GESTION DE CONTENIDOS EN APLICACIONES MOVILES

Las aplicaciones descritas hasta el momento representaban la información a procesar en forma de variables. El problema de estas variables es que dejan de existir en el momento en que la aplicación es destruida. En muchas ocasiones vamos a necesitar almacenar información de manera permanente. Las alternativas más habituales para conservar esta información son los ficheros, las bases de datos o servicios a través de la red. Estas técnicas no solo permiten mantener a buen recaudo los datos de la aplicación, sino que también vamos a poder compartir estos datos con otras aplicaciones y usuarios. De forma adicional, el sistema Android pone a nuestra disposicióndos nuevos mecanismos para almacenar datos, las preferencias y ContentProvider.

Alternativas para guardar datos permanentemente en Android

Existen muchas alternativas para almacenar información de forma permanente en un sistema informático. A continuación mostramos una lista de las más habituales utilizadas en Android:

- **Preferencias:** Es un mecanismo liviano que permite almacenar y recuperar datos primitivos en la forma de pares clave/valor. Este mecanismo se suele utilizar para almacenar los parámetros de configuración de una aplicación.
- **Ficheros:** Puedes almacenar los ficheros en la memoria interna del dispositivo o en un medio de almacenamiento removible como una tarjeta SD. También puedes utilizar ficheros añadidos a tu aplicación como recursos.
- **XML:** Se trata de un estándar fundamental para la representación de datos, en Internet y en muchos otros entornos (como en el Android SDK). En Android disponemos de las librerías SAX y DOM para manipular datos en XML.
- **Base 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 (*content providers*):** Un proveedor de contenidos es un componente opcional de una aplicación que expone el acceso de lectura / escritura de sus datos a otras aplicaciones. Está sujeto a las restricciones de seguridad que se quieran imponer. Los proveedores de contenido implementan una sintaxis estándar para acceder a sus datos mediante URI (Uniform Resource Identifiers) y un mecanismo de acceso para devolver los datos, similar a SQL. Android provee algunos proveedores de contenido para tipos de datos estándar, tales como contactos personales, ficheros multimedia, etc.
- **Internet:** No olvides que también puedes usar la nube para almacenar y recuperar datos.

Ficheros en Android

En Android también podremos manipular ficheros tradicionales de una forma muy similar a como se realiza en Java.

Lo primero que hay que tener en cuenta es ***dónde*** queremos almacenar los ficheros y el ***tipo de acceso*** que queremos tener a ellos. Así, podremos leer y escribir ficheros localizados en:

1. La **memoria interna** del dispositivo.
2. La **tarjeta SD** externa, si existe.
3. La propia aplicación, en forma de **recurso**, son de solo lectura.

Memoria Interna

Veamos en primer lugar cómo trabajar con la memoria interna del dispositivo. Cuando almacenamos ficheros en la memoria interna debemos tener en cuenta las **limitaciones de espacio** que tienen muchos dispositivos, por lo que no deberíamos abusar de este espacio utilizando ficheros de gran tamaño. Por defecto, los ficheros almacenados aquí solo son accesibles para la propia aplicación que los creó, no pueden ser leídos por otras aplicaciones, ni siquiera por el usuario del teléfono. Cada aplicación dispone de una carpeta especial para almacenar ficheros (data/data/nombre_paquete/files). La ventaja de utilizar esta carpeta es que cuando se desinstala la aplicación los ficheros que se han creado se eliminarán.

Recordad que el sistema de ficheros se sustenta en la capa Linux, por lo que Android hereda su estructura. Cuando se instala una aplicación, Android crea un nuevo usuario Linux asociado a la aplicación y es este usuario el que podrá o no acceder a los ficheros.

Para trabajar con ficheros, puedes utilizar cualquier rutina del paquete `java.io`. Adicionalmente se han creado métodos adicionales asociados a la clase `Context` para facilitar el trabajo con ficheros almacenados en la memoria interna.

Escribir ficheros en la memoria interna es muy sencillo. Android proporciona para ello el método `openFileOutput()`, que recibe como parámetros el nombre del fichero (no puede contener subdirectorios, el fichero siempre se almacena en la carpeta reservada para la aplicación) y el modo de acceso con el que queremos abrir el fichero. Este modo de acceso puede variar entre `MODE_PRIVATE` para acceso privado desde nuestra aplicación (crea el fichero o lo sobrescribe si ya existe), `MODE_APPEND` para añadir datos a un fichero ya existente, `MODE_WORLD_READABLE` para permitir a otras aplicaciones leer el fichero, o `MODE_WORLD_WRITABLE` para permitir a otras aplicaciones escribir sobre el fichero. Los dos últimos no deberían utilizarse dada su peligrosidad, de hecho, han sido declarados como obsoletos (*deprecated*) en la API 17.

Este método devuelve una referencia al *stream* de salida asociado al fichero (en forma de objeto `FileOutputStream`), a partir del cual ya podremos utilizar los métodos de manipulación de ficheros tradicionales del lenguaje java (api `java.io`). Como ejemplo, convertiremos este *stream* a un `OutputStreamWriter` para escribir una cadena de texto al fichero.

```
try{
    OutputStreamWriter osw=
            new OutputStreamWriter(openFileOutput("prueba_int.txt",
                    Context.MODE_PRIVATE));
    osw.write("Escribiendo un texto de prueba");
    osw.close();
}
```

```
catch (Exception e) {
    Log.e ("Ficheros", "ERROR!! al escribir fichero en memoria interna");
}
```

Ya hemos creado un fichero de texto en la memoria interna, ¿pero dónde se almacena exactamente? Android almacena por defecto los ficheros creados en una ruta determinada, (la carpeta reservada para la aplicación) que en este caso seguirá el siguiente patrón:

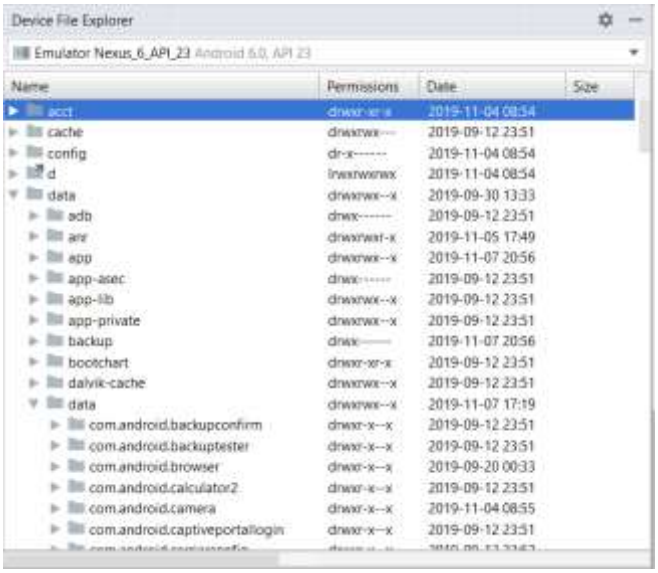
`/data/data/paquete.java/files/nombre_fichero`

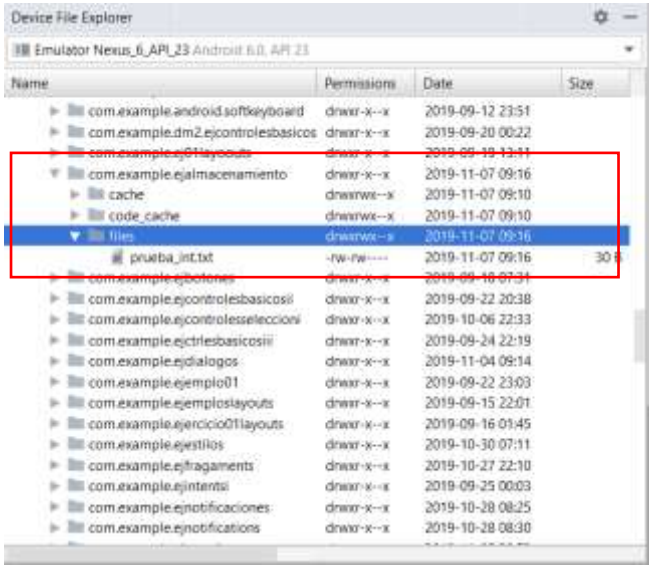
El fichero siempre se almacena en la carpeta reservada para la aplicación (`/data/data/paquete/files`). Recordad SIEMPRE cerrar los ficheros con el método `close()`.

En este caso particular, la ruta será

`/data/data/com.example.ejalmacenamiento/files/prueba_int.txt`

Si ejecutamos el código anterior lo podremos comprobar en el *Device File Explorer* cómo el fichero se crea correctamente en la ruta indicada. **View → Tools Windows → Device File Explorer.**





Por otra parte, leer ficheros desde la memoria interna es igual de sencillo, y procederemos de forma análoga, con la única diferencia de que utilizaremos el método `openFileInput()` para abrir el fichero, y los métodos de lectura de `java.io` para leer el contenido.

```
try{
    BufferedReader fin =
            new BufferedReader(
                    new InputStreamReader(
                            openFileInput("prueba_int.txt")));

    String texto = fin.readLine();
    fin.close();

    Log.i("Ficheros", "Fichero leido!");
    Log.i("Ficheros", "Texto: " + texto);
}
catch (Exception ex){
```

```
        Log.e("Ficheros", "Error al leer fichero desde memoria interna");
    }
```

Es muy **importante** hacer un manejo cuidadoso de los errores. De hecho, el acceso a ficheros ha de realizarse de forma obligatoria dentro de una sección `try/catch`.

En forma de recurso

La otra forma de almacenar ficheros en la memoria interna del dispositivo es incluirlos como *recurso* en la propia aplicación. Aunque este método es útil en muchos casos, sólo debemos utilizarlo cuando no necesitemos realizar modificaciones sobre los ficheros, ya que tendremos limitado el acceso a **sólo lectura**.

Hay dos alternativas para esto: usar la carpete `res/raw` o `assets`. La principal diferencia a la hora de utilizar una u otra carpeta está en la forma de identificar el fichero. Por ejemplo, si incluimos un fichero llamado `datos.txt` en la carpeta `res/raw`, para acceder a él utilizaremos `context.getResources.openRawResource(R.raw.datos)`. Y si por el contrario, dejamos el fichero en la carpeta `assets`, utilizaremos `context.getAssets().open("datos.txt")` para acceder a él. Otra diferencia es que dentro de la carpeta `assets` se podrán crear subcarpetas para organizar los ficheros.

Recordad, que tanto en la carpeta `raw` con en `assets`, los ficheros **nunca son comprimidos**.

Para poder incluir un fichero como recurso de la aplicación en la carpeta `/res/raw`, que no está incluida por defecto, tendremos que crearla manualmente. Para ello, pincharemos con el botón derecho sobre el directorio/carpeta /res y le diremos: **New → Android resource Directory**

Una vez creada la carpeta `raw` podremos colocar en ella cualquier fichero que queramos que se incluya con la aplicación en tiempo de compilación en forma de recurso. Como ejemplo, incluiremos un fichero de texto llamado "***prueba_raw.txt***". Ya en tiempo de ejecución podremos acceder a este fichero, sólo en modo de lectura, de una forma similar a la que ya hemos visto para el resto de ficheros en memoria interna.

