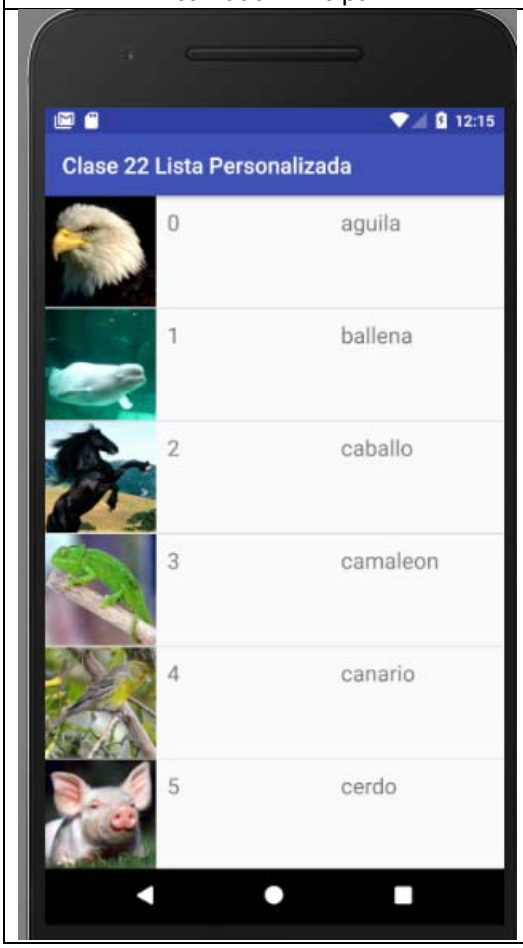
PRACTICA 22. MANEJO DE LISTVIEW PERSONALIZADOS.

En esta práctica veremos cómo crear interfaces personalizadas, como las que se ilustra a continuación:

Actividad Principal	Actividad Secundaria
	

Genere un proyecto del tipo empty activity y cambie el layout a LinearLayout.

```
<?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"

tools:context="com.example.delio.clase22listapersonalizada.MainActivity">

</LinearLayout>
```

A esta actividad agregamos un <ListView>

```
<ListView
    android:id="@+id/lstAnimales"
    android:padding="10sp"
    android:layout_margin="10dp"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
</ListView>
```

Ahora agregamos un layout de nombre itemlist.xml, el cual tendrá una distribución de una imagen y dos textos, como se indica:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/item"
    android:orientation="horizontal"
    android:layout_margin="10dp"
    android:padding="10dp"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <ImageView
        android:id="@+id/imgAnimal"
        android:layout_width="75dp"
        android:layout_height="75dp"
        android:src="@android:drawable/ic_dialog_info"/>
    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="match_parent">
        <TextView
            android:id="@+id/txtPosicion"
            android:layout_weight="1"
            android:textSize="20sp"
            android:layout_margin="10dp"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />
        <TextView
            android:id="@+id/txtAnimal"
            android:layout_weight="1"
            android:textSize="20sp"
            android:layout_margin="10dp"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />
    </LinearLayout>
</LinearLayout>
```

Ahora pasaremos a crear una clase base que represente de forma lógica lo que se va a desplegar, en este caso, el nombre del animal y el recurso de la imagen:

```
public class Animal {


        private String AnimalName;
      private int ImageID;


}
```

Así debe de quedar la clase base.

```
public class Animal {
    private String AnimalName;
    private int ImageID;

    public Animal(String animalName, int imageID) {
        AnimalName = animalName;
        ImageID = imageID;
    }

    public String getAnimalName() {
        return AnimalName;
    }

    public void setAnimalName(String animalName) {
        AnimalName = animalName;
    }

    public int getImageID() {
        return ImageID;
    }

    public void setImageID(int imageID) {
        ImageID = imageID;
    }
}
```

Adaptador

Una vez hecho esto, debemos ahora de crear nuestro adaptador de datos en función de esta última clase generada.

