Programación multimedia y dispositivos móviles

UF3.7 RecyclerViews

Universidad Europea

LAUREATE INTERNATIONAL UNIVERSITIES

RecyclerViews

El RecyclerView es una versión más flexible y avanzada de ListView.

Usamos RecyclerView para mostrar grandes conjuntos de datos de manera que el desplazamiento entre ellos sea eficiente al mantener una cantidad limitada de vistas.

Lo que hace un RecyclerView es mostrar datos cuyos elementos se van reciclando cuando ya no son visibles por el scroll de la lista.

RecyclerViews

Cuando usemos RecyclerView tendremos que utilizar estos 3 componentes:

- RecyclerView.Adapter
- RecyclerView.ViewHolder
- LayoutManager o administrador de diseño

El adaptador acercará el modelo de datos que debe ser mostrados y el LayoutManager será el responsable de posicionar cada ítem dentro del RecyclerView y decidir cuándo reciclar las vistas de items que ya no son visibles.

Además habrá que definir un **layout** xml para especificar el formato gráfico con el que se mostrarán los datos.

RecyclerViews. activity.xml

Para incluir un **RecyclerView** en el layout de nuestro activity se deberá incluir el siguiente código:

```
<android.support.v7.widget.RecyclerView
    android:id="@+id/my_recycler_view"
    android:scrollbars="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

Debemos asegurarnos de que en la sección de dependencias (dependencies) del fichero ***build.gradle*** del módulo principal está la referencia a la librería de soporte *recyclerview-v7*, lo que nos permitirá el uso del componente RecyclerView en la aplicación:

```
compile 'com.android.support:recyclerview-v7:+'
```

RecyclerViews. Adapter

Para definir el **adaptador** habrá que crear una clase que extienda de **RecyclerView.Adapter** que usará una instancia de **RecyclerView.ViewHolder**.

```
public class MiAdaptador extends
RecyclerView.Adapter<MiAdaptador.ViewHolder> {}
```

Este ViewHolder se encargará de tomar los valores del layout.

Si necesitamos manejar un ViewHolder en concreto se deberá definir con una clase interna.

```
public class MiAdaptador extends
RecyclerView.Adapter<MiAdaptador.MiViewHolder>
{}
```

Dentro de la clase MiAdaptador se deberá incluir lo siguiente:

Se deberá definir un atributo para almacenar la lista con los datos.

```
private String[] datos;
```

O

```
private ArrayList<Patron> datos;
```

En el caso de que los datos a representar sean algo más que una simple cadena de texto.

El adaptador deberá tener un constructor donde se inicialice la lista de datos:

```
public MiAdaptador(String[] datos) {
   this.datos = datos;
}
```

O

```
public MiAdaptador(ArrayList<Patron> datos) {
        this.datos = datos;
}
```

En el caso de necesitar un ViewHolder específico:

```
public static class MiViewHolder
            extends RecyclerView.ViewHolder {
    private TextView txtUno;
    private TextView txtDos;

    public MiViewHolder(View itemView) {
        super(itemView);
        txtUno = (TextView)itemView.findViewById(R.id.txtUno);
        txtDos = (TextView)itemView.findViewById(R.id.txtDos);
    }

    public void bindPatron(Patron p) {
    txtUno.setText(p.getAtributo1());
    txtDos.setText(p.getAtributo2());
    }
}
```

Y se deberán implementar los siguientes métodos:

El método **onCreateViewHolder**() que se limita a “inflar” una vista a partir del layout correspondiente a los elementos de la lista, y crear y devolver un nuevo ViewHolder pasándole dicha vista como parámetro.

