos[1].substring(0,datos[1].length()-6);
            DireccionesIPs.add(DirIP.trim());
        }
    }

    aux = br.readLine();
}
}catch( IOException excepcionES ) {
    excepcionES.printStackTrace();
}
// Creamos un componente; aquí hay polimorfismo...
}

final JList ListaDirMacs = new JList(DireccionesMAC);
DefaultListCellRenderer cellRenderer = (DefaultListCellRenderer)
    ListaDirMacs.getCellRenderer();
cellRenderer.setHorizontalAlignment(SwingConstants.CENTER);
ListaDirMacs.setVisibleRowCount(2);
ListaDirMacs.setPrototypeCellValue(" C0-18-85-6E-3E-76 ");
ListaDirMacs.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
ListaDirMacs.addListSelectionListener(new ListSelectionListener() {
    public void valueChanged(ListSelectionEvent evt) {
        mac.setText(" MAC: " + (String)ListaDirMacs.getSelectedValue());
    }
});

final JList ListaDirIP = new JList(DireccionesIPs);
cellRenderer = (DefaultListCellRenderer)ListaDirIP.getCellRenderer()
;
cellRenderer.setHorizontalAlignment(SwingConstants.CENTER);
ListaDirIP.setVisibleRowCount(2);
ListaDirIP.setPrototypeCellValue(" 255.255.255.255 ");

ListaDirIP.setAlignmentY(CENTER_ALIGNMENT);
ListaDirIP.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
ListaDirIP.addListSelectionListener(new ListSelectionListener() {
    public void valueChanged(ListSelectionEvent evt) {

```

```

        direccionlocal.setText("IP Local: " + (String)ListaDirIP .
            getSelectedValue());
    });
});

final JTextField CampoEtiquetaPuerto = new JTextField("Direcciones
MAC : "); // un área de texto
CampoEtiquetaPuerto.setEditable(false);
CampoEtiquetaPuerto.setBorder(null);
CampoEtiquetaPuerto.setBackground(color);
CampoEtiquetaPuerto.setForeground(Color.white);
CampoEtiquetaPuerto.setFont(boldUnderline);

final JTextField CampoEtiquetaDirIps = new JTextField("Direcciones
IPs : "); // un área de texto
CampoEtiquetaDirIps.setEditable(false);
CampoEtiquetaDirIps.setBorder(null);
CampoEtiquetaDirIps.setBackground(color);
CampoEtiquetaDirIps.setForeground(Color.white);
CampoEtiquetaDirIps.setFont(boldUnderline);

// Añadir el componente al panel de la ventana
PanelPropiedades.add(CampoEtiquetaPuerto);
PanelPropiedades.add(new JScrollPane(ListaDirMacs));

// Botones Aceptar y Cancelar
 JButton btnAceptar = new JButton("Aceptar");
 JButton btnCancelar = new JButton("Cancelar");
 PanelBotonesPropiedades.add(btnAceptar);
 PanelBotonesPropiedades.add(btnCancelar);

// Acciones de los botones
btnAceptar.addActionListener( new ActionListener() {
    public void actionPerformed( ActionEvent evento ) {
        // Ventana Propiedades
        VentanaPropiedades.dispose();
    }
}); // fin de la llamada a addActionListener

btnCancelar.addActionListener( new ActionListener() {
    public void actionPerformed( ActionEvent evento ) {
        // Ventana Propiedades
        VentanaPropiedades.dispose();
    }
}); // fin de la llamada a addActionListener

VentanaPropiedades.getRootPane().setDefaultButton(btnAceptar);

JTextField Espacio = new JTextField(2);
Espacio.setBackground(color);
Espacio.setBorder(null);
PanelPropiedades.add(Espacio);
PanelPropiedades.add(CampoEtiquetaDirIps);
PanelPropiedades.add(ListaDirIP);
PanelPropiedades.setBackground(color);
PanelBotonesPropiedades.setBackground(color);
JTextField Titulo = new JTextField("Direcciones de red del sistema")
;
Titulo.setFont(FuenteTituloSubventana);
Titulo.setBackground(color);
Titulo.setForeground(Color.white);
Titulo.setBorder(null);
Titulo.setEditable(false);

```

```

PanelTitulo.add(Titulo);
PanelTitulo.setBackground(color);

contenedor.add(PanelTitulo, BorderLayout.NORTH);
contenedor.add(PanelPropiedades, BorderLayout.CENTER);
contenedor.add(PanelBotonesPropiedades, BorderLayout.SOUTH);

// mostrar la ventana; igual que en AWT
VentanaPropiedades.setVisible(true);
VentanaPropiedades.setResizable(false);
}

private void CrearVentanaScripts(final int i){
    final JFrame VentanaPropiedades = new JFrame("Scripts");
    VentanaPropiedades.setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
    VentanaPropiedades.setSize(350, 200);

    JPanel PanelPropiedades = new JPanel();
    JPanel PanelBotonesPropiedades = new JPanel();

    Container contenedor = VentanaPropiedades.getContentPane();
    VentanaPropiedades.setLocationRelativeTo(null);

    // Lista de Scripts...
    Vector<String> Scripts = new Vector<String>();
    try {
        if(SistemaOperativo.equals("Windows")){
            p = Runtime.getRuntime().exec ("cmd /c " + "dir /B " +
                DireccionScripts);
        }else{
            String[] command = {"sh", "-c", "ls " + DireccionScripts};
            p = Runtime.getRuntime().exec (command);
        }
        is = p.getInputStream();

        if(SistemaOperativo.equals("Windows"))
            br = new BufferedReader (new InputStreamReader (is, "Cp850"));
        else
            br = new BufferedReader (new InputStreamReader (is));
    }

    // Se lee la primera linea
    aux = br.readLine();

    // Mientras se haya leido alguna linea
    while (aux!=null) {

        Scripts.add(aux);
        aux = br.readLine();
    }
} catch ( IOException excepcionES ) {
    excepcionES.printStackTrace();
}

JTextField EtiquetaScript = null;

switch(i){
case 1:
    EtiquetaScript = new JTextField("Elija el Script que desea ver
    "); // un área de texto
    break;
case 2:
}

```

```

EtiquetaScript = new JTextField("Elija el Script que desea
                                ejecutar"); // un área de texto
break;
}

EtiquetaScript.setEditable(false);
EtiquetaScript.setBorder(null);
EtiquetaScript.setBackground(color);
EtiquetaScript.setForeground(Color.white);
EtiquetaScript.setFont(FuenteTituloSubventana);

final JList ListaScripts = new JList(Scripts);
ListaScripts.setVisibleRowCount(4);
ListaScripts.setSelectionMode(ListSelectionModel.SINGLE_SELECTION
);
ListaScripts.addListSelectionListener(new ListSelectionListener()
{
    public void valueChanged(ListSelectionEvent evt) {
        nombreScript = (String)ListaScripts.getSelectedValue();
    }
});

PanelPropiedades.add(EtiquetaScript);
PanelPropiedades.add(new JScrollPane(ListaScripts));

// Botones Aceptar y Cancelar
 JButton btnAceptar = new JButton("Aceptar");
 JButton btnCancelar = new JButton("Cancelar");
 PanelBotonesPropiedades.add(btnAceptar);
 PanelBotonesPropiedades.add(btnCancelar);

// Acciones de los botones
btnAceptar.addActionListener( new ActionListener() {
    public void actionPerformed( ActionEvent evento ) {
        // Ventana Propiedades
        VentanaPropiedades.dispose();

        switch(i){
            case 1:
                VerScript(nombreScript);
                break;
            case 2:
                EjecutarScript(nombreScript);
                break;
        }
        ReiniciarServidor();
    }
}); // fin de la llamada a addActionListener

btnCancelar.addActionListener( new ActionListener() {
    public void actionPerformed( ActionEvent evento ) {
        // Ventana Propiedades
        VentanaPropiedades.dispose();
    }
}); // fin de la llamada a addActionListener

VentanaPropiedades.getRootPane().setDefaultButton(btnAceptar);

contenedor.add(PanelPropiedades, BorderLayout.CENTER);
contenedor.add(PanelBotonesPropiedades, BorderLayout.SOUTH);

```

```

    PanelPropiedades.setBackground(color);
    PanelBotonesPropiedades.setBackground(color);

    // mostrar la ventana; igual que en AWT
    VentanaPropiedades.setVisible(true);
    VentanaPropiedades.setResizable(false);
}

private void CrearVentanaOrdenLibre() {
    final JFrame VentanaPropiedades = new JFrame("Orden Libre");
    VentanaPropiedades.setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
    VentanaPropiedades.setSize(350, 100);
    JPanel PanelPropiedades = new JPanel();
    JPanel PanelBotonesPropiedades = new JPanel();

    Container contenedor = VentanaPropiedades.getContentPane();
    VentanaPropiedades.setLocationRelativeTo(null);

    final JTextField Orden = new JTextField("Introduzca aquí la orden
        a ejecutar ",30); // un área de texto
    Orden.addFocusListener(new java.awt.event.FocusAdapter() {
        public void focusGained(java.awt.event.FocusEvent evt) {
            Orden.selectAll();
        }
    });

    PanelPropiedades.add(Orden);

    // Botones Aceptar y Cancelar
    JButton btnAceptar = new JButton("Aceptar");
    JButton btnCancelar = new JButton("Cancelar");
    PanelBotonesPropiedades.add(btnAceptar);
    PanelBotonesPropiedades.add(btnCancelar);

    VentanaPropiedades.getRootPane().setDefaultButton(btnAceptar);

    // Acciones de los botones
    btnAceptar.addActionListener( new ActionListener() {
        public void actionPerformed( ActionEvent evento ) {
            // Ventana Propiedades
            VentanaPropiedades.dispose();
            EjecutarOrdenLibre(Orden.getText().toString());
            ReiniciarServidor();
        }
    });
    // fin de la llamada a addActionListener

    btnCancelar.addActionListener( new ActionListener() {
        public void actionPerformed( ActionEvent evento ) {
            // Ventana Propiedades
            VentanaPropiedades.dispose();
        }
    });
    // fin de la llamada a addActionListener

    PanelPropiedades.setBackground(color);
    PanelBotonesPropiedades.setBackground(color);

    contenedor.add(PanelPropiedades, BorderLayout.CENTER);
    contenedor.add(PanelBotonesPropiedades, BorderLayout.SOUTH);
}

```

```

// mostrar la ventana; igual que en AWT
VentanaPropiedades.setVisible(true);
VentanaPropiedades.setResizable(false);
}

private class ManejadorBotones implements ActionListener{
    public void actionPerformed(ActionEvent evento) {
        // TODO Auto-generated method stub
        try{
            GetIP ip = new GetIP();
            direccionIP.setText("IP Pública: " + ip.IPAddress.
                toString().trim());
            dirmac= new MacAddress();
            mac.setText(" MAC: " + dirmac.Mac.toUpperCase().trim());
            addr = InetAddress.getLocalHost();
            direccionLocal.setText("IP Local: " + addr.
                getHostAddress().trim());
            salida.close();
            servidor.close();
            conexion.close();
        }catch(NullPointerException | IOException e){
        }
        ReiniciarServidor();
    }
}

private void ReiniciarServidor(){
    if(!servidor.isClosed()){
        try {
            servidor.close();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        start();
    }
}

public void start(){
    HiloEjecutarServidor = new HiloEjecutarServidor();
    HiloEjecutarServidor.start();
}

private class HiloEjecutarServidor extends Thread {
    public void run() {
        EjecutarServidor(Puerto);
    }
}

public void EjecutarServidor(int Puerto){
    try{
        servidor= new ServerSocket(Puerto);
        areaPantalla.append( "\n\n    Esperando una conexión en el
puerto " + String.valueOf(Puerto) + ".....\n\n");
        while(true){
            try {

```

```

// Aceptamos la conexión
conexion = servidor.accept();

// Informamos de la Conexión recibida desde el terminal
areaPantalla.append( "\n    Conexión " + " recibida de: "
+ conexion.getInetAddress().getHostName() + "\n");

EstablecerFlujos();

mensaje = (String) entrada.readObject();
while(true){
    ProcesarEntrada(mensaje);
    mensaje = (String) entrada.readObject();
}
} catch ( EOFException excepcionEOF ) {
    areaPantalla.append("\n\n    Se ha terminado la conexión
... Reiniciando el servidor...\n");
} catch ( ClassNotFoundException e ) {
    areaPantalla.append("\n    ClassNotFoundException (
        EjecutarServidor )");
} finally{
    //salida.close();
    //servidor.close();
    ReiniciarServidor();
}
}catch ( IOException excepcionES ) {
    //excepcionES.printStackTrace();
}

}

private void CrearCarpetaScripts(){
    DireccionScripts = System.getProperty("user.dir") + java.io.File.
        separator + "Scripts";
    File CarpetaScripts = new File(DireccionScripts);

    if(!CarpetaScripts.exists())
        CarpetaScripts.mkdir();
}

private void EstablecerFlujos(){
    // establecer flujo de salida para los objetos
    try {
        salida = new ObjectOutputStream( conexion.getOutputStream() );
        salida.flush(); // vaciar búffer de salida para enviar
        informacion de encabezado
        entrada = new ObjectInputStream( conexion.getInputStream() );
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

private void ProcesarEntrada(String orden) throws IOException{
    if(orden.matches("MainMenu"))
        Enviar(SistemaOperativo);

    if(orden.matches("TerminarConexion")){

```

```
areaPantalla.append("\n\n    Se ha terminado la conexión...\n    Reiniciando el servidor...\\n");
salida.close();
servidor.close();
}

if(orden.matches("Listado")){
    ListadoProcesos();
}

if(orden.startsWith("taskkill")){
    ProcesarTaskKillWindows(orden);
}

if(orden.matches("Navegacion")){
    ProcesarNavegacion();
}

if(orden.matches("InfDiscos")){
    InformacionDiscos();
}

if(orden.matches("Red")){
    InformacionRed();
}

if(orden.matches("Memoria")){
    InformacionMemoria();
}

if(orden.matches("SistemaOperativo")){
    InformacionSistemaOperativo();
}

if(orden.matches("OrdenLibre")){
    OrdenLibre();
}

if(orden.matches("Reiniciar")){
    Reiniciar();
}

if(orden.matches("Apagar")){
    Apagar();
}

if(orden.matches("Scripts")){
    Scripts();
}

if(orden.startsWith("EjecutarScript")){
    String fichero = mensaje.replace("EjecutarScript ", "");
    EjecutarScript(fichero);
}

if(orden.startsWith("EliminarScript")){
    String fichero = mensaje.replace("EliminarScript ", "");
    EliminarScript(fichero);
}

if(orden.equals("TransferirScript")){
    try {
        mensaje = (String) entrada.readObject();
        String nombrefichero = mensaje;
        mensaje = (String) entrada.readObject();
        String texto = mensaje;

        TransferirScript(nombrefichero, texto);
    }
}
```

```

        } catch (ClassNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    if(orden.equals("VerScript")){
        try {
            mensaje = (String) entrada.readObject();
            String nombrefichero = mensaje;

            VerScript(nombrefichero);
        } catch (ClassNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}

private void EjecutarScript(String fichero){
    try{
        areaPantalla.append("\n\t EjecutandoScript: " + fichero + "\n");
        if(SistemaOperativo.equals("Windows")){
            if(fichero.endsWith("py")){
                p = Runtime.getRuntime().exec ("cmd /c " + "cd \\" + DirecciónScripts + "\\" && python \\" + fichero + "\\");
            }
            if(fichero.endsWith("bat")){
                areaPantalla.append("\n\t Se ejecuta: cd \\" + DirecciónScripts + "\\" && \\" + fichero + "\\");
                p = Runtime.getRuntime().exec ("cmd /c " + "cd \\" + DirecciónScripts + "\\" && \\" + fichero + "\\");
            }
            else{
                p = Runtime.getRuntime().exec ("cmd /c \\" + DirecciónScripts + "\\\" + fichero + "\\");
            }
        }
        else{
            if(fichero.endsWith("bin") || fichero.endsWith("run") || fichero.endsWith("sh")){
                String [] command = {"sh", "-c", "cd \\" + DirecciónScripts + "\\" && chmod 777 " + "\\" + fichero + "\\" && ./\\" + fichero + "\\"};
                p = Runtime.getRuntime().exec (command);
            }
            if(fichero.endsWith("py")){
                String [] command = {"sh", "-c", "cd \\" + DirecciónScripts + "\\" && chmod 777 " + "\\" + fichero + "\\" && python \\" + fichero + "\\"};
                p = Runtime.getRuntime().exec (command);
            }
        }
    }catch ( IOException excepcionES ) {
        excepcionES.printStackTrace();
    }
    IndicadorPrueba=false;
}

private void EliminarScript(String fichero){
    try{
        areaPantalla.append("\n\t EliminandoScript: " + fichero + "\n");
    }
}

```

```

if(SistemaOperativo.equals("Windows"))
    p = Runtime.getRuntime().exec ("cmd /c " + del + " " +
        DireccionScripts + "\\" + fichero + "\\");
else{
    String [] command = {"sh","-c","rm -f " + "\\" + 
        DireccionScripts + "/" + fichero + "\\"};
    p = Runtime.getRuntime().exec (command);
}

if (!IndicadorPrueba)
    Enviar("FinEliminarScript");
}catch ( IOException excepcionES ) {
    excepcionES.printStackTrace();
}

IndicadorPrueba=false;
}

private void VerScript( String fichero){
    File archivo = null;
    FileReader fr = null;
    BufferedReader br = null;
    if(IndicadorPrueba){
        areaPantalla.append("\n\n");
        areaPantalla.append("*****");
        areaPantalla.append("*/");
        PRUEBAS DE
        FUNCIONES: VerScripts
        */
        areaPantalla.append("*****");
        areaPantalla.append("n");
        */
        */
    }
    areaPantalla.append("\n\tVerScript: " + fichero +"\n");

try{
    if(SistemaOperativo.equals("Windows"))
        archivo = new File (DireccionScripts+"\\"+fichero);
    else
        archivo = new File (DireccionScripts+"/"+fichero);

    fr = new FileReader (archivo);
    br = new BufferedReader(fr);

    // Lectura del fichero
    String linea;
    while((linea=br.readLine())!=null){
        if(!IndicadorPrueba)
            Enviar(linea);
        areaPantalla.append("\t" + linea + "\n");
    }
    if(!IndicadorPrueba)
        Enviar("FinLecturaScript");

}catch ( IOException excepcionES ) {
    excepcionES.printStackTrace();
}finally{
    // En el finally cerramos el fichero, para asegurarnos
    // que se cierra tanto si todo va bien como si salta
    // una excepcion.
    try{
        if( null != fr ){

```

```

        fr.close();
    }
} catch (Exception e2){
    e2.printStackTrace();
}
}

if(IndicadorPrueba){
    areaPantalla.append("\n");
    areaPantalla.append("*****");
    areaPantalla.append("\n");
}
IndicadorPrueba=false;
}

private void TransferirScript(String nombrefichero, String texto){
    areaPantalla.append("\n\tTransferirScript: " + nombrefichero + "
\n\tContenido: \n");

