Control de Sistemas Remotos mediante tecnología Android : REMSYS

Miguel García Ponce

Universidad de Cádiz

Indice General

Indice General

Descripción general del proyecto

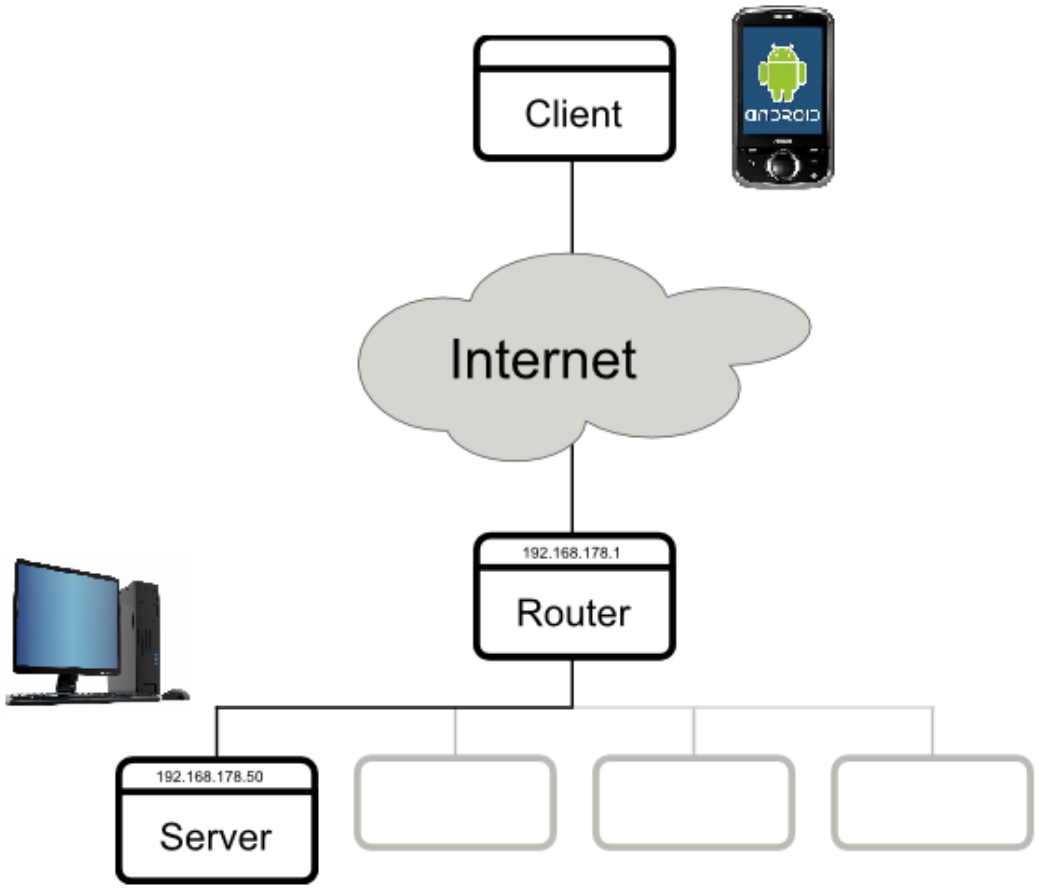
- El proyecto consistirá en la construcción de un software cliente y otro servidor, el cual el cliente se conectará mediante sockets a este segundo, y mediante un protocolo de comunicación obtener distinta información y poder actuar sobre ello desde un terminal móvil.
- Funcionalidades:

1. Encendido Remoto
2. Apagado
3. Reinicio
4. Información de discos (Particiones)
5. Información de red.
6. Información del S.O.
7. Encendido Remoto
8. Procesos
9. Navegación
10. Orden Libre
11. Scripts

Indice General

Entorno cliente-servidor

Esquema general del funcionamiento de RemSys



Indice General

Java 2 Standard Edition o J2SE (I)

Fundamentos de Java

- Java Platform, Standard Edition o Java SE, es una colección de APIs del lenguaje de programación Java útiles para muchos programas de la Plataforma Java.
- Las aplicaciones Java estan típicamente compiladas en un bytecode. En el tiempo de ejecucion, el bytecode es interpretado o compilado a codigo nativo para la ejecución, aunque la ejecucion directa por hardware del bytecode por un procesador Java tambien es posible. Podemos por lo tanto decir que es independiente de la plataforma, ya que se ejecuta en la maquina virtual de java (JVM).

Java 2 Standard Edition o J2SE (I)

Fundamentos de Java

- Java Platform, Standard Edition o Java SE, es una colección de APIs del lenguaje de programación Java útiles para muchos programas de la Plataforma Java.

- Las aplicaciones Java estan típicamente compiladas en un bytecode. En el tiempo de ejecución, el bytecode es 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).

Java 2 Standard Edition o J2SE (II)

Objetivos principales de Java

- Deberá usar el paradigma de la programación orientada a objetos.
- Debera permitir la ejecucion de un mismo programa en multiples sistemas operativos.
- Debera incluir por defecto soporte para trabajo en red.
- Debera diseñarse para ejecutar codigo en sistemas remotos de forma segura.
- Debera ser facil de usar y tomar lo mejor de otros lenguajes orientados a objetos, como C++.

Java 2 Standard Edition o J2SE (II)

Objetivos principales de Java

- Deberá usar el paradigma de la programación orientada a objetos.
- Deberá permitir la ejecución de un mismo programa en múltiples sistemas operativos.
- Debera incluir por defecto soporte para trabajo en red.
- Debera diseñarse para ejecutar codigo en sistemas remotos de forma segura.
- Debera ser facil de usar y tomar lo mejor de otros lenguajes orientados a objetos, como C++.

Java 2 Standard Edition o J2SE (II)

Objetivos principales de Java

- Deberá usar el paradigma de la programación orientada a objetos.
- Deberá permitir la ejecución de un mismo programa en múltiples sistemas operativos.
- Deberá incluir por defecto soporte para trabajo en red.
- Debera diseñarse para ejecutar codigo en sistemas remotos de forma segura.
- Debera ser facil de usar y tomar lo mejor de otros lenguajes orientados a objetos, como C++.

