PRACTICA 23 LISTAS PERSONALIZADAS CON VIEWHOLDERS

Comenzamos con la creación del proyecto usando una actividad vacia.

- Cambiamos a Linear Layout
- Agregamos un ListView
-

```
<?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:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"

tools:context="com.example.delio.tema23listviewspersonalizadosconholders.MainActivity">

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

CLASE ANIMAL

Ahora procedemos a crear una clase que represente al objeto que vamos a desplegar en la lista.

```
public class Animal {
    private int imgAnimal;
    private String txtNameAnimal;

    public Animal(int imgAnimal, String txtNameAnimal) {
        this.imgAnimal = imgAnimal;
        this.txtNameAnimal = txtNameAnimal;
    }

    public int getImgAnimal() {
        return imgAnimal;
    }

    public void setImgAnimal(int imgAnimal) {
        this.imgAnimal = imgAnimal;
    }

    public String getTxtNameAnimal() {
        return txtNameAnimal;
    }

    public void setTxtNameAnimal(String txtNameAnimal) {
        this.txtNameAnimal = txtNameAnimal;
    }
}
```

ADAPTADOR Y HOLDER

Creamos un adaptador de datos que usará un objeto holder para optimizar el manejo interno. Veamos el código de inicio:

```
public class AnimalAdapter extends ArrayAdapter<Animal>{

    private final List<Animal> datos;
    private final Context context;

    public AnimalAdapter(Context context, List<Animal>
datos){
        super(context,R.layout.itemlist, datos);
        this.context = context;
        this.datos = datos;
    }

}
```

Ahora agregaremos el patrón ViewHolder que sirve para guardar las referencias de los elementos que se desplegaran:

```
static class ViewHolder {
    protected ImageView imageAnimalHolder;
    protected TextView textPosicionHolder;
    protected TextView textDescripcionHolder;
}
```

NOTA: Debe ser static y protected.

Ahora debemos de configurar la lógica de la vista:

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {

    View view = null;
    if(convertView==null){
        LayoutInflater inflater = LayoutInflater.from(context);
        view = inflater.inflate(R.layout.itemlist,null);
        final ViewHolder viewHolder = new ViewHolder();
        viewHolder.imageAnimalHolder =
            (ImageView) view.findViewById(R.id.imgAnimal);
        viewHolder.textPosicionHolder =
            (TextView) view.findViewById(R.id.txtPosicion);
        viewHolder.textDescripcionHolder =
            (TextView) view.findViewById(R.id.txtAnimal);

        // Guarda la referencia a cada objeto creado (uno por cada
```

```
elemento de la lista)
        view.setTag(viewHolder);

    } else {
        view = convertView;
    }


    ViewHolder holder = (ViewHolder) view.getTag();


holder.imageAnimalHolder.setImageResource(datos.get(position).getImgAnima
l());
    holder.textPosicionHolder.setText(String.valueOf(position));

holder.textDescripcionHolder.setText(datos.get(position).getTxtNameAnimal
());

    return view;
}
```

El manejo de la vista se basa en la clase ViewHolder y solo la primera vez es cuando se carga cada elemento de la vista, ya que si esta creada, solo se recupera, ya que el objeto ViewHolder, almacena cada uno de los elementos que se desplegaran en la ListView.

ACTIVIDAD SEGUNDA

Esta actividad desplegara la ficha técnica del animal en cuestión que fue seleccionado en la actividad principal:

```
<?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:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"

tools:context="com.example.delio.tema23listviewspersonalizadosconholders.
DescribeAnimalActivity">

    <ImageView
        android:id="@+id/imgImagenAnimal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <TextView
        android:id="@+id/txtDescripcion"
        android:padding="10dp"
        android:textSize="20sp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <Button
        android:id="@+id/btnRegresar"
        android:textSize="20dp"
        android:text="@string/regresar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
</LinearLayout>
```

Y la lógica es:

```
public class DescribeAnimalActivity extends AppCompatActivity {

    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_describe_animal);

        Bundle bundle = getIntent().getExtras();
        imageView = (ImageView) findViewById(R.id.imgImagenAnimal);
        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();
            }
        });



    }
}
```

ACTIVIDAD PRINCIPAL

```
<?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:orientation="vertical"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context="com.example.delio.tema23listviewspersonalizadosconholders.
MainActivity">

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

La lógica es la siguiente:

```
private ArrayList<Animal> animales;
private ListView lstAnimales;
private AnimalAdapter 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 AnimalAdapter(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);

    lstAnimales.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> adapterView, View view,
int position, long l) {
            Intent intentDescribir = new Intent(getApplicationContext(),
DescribeAnimalActivity.class);
            intentDescribir.putExtra("imagen",
animales.get(position).getImgAnimal());
            intentDescribir.putExtra("posicion", position);
            startActivity(intentDescribir);
```

```
        }
    });
}


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));
}
```

Con esto terminanos nuestra aplicación con objetos ViewHolder.