FileWriter fichero = null;
PrintWriter pw = null;

try {
    if(SistemaOperativo.equals("Windows"))
        fichero = new FileWriter(DireccionScripts+"\\"+nombrefichero);
    else
        fichero = new FileWriter(DireccionScripts+"/"+nombrefichero);
    ;

pw = new PrintWriter(fichero);

String data[];
data=texto.split("\n");
for(int i=0;i<data.length;i++){
    areaPantalla.append("\n\t" + data[i]);
    pw.println(data[i]);
}

} catch (Exception e) {
    e.printStackTrace();
} finally {
    try {
        // Nuevamente aprovechamos el finally para
        // asegurarnos que se cierra el fichero.
        if (null != fichero)
            fichero.close();
    } catch (Exception e2) {
        e2.printStackTrace();
    }
}
}

private void ListadoScripts(){
try {
    if(SistemaOperativo.equals("Windows")){
        p = Runtime.getRuntime().exec ("cmd /c " + "dir /B \\" + DireccionScripts + "\\");
    }else{
        String [] command = {"sh","-c","ls \\" + DireccionScripts + "}";
        p = Runtime.getRuntime().exec (command);
    }

    is = p.getInputStream();
}
}

```

```

if(SistemaOperativo.equals("Windows"))
    br = new BufferedReader (new InputStreamReader ( is , "Cp850"));
;
else
    br = new BufferedReader (new InputStreamReader ( is ));

// Se lee la primera linea
aux = br.readLine();

// Mientras se haya leido alguna linea
while (aux!=null) {
    if (!IndicadorPrueba && (aux.endsWith(".py") || aux.endsWith(
        ".sh") || aux.endsWith(".bat") || aux.endsWith(".run")
    ) || aux.endsWith(".bin"))){
        Enviar(aux);
        areaPantalla.append("\n\t" + aux );
    }
    aux = br.readLine();
}
catch ( IOException excepcionES ) {
    excepcionES.printStackTrace();
}

if (!IndicadorPrueba)
    Enviar("Fin_Scripts");

IndicadorPrueba=false;
areaPantalla.append("\n\n");
}

private void Scripts(){
    areaPantalla.append("\n\t DireccionScripts: " + DireccionScripts
        );
    areaPantalla.append("\n\n\t· Obteniendo el listado de los Scripts"
        );
    areaPantalla.append("\n____________________________________");
    ListadoScripts();
}

private void Reiniciar(){
    areaPantalla.append("\nReiniciando... \n");
    try {
        if(SistemaOperativo.equals("Windows"))
            p = Runtime.getRuntime().exec ("cmd /c shutdown -r");
        else{
            String [] command = {"sh","-c","reboot"};
            p = Runtime.getRuntime().exec (command);
        }
    } catch ( IOException e ) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

private void Apagar(){
    areaPantalla.append("\nApagando... \n");
}

```

```

try {
    if(SistemaOperativo.equals("Windows"))
        p = Runtime.getRuntime().exec ("cmd /c shutdown -s");
    else{
        String [] command = {"sh","-c","halt"};
        p = Runtime.getRuntime().exec (command);
    }
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

private void EjecutarOrdenLibre(String mensaje){
    if(IndicadorPrueba){
        areaPantalla.append(
            "*****\n");
        areaPantalla.append("/*\n");
        areaPantalla.append("*****\n");
        areaPantalla.append("*/\n");
    }

    areaPantalla.append("\n      ---> Ejecutando orden libre: " + mensaje
    );

    try {
        if(SistemaOperativo.equals("Windows"))
            p = Runtime.getRuntime().exec ("cmd /c " + mensaje);
        else{
            String [] command = {"sh","-c",mensaje};
            p = Runtime.getRuntime().exec (command);
        }

        is = p.getInputStream();

        if(SistemaOperativo.equals("Windows"))
            br = new BufferedReader (new InputStreamReader (is , "Cp850"));
        else
            br = new BufferedReader (new InputStreamReader (is));

        // Se lee la primera linea
        aux = br.readLine();

        // Mientras se haya leido alguna linea
        while (aux!=null) {
            if(!IndicadorPrueba)
                Enviar(aux);
            else
                areaPantalla.append ("\n" + aux);
            aux = br.readLine();
        }
    }catch ( IOException excepcionES ) {
        excepcionES.printStackTrace();
    }
    if(IndicadorPrueba)
        areaPantalla.append("\n\n");
    "*****\n";
}

```

```

    IndicadorPrueba=false;
}

private void OrdenLibre(){
    try {
        mensaje = (String) entrada.readObject();

        EjecutarOrdenLibre(mensaje);
    }catch ( IOException excepcionES ) {
        excepcionES.printStackTrace();
    } catch ( ClassNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    Enviar("Fin_OrdenLibre");
}

private void InformacionSistemaOperativo(){
    if(IndicadorPrueba){
        areaPantalla.append("\n\n"
            + "*****"
            + "\n");
        areaPantalla.append("/*"
            + "PRUEBAS DE"
            + " FUNCIONES: Informacion de Sistema Operativo"
            + " */\n");
        areaPantalla.append(""
            + "*****"
            + "\n");
    }else{
        areaPantalla.append("\n  ----> Obteniendo información del"
            + " Sistema Operativo...\n\n");
    }

    Vector<Propiedad> Propiedades = new Vector<Propiedad>();
    Propiedades.add(new Propiedad("Version Java" , "java.version"));
    Propiedades.add(new Propiedad("Fabricante" , "java.vendor"));
    Propiedades.add(new Propiedad("URL Fabricante" , "java.vendor.url"));
    Propiedades.add(new Propiedad("Directorio de instalación" , "java.home"));
    Propiedades.add(new Propiedad("JVM versión" , "java.vm.specification.version"));
    Propiedades.add(new Propiedad("Fabricante JVM" , "java.vm.specification.vendor"));
    Propiedades.add(new Propiedad("Nombre JVM" , "java.vm.specification.name"));
    Propiedades.add(new Propiedad("Version implementación JVM" , "java.vm.version"));
    Propiedades.add(new Propiedad("Fabricante implementación JVM" , "java.vm.vendor"));
    Propiedades.add(new Propiedad("Version implementación Java" , "java.vm.name"));
    Propiedades.add(new Propiedad("Version JRE" , "java.specification.version"));
    Propiedades.add(new Propiedad("Fabricante JRE" , "java.specification.vendor"));
    Propiedades.add(new Propiedad("Nombre JRE" , "java.specification.name"));
    Propiedades.add(new Propiedad("Número de la versión de la clase Java" , "java.class.version"));
    Propiedades.add(new Propiedad("Java Path" , "java.class.path"));
    Propiedades.add(new Propiedad("Path Librerías Java" , "java.library.path"));
    Propiedades.add(new Propiedad("Directorio temporal por defecto" ,
        "java.io.tmpdir"));
}

```

```

Propiedades.add(new Propiedad("Path del directorio de extensiones"
    , "java.ext.dirs"));
Propiedades.add(new Propiedad("Nombre del Sistema Operativo" , "os.
    name"));
Propiedades.add(new Propiedad("Arquitectura del Sistema Operativo"
    , "os.arch"));
Propiedades.add(new Propiedad("Version del Sistema Operativo" , "os
    .version"));
Propiedades.add(new Propiedad("Nombre de la cuenta de usuario" , "
    user.name"));
Propiedades.add(new Propiedad("Directorio de casa del usuario" , "
    user.home"));
Propiedades.add(new Propiedad("Directorio actual de trabajo" , "
    user.dir"));

if (!IndicadorPrueba){
    for(int i=0;i<Propiedades.size();i++){
        Enviar(Propiedades.get(i).Nombre() + ":" + System.
            getProperty(Propiedades.get(i).Orden()));
        areaPantalla.append("\t" + Propiedades.get(i).Nombre() + ":" +
            System.getProperty(Propiedades.get(i).Orden()) + "\n");
    }
    Enviar("Fin_SistemaOperativo");
} else{
    for(int i=0;i<Propiedades.size();i++){
        areaPantalla.append("\n" + Propiedades.get(i).Nombre() + ":" +
            System.getProperty(Propiedades.get(i).Orden()));
        areaPantalla.append("\n");
    }
}

if(IndicadorPrueba)
    areaPantalla.append("\n\n" +
        "/*****\n");
IndicadorPrueba=false;
}

private void InformacionMemoria(){
    String memftotal=null , memfdis=null , memvirtamax=null , memvirdis=
        null , memvirus=null ;
    if(IndicadorPrueba){
        areaPantalla.append("\n\n" +
            "/*****\n");
        areaPantalla.append("/*\n");
        PRUEBAS DE
        FUNCIONES: Informacion de Memoria
        */\n");
        areaPantalla.append("/*\n");
        areaPantalla.append("\n" +
            "/*****\n");
    } else{
        areaPantalla.append("\n    ----> Obteniendo informacion de la
            memoria ... \n");
    }

    try {
        if(SistemaOperativo.equals("Windows"))
            p = Runtime.getRuntime().exec ("cmd /c systeminfo");
        else{
            String [] command = {"sh","-c","cat /proc/meminfo | awk '{
                print $1,$2}'"};

```

```

        p = Runtime.getRuntime().exec (command);
    }
    is = p.getInputStream();

    if(SistemaOperativo.equals("Windows"))
        br = new BufferedReader (new InputStreamReader (is , "Cp850"));
    ;
    else
        br = new BufferedReader (new InputStreamReader (is));

    // Se lee la primera linea
    aux = br.readLine();

    // Mientras se haya leido alguna linea
    while (aux!=null) {
        if(SistemaOperativo.equals("Windows")){
            if(aux.startsWith("Cantidad total de memoria física")){
                String datos [];
                datos=aux.split(":");
                memftotal=datos [1];
            }

            if(aux.startsWith("Memoria física disponible")){
                String datos [];
                datos=aux.split(":");
                memfdis=datos [1];
            }

            if(aux.startsWith("Memoria virtual: tamaño")){
                String datos [];
                datos=aux.split(":");
                memvirtammax=datos [2];
            }

            if(aux.startsWith("Memoria virtual: disponible")){
                String datos [];
                datos=aux.split(":");
                memvirdis=datos [2];
            }

            if(aux.startsWith("Memoria virtual: en uso")){
                String datos [];
                datos=aux.split(":");
                memvirus=datos [2];
            }
        }else{
            //Linux
            if(aux.startsWith("MemTotal:")){
                String datos [];
                datos=aux.split(" ");
                memftotal=datos [1] + " Kb";
            }

            if(aux.startsWith("MemFree:")){
                String datos [];
                datos=aux.split(" ");
                memfdis=datos[1]+ " Kb";
            }

            if(aux.startsWith("VmallocTotal:")){
                String datos [];
                datos=aux.split(" ");
                memvirtammax=datos[1]+ " Kb";
            }

            if(aux.startsWith("VmallocUsed:")){
                String datos [];
                datos=aux.split(" ");
                memvirus=datos[1]+ " Kb";
            }

            if(aux.startsWith("VmallocChunk:")){
                String datos [];

```

```

        datos=aux.split(" ");
        memvirdis=datos[1]+ " Kb";
    }

}

aux = br.readLine();

}

if (!IndicadorPrueba){

    Enviar(memftotal);
    Enviar(memfdis);
    Enviar(memvirtamax);
    Enviar(memvirdis);
    Enviar(memvirus);
}

areaPantalla.append("\n\tMemoria Física Total: " + memftotal +
    "\n\tMemoria Física Disponible: " + memfdis + "\n\
    Memoria Virtual (Tamaño máximo): " + memvirtamax +
    "\n\tMemoria Virtual Disponible: " + memvirdis + "\n\
    Memoria Virtual Usada: " + memvirus + "\n\n");

} catch ( IOException excepcionES ) {
    excepcionES.printStackTrace();
}
if (!IndicadorPrueba)
    Enviar("Fin_Memoria");
else
    areaPantalla.append("\n\n" +
        "*****" );
IndicadorPrueba=false;
}

private void InformacionRed(){
    if(IndicadorPrueba){
        areaPantalla.append("\n\n" +
            "*****" );
        areaPantalla.append("/*" *
    FUNCIONES:   Informacion de Red
    */\n");
        areaPantalla.append("/*" *
    FUNCIONES:   Informacion de Red
    */\n");
    } else{
        areaPantalla.append("\n    ----> Obteniendo informacion de la
red... \n");
    }
    String res=null;
    //String Adaptador = null , dirfis = null , dirip = null , descripcion =
    //null , mask=null , estado=null , dhcp=null ;
    try {
        if(SistemaOperativo.equals("Windows")){
            p = Runtime.getRuntime().exec ("cmd /c ipconfig /all");
            is = p.getInputStream();
        }
    }
}

```

```

br = new BufferedReader (new InputStreamReader (is , "Cp850"))
;

// Se lee la primera linea
aux = br.readLine();

// Mientras se haya leido alguna linea
while (aux!=null) {
    try{
        while(aux != null && (!aux.startsWith("Adaptador de
            LAN") & !aux.startsWith("Adaptador de Ethernet")))
        ){
            aux=br.readLine();
        }

        if(res!=null)
            res=res+"\n";

        if(aux!=null){
            areaPantalla.append("\n\t" + aux + "\n");
            //Adaptador = aux;
            if(res!=null)
                res= res + aux + "\n";
            else
                res= aux +"\n";
            aux=br.readLine();
        }

        while(aux != null && (!aux.startsWith("Adaptador de
            LAN") & !aux.startsWith("Adaptador de Ethernet")))
        ){
            if(aux.startsWith("    Dirección física")){
                areaPantalla.append("\t" + aux + "\n");
                res= res + aux + "\n";
                //dirfis = aux;
            }

            if(aux.startsWith("    Dirección IPv4")){
                areaPantalla.append("\t" + aux + "\n");
                res= res + aux + "\n";
                //dirip = aux;
            }

            if(aux.startsWith("    Descripción")){
                areaPantalla.append("\t" + aux + "\n");
                res= res + aux + "\n";
                //descripcion = aux;
            }

            if(aux.startsWith("    Máscara")){
                areaPantalla.append("\t" + aux + "\n");
                res= res + aux + "\n";
                //mask = aux;
            }

            if(aux.startsWith("    Estado")){
                areaPantalla.append("\t" + aux + "\n");
                res= res + aux + "\n";
                //estado = aux;
            }

            if(aux.startsWith("    DHCP")){
                areaPantalla.append("\t" + aux + "\n");
                res= res + aux + "\n";
                //dhcp = aux;
            }

            if(aux.startsWith("Adaptador de túnel"))

```

```

        break;

    aux=br.readLine();
}

}catch(NullPointerException e){
    e.printStackTrace();
}

Vector<Integer> indices = new Vector<Integer>();
String[] datos;
datos = res.split("\n");
for(int i=0;i<datos.length;i++){
    if(datos[i].trim().length() == 0)
        indices.add(i);
    //areaPantalla.append("\nDatos["+i+" ]= "+ datos[i]);
}
Vector<Red> AdaptadoresRed = new Vector<Red>();

String Adaptador = null , dirfis = null , dirip = null ,
descripcion = null , mask=null , estado=null , dhcp=null ;
for(int i=0;i<indices.get(0);i++){

    if(datos[i].contains("Adaptador"))
        Adaptador=datos[i];
    if(datos[i].contains("Dirección física"))
        dirfis=datos[i];
    if(datos[i].contains("Dirección IPv4"))
        dirip=datos[i];
    if(datos[i].contains("Descripción"))
        descripcion=datos[i];
    if(datos[i].contains("Máscara"))
        mask=datos[i];
    if(datos[i].contains("Estado"))
        estado=datos[i];
    if(datos[i].contains("DHCP"))
        dhcp=datos[i];

}
AdaptadoresRed.add(new Red(Adaptador , dirfis , dirip ,
descripcion ,mask , estado ,dhcp ));

Adaptador = dirfis = dirip = descripcion = mask= estado=dhcp
=null ;

if(indices.size ()>1){

for(int i = 0 ; i<indices.size ()-1;i++){
    for(int j=indices.get(i);j<indices.get(i+1);j++){
        if(datos[j].contains("Adaptador"))
            Adaptador=datos[j];
        if(datos[j].contains("Dirección física"))
            dirfis=datos[j];
        if(datos[j].contains("Dirección IPv4"))
            dirip=datos[j];
    }
}
}

```

```

        dirip=datos[j];
        if(datos[j].contains("Descripción"))
            descripcion=datos[j];
        if(datos[j].contains("Máscara"))
            mask=datos[j];
        if(datos[j].contains("Estado"))
            estado=datos[j];
        if(datos[j].contains("DHCP"))
            dhcp=datos[j];
    }
    AdaptadoresRed.add(new Red(Adaptador, dirfis, dirip,
                               descripcion, mask, estado, dhcp));
}
}

Adaptador = dirfis = dirip = descripcion = mask= estado=dhcp
=null;

for(int i=indices.size()-1;i<datos.length;i++){
    if(datos[i].contains("Adaptador"))
        Adaptador=datos[i];
    if(datos[i].contains("Dirección física"))
        dirfis=datos[i];
    if(datos[i].contains("Dirección IPv4"))
        dirip=datos[i];
    if(datos[i].contains("Descripción"))
        descripcion=datos[i];
    if(datos[i].contains("Máscara"))
        mask=datos[i];
    if(datos[i].contains("Estado"))
        estado=datos[i];
    if(datos[i].contains("DHCP"))
        dhcp=datos[i];
}
AdaptadoresRed.add(new Red(Adaptador, dirfis, dirip,
                           descripcion, mask, estado, dhcp));

for(int i=0;i<AdaptadoresRed.size();i++){
/* areaPantalla.append("\nAdaptador("+i+")\n");
areaPantalla.append("\n" +AdaptadoresRed.get(i).
Adaptador() +
"\n" +AdaptadoresRed.get(i).DirFis() +
"\n" +AdaptadoresRed.get(i).DirIP() +
"\n" +AdaptadoresRed.get(i).Descripcion() +
"\n" +AdaptadoresRed.get(i).Mascara() +
"\n" +AdaptadoresRed.get(i).Estado() +
"\n" +AdaptadoresRed.get(i).DHCP());*/
// Enviar datos
if(!IndicadorPrueba){
    String EnviarDatos[];
    if(AdaptadoresRed.get(i).Adaptador()!=null){
        EnviarDatos = AdaptadoresRed.get(i).Adaptador().
        split(":");
        Enviar(EnviarDatos[0]);
    }else{
        Enviar("");
    }
    if(AdaptadoresRed.get(i).DirFis()!=null){
        EnviarDatos = AdaptadoresRed.get(i).DirFis().split(
            ":");
        Enviar(EnviarDatos[1]);
    }else{
        Enviar("");
    }
}
}

```

```

        if(AdaptadoresRed.get(i).DirIP() != null){
            EnviarDatos = AdaptadoresRed.get(i).DirIP().split(":");
            Enviar(EnviarDatos[1]);
        } else{
            Enviar("");
        }

        if(AdaptadoresRed.get(i).Descripcion() != null){
            EnviarDatos = AdaptadoresRed.get(i).Descripcion().
                split(":");
            Enviar(EnviarDatos[1]);
        } else{
            Enviar("");
        }

        if(AdaptadoresRed.get(i).Mascara() != null){
            EnviarDatos = AdaptadoresRed.get(i).Mascara().split(
                ":");
            Enviar(EnviarDatos[1]);
        } else{
            Enviar("");
        }

        if(AdaptadoresRed.get(i).Estado() != null){
            EnviarDatos = AdaptadoresRed.get(i).Estado().split(
                ":");
            Enviar(EnviarDatos[1]);
        } else{
            Enviar("");
        }

        if(AdaptadoresRed.get(i).DHCP() != null){
            EnviarDatos = AdaptadoresRed.get(i).DHCP().split(":");
            Enviar(EnviarDatos[1]);
        } else{
            Enviar("");
        }
    }

} else{
    // Red en Linux
    String Adaptador = null, dirfis = null, dirip = null,
        descripcion = null, mask=null, estado=null, dhcp=null;

    String[] command = {"sh", "-c", "ifconfig | grep \"Link encap",
        "\\" | awk '{print $1};\""};
    p = Runtime.getRuntime().exec(command);
    is = p.getInputStream();
    br = new BufferedReader (new InputStreamReader (is));

    // Se lee la primera linea
    aux = br.readLine();

    // Mientras se haya leido alguna linea
    while (aux!=null) {
        String[] command1 = {"sh", "-c", "ifconfig " + aux.trim()};
        Process p1 = Runtime.getRuntime().exec(command1);
        InputStream is1 = p1.getInputStream();
        BufferedReader br1 = new BufferedReader (new
            InputStreamReader (is1));
}

```

```

// Se lee la primera linea
String aux1 = br1.readLine();
// Mientras se haya leido alguna linea
while (aux1!=null) {
    if(aux1.contains("direcciónHW"))
        dirfis = aux1.substring(aux1.trim().length()-17);
    if(aux1.contains("Direc. inet")){
        String datos[] = aux1.split(" ");
        for(int cont=0;cont<datos.length;cont++){
            if(datos[cont].startsWith("inet"))
                dirip = datos[cont].substring(5);

            if(datos[cont].startsWith("Másc:"))
                mask = datos[cont].substring(5);
        }
    }
    aux1 = br1.readLine();
}

String [] command2 = {"sh","-c","ifplugstatus " + aux.trim
());
Process p2 = Runtime.getRuntime().exec(command2);
InputStream is2 = p2.getInputStream();
BufferedReader br2 = new BufferedReader (new
InputStreamReader (is2));

// Se lee la primera linea
String aux2 = br2.readLine();
// Mientras se haya leido alguna linea
while (aux2!=null) {
    if(aux2.endsWith("link beat detected"))
        estado="Conectado";
    else
        estado="Desconectado";

    aux2 = br2.readLine();
}

Adaptador = aux;
if(aux.startsWith("eth"))
    descripcion = "Adaptador alámbrico";

if(aux.startsWith("lo"))
    descripcion = "Interfaz de Retorno";

if(aux.startsWith("wlan"))
    descripcion = "Adaptador inalámbrico";

areaPantalla.append("\n\tAdaptador: " + Adaptador + "\n\
\tDirección física: " + dirfis + "\n\tDirección IP: "
+ dirip + "\n\tDescripción: " + descripcion + "\n\
\tMáscara: " + mask + "\n\tEstado: " + estado + "\n\n");
}

if(!IndicadorPrueba){
    if(Adaptador!=null)
        Enviar(Adaptador);
    else
        Enviar("");
}

if(dirfis!=null)

```

```

        Enviar( dirfis );
    else
        Enviar( "" );

    if( dirip!=null)
        Enviar( dirip );
    else
        Enviar( "" );

    if( descripcion!=null)
        Enviar( descripcion );
    else
        Enviar( "" );

    if( mask!=null)
        Enviar( mask );
    else
        Enviar( "" );

    if( estado!=null)
        Enviar( estado );
    else
        Enviar( "" );

    if( dhcp!=null)
        Enviar( dhcp );
    else
        Enviar( "" );
}

aux = br.readLine();

}

}

}catch ( IOException excepcionES ) {
    excepcionES.printStackTrace();
}
if( !IndicadorPrueba )
    Enviar("Fin_Red");
else
    areaPantalla.append("\n"
    +*****
    "\n");
IndicadorPrueba=false;
}

private void InformacionDiscos(){
    if(IndicadorPrueba){
        areaPantalla.append("\n\n"
        +*****
        "\n");
        areaPantalla.append("*
PRUEBAS DE
FUNCIONES: Informacion de Discos
*/\n");
        areaPantalla.append("*
/*****\n");
    }
    else{
        areaPantalla.append("\n"  ----> Obteniendo informacion de los
discos duros...\n\n");
    }
}

```

```

Vector<EspacioDisco> Discos = new Vector<EspacioDisco>();
String [] datos;
String TipoSisFich = null;
try {
    if(SistemaOperativo.equals("Windows"))
        p = Runtime.getRuntime().exec ("cmd /c for /F \"tokens=2*\"%a in ('fsutil fsinfo drives') do @echo %a %b");
    else{
        String [] command = {"sh","-c","df -T | grep /dev/ | awk '{\n            print $1,$2,$3,$4,$5,$6,$7}'"};
        p = Runtime.getRuntime().exec (command);
    }
    // Se obtiene el stream de salida del programa
    is = p.getInputStream();
    // Se prepara un bufferedReader para poder leer la salida más
    // comodamente.
    br = new BufferedReader (new InputStreamReader (is));
    // Se lee la primera linea
    aux = br.readLine();

    // Mientras se haya leido alguna linea
    while (aux!=null) {
        if(SistemaOperativo.equals("Windows")){
            datos = aux.split(" ");
            for(int i=0;i<datos.length ; i++){
                p = Runtime.getRuntime().exec ("cmd /c fsutil fsinfo
                    drivetype " + datos[i]);
                // Se obtiene el stream de salida del programa
                is = p.getInputStream();

                // Se prepara un bufferedReader para poder leer la
                // salida más comodamente.
                br = new BufferedReader (new InputStreamReader (is , "Cp850"));

                // Se lee la primera linea
                aux = br.readLine();
                if(aux.equals(datos[i] + " - Unidad fija")){
                    Process p1 = Runtime.getRuntime().exec ("cmd /c
                        fsutil fsinfo volumeinfo " + datos[i]);
                    // Se obtiene el stream de salida del programa
                    // Se prepara un bufferedReader para poder leer la
                    // salida más comodamente.
                    InputStream is1 = p1.getInputStream();
                    BufferedReader br1 = new BufferedReader (new
                        InputStreamReader (is1 , "Cp850"));
                    // Se lee la primera linea
                    String aux1 = br1.readLine();

                    // Mientras se haya leido alguna linea
                    while (aux1!=null) {
                        if(aux1.startsWith("Nombre del")){
                            String dat[] = aux1.split(":");
                            TipoSisFich = dat[1];
                        }
                        aux1 = br1.readLine();
                    }
                    p1 = Runtime.getRuntime().exec ("cmd /c fsutil
                        volume diskfree " + datos[i]);
                    // Se obtiene el stream de salida del programa
                    // Se prepara un bufferedReader para poder leer la
                    // salida más comodamente.
                    is1 = p1.getInputStream();
                }
            }
        }
    }
}

```

```

br1 = new BufferedReader (new InputStreamReader (
    is1 , "Cp850" ));
// Se lee la primera linea
aux1 = br1.readLine ();
String d [];
long byteslibres = 0;
long bytestotal = 0;
// Mientras se haya leido alguna linea
while (aux1!=null) {
    d=aux1.split (":");
    if(d[0].trim().endsWith("bytes libres")){
        byteslibres = Long.valueOf(d[1].trim() .
            toString ());
    }
    if(d[0].trim().endsWith("bytes")){
        bytestotal = Long.valueOf(d[1].trim() .
            toString ());
    }
    aux1 = br1.readLine ();
}
EspacioDisco aux = new EspacioDisco (datos [ i ] ,
    TipoSisFich , byteslibres , bytestotal );
Discos.add (aux );
}

}else{
// Linux
datos = aux .split (" ");
EspacioDisco aux = new EspacioDisco (datos [ 0 ] , datos [ 1 ] ,
    Long.valueOf (datos [ 4 ]) *1024 , Long.valueOf (datos [ 2 ]) *
    *1024 );
Discos.add (aux );
}

aux = br .readLine ();
}

for (int cont=0; cont< Discos .size (); cont++){
areaPantalla.append ("\n\tNombre: " + Discos .get (cont) .
    NombreDisco () + "\tTipoSistema: " + Discos .get (cont) .
    TipoSistemaDisco () + "\tTam total: " + String .valueOf (
    Discos .get (cont) .EspacioTotal ()) + "\t\tTam libre: " +
    String .valueOf (Discos .get (cont) .EspacioLibre ()) + "\n");
;
areaPantalla.append ("\t\tPorcentaje Libre: " + Discos .get (
    cont) .PorLibre () + "\t\tPorcentaje Usado: " + Discos .get (
    cont) .PorUsado () + "\n\n");
if (!IndicadorPrueba){
Enviar (Discos .get (cont) .NombreDisco ());
Enviar (Discos .get (cont) .TipoSistemaDisco ());
Enviar (String .valueOf (Discos .get (cont) .EspacioTotal ()) );
Enviar (String .valueOf (Discos .get (cont) .PorUsado ()) );
}
}

} catch ( IOException excepcionES ) {
excepcionES.printStackTrace ();
}

if (!IndicadorPrueba)
Enviar ("Fin_InfDiscos ");
else
areaPantalla.append ("\n\n"
/***** **** */

```

```

        n");
    IndicadorPrueba=false;
}

private void ListadoProcesos(){
    if(IndicadorPrueba){
        areaPantalla.append("\n\n
        /*****n*****
        n");
        areaPantalla.append("/*
        FUNCIONES: Listado de Procesos
        */\n");
        areaPantalla.append("/*
        /*****n*****
        n");
    } else{
        areaPantalla.append("\n    --> Obteniendo el listado de
        procesos activos en el sistema...\n");
    }
    areaPantalla.append("\n\tPID\tTAM.\tNombre ");
    areaPantalla.append("\n
    ");
    try {
        if(SistemaOperativo.equals("Windows")){
            p = Runtime.getRuntime().exec ("cmd /c for /F \'tokens
            =1,2,5\' % in (\\'tasklist /NH /FO TABLE ^| sort \') do
            @echo %b % %");
        } else{
            String [] command = {"sh","-c","ps -A -1 --sort cmd | awk '{
                print $4,$14,$10}\'"};
            p = Runtime.getRuntime().exec (command);
        }
        is = p.getInputStream();
        br = new BufferedReader (new InputStreamReader (is));
        // Se lee la primera linea
        aux = br.readLine();

        // Mientras se haya leido alguna linea
        while (aux!=null) {
            if(!IndicadorPrueba)
                Enviar(aux);
            String datos [];
            datos = aux.split(" ");
            if(!datos[0].contains("Idle"))
                areaPantalla.append("\n\t" + datos[0] + "\t" + datos[2]
                + "\t" + datos[1]);
            aux = br.readLine();
        }
        if(!IndicadorPrueba)
            Enviar("fin");
    } catch ( IOException excepcionES ) {
        excepcionES.printStackTrace();
    }
    if(IndicadorPrueba)
        areaPantalla.append("\n\n
        /*****n*****
        n");
}

```

```

    areaPantalla.append("\n\n");

    IndicadorPrueba=false;
}

private void ProcesarTaskKillWindows(String orden){
    String datos[];
    datos = orden.split(" ");
    areaPantalla.append("\n    --> Eliminando el proceso con PID: " +
        datos[3]);
    try{
        if(SistemaOperativo.equals("Windows")){
            p = Runtime.getRuntime().exec("cmd /c " + orden);
        }else{
            String [] command = {"sh","-c","kill " + datos[3]};
            p = Runtime.getRuntime().exec(command);
        }
        salida.flush();
        salida.writeObject("fin_elimina_proceso");
    }catch( IOException excepcionES ){
        excepcionES.printStackTrace();
    }
}

private void ProcesarNavegacion() throws IOException{
    areaPantalla.append(" \n    --> Navegación por árbol de
    directorios comenzada....\n");
    try {
        // Leemos entrada desde Navegador
        mensaje = (String) entrada.readObject();
        while(!mensaje.equals("Salir_Navegacion")){
            if(mensaje.equals("MostrarMiPc")){
                MostrarMiPc();
            }
            if(mensaje.equals("ActualizarGridView")){
                ActualizarGridView();
            }
            if(mensaje.equals("EjecutarFichero")){
                EjecutarFichero();
            }
            if(mensaje.equals("TransferirFichero")){
                TransferirFichero();
            }
            if(mensaje.equals("Copia")){
                CopiarFichero();
            }
            if(mensaje.equals("Corte")){
                CorteFichero();
            }
            if(mensaje.equals("EliminarFichero")){
                EliminarFichero();
            }
            if(mensaje.equals("RenombrarFichero")){
                RenombrarFichero();
            }
        }
    }
}
```

```

if(mensaje.equals("CrearFichero")){
    CrearFichero();
}

mensaje = (String) entrada.readObject();

areaPantalla.append("\n\n    --> Navegación por árbol de
    directorios terminada.\n\n");
} catch (IOException e1) {
    // TODO Auto-generated catch block
    areaPantalla.append("\n      Se ha terminado la conexión... ");
    start();

    e1.printStackTrace();
} catch (ClassNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
    areaPantalla.append("\nNavegacion ClassNotFoundException");
}
}

private void CrearFichero(){
    try {
        mensaje = (String) entrada.readObject();
        String Ruta=mensaje;

        mensaje = (String) entrada.readObject();
        String Fichero=mensaje;

        mensaje = (String) entrada.readObject();
        String Contenido=mensaje;
        areaPantalla.append("\n\t CrearFichero \tRuta: " + Ruta + "\\
\tFichero: " + Fichero + "\n\t Contenido:");

        FileWriter fichero = null;
        PrintWriter pw = null;
        try {
            if(SistemaOperativo.equals("Windows"))
                fichero = new FileWriter(Ruta+Fichero);
            else
                fichero = new FileWriter(Ruta+"/"+Fichero);
            pw = new PrintWriter(fichero);

            String data[];
            data=Contenido.split("\n");
            for(int i=0;i<data.length;i++){
                areaPantalla.append("\n\t" + data[i]);
                pw.println(data[i]);
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            try {
                // Nuevamente aprovechamos el finally para
                // asegurarnos que se cierra el fichero.
                if (null != fichero)
                    fichero.close();
            } catch (Exception e2) {
                e2.printStackTrace();
            }
        }
    } catch (ClassNotFoundException | IOException e) {
}
}

```

```

        // TODO Auto-generated catch block
        e.printStackTrace();
    }

    private void RenombrarFichero(){
        try {
            mensaje = (String) entrada.readObject();
            String Ruta=mensaje;

            mensaje = (String) entrada.readObject();
            String Fichero=mensaje;

            mensaje = (String) entrada.readObject();
            String NombreNuevo=mensaje;
            areaPantalla.append("\n\t Renombrar Fichero:\tFichero Origen:
                " + Ruta+Fichero + "\tNombre Nuevo: " + NombreNuevo);

            if(SistemaOperativo.equals("Windows")){
                p = Runtime.getRuntime().exec ("cmd /c " +"move /Y \\" +
                    Ruta+Fichero + "\\\" \\" + Ruta+NombreNuevo + "\\\"");
            } else{
                String [] command = {"sh","-c","mv -f " + "\\" + Ruta + "/" +
                    Fichero + "\\" + "\\" + Ruta + "/" + NombreNuevo + "\\"};
                p = Runtime.getRuntime().exec (command);
            }

            Enviar("Fichero " + Fichero + " renombrado con exito");
        } catch (IOException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        } catch (ClassNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    private void EliminarFichero(){
        try {
            mensaje = (String) entrada.readObject();
            String FicheroOrigen=mensaje;
            areaPantalla.append("\n\t Eliminar Fichero:\tFichero Origen: "
                + FicheroOrigen);

            if(SistemaOperativo.equals("Windows")){
                p = Runtime.getRuntime().exec ("cmd /c " +"del \\" +
                    FicheroOrigen + "\\\"");
            } else{
                String [] command = {"sh","-c","rm -f " + "\\" + FicheroOrigen + "\\"};
                p = Runtime.getRuntime().exec (command);
            }

            Enviar("Fichero " + FicheroOrigen + " eliminado con exito");
        } catch (IOException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        } catch (ClassNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    private void CopiarFichero(){
        try {
            mensaje = (String) entrada.readObject();

```

```

String FicheroOrigen=mensaje;
mensaje = (String) entrada.readObject();
String RutaDestino=mensaje;
areaPantalla.append("\n\t CopiarFichero:\tFichero Origen: " +
FicheroOrigen + "\tRuta Destino: " + RutaDestino);

if(SistemaOperativo.equals("Windows")){
    p = Runtime.getRuntime().exec ("cmd /c " +"copy /Y \\" +
FicheroOrigen + "\\" + RutaDestino + "\\\"");
} else{
    String [] command = {"sh","-c","cp -f " + "\\" + FicheroOrigen + "\\" + RutaDestino + "/\\"};
    p = Runtime.getRuntime().exec (command);
}

Enviar("Copia realizada con éxito");

} catch (ClassNotFoundException e) {
// TODO Auto-generated catch block
e.printStackTrace();
} catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}

private void CorteFichero(){
try {
mensaje = (String) entrada.readObject();
String FicheroOrigen=mensaje;

mensaje = (String) entrada.readObject();
String RutaDestino=mensaje;
areaPantalla.append("\n\t Cortar Fichero:\tFichero Origen: " +
FicheroOrigen + "\tRuta Destino: " + RutaDestino);

if(SistemaOperativo.equals("Windows")){
    p = Runtime.getRuntime().exec ("cmd /c " +"move /Y \\" +
FicheroOrigen + "\\" + RutaDestino + "\\\"");
} else{
    String [] command = {"sh","-c","cp -f " + "\\" + FicheroOrigen + "\\" + RutaDestino + "/\\"};
    p = Runtime.getRuntime().exec (command);
    String [] command1 = {"sh","-c","rm -f " + "\\" + FicheroOrigen + "\\\""};
    p = Runtime.getRuntime().exec (command1);
}

Enviar("Pegado realizado con éxito");

} catch (ClassNotFoundException e) {
// TODO Auto-generated catch block
e.printStackTrace();
} catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}

private void EjecutarFichero(){
try {
mensaje = (String) entrada.readObject();

```