Java 2 Standard Edition o J2SE (II)

Objetivos principales de Java

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

Java 2 Standard Edition o J2SE (II)

Objetivos principales de Java

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

Indice General

Plataforma Android (I)

Android: Información general

Android es una plataforma de software y un sistema operativo para dispositivos móviles basada en un kernel Linux, desarrollada por Google y mas 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 ocialmente por Google. La mayor parte de la plataforma de Android esta disponible bajo licencia de software libre de Apache y otras licencias de código abierto.

Plataforma Android (II)

Soporte de Java

Aunque la mayoría de las aplicaciones estan 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 dipositivos móviles que funcionan con batería y que tienen memoria y procesador limitados.

- Inconveniente: RMI (Java Remote Method Invocation) es un mecanismo ofrecido por Java para invocar un método de manera remota, no soportado en la plataforma Android.

Maquina Virtual Dalvik $\neq$ Java Virtual Machine

Plataforma Android (III)

Arquitectura Android

Básicamente, Android tiene las siguientes capas:

1. aplicaciones (escritas en java, ejecutandose en la máquina de Dalvik)
2. bibliotecas y servicios del framework (escritos mayormente en java)
3. el código de las aplicaciones y la mayoría de los frameworks se ejecutan en una máquina virtual
4. bibliotecas nativas, demonios y servicios (escritos en C o C++)
5. el núcleo de Linux, que incluye los drivers para hardware, red, sistema de ficheros y comunicación entre procesos

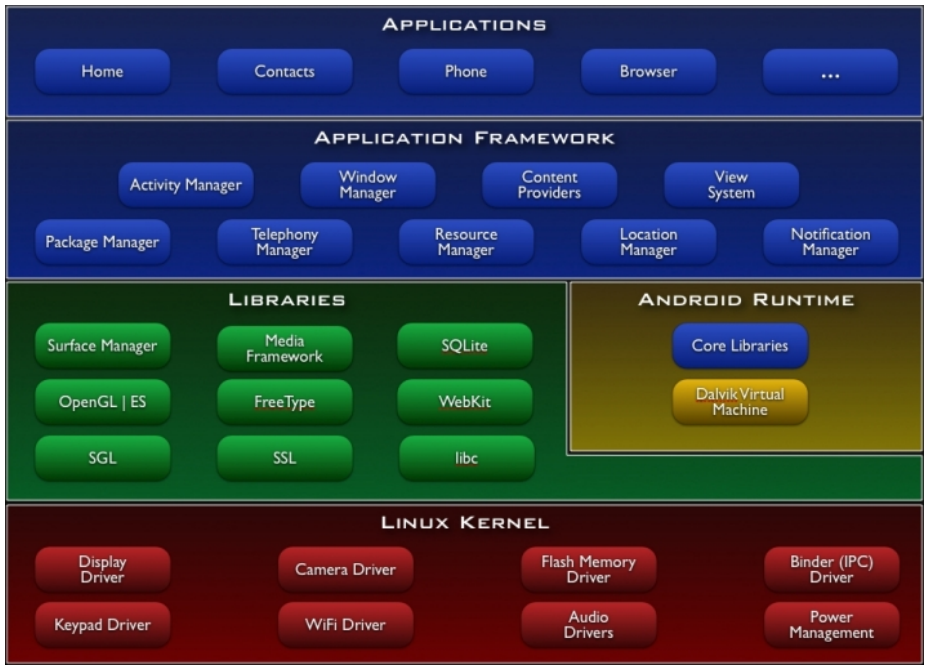


Figura : Arquitectura plataforma Android

Plataforma Android (IV)

Actualizaciones Android

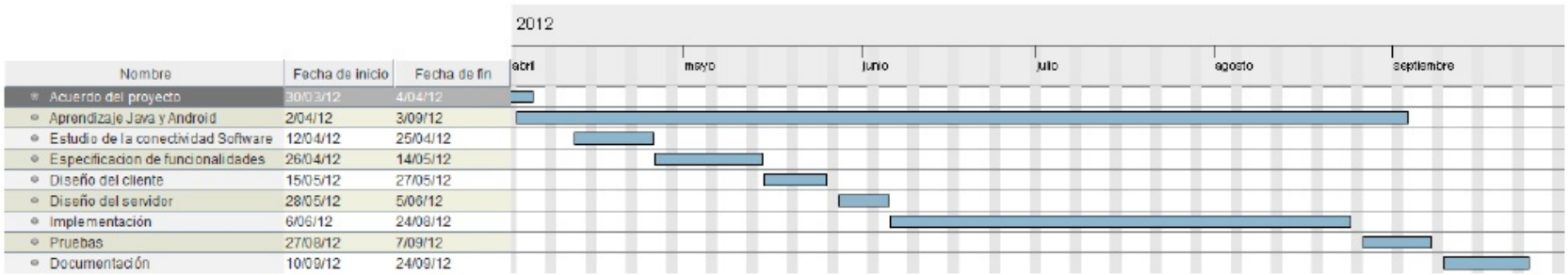
Es una pieza fundamental en el éxito del sistema Android. Las actualizaciones permiten que el sistema este siempre en continua evolución, solventando los principales errores (bugs) encontrados en versiones anteriores y optimizando otros muchos aspectos para hacer que el sistema sea mucho mas flexible y eficiente.

Platform	Codename	API Level	Distribution
Android 1.5	Cupcake	3	0.3%
Android 1.6	Donut	4	0.7%
Android 2.1	Eclair	7	5.5%
Android 2.2	Froyo	8	20.9%
Android 2.3 - Android 2.3.2	Gingerbread	9	0.5%
Android 2.3.3 - Android 2.3.7		10	63.9%
Android 3.0	Honeycomb	11	0.1%
Android 3.1		12	1.0%
Android 3.2		13	2.2%
Android 4.0 - Android 4.0.2	Ice Cream Sandwich	14	0.5%
Android 4.0.3 - Android 4.0.4		15	4.4%

