Ficheros

paco@portadaalta.es

Índice

- Introducción
- Alternativas para guardar datos permanentemente
- Ficheros
- Internet
- Dónde se almacenan los ficheros
- Permisos de los ficheros
- Almacenamiento en memoria interna
- Almacenamiento en memoria externa
- Paquetes
- Propiedades de un fichero

Índice

- Escribir en un fichero en memoria interna
- Comprobar la memoria externa
- Escribir en memoria externa
- Ejercicios de escritura
- Leer de un fichero en memoria interna
- Leer de memoria externa
- Acceso a un fichero de los recursos
- Acceso a un fichero en /assets
- Ejercicios de lectura
- Usar un explorador de archivos
- Material File Picker
- Ejercicio propuesto

Introducción

- Los programas guardan datos en variables y objetos.
- El problema es que estas estructuras se almacenan en memoria volátil y desaparecen cuando se apaga el dispositivo.
- La solución es guardar en un sistema de almacenamiento permanente la información que nos interese conservar.
- Hay varias alternativas para almacenar la información.

Alternativas para guardar datos permanentemente

- **Ficheros**

Puedes almacenar los ficheros en la memoria interna del dispositivo o en un medio de almacenamiento removible como una tarjeta SD.

- **Internet**

No te olvides que también puedes usar la nube para almacenar y recuperar datos.

- **Bases de datos**

Las APIs de Android contienen soporte para SQLite. Tu aplicación puede crear y usar base de datos SQLite de forma muy sencilla y con toda la potencia que nos da el lenguaje SQL.

- **Proveedores de contenidos**

Es un componente opcional de una aplicación que expone el acceso de lectura / escritura de sus datos a otras aplicaciones.

Ficheros

- Se pueden usar tres tipos de ficheros en Android:
  - En la memoria interna
  - En la memoria externa
  - En los recursos de la aplicación

- Android hereda el sistema de ficheros de Linux.

- Por defecto, los ficheros solo son accesibles por la aplicación que los crea.

- También se puede dar acceso de lectura o escritura al resto de aplicaciones.

Internet

Los dispositivos Android suelen disponer de conexión a Internet (por wifi o por tarifa de datos) por lo que también se puede almacenar y recuperar datos en la red (en nuestros propios servidores o en la nube).

Ventajas:

- Compartir información con otros usuarios
- Acceso a los datos desde distintos dispositivos

Desventajas:

- La aplicación no funciona si no dispone de conexión

Alternativas:

- Definir nuestro propio protocolo (usando sockets)
- Usar protocolos de transferencia de ficheros (http, ftp, smtp, . . .)
- Servicios web

Dónde se almacenan los ficheros

- En la memoria interna:
  - Se crea una específica para cada aplicación
  - Se elimina cuando se desinstala la aplicación

- En la memoria externa:
  - Normalmente hay una tarjeta extraíble (SD)
  - Se utiliza para compartir ficheros

- En los recursos de la aplicación:
  - Son de solo lectura
  - Se añaden en el proceso de creación de la aplicación

Permisos de los ficheros

- Android hereda el sistema de ficheros de Linux:
  - Cuando se instala una aplicación se crea un nuevo usuario que se asocia a la aplicación

- Por defecto, los ficheros solo son accesibles por la aplicación que los crea (MODE_PRIVATE). Ni siquiera el usuario del teléfono tendrá acceso.

- Si queremos dar acceso al resto de aplicaciones podemos usar los modos MODE_WORLD_READABLE y MODE_WORLD_WRITEABLE.

-Existe un cuarto modo para abrir un fichero y añadir información al final: MODE_APPEND.

Almacenamiento en la memoria interna

- Los ficheros de uso exclusivo de la aplicación han de estar almacenados en la carpeta reservada para la aplicación:

/data/data/nombre_del_paquete/files

- Se borran al desinstalar la aplicación.

- La información sobre la ruta a los ficheros de la aplicación está en el contexto de la aplicación.

- Obtención del contexto:

    getApplicationContext ()

Almacenamiento en la memoria externa

- Los dispositivos Android suelen disponer de memoria externa.

- Antes de acceder a ella hay que comprobar su estado con
Environment.getExternalStorageState()

- La memoria externa suele montarse en */sdcard* o */media/sdcard* o */mnt/sdcard* , pero es más seguro obtener su ruta:
Environment.getExternalStorageDirectory()

- A partir de la versión 1.6 hay que declarar el permiso de escritura (WRITE_EXTERNAL_STORAGE) en el manifiesto si se va a escribir en la tarjeta:

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

- A partir de la vesión 2.2 (API 8) se dispone de carpetas de uso específico de cada aplicación.

Paquetes

- Se puede utilizar cualquier clase del paquete **java.io** para trabajar con ficheros.

Clases:

fichero	File
lectura	FileInputStream InputStreamReader BufferedReader
escritura	FileOutputStream OutputStreamWriter BufferedWriter

Excepciones

IOException	Error de Entrada/Salida
FileNotFoundException	Fichero no encontrado

Propiedades de un fichero

File miFichero;
La clase File es una representación “abstracta” de una entidad del sistema de ficheros identificada por una ruta.

miFichero = new File(getFilesDir(), nombreFichero);
Constructor de la clase File usando la ruta y el nombre del fichero.
El fichero referenciado por un objeto de la clase File puede existir o no.

miFichero.exists()
Devuelve un valor lógico, un boolean, indicando si el fichero puede ser encontrado en el sistema de ficheros subyacente.

miFichero.getAbsolutePath()
Devuelve la ruta absoluta al fichero.

miFichero.lastModified()
Devuelve el tiempo (un valor long) en el que fue modificado el fichero por última vez, medido en milisegundos desde el 1 de Enero de 1970.