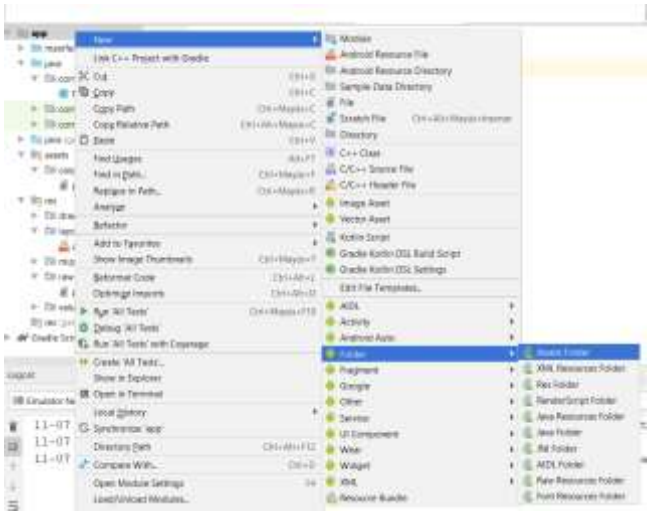
Para acceder al fichero, accederemos en primer lugar a los recursos de la aplicación con el método `getResources()` y sobre éstos utilizaremos el método `openRawResource(`*`id_del_recurso`*`)` para abrir el fichero en modo lectura. Este método devuelve un objeto `InputStream`, que ya podremos manipular como queramos mediante los métodos de la API `java.io`. Como ejemplo, **nosotros convertiremos el stream en un objeto `BufferedReader`** para leer el texto contenido en el fichero de ejemplo (por supuesto los ficheros de recurso también pueden ser binarios, como por ejemplo ficheros de imagen, video, etc). El código quedaría:

```
try {
    InputStream fraw = getResources().openRawResource(R.raw.prueba_raw);
    BufferedReader brin = new BufferedReader( new InputStreamReader(fraw));
    String linea= brin.readLine();
    while (linea!=null){
        Log.i("Ficheros", linea);
        linea=brin.readLine();
    }
    fraw.close();
}
```

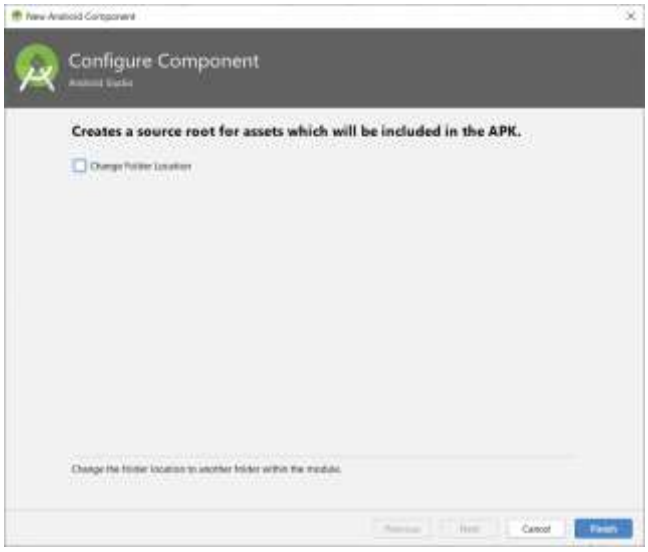
```
catch (Exception ex) {
    Log.e ("Ficheros", "Error al leer fichero desde recurso raw");
}
```

Como podemos ver en el código anterior, al método `openRawResource()` le pasamos como parámetro el ID del fichero incluido como recurso, que seguirá el patrón "**`R.raw.`***`nombre_del_fichero`*", por lo que en nuestro caso particular será `R.raw.prueba_raw`.

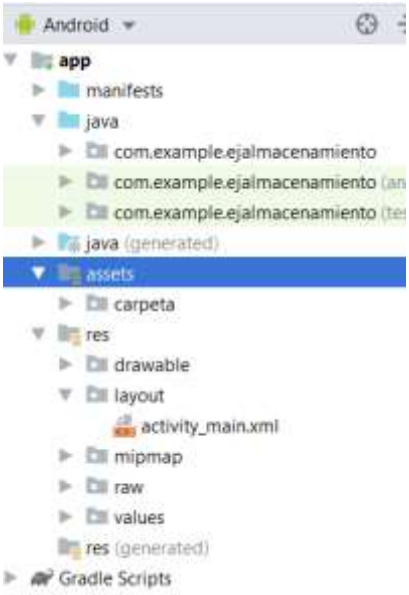
Para poder incluir un fichero como recurso de la aplicación en la carpeta `assets`, tendremos que crearla manualmente. Para ello, nos situaremeos en la carpeta `app` y pulsaremos el botón derecho del ratón y le diremos: **New → Folder → Assets Folder**



En la siguiente pantalla dejamos los valores por defecto:



Ahora ya tenemos creada la carpeta `assets`. A continuación vamos a crear la subcarpeta "carpeta" dentro de `assets`. Pulsamos botón derecho sobre `assets` y seleccionamos **New → Directory** e introducimos el nombre *carpeta*.



