DokuWiki

GUÍA DE LABORATORIO NRO.3

OBJETIVO

En esta práctica aprenderemos a manejar el componente ListView, para que a futuro puedas implementarlo en tus aplicaciones.

Componentes de Listado en Android

Los componentes de listado son objetos que permiten mostrar una lista de elementos desplazables. En Android tiene las siguientes componentes:

1. ListView (vista de lista)
2. GridView (Vista de grilla o cuadrícula)
3. Spinner
4. AutoCompleteTextView
5. RecyclerView

Pasos para mostrar datos con ListView

El componente ListView es un control que permite mostrar datos en forma de lista.

Para mostrar datos en un ListView generalmente se sigue los siguientes pasos:

1. Se obtienen los datos (API Rest, BD Interna, etc.) en un List.
2. Se crea un Adapter en base a la Lista obtenida.
3. Se agrega el adapter al componente ListView.

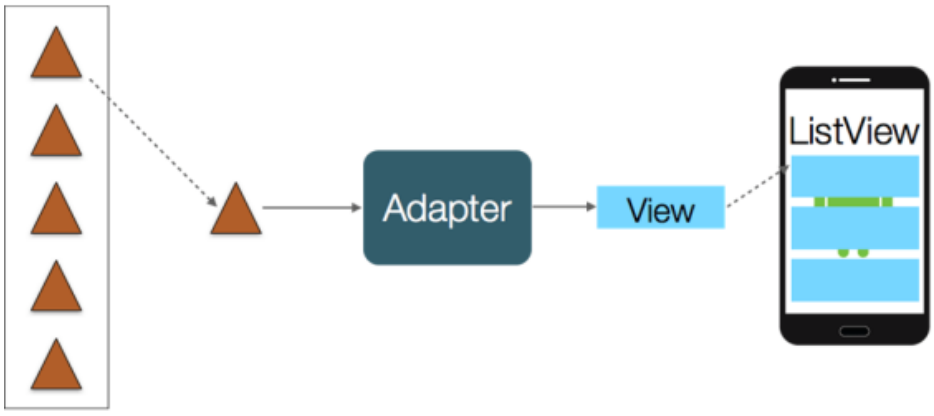


Figure 1: Forma de cómo se carga datos en un ListView

En este ejemplo mostraremos una lista de datos de datos personales.

Comenzamos creando un nuevo Proyecto Android con la plantilla *Empty Views Activity*.

Creando el componente listView

Dentro de activity_main.xml agregamos el componente ListView:

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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=".MainActivity">

    <ListView
        android:id="@+id/listView"
```

```
            android:layout_width="match_parent"
            android:layout_height="match_parent" />

    </RelativeLayout>
```

Creando un Modelo de datos

El modelo de datos que vamos a listar los representaremos en un objeto de tipo Personas (Data Class Kotlin). Esta clase representa cada item de la lista que será enviado al Adapter. *File > New > Kotlin Class File > Data Class*

Persona.kt

```
    data class Persona(
        var nombre: String? = null,
        var apellido: String? = null,
        var telefono: String? = null,
        var imagen: String? = null
    ) {
        constructor(nombre: String, apellido: String, telefono: String) : this(nombre, apellido, telefono, null)

        override fun toString(): String {
            return nombre ?: ""
        }
    }
```

Creando un Adapter

El adapter se lo puede crear de forma personalizada o implementar directamente. Por el momento lo implementaremos directamente instanciando la clase ArrayAdapter:

MainActivity.kt

```
    import androidx.appcompat.app.AppCompatActivity
    import android.os.Bundle
    import android.widget.ArrayAdapter
    import android.widget.ListView

    class MainActivity : AppCompatActivity() {
        override fun onCreate(savedInstanceState: Bundle?) {
            super.onCreate(savedInstanceState)
            setContentView(R.layout.activity_main)


            val personasList = mutableListOf<Persona>(
                Persona("JOSE", "MORALES", "2313526"),
                Persona("MARIA", "FERNANDEZ", "2343551"),
                Persona("ELENA", "PEÑA", "2343559"),
                Persona("ADOLFO", "CHOQUE", "2343557"),
                Persona("RENE", "PEREZ", "2343554"),
                Persona("WENDY", "JIMENEZ", "2343552"),
                Persona("JHONNY", "COLQUE", "2341556")
            )
            val listView = findViewById<ListView>(R.id.listView)
            val arrayAdapter = ArrayAdapter(this, android.R.layout.simple_list_item_1, personasList)
            listView.adapter  = arrayAdapter
        }
    }
```

Al ejecutar nuestra aplicación se podrá visualizar el listview:

Figure 2: ListView en ejecución

Agregando Eventos al ListView

Para obtener el item que ha sido seleccionado de un ListView se debe agregar el método `setOnItemClickListener(OnClickListener)` al objeto que contienen el ListView (`listview`):

```
   listView.setOnItemClickListener { parent, view, position, id ->
            val p = arrayAdapter.getItem(position)
            Toast.makeText(applicationContext, "PERSONA SELECCIONADA: ${p?.nombre}", Toast.LENGTH_SHORT).show()
        }
```

ListView con Selección múltiple

Para dar soporte de selección multiple a un ListView se debe invocar al método: `listView.setChoiceMode(AbsListView.CHOICE_MODE_MULTIPLE);` y usar `android.R.layout.simple_list_item_multiple_choice` como layout del Adapter.

```
listView.choiceMode = AbsListView.CHOICE_MODE_MULTIPLE
val arrayAdapter = ArrayAdapter(applicationContext, android.R.layout.simple_list_item_multiple_choice, personasList)
listView.adapter = arrayAdapter
```

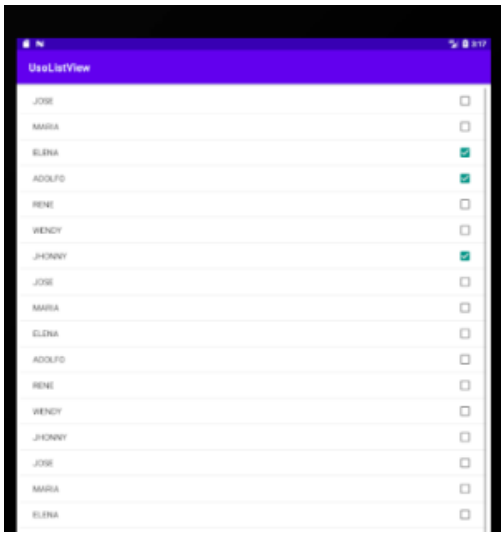


Figure 3: ListView con múltiples selecciones

Para obtener la lista de los items seleccionados se lo puede hacer de la siguiente forma:

```
listView.onItemClickListener = AdapterView.OnItemClickListener { parent, view, position, id ->
            val seleccionados = listView.checkedItemPositions
            for (i in 0 until seleccionados.size()) {
                val checked = seleccionados.valueAt(i)
                if (checked) {
                    val positionSel = seleccionados.keyAt(i)
                    val item = arrayAdapter.getItem(positionSel)
                    Log.i("SELECCIONADOS" ," | ${item?.nombre} | ")

                }
            }
        }
```

ListView con layout y Adapter Personalizado

Un listview puede ser personalizado agregandole un layout y un Adapter personalizado. Para ello se debe agregar un nuevo recurso de tipo layout: `app > res > layout > clic-derecho > New > Layout Resource File`. Este layout los llamaremos (item_view.xml) y lo configuraremos de tal forma que muestre el nombre completo, el numero de teléfono y una imagen de perfil.

item_view.xml

```
<?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"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_weight="10"
    android:orientation="horizontal">

    <ImageView
        android:id="@+id/imagenPerfil"
        android:layout_width="60dp"
        android:layout_height="60dp" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <TextView
            android:id="@+id/txtNombre"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="TextView"
```

```
                android:textAppearance="@style/TextAppearance.AppCompat.Display1" />

            <TextView
                android:id="@+id/txtCelular"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:text="TextView"
                android:textAppearance="@style/TextAppearance.AppCompat.Body2" />
        </LinearLayout>

    </LinearLayout>
```

Para construir un *Adapter* podemos extender a la clase *BaseAdapter* o *ArrayAdapter*. Para nuestro ejemplo lo llamaremos PersonasAdapter y extenderá a ArrayAdapter.

PersonasAdapter.kt

```
import android.content.Context
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ArrayAdapter
import android.widget.ImageView
import android.widget.TextView
import com.bumptech.glide.Glide


class PersonasAdapter(context: Context, personasList: List<Persona>) : ArrayAdapter<Persona>(context, 0, personasList) {
    private val ctx: Context = context

    override fun getView(position: Int, convertView: View?, parent: ViewGroup): View {
        var convertViewLocal = convertView
        if (convertViewLocal == null) {
            convertViewLocal = LayoutInflater.from(context).inflate(R.layout.item_views, parent, false)
        }

        val img = convertViewLocal!!.findViewById<ImageView>(R.id.imagenPerfil)
        val p = getItem(position)
        Glide.with(ctx)
            .load(p?.imagen)
            .centerCrop()
            .placeholder(R.drawable.usuarioicono)
            .into(img)

        val txtNombre = convertViewLocal.findViewById<TextView>(R.id.txtNombre)
        txtNombre.text = "${p?.nombre} ${p?.apellido}"

        val txtCelular = convertViewLocal.findViewById<TextView>(R.id.txtCelular)
        txtCelular.text = p?.telefono

        return convertViewLocal
    }
}
```

Finalmente agregamos PersonasAdapter a nuestro listview que lista personas con su imagen de perfil de la siguiente forma:

```
ListView listView = findViewById(R.id.listView);


val personasAdapter = PersonasAdapter(applicationContext,personasList)
listView.adapter = personasAdapter
```

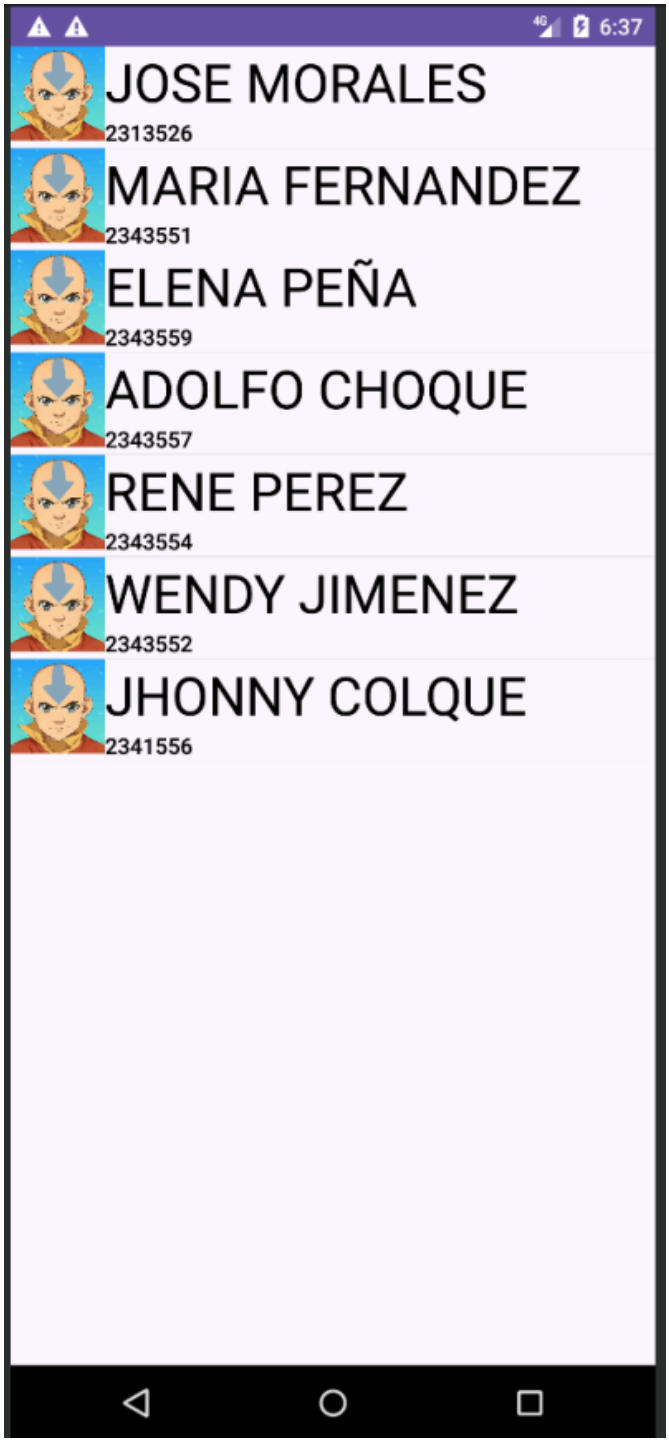
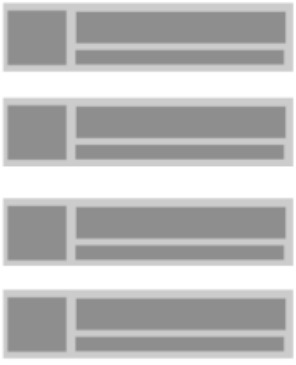


Figure 4: Listview con layout y adapter personalizado

Laboratorio Nro. 03

Crear una App que implemente una lista de contactos (Personas) con un Adapter y Layout personalizado:



Adicionar componentes (botones) que permitan agregar, modificar y eliminar los elementos del ListView. Todos los datos se deben actualizar en la lista asociado al ListView.

materias/tecnologias-emergentes/unidad_1/04_listview_kotlin.txt · Last modified: 2024/03/20 22:38 by teoteo