miFichero.length()
Devuelve el tamaño del archivo en bytes

Escribir en un fichero en memoria interna (I)

```
String nombreFichero = "datos.txt";
String texto = "texto almacenado";
FileOutputStream fos = null;

try {
    //fos = getApplicationContext().openFileOutput(nombreFichero,Context.MODE_PRIVATE);
    fos = openFileOutput(nombreFichero,Context.MODE_PRIVATE);
    fos.write(texto.getBytes());
    //fos.close();
    } catch (FileNotFoundException e) {
        Log.e("Error: el fichero no existe",e.getMessage());
    } catch (IOException e) {
        Log.e("Error de E/S", e.getMessage());
    } finally {
            try {
                if (fos != null)
                    fos.close();
            } catch (IOException e) {
                    Log.e("Error al cerrar el fichero", e.getMessage());
            }
      }
}
```

Escribir en un fichero en memoria interna (II)

```
String nombreFichero = "fichero.txt";
String texto = "texto almacenado";
File miFichero;
FileOutputStream fos = null;

try {
    //mifichero = new File(getApplicationContext().getFilesDir(), nombreFichero);
    miFichero = new File(getFilesDir(), nombreFichero);
    fos = new FileOutputStream (miFichero) ;
    fos.write(texto.getBytes());
    } catch . . .

}
```

Escribir en un fichero en memoria interna (III)

```
 String cadena = “texto a guardar”;
String nombreFichero = "resultado.txt";
 String codigo = “UTF-8”;
File miFichero;
FileOutputStream fos = null;
 OutputStreamWriter osw = null;
 BufferedWriter out = null;

 try {
    //mifichero = new File(getApplicationContext().getFilesDir(), nombreFichero);
    miFichero = new File(getFilesDir(), nombreFichero);
    fos = new FileOutputStream(miFichero);
     osw = new OutputStreamWriter(fos, codigo);
    out = new BufferedWriter(osw, 32768);  //tamaño del buffer de 32 kbytes
    out.write(cadena);
} catch (IOException excep){
    Log.e("Error", excep.getMessage());
} finally {
 try {
        if (out!= null)
        out.close();
 } catch (IOException e) {
        Log.e("Error al cerrar", e.getMessage());
 }
}
```

Escribir en un fichero en memoria interna (IV)

```
public boolean escribirInterna(String fichero, String cadena, Boolean anadir, String codigo) {
    File miFichero;

    miFichero = new File(contexto.getFilesDir(), fichero);
    return escribir(miFichero, cadena, anadir, codigo);
}

private boolean escribir(File fichero, String cadena, Boolean anadir, String codigo) {
    FileOutputStream fos = null;
    OutputStreamWriter osw = null;
    BufferedWriter out = null;
    boolean correcto = false;

    try {
        fos = new FileOutputStream(fichero, anadir);
        osw = new OutputStreamWriter(fos, codigo);
        out = new BufferedWriter(osw);
        out.write(cadena);
    } catch (IOException e) {
        Log.e("Error de E/S", e.getMessage());
    } finally {
        try {
            if (out != null) {
                out.close();
                correcto = true;
            }
        } catch (IOException e) {
            Log.e("Error al cerrar", e.getMessage());
        }
    }
    return correcto;
}
```

Escribir en un fichero en memoria interna (IV)

```
public String mostrarPropiedadesInterna (String fichero) {
        File miFichero;

        miFichero = new File(contexto.getFilesDir(), fichero);

        return mostrarPropiedades(miFichero);
    }

    public String mostrarPropiedades (File fichero) {
        SimpleDateFormat formato = null;
        StringBuffer txt = new StringBuffer();

        try {
            if (fichero.exists()) {
                txt.append("Nombre: " + fichero.getName() + '\n');
                txt.append("Ruta: " + fichero.getAbsolutePath() + '\n');
                txt.append("Tamaño (bytes): " + Long.toString(fichero.length()) + '\n');
                formato = new SimpleDateFormat("dd-MM-yyyy hh:mm:ss", Locale.getDefault());
                txt.append("Fecha: " + formato.format(new Date(fichero.lastModified())) + '\n');
            } else
                txt.append("No existe el fichero " + fichero.getName() + '\n');

        } catch (Exception e) {
            Log.e("Error", e.getMessage());
            txt.append(e.getMessage());
        }
        return txt.toString();
    }
```

Comprobar la memoria externa

Se usa el método *getExternalStorageState()*, de la clase Environment, que devuelve varios estados:

MEDIA_MOUNTED_READ_ONLY: solo se puede leer

MEDIA_MOUNTED: se puede leer y escribir

Comprobar la memoria externa

```
public boolean disponibleEscritura(){
        boolean escritura = false;

        //Comprobamos el estado de la memoria externa (tarjeta SD)
        String estado = Environment.getExternalStorageState();
        if (estado.equals(Environment.MEDIA_MOUNTED))
           escritura = true;

        return escritura;
}

public boolean disponibleLectura(){
        boolean lectura = false;

        //Comprobamos el estado de la memoria externa (tarjeta SD)
        String estado = Environment.getExternalStorageState();
        if (estado.equals(Environment.MEDIA_MOUNTED_READ_ONLY)
              || estado.equals(Environment.MEDIA_MOUNTED))
           lectura = true;

        return lectura;
}
```

Escribir en memoria externa

```
public boolean escribirExterna(String fichero, String cadena, Boolean anadir, String codigo) {
    File miFichero, tarjeta;

    tarjeta = Environment.getExternalStorageDirectory();
    //tarjeta = Environment.getExternalStoragePublicDirectory("datos/programas/");
    //tarjeta.mkdirs();
    miFichero = new File(tarjeta.getAbsolutePath(), fichero);

    return escribir(miFichero, cadena, anadir, codigo);
}
```