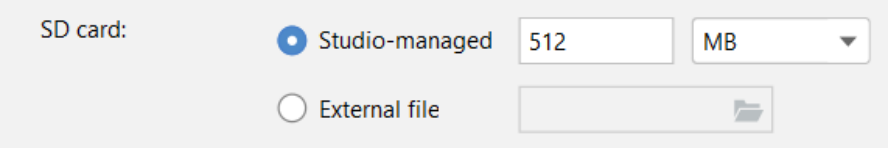
Una vez que tenemos la estructura de carpetas creada, incluimos en su interior un fichero de texto "**prueba_assets.txt**" y ya podemos en tiempo de ejecución acceder a dicho fichero para lectura.

```
try{
    InputStream fraw =
                getResources().getAssets().open("carpeta/prueba_assets.txt");
    BufferedReader brin = new BufferedReader(new InputStreamReader(fraw));
    String linea = brin.readLine();
    while (linea != null) {
        Log.i ("Ficheros", linea);
        linea = brin.readLine();
    }
    fraw.close();
}
catch (Exception ex){
    Log.e("Ficheros", "Error al leer fichero de recursos en Assets");
}
```

Memoria externa SD

La memoria interna suele ser relativamente limitada y no es aconsejable almacenar en ella ficheros de gran tamaño. La alternativa natural es utilizar para ello la memoria externa del dispositivo, constituida normalmente por una tarjeta de **memoria SD**, aunque en dispositivos recientes también está presente en forma de almacenamiento no extraíble del dispositivo, aunque no por ello debe confundirse con la memoria interna. A diferencia de la memoria interna, el almacenamiento externo es público, es decir, todo lo que escribamos en él **podrá ser leído por otras aplicaciones y por el usuario**, por tanto hay que tener cierto cuidado a la hora de decidir lo que escribimos en memoria interna y externa.

Para poder probar aplicaciones que hagan uso de la memoria externa en el emulador de Android necesitamos que tenga establecido correctamente el tamaño de la tarjeta SD. En el caso de ejemplo, se ha definido un tamaño de tarjeta de 512 Mb:



A diferencia de la memoria interna, la tarjeta de memoria **no tiene por qué estar presente en el dispositivo**, e incluso estándolo **puede no estar reconocida** por el sistema. Por tanto, el primer paso recomendado a la hora de trabajar con ficheros en memoria externa es asegurarnos de que dicha memoria está presente y disponible para leer y/o escribir en ella.

Para esto la API de Android proporciona (como método estático de la clase `Environment`) el método `getExternalStorageStatus()`, que nos dice si la memoria externa está **disponible y si se puede leer y escribir en ella**. Este método devuelve una serie de valores que nos indicarán el estado de la memoria externa, siendo los más importantes los siguientes:

- `MEDIA_MOUNTED`, que indica que la memoria externa está disponible y podemos tanto leer como escribir en ella.
- `MEDIA_MOUNTED_READ_ONLY`, que indica que la memoria externa está disponible pero sólo podemos leer de ella.
- Otra serie de valores que indicarán que existe algún problema y que por tanto no podemos ni leer ni escribir en la memoria externa (`MEDIA_UNMOUNTED`, `MEDIA_REMOVED`, `MEDIA_EJECTING`…). Para consultar todos estos estados visitar la documentación oficial de la clase Environment.

Teniendo todo esto en cuenta, podemos realizar un chequeo previo del estado de la memoria externa del dispositivo de la siguiente manera:

```
boolean sdDisponible = false;
boolean sdAccesoEscritura= false;

//Comprobamos el estado de la memoria exterena
String estado= Environment.getExternalStorageState();
Log.i("Memoria", estado);

if (estado.equals(Environment.MEDIA_MOUNTED)) {
    sdDisponible = true;
    sdAccesoEscritura = true;
}
else if (estado.equals(Environment.MEDIA_MOUNTED_READ_ONLY)) {
    sdDisponible = true;
    sdAccesoEscritura = false;
}
else {
    sdDisponible = false;
    sdAccesoEscritura = false;
}
```

Una vez chequeado el estado de la memoria externa, y dependiendo del resultado obtenido, ya podremos leer o escribir en ella cualquier tipo de fichero.

Empezaremos por la escritura. Para escribir un fichero en la memoria externa tenemos que obtener en primer lugar la ruta al *directorio raíz* de esta memoria. Para ello podemos utilizar el método `getExternalStorageDirectory()` de la clase `Environment`, que nos devolverá **un**

objeto `File` con la ruta de dicho directorio. A partir de este objeto, podremos construir otro con el nombre elegido para nuestro fichero (por ejemplo "*prueba_sd.txt*"), creando un nuevo objeto `File` que combine ambos elementos. Tras esto, ya sólo queda encapsularlo en algún objeto de escritura de ficheros de la API de java y escribir algún dato de prueba. En nuestro ejemplo lo convertiremos una vez más a un objeto `OutputStreamWriter` para escribir en el fichero un mensaje de texto. El código quedaría:

```
try {
    File ruta_sd = Environment.getExternalStorageDirectory();

    File f = new File (ruta_sd.getAbsolutePath(),"prueba_sd.txt");

    OutputStreamWriter osw =
                    new OutputStreamWriter(new FileOutputStream(f));

    osw.write("Primera linea en fichero de prueba en memoria externa.\n");
    osw.close();
    Log.i ("Ficheros", "fichero escrito correctamente");
}

catch (Exception ex) {
    Log.e ("Ficheros", "Error al escribir fichero en tarjeta SD");
}
```

El código anterior funciona sin problemas pero escribirá el fichero directamente en la carpeta raíz de la memoria externa. Esto, aunque en ocasiones puede resultar necesario, no es una buena práctica. Lo correcto sería disponer de una carpeta propia para nuestra aplicación, lo que además tendrá la ventaja de que al desinstalar la aplicación también se liberará este espacio. Esto lo conseguimos utilizando el método `getExternalFilesDir(null)` en vez de `getExternalStorageDirectory()`. El método `getExternalFilesDir()` devuelve directamente la ruta de una carpeta específica para nuestra aplicación dentro de la memoria externa siguiendo el siguiente patrón:

```
<raíz_mem_ext>/Android/data/nuestro.paquete.java/files
```

