Anexo II. Capturar y almacenar imágenes

En el uso de las aplicaciones móviles, una de las principales funcionalidades suele ser la posibilidad de añadir imágenes, ya bien sea desde la galería, o desde la cámara de fotos del propio dispositivo.

El siguiente ejemplo, ***Capture And Save***, muestra como capturar imágenes desde la cámara de fotos y almacenar esas imágenes en el directorio privado de la aplicación, esto permite evitar tener que pedir permiso de almacenamiento.

El primer paso será crear un ***FileProvider***[1], este permite a la aplicación disponer de una forma segura de un contralor URI de contenido. Esto será en el fichero *Manifest*, en la etiqueta `application` se añadirá un nuevo hijo `provider`.

```
<application
    … >
    …
    <provider
        android:name="androidx.core.content.FileProvider"
        android:authorities="es.javiercarrasco.captureandsave"
        android:exported="false"
        android:grantUriPermissions="true">
        <meta-data
            android:name="android.support.FILE_PROVIDER_PATHS"
            android:resource="@xml/provider_file" />
    </provider>
    …
</application>
```

Los atributos utilizados para crear este ***provider***[2] son los siguientes:

- ***name***: se utiliza para especificar el nombre del proveedor de contenido, en este ejemplo ***androidx.core.content.FileProvider***.
- ***authorities***: es la lista de autoridades URI que identificarán los datos, no hay valor por defecto, por lo que al menos, deberá existir uno, el nombre de la subclase que implementa el proveedor.
- ***exported***: se utiliza para indicar si el proveedor de contenido estará disponible para otras aplicaciones, haciendo uso de la URI. Con el valor *false* se deniega dicho uso.
- ***grantUriPermissions***: se establece a *true* para conceder permiso al proveedor de contenido.

El elemento secundario `<meta-data>` incluido en el *provider* indicará los directorios que se compartirán, indicados en el fichero `provider_file.xml`, que se creará a continuación.

1 Configurar el uso de archivos compartidos (https://developer.android.com/training/secure-file-sharing/setup-sharing)
2 *Provider* (https://developer.android.com/guide/topics/manifest/provider-element)

Para crear el fichero `provider_file.xml` bastará con crear un nuevo ***Android Resource File*** con la siguiente configuración.

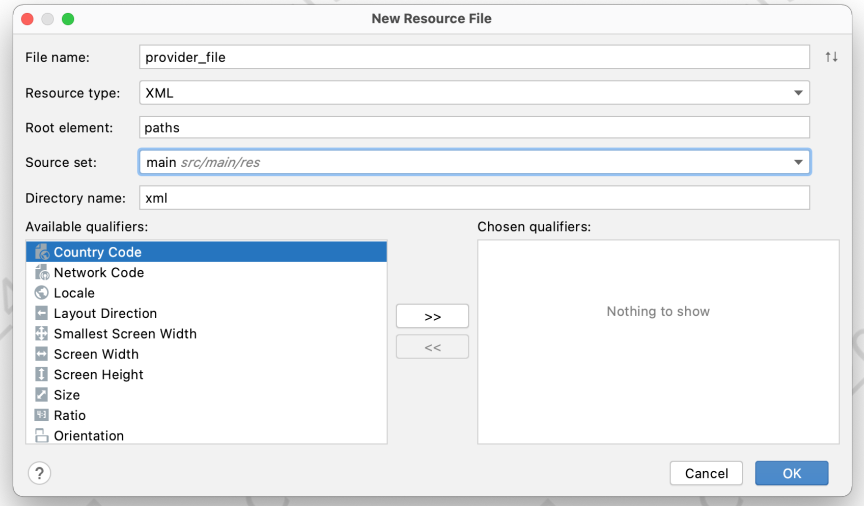

Figura 1

Esto creará un nuevo directorio, `res/xml/`, con el fichero XML, en el cual se especificará el nombre de los directorios a utilizar, que para este ejemplo quedará como se muestra.

```xml
<?xml version="1.0" encoding="utf-8"?>
<paths>
    <external-files-path
        name="images"
        path="Pictures" />
</paths>
```