```
@Override
public MiAdaptador.ViewHolder onCreateViewHolder(
ViewGroup parent, int viewType) {
    View v = LayoutInflater.from(parent.getContext())
                .inflate(R.layout.mi_layout, parent, false);
    ViewHolder vh = new ViewHolder(v);
    return vh;
}
```

En el caso de tener un ViewHolder específico:

```
@Override
public MiViewHolder onCreateViewHolder(ViewGroup viewGroup,
int viewType) {

    View itemView =
LayoutInflater.from(viewGroup.getContext())
    .inflate(R.layout.mi_layout, viewGroup, false);

    MiViewHolder mvh = new MiViewHolder(itemView);

    return mvh;
}
```

En **onBindViewHolder**() tan sólo tendremos que recuperar el objeto correspondiente a la posición recibida como parámetro y asignar sus datos sobre el ViewHolder también recibido como parámetro:

```
@Override
public void onBindViewHolder(ViewHolder holder, int position) {
    // get element from your dataset at this position
    // replace the contents of the view with that element
    holder.TextView.setText(datos[position]);
}
```

Para un ViewHolder específico:

```
public void onBindViewHolder(MiViewHolder holder, int position) {
    holder.txtUno.setText(datos.get(position).getAtributo1());
    holder.txtDos.setText(datos.get(position).getAtributo2());
}
```

Y el método **getItemCount**() que tan sólo devolverá el tamaño de la lista de datos:

```
@Override
    public int getItemCount() {
        return datos.size();
    }
```

Con esto tendríamos finalizado el adaptador, por lo que ya podríamos asignarlo al RecyclerView en nuestra. Esto es tan sencillo como lo era en el caso de ListView.

RecyclerViews. Activity.java

Tendremos que crear nuestro adaptador pasándole como parámetro la lista de datos y asignarlo al control RecyclerView mediante setAdapter():

```
    miRecyclerView =
(RecyclerView)findViewById(R.id.my_recycler_view);

    miRecyclerView.setHasFixedSize(true);

    miLayoutManager = new LinearLayoutManager(this);
    miRecyclerView.setLayoutManager(miLayoutManager);

    miAdapter = new MiAdaptador(datos);
    miRecyclerView.setAdapter(miAdapter);
```

RecyclerViews. Activity.java

El método **setHasFixedSize**(), aunque no es obligatorio, sí es conveniente usarlo cuando estemos seguros de que el tamaño de nuestro RecyclerView no va a variar (por ejemplo debido a cambios en el contenido del adaptador), ya que esto permitirá aplicar determinadas optimizaciones sobre el control.

El siguiente paso será asociar al RecyclerView un LayoutManager determinado, para determinar la forma en la que se distribuirán los datos en pantalla.

Si lo que queremos es mostrar elementos en una lista de desplazamiento horizontal o vertical usaremos el **LinearLayoutManayer** incorporado en el RecyclerView.

Si lo que queremos es mostrar elementos en una cuadrícula usaremos el **GridLayoutManager.**

RecyclerViews. Eventos

La gran sorpresa del RecyclerView es que no incluye un evento onItemClick() como ocurría en el caso de ListView.

RecyclerView delega también esta tarea a otro componente, en este caso a la propia vista que conforma cada elemento de la colección.

Aprovecharemos la creación de cada nuevo ViewHolder para asignar a su vista asociada el evento onClick. Además desde fuera del adaptador, incluiremos el listener correspondiente como atributo del adaptador, y dentro de éste nos limitaremos a asignar el evento a la vista del nuevo ViewHolder y a lanzarlo cuando sea necesario desde el método onClick().

implements View.OnClickListener

RecyclerViews. Eventos

```
public class MiAdaptador extends
RecyclerView.Adapter<MiAdaptador.MiViewHolder>
implements View.OnClickListener {

    …

    private View.OnClickListener listener;

    …
```

En la definición del ViewHolder:

```
    …

    itemView.setOnClickListener(this);

    …
```

RecyclerViews. Eventos

El método setOnClickListener() nos servirá para asociar el listener real a nuestro adaptador en el momento de crearlo :

```
    public void
setOnClickListener(View.OnClickListener listener) {
        this.listener = listener;
    }
```

La implementación del onClick(), se limitará a lanzar el mismo evento sobre el listener externo

```
    @Override
    public void onClick(View view) {
        if(listener != null)
            listener.onClick(view);
    }
```

Por último en la actividad, tras crear el adaptador:

```
    miAdapter = new MiAdaptador(datos);

    miAdapter.setOnClickListener(new
View.OnClickListener() {
        @Override
        public void onClick(View v) {
            String msg = "Seleccionada la opción "
+ miRecyclerView.getChildPosition(v) ;
            Toast.makeText(MainActivity.this,
msg,Toast.LENGTH_SHORT).show();
            }
        });
```

RecyclerViews. Ejemplo

Ahora crearemos un nuevo proyecto basándonos en el proyecto ListView2.

- En el activity.xml en vez de un ListView incluiremos un RecyclerView.
- Usaremos el mismo LinearLayout, listview_item.xml, para cada uno de los elementos.
- Necesitaremos la clase ListaItem que encapsule los datos de cada elemento.
- Tendremos que crear la clase Adaptador.
- Crearemos una clase Datos como fuente de datos.
- Relacionaremos todo en el Activity.java

RecyclerViews. Ejemplo

La clase Adaptador:

```
public class MiAdaptador extends RecyclerView.Adapter<MiAdaptador.MiViewHolder>
implements View.OnClickListener {
    private ArrayList<ListaItem> datos;
    private View.OnClickListener listener;

    public static class MiViewHolder extends RecyclerView.ViewHolder {
        private ImageView imagen;
        private TextView textoSup;
        private TextView textoInf;

        public MiViewHolder(View view) {
            super(view);
            imagen = (ImageView) view.findViewById(R.id.imageView);
            textoSup = (TextView) view.findViewById(R.id.tvSuperior);
            textoInf = (TextView) view.findViewById(R.id.tvInferior);
        }

        public void bindListaItem(ListaItem li) {
        imagen.setImageResource(li.getImagen());
            textoSup.setText(li.getTextoSup());
            textoInf.setText(li.getTextoInf());
        }
    }
```

RecyclerViews. Ejemplo

```
    public MiAdaptador(ArrayList<ListaItem> datos) {
        this.datos = datos;
    }

    @Override
    public MiAdaptador.MiViewHolder onCreateViewHolder(
ViewGroup parent, int viewType) {
        View v = LayoutInflater.from(parent.getContext())
              .inflate(R.layout.listview_item, parent, false);
        v.setOnClickListener(this);
        MiViewHolder vh = new MiViewHolder(v);
        return vh;
    }

    public void onBindViewHolder(MiViewHolder holder, int position) {
        holder.imagen.setImageResource(datos.get(position).getImagen());
        holder.textoSup.setText(datos.get(position). getTextoSup());
        holder.textoInf.setText(datos.get(position). getTextoInf());
    }

    @Override
    public int getItemCount() {
        return datos.size();
    }
```

RecyclerViews. Ejemplo

```
    public void setOnClickListener(View.OnClickListener listener) {
        this.listener = listener;
    }

    @Override
    public void onClick(View view) {
        if(listener != null)
            listener.onClick(view);
    }


}
```

Creamos la clase Datos:

```
public class Datos {
    private ArrayList<ItemLista> lista;

    public Datos() {
     lista = new ArrayList<ItemLista>();
     cargarDatos();
    }

    private void cargarDatos() {
        lista.add(new ItemLista(R.drawable.im_buho, "BUHO",
    "Búho es el nombre común de aves de la familia Strigidae, del orden de las
estrigiformes o aves rapaces nocturnas. Habitualmente designa especies que, a
diferencia de las lechuzas, tienen plumas alzadas que parecen orejas."));
        lista.add(new ItemLista(R.drawable.im_colibri, "COLIBRÍ",
    "Los troquilinos (Trochilinae) son una subfamilia de aves apodiformes de la
familia Trochilidae, conocidas vulgarmente como colibríes, quindes, tucusitos,
picaflores, chupamirtos, chuparrosas, huichichiquis (idioma nahuatl), mainumby (idioma
guaraní) o guanumby. Conjuntamente con las ermitas, que pertenecen a la subfamilia
Phaethornithinae, conforman la familia Trochilidae que, en la sistemática de Charles
Sibley, se clasifica en un orden propio: Trochiliformes, independiente de los vencejos
del orden Apodiformes. La subfamilia Trochilinae incluye más de 100 géneros que
comprenden un total de 330 a 340 especies."));
```

RecyclerViews. Ejemplo