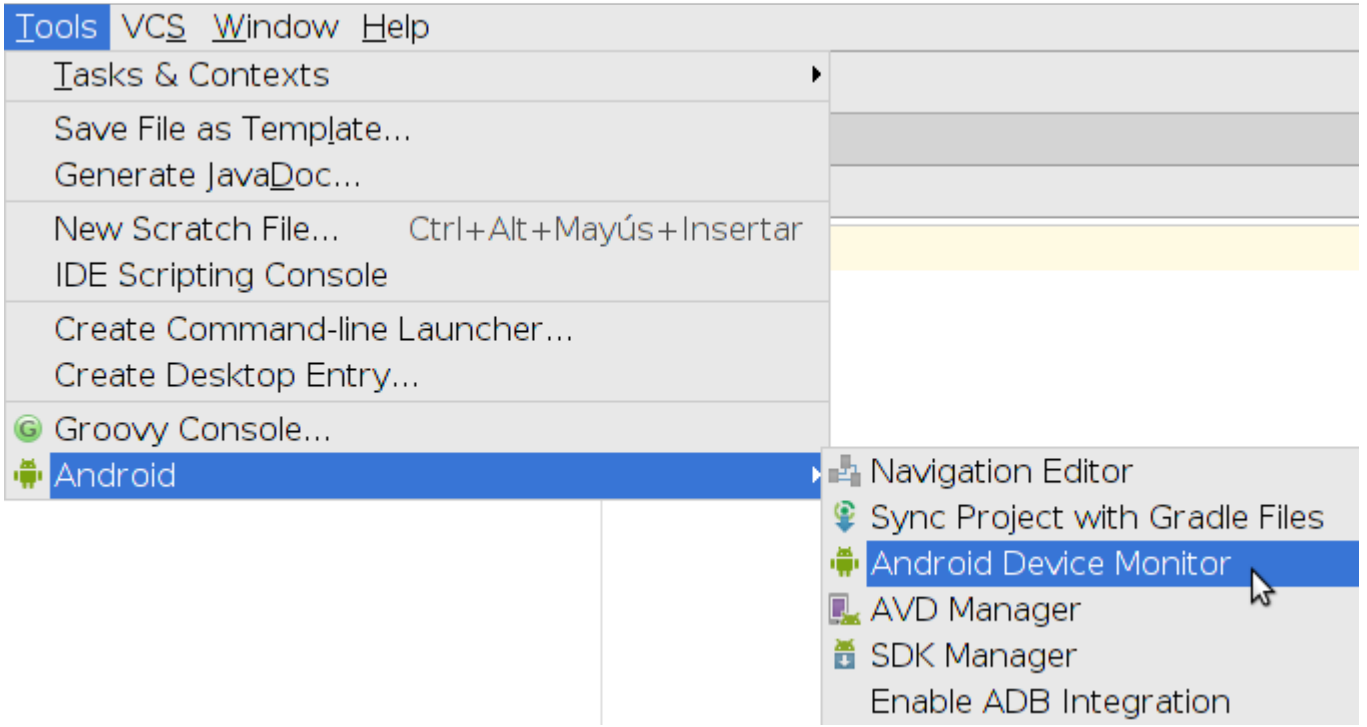
No olvidar poner el permiso de escritura en el manifiesto:
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>

Ejercicio: Escribir en memoria interna

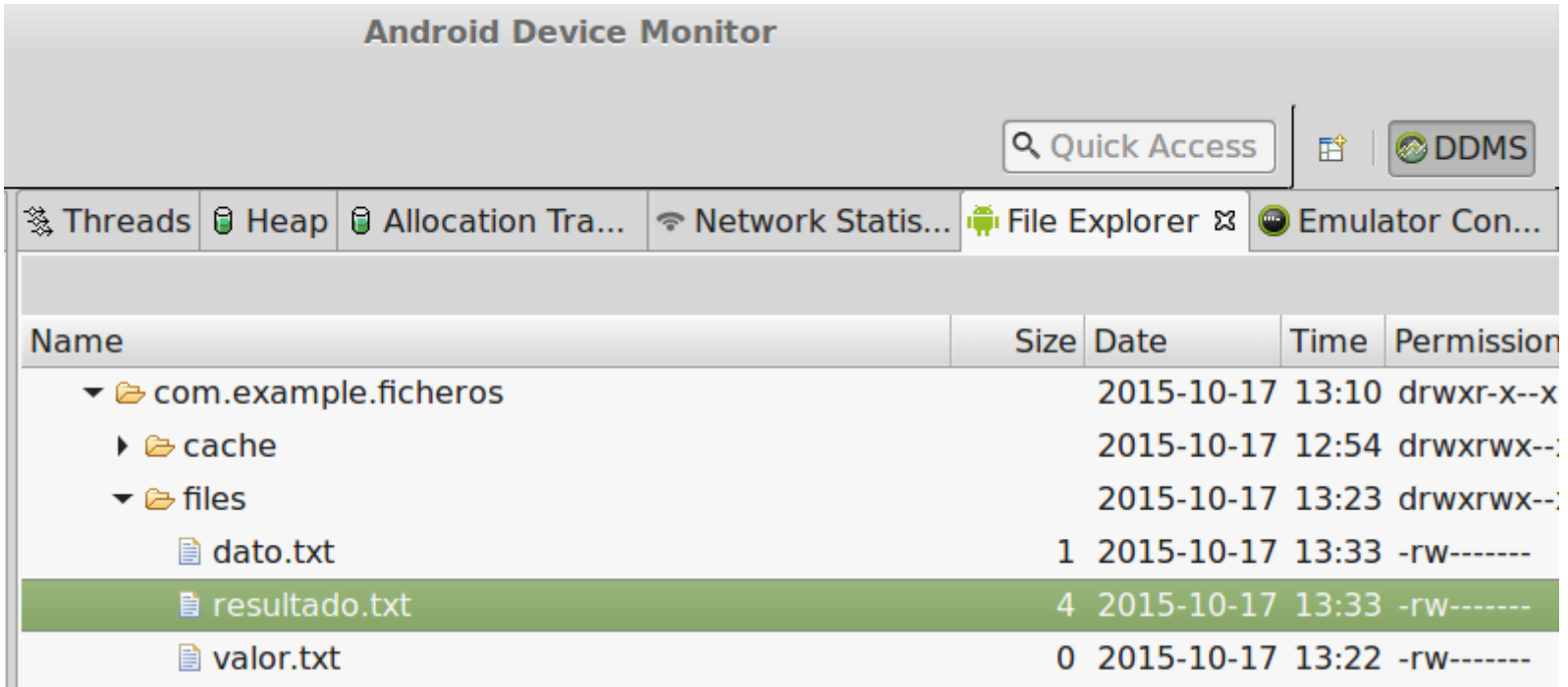
Crear una aplicación que sume dos números y almacene el resultado en un fichero en memoria interna.