```

String Ruta=mensaje;
mensaje = (String) entrada.readObject();
String FicheroEjecutable=mensaje;
mensaje = (String) entrada.readObject();
String argumentos=mensaje;
areaPantalla.append("\n\t Ejecutar Fichero:\tRuta: " + Ruta +
"\tEjecutable: " + FicheroEjecutable + "\tArgumentos: " +
argumentos);

// Compilamos y ejecutamos
String eje [] = FicheroEjecutable.split("\\.");
String Ejecutable = eje[0];
String Extension = eje[1];

if(SistemaOperativo.equals("Windows")){
    // Ejecución de Ficheros .java en Windows
    if(Extension.equals("java")){
        areaPantalla.append("\n\t ·Se ejecuta: cmd /c cd \"\" + "
            Ruta + "\" && " + "javac \"\" + Ejecutable + "." + "
            Extension + "\" && java \"\" + Ejecutable + "\" " +
            argumentos);
        p = Runtime.getRuntime().exec ("cmd /c cd \"\" + Ruta + "
            "\" && " + "javac \"\" + Ejecutable + "." + Extension
            + "\" && java \"\" + Ejecutable + "\" " + argumentos);
    }
    // Ejecución de Ficheros .exe y .bat en Windows
    if(Extension.equals("exe") || Extension.equals("bat")){
        areaPantalla.append("\n\t ·Se ejecuta: cmd /c " + "\" + "
            Ruta + Ejecutable + "." + Extension + "\" " +
            argumentos);
        p = Runtime.getRuntime().exec ("cmd /c " + "\" + Ruta +
            Ejecutable + "." + Extension + "\" " + argumentos);
    }
    // Ejecución de Ficheros .py en Windows
    if(Extension.equals("py")){
        areaPantalla.append("\n\t ·Se ejecuta: cmd /c " + "cd \"\" +
            Ruta + "\" && python \"\" + FicheroEjecutable + "
            "\" " + argumentos);
        p = Runtime.getRuntime().exec ("cmd /c " + "cd \"\" + Ruta
            + "\" && python \"\" + FicheroEjecutable + "\" " +
            argumentos );
    }
} else{
    // Ejecución de Ficheros .java en Linux
    if(Extension.equals("java")){
        areaPantalla.append("\n\t ·Se ejecuta: javac \"\" + Ruta +
            "/" + Ejecutable + ".java\"\" && cd \"\" + Ruta + "\""
            + "&& " + "java \"\" + Ejecutable + "\" " + argumentos);
        String [] command = {"sh","-c","javac \"\" + Ruta + "/" + "
            Ejecutable + ".java\"\" && cd \"\" + Ruta + "\" && "
            "java \"\" + Ejecutable + "\" " + argumentos};
        p = Runtime.getRuntime().exec (command);
    }
    // Ejecución de Ficheros .bin, .run y .sh en Linux
    if(Extension.equals("bin") || Extension.equals("run") ||
        Extension.equals("sh")){
        areaPantalla.append("\n\t ·Se ejecuta: cd \"\" + Ruta + "
            "\" && ./\"\" + FicheroEjecutable + "\" " + argumentos
            );
        String [] command = {"sh","-c","cd \"\" + Ruta + \"\" &&
            "./\"\" + FicheroEjecutable + "\" " + argumentos};
        p = Runtime.getRuntime().exec (command);
    }
    // Ejecución de Ficheros .py en Linux
}

```

```

        if(Extension.equals("py")){
            areaPantalla.append("\n\t ·Se ejecuta: cd \\" + Ruta + "
                +" && python \\" + FicheroEjecutable + "\\" + +
                argumentos);
            String [] command = {"sh","-c","cd \\" + Ruta + "\\" &&
                python \\" + FicheroEjecutable + "\\" + argumentos
                };
            p = Runtime.getRuntime().exec (command);
        }
    }

} catch (ClassNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

}

private void ActualizarGridView(){
try {
    mensaje = (String) entrada.readObject();
    areaPantalla.append("\n\t (ActualizarGridView) Directorio: " +
        mensaje);
    if(SistemaOperativo.equals("Windows"))
        p = Runtime.getRuntime().exec ("cmd /c for /F \"tokens=3*\"%
            % in ('dir " + "\" + mensaje + "\" + "\\') do @echo
            % %");
    else{
        String [] command = {"sh","-c","ls -l \"/" + mensaje + "\\" |
            awk '{ for (x=1; x<NF; x++) { if(x==1 || x>=9){printf
            $x \" \\";} }; print \"\\\" }'\""};
        p = Runtime.getRuntime().exec (command);
    }
    // Se obtiene el stream de salida del programa
    is = p.getInputStream();
    // Se prepara un bufferedReader para poder leer la salida más
    comodamente.
    if(SistemaOperativo.equals("Windows"))
        br = new BufferedReader (new InputStreamReader (is , "Cp850"));
    ;
    else{
        br = new BufferedReader (new InputStreamReader (is));
    }
    // Se lee la primera linea
    aux = br.readLine();
    // Mientras se haya leido alguna linea
    if(SistemaOperativo.equals("Windows")){
        while (aux!=null) {
            if(aux.endsWith("bytes") || aux.endsWith("bytes libres"))
            {
            }else{
                salida.flush();
                salida.writeObject(aux);
            }
            aux = br.readLine();
        }
    }else{
        while (aux!=null) {
            if(!aux.startsWith("total")){
                salida.flush();
                salida.writeObject(aux);
            }
            aux = br.readLine();
        }
    }
}
}

```

```

    }

    salida.flush();
    salida.writeObject("fin_listado_dir");

} catch (ClassNotFoundException | IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
}

private void MostrarMiPc(){
// Mostramos MiPC (unidades)

String [] datos;
try {
    areaPantalla.append("\n\t Mostrand MiPc... ");
    if(SistemaOperativo.equals("Windows"))
        p = Runtime.getRuntime().exec ("cmd /c for /F \"tokens=2*\"%a in ('fsutil fsinfo drives') do @echo %a %b");
    else{
        String [] command = {"sh","-c","ls -l | awk '{ print $1,$9}'"};
        p = Runtime.getRuntime().exec (command);
    }

    // Se obtiene el stream de salida del programa
    is = p.getInputStream();
    // Se prepara un bufferedReader para poder leer la salida más
    // comodamente.
    br = new BufferedReader (new InputStreamReader (is));
    // Se lee la primera linea
    aux = br.readLine();

    // Mientras se haya leido alguna linea
    while (aux!=null) {
        datos = aux.split(" ");
        if(SistemaOperativo.equals("Windows")){
            for(int i=0;i<datos.length; i++){
                p = Runtime.getRuntime().exec ("cmd /c fsutil fsinfo
                    drivetype " + datos[i]);
                // Se obtiene el stream de salida del programa
                is = p.getInputStream();

                // Se prepara un bufferedReader para poder leer la
                // salida más comodamente.
                br = new BufferedReader (new InputStreamReader (is,
                    Cp850"));

                // Se lee la primera linea
                aux = br.readLine();
                if(aux.equals(datos[i] + " - Unidad fija")){
                    salida.flush();
                    salida.writeObject(datos[i] + " Unidadfija");
                }
                if(aux.equals(datos[i] + " - Unidad de CD-ROM")){
                    salida.flush();
                    salida.writeObject(datos[i] + " CD-ROM");
                }
            }
        }else{
            if(!datos[0].startsWith("total")){
                salida.flush();
                salida.writeObject(datos[1] + " " + datos[0]);
            }
        }
    }
}
}

```

```

        aux = br.readLine();
    }

    Enviar("Fin_MostrarMiPc");
} catch ( IOException excepcionES ) {
    excepcionES.printStackTrace();
}

private class Transferirfichero implements Runnable{
    private String Ruta;
    private String Fichero;
    private int PuertoT;

    Transferirfichero(String r, String f, int pto){
        Ruta=r;
        Fichero=f;
        PuertoT=pto;
        try {
            servidortrans= new ServerSocket(PuertoT);
        } catch ( IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        Thread t=new Thread (this);
        t.start();
    }

    @Override
    public void run() {
        // TODO Auto-generated method stub
        areaPantalla.append( "\n\n    Esperando una conexión en el
                            puerto " + String.valueOf(PuertoT) + "....\n\n");
        Enviar("ListoTransferencia:" + PuertoT);
        try{
            conexiontrans = servidortrans.accept();
            // Informamos de la Conexión recibida desde el terminal

            areaPantalla.append( "\n    Conexión trans" + " recibida de:
                            " + conexiontrans.getInetAddress().getHostName() + "\n"
                            );
            areaPantalla.append("\n\t Transferencia de fichero \tRuta:
                            " + Ruta + "\tEjecutable: " + Fichero);

            final BufferedOutputStream outStream = new
                BufferedOutputStream(conexiontrans.getOutputStream());
            final BufferedInputStream inStream;
            if(SistemaOperativo.equals("Windows"))
                inStream = new BufferedInputStream(new FileInputStream(
                    Ruta+Fichero));
            else
                inStream = new BufferedInputStream(new FileInputStream(
                    Ruta+"/"+Fichero));

            final byte[] buffer = new byte[4096];
            for (int read = inStream.read(buffer); read >= 0; read =
                inStream.read(buffer))
                outStream.write(buffer, 0, read);

            inStream.close();
            outStream.close();
            servidortrans.close();
        } catch ( IOException e) {
    }
}

```

```

        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

private void TransferirFichero(){
    try {
        mensaje = (String) entrada.readObject();
        String Ruta=mensaje;

        mensaje = (String) entrada.readObject();
        String Fichero=mensaje;

        new Transferirfichero(Ruta,Fichero,PuertoTransferencia);
    } catch (ClassNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (IOException e1) {
        // TODO Auto-generated catch block
        e1.printStackTrace();
    }
}

private void Enviar(String n){
    try {
        salida.flush();
        salida.writeObject(n);
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

private void InicializaServidor(){
    CrearCarpetaScripts();

    areaPantalla.append( "\n    Bienvenido al Servidor RemSys... ");
    areaPantalla.append( "\n    Con este servidor podrá obtener
                     información de este sistema en su móvil en todo momento
                     mediante la red");
    areaPantalla.append( "\n    El puerto de escucha está configurado
                     en el " + Puerto + " y el de transferencia en el: " +
                     PuertoTransferencia);
    areaPantalla.append( "\n    Para configurar el puerto de escucha/
                     Transferencia del servidor, seleccione Configuracion...
                     Configurar puerto");
    areaPantalla.append( "\n    Recuerde que además debe de configurar
                     el router para redirigir las peticiones entrantes de dicho
                     puerto a la ip local del sistema... ");
    areaPantalla.append( "\n    Para otras consultas, por favor, véase
                     el manual... \n");
    areaPantalla.setAutoscrolls(true);
}

public static void main( String args[] )
{
    Servidor aplicacion = new Servidor();
    aplicacion.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
    aplicacion.InicializaServidor();
    aplicacion.start();
}
// fin de la clase Servidor

```

## 12.2. Código cliente Remsys

Conexion.java

---

```

package garciaponce.miguel;

import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;
import android.app.Activity;
import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.util.Log;

public class Conexion extends Service {
    public static Activity ACTIVIDAD;
    public static Socket conexion=null;
    public static ObjectOutputStream salida;
    public static ObjectInputStream entrada;
    private static String IP;
    private static int sckt;
    private static boolean conectado=false;

    public static void establecerIP(String ip){
        IP=ip;
    }

    public static String ObtenerIP(){
        return IP;
    }

    public static boolean ObtenerEstado(){
        return conectado;
    }

    public static int ObtenerPuerto(){
        return sckt;
    }
    public static void establecerSocket(int socket){
        sckt=socket;
    }

    public static void establecerActividadPrincipal(Activity actividad)
    {
        Conexion.ACTIVIDAD=actividad;
    }

    public void onCreate()
    {
        super.onCreate();
        // Iniciamos el servicio
        Conexion.iniciarServicio();
        Log.i(getClass().getSimpleName() , "Servicio iniciado");
    }

    public void onDestroy()
    {
        super.onDestroy();
    }
}

```

```

// Detenemos el servicio
Conexion.finalizarServicio();

Log.i(getClass().getSimpleName(), "Servicio detenido");

public IBinder onBind(Intent intent)
{
    // No usado de momento, sólo se usa si se va a utilizar IPC
    // (Inter-Process Communication) para comunicarse entre procesos
    return null;
}

public static void iniciarServicio()
{
    try
    {
        // Conectamos y obtenemos flujos.
        conexion = new Socket(IP, sckt);
        conectado=true;
        ObtenerFlujos();
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}

public static void ObtenerFlujos(){
    try {
        salida = new ObjectOutputStream( conexion.getOutputStream() );
        salida.flush();
        entrada = new ObjectInputStream( conexion.getInputStream() );
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

public static void finalizarServicio()
{
    try
    {
        conexion.close();
    }
    catch(Exception e)
    {
    }
}
}

```

## CrearFichero.java

---

```

package garciaponce.miguel;

import java.io.IOException;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

```

```

public class CrearFichero extends Activity {
    private Button CrearFich;
    private String ruta;
    private EditText NombreFichero;
    private EditText ContenidoFichero;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.crearfichero);
        Conexion.ACTIVIDAD=this;

        // Si le damos al botón crear fichero
        CrearFich = (Button) findViewById(R.id.btncrearfich);
        CrearFich.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                // Obtenemos la ruta
                Intent intent =getIntent();
                ruta =intent.getStringExtra("Ruta");
                // Obtenemos los datos de nombre del fichero y su
                // contenido
                NombreFichero = (EditText) findViewById(R.id.Nfich);
                ContenidoFichero = (EditText) findViewById(R.id.ConFich)
                ;

                // Mandamos los datos al servidor.
                try {
                    Conexion.salida.flush();
                    Conexion.salida.writeObject("CrearFichero");
                    Conexion.salida.flush();
                    Conexion.salida.writeObject(ruta);
                    Conexion.salida.flush();
                    Conexion.salida.writeObject(NombreFichero.getText()
                        .toString());
                    Conexion.salida.flush();
                    Conexion.salida.writeObject(ContenidoFichero.getText()
                        .toString());
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
                // Volvemos al layout anterior, pasandole la ruta para
                // que la actualice.
                Intent intent1 = new Intent(CrearFichero.this, Navegador
                    .class);
                intent1.putExtra("IndicadorCrearFichero", true);
                intent1.putExtra("Ruta", ruta);
                startActivity(intent1);
            }
        });
    }
}

```

CrearHost.java

---

```

package garciaponce.miguel;

import android.app.Activity;
import android.content.Intent;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

```

```

public class CrearHost extends Activity {
    private EditText NombreMaquina;
    private EditText DirIP;
    private EditText Puerto;
    private EditText DirMac;
    private Button AgregaHost;
    private String NombreBD = "BDRemSys";
    private HostsSQL usdbh;
    private SQLiteDatabase db;
    private Toast toast;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.creahost);

        // Accedemos a la BD.
        usdbh = new HostsSQL(CrearHost.this, NombreBD, null, 1);
        db = usdbh.getWritableDatabase();

        // Si pulsamos sobre agregarHost
        AgregaHost = (Button) findViewById(R.id.CreaHost);
        AgregaHost.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                // Obtenemos los datos y componentes.
                String Usuario = SesiónLocal.NombreUsuario;
                NombreMaquina = (EditText) findViewById(R.id.NOMBREMAQUINA);
                DirIP = (EditText) findViewById(R.id.DirIP);
                Puerto = (EditText) findViewById(R.id.Puerto);
                DirMac = (EditText) findViewById(R.id.DirMAC);

                // Ejecutamos la sentencia SQL para agregarla a la base
                // de datos a la tabla Host.
                db.execSQL("INSERT INTO Hosts (usuario,nombrehost,dirip,
                    puerto,dirmac) VALUES ('" + Usuario.trim() +
                    "','" + NombreMaquina.getText().toString().trim() +
                    "','" + DirIP.getText().toString().trim() +
                    "','" + Puerto.getText().toString().trim() +
                    "','" + DirMac.getText().toString().trim() +
                    "')");

                // Visualizamos un toast para informar de la acción
                // anterior.
                toast = Toast.makeText(getApplicationContext(),"Host
                    creado correctamente", Toast.LENGTHSHORT);
                toast.show();

                // Cerramos la base de datos.
                db.close();

                // Volvemos al Layout anterior.
                Intent intent = new Intent(CrearHost.this, ListaHosts.
                    getClass());
                startActivity(intent);
            }
        });
    }
}

```

CrearScript.java

---

```
package garciaponce.miguel;
```

```
import java.io.IOException;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;

public class CrearScript extends Activity {

    private Button TransferirScript;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.creascript);
        Conexion.ACTIVIDAD=this;

        // Si pulsamos sobre transferirScript
        TransferirScript = (Button) findViewById(R.id.TransferirScript)
        ;
        TransferirScript.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                // Pedimos el nombre del script
                final AlertDialog.Builder alert = new AlertDialog.Builder(
                        CrearScript.this);
                alert.setMessage("Nombre del Script: ");
                final EditText input = new EditText(CrearScript.this);
                alert.setView(input);
                // Si pulsamos sobre Transferir con los datos
                // introducidos.
                alert.setPositiveButton("Transferir", new DialogInterface.
                        OnClickListener() {
                    public void onClick(DialogInterface dialog, int
                            whichButton) {
                        // Mandamos datos al servidor.
                        try{
                            EditText TextoScript = (EditText) findViewById(R
                                    .id.TextoScript);
                            Conexion.salida.flush();
                            Conexion.salida.writeObject("TransferirScript");
                            Conexion.salida.flush();
                            Conexion.salida.writeObject(input.getText().
                                    toString());
                            Conexion.salida.flush();
                            Conexion.salida.writeObject(TextoScript.getText
                                    () .toString());
                        }
                        catch (IOException e) {
                            // TODO Auto-generated catch block
                            e.printStackTrace();
                        } finally{
                            // Al acabar, volvemos al layout Scripts
                            Intent intent = new Intent(CrearScript.this ,
                                    Scripts.class);
                            startActivity(intent);
                        }
                    }
                });
            }
        });

        alert.setNegativeButton("Cancelar",
                new DialogInterface.OnClickListener() {
```

```

        public void onClick(DialogInterface dialog ,
                           int whichButton) {
            dialog.cancel();
        });
    });
}
}

```

## CrearUsuario.java

---

```

package garciaponce.miguel;

import android.app.Activity;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class CrearUsuario extends Activity {
    private EditText Usuario;
    private EditText Contrasena;
    private EditText RepContrasena;
    private Button CrearUsu;
    private String NombreBD = "BDRemSys";
    private String NombreTabla = "Usuarios";
    private UsuariosSQL usdbh;
    private SQLiteDatabase db;
    private Cursor c;
    private Toast toast;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.creausuario);

        CrearUsu = (Button) findViewById(R.id.BotonCreaUsu);

        // Cuando le clickeemos en el boton crearUsuario
        CrearUsu.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                Usuario = (EditText) findViewById(R.id.editTextUsu);
                Contrasena = (EditText) findViewById(R.id.editTextCont);
                RepContrasena = (EditText) findViewById(R.id.
                    editTextRepCont);

                // Comprobamos que el usuario no exista.
                usdbh = new UsuariosSQL(CrearUsuario.this, NombreBD ,
                    null, 1);
                db = usdbh.getWritableDatabase();
                db.execSQL("CREATE TABLE IF NOT EXISTS Usuarios (nombre
                    TEXT, contrasena TEXT , PRIMARY KEY(nombre))");

                c = db.rawQuery(" SELECT nombre,contrasena FROM Usuarios
                    WHERE nombre='\'' + Usuario.getText().toString() +
                    '\'', null);
            }
        });
    }
}

```

```

switch(c.getCount()){
    case 1:
        Log.i("Fila BD ", "Ya existe usuario");
        toast = Toast.makeText(getApplicationContext(),"El
            usuario ya existe.", Toast.LENGTHSHORT);
        toast.show();

        break;
    case 0:
        Log.i("Fila BD ", "No existe usuario");
        if(Contrasena.getText().toString().equals(
            RepContrasena.getText().toString())){
            //Si hemos abierto correctamente la base de
            datos
            if(db != null)
            {
                db.execSQL("INSERT INTO " + NombreTabla +
                    " (nombre,contrasena) VALUES ('" +
                    Usuario.getText().toString().trim() +
                    "','" + Contrasena.getText().toString().trim() +
                    ")");
            }
            toast = Toast.makeText(getApplicationContext()
                () ,"Usuario creado con éxito.", Toast.
                LENGTHSHORT);
            toast.show();

            Intent intent = new Intent(CrearUsuario.this
                , Login.class);
            startActivity(intent);

        }else{
            toast = Toast.makeText(getApplicationContext()
                () ,"Las contraseñas no coinciden", Toast.
                LENGTHSHORT);
            toast.show();
        }
        break;
    default:
        Log.i("Fila BD ", "Usuario duplicado");
        break;
}
db.close();
}

);
}
}

```

InfDiscos.java

---

```

package garciaponce.miguel;

import java.io.IOException;
import java.util.Vector;

import android.app.Activity;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;

```

```

public class InfDiscos extends Activity {
    private Vector<Disco> discos;
    private ListView lstDiscos;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.infdiscos);
        Conexion.ACTIVIDAD=this;

        discos = new Vector<Disco>();

        try {
            // Enviamos el mensaje Navegacion para obtener las unidades (Mi
            // PC) y empezar a visualizarlas.
            Conexion.salida.flush();
            Conexion.salida.writeObject("InfDiscos");

            String mensaje = (String) Conexion.entrada.readObject();
            // Mientras no recibamos "FIN_InfDiscos", obtenemos los datos
            // de nombre, tipo, ocupado y tamaño total.
            while(!mensaje.equals("Fin_InfDiscos")){
                String nombre = mensaje;
                mensaje = (String) Conexion.entrada.readObject();
                String Tipo = mensaje;
                mensaje = (String) Conexion.entrada.readObject();
                int TamTotal = Integer.parseInt(mensaje);
                mensaje = (String) Conexion.entrada.readObject();
                int Ocupado = Integer.parseInt(mensaje);
                Disco aux = new Disco(nombre,Tipo,Ocupado,TamTotal);
                discos.add(aux);
                mensaje = (String) Conexion.entrada.readObject();
            }

            // Establecemos adaptador de Discos y registramos
            // MenuContextual.
            AdaptadorDisco adaptador = new AdaptadorDisco(this);
            lstDiscos = (ListView)findViewById(R.id.ListaDiscos);
            lstDiscos.setAdapter(adaptador);
            registerForContextMenu(lstDiscos);

        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}

class AdaptadorDisco extends ArrayAdapter {
    Activity context;

    AdaptadorDisco(Activity context) {
        super(context, R.layout.disco, discos);
        this.context = context;
    }

    public View getView(int position, View convertView, ViewGroup
    parent)
    {
        View item = convertView;

```

```

ViewHolder holder;
if(item == null)
{
    LayoutInflator inflater = context.getLayoutInflater();
    item = inflater.inflate(R.layout.disco, null);

    holder = new ViewHolder();
    holder.Nombre = (TextView)item.findViewById(R.id.NombreDisco);
    holder.Tipo = (TextView)item.findViewById(R.id.TipoDisco);
    holder.PorOcupado = (TextView)item.findViewById(R.id.PorcentajeOcupado);
    holder.Tam = (TextView)item.findViewById(R.id.TamTotal);

    item.setTag(holder);
}
else
{
    holder = (ViewHolder)item.getTag();
}

holder.Nombre.setText(discos.elementAt(position).NombreDisco());
holder.Tipo.setText(discos.elementAt(position).TipoSistemaDisco());
holder.PorOcupado.setText(String.valueOf(discos.elementAt(position).PorUsado()) + "%");
holder.Tam.setText(String.valueOf(discos.elementAt(position).Tamanio())+ "Gb");
return(item);
}

static class ViewHolder {
    public TextView Nombre;
    public TextView Tipo;
    public TextView PorOcupado;
    public TextView Tam;
}
}

```

---

InfMem.java

---

```

package garciaponce.miguel;

import java.io.IOException;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;

public class InfMem extends Activity {
    private String memftotal=null, memfdis=null, memvirtammax=null,
               memvirdis=null, memvirus=null;
    private TextView MemFisTot;
    private TextView MemFisDis;
    private TextView MemVirTot;
    private TextView MemVirDis;
    private TextView MemVirUs;
    private Button btnKb;
    private Button btnMb;

    /** Called when the activity is first created. */
    @Override

```

```

public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.infmem);
    Conexion.ACTIVIDAD=this;

    // Obtenemos los componentes
    MemFisTot = (TextView) findViewById(R.id.MemFisTot);
    MemFisDis =(TextView) findViewById(R.id.MemFisDis);
    MemVirTot = (TextView) findViewById(R.id.MemVirTot);
    MemVirDis = (TextView) findViewById(R.id.MemVirDis);
    MemVirUs = (TextView) findViewById(R.id.MemVirUs);

    btnKb = (Button) findViewById(R.id.btnKb);
    btnMb = (Button) findViewById(R.id.btnMb);

    // Mandamos la orden de MEmoria
    try {
        Conexion.salida.flush();
        Conexion.salida.writeObject("Memoria");

        // Obtenemos los datos del servidor
        String mensaje = (String) Conexion.entrada.readObject();
        while(!mensaje.equals("Fin_Memoria")){
            memftotal=mensaje;
            mensaje = (String) Conexion.entrada.readObject();
            memfdis=mensaje;
            mensaje = (String) Conexion.entrada.readObject();
            memvirtamax=mensaje;
            mensaje = (String) Conexion.entrada.readObject();
            memvirdis=mensaje;
            mensaje = (String) Conexion.entrada.readObject();
            memvirus=mensaje;
            mensaje = (String) Conexion.entrada.readObject();

        }
        // Dependiendo de el sistema operativo que tenga el sistema
        // remoto, vamos a activar o desactivar KB o MB.
        if(SesionLocal.ObtenerSistemaOperativo().equals("Linux"))
            btnKb.setEnabled(false);
        else
            btnMb.setEnabled(false);

        MemFisTot.setText(memftotal.trim());
        MemFisDis.setText(memfdis.trim());
        MemVirTot.setText(memvirtamax.trim());
        MemVirDis.setText(memvirdis.trim());
        MemVirUs.setText(memvirus.trim());

        // Actuamos segun la pulsacion de los botones KB y Mb, segun
        // linux o Windows.
        btnMb.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                btnMb.setEnabled(false);
                btnKb.setEnabled(true);

                if(SesionLocal.ObtenerSistemaOperativo().equals("Linux
                    ")){
                    int res= Integer.valueOf(memftotal.substring(0,
                        memftotal.length()-3))/1024;
                    MemFisTot.setText(String.valueOf(res)+" Mb");
                    res= Integer.valueOf(memfdis.substring(0, memfdis.
                        length()-3))/1024;
                    MemFisDis.setText(String.valueOf(res)+" Mb");
                    res= Integer.valueOf(memvirtamax.substring(0,
                        memvirtamax.length()-3))/1024;
                    MemVirTot.setText(String.valueOf(res)+" Mb");
                }
            }
        });
    }
}

```



InfProcesos.java


---

```

package garciaponce.miguel;

import java.io.IOException;
import java.util.Vector;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.ContextMenu;
import android.view.ContextMenu.ContextMenuItemInfo;
import android.view.KeyEvent;
import android.view.LayoutInflater;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemLongClickListener;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

public class InfProcesos extends Activity {
    private ListView lstOpciones;
    Vector<Proceso> procesos;
    private int pos;
    Toast toast;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.infprocesos);

        // Creamos la estructura de datos.
        procesos = new Vector<Proceso>();

        // Mandamos orden de listar procesos
        try {
            Conexion.salida.flush();
            Conexion.salida.writeObject("Listado");
            String mensaje = (String) Conexion.entrada.readObject();
            String[] datos;
            // Obtenemos los datos, introduciendolos en el vector.
            while(!mensaje.equals("fin")){
                datos = mensaje.split(" ");
                if(!datos[0].equals("PID") && !datos[1].equals("CMD") &&
                   !datos[1].equals("SZ")){
                    Proceso aux = new Proceso(datos[0],datos[1],datos[2]);
                    procesos.add(aux);
                }
            }
            mensaje = (String) Conexion.entrada.readObject();
        }

        // Establecemos el adaptador de procesos y registramos menu
        // contextual
        AdaptadorProcesos adaptador = new AdaptadorProcesos(this);
        lstOpciones = (ListView)findViewById(R.id.ListaProcesos);
        lstOpciones.setAdapter(adaptador);
        registerForContextMenu(lstOpciones);

        // Si hacemos una pulsacion larga, registramos la posicion.
        lstOpciones.setOnItemLongClickListener(new
            OnItemLongClickListener() {
                public boolean onItemLongClick(AdapterView<?> parent,
                    final View v, int position, long id) {

```

```

        // record position/id/whatever here
        pos=position;
        return false;
    });
}

} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ClassNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
}

@Override
public void onCreateContextMenu(ContextMenu menu, View v,
        ContextMenuItemInfo menuInfo)
{
    super.onCreateContextMenu(menu, v, menuInfo);

    MenuInflater inflater = getMenuInflater();

    if(v.getId() == R.id.ListaProcesos)
    {
        inflater.inflate(R.menu.menuprocesos, menu);
    }
}

// Actuamos segun el menu contextual: eliminar proceso.
@Override
public boolean onContextItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.menueliminarproceso:
            Toast toast1 = Toast.makeText(getApplicationContext(),"Se ha
                mandado la orden para eliminar el proceso " + procesos
                .get(pos).getNom(), Toast.LENGTH_LONG);
            toast1.show();
            try {
                Conexion.salida.flush();
                Conexion.salida.writeObject("taskkill /F /PID " +
                    procesos.get(pos).getPID());
                String mensaje = (String) Conexion.entrada.readObject();
                while(!mensaje.equals("fin_elimina_proceso")){
                    mensaje = (String) Conexion.entrada.readObject();
                }
                toast1 = Toast.makeText(getApplicationContext(),"Se ha
                    eliminado el proceso ", Toast.LENGTH_LONG);
                toast1.show();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (ClassNotFoundException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            // Cuando se elimina el proceso, recargamos el layout con
            // los nuevos datos.(sin el proceso ya eliminado)
            Intent intent = new Intent(InfProcesos.this , InfProcesos.
                getClass());
            startActivity(intent);
            return true;
        default:
            return super.onContextItemSelected(item);
    }
}

```

```

@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    if ((keyCode == KeyEvent.KEYCODEBACK)) {
        Intent intent = new Intent(InfProcesos.this, Mainmenu.class);
        startActivity(intent);
    }
    return super.onKeyDown(keyCode, event);
}

class AdaptadorProcesos extends ArrayAdapter {
    Activity context;

    AdaptadorProcesos(Activity context) {
        super(context, R.layout.proc, procesos);
        this.context = context;
    }

    public View getView(int position, View convertView, ViewGroup
        parent)
    {
        View item = convertView;
        ViewHolder holder;

        if(item == null)
        {
            LayoutInflater inflater = context.getLayoutInflater();
            item = inflater.inflate(R.layout.proc, null);

            holder = new ViewHolder();
            holder.DatoProc1 = (TextView)item.findViewById(R.id.
                DatoProc1);
            holder.DatoProc2 = (TextView)item.findViewById(R.id.
                DatoProc2);
            holder.DatoProc3 = (TextView)item.findViewById(R.id.
                DatoProc3);

            item.setTag(holder);
        }
        else
        {
            holder = (ViewHolder)item.getTag();
        }

        holder.DatoProc1.setText(procesos.elementAt(position).getNom());
        holder.DatoProc2.setText(procesos.elementAt(position).getTam()
            + " Kb");
        holder.DatoProc3.setText("PID: " + procesos.elementAt(position)
            .getPID());

        return(item);
    }

    static class ViewHolder {
        public TextView DatoProc1;
        public TextView DatoProc2;
        public TextView DatoProc3;
    }
}

package garciaponce.miguel;

```

InfRed.java

```

import java.io.IOException;
import java.util.Vector;

import android.app.Activity;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.ListView;
import android.widget.TextView;

public class InfRed extends Activity {
    private Vector<Red> DispRed;
    private ListView lstDispRed;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.infred);
        Conexion.ACTIVIDAD=this;

        //Creamos la estructura de datos necesaria
        DispRed = new Vector<Red>();

        // Enviamos orden al servidor para obtener datos de red
        try {
            Conexion.salida.flush();
            Conexion.salida.writeObject("Red");

            // Obtenemos los datos del servidor.
            String mensaje = (String) Conexion.entrada.readObject();
            while(!mensaje.equals("Fin_Red")){
                // Obtenemos los valores desde el servidor.
                String Adaptador = null, dirfis = null, dirip = null,
                    descripcion = null, mask=null, estado=null, dhcp=null;
                Adaptador=mensaje;
                mensaje = (String) Conexion.entrada.readObject();
                dirfis=mensaje;
                mensaje = (String) Conexion.entrada.readObject();
                dirip=mensaje;
                mensaje = (String) Conexion.entrada.readObject();
                descripcion=mensaje;
                mensaje = (String) Conexion.entrada.readObject();
                mask=mensaje;
                mensaje = (String) Conexion.entrada.readObject();
                estado=mensaje;
                mensaje = (String) Conexion.entrada.readObject();
                dhcp=mensaje;

                Red aux = new Red(Adaptador, dirfis, dirip, descripcion, mask,
                    estado, dhcp);
                DispRed.add(aux);
                mensaje = (String) Conexion.entrada.readObject();
            }
        }

        for(int i=0;i<DispRed.size();i++){
            if(DispRed.get(i).Adaptador().equals("") && DispRed.get(i).DirFis().equals("") && DispRed.get(i).DirIP().equals("") && DispRed.get(i).Descripcion().equals("") && DispRed.get(i).Mascara().equals("") && DispRed.get(i).Estado().equals("") && DispRed.get(i).DHCP().equals(""))
                DispRed.remove(i);
        }
    }
}

```

```

// Establecemos adaptador
AdaptadorRed adaptador = new AdaptadorRed(this);
lstDispRed = (ListView) findViewById(R.id.ListaRed);
lstDispRed.setAdapter(adaptador);

} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ClassNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

}

class AdaptadorRed extends ArrayAdapter {

    Activity context;

    AdaptadorRed(Activity context) {
        super(context, R.layout.red, DispRed);
        this.context = context;
    }

    public View getView(int position, View convertView, ViewGroup
parent)
{
    View item = convertView;
    ViewHolder holder;

    if(item == null)
    {
        LayoutInflator inflater = context.getLayoutInflater();
        item = inflater.inflate(R.layout.red, null);

        holder = new ViewHolder();
        holder.Adaptador = (TextView)item.findViewById(R.id.
Adaptador);
        holder.DirFis = (TextView)item.findViewById(R.id.DirFisica);
        holder.DirIP = (TextView)item.findViewById(R.id.DirIP);
        holder.Descripcion = (TextView)item.findViewById(R.id.
Descripcion);
        holder.Mascara = (TextView)item.findViewById(R.id.Mascara);
        holder.Estado = (TextView)item.findViewById(R.id.Estado);
        holder.DHCP = (TextView)item.findViewById(R.id.DHCP);
        holder.TextoDHCP = (TextView)item.findViewById(R.id.
textView12);
        holder.ImagenRed= (ImageView)item.findViewById(R.id.
ImagenRed);
        LinearLayout.LayoutParams layoutParams = new LinearLayout.
LayoutParams(60, 60);
        holder.ImagenRed.setLayoutParams(layoutParams);
    }

    item.setTag(holder);
}
else
{
    holder = (ViewHolder)item.getTag();
}

holder.Adaptador.setText(DispRed.elementAt(position).Adaptador
());
holder.DirFis.setText(DispRed.elementAt(position).DirFis());
holder.DirIP.setText(DispRed.elementAt(position).DirIP());
holder.Descripcion.setText(DispRed.elementAt(position).
Descripcion());
holder.Mascara.setText(DispRed.elementAt(position).Mascara());
}

```

```

        if(holder.DirIP.getText().toString().trim().length()>0){
            holder.Estado.setText("Conectado");
        }else{
            holder.Estado.setText("Desconectado");
        }

        holder.DHCP.setText(DispRed.elementAt(position).DHCP());
        if(SesionLocal.ObtenerSistemaOperativo().equals("Linux")){
            holder.TextoDHCP.setVisibility(View.INVISIBLE);
        }

        if(holder.Adaptador.getText().toString().startsWith("Adaptador
            de Ethernet") || holder.Adaptador.getText().toString().
            startsWith("eth") || holder.Adaptador.getText().toString().
            startsWith("lo")){
            holder.ImagenRed.setBackgroundResource(R.drawable.
                iconoethernet);
        }else{
            holder.ImagenRed.setBackgroundResource(R.drawable.
                iconowireless);
        }
        return(item);
    }

    static class ViewHolder {
        public TextView Adaptador;
        public TextView DirFis;
        public TextView DirIP;
        public TextView Descripcion;
        public TextView Mascara;
        public TextView Estado;
        public TextView DHCP;
        public ImageView ImagenRed;
        public TextView TextoDHCP;
    }
}

```

---

InfSisOp.java

---

```

package garciaponce.miguel;

import java.io.IOException;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class InfSisOp extends Activity {
    private TextView PropiedadesSistemaOperativo;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.infsisop);
        Conexion.ACTIVIDAD=this;

        PropiedadesSistemaOperativo = (TextView) findViewById(R.id.
            PropiedadesSistemaOperativo);

        // Mandamos orden para obtener la informacion
        try {
            Conexion.salida.flush();
            Conexion.salida.writeObject("SistemaOperativo");
        }
    }
}

```