Figura : Versiones y actualizaciones Android

Indice General

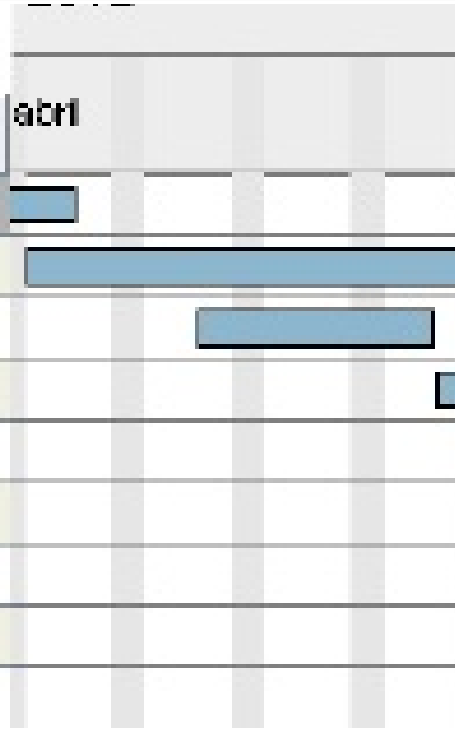
Diagrama de Gantt



Nombre	Fecha de inicio	Fecha de fin
Acuerdo del proyecto	30/03/12	4/04/12
Aprendizaje Java y Android	2/04/12	3/09/12
Estudio de la conectividad Software	12/04/12	25/04/12
Especificacion de funcionalidades	26/04/12	14/05/12
Diseño del cliente	15/05/12	27/05/12
Diseño del servidor	28/05/12	5/06/12
Implementación	6/06/12	24/08/12
Pruebas	27/08/12	7/09/12
Documentación	10/09/12	24/09/12

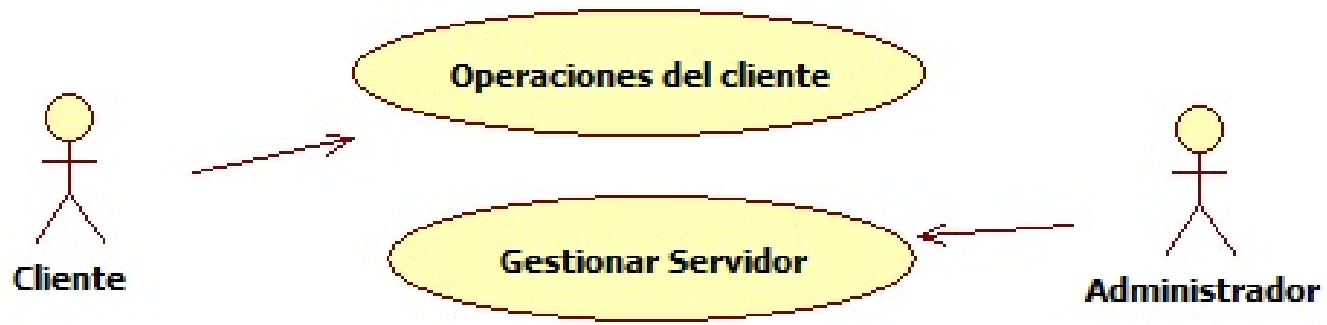
Diagrama de Gantt

Nombre	Fecha de inicio	Fecha de fin
Acuerdo del proyecto	30/03/12	4/04/12
Aprendizaje Java y Android	2/04/12	3/09/12
Estudio de la conectividad Software	12/04/12	25/04/12
Especificacion de funcionalidades	26/04/12	14/05/12
Diseño del cliente	15/05/12	27/05/12
Diseño del servidor	28/05/12	5/06/12
Implementación	6/06/12	24/08/12
Pruebas	27/08/12	7/09/12
Documentación	10/09/12	24/09/12



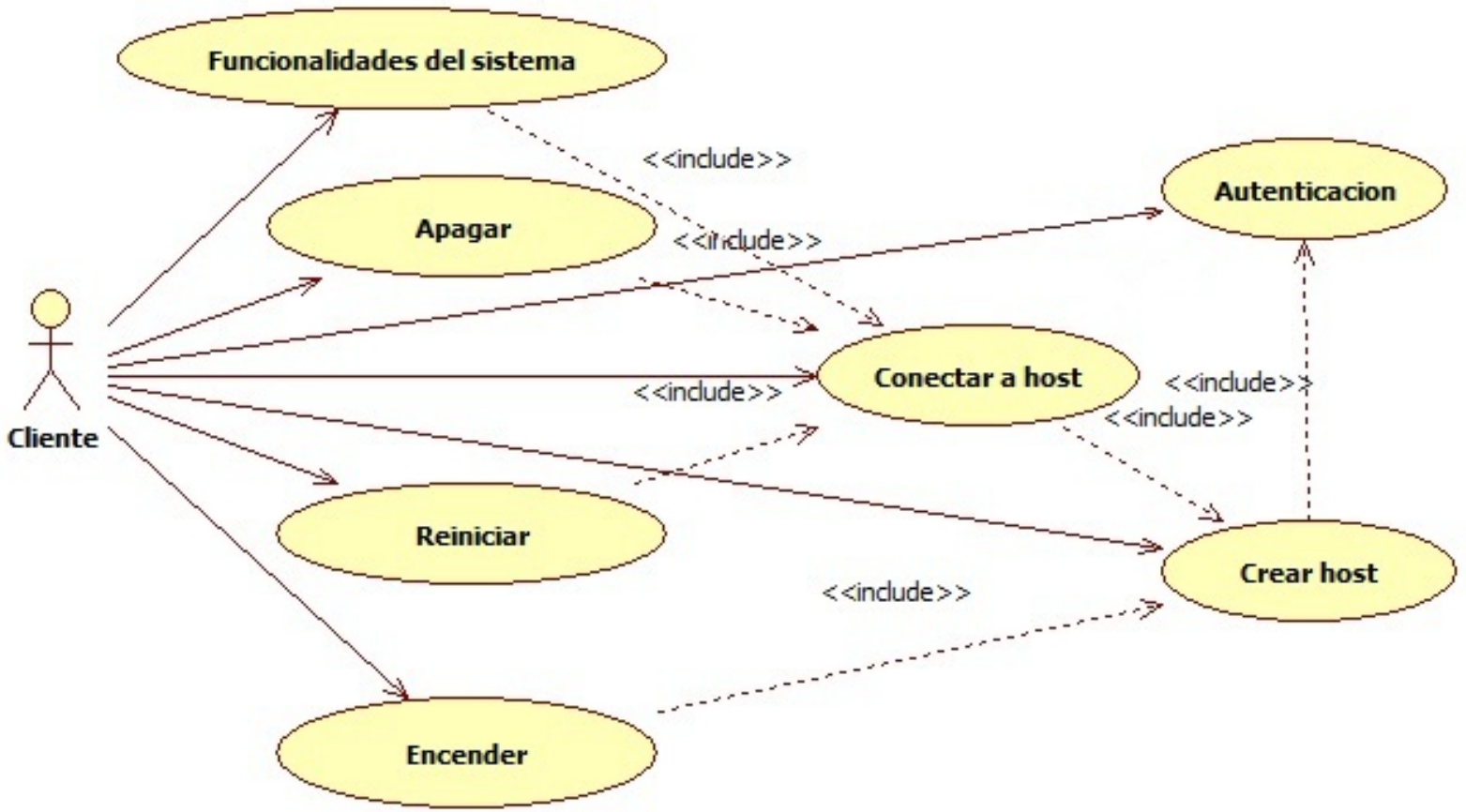
Indice General

Análisis(I)
Casos de uso general



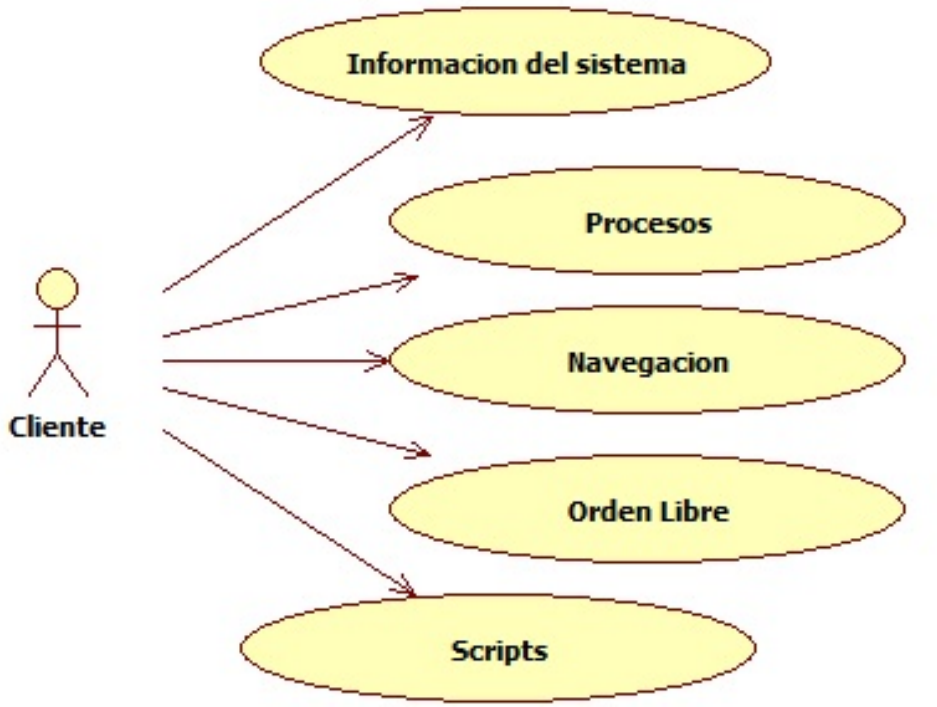
Análisis(II)

Caso de uso Operaciones del cliente



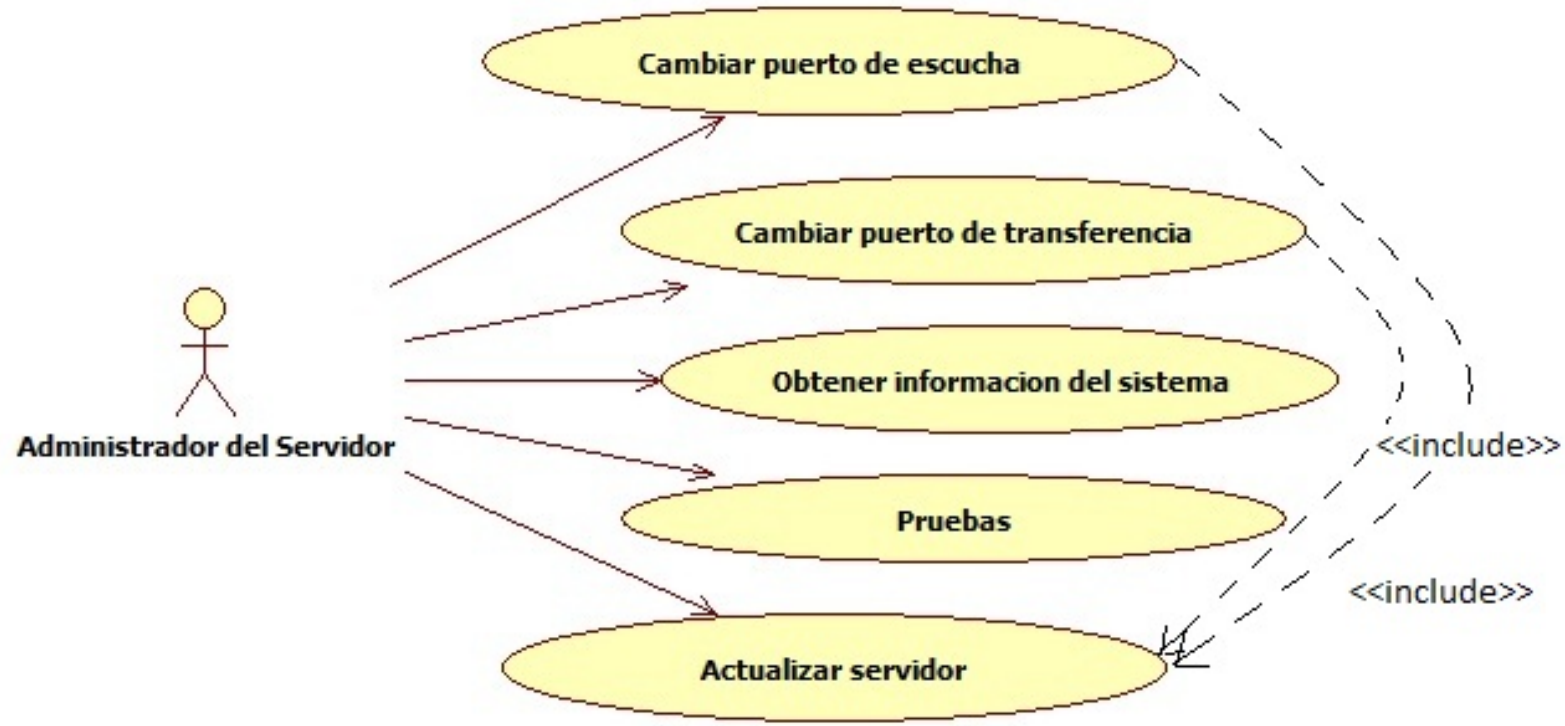
Análisis(III)

Caso de uso Funcionalidades del sistema



Análisis(IV)

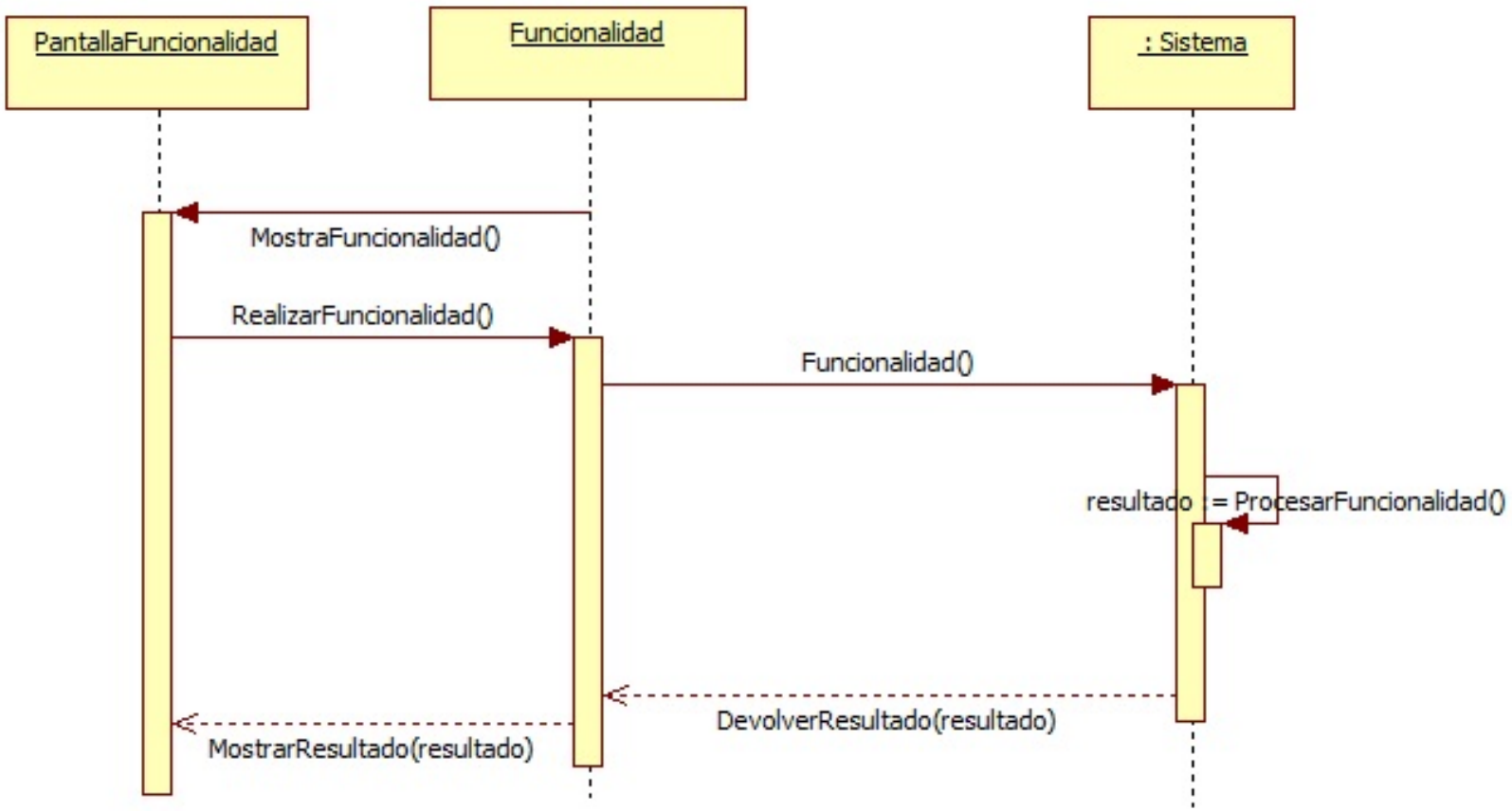
Caso de uso Funcionalidad del Administrador del Servidor



Indice General

Diseño

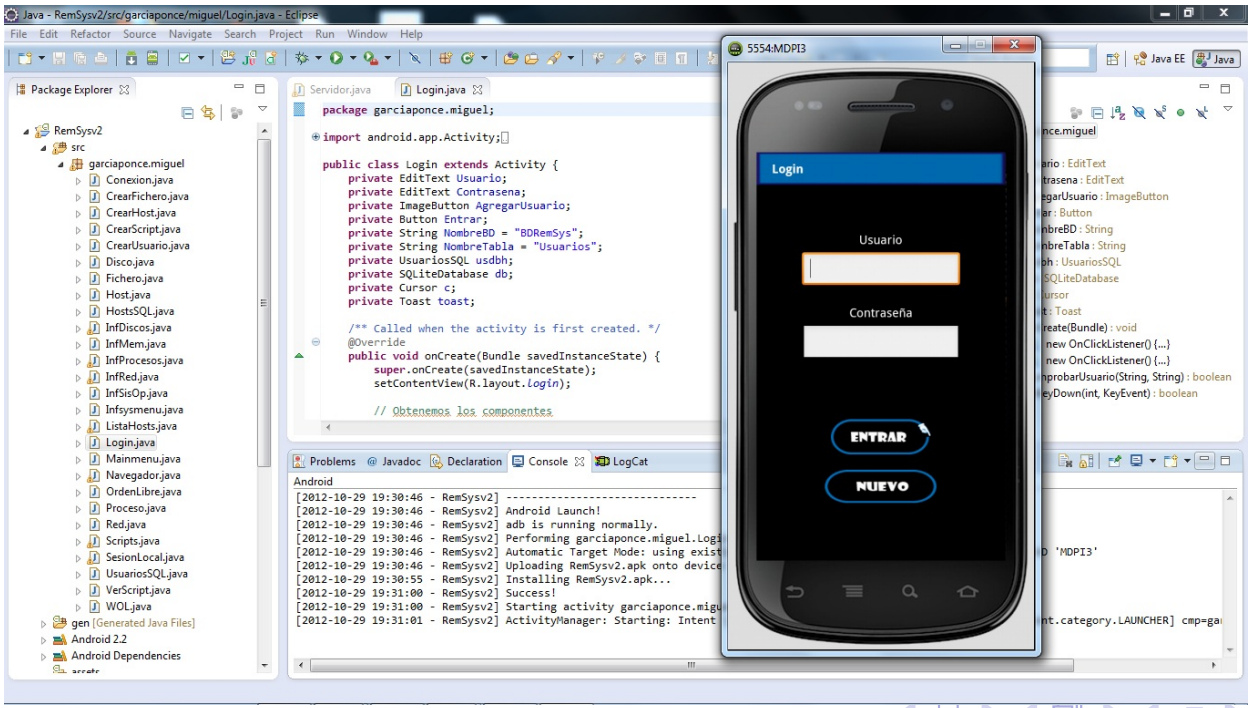
Diagrama de secuencia general del procesamiento de una funcionalidad



Indice General

Eclipse y Android SDK

Eclipse es un entorno de desarrollo integrado de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Además se puede integrar el Kit de Desarrollo de Software de Android (SDK)



Indice General

Funcionalidades

Funciones cliente:

1. Crear/Eliminar/Conectar host.
2. Encender/Apagar/Reiniciar host.
3. Información discos, red, memoria y sistema operativo.
4. Listado y eliminacion de procesos.
5. Navegación por el árbol de directorios.
6. Orden libre.
7. Ver/Ejecutar/Eliminar/Crear Scripts.

Funciones servidor:

1. Cambiar puerto escucha/transferencia.
2. Procesar funcionalidad desde cliente.
3. Información de direcciones IPs y MACs.
4. Pruebas.
5. Actualizar servidor.

Indice General

Conectividad mediante Sockets

Conexión: Sockets

Socket designa un concepto abstracto por el cual dos programas (posiblemente situados en computadoras distintas) pueden intercambiar cualquier flujo de datos, generalmente de manera fiable y ordenada.

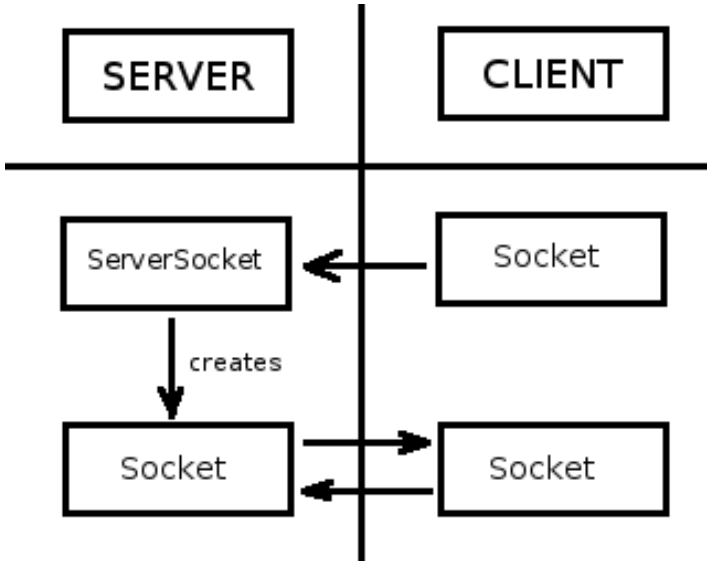


Figura : Sockets

Conectividad mediante Sockets

Código de creación, escucha y procesado del servidor

```
public void EjecutarServidor(int Puerto){
        try{
            servidor= new ServerSocket(Puerto);

            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();
        }
}
```

Conectividad mediante Sockets

Código de creación del socket cliente

```
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();
```

Indice General

Implementación del servidor

Implementación del servidor

Características del servidor RemSys:

1. Implementación mediante Java Swing
2. Espera conexiones entrantes de un dispositivo móvil Android(cliente), mediante Sockets.
3. Obtiene información mediante comandos del sistema
4. Procesa y envía la información obtenida de vuelta al cliente para su presentación.

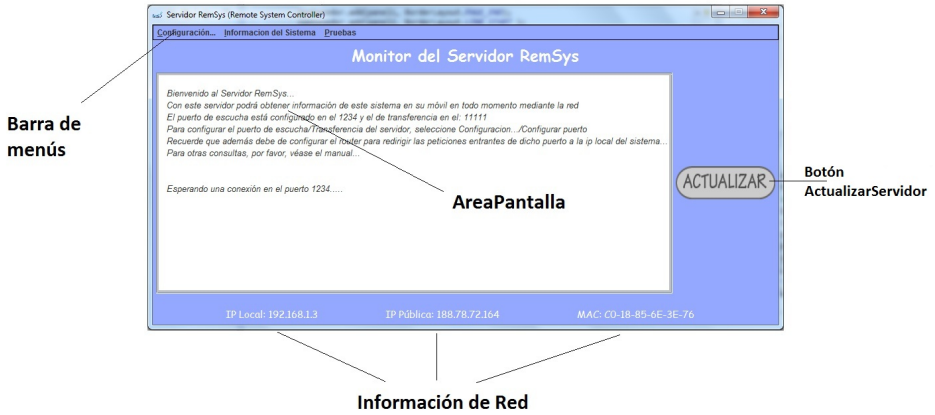
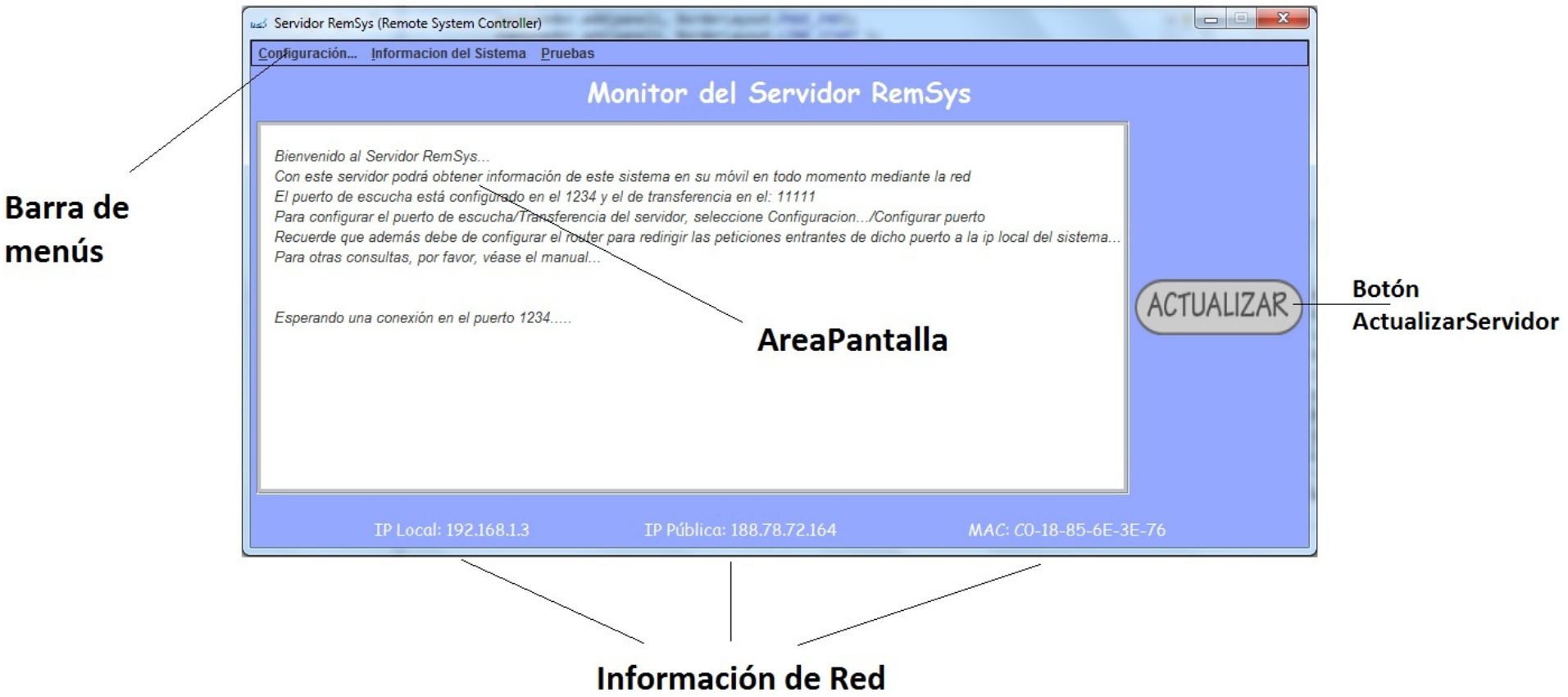


Figura : Implementación del servidor

Implementación del servidor

Implementación del servidor



Comandos utilizados

Comandos utilizados al realizar las funcionalidades

Funcionalidad	Windows	Linux
Información de discos	fsutil fsinfo, fsutil volume	df
Información de red	ipconfig	ifconfig
Información de Sistema Operativo	System.getProperty	System.getProperty
Información de Memoria	systeminfo	cat /proc/meminfo
Procesos	tasklist, taskkill	ps,kill
Renombrar	move	mv
Copiar/Cortar-Pegar	copy, move	cp,mv
Apagar	shutdown -s	halt
Reiniciar	shutdown -r	reboot

Indice General

Implementación del cliente

Acceso a las funcionalidades

Pasos previos para el acceso a las funcionalidades:

1. Crear un usuario mediante nombre y contraseña.
2. Acceder con las credenciales anteriormente creadas.
3. Crear un host al cual conectarse especificando:
    1. Nombre del host
    2. Dirección IP
    3. Puerto
    4. Dirección MAC (opcional)
4. Conectarse al host creado.

Implementación del cliente

Acceso a las funcionalidades

Menú Principal

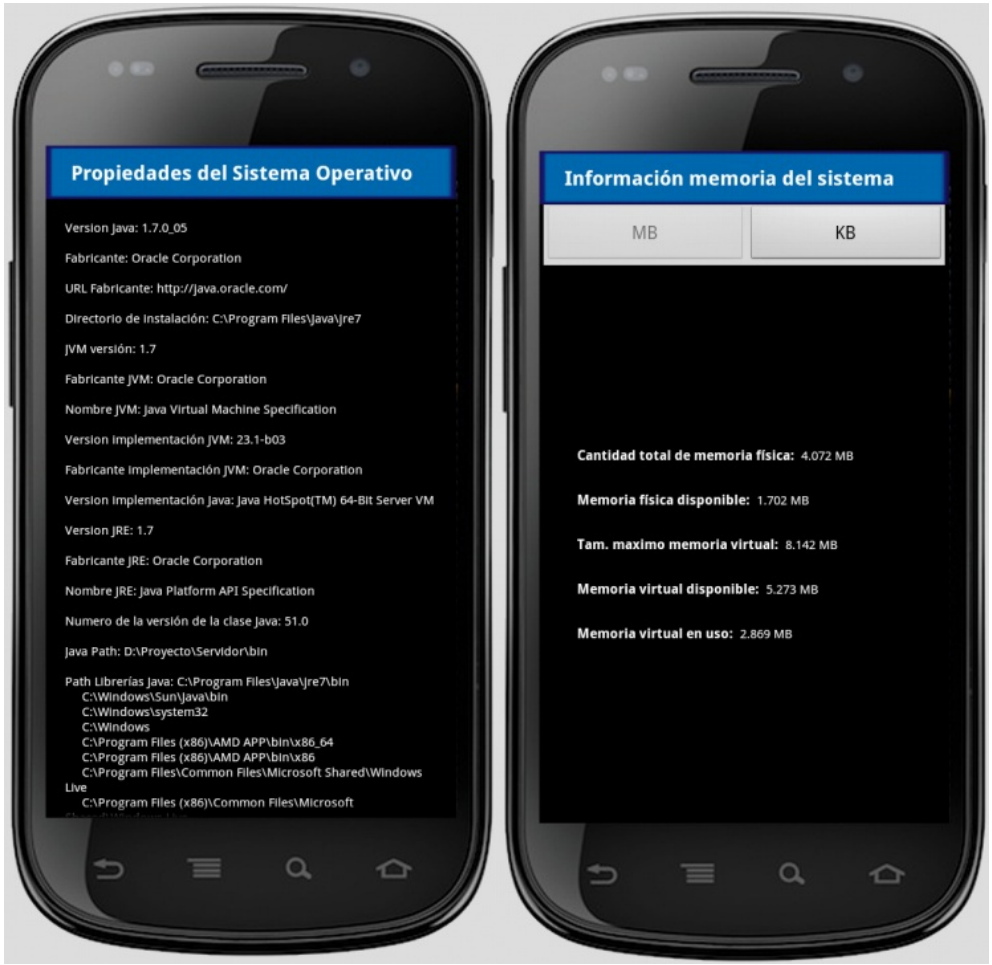
Acceso a las funcionalidades: Menús , Apagar y Reiniciar

Información del sistema

Información de discos y red

Información del sistema

Información del Sistema Operativo y memoria



Procesos

Listado de procesos y eliminar proceso

Navegación por el árbol de directorios

Navegador

Las funcionalidades que podemos realizar en dicha actividad son:

1. Ejecución de ficheros: .exe, .bat, .java, .sh, .run, .py y .bin
2. Compilación y ejecución de ficheros .java
3. Eliminar y renombrar ficheros
4. Cortar/Copiar y pegar ficheros
5. Crear fichero
6. Transferir fichero a SD-Card

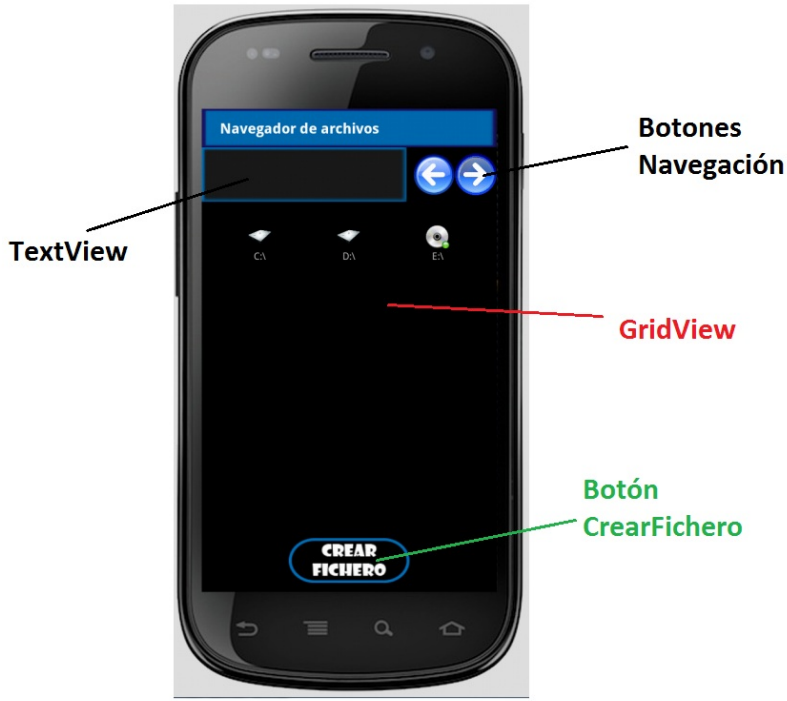


Figura : Funcionalidad Navegación

Orden Libre
Orden Libre

Para realizar esta funcionalidad seguimos los siguientes pasos:

1. Introducir orden en el formulario
2. Pulsar sobre el botón Ejecutar

Obtendremos la salida de la orden en el recuadro inferior.

Figura : Funcionalidad Orden Libre

Scripts
Scripts

Las funcionalidades que podemos realizar aquí son:

1. Ver Script
2. Ejecutar Script
3. Eliminar Script
4. Crear Script

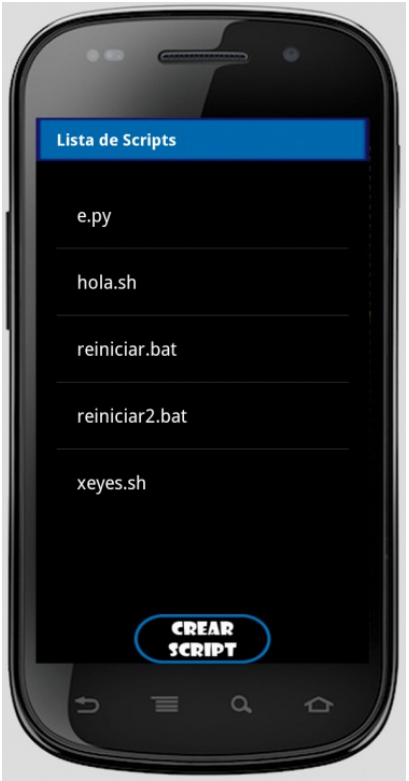


Figura : Funcionalidad Scripts

Indice General

Problemas

Problemas encontrados en la realización del proyecto

1. No existencia de RMI en la máquina virtual de Dalvik.
2. Procesamiento de la información obtenida.
3. Sincronización para la transferencia de ficheros a SD-Card.
4. Codificación de caracteres en la salida de las órdenes del sistema (Cp850).

Indice General

Conclusiones

- Aprendizaje Java y Android.
- Satisfacción personal al realizar un proyecto tan flexible y que permite realizar muchas funcionalidades.
- Visión web.
- Adquisición de conocimientos.

Fin

Gracias por su atención,

¿Alguna pregunta?