

Documentación Actividad 08

ListView

Desarrollo móvil integral

Jose Manuel Bautista Morales

Docente: ING. Abel Jerónimo Vargas

10 de noviembre de 2023

9.º GRADO

ÍNDICE

[ÍNDICE](#)

[INTRODUCCIÓN](#)

[DESARROLLO](#)

[ListView](#)

[Vista XML](#)

[Clase Java \(main\)](#)

[EJECUCIÓN](#)

[Inicio](#)

[CONCLUSIÓN](#)

INTRODUCCIÓN

En el vasto panorama del desarrollo de aplicaciones Android, uno de los componentes más fundamentales y versátiles que nos ofrece el kit de herramientas es el `ListView`. Este componente esencial facilita la presentación eficiente y ordenada de datos en forma de listas, ofreciendo a los desarrolladores una herramienta poderosa para crear interfaces de usuario dinámicas y atractivas.

DESARROLLO

`ListView`

- ¿Qué es un `ListView`?:

En términos sencillos, un `ListView` es un contenedor que permite mostrar una lista de elementos desplazables verticalmente. Este componente se adapta perfectamente a una amplia gama de aplicaciones, desde simples listas de contactos hasta complejas interfaces de usuario que presentan información detallada de manera estructurada.

Vista XML

Esta vista XML representa la interfaz de usuario de una aplicación básica de Android que incluye un `ListView` y elementos relacionados para realizar operaciones de suma.

- **RelativeLayout Principal:**

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp"
    tools:context=".MainActivity">
```

- Este es el contenedor principal que alberga todos los elementos de la interfaz de usuario.

- `android:padding="16dp"` proporciona un espacio de relleno uniforme alrededor del contenido del `RelativeLayout`.
- **TextView (textViewTitle):**

```
<TextView
    android:id="@+id/textViewTitle"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Mi ListView Basico"
    android:textSize="24sp"
    android:textColor="#FFFFFF"
    android:gravity="center"
    android:layout_marginBottom="16dp"/>
```

- Un `TextView` que actúa como el título de la aplicación.
- `android:layout_width="match_parent"` y `android:layout_height="wrap_content"` aseguran que ocupe todo el ancho y se ajuste al contenido en altura.
- `android:text="Mi ListView Basico"` establece el texto del título.
- `android:textSize="24sp"` define el tamaño del texto.
- `android:textColor="#FFFFFF"` establece el color del texto a blanco.
- `android:gravity="center"` centra el texto horizontalmente.
- `android:layout_marginBottom="16dp"` agrega un margen en la parte inferior del título.
- **ListView (listView):**

```
<ListView
    android:id="@+id/listView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_below="@id/textViewTitle"
    android:layout_above="@+id/layoutBottom"
    android:background="#396f33"/>
```

- El `ListView` que mostrará una lista de elementos, como el historial de sumas.
- `android:layout_below="@id/textViewTitle"` coloca el `ListView` debajo del título.
- `android:layout_above="@+id/layoutBottom"` asegura que el `ListView` no se extienda hasta el `layoutBottom`.

- `android:background="#396f33"` establece un fondo verde para el `ListView`.
- **RelativeLayout (layoutBottom):**

```
<RelativeLayout
    android:id="@+id/layoutBottom"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true">
```

-
- Un segundo `RelativeLayout` que contiene elementos de entrada (`EditText`) y un botón.
- `android:layout_alignParentBottom="true"` posiciona este `RelativeLayout` en la parte inferior del `RelativeLayout` principal.
- **EditTexts (tfNum1 y tfNum2):**

```
<EditText
    android:id="@+id/tfNum1"
    android:layout_width="100dp"
    android:layout_height="65dp"
    android:layout_alignParentStart="true"
    android:layout_marginEnd="8dp"
    android:inputType="number"
    android:hint="Número 1"/>

<EditText
    android:id="@+id/tfNum2"
    android:layout_width="100dp"
    android:layout_height="67dp"
    android:layout_toEndOf="@+id/tfNum1"
    android:layout_marginStart="8dp"
    android:inputType="number"
    android:hint="Número 2"/>
```

- Dos `EditText` que permiten al usuario ingresar números para realizar la operación de suma.
- `android:id` asigna identificadores únicos a estos elementos.
- `android:layout_width` y `android:layout_height` establecen las dimensiones.
- `android:layout_alignParentStart="true"` y `android:layout_toEndOf="@+id/tfNum1"` controlan su posición relativa.

- **Button (btnCalcular):**

```
<Button
    android:id="@+id/btnCalcular"
    android:layout_width="130dp"
    android:layout_height="70dp"
    android:layout_alignParentEnd="true"
    android:layout_marginTop="8dp"
    android:background="#00FF00"
    android:text="Sumar"/>
```

- Un **Button** con el texto "Sumar" que inicia la operación de suma.
- **android:id** asigna un identificador único.
- **android:layout_width** y **android:layout_height** establecen las dimensiones.
- **android:layout_alignParentEnd="true"** coloca el botón en la esquina superior derecha.
- **android:background="#00FF00"** establece el fondo del botón en verde.

Clase Java (main)

Clase Java que representa la lógica de la aplicación(la suma).

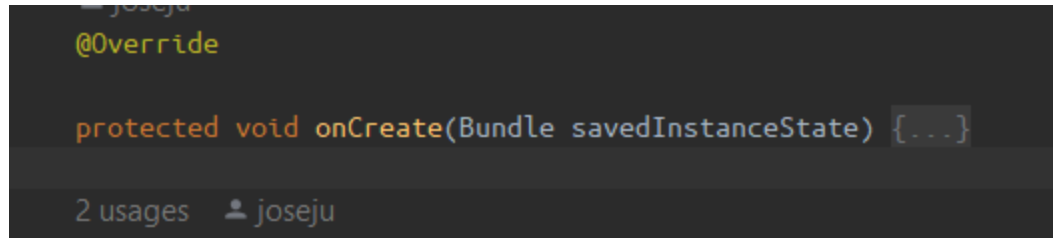
1. Declaración de Variables:

```
3 usages
private EditText numero1, numero2;
2 usages
private Button calcular;
3 usages
private ListView listView;
3 usages
private ArrayList<String> resultList;
4 usages
private ArrayAdapter<String> adapter;
```

- **numero1, numero2:** Objetos **EditText** para capturar los números ingresados por el usuario.
- **calcular:** Objeto **Button** que activa la función de cálculo.
- **listView:** Objeto **ListView** que muestra el historial de operaciones.
- **resultList:** Lista que almacena las operaciones realizadas.

- **adapter**: Objeto **ArrayAdapter** que actúa como puente entre la lista (**resultList**) y el **ListView**.

2. **onCreate(Bundle savedInstanceState):**



```

joseju
@Override

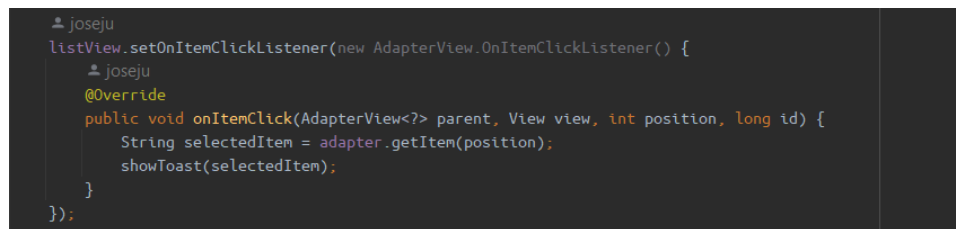
protected void onCreate(Bundle savedInstanceState) {...}

2 usages  joseju

```

- Método llamado cuando la actividad se crea.
- **setContentView(R.layout.activity_main)**: Asocia la actividad con su diseño XML.
- Asigna los elementos de la interfaz de usuario a las variables correspondientes.
- Crea un nuevo **ArrayAdapter** y lo asigna al **ListView**.

3. **listView.setOnItemClickListener:**



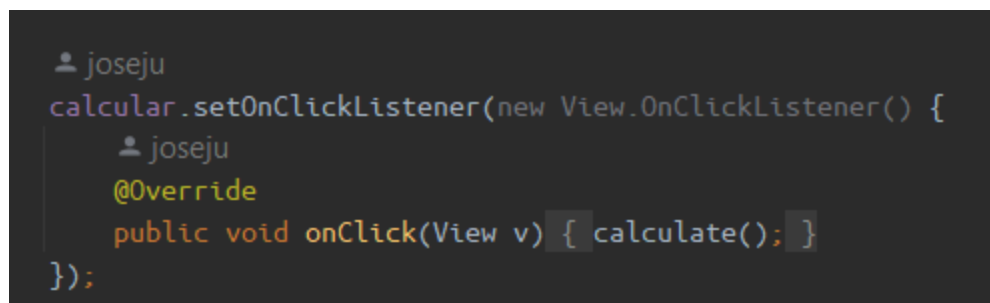
```

joseju
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    joseju
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        String selectedItem = adapter.getItem(position);
        showToast(selectedItem);
    }
});

```

- Establece un escuchador de clics en los elementos del **ListView**.
- Muestra un mensaje emergente (**Toast**) con el elemento seleccionado.

4. **calcular.setOnClickListener:**



```

joseju
calcular.setOnClickListener(new View.OnClickListener() {
    joseju
    @Override
    public void onClick(View v) { calculate(); }
});

```

- Establece un escuchador de clics para el botón "Sumar".
- Llama al método **calculate** cuando se hace clic en el botón.

5. **showToast(String message):**

```
2 usages  joseju
private void showToast(String message) {
    Toast.makeText( context: this, message, Toast.LENGTH_SHORT).show();
}
```

- Muestra un mensaje emergente (**Toast**) con el mensaje proporcionado.

6. **calculate():**

```
1 usage  joseju
private void calculate() {
    String num1Str = numero1.getText().toString();
    String num2Str = numero2.getText().toString();

    if (!num1Str.isEmpty() && !num2Str.isEmpty()) {
        double num1 = Double.parseDouble(num1Str);
        double num2 = Double.parseDouble(num2Str);
        double result = num1 + num2;

        String calculation = num1Str + " + " + num2Str + " = " + result;
        resultList.add(calculation);
        adapter.notifyDataSetChanged();

        numero1.setText("");
        numero2.setText("");

        hideKeyboard();
        showToast("Suma realizada Correctamente.");
    }
}
```

- Obtiene los valores ingresados por el usuario.
- Verifica que ambos campos no estén vacíos.
- Realiza la suma y muestra el resultado en el **ListView**.
- Limpia los campos de entrada y oculta el teclado virtual.
- Muestra un mensaje emergente confirmando que la suma se realizó correctamente.

7. `hideKeyboard()`:

