**MANEJO DE ARCHIVOS**

El manejo de archivos en una aplicación móvil es un tópico obligado, ya que en muchas ocasiones se desea tener acceso a contenido del mismo dispositivo y/o a bases de datos almacenadas de forma local. Y debido a que la base de los sistemas operativos sin exceptuar a Android son los archivos, queda de más mencionar que tanto se puede lograr con el manejo de los mismos.

Es por lo anterior que realizaremos una aplicación que pueda leer y escribir archivos tanto en el almacenamiento interno como en el externo.

1. Leer y escribir datos del almacenamiento interno (app sandbox). App sandbox lo podemos entender como un espacio reservado únicamente para la aplicación.
2. Crear un nuevo proyecto del tipo Aplicación de Android (Xamarin)
3. Llamarlo ManejoDeArchivos
4. Aplicación vacía
5. Minimun Android Version (4.4)
6. Abrir el archivo activity\_main.axml y sustituir el código que contiene, por este otro:

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical">

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:id="@+id/btnAbrirArchivo"

android:text="Abrir archivo"/>

<EditText

android:inputType="textMultiLine"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:minWidth="25px"

android:minHeight="25px"

android:id="@+id/etContentFile"

android:text="Hola mundo\nLínea 2\nLínea 3"/>

</LinearLayout>

1. Crear una nueva clase llamada AlmacenamientoInternoFiles.
2. Agregarle el nombre de espacio System.IO y agregar el siguiente código a la clase:

string pathAlmacenamientoInterno;

string pathAlmacenamientoInternoTemp;

public AlmacenamientoInternoFiles(string filesDir,string cacheDir)

{

pathAlmacenamientoInterno = filesDir;

pathAlmacenamientoInternoTemp = cacheDir;

}

public async Task<string> leerArchivo(string fileName)

{

string pathFile = Path.Combine(pathAlmacenamientoInterno,fileName);

string texto = "";

if (File.Exists(pathFile))

{

var reader = new StreamReader(pathFile);

string linea = "";

while ((linea = await reader.ReadLineAsync()) != null)

texto += linea;

}

return texto;

}

public void escribirArchivo(string fileName,string contenido)

{

string pathFile = Path.Combine(pathAlmacenamientoInterno, fileName);

File.WriteAllText(pathFile, contenido);

}

public string leerArchivoTemporal(string fileNameTmp)

{

fileNameTmp = fileNameTmp.Insert(fileNameTmp.IndexOf('.'), "\_tmp");

string pathFileTmp = Path.Combine(pathAlmacenamientoInternoTemp, fileNameTmp);

if (File.Exists(pathFileTmp))

return File.ReadAllText(pathFileTmp);

return "";

}

public void escribirArchivoTemporal(string fileNameTmp, string contenido)

{

fileNameTmp = fileNameTmp.Insert(fileNameTmp.IndexOf('.'), "\_tmp");

string pathFileTmp = Path.Combine(pathAlmacenamientoInternoTemp, fileNameTmp);

File.WriteAllText(pathFileTmp, contenido);

}

...

Los métodos leerArchivo() y escribirArchivo() son para leer y escribir respectivamente el texto de un determinado archivo en el almacenamiento local.

Los métodos leerArchivoTemporal() y escribirArchivoTemporal() son para leer y escribir respectivamente el texto de un determinado archivo temporal en el almacenamiento local temporal.

1. En el método OnCreate de la clase MainActivity, después de SetContentView(), agregar el siguiente código para crear un objeto que nos permita crear un archivo en el almacenamiento local y posteriormente leerlo y mostrarlo en el EditText por medio de un click del botón btnAbrirArchivo:

...

EditText etContentFile = FindViewById<EditText>(Resource.Id.etContentFile);

Button btnAbrirArchivo = FindViewById<Button>(Resource.Id.btnAbrirArchivo);

AlmacenamientoInternoFiles aiFiles = new AlmacenamientoInternoFiles(FilesDir.AbsolutePath, CacheDir.AbsolutePath);

aiFiles.escribirArchivo("archivo1.txt","Este es el contenido del archivo.\nEsta es la línea 2.\n Y esta es la línea 3.");

btnAbrirArchivo.Click += async (sender, e)=> {

etContentFile.Text = "";

etContentFile.Text = await aiFiles.leerArchivo("archivo1.txt");

};

...

Nota: FilesDir y CacheDir son objetos tipo Java.IO.File, y solamente están disponibles en el contexto de una Activity o AppCompatActivity, por eso es que está disponible desde el método OnCreate() y desde otra clase que no sea de tipo Activity o AppCompatActivity, no pueden ser accesadas.

La ubicación temporal CacheDir, solamente permite almacenar archivos con un espacio máximo de 1MB.

1. Guardar todo, compilar e implementar en emulador o dispositivo.
2. Ahora vamos a escribir y leer archivos temporales.
3. Agregar un botón al diseño activity\_main.xml, para que al darle click, se lea y muestre el archivo temporal en el EditText, para ello agregar el siguiente código después del cierre del Button con id “btnAbrirArchivo”:

...

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:id="@+id/btnAbrirArchivoTmp"

android:text="Abrir archivo temporal"/>

...

1. Crear la instancia del botón recién creado, crear un archivo temporal y delegarle al evento click del nuevo botón la tarea de abrir y mostrar el archivo temporal, para ello agregar el siguiente código dentro del método OnCreate() de la clase MainActivity, después de la subscripción del evento click al botón btnAbrirArchivo:

...

Button btnAbrirArchivoTmp = FindViewById<Button>(Resource.Id.btnAbrirArchivoTmp);

aiFiles.escribirArchivoTemporal("archivo1.txt", "Este es el contenido del archivo temporal.\nEsta es la línea 2.\n Y esta es la línea 3.");

btnAbrirArchivoTmp.Click += delegate {

etContentFile.Text = "";

etContentFile.Text = "Archivo archivo1\_tmp.txt almacenado en: " + System.IO.Path.Combine(CacheDir.AbsolutePath, "archivo1\_tmp.txt") + "\n";

etContentFile.Text += aiFiles.leerArchivoTemporal("archivo1.txt");

};

...

1. Borrar el archivo temporal cuando la aplicación es destruida, esto lo hacemos sobrescribiendo el método OnDestroy() de la clase main activity. Para ello primero hacemos que el objeto “aiFiles” contenido en el método OnCreate() sea global, en seguida agregamos el siguiente método a la clase AlmacenamientoInternoFiles después del método escribirArchivoTemporal():

...

public void eliminarArchivoTemporal(string fileNameTmp)

{

fileNameTmp = fileNameTmp.Insert(fileNameTmp.IndexOf('.'), "\_tmp");

string pathFileTmp = Path.Combine(pathAlmacenamientoInternoTemp, fileNameTmp);

File.Delete(fileNameTmp);

}

...

y posteriormente agregamos el siguiente código después del método OnCreate() de MainActivity:

...

protected override void OnDestroy()

{

base.OnDestroy();

aiFiles.eliminarArchivoTemporal("archivo1.txt");

}

...

En realidad borrar los archivos temporales de una aplicación no es obligatorio, ya que cuando el sistema Android requiere liberar espacio, los borra automáticamente

1. Guardar todo, compilar e implementar en emulador o dispositivo.
2. Leer y escribir archivos en el almacenamiento externo. El escribir archivos en el almcenamiento externo es muy similar a hacerlo en el interno, con la principal diferencia de que para el almacenamiento externo se le tienen que asignar permisos a la aplicación.
3. Ir al menú Proyecto->Propiedades de ManejoDeArchivos…->Manifiesto de Android->Permisos necesarios:
4. Seleccionar los permisos READ\_EXTERNAL\_STORAGE y WRITE\_EXTERNAL\_STORAGE y guardar cambios.
5. Verificamos que el medio externo esté disponible y sea escribible, para ello primero agregamos un botón a main\_activity.axml, después del botón con id “btnAbrirArchivoTmp”:

...

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:id="@+id/btnVEAE"

android:text="Verificar estado alacenamiento externo"/>

...

En segunda instancia creamos una referencia al botón “btnVEAE” y le suscribimos el método click para que verifique si el almacenamiento está disponible y si es escribible y/o leíble, esto dentro del método OnCreate() en la clase MainActivity, después de btnAbrirArchivoTmp.Click …:

...

Button btnVEAE = FindViewById<Button>(Resource.Id.btnVEAE);

btnVEAE.Click += delegate {

bool escribible = Android.OS.Environment.ExternalStorageState == Android.OS.Environment.MediaMounted;

bool leible = escribible || Android.OS.Environment.ExternalStorageState == Android.OS.Environment.MediaMountedReadOnly;

etContentFile.Text = string.Format("El almacenamiento externo {0} esta disponible, {1} es escribible y {2} es leíble",escribible?"si":"no",escribible?"si":"no",leible?"si":"no");

etContentFile.Text += string.Format("\nLa ruta del almacenamiento externo es: {0}",GetExternalFilesDir(null).AbsolutePath);

etContentFile.Text += string.Format("\nLa ruta del almacenamiento externo temporal es: {0}", ExternalCacheDir.AbsolutePath);

};

...

Con los métodos GetExternalFilesDirs() y GetExternalCacheDirs() podemos obtener los directorios de otros dispositivos de almacenamiento externo, tanto para alojar archivos persistentes asi comotemporales respectivamente. Por último con el atributo FreeSpace de FilesDir, se puede obtener el espacio disponible en el almacenamiento (sistema de archivos) externo primario.

Existen más características sobre el manejo de archivos en el almacenamiento externo y externo, por medio del método Android.OS.Environment.GetExternalStoragePublicDirectory(), combinado con los directorios públicos proveídos por Android.OS.Environment.

1. Guardar todo, compilar e implementar en emulador o dispositivo.

**Actividad complementaria**

Nota: Esta actividad es la onceava que impacta en la calificación para el criterio de evaluación “Prácticas varias”.

1. Realizar una aplicación que busque imágenes jpeg en todo los almacenamientos interno y externos del dispositivo y que muestre las ubicaciones en un EditText.