Con esta configuración, la ruta donde se guardarán las imágenes capturadas será la siguiente:

```
/storage/self/primary/Android/data/es.javiercarrasco.captureandsave/files/Pictures
```

Además de la configuración del provider, se añadirá el siguiente permiso y requerimiento hardware para el uso de la cámara.

```xml
<uses-permission
    android:name="android.permission.CAMERA"
    android:required="true" />

<uses-feature
    android:name="android.hardware.camera"
    android:required="true" />
```

Con estos pasos, ya estaría preparado el sistema de almacenamiento, centraremos la atención ahora en obtener las imágenes a tamaño completo y almacenarlas. Para obtener la imagen completa, deberá pasarse a la cámara el contenedor de la imagen, es decir, el fichero donde debe almacenarse.

```
private var photoFile: File? = null
```

Justo antes de lanzar el *intent* sobre la cámara, se creará el contenedor y el *FileProvider* para poder almacenar el resultado (la imagen en cuestión).

```
// Se crea el fichero donde se guardará la imagen.
photoFile = ManageFiles().createImageFile(this)

val fileProvider =
    FileProvider.getUriForFile( // En base al provider creado en el Manifest.
        this,
        "es.javiercarrasco.captureandsave",
        photoFile!!
    )

// Se crea el intent y se le pasa el contenedor del fichero a recuperar.
val intent = Intent(MediaStore.ACTION_IMAGE_CAPTURE).apply {
    putExtra(
        MediaStore.EXTRA_OUTPUT, fileProvider
    )
}

resultTakePicture.launch(intent)
```

El método `createImageFile()` se encargará de crear el nombre del fichero y obtener la ruta completa.

```
fun createImageFile(context: Context): File {
    // Se crea un timeStamp para el nombre del fichero.
    val timeStamp = SimpleDateFormat("yyyyMMdd").format(Date())

    // Se obtiene el directorio según el path creado utilizando el provider.
    val directoryStorage = context.getExternalFilesDir(
        Environment.DIRECTORY_PICTURES
    )

    return File.createTempFile("IMG_$timeStamp", ".jpg", directoryStorage)
}
```

Una vez lanzado el *intent*, ya sólo queda esperar el resultado de la cámara, que puede ser *RESULT_OK* si se acepta por buena la foto o *RESULT_CANCELED* si se descarta. En este caso, se recogerá el resultado utilizando el intercambio de información por contrato.

```
// Propiedades creadas en la clase, el adaptador para el RV y un array
// para almacenar las imágenes capturadas.
private val myAdapter: ImageRVAdapter = ImageRVAdapter()
private val images: MutableList<File> = ArrayList()

…

// Se evalúa el resultado obtenido de la cámara.
var resultTakePicture = registerForActivityResult(StartActivityForResult())
{ result ->
    if (result.resultCode == Activity.RESULT_OK) {
        images.add(photoFile!!)
        photoFile = null
        myAdapter.notifyItemInserted(images.size)
    } else { // No se captura imagen.
        Toast.makeText(
            applicationContext,
            R.string.info_picture_canceled,
            Toast.LENGTH_SHORT
        ).show()
    }
}
```

Como puedes observar, cuando se captura la imagen correctamente, se añade a un array, se limpia la variable `photoFile` y se actualiza el adaptador del *Recycler View*. Esta acción permite almacenar las imágenes en el directorio especificado, observa el tamaño de las imágenes, es mayor que obtenido con los *thumbnails*.

Figura 3

Puedes consultar el código completo de la aplicación en el repositorio, en la sección ANEXOS.

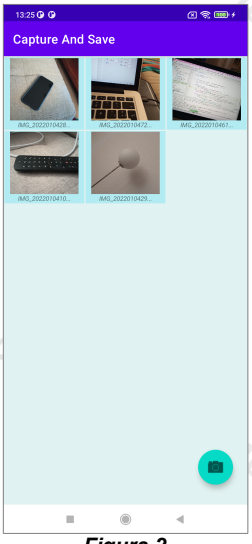

Figura 2