```
public class AdapterAnimal {

}
```

Haga que extienda de la clase BaseAdapter:

```
import android.widget.ArrayAdapter;

/**
 * Created by delio on 12/10/2017.
 */

public class AdapterAnimal extends ArrayAdapter<Animal> {

}
```

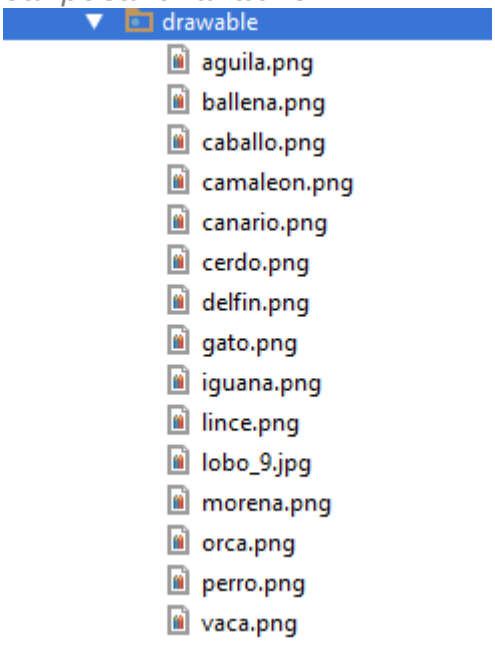
Vamos a modificar el código, creamos su constructor y atributos de la clase de la siguiente forma:

```
public class AdapterAnimal extends ArrayAdapter<Animal> {
    private Context context;
    // Creamos una Lista objetos  del tipo animal
    private ArrayList<Animal> datos;

    public AdapterAnimal(Context context, ArrayList datos) {
        super(context, R.layout.itemlist, datos);
        // Guardamos los parámetros en variables de clase.
        this.context = context;
        this.datos = datos;

    }
}
```

Después de preparar el objeto, debemos de manejar la vista mediante el código:

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {
    // En primer lugar "inflamos" una nueva vista, que será la que se
    // mostrará en la celda del ListView. Para ello primero creamos el
    // inflater, y después inflamos la vista.
    LayoutInflater inflater = LayoutInflater.from(context);
    View item = inflater.inflate(R.layout.itemlist, null);

    // A partir de la vista, recogeremos los controles que contiene para
    // poder manipularlos.
    // Recogemos el ImageView y le asignamos una foto.
    ImageView imgAnimal = (ImageView) item.findViewById(R.id.imgAnimal);
    imgAnimal.setImageResource(datos.get(position).getImageID());

    // Recogemos el TextView para mostrar el nombre y establecemos el
    // nombre.
    TextView txtNombre = (TextView) item.findViewById(R.id.txtAnimal);
    txtNombre.setText(datos.get(position).getAnimalName());

    // Recogemos el TextView para mostrar el número de celda y lo
    // establecemos.
    TextView txtPosicion = (TextView)
item.findViewById(R.id.txtPosicion);
    txtPosicion.setText(String.valueOf(position));

    // Devolvemos la vista para que se muestre en el ListView.
    return item;
}
```

Ahora debemos de copiar las imágenes de los animales en la carpeta drawable:

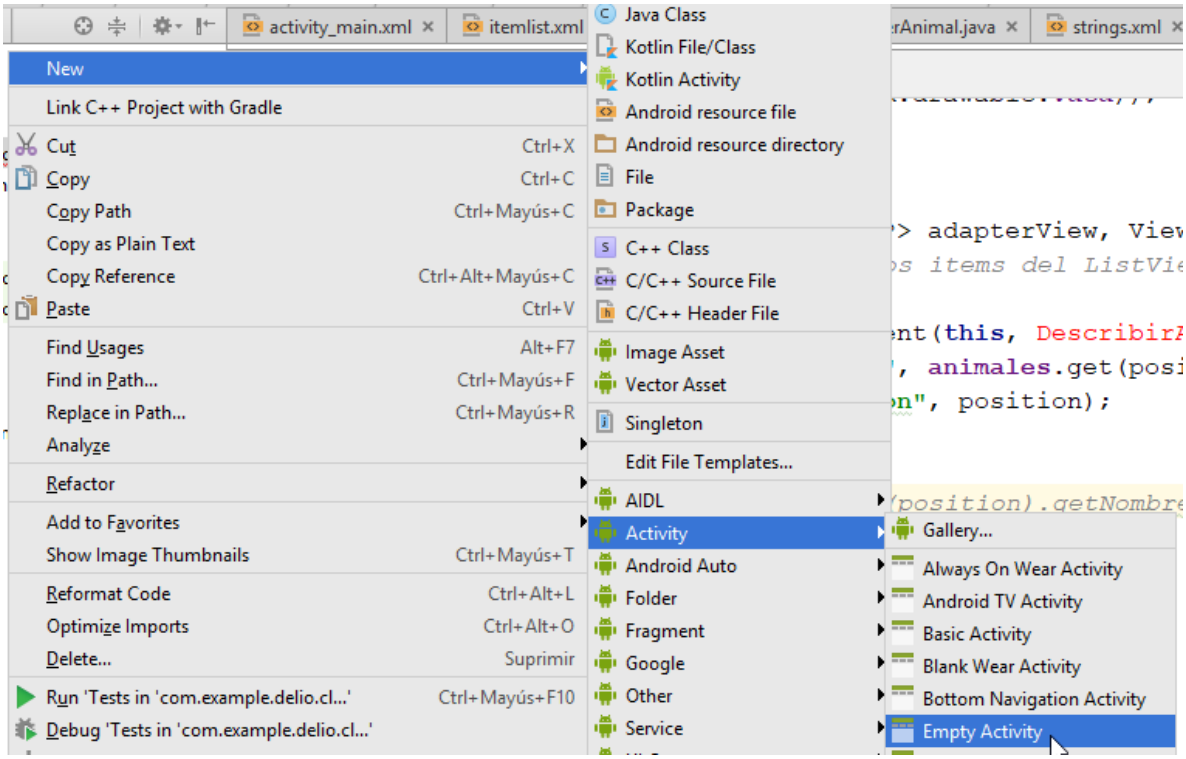


Posteriormente modificar el archivo string.xml agregando un arreglo con una breve descripción de cada animal.

```
<resources>
    <string name="app_name">Clase 22 Lista Personalizada</string>
<string-array name="descripcion">
    <item>El Aguila es una ave con capacidades de vision
excelentes</item>
    <item>La ballena cetaceo mamifero mas grande del oceanno </item>
    <item>El caballo animal cuadrupedo usado para transportar
personas</item>
    <item>El camaleon animal que su principal atractivo es el arte de
camuflagearse</item>
    <item>El canario es una ave con excelentes dotes de canto</item>
    <item>El cerdo un animal de granja y de el se obtienen varios
productos consumibles</item>
    <item>El delfin mamifero marino, agil e inteligente que convive en
grupos</item>
    <item>El Gato es un felino casero ideal para proteger la casa de
roedores</item>
    <item>La iguana es un animal de ornato</item>
    <item>El lince es un animal en peligro de desaparecer</item>
    <item>El lobo es un cazador solitario que vive en las zonas
frias</item>
    <item>La morena anguila agresiva de dientes filosos</item>
    <item>La Orca es un mamifero marino de la familia de los
delfines</item>
    <item>El perro es el mejor animal de compañia para un niño</item>
    <item>La vaca animal de granja productor de leche y carne</item>
</string-array>

</resources>
```

SEGUNDA ACTIVIDAD.

Necesitamos preparar la segunda actividad que reaccionara al seleccionar un elemento de la lista, por lo tanto creamos otra actividad vacía.



A la cual llamaremos DescribirAnimal (Recuerde que debe ser Linear Layout).