En el caso del ejemplo, si utilizamos

```
File ruta_sd = getApplicationContext().getExternalFilesDir(null);
```

la ruta de la carpeta donde escribir el fichero será:

```
/storage/emulated/0/Android/data/com.example.ejalmacenamiento/files
```

Si en lugar de null le indicamos como parámetro un tipo de datos determinado (`DIRECTORY_MUSIC`, `DIRECTORY_PICTURES`, `DIRECTORY_MOVIES`, `DIRECTORY_RINGTONES`, `DIRECTORY_ALARMS`, `DIRECTORY_NOTIFICATIONS`, `DIRECTORY_PODCASTS`) nos devolverá una subcarpeta dentro de la anterior con su nombre correspondiente. Así, por ejemplo, una llamada al método `getExternalFilesDir(Environment.DIRECTORY_MUSIC)` nos devolvería la siguiente carpeta:

```
<raíz_mem_ext>/Android/data/nuestro.paquete.java/files/Music
```

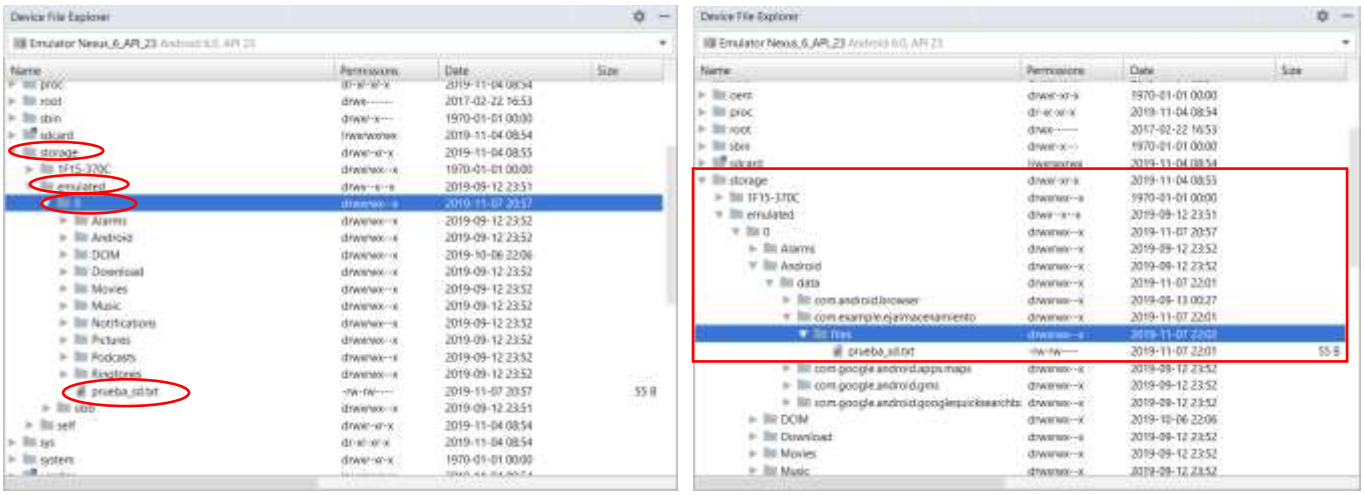
Esto último, además, ayuda a Android a saber qué tipo de contenidos hay en cada carpeta, de forma que puedan clasificarse correctamente por ejemplo en la galería multimedia.

Sea como sea, para tener acceso a la memoria externa tendremos que especificar en el fichero *AndroidManifest.xml* que nuestra aplicación necesita permiso de escritura en dicha memoria. Para añadir un nuevo permiso usaremos la cláusula `<uses-permission>` como siempre, utilizando el valor concreto "`android.permission.WRITE_EXTERNAL_STORAGE`". Con esto, nuestro fichero *AndroidManifest.xml* quedaría así:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.ejalmacenamiento">

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Si ejecutamos el código y vamos al explorador de archivos *Device File Explorer* podremos comprobar cómo se ha creado correctamente el fichero en el *directorio raiz* de nuestra SD. Esta ruta puede variar entre dispositivos, pero para Android 2.x suele localizarse en la carpeta `/sd-card/`, mientras que para Android 4 suele estar en `/mnt/sdcard/` y en nuestro caso lo podemos ver en `/sdcard` o `/mnt/sdcard` o `/storage/emulated/0/`

Y si se ha utilizado la opción `getExternalFilesDir(null);`, podemos ver como se ha creado el fichero en una carpeta cuya ruta incluye el nombre del paquete de la aplicación.



Por su parte, leer un fichero desde la memoria externa es igual de sencillo. Obtenemos el directorio raíz de la memoria externa con `getExternalStorageDirectory()`, o la carpeta específica de nuestra aplicación con `getExternalFilesDir()` como ya hemos visto. Creamos un objeto `File` que combine esa ruta con el nombre del fichero a leer y lo encapsulamos dentro de algún objeto que facilite la lectura, nosotros para leer texto utilizaremos como siempre un `BufferedReader`.

