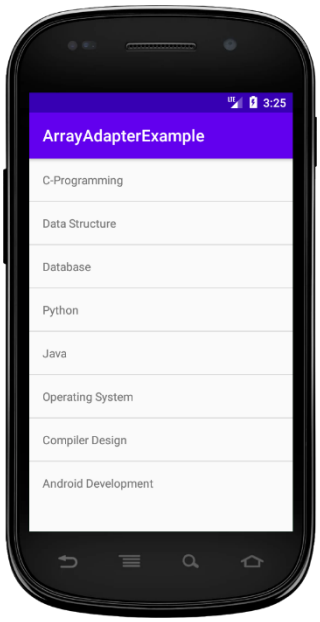
Adaptadores y Listas

Una lista es un *widget* que muestra una serie de elementos ordenados de forma vertical y con scroll. Es muy posible que los estés utilizando todo el rato sin ni siquiera darte cuenta, como por ejemplo, cuando eliges una canción de una app de música.

Listas



Android ofrece dos componentes para trabajar con listas: el **ListView** y el **RecyclerView**. Cualquiera de los dos puede ser incluido en un layout de la forma habitual, normalmente ocupando toda la pantalla. **ListView** existe desde la primera versión de Android, mientras que **RecyclerView** desde Android 5.0 (*Lollipop*). Se recomienda usar siempre la **RecyclerView**, dado que está pensada para grandes listados y hace énfasis en la reutilización de vistas de los elementos que no se están mostrando en pantalla en ese momento, mejorando la eficiencia. Eso sí, programarla es un poco más complejo. Adicionalmente, los adaptadores usados para una **RecyclerView** son diferentes a los de la **ListView**.

Adaptadores

Los **adaptadores (adapters)** son vitales para trabajar con listas, ya que vinculan cada elemento que queremos mostrar en la lista con cada uno de los ítems de la lista. Por explicarlo brevemente, si queremos mostrar en una lista el contenido de un ArrayList de objetos Factura, tendremos que decirle a la lista exactamente cómo y dónde tiene que poder el producto, el precio, la fecha, etc. de cada una de las Facturas. De esto es de lo que se encarga un **adapter**.

Todos los adapters de un **ListView** descienden de la clase BaseAdapter. Android dispone de dos, el **ArrayAdapter** y el **CursorAdapter**. El primero maneja datos provenientes de un ArrayList y el segundo de una Base de Datos. Obviamente, también puedes crearte tu propio adapter customizado, algo bastante habitual.

El cambio, no se ofrece ningún adaptador para **RecyclerView**. Todos los adaptadores deberán de heredar de RecyclerView.Adapter y ser customizados.

Por cierto… los adapters NO SON exclusivos de las listas…

Adapters para [Java]

Antes de hacer nada, hay que definir un nuevo *layout* para cada uno de los ítems de la lista. Este nuevo *layout* que he llamado **list_line.xml** debe de tener en este caso un TextView para mostrar el contenido de *valores*.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:id="@+id/itemTextView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center" />

</LinearLayout>
```

Una vez lo tenemos, basta con asignárselo a la lista.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    ListView myList = findViewById(R.id.myList);

    List<String> valores = new ArrayList<>(Arrays.asList("Elemento1", "Elemento2", "Elemento3"));

    ArrayAdapter<String> arrayAdapter = new ArrayAdapter<>( context: this,
            R.layout.list_line, R.id.itemTextView , valores);

    myList.setAdapter(arrayAdapter);
}
```

Mi adapter personalizado [Java]

El adapter básico que hemos visto anteriormente está bien para mostrar un sólo texto… pero muchos de los listados incluyen imágenes, botones que se pueden pulsar, etc. Para hacer esto, nos tenemos que crear nuestro propio adapter. Partamos de la idea de que tenemos que mostrar un ArrayList de objetos en un **ListView**. Lo primero será definir nuestro propio *layout* **list_line.xml** y especificar los elementos de cada ítem de la lista: TextView, ImageView…

A continuación, creamos una nueva clase **adapter** que extenderá de **ArrayAdapter**. Le pasaremos el ArrayList de los elementos a mostrar y el Contexto. En el método **getView ()** estaremos referenciando cada uno de los elementos del **list_line.xml** y relacionándolos con cada uno de los objetos del ArrayList. **Position**, la variable que controla exactamente qué fila se está mostrando en ese momento.

```
public class TaskAdapter extends ArrayAdapter <Objeto> {

    private ArrayList<Objeto> listado;
    private Context context;

    public TaskAdapter(Context context, int textViewResourceId, ArrayList<Objeto> listado) {
        super(context, textViewResourceId, listado);
        this.listado = listado;
        this.context = context;
    }

    @Override
    public int getCount() {
        return super.getCount();
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        LayoutInflater inflater = (LayoutInflater) getContext().getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        View view = inflater.inflate(R.layout.linea_layout, null);

        TextView taskItemOne = (TextView) view.findViewById(R.id.textView);

        taskItemOne.setText(listado.get(position).getName());

        return view;
    }
}
```

Este ejemplo el adaptador únicamente tiene un TextView; pero puedes añadir todos los que quieras. Es incluso posible añadir *widgets* y asignarles un evento. Por ejemplo, puedes hacer que un icono en cada ítem de la lista sirva para borrar ese elemento, o añadir un 'like', etc.

Ahora que ya lo tienes todo definido, basta con utilizar el adapter en nuestra *activity*:

```
MyAdapter myAdapter = new TaskAdapter(MainActivity.this, R.layout.linea_layout, listado);
listView.setAdapter(myAdapter);
```

Si realizar cambios el **ListView**, por ejemplo, modificando el listado, será preciso que refresques el adaptador mediante la instrucción **notifyDataSetChanged ().**

Práctica 21

[Java] Crea una app que muestre en un ListView una serie de objetos Usuario. Cada objeto Usuario tiene un nombre, un apellido y una edad. Crea un adapter que muestre los tres elementos. Utiliza un **ListView**.

Práctica 22

[Java] Crea una app que muestre en un ListView una serie de objetos Usuario. Cada objeto Usuario tiene un nombre, un apellido y una edad. Crea un adapter que muestre los tres elementos. Utiliza un **RecyclerView**.

Práctica 23

[Java] Crea una app que muestre en un ListView una serie de objetos Usuario. Cada objeto Usuario tiene un nombre, un apellido y una edad. Crea un adapter que muestre los tres elementos, pero en lugar de la edad, muestra un icono si es mayor de 18 y otro diferente si no lo es. Pulsar el icono hará que éste cambie. (Acuérdate de refrescar el ListView)