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

tools:context="com.example.delio.clase22listapersonalizada.DescribirAnima
lActivity">

    <ImageView
        android:id="@+id/imageAnimal"
        android:scaleType="fitCenter"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

    <TextView
```

```
        android:id="@+id/txtDescripcion"
        android:layout_width="match_parent"
        android:textSize="25sp"
        android:padding="16dp"
        android:layout_height="wrap_content" />

    <Button
        android:id="@+id/btnRegresar"
        android:text="Regresar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

</LinearLayout>
```

Una vez hecho esto, necesitamos crear la lógica de la segunda actividad:

```
private ImageView imageView;
private TextView textView;
private Button button;
private String [] descrpcionAnimales;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_describir_animal);

    Bundle bundle = getIntent().getExtras();
    imageView = (ImageView) findViewById(R.id.imageAnimal);
    textView  = (TextView) findViewById(R.id.txtDescripcion);
    int pos = bundle.getInt("posicion");

    descrpcionAnimales =
getResources().getStringArray(R.array.descripcion);
    imageView.setImageResource((int) bundle.get("imagen"));

    String describeAnimal = descrpcionAnimales[pos];

    textView.setText(describeAnimal);

    button = (Button) findViewById(R.id.btnRegresar);
    button.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            finish();
        }
    });

}
```

NOTA: Usamos el objeto bundle para recuperar parámetros que servirán para configurar la presentación de la actividad.

Ahora nos toca crear la lógica para la actividad principal, comenzamos por implementar el evento.

```
implements AdapterView.OnItemClickListener
```

Que servirá para llamar a la segunda actividad, donde se ilustra el animal y una breve descripción.

```
private ArrayList<Animal> animales;
private ListView lstAnimales;
private AdapterAnimales adapter;

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

    // Inicializamos las variables.
    animales = new ArrayList<Animal>();

    rellenarArrayList();

    adapter = new AdapterAnimal(this, animales);

    lstAnimales = (ListView) findViewById(R.id.lstAnimales);
    // Asignamos el Adapter al ListView, en este punto
hacemos que el
    // ListView muestre los datos que queremos.
    lstAnimales.setAdapter(adapter);
    // Asignamos el Listener al ListView para cuando pulsamos
sobre uno de
    // sus items.
    lstAnimales.setOnItemClickListener(this);
}
```

Este método carga las imágenes al objeto ArrayList.

```
private void rellenarArrayList() {
    animales.add(new Animal("aguila", R.drawable.aguila));
    animales.add(new Animal("ballena", R.drawable.ballena));
    animales.add(new Animal("caballo", R.drawable.caballo));
    animales.add(new Animal("camaleon",
R.drawable.camaleon));
    animales.add(new Animal("canario", R.drawable.canario));
    animales.add(new Animal("cerdo", R.drawable.cerdo));
    animales.add(new Animal("delfin", R.drawable.delfin));
    animales.add(new Animal("gato", R.drawable.gato));
```

```
    animales.add(new Animal("iguana", R.drawable.iguana));
    animales.add(new Animal("lince", R.drawable.lince));
    animales.add(new Animal("lobo", R.drawable.lobo_9));
    animales.add(new Animal("morena", R.drawable.morena));
    animales.add(new Animal("orca", R.drawable.orca));
    animales.add(new Animal("perro", R.drawable.perro));
    animales.add(new Animal("vaca", R.drawable.vaca));
}
```

Este llamara a la segunda actividad:

```
@Override
public void onItemClick(AdapterView<?> adapterView, View
view, int position, long ID) {
    // Al hacer click sobre uno de los items del ListView
mostramos los
    // datos en los TextView.
    Intent intentDescribir = new Intent(this,
DescribirAnimalActivity.class);
    intentDescribir.putExtra("imagen",
animales.get(position).getImageID());
    intentDescribir.putExtra("posicion", position);
    startActivity(intentDescribir);

}
```