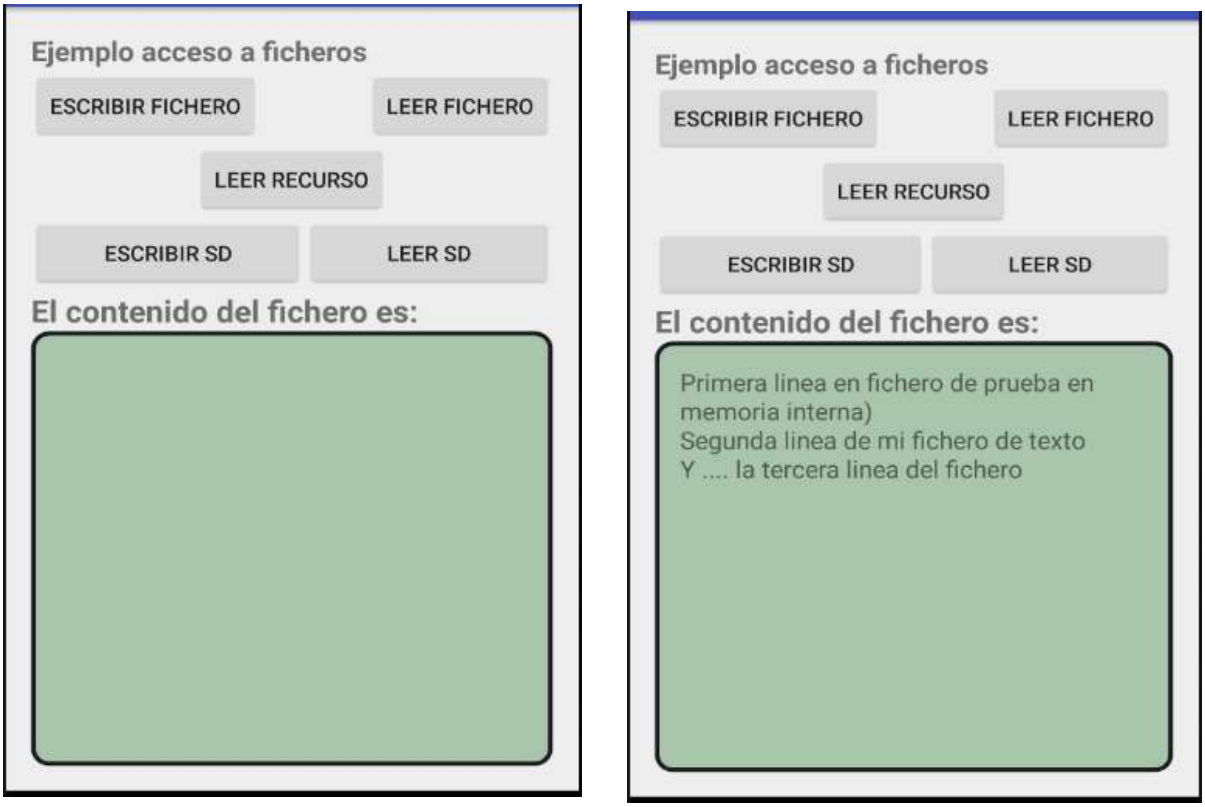
```
try {
    File ruta_sd = Environment.getExternalStorageDirectory();

    File f= new File(ruta_sd.getAbsolutePath(), "prueba_sd.txt");
    BufferedReader br = new BufferedReader(
                        new InputStreamReader(new FileInputStream(f)));

    String linea= br.readLine();
    br.close();

    Log.i("Ficheros", linea);

}catch (Exception ex){
    Log.e("Ficheros", "ERROR!! en la lectura del fichero en SD");
}
```

Como vemos, el código es análogo al que hemos visto para la escritura de ficheros.

Ejemplo 01: Ficheros

El ejemplo completo quedaría:

Este podría ser el layout de la aplicación, que cuando se pulsa sobre alguno de los botones de leer, se leerá del lugar que corresponda (Memoria interna, Recurso o Memoria externa) y se visualizaría el contenido del fichero en el `TextView` de la parte de abajo.



activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:gravity="end">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/et_Presentacion"
        android:id="@+id/etPresentacion"
        android:textSize="18dp"
        android:textStyle="bold"/>

    <Button
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:id="@+id/BtEscribirFichero"
        android:text="@string/btn_w_fichero"
        android:layout_below="@id/etPresentacion"
        android:onClick="escribirFichero"/>

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/BtLeerFichero"
        android:text="@string/btn_r_fichero"
        android:layout_below="@+id/etPresentacion"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true"
        android:onClick="leerFichero"/>

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/BtLeerRecurso"
        android:text="@string/btn_r_Recurso"
        android:layout_below="@+id/BtEscribirFichero"
        android:layout_centerHorizontal="true"
        android:onClick="leerRecurso"/>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_below="@id/BtLeerRecurso"
        android:id="@+id/lLMemoriaSD">

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/BtEscribirMemoriaSD"
            android:text="@string/btn_w_memoriaSD"
            android:layout_weight="1"/>

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/BtLeerMemoriaSD"
            android:text="@string/btn_r_memoriaSD"
            android:layout_weight="1"/>

    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:layout_below="@id/lLMemoriaSD">
```

```
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/LblCbecera"
            android:text="@string/lbl_cabecera"
            android:textSize="20dp"
            android:textStyle="bold"/>

        <TextView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:id="@+id/LblContenidoFichero"
            android:background="@drawable/rounded_corner"
            android:textSize="16dp"/>
    </LinearLayout>

</RelativeLayout>
```

/res/drawable/rounded_corner.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Para poner borde -->
<shape
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">

    <!-- view background color -->
    <solid
        android:color="#a9c5ac" >
    </solid>

    <!-- view border color and width -->
    <stroke
        android:width="3dp"
        android:color="#1c1b20" >
    </stroke>

    <!-- If you want to add some padding -->
    <padding
        android:left="16dp"
        android:top="16dp"
        android:right="16dp"
        android:bottom="16dp"     >
    </padding>

    <!-- Here is the corner radius -->
    <corners
        android:radius="10dp"    >
    </corners>

</shape>
```