```

// Recibimos datos del sistema operativo
String mensaje = (String) Conexion.entrada.readObject();
while(!mensaje.equals("Fin_SistemaOperativo")){
    if(mensaje.startsWith("Path Librerías Java")){
        String datos [];
        datos = mensaje.split(";");
        PropiedadesSistemaOperativo.append(datos[0] + "\n");
        for(int i =1;i<datos.length;i++){
            if(datos[i].length()>3)
                PropiedadesSistemaOperativo.append("\t" + datos[i].
                    trim() + "\n");
        }
        PropiedadesSistemaOperativo.append("\n");
    } else{
        PropiedadesSistemaOperativo.append(mensaje + "\n\n");
    }
    mensaje = (String) Conexion.entrada.readObject();
}
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ClassNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
}
}

```

## Infsysmenu.java

```
package garciaponce.miguel;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class Infsysmenu extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.infsysmenu);
        Conexion.ACTIVIDAD=this;

        // Obtenemos componentes y pasamos al layout segun los botones pulsados.
        Button btnDiscos = (Button)findViewById(R.id.button1);
        btnDiscos.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                Intent intent = new Intent(Infsysmenu.this , InfDiscos.class);
                startActivity(intent);
            }
        });

        Button btnRed = (Button)findViewById(R.id.button2);
        btnRed.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                Intent intent = new Intent(Infsysmenu.this , InfRed.class);
                startActivity(intent);
            }
        });
    }
}
```

```
});  
  
Button btnSO = (Button) findViewById(R.id.Button01);  
btnSO.setOnClickListener(new OnClickListener() {  
    public void onClick(View v) {  
        Intent intent = new Intent(InfoSysMenu.this, InfoSysOp.  
            class);  
        startActivity(intent);  
    }  
});  
  
Button btnMem = (Button) findViewById(R.id.Button02);  
btnMem.setOnClickListener(new OnClickListener() {  
    public void onClick(View v) {  
        Intent intent = new Intent(InfoSysMenu.this, InfoMem.class  
            );  
        startActivity(intent);  
    }  
});  
}  
}
```

## ListaHosts.java

```
package garciaponce.miguel;

import java.util.Vector;

import android.app.Activity;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.util.Log;
import android.view.ContextMenu;
import android.view.ContextMenu.ContextMenuItemInfo;
import android.view.KeyEvent;
import android.view.LayoutInflater;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemLongClickListener;
import android.widget.ArrayAdapter;
import android.widget.ImageButton;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

public class ListaHosts extends Activity {

    private ListView ListaHosts;
    private Vector<Host> Hosts;
    private ImageButton BtnAregar;
    private int pos;
    private String NombreBD = "BDRemSys";
    private String NombreTabla = "Hosts";
    private UsuariosSQL usdbh;
    private SQLiteDatabase db;
    private Cursor c;
    private Toast toast;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }
}
```

```

setContentView(R.layout.listahosts);

// Creamos el Vector de Hosts y obtenemos los componentes del
// Layout
Hosts = new Vector<Host>();
ListaHosts = (ListView) findViewById(R.id.ListaHosts);
BtnAregar = (ImageButton) findViewById(R.id.BotonAregarHost);

// Acceder a los datos de la base de datos para almacenarlos en
// el vector Hosts
usdbh = new UsuariosSQL(ListaHosts.this, NombreBD, null, 1);
db = usdbh.getWritableDatabase();

// Consulta para saber los host que pertenecen al usuario ya
// logeado
// Creamos la table si no existe.
db.execSQL("CREATE TABLE IF NOT EXISTS Hosts (usuario TEXT,
    nombrehost TEXT, dirip TEXT, puerto TEXT, dirmac TEXT, PRIMARY
    KEY(usuario ,nombrehost ,dirip ,puerto))");
c = db.rawQuery("SELECT * FROM " + NombreTabla + " WHERE usuario
    =" + SesionLocal.NombreUsuario + "\'', null);

if (c.moveToFirst()) {
    // Recorremos el cursor hasta que no haya más registros
    do {
        Host host = new Host(c.getString(1), c.getString(2), c.
            getString(3), c.getString(4));
        Hosts.add(host);
        Log.i("ListaHosts", c.getString(0) + " " + c.getString(1) +
            " " + c.getString(2) + " " + c.getString(3) + " " + c.
            getString(4));
    } while (c.moveToNext());
}

// Creamos el Adaptador
AdaptadorHosts adaptador = new AdaptadorHosts(this);
ListaHosts.setAdapter(adaptador);
// Registramos el menu Contextual
registerForContextMenu(ListaHosts);

ListaHosts.setOnItemLongClickListener(new OnItemLongClickListener
() {
    public boolean onItemLongClick(AdapterView<?> parent, final
        View v, int position, long id) {
        // Cuando hacemos una pulsación larga, guardamos la
        // posición.
        pos=position;
        return false;
    }
});

// Si le damos al botón agregar, cambiamos de Layout para crear
// un nuevo host.
BtnAregar.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        Intent intent = new Intent(ListaHosts.this, CrearHost.
            class);
        startActivity(intent);
    }
});

@Override
public void onCreateContextMenu(ContextMenu menu, View v,
    ContextMenuItemInfo menuInfo)

```

```

{
    super.onCreateContextMenu(menu, v, menuInfo);
    MenuInflater inflater = getMenuInflater();
    if(v.getId() == R.id.ListaHosts)
    {
        inflater.inflate(R.menu.menuhosts, menu);
    }
}

@Override
public boolean onContextItemSelected(MenuItem item) {
    // Menú contextual, Conectar, Eliminar y Encender.
    switch (item.getItemId()) {
        case R.id.menuhost1:
            Conectar();
            return true;
        case R.id.menuhost2:
            // Eliminar Host de la lista de hosts de la Base de Datos y
            // volver a recargar el layout
            db.execSQL("DELETE FROM " + NombreTabla + " WHERE usuario='"
                + SesionLocal.NombreUsuario + "' AND nombrehost='"
                + Hosts.get(pos).getNom() + "'");
            db.close();
            // Recargamos el Layout
            Intent intent = new Intent(ListaHosts.this, ListaHosts.class);
            startActivity(intent);
            return true;
        case R.id.menuhost3:
            // Si encendemos el sistema, pasamos al siguiente intent los
            // datos necesarios con IP y MAC.
            Intent intent1 = new Intent(ListaHosts.this, WOL.class);
            intent1.putExtra("DirIP", Hosts.get(pos).getIP());
            intent1.putExtra("DirMac", Hosts.get(pos).getMac());
            startActivity(intent1);
            return true;
        default:
            return super.onContextItemSelected(item);
    }
}

class AdaptadorHosts extends ArrayAdapter {
    Activity context;

    AdaptadorHosts(Activity context) {
        super(context, R.layout.host, Hosts);
        this.context = context;
    }

    public View getView(int position, View convertView, ViewGroup
        parent)
    {
        View item = convertView;
        ViewHolder holder;

        if(item == null)
        {
            LayoutInflater inflater = context.getLayoutInflater();
            item = inflater.inflate(R.layout.host, null);

            holder = new ViewHolder();
            holder.DatoHost1 = (TextView)item.findViewById(R.id.NombreHost);
            holder.DatoHost2 = (TextView)item.findViewById(R.id.IPHost);
        }
        return item;
    }
}

class ViewHolder {
    TextView DatoHost1;
    TextView DatoHost2;
}

```

```

holder.DatoHost3 = (TextView) item.findViewById(R.id.
    PuertoHost);
holder.DatoHost4 = (TextView) item.findViewById(R.id.MacHost)
    ;
item.setTag(holder);
}
else
{
    holder = (ViewHolder)item.getTag();
}

holder.DatoHost1.setText(Hosts.elementAt(position).getNom());
holder.DatoHost2.setText(Hosts.elementAt(position).getIP());
holder.DatoHost3.setText(Hosts.elementAt(position).getPuerto())
    ;
holder.DatoHost4.setText(Hosts.elementAt(position).getMac());

return(item);
}

static class ViewHolder {
    public TextView DatoHost1;
    public TextView DatoHost2;
    public TextView DatoHost3;
    public TextView DatoHost4;
}

private void Conectar(){
    toast = Toast.makeText(getApplicationContext(),"Conectando al host
        " + Hosts.get(pos).getIP(), Toast.LENGTHLONG);
    toast.show();

    // Establecemos los parametros del Servicio Conexión
    Conexion.establecerIP(Hosts.get(pos).getIP());
    Conexion.establecerSocket(Integer.parseInt(Hosts.get(pos).
        getPuerto()));

    // Iniciamos el servicio Conexión
    Intent servicio = new Intent(ListaHosts.this, Conexion.class);
    if(startService(servicio)==null){
        Log.i("Conectar","No se ha podido iniciar el servicio Conexion");
    } else {
        Log.i("Conectar","Servicio Conexion iniciado correctamente");
    }

    // Cambiamos el Layaout al menú principal, una vez hemos iniciado
    // el servicio Conexion.
    Intent intent = new Intent(ListaHosts.this , Mainmenu.class);
    startActivity(intent);
}

@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    if ((keyCode == KeyEvent.KEYCODE_BACK)) {
        Log.i("ListaHosts","Vuelve a login");
        Intent intent = new Intent(ListaHosts.this , Login.class);
        startActivity(intent);
    }
    return super.onKeyDown(keyCode, event);
}
}

```

## Login.java

---

```

package garciaponce.miguel;

import android.app.Activity;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.util.Log;
import android.view.KeyEvent;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.Toast;

public class Login extends Activity {
    private EditText Usuario;
    private EditText Contrasena;
    private ImageButton AgregarUsuario;
    private Button Entrar;
    private String NombreBD = "BDRemSys";
    private String NombreTabla = "Usuarios";
    private UsuariosSQL usdbh;
    private SQLiteDatabase db;
    private Cursor c;
    private Toast toast;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.login);

        // Obtenemos los componentes
        AgregarUsuario = (ImageButton) findViewById(R.id.BotonAgregar);
        Entrar = (Button) findViewById(R.id.BotonEntrar);

        // Si pulsamos sobre el botón agregar
        AgregarUsuario.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                Intent intent = new Intent(Login.this, CrearUsuario.class);
                startActivity(intent);
            }
        });

        // Si pulsamos sobre el botón entrar
        Entrar.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                Usuario = (EditText) findViewById(R.id.editTextUsuario);
                Contrasena = (EditText) findViewById(R.id.editTextContrasena);

                // Comprobamos si el usuario y contraseña se encuentran
                // en la base de datos.
                if(ComprobarUsuario(Usuario.getText().toString().trim(),
                        Contrasena.getText().toString().trim())){
                    // Iniciamos el servicio para el control de la sesión
                    // (Nombreusuario)
                    Intent sesion = new Intent(Login.this, SesionLocal.class);
                    if(startService(sesion)==null){
                        Log.i("SesionLocal","No se ha podido iniciar el
                                servicio");
                    } else {

```

```

        Log.i("SesionLocal","Servicio iniciado
               correctamente");
    }
    SesionLocal.NombreUsuario=Usuario.getText().toString
    () . trim() ;
    // Una vez comprobada las credenciales, pasamos a la
    // ListaHosts
    Intent intent = new Intent(Login.this , ListaHosts.
        class);
    startActivity(intent);
}else{
    // Ha habido algun error en la comprobación de
    // usuarios
    toast = Toast.makeText(getApplicationContext(),"
        Credenciales incorrectas", Toast.LENGTHSHORT);
    toast.show();
}
}
}
}

private boolean ComprobarUsuario(String usu, String contra){
    String Ctr = null;
    // Accedemos a la BD
    usdbh = new UsuariosSQL(Login.this , NombreBD , null , 1);
    db = usdbh.getWritableDatabase();
    // Si no existe la creamos
    db.execSQL("CREATE TABLE IF NOT EXISTS Usuarios (nombre TEXT,
        contrasena TEXT , PRIMARY KEY(nombre))");
    // Ejecutamos la sentencia SQL
    c = db.rawQuery(" SELECT contrasena FROM " + NombreTabla + "
        WHERE nombre='\" + usu + "\'", null);

    //Nos aseguramos de que existe al menos un registro
    if(c.getCount()==0){
        return false;
    }else{
        if (c.moveToFirst()){
            //Recorremos el cursor hasta que no haya más registros
            do {
                Ctr = c.getString(0);
                Log.i("Comprobar Usuario ", "Ctr: " + Ctr + "contra: "
                    + contra);
            } while(c.moveToNext());
        }
        db.close();
        // Si hemos encontrado alguna coincidencia (existe usuario),
        // devolvemos true
        if(Ctr.equals(contra)){
            return true;
        }
    }
    return false;
}

@Override
public boolean onKeyDown(int keyCode , KeyEvent event) {
    if ((keyCode == KeyEvent.KEYCODEBACK)) {
        moveTaskToBack(true);
        return true;
    }
    return super.onKeyDown(keyCode , event);
}

```

```
    }
}
```

---

### MainMenu.java

---

```
package garciaponce.miguel;

import java.io.IOException;
import java.io.OptionalDataException;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.KeyEvent;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;

public class Mainmenu extends Activity {
    private Button btnInfSys;
    private Button btnProcesos;
    private Button btnNavegacion;
    private Button btnScripts;
    private Button btnOrdenLibre;
    private Button btnReinicio;
    private Button btnApagado;
    private Toast toast;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.mainmenu);
        Conexion.ACTIVIDAD=this;
        if(Conexion.conexion == null){
            toast = Toast.makeText(getApplicationContext(),"No se ha
                conectado" , Toast.LENGTHLONG);
            toast.show();
            Intent intent = new Intent(Mainmenu.this , Login.class);
            startActivity(intent);
        }else{
            if(Conexion.conexion.isConnected()){
                toast = Toast.makeText(getApplicationContext(),"Conectando
                    al host " + Conexion.ObtenerIP() , Toast.LENGTH_LONG);
                toast.show();
            }
        }
        // Mandamos mensaje para obtener el sistema operativo del
        // sistema remoto.
        try {
            Conexion.salida.flush();
            Conexion.salida.writeObject("MainMenu");
            String mensaje = (String) Conexion.entrada.readObject();
            SesionLocal.establecerSistemaOperativo(mensaje);
        } catch (OptionalDataException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        } catch (ClassNotFoundException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        } catch (IOException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
    }
}
```

```

// Obtenemos componentes gráficos y actuamos según botones.
final AlertDialog.Builder alert = new AlertDialog.Builder(this);
btnInfSys = (Button) findViewById(R.id.BotonInfSistema);
btnInfSys.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        Intent intent = new Intent(Mainmenu.this , Infsysmenu.class);
        startActivity(intent);
    }
});

btnProcesos = (Button) findViewById(R.id.BotonProcesos);
btnProcesos.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        Intent intent = new Intent(Mainmenu.this , InfProcesos.class);
        startActivity(intent);
    }
});

btnNavegacion = (Button) findViewById(R.id.BotonNavegacion);
btnNavegacion.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        Intent intent = new Intent(Mainmenu.this , Navegador.class);
        startActivity(intent);
    }
});

btnScripts = (Button) findViewById(R.id.BotonScripts);
btnScripts.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        Intent intent = new Intent(Mainmenu.this , Scripts.class);
        startActivity(intent);
    }
});

btnOrdenLibre = (Button) findViewById(R.id.BotonOrdenLibre);
btnOrdenLibre.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        Intent intent = new Intent(Mainmenu.this , OrdenLibre.class);
        startActivity(intent);
    }
});

btnReinicio = (Button) findViewById(R.id.BotonReiniciar);
btnReinicio.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        alert.setMessage("¿Seguro que desea reiniciar el sistema?");
        alert.setPositiveButton("Reiniciar", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog , int whichButton) {
                try {
                    Conexion.salida.flush();
                    Conexion.salida.writeObject("Reiniciar");
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }
        });
    }
});

```

```

        alert.setNegativeButton("Cancelar",
            new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog,
                    int whichButton) {
                    dialog.cancel();
                }
            });
        alert.show();
    });
}

btnApagado = (Button)findViewById(R.id.BotonApagar);
btnApagado.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        alert.setMessage("¿Seguro que desea apagar el sistema?");
        alert.setPositiveButton("Apagar", new DialogInterface.
            OnClickListener() {
                public void onClick(DialogInterface dialog, int
                    whichButton) {
                    try {
                        Conexion.salida.flush();
                        Conexion.salida.writeObject("Apagar");
                    } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }
                }
            });
        alert.setNegativeButton("Cancelar",
            new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog,
                    int whichButton) {
                    dialog.cancel();
                }
            });
        alert.show();
    }
});

@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    if ((keyCode == KeyEvent.KEYCODEBACK)) {
        // Si pulsamos el boton BACK, finalizamos el servicio de
        // conexion y volvemos al layout ListaHosts

        try {
            Conexion.salida.flush();
            Conexion.salida.writeObject("TerminarConexion");
        } catch (OptionalDataException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        } catch (IOException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }

        Intent servicio = new Intent(Mainmenu.this, Conexion.class);
        if(stopService(servicio))
        {
            Log.i("Conectar","Servicio finalizado correctamente");
        }
        else
        {
    }
}

```

```

        Log.i("Conectar","No se ha podido finalizar servicio
                Conexion");
    }

    Intent intent = new Intent(Mainmenu.this , ListaHosts.class);
    startActivity(intent);
}
return super.onKeyDown(keyCode, event);
}
}

```

## Navegador.java

---

```

package garciaponce.miguel;

import java.io.BufferedInputStream;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OptionalDataException;
import java.net.Socket;
import java.util.Vector;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.DialogInterface.OnCancelListener;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.Environment;
import android.util.Log;
import android.view.ContextMenu;
import android.view.ContextMenu.ContextMenuItemInfo;
import android.view.KeyEvent;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemLongClickListener;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.GridView;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;

public class Navegador extends Activity {
    private TextView Ruta;
    private String NuevaRuta;
    private GridView ListaFicheros;
    private Vector<Fichero> Ficheros;
    private Button btnCrearFichero;
    private Vector<String> RutaPrevias;
    private AdaptadorFicheros adaptador;
    private ImageButton ImagenFlechaIzq;
    private ImageButton ImagenFlechaDer;
    private String mensaje;
    private int pos;
}

```

```

private boolean IndicadorOpcionPegar;
private boolean IndicadorCopia;
private boolean IndicadorCorte;
private String FicheroOrigenPegado;
private String argumentos;
private boolean IndicadorCFichero=false;
private String RutaCrearFichero;
private Socket conexiontrans;
private AlertDialog alertDialog;
private ProgressDialog pDialog;
private MiTareaAsincrona tarea1;
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.navegador);

    // Inicialización
    IndicadorOpcionPegar=false;
    RutaPrevias=new Vector<String>();
    Ficheros = new Vector<Fichero>();

    Ruta = (TextView) findViewById(R.id.TextViewRuta);

    // Comprobamos si hemos vuelto de la actividad CrearFichero
    Bundle b = this.getIntent().getExtras();
    if(b != null){
        IndicadorCFichero = b.getBoolean("IndicadorCrearFichero");
        RutaCrearFichero = b.getString("Ruta");
    }

    // Si hemos venido de la actividad anterior, actualizamos ruta y
    // GridView
    if(IndicadorCFichero){
        Ruta.setText(RutaCrearFichero);
        ActualizarGridViewAnterior();
    }

    // Establecemos el adaptador.
    adaptador = new AdaptadorFicheros(this);
    ListaFicheros = (GridView)findViewById(R.id.GridViewFicheros)
        ;
    ListaFicheros.setAdapter(adaptador);

} else{ // Sino, comenzamos una nueva navegación, estableciendo
    // la ruta y obteniendo MiPc

    Ruta.setText("");
    try {
        // Enviamos el mensaje Navegacion para obtener las
        // unidades (Mi PC) y empezar a visualizarlas.
        Conexion.salida.flush();
        Conexion.salida.writeObject("Navegacion");
        MostrarMiPc();

    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

// Registramos menu contextual para el GridView
registerForContextMenu(ListaFicheros);

// Cuando pulsemos sobre algun fichero, actualizamos ruta y
// GridView

```

```

ListaFicheros.setOnItemClickListener(new android.widget.
    AdapterView.OnItemClickListener(){
    public void onItemClick(AdapterView<?> parent, final View v,
        int position, long id) {
        // record position/id/whatever here
        ActualizarRuta(parent, v, position, id);
        ActualizarGridView(parent, v, position, id);
    });
    // Al pulsar largo sobre cualquier elemento, guardamos su
    posicion
    ListaFicheros.setOnItemLongClickListener(new
        OnItemLongClickListener() {
        public boolean onItemLongClick(AdapterView<?> parent, final
            View v, int position, long id) {
            // record position/id/whatever here
            pos=position;
            return false;
        });
    // Pulsación del ImageButton FlechaIzquierda
    ImagenFlechaIzq = (ImageButton)findViewById(R.id.ImageFlechaIzq);
    ImagenFlechaIzq.setOnClickListener(new Button.OnClickListener() {
        public void onClick(View v) {
            // Si no hay ruta especificada (principio), no hacemos
            nada
            if(SesionLocal.ObtenerSistemaOperativo().equals("Windows"))
            {
                //Windows
                if(Ruta.getText().toString().length()!=0){
                    //Guardamos la RutaPrevia para el
                    botónFlechaDerecha
                    NuevaRuta="";
                    int cont=RutaAnteriorGuardado();
                    // Si estamos listando una unidad, para volver a MiPc
                    // recargamos el layout
                    if(cont==0){
                        MostrarMiPc();
                        Ruta.setText("");
                    }else{
                        Ruta.setText(NuevaRuta);
                        // Mandamos el directorio a listar.
                        ActualizarGridViewAnterior();
                    }
                }
            }else{
                //Linux
                if(Ruta.getText().toString().length()>1){
                    String NuevaRuta="";
                    int cont=RutaAnteriorGuardado();
                    // Separamos la ruta por el carácter "/"
                    String datos[] = Ruta.getText().toString().split("/");
                    for(int i=1;i<(datos.length-1);i++)
                        NuevaRuta=NuevaRuta + "/" + datos[i];
                    NuevaRuta = NuevaRuta.trim();
                    if(NuevaRuta.equals(""))
                        NuevaRuta="/";
                    Ruta.setText(NuevaRuta);
                    // Mandamos el directorio a listar.
                    ActualizarGridViewAnterior();
                }
            }
        }
    });
}

```

```

        }

    });

// Flecha Derecha
ImagenFlechaDer = (ImageButton) findViewById(R.id.ImageFlechaDer);
ImagenFlechaDer.setOnClickListener(new Button.OnClickListener() {
    public void onClick(View v) {
        if(RutaPrevias.size()!=0){
            if(RutaPrevias.lastElement().length()>Ruta.getText()
               ().length()){
                Ruta.setText(RutaPrevias.lastElement());
            }
            // Eliminamos el ultimo elemento de las rutas
               previas
RutaPrevias.removeElementAt(RutaPrevias.size()-1);
// Obtenemos los resultados y almacenamos en un
               vector Ficheros para visualizarlos en el
               GridView
ActualizarGridViewAnterior();
    }
}

// Boton Crear Fichero
btnCrearFichero = (Button) findViewById(R.id.BotonCrearFichero);
btnCrearFichero.setOnClickListener(new Button.OnClickListener() {
    public void onClick(View v) {
        // Nos cercioramos de que sea una ruta válida (no se
           puede crear en una ruta vacía)
if((Ruta.getText().length()>=3 && SesionLocal.
   ObtenerSistemaOperativo().equals("Windows")) || (
   Ruta.getText().length()>=1 && SesionLocal.
   ObtenerSistemaOperativo().equals("Linux"))){
            Intent intent = new Intent(Navegador.this,
               CrearFichero.class);
            intent.putExtra("Ruta", Ruta.getText().toString());
            startActivity(intent);
        }
    }
});

private int RutaAnteriorGuardado(){
    int cont = 0;

    if(SesionLocal.ObtenerSistemaOperativo().equals("Windows")){
        // Añadimos la ruta actual a las previas
RutaPrevias.add(Ruta.getText().toString());

        // Establecemos NuevaRuta a la RutaAnterior
String ruta[];
ruta=Ruta.getText().toString().split("\\\\\\");
cont=0;
for(int i=0;i<(ruta.length-1);i++){
    if(NuevaRuta.length()==0){
        NuevaRuta=ruta[i] + "\\";
    }else{
        NuevaRuta=NuevaRuta+ruta[i]+ "\\";
    }
    cont++;
}
} else{
    //Linux
RutaPrevias.add(Ruta.getText().toString());
}
}

```

```

    }
    for(int i=0;i<RutaPrevias.size();i++){
        Log.i("RutaPRevias", RutaPrevias.get(i));
    }
    return cont;
}

// Procedimiento para tratar con directorios . y ..
private void ActualizarRutaAnterior(){
    if(SesionLocal.ObtenerSistemaOperativo().equals("Windows")){
        String ruta[];
        String NuevaRuta="";
        ruta=Ruta.getText().toString().split("\\\\\\");
        for(int i=0;i<(ruta.length-1);i++){
            if(NuevaRuta.length()==0){
                NuevaRuta=ruta[i] + "\\";
            }else{
                NuevaRuta=NuevaRuta+ruta[i]+ "\\";
            }
        }
        Ruta.setText(NuevaRuta);
    }else{
        //Linux
    }
}

private void ActualizarGridViewAnterior(){
    try {
        Conexion.salida.flush();
        Conexion.salida.writeObject("ActualizarGridView");
        Conexion.salida.flush();
        Conexion.salida.writeObject(Ruta.getText());

        // Obtenemos los resultados y almacenamos en un vector
        // Ficheros para visualizarlos en el GridView
        mensaje = (String) Conexion.entrada.readObject();
        Ficheros.clear();
        int cont=0;
        String[] datos_recibidos;
        while(!mensaje.equals("fin_listado_dir")){
            if(SesionLocal.ObtenerSistemaOperativo().equals("Windows")){
                if(cont > 2){
                    String NombreFichero="";
                    datos_recibidos = mensaje.split(" ");
                    for(int i=1;i<datos_recibidos.length;i++){
                        NombreFichero = NombreFichero+" "+datos_recibidos[i];
                    }
                    Fichero aux = new Fichero(datos_recibidos[0].toString()
                        ().trim(),NombreFichero.trim());
                    Ficheros.add(aux);
                }
            }else{
                String NombreFichero="";
                datos_recibidos = mensaje.split(" ");
                for(int i=1;i<datos_recibidos.length;i++){
                    NombreFichero = NombreFichero+" "+datos_recibidos[i];
                }
                Fichero aux = new Fichero(datos_recibidos[0].toString()
                    .trim(),NombreFichero.trim());
                Ficheros.add(aux);
            }
            cont++;
            mensaje = (String) Conexion.entrada.readObject();
        }
    }
}

```

```

    // Establecemos el adaptador con el nuevo Vector<Fichero>
    obtenido.
    ListaFicheros = (GridView) findViewById(R.id.GridViewFicheros);
    ListaFicheros.setAdapter(adaptador);
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ClassNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

// Funcion encargada de la actualización de la Ruta.
private void ActualizarRuta(AdapterView<?> parent, final View v, int
position, long id){
    // Actualizamos la ruta
    if(SesionLocal.ObtenerSistemaOperativo().equals("Windows")){
        if(Ruta.getText().length() == 0){
            Ruta.setText(Ficheros.get(position).getNombre());
        } else{
            // Si el fichero que pulsamos es .. , volvemos a la ruta
            // anterior
            if(Ficheros.get(position).getNombre().equals("..")){
                ActualizarRutaAnterior();
            } else{
                // Si el fichero el cual pulsamos es un punto, no hacemos
                // nada en el cambio de ruta
                if(Ficheros.get(position).getNombre().equals(".")){
                    // Si el fichero el cual pulsamos es .. , volvemos a la ruta
                    // anterior
                    if(Ruta.getText().length() == 0){
                        Ruta.setText(Ficheros.get(position).getNombre());
                    } else{
                        Ruta.setText(Ruta.getText() + Ficheros.get(position).
getNombre() + "\\" );
                    }
                }
            }
        }
    } else{
        //Linux
        if(Ruta.getText().toString().length() > 1){
            Ruta.setText(Ruta.getText().toString() + "/" + Ficheros.get(
                position).getNombre());
        } else{
            Ruta.setText(Ruta.getText().toString() + Ficheros.get(
                position).getNombre());
        }
    }
}

private void ActualizarGridView(AdapterView<?> parent, final View v,
int position, long id){
    // Mandamos el directorio a listar.
    try {
        Conexion.salida.flush();
        Conexion.salida.writeObject("ActualizarGridView");
        Conexion.salida.flush();
        Conexion.salida.writeObject(Ruta.getText());
        // Obtenemos los resultados y almacenamos en un vector
        // Ficheros para visualizarlos en el GridView
        mensaje = (String) Conexion.entrada.readObject();
        Ficheros.clear();
        int cont=0;
        String[] datos_recibidos;
        while(!mensaje.equals("fin_listado_dir")){

```

```

        if(SesionLocal.ObtenerSistemaOperativo().equals("Windows")
        )) {
            if(cont > 2 ){
                String NombreFichero="";
                datos_recibidos = mensaje.split(" ");
                for(int i=1;i<datos_recibidos.length;i++){
                    NombreFichero = NombreFichero+" "+
                        datos_recibidos[i];
                }
                Fichero_aux = new Fichero(datos_recibidos[0].
                    toString().trim(),NombreFichero.trim());
                Ficheros.add(aux);
            }
        } else{
            String NombreFichero="";
            datos_recibidos = mensaje.split(" ");
            for(int i=1;i<datos_recibidos.length;i++){
                NombreFichero = NombreFichero+" "+
                    datos_recibidos[i];
            }
            Fichero_aux = new Fichero(datos_recibidos[0].
                toString().trim(),NombreFichero.trim());
            Ficheros.add(aux);
        }
        cont++;
        mensaje = (String) Conexion.entrada.readObject();
    }

    ListaFicheros = (GridView)findViewById(R.id.GridViewFicheros
    );
    ListaFicheros.setAdapter(adaptador);
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ClassNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
}

private void MostrarMiPc(){
try {
    // Si es Linux
    if(SesionLocal.ObtenerSistemaOperativo().equals("Linux"))
        Ruta.setText("/");
    Conexion.salida.flush();
    Conexion.salida.writeObject("MostrarMiPc");
    Ficheros.clear();

    // Obtenemos los datos del servidor.
    mensaje = (String) Conexion.entrada.readObject();
    String [] datos;
    while(!mensaje.equals("Fin_MostrarMiPc")){
        datos = mensaje.split(" ");
        Fichero_aux = new Fichero(datos[1].toString().trim(),datos
            [0].toString().trim());
        Ficheros.add(aux);
        Log.i("Ficheros",aux.getIcono()+" "+aux.getNombre());
        mensaje = (String) Conexion.entrada.readObject();
    }
    // Establecemos el adaptador
    adaptador = new AdaptadorFicheros(this);
    ListaFicheros = (GridView)findViewById(R.id.GridViewFicheros)
    ;
    ListaFicheros.setAdapter(adaptador);
} catch (IOException e) {
}
}

```

```

        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (ClassNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

// Adaptador del GridView
class AdaptadorFicheros extends ArrayAdapter {
    Activity context;

    AdaptadorFicheros(Activity context) {
        super(context, R.layout.fichero, Ficheros);
        this.context = context;
    }

    public View getView(int position, View convertView, ViewGroup
        parent)
    {
        View item = convertView;
        ViewHolder holder;

        if(item == null)
        {
            LayoutInflator inflater = context.getLayoutInflater();
            item = inflater.inflate(R.layout.fichero, null);

            holder = new ViewHolder();
            holder.icono = (ImageView)item.findViewById(R.id.
                ImagenFichero);
            holder.nombre = (TextView)item.findViewById(R.id.
                TextoFichero);
            // Cambiamos el tamaño de la imagen del fichero.
            LinearLayout.LayoutParams layoutParams = new LinearLayout.
                LayoutParams(40, 40);
            holder.icono.setLayoutParams(layoutParams);
            holder.nombre.setTextSize(11);

            item.setTag(holder);
        }
        else
        {
            holder = (ViewHolder)item.getTag();
        }

        holder.nombre.setText(Ficheros.elementAt(position).getNombre());
        ;
        // Procesamiento de iconos
        if(Ficheros.elementAt(position).getIcono().equals("<DIR>"))
        ){
            holder.icono.setBackgroundResource(R.drawable.discoduro);
        }else{
            if(Ficheros.elementAt(position).getIcono().equals("CD-ROM"))
            {
                holder.icono.setBackgroundResource(R.drawable.cdrom);
            }else{
                if(Ficheros.elementAt(position).getIcono().equals("<DIR>")
                    ) || Ficheros.elementAt(position).getIcono().
                    startsWith("d")){
                    holder.icono.setBackgroundResource(R.drawable.carpeta);
                };
            }else{
                if(Ficheros.elementAt(position).getNombre().endsWith(
                    ".exe")){

```





```

{
    // Ficheros exe, bat, exe, java , bin, sh , run
    if(Ficheros.get(pos).getNombre().endsWith(".exe") || Ficheros.
        get(pos).getNombre().endsWith(".bat") || Ficheros.get(pos)
        .getNombre().endsWith(".java") || Ficheros.get(pos).
        getNombre().endsWith(".sh") || Ficheros.get(pos).getNombre()
        .endsWith(".run") || Ficheros.get(pos).getNombre().
        endsWith(".py") || Ficheros.get(pos).getNombre().endsWith(
        ".bin")){
        inflater.inflate(R.menu.menunavegadorejecutar, menu);
    }else{
        // Ficheros txt , jpg , png , pdf
        if(Ficheros.get(pos).getNombre().endsWith(".txt") ||
            Ficheros.get(pos).getNombre().endsWith(".jpg") ||
            Ficheros.get(pos).getNombre().endsWith(".mp3") ||
            Ficheros.get(pos).getNombre().endsWith(".png") ||
            Ficheros.get(pos).getNombre().endsWith(".pdf"))
            inflater.inflate(R.menu.menunavegadortransferir,
                menu);
        else{
            // Demás ficheros (Cuya longitud de ruta sea mayor
            // de 0)
            if(Ruta.length()>0)
                inflater.inflate(R.menu.menunavegador, menu);
        }
    }
}

// Tratamiento del menu contextual
@Override
public boolean onContextItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        // Si pulsamos sobre ejecutar.
        case R.id.menunavegadorejecutar:
            Ejecutar();
            return true;

        // Si pulsamos sobre Tranferir a SD-Card
        case R.id.menunavegadortransferir:
            Dialog = new ProgressDialog(Navegador.this);
            pDialog.setProgressStyle(ProgressDialog.STYLE_SPINNER);
            pDialog.setMessage("Transfiriendo fichero " + Ficheros.get(
                pos).getNombre().toString());
            pDialog.show();
            pDialog.setCancelable(false);
            tarea1 = new MiTareaAsincrona();
            tarea1.execute();

            return true;

        // Si pulsamos sobre copiar
        case R.id.menunavegadocopiar:
            IndicadorOpcionPegar=true;
            IndicadorCopia=true;
            IndicadorCorte=false;
            if(SesionLocal.ObtenerSistemaOperativo().equals("Windows"))
                FicheroOrigenPegado = Ruta.getText().toString()+Ficheros.get(
                    pos).getNombre().toString();
            else
                FicheroOrigenPegado=Ruta.getText().toString()+"/"+Ficheros.
                    get(pos).getNombre().toString();

            return true;
        // Si pulsamos sobre Cortar
        case R.id.menunavegadocortar:
            IndicadorOpcionPegar=true;
            IndicadorCorte=true;
    }
}

```

```

IndicadorCopia=false;
if(SesionLocal.ObtenerSistemaOperativo().equals("Windows"))
    FicheroOrigenPegado=Ruta.getText().toString() + Ficheros.get(
        pos).getNombre().toString();
else
    FicheroOrigenPegado=Ruta.getText().toString() + "/" + Ficheros.
        get(pos).getNombre().toString();

return true;
// Si pulsamos sobre eliminar
case R.id.menuNavegadorEliminar:
    EliminarFichero();
    return true;
// Si pulsamos sobre renombrar
case R.id.menuNavegadorRenombrar:
    RenombrarFichero();
    return true;
default:
    return super.onContextItemSelected(item);
}

// Renombrar Fichero
private void RenombrarFichero(){
    final AlertDialog.Builder alert = new AlertDialog.Builder(this);
    alert.setMessage("Nuevo nombre para el fichero " + Ficheros.get(
        pos).getNombre());
    final EditText input = new EditText(this);
    alert.setView(input);
    alert.setPositiveButton("Renombrar", new DialogInterface.
        OnClickListener() {
            public void onClick(DialogInterface dialog, int whichButton)
            {
                try {
                    Enviar("RenombrarFichero");
                    Enviar(Ruta.getText().toString());
                    Enviar(Ficheros.get(pos).getNombre().toString());
                    Enviar(input.getText().toString());
                    mensaje = (String) Conexion.entrada.readObject();
                    Toast toast1 = Toast.makeText(getApplicationContext(),
                        mensaje, Toast.LENGTHLONG);
                    toast1.show();
                } catch (OptionalDataException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                } catch (ClassNotFoundException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                } finally{
                    ActualizarGridViewAnterior();
                }
            }
        });
    alert.setNegativeButton("Cancelar",
        new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int
                whichButton) {
                dialog.cancel();
            }
        });
    alert.show();
}

```

