Almacenamiento externo (1)

Otra opción de almacenamiento es trabajar con archivos externos a la aplicación.

Existen dos tipos de archivos externos:

- Almacenamiento externo desmontable (SD, USB).
- Almacenamiento externo no desmontable (memoria física del teléfono).

Los archivos creados en el almacenamiento externo se abren en modo lectura tanto para el usuario como para el resto de aplicaciones, y el usuario podrá eliminarlos sin desinstalar la aplicación.

17

Almacenamiento externo (2)

- Comprobar disponibilidad

El medio de almacenamiento externo puede no estar disponible en el dispositivo, así que antes de realizar cualquier operación de lectura/escritura hemos de comprobar su disponibilidad mediante el método *getExternalStorageState()* de la clase *Environment*.

```
String estadoAE = Environment.getExternalStorageState();
switch (estadoAE) {
    case Environment.MEDIA_MOUNTED:
        // Disponible para lectura/excritura
    case Environment.MEDIA_MOUNTED_READ_ONLY:
        // Disponible sólo lectura
    case Environment.MEDIA_REMOVED:
        // No disponible
    default:
        // Resto de casos
}
```

18

Almacenamiento externo (3)

- Comprobar disponibilidad

Los posibles estados del medio de almacenamiento son los siguientes:

- MEDIA_BAD_REMOVAL: desconectado sin haber sido desmontado.
- MEDIA_CHECKING: comprobando el almacenamiento.
- MEDIA_MOUNTED: disponible (lectura/escritura).
- MEDIA_MOUNTED_READ_ONLY: disponible (sólo lectura).
- MEDIA_NOFS: formato no soportado.
- MEDIA_REMOVED: no disponible.
- MEDIA_SHARED: no montado, compartido con PC (USB).
- MEDIA_UNMOUNTABLE: no puede ser montado.
- MEDIA_UNMOUNTED: disponible pero no está montado.

19

Almacenamiento externo (4)

- Acceder a archivos privados de una app

Podemos obtener las rutas de los directorios de nuestra app utilizando el método *getExternalFileDir()* pasándole como parámetro el tipo de directorio:

- null: directorio raíz de la aplicación.
- DIRECTORY_DCIM: imágenes y fotos.
- DIRECTORY_DOWNLOADS: archivos descargados.
- DIRECTORY_MOVIES: videos.
- DIRECTORY_MUSIC: música.
- etc.

```
File archivo = getExternalFilesDir(Environment.DIRECTORY_DCIM);
String dir = archivo.getAbsolutePath();
```

20

Almacenamiento externo (5)

- Acceder a archivos compartidos

Los archivos compartidos se guardan en un directorio accesible para el usuario y el resto de aplicaciones.

Estos archivos no se eliminan al desinstalar la aplicación porque pueden seguir siendo necesarios.

Podemos obtener la ruta raíz de estos archivos mediante *Environment.getExternalStorageDirectory()*, y las rutas específicas para cada tipo de archivo con *Environment.getExternalStoragePublicDirectory(tipo)*, utilizando los mismos tipos del apartado anterior.

```
File archivo;
archivo = Environment.getExternalStorageDirectory();
String dirCompartido = archivo.getAbsolutePath();
archivo = Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DCIM);
String dirCompartidoDCIM = archivo.getAbsolutePath();
```

Almacenamiento externo (6)

- Escribir archivo

Tendremos que obtener la ruta del directorio en la que se almacenan nuestros archivos externos, obtener la referencia al archivo y almacenar los datos con los métodos de escritura de archivo habituales.

Nota: requiere permiso **WRITE_EXTERNAL_STORAGE**

```
try {
  String ruta = Environment.getExternalStorageDirectory().getAbsolutePath();
  File f = new File(ruta, "miArchivo.txt");
  OutputStreamWriter fout = new OutputStreamWriter(new FileOutputStream(f));
  fout.write("Texto de prueba");
  fout.close();
} catch (Exception ex) {
  // No se puede escribir el archivo
}
```

22

Almacenamiento externo (7)

- Leer archivo

Tendremos que obtener la ruta del directorio en la que se almacenan nuestros archivos externos, obtener la referencia al archivo y leer los datos con los métodos de lectura de archivo habituales.

Nota: requiere permiso **READ_EXTERNAL_STORAGE**

```
try {
  String ruta = Environment.getExternalStorageDirectory().getAbsolutePath();
  File f = new File(ruta, "miArchivo.txt");
  InputStreamReader fin = new InputStreamReader(new FileInputStream(f));
  char[] buffer = new char[1024];
  StringBuilder texto = new StringBuilder();
  while ((fin.read(buffer))!=-1) {
    texto.append(new String(buffer));
  }
} catch (Exception ex) {
  // No se puede leer el archivo
}
```