También se mostrarán las propiedades del fichero creado: su ruta, su tamaño y la fecha de la última modificación.

Ejercicio: Escribir en memoria interna



Android Device Monitor



Ejercicio: Escribir en memoria externa

Crear una aplicación que sume dos números y almacene el resultado en un fichero en memoria externa.

También se mostrarán las propiedades del fichero creado: su ruta, su tamaño y la fecha de la última modificación.

Leer de un fichero en memoria interna (I)

```
public String leerInterna(String fichero){
	FileInputStream fis = null;
	StringBuilder miCadena = new StringBuilder();
	int n;
	boolean correcto = false;

	try {
		fis = getApplicationContext().openFileInput(fichero);
		while ((n = fis.read()) != -1)
			miCadena.append((char) n);
	} catch (IOException e) {
		Log.e("Error", e.getMessage());
	} finally {
		try {
			if (fis != null) {
				fis.close();
				correcto = true;
			}
		} catch (IOException e) {
			Log.e("Error al cerrar", e.getMessage());
		}
	}
	if (correcto)
		return miCadena.toString();
	else
		return "0";
}
```

Leer de un fichero en memoria interna (II)

```
public String leerInterna(String fichero) {
        File miFichero;
        FileInputStream fis = null;
        StringBuilder miCadena = new StringBuilder();
        int n;
        boolean correcto = false;

        try {
                //mifichero = new File(getApplicationContext().getFilesDir(), nombreFichero);
            miFichero = new File(getFilesDir(), fichero);
                fis = new FileInputStream (miFichero) ;
                while ((n = fis.read()) != -1)
                        miCadena.append((char) n);
        } catch . . .
```

Leer de un fichero en memoria interna (III)

```
public class Resultado {
    private boolean codigo;  //true es correcto y false indica error
    private String mensaje;
    private String contenido;

    public boolean getCodigo() {
        return codigo;
    }
    public void setCodigo(boolean codigo) {
        this.codigo = codigo;
    }
    public String getMensaje() {
        return mensaje;
    }
    public void setMensaje(String mensaje) {
        this.mensaje = mensaje;
    }
    public String getContenido() {
        return contenido;
    }
    public void setContenido(String contenido) {
        this.contenido = contenido;
    }
}
```

Leer de un fichero en memoria interna (III)

```
public Resultado leerInterna(String fichero, String codigo){
    File miFichero;

    //mifichero = new File(getApplicationContext().getFilesDir(), nombreFichero);
    miFichero = new File(contexto.getFilesDir(), fichero);

    return  leer(miFichero, codigo);
}
```

Leer de un fichero en memoria interna (III)

```
private Resultado leer(File fichero, String codigo){
    FileInputStream fis = null;
    InputStreamReader isw = null;
    BufferedReader in = null;
    //String linea;
    StringBuilder miCadena = new StringBuilder();
    Resultado resultado= new Resultado();
    int n;

    resultado.setCodigo(true);
    try {
        fis = new FileInputStream(fichero);
        isw = new InputStreamReader(fis, codigo);
        in = new BufferedReader(isw);
        while ((n = in.read()) != -1)
            miCadena.append((char) n);
        //while ((linea = in.readLine()) != null)
            //miCadena.append(linea).append('\n');
```

Leer de un fichero en memoria interna (III)

```
    } catch (IOException e) {
        Log.e("Error", e.getMessage());
        resultado.setCodigo(false);
        resultado.setMensaje(e.getMessage());
    } finally {
        try {
            if (in != null) {
                in.close();
                resultado.setContenido(miCadena.toString());
            }
        } catch (IOException e) {
            Log.e("Error al cerrar", e.getMessage());
            resultado.setCodigo(false);
            resultado.setMensaje(e.getMessage());
        }
    }
    return resultado;
}
```

Leer de memoria externa

```
public Resultado leerExterna(String fichero, String codigo){
    File miFichero, tarjeta;

    //tarjeta = Environment.getExternalStoragePublicDirectory("datos/programas/");
     //tarjeta.mkdirs();
    tarjeta = Environment.getExternalStorageDirectory();
    miFichero = new File(tarjeta.getAbsolutePath(), fichero);

    return leer(miFichero, codigo);
}
```

Acceso a un fichero de los recursos

- También existe la posibilidad de almacenar ficheros en el paquete de la aplicación

- Estos ficheros no pueden ser modificados.

- Por ejemplo, si se guarda el fichero *datos.txt* en la carpeta */res/raw*, se puede acceder a él usando:

getResources.openRawResource(R.raw.datos)

```
InputStream is = null;
StringBuilder miCadena = new StringBuilder();
int n;

try {
      //is = getApplicationContext().getResources().openRawResource(R.raw.datos);
    is = getResources().openRawResource(R.raw.datos);
      while ((n = is.read()) != -1) {
            miCadena.append((char) n);
      }
} catch . . .
```

Acceso a un fichero de los recursos

```
public Resultado leerRaw(String fichero){
     //fichero tendrá el nombre del fichero raw sin la extensión
    InputStream is = null;
    StringBuilder miCadena = new StringBuilder();
    int n;
    Resultado resultado = new Resultado();

    resultado.setCodigo(true);
    try {
        //is = contexto.getResources().openRawResource(R.raw.numero);
        is = contexto.getResources().openRawResource(
                    contexto.getResources().getIdentifier(fichero,"raw", contexto.getPackageName()));
        while ((n = is.read()) != -1) {
            miCadena.append((char) n);
        }
    } catch (Exception e) {
        Log.e("Error", e.getMessage());
        resultado.setCodigo(false);
        resultado.setMensaje(e.getMessage());
    } finally {
        try {
            if (is != null) {
                is.close();
                resultado.setContenido(miCadena.toString());
            }
        } catch (Exception e) {
            Log.e("Error al cerrar", e.getMessage());
            resultado.setCodigo(false);
            resultado.setMensaje(e.getMessage());
        }
    }
   return resultado;
}
```

Acceso a un fichero en /assets

Android ofrece más de un directorio en el que guardar ficheros que se incluirán en el paquete. La carpeta /assets está en el mismo nivel que el directorio /res, lo que significa que no es parte de los subdirectorios del mismo.

