Interfaz de usuario en Android: Controles de selección (IV)

Tras haber visto en artículos anteriores los dos controles de selección más comunes en cualquier interfaz gráfica, como son las listas desplegables (`Spinner`) y las listas "fijas" (`ListView`), tanto en su versión básica como optimizada, en este nuevo artículo vamos a terminar de comentar los controles de selección con otro menos común pero no por ello menos útil, el control `GridView`.

El control `GridView` de Android presenta al usuario un conjunto de opciones seleccionables distribuidas de forma tabular, o dicho de otra forma, divididas en filas y columnas. Dada la naturaleza del control ya podéis imaginar sus propiedades más importantes, que paso a enumerar a continuación:

- `android:numColumns`, indica el número de columnas de la tabla o "`auto_fit`" si queremos que sea calculado por el propio sistema operativo a partir de las siguientes propiedades.
- `android:columnWidth`, indica el ancho de las columnas de la tabla.
- `android:horizontalSpacing`, indica el espacio horizontal entre celdas.
- `android:verticalSpacing`, indica el espacio vertical entre celdas.
- `android:stretchMode`, indica qué hacer con el espacio horizontal sobrante. Si se establece al valor "`columnWidth`" este espacio será absorbido a partes iguales por las columnas de la tabla. Si por el contrario se establece a "`spacingWidth`" será absorbido a partes iguales por los espacios entre celdas.

Veamos cómo definiríamos un `GridView` de ejemplo en nuestra aplicación:

```
<GridView android:id="@+id/GridOpciones"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:numColumns="auto_fit"
    android:columnWidth="80px"
    android:horizontalSpacing="5dp"
    android:verticalSpacing="10dp"
    android:stretchMode="columnWidth" />
```

Una vez definida la interfaz de usuario, la forma de asignar los datos desde el código de la aplicación es completamente análoga a la ya comentada tanto para las listas desplegables como para las listas estáticas: creamos un array genérico que contenga nuestros datos de prueba, declaramos un adaptador de tipo `ArrayAdapter` (como ya comentamos, si los datos proceden de una base de datos lo normal será utilizar un`SimpleCursorAdapter`, pero de eso nos ocuparemos más adelante en el curso) pasándole en este caso un layout genérico (`simple_list_item_1`, compuesto por un simple `TextView`) y asociamos el adaptador al control `GridView` mediante su método `setAdapter()`:

```
private String[] datos = new String[50];
//...
for(int i=1; i<=50; i++)
        datos[i-1] = "Dato " + i;

ArrayAdapter<String> adaptador =
        new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, dato

grdOpciones = (GridView)findViewById(R.id.GridOpciones);

grdOpciones.setAdapter(adaptador);
```

Por defecto, los datos del array se añadirán al control `GridView` ordenados por filas, y por supuesto, si no caben todos en la pantalla se podrá hacer *scroll* sobre la tabla. Vemos en una imagen cómo queda nuestra aplicación de prueba:

En cuanto a los eventos disponibles, el más interesante vuelve a ser el lanzado al seleccionarse una celda determinada de la tabla: `onItemClick`. Este evento podemos capturarlo de la misma forma que hacíamos con los controles `Spinner` y `ListView`. Veamos un ejemplo de cómo hacerlo:

```
grdOpciones.setOnItemClickListener(
    new AdapterView.OnItemClickListener() {
        public void onItemClick(AdapterView<?> parent,
            android.view.View v, int position, long id) {
                lblMensaje.setText("Opción seleccionada: "
                    + parent.getItemAtPosition(position));
            }
    });
```

Todo lo comentado hasta el momento se refiere al uso básico del control `GridView`, pero por supuesto podríamos aplicar de forma prácticamente directa todo lo comentado para las listas en los dos artículos anteriores, es decir, la personalización de las celdas para presentar datos complejos creando nuestro propio adaptador, y las distintas optimizaciones para mejorar el rendimiento de la aplicación y el gasto de batería.

Puedes consultar y/o descargar el código completo de los ejemplos desarrollados en este artículo accediendo a la página del curso en GitHub.