```

// EliminarFichero
private void EliminarFichero(){
    // Confirmacion
    final AlertDialog.Builder alert = new AlertDialog.Builder(this);
    alert.setMessage("Seguro que quieres eliminar el fichero " +
        Ficheros.get(pos).getNombre());
    alert.setPositiveButton("Eliminar", new DialogInterface.
        OnClickListener() {
            public void onClick(DialogInterface dialog, int whichButton)
            {
                try {
                    // Enviar datos al Servidor.
                    Enviar("EliminarFichero");
                    if(SesionLocal.ObtenerSistemaOperativo().equals(
                        "Windows"))
                        Enviar(Ruta.getText().toString()+Ficheros.get(pos).
                            getNombre().toString());
                    else
                        Enviar(Ruta.getText().toString() + "/" + Ficheros.get(
                            pos).getNombre().toString());

                    mensaje = (String) Conexion.entrada.readObject();
                    Toast toast1 = Toast.makeText(getApplicationContext(),
                        mensaje, Toast.LENGTHLONG);
                    toast1.show();
                } catch (OptionalDataException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                } catch (ClassNotFoundException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                } finally{
                    // Actualizamos GridView
                    ActualizarGridViewAnterior();
                }
            }
        });
    alert.setNegativeButton("Cancelar",
        new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int
                whichButton) {
                dialog.cancel();
            }
        });
    alert.show();
}

// TransferirFichero => hilo que intenta conectar a un nuevo puerto
// abierto (11111)
private class Transferirfichero implements Runnable{
    private String ip;
    private String nombre;
    private int Puerto;

    Transferirfichero(String i, String n,int puerto){
        ip=i;
        nombre=n;
        Puerto=puerto;
    }

    public void run() {

```

```

// TODO Auto-generated method stub

try {
    conexiontrans = new Socket(ip, Puerto);
    File folder = new File(Environment.getExternalStorageDirectory() + "/RemSys/");
    // Si la carpeta no existe, se crea
    if (!folder.exists()) {
        folder.mkdir();
    }

    // Abrimos fichero para escritura y mediante los flujos
    // adecuados escribimos lo que nos venga de el
    File f = new File(folder.getAbsolutePath(), nombre);
    byte[] b = new byte[1024];
    int len = 0;
    int bytcount = 1024;
    FileOutputStream inFile = new FileOutputStream(f);
    InputStream is = conexiontrans.getInputStream();
    BufferedInputStream in2 = new BufferedInputStream(is, 1024);
    while ((len = in2.read(b, 0, 1024)) != -1) {
        bytcount = bytcount + 1024;
        inFile.write(b, 0, len);
    }
    // Cerramos flujos y socket
    in2.close();
    inFile.close();
    conexiontrans.close();
} catch (FileNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (OptionalDataException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
}

private class MiTareaAsincrona extends AsyncTask<Void, Integer, Boolean> {

    @Override
    protected void onProgressUpdate(Integer... values) {
    }

    @Override
    protected void onPreExecute() {
        Enviar("TransferirFichero");
        Enviar(Ruta.getText().toString());
        Enviar(Ficheros.get(pos).getNombre().toString());
    }

    @Override
    protected void onCancelled() {
    }

    @Override
    protected void onPostExecute(Boolean result) {
        pDialog.dismiss();
        Toast.makeText(Navegador.this, "Fichero " + Ficheros.get(pos).getNombre().toString() + " transferido", Toast.LENGTH_LONG

```

```

) . show ( ) ;
}

@Override
protected Boolean doInBackground(Void... params) {
    // TODO Auto-generated method stub
    try {
        // Obtenemos mensaje de sincronización , en este momento el
        // servidor espera una conexión por el puerto indicado
        mensaje = (String) Conexion.entrada.readObject();
        String [] datos = mensaje.split(":");
        // Creamos el hilo que va a servir para la transferencia.
        if(datos[0].equals("ListoTransferencia")){
            Log.i("TransferirFichero", "new Transferirfichero(" +
                Conexion.ObtenerIP() + "," + Ficheros.get(pos).
                getNombre().toString() + "," + Integer.parseInt(datos
                [1]) + ")");
            Thread t = new Thread(new Transferirfichero(Conexion.
                ObtenerIP(), Ficheros.get(pos).getNombre().toString()
                , Integer.parseInt(datos[1])));
            t.start();
            try {
                t.join();
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
        catch (OptionalDataException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        catch (ClassNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
    return null;
}

// Ejecutar un fichero con argumentos.
private void Ejecutar(){
    final AlertDialog.Builder alert = new AlertDialog.Builder(this);
    alert.setMessage("Introduce los parámetros para la ejecución del
        fichero " + Ficheros.get(pos).getNombre());
    final EditText input = new EditText(this);
    alert.setView(input);
    alert.setPositiveButton("Ejecutar", new DialogInterface.
        OnClickListener() {
            public void onClick(DialogInterface dialog, int whichButton)
            {
                argumentos = input.getText().toString().trim();
                EjecutarFichero();
            }
        });
    alert.setNegativeButton("Cancelar",
        new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int
                whichButton) {

```

```

        dialog.cancel();
    }
}
alert.show();
}

private void EjecutarFichero(){
    // Establece comunicacion y envia datos para la ejecucion del
    // fichero.
    Enviar("EjecutarFichero");
    Enviar(Ruta.getText().toString());
    Enviar(Ficheros.get(pos).getNombre().toString());
    Enviar(argumentos);
}

private void Enviar(String n){
try{
    Conexion.salida.flush();
    Conexion.salida.writeObject(n);

} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
}

// Boton menu del telefono para la opcion de pegado de ficheros.
@Override
public boolean onPrepareOptionsMenu(Menu menu) {
    menu.clear();

    if(IndicadorOpcionPegar && Ruta.getText().toString().length()>0) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.menupegar, menu);
    }
    return super.onPrepareOptionsMenu(menu);
}

// Procesamiento de pegado de fichero, distinguiendo si se ha
// copiado o cortado.
private void PegarFichero(){
    if(IndicadorCopia){
        Enviar("Copia");
    }else{
        Enviar("Corte");
    }
    Enviar(FicheroOrigenPegado);
    Enviar(Ruta.getText().toString());
    try {
        String mensaje = (String) Conexion.entrada.readObject();
        Toast toast1 = Toast.makeText(getApplicationContext(),mensaje,
            Toast.LENGTHLONG);
        toast1.show();
    } catch (OptionalDataException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (ClassNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } finally{
        // Actualizamos Gridview y indicadores.
        ActualizarGridViewAnterior();
    }
}

```

```

        if(IndicadorCorte)
            IndicadorOpcionPegar=IndicadorCorte=false;
    }
}

// Seleccion de la opcion de pegado.
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.menuPegar:
            PegarFichero();
    }
    return true;
}
}

```

## OrdenLibre.java

---

```

package garciaponce.miguel;

import java.io.IOException;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class OrdenLibre extends Activity {
    private TextView SalidaOrden;
    private EditText Orden;
    private Button BtnEjecutar;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.ordenlibre);
        Conexion.ACTIVIDAD=this;

        Orden = (EditText) findViewById(R.id.Oorden);
        BtnEjecutar = (Button) findViewById(R.id.BotonEjecutar);

        BtnEjecutar.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                ObtenerSalida();
            }
        });
    }

    private void ObtenerSalida(){
        SalidaOrden = (TextView) findViewById(R.id.SalidaOrden);

        if(SalidaOrden.getText().length()>0)
            SalidaOrden.setText("");

        try{
            Conexion.salida.flush();
            Conexion.salida.writeObject("OrdenLibre");
            Conexion.salida.flush();
            Conexion.salida.writeObject(Orden.getText().toString().trim());

            String mensaje = (String) Conexion.entrada.readObject();
            while(!mensaje.equals("Fin_OordenLibre")){
                SalidaOrden.append(mensaje + "\n");
            }
        }
    }
}

```

```

        mensaje = (String) Conexion.entrada.readObject();
    }

} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ClassNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

}
}

```

## Scripts.java

---

```

package garciaponce.miguel;

import java.io.IOException;
import java.util.Vector;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.ContextMenu;
import android.view.ContextMenu.ContextMenuItemInfo;
import android.view.KeyEvent;
import android.view.LayoutInflater;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemLongClickListener;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.widget.TextView;

public class Scripts extends Activity {
    private Vector<String> Scripts;
    private ListView lstScripts;
    private int pos;
    private Button CrearScript;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.scripts);
        Conexion.ACTIVIDAD=this;

        Scripts = new Vector<String>();

        try {
            // Enviamos el mensaje Navegacion para obtener los Scripts del
            // directorio
            Conexion.salida.flush();
            Conexion.salida.writeObject("Scripts");

            String mensaje = (String) Conexion.entrada.readObject();
            while(!mensaje.equals("Fin_Scripts")){
                Scripts.add(mensaje);
                mensaje = (String) Conexion.entrada.readObject();
            }
        }
    }
}

```

```

// Establecemos el adaptador y registramos menu contextual
AdaptadorScripts adaptador = new AdaptadorScripts(this);
lstScripts = (ListView) findViewById(R.id.ListaScripts);
lstScripts.setAdapter(adaptador);
registerForContextMenu(lstScripts);

// Registraremos posicion del elemento seleccionado
lstScripts.setOnItemLongClickListener(new
    OnItemLongClickListener() {
        public boolean onItemLongClick(AdapterView<?> parent,
            final View v, int position, long id) {
            // record position/id/whatever here
            pos=position;
            return false;
        }
    });

// Boton CrearScript => nuevo layout
CrearScript = (Button) findViewById(R.id.CrearScript);
CrearScript.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        Intent intent = new Intent(Scripts.this, CrearScript.
            class);
        startActivity(intent);
    }
});

} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ClassNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (NullPointerException e){
    // TODO Auto-generated catch block
    e.printStackTrace();
}

}

@Override
public void onCreateContextMenu(ContextMenu menu, View v,
    ContextMenuInfo menuInfo)
{
    super.onCreateContextMenu(menu, v, menuInfo);

    MenuInflater inflater = getMenuInflater();

    if(v.getId() == R.id.ListaScripts){
        // Segun sea el sistema, obtenemos los menus contextuales de
        // los scripts a ejecutar
        if(SesionLocal.ObtenerSistemaOperativo().equals("Windows")){
            if(Scripts.get(pos).toString().endsWith(".exe") || Scripts.
                get(pos).toString().endsWith(".bat") || Scripts.get(pos).
                toString().endsWith(".cmd") || Scripts.get(pos).
                toString().endsWith(".py"))
                inflater.inflate(R.menu.menuscripts, menu);
        }else{
            if(Scripts.get(pos).toString().endsWith(".sh") || Scripts.
                get(pos).toString().endsWith(".py") || Scripts.get(pos).
                toString().endsWith(".run") || Scripts.get(pos).
                toString().endsWith(".bin"))
                inflater.inflate(R.menu.menuscripts, menu);
        }
    }
}

```

```

// Menu contextual (Ejecutar, eliminar y ver script)
public boolean onContextItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        // Si pulsamos sobre ejecutar.
        case R.id.menuscript1:
            EjecutarScript();
            return true;

        // Si pulsamos sobre Transferir a SD-Card
        case R.id.menuscript2:
            EliminarScript();
            return true;

        case R.id.menuscript3:
            VerScript();
            return true;

        default:
            return super.onContextItemSelected(item);
    }
}

private void VerScript(){
    Intent intent = new Intent(Scripts.this, VerScript.class);
    intent.putExtra("Fichero", Scripts.get(pos).toString());
    startActivity(intent);
}

private void EjecutarScript(){
    try {
        Conexion.salida.flush();
        Conexion.salida.writeObject("EjecutarScript " + Scripts.get(pos).toString());
        Intent intent = new Intent(Scripts.this, Scripts.class);
        startActivity(intent);
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

// Botón BACK del teléfono
@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    if ((keyCode == KeyEvent.KEYCODE_BACK)) {
        try {
            Conexion.salida.flush();
            Conexion.salida.writeObject("SalirScripts");
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
    Intent intent = new Intent(Scripts.this, Mainmenu.class);
    startActivity(intent);
    return super.onKeyDown(keyCode, event);
}

```

```

private void EliminarScript (){
    try {
        Conexion.salida.flush ();
        Conexion.salida.writeObject ("EliminarScript " + Scripts.get(pos)
            .toString ());
    }

    String mensaje = (String) Conexion.entrada.readObject ();
    if(mensaje.equals("FinEliminarScript")){
        Intent intent = new Intent(Scripts.this , Scripts.class);
        startActivity(intent);
    }
}

catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
catch (ClassNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
}

class AdaptadorScripts extends ArrayAdapter {
    Activity context;

    AdaptadorScripts(Activity context) {
        super(context , R.layout.script , Scripts);
        this.context = context;
    }

    public View getView(int position , View convertView , ViewGroup
        parent)
    {
        View item = convertView;
        ViewHolder holder;

        if(item == null)
        {
            LayoutInflater inflater = context.getLayoutInflater();
            item = inflater.inflate(R.layout.script , null);

            holder = new ViewHolder();
            holder.Nombre = (TextView)item.findViewById(R.id.
                NombreScript);

            item.setTag(holder);
        }
        else
        {
            holder = (ViewHolder)item.getTag();
        }

        holder.Nombre.setText(Scripts.get(position).toString());
        return(item);
    }

    static class ViewHolder {
        public TextView Nombre;
    }
}

```

```
package garciaponce.miguel;

import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;

import android.app.Activity;
import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.util.Log;

public class SesionLocal extends Service
{
    public static Activity ACTIVIDAD;
    public static String NombreUsuario;
    public static String SistemaOperativo;

    public static void establecerActividadPrincipal(Activity actividad)
    {
        SesionLocal.ACTIVIDAD=actividad;
    }

    public static void establecerSistemaOperativo(String Sis)
    {
        SesionLocal.SistemaOperativo=Sis;
    }

    public static String ObtenerSistemaOperativo()
    {
        return SistemaOperativo;
    }

    public void onCreate()
    {
        super.onCreate();

        // Iniciamos el servicio
        this.iniciarServicio();

        Log.i(getClass().getSimpleName(), "Servicio iniciado");
    }

    public void onDestroy()
    {
        super.onDestroy();

        // Detenemos el servicio
        this.finalizarServicio();

        Log.i(getClass().getSimpleName(), "Servicio detenido");
    }

    public IBinder onBind(Intent intent)
    {
        // No usado de momento, sólo se usa si se va a utilizar IPC
        // (Inter-Process Communication) para comunicarse entre procesos
        return null;
    }

    public void iniciarServicio()
    {

    }

    public void finalizarServicio()
```

```
{
}
```

## VerScript.java

---

```
package garciaponce.miguel;

import java.io.IOException;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.KeyEvent;
import android.widget.EditText;
import android.widget.TextView;

public class VerScript extends Activity {
    private EditText TextoVerScript;
    private TextView Titulo;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.verscript);
        Conexion.ACTIVIDAD=this;

        Intent intent = getIntent();
        String fichero = intent.getExtras().getString("Fichero");

        TextoVerScript = (EditText) findViewById(R.id.TextoVerScript);
        Titulo = (TextView) findViewById(R.id.TituloVerScript);
        Titulo.append(": " + fichero);
        try {
            Conexion.salida.flush();
            Conexion.salida.writeObject("VerScript");
            Conexion.salida.flush();
            Conexion.salida.writeObject(fichero);

            String mensaje = (String) Conexion.entrada.readObject();

            while (!mensaje.equals("FinLecturaScript")){
                TextoVerScript.append(mensaje + "\n");
                mensaje = (String) Conexion.entrada.readObject();
            }
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        TextoVerScript.setKeyListener(null);
    }

    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event) {
        if ((keyCode == KeyEvent.KEYCODE_BACK)) {
            Intent intent = new Intent(VerScript.this, Scripts.class);
            startActivity(intent);
        }
        return super.onKeyDown(keyCode, event);
    }
}
```

## WOL.java

---

```

package garciaponce.miguel;

import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;

public class WOL extends Activity {
    private EditText IPDif;
    private EditText DirMacWOL;
    private Button btnEncender;
    private String IP;
    private String MAC;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.wakeonlan);

        Bundle extras = getIntent().getExtras();
        if(extras != null)
        {
            IP = extras.getString("DirIP");
            MAC = extras.getString("DirMac");
        }

        IPDif = (EditText) findViewById(R.id.IPDifusion);
        DirMacWOL = (EditText) findViewById(R.id.DirMacWOL);
        IPDif.setText(IP);
        DirMacWOL.setText(MAC);

        btnEncender = (Button) findViewById(R.id.BtnEncender);
        btnEncender.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                WakeOnLan(IPDif.getText().toString(), DirMacWOL.getText().toString());
                Intent intent = new Intent(WOL.this, ListaHosts.class);
                startActivity(intent);
            }
        });
    }

    public static final int PORT = 9;

    protected void WakeOnLan(String ipStr, String macStr) {
        try {
            // Construcción del paquete mágico
            // El paquete mágico es una trama ethernet que comienza con 6
            // bytes de cabecera FF FF FF FF FF FF y sigue con 16
        }
    }
}

```

```

repeticiones de la dirección física MAC
byte[] macBytes = getMacBytes(macStr);
byte[] bytes = new byte[6 + 16 * macBytes.length];
for (int i = 0; i < 6; i++) {
    bytes[i] = (byte) 0xff;
}
for (int i = 6; i < bytes.length; i += macBytes.length)
{
    System.arraycopy(macBytes, 0, bytes, i, macBytes.
        length);
}

InetAddress address = InetAddress.getByName(ipStr);
DatagramPacket packet = new DatagramPacket(bytes, bytes.
    length, address, PORT);
DatagramSocket socket = new DatagramSocket();
socket.send(packet);
socket.close();

Log.i("WOL", "Wake-on-LAN packet sent.");
} catch (Exception e) {
    Log.i("WOL", "Failed to send Wake-on-LAN packet: " + e);
}

}

private byte[] getMacBytes(String macStr) throws
IllegalArgumentException {
byte[] bytes = new byte[6];
String[] hex = macStr.split("(:|-)");
// Si la longitud es distinta de 6 lanzamos excepción
if (hex.length != 6) {
    throw new IllegalArgumentException("Invalid MAC address .
        ");
}
try {
    // Convertimos a un vector de byte
    for (int i = 0; i < 6; i++) {
        bytes[i] = (byte) Integer.parseInt(hex[i], 16);
    }
} catch (NumberFormatException e) {
    throw new IllegalArgumentException("Direccion MAC
        hexadecimal invalida");
}
return bytes;
}
}

```

# Bibliografía

- [1] Como Programar en Java (DEITEL, HARVEY M. )
- [2] Learning Android (Marko Gargenta)
- [3] <http://www.sgoliver.net/> (blog sobre Android)
- [4] <http://developer.android.com/> (documentación oficial Android)
- [5] Ingeniería del Software. Un enfoque práctico. McGraw Hill 2005 (S. Pressman, Roger.)
- [6] Administración de sistemas Linux (Tom Adelstein, Bill Lubanovic)