String.xml

```
<resources>
    <string name="app_name">EJ Ficheros Completo</string>
    <string name="et_Presentacion">Ejemplo acceso a ficheros</string>
    <string name="btn_w_fichero">Escribir Fichero</string>
    <string name="btn_r_fichero">Leer Fichero</string>
    <string name="btn_r_Recurso">Leer Recurso</string>
    <string name="btn_w_memoriaSD">Escribir SD</string>
    <string name="btn_r_memoriaSD">Leer SD</string>
    <string name="lbl_cabecera">El contenido del fichero es:</string>
</resources>
```

MainActivity.java

```
package com.example.ejficheroscompleto;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;

import android.Manifest;
import android.app.Activity;
import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.os.Environment;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;

public class MainActivity extends AppCompatActivity {

    private static final int SOLICITUD_PERMISO_WRITE_SD = 0;
    private static final int SOLICITUD_PERMISO_READ_SD = 1;

    private TextView tvContenidoFichero;
    private Button btnEscribirMemSD;
    private Button btnLeerMemSD;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        tvContenidoFichero = findViewById(R.id.LblContenidoFichero);
        btnEscribirMemSD = findViewById(R.id.BtEscribirMemoriaSD);
        btnEscribirMemSD.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (comprobarPermisos()){
                    Toast.makeText(MainActivity.this,
                            "Tenemos permisos para escribir",
                             Toast.LENGTH_SHORT).show();
                    if (sdDisponible()) {
                        escribirEnSD();
                    }
                    else
                        Toast.makeText(MainActivity.this,
                                "Tarjeta NO lista para poder escribir",
                                 Toast.LENGTH_SHORT).show();
                }
                else {
                    solicitarPermiso(
                            Manifest.permission.WRITE_EXTERNAL_STORAGE,
                            "Sin este permiso no se puede ESCRIBIR en el
                             dispositivo externo",
                            SOLICITUD_PERMISO_WRITE_SD, MainActivity.this);
                }
            }
        });

        btnLeerMemSD = findViewById(R.id.BtLeerMemoriaSD);
        btnLeerMemSD.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (comprobarPermisos()){
                    Toast.makeText(MainActivity.this,
                           "Tenemos permisos para leer",
                            Toast.LENGTH_SHORT).show();
                    if (sdDisponible()) {
                        leerDeSD();
                    }
                    else
                        Toast.makeText(MainActivity.this,
                               "Tarjeta NO lista para poder leer",
                                Toast.LENGTH_SHORT).show();
                }
                else {
                    solicitarPermiso(
                            Manifest.permission.WRITE_EXTERNAL_STORAGE,
                            "Sin este permiso no se puede LEER en el
                             dispositivo externo",
                             SOLICITUD_PERMISO_WRITE_SD, MainActivity.this);
                }
            }
        });
```

```
}

public void escribirFichero(View view) {

    try{
        OutputStreamWriter osw=
                new OutputStreamWriter(openFileOutput("prueba_int.txt",
                        Context.MODE_PRIVATE));
        osw.write("Primera linea en fichero de prueba en
                  memoria interna)\n");
        osw.write("Segunda linea de mi fichero de texto\n");
        osw.write("Y .... la tercera linea del fichero\n");

        Log.i ("Ficheros", "Escribiendo en fichero de memoria interna");

        osw.close();
    }
    catch (Exception e) {
        Log.e ("Ficheros",
                "ERROR!! al escribir fichero a memoria interna");
    }
}

public void leerFichero (View view){

    try
    {
        BufferedReader fin =
                new BufferedReader(
                        new InputStreamReader(
                                openFileInput("prueba_int.txt")));
        String texto="";
        String linea= fin.readLine();
        while (linea!=null){
            texto=texto+linea+"\n";
            Log.i("Ficheros", linea);
            linea=fin.readLine();
        }

        fin.close();
        tvContenidoFichero.setText(texto);
    }
    catch (Exception ex)
    {
        Log.e("Ficheros", "Error al leer fichero desde memoria interna");
    }
}

public void leerRecurso(View view){

    try {
        InputStream fraw =
                   getResources().openRawResource(R.raw.prueba_raw);
        BufferedReader brin =
                   new BufferedReader( new InputStreamReader(fraw));
```

```
            String texto="";
            String linea= brin.readLine();
            while (linea!=null){
                texto=texto+linea+"\n";
                Log.i("Ficheros", linea);
                linea=brin.readLine();
            }
            fraw.close();
            tvContenidoFichero.setText(texto);
        }
        catch (Exception ex) {
            Log.e ("Ficheros", "Error al leer fichero desde recurso raw");
        }
    }

    private boolean comprobarPermisos (){
        //Comprobamos que tenemos permiso para realizar la operación.
        if (ContextCompat.checkSelfPermission(this,
                Manifest.permission.WRITE_EXTERNAL_STORAGE) ==
                        PackageManager.PERMISSION_GRANTED) {
            return true;
        }
        else {
            return false;
        }
    }

    private boolean sdDisponible (){
        boolean sdDisponible = false;
        boolean sdAccesoEscritura= false;

        //Comprobamos el estado de la memoria externa
        String estado = Environment.getExternalStorageState();
        Log.i("Memoria", estado);

        if (estado.equals(Environment.MEDIA_MOUNTED)) {
            sdDisponible = true;
            sdAccesoEscritura = true;
        } else if (estado.equals(Environment.MEDIA_MOUNTED_READ_ONLY)) {
            sdDisponible = true;
            sdAccesoEscritura = false;
        } else {
            sdDisponible = false;
            sdAccesoEscritura = false;
        }
        if (sdDisponible)
            Toast.makeText(getApplicationContext(),
                           "Tengo Tarjeta SD",
                           Toast.LENGTH_SHORT).show();
        if (sdAccesoEscritura)
            Toast.makeText(getApplicationContext(),
                           "La tarjeta SD es escribible",
                            Toast.LENGTH_SHORT).show();

        if (sdDisponible &&sdAccesoEscritura)
```

```
            return true;
        else
            return false;
    }

    private void escribirEnSD (){
         try {
            File ruta_sd = Environment.getExternalStorageDirectory();

            File f = new File(ruta_sd.getAbsolutePath(), "prueba_sd.txt");

            OutputStreamWriter osw =
                        new OutputStreamWriter(new FileOutputStream(f));

            osw.write("Primera linea en fichero de prueba
                        en memoria externa (Tarjeta SD)\n");
            osw.write("Segunda linea de mi fichero de texto
                        escrito en SD \n");
            osw.write("Y esta es .... la tercera linea del fichero\n");
            osw.close();
            Log.i("Ficheros", "fichero escrito correctamente");
         } catch (Exception ex) {
            Log.e("Ficheros", "Error al escribir fichero en tarjeta SD");
         }
     }

    private void leerDeSD(){
         try {
            File ruta_sd = Environment.getExternalStorageDirectory();

            File f = new File(ruta_sd.getAbsolutePath(), "prueba_sd.txt");
            BufferedReader br = new BufferedReader(
                        new InputStreamReader(new FileInputStream(f)));

            String texto = "";
            String linea = br.readLine();
            while (linea != null) {
                texto = texto + linea + "\n";
                linea = br.readLine();
            }
            br.close();

            Log.i("Ficheros", texto);
            tvContenidoFichero.setText(texto);
         } catch (Exception ex) {
            Log.e("Ficheros", "ERROR!! en la lectura del fichero en SD");
         }
     }

    private static void solicitarPermiso (final String permiso,
                                          String justificacion,
                                          final int requestCode,
                                          final Activity actividad){

        if (ActivityCompat.shouldShowRequestPermissionRationale(
```

```
                                                        actividad, permiso)){
                //Informamos al usuario para qué y por qué
                //se necesitan los permisos
                new AlertDialog.Builder(actividad)
                        .setTitle("Solicitud de permiso")
                        .setMessage(justificacion)
                        .setPositiveButton("OK",
                                  new DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface dialog,
                                                int which) {
                                ActivityCompat.requestPermissions(actividad,
                                            new String[]{permiso},requestCode);
                            }
                        }).show();
            }
            else {
                //Muestra el cuadro de dialogo para la solicitud de permisos y
                //registra el permiso según respuesta del usuario
                ActivityCompat.requestPermissions(actividad,
                        new String[]{permiso}, requestCode);
            }
        }

        @Override
        public void onRequestPermissionsResult(int requestCode,
                                               @NonNull String[] permissions,
                                               @NonNull int[] grantResults) {

            if (requestCode == SOLICITUD_PERMISO_WRITE_SD){
                if (grantResults.length >= 1 &&
                        grantResults[0]==PackageManager.PERMISSION_GRANTED){

                    Log.e("AAAA", "Escribir Memoria SD");
                    sdDisponible();
                    escribirEnSD();
                }else {
                    Toast.makeText(this,
                           "No se puede ESCRIBIR en memoria SD",
                            Toast.LENGTH_LONG ).show();
                }
            }
            else if (requestCode == SOLICITUD_PERMISO_READ_SD){
                if (grantResults.length == 1 &&
                        grantResults[0]==PackageManager.PERMISSION_GRANTED){

                    leerDeSD();
                }else {
                    Toast.makeText(this,
                                "No se puede LEER de memoria SD",
                                Toast.LENGTH_LONG ).show();
                }
            }
        }
    }
```