```
        lista.add(new ItemLista(R.drawable.im_cuervo, "CUERVO",
    "El cuervo común (Corvus corax) es una especie de ave paseriforme de la familia de los córvidos (Corvidae). Presente en todo el hemisferio septentrional, es la especie de córvido con la mayor superficie de distribución. Con el cuervo de pico grueso, es el mayor de los córvidos y probablemente la paseriforme más pesada; en su madurez, el cuervo común mide entre 52 y 69 centímetros de longitud y su peso varía de 0,69 a 1,7 kilogramos. Los cuervos comunes viven generalmente de 10 a 15 años pero algunos individuos han vivido 40 años. Los juveniles pueden desplazarse en grupos pero las parejas ya formadas permanecen juntas toda su vida, cada pareja defendiendo un territorio. Existen 8 subespecies conocidas que se diferencian muy poco aparentemente, aunque estudios recientes hayan demostrado diferencias genéticas significativas entre las poblaciones de distintas regiones."));
        lista.add(new ItemLista(R.drawable.im_flamenco, "FLAMENCO",
     "Los fenicopteriformes (Phoenicopteriformes), los cuales reciben el nombre vulgar de flamencos, son un orden de aves neognatas, con un único género viviente: Phoenicopterus. Son aves que se distribuyen tanto por el hemisferio occidental como por el hemisferio oriental: existen cuatro especies en América y dos en el Viejo Mundo. Tienen cráneo desmognato holorrino, con 16 a 20 vértebras cervicales y pies anisodáctilos."));
```

RecyclerViews. Ejemplo

```
        lista.add(new ItemLista(R.drawable.im_kiwi, "KIWI",
    "Los kiwis (Apterix, gr. 'sin alas') son un género de aves paleognatas compuesto
por cinco especies endémicas de Nueva Zelanda.1 2 Son aves no voladoras pequeñas,
aproximadamente del tamaño de una gallina. Antes de la llegada de los humanos
alrededor del año 1300, en Nueva Zelanda los únicos mamíferos que había eran
murciélagos, y los nichos ecológicos que en otras partes del mundo eran ocupados por
animales tan diversos como caballos, lobos y ratones fueron utilizados en Nueva
Zelanda por pájaros (y en menor proporción por ciertas especies de reptiles). La
denominación kiwi es maorí, idioma del pueblo homónimo de linaje malayopolinesio que
colonizó Nueva Zelanda antes de la llegada de los europeos."));
        lista.add(new ItemLista(R.drawable.im_loro, "LORO",
    "Las Psitácidas (Psittacidae) son una familia de aves psitaciformes llamadas
comúnmente loros o papagayos, e incluye a los guacamayos, las cotorras, los
periquitos, los agapornis y formas afines."));
        lista.add(new ItemLista(R.drawable.im_pavo, "PAVO",
    "Pavo es un género de aves galliformes de la familia Phasianidae, que incluye dos
especies, el pavo real común (Pavo cristatus) y el pavo real cuelliverde (Pavo
muticus).1"));
        lista.add(new ItemLista(R.drawable.im_pinguino, "PINGÜINO",
     "Los pingüinos (familia Spheniscidae, orden Sphenisciformes) son un grupo de aves
marinas, no voladoras, que se distribuyen únicamente en el Hemisferio Sur, sobre todo
en sus altas latitudes."));
    }
}
```

RecyclerViews. Ejemplo

En el Activity.java:

```
protected void onCreate(Bundle savedInstanceState) {
    miRecyclerView = (RecyclerView)findViewById(R.id.my_recycler_view);

    miRecyclerView.setHasFixedSize(true);

    miLayoutManager = new LinearLayoutManager(this);
    miRecyclerView.setLayoutManager(miLayoutManager);

    miAdapter = new MiAdaptador(datos);
    miAdapter.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            String msg = "Seleccionada la opción " + miRecyclerView.
miRecyclerView.indexOfChild(v);
            Toast.makeText(MainActivity.this,
msg,Toast.LENGTH_SHORT).show();
            }
        });

    miRecyclerView.setAdapter(miAdapter);
}
```

Universidad Europea

LAUREATE INTERNATIONAL UNIVERSITIES

Madrid Valencia Canarias