A los archivos colocados en el directorio /assets no se les generan IDs en R.java. Somos nosotros los que debemos especificar la ruta para leerlo.

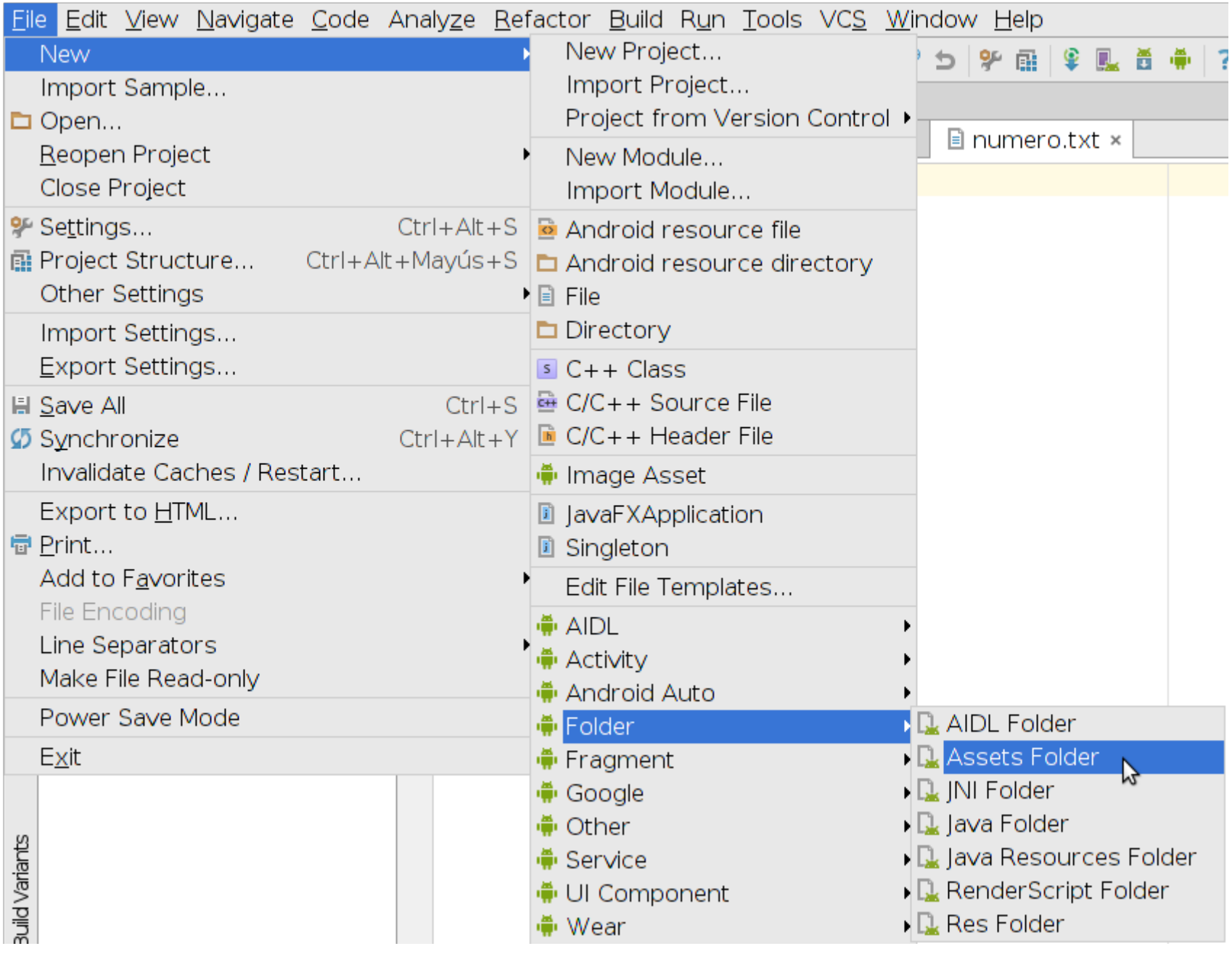
La ruta al fichero es una ruta relativa que comienza con /assets. Usaremos la clase AssetManager para acceder a estos ficheros:

Acceso a un fichero en /assets

```
public String leerAsset(String fichero){
    AssetManager am = contexto.getAssets();
    InputStream is = null;
    StringBuilder miCadena = new StringBuilder();
    int n;
    boolean correcto = false;

    try {
        is = am.open(fichero);
        while ((n = is.read()) != -1) {
            miCadena.append((char) n);
        }
    } catch . . .
```

Acceso a un fichero en /assets

Creación de la carpeta assets:



Acceso a un fichero en /assets

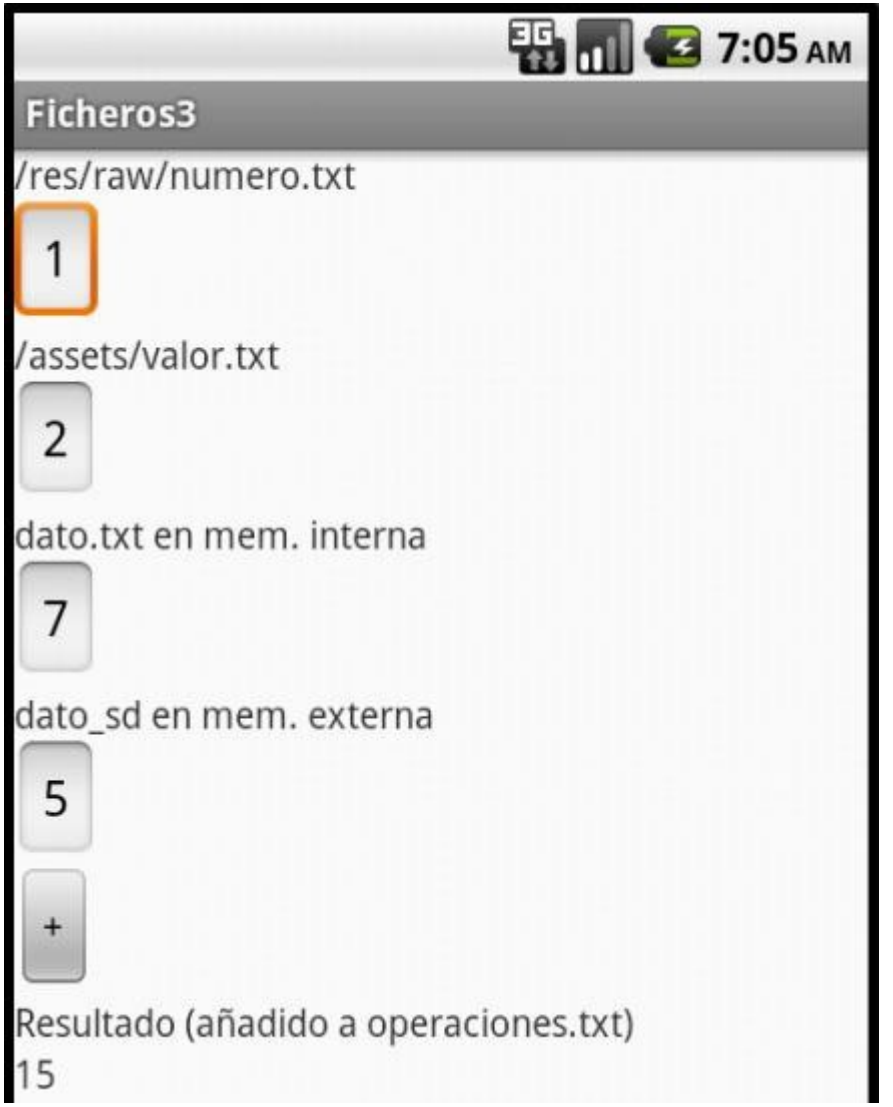
```
public Resultado leerAsset(String fichero){
    AssetManager am = contexto.getAssets();
    InputStream is = null;
    StringBuilder miCadena = new StringBuilder();
    int n;
    Resultado resultado = new Resultado();

    resultado.setCodigo(true);
    try {
        is = am.open(fichero);
        while ((n = is.read()) != -1) {
            miCadena.append((char) n);
        }
    } catch (IOException e) {
        Log.e("Error", e.getMessage());
        resultado.setCodigo(false);
        resultado.setMensaje(e.getMessage());
    } finally {
        try {
            if (is != null) {
                is.close();
                resultado.setContenido(miCadena.toString());
            }
        } catch (Exception e) {
            Log.e("Error al cerrar", e.getMessage());
            resultado.setCodigo(false);
            resultado.setMensaje(e.getMessage());
        }
    }
   return resultado;
}
```

Ejercicio: Leer ficheros

Crear una aplicación que sume 4 números. Inicialmente los datos se leerán de 4 ficheros diferentes:

- fichero *numero.txt* en /res/raw
- fichero *valor.txt* en /assets
- fichero *dato.txt* en memoria interna
- fichero *dato_sd.txt* en memoria externa

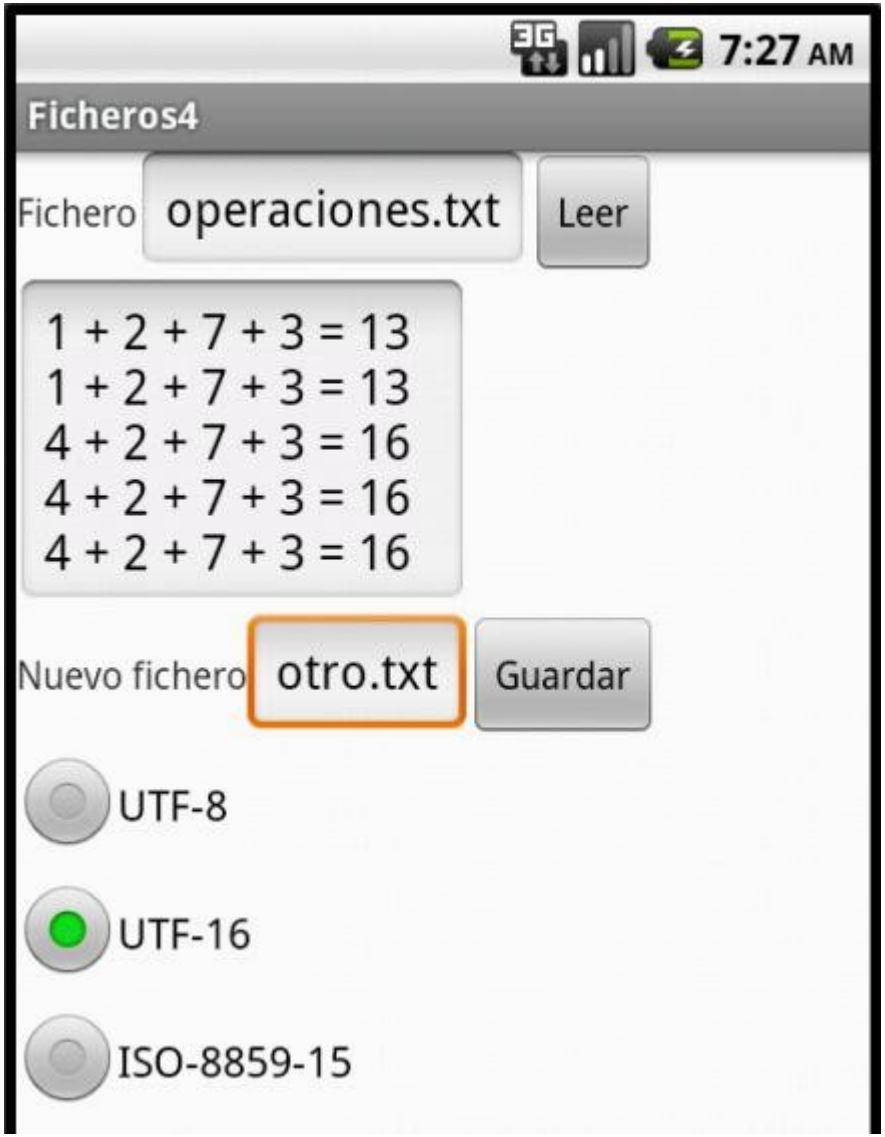
Además, la operación realizada se añadirá al fichero *operaciones.txt* en la memoria externa.



Ejercicio: Codificación

Crear una aplicación que pida un nombre de fichero, el cual existirá en la memoria externa, y muestre su contenido en pantalla según la codificación elegida: UTF-8, UTF-16 o ISO-8859-15.

También permitirá guardarlo en la tarjeta de memoria con otro nombre y una codificación diferente:
UTF-8, UTF-16 o ISO-8859-15.

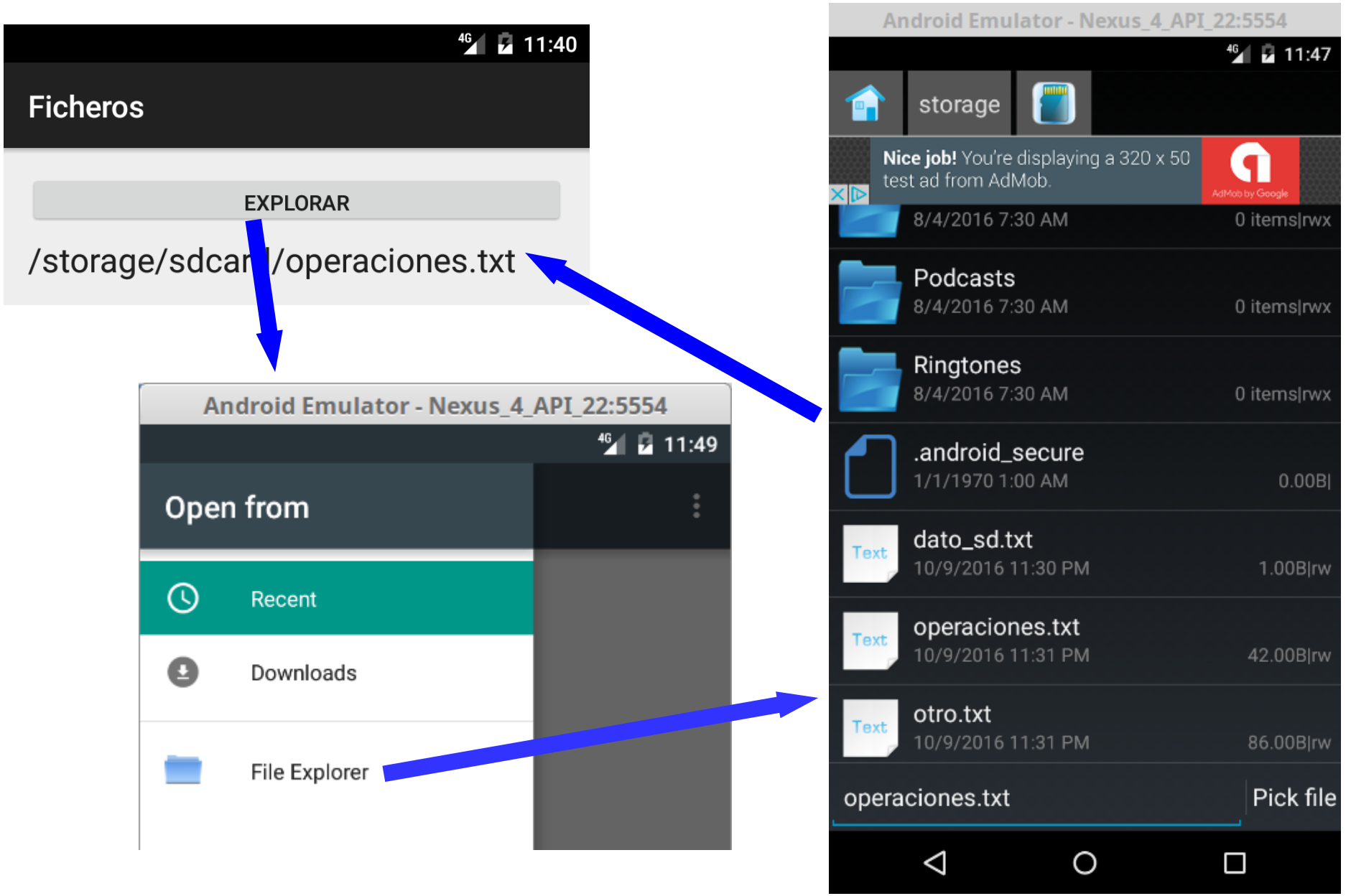


Usar un explorador de archivos

Se puede abrir cualquier fichero que tengamos en nuestro dispositivo Android usando un administrador de archivos instalado, como por ejemplo ASTRO o ES File Explorer.

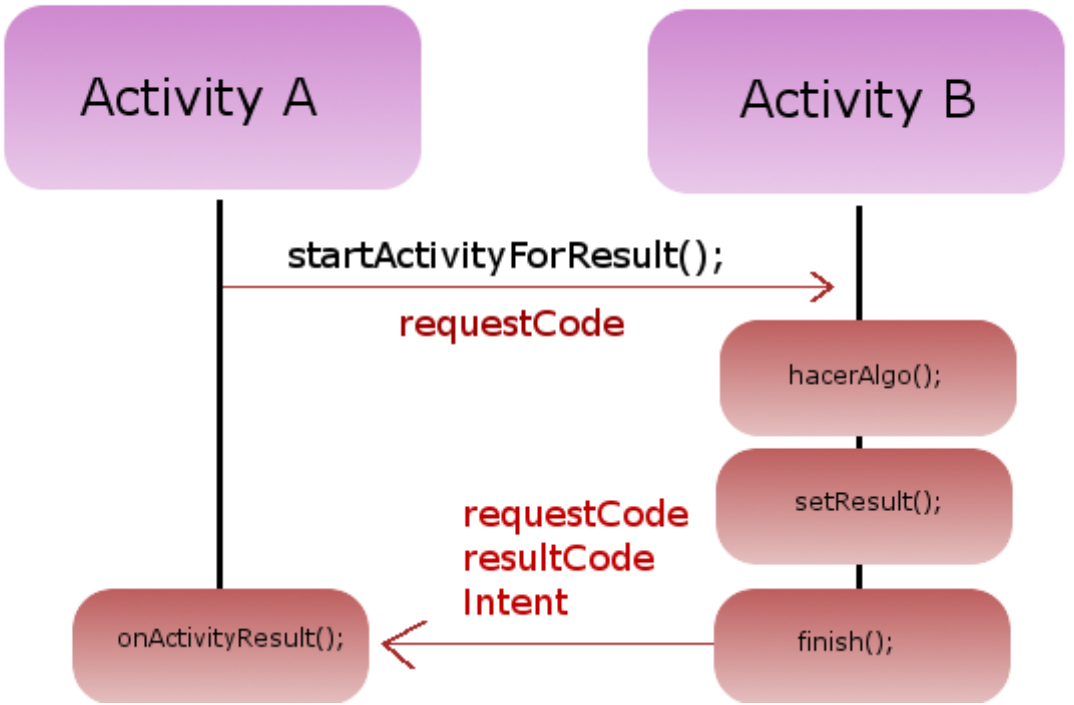
Para instalar algún administrador de archivos en el emulador usaremos aptoide.

Usar un explorador de archivos



Usar un explorador de archivos

Lanzar una actividad y obtener un resultado



Más información:

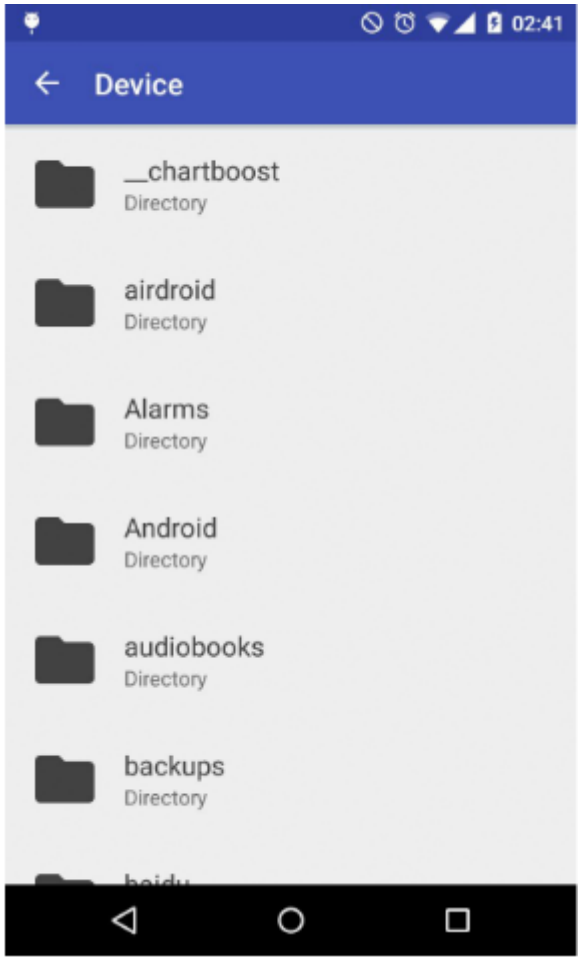
- Obtener resultados de otra Actividad
- Intents y navegación entre actividades

Usar un explorador de archivos

MainActivity.java

```
public class MainActivity extends AppCompatActivity implements OnClickListener {
    private static final int ABRIRFICHERO_REQUEST_CODE = 1;
    private Button botonAbrir;
    private TextView txtInfo;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        botonAbrir = (Button) findViewById(R.id.botonAbrir);
        txtInfo = (TextView) findViewById(R.id.txtInfo);
        botonAbrir.setOnClickListener(this);
    }
    @Override
    public void onClick (View v) {
        Intent intent = new Intent(Intent.ACTION_GET_CONTENT);
        intent.setType("file/*");
        if (intent.resolveActivity(getPackageManager()) != null)
                startActivityForResult(intent, ABRIRFICHERO_REQUEST_CODE);
        else
                //informar que no hay ninguna aplicación para manejar ficheros
                Toast.makeText(this, "No hay aplicación para manejar ficheros", Toast.LENGTH_SHORT).show();
    }
    @Override
    protected void onActivityResult (int requestCode, int resultCode, Intent data) {
          if (requestCode == ABRIRFICHERO_REQUEST_CODE)
              if (resultCode == RESULT_OK) {
                 // Mostramos en la etiqueta la ruta del archivo seleccionado
                 String ruta = data.getData().getPath();
                 txtInfo.setText(ruta);
              }
                  else
                  Toast.makeText(this, "Error: " + resultCode, Toast.LENGTH_SHORT).show();
    }
}
```

Material File Picker



Ejercicio propuesto

Se desea leer un fichero grande (de 1 Mb. aproximadamente) en memoria externa y escribir su contenido en otro fichero.

¿Hay mucha diferencia (en tiempo de ejecución de la aplicación) si se accede a los ficheros byte a byte o si se accede usando un buffer?

¿Dudas?

¿Sugerencias?

paco@portadaalta.es