AndroidManifest.xml

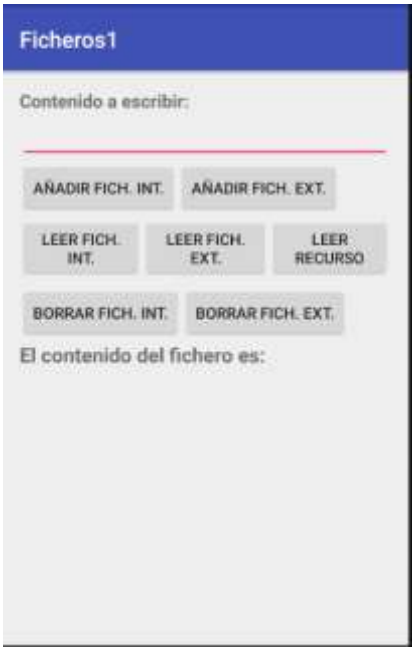
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.ejficheroscompleto">
    <uses-permission
        android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Ejercicios Ficheros

Ejercicio 1

Realiza un programa que:

- Añada contenidos de un `EditText` a un fichero interno.
- Borre un fichero interno.
- Visualice el contenido de un fichero interno en un `TextView`.
- Añada contenidos de un `EditText` a un fichero externo.
- Borre un fichero externo.
- Visualice el contenido de un fichero externo en un `TextView`.
- Visualice el contenido de un fichero de recursos.



Ejercicio 2

Realizar un programa que cargue un `spinner` con los nombres de provincias a partir de un fichero de recurso.

Ejercicio 3

Realizar un programa que cargue un `listview` Mis Webs Favoritas partir de un fichero de recurso, que va a contener la siguiente estructura:
`Nombre; enlace; logo; identificador;` por ejemplo:

Bing; http://www.bing.com; bing; 1;
Yahoo; http://www.yahoo.com; yahoo; 2;
Google; http://www.google.com; google; 3;
Bing-es; http://www.bing.es; bing; 4;
Yahoo-es; http://www.yahoo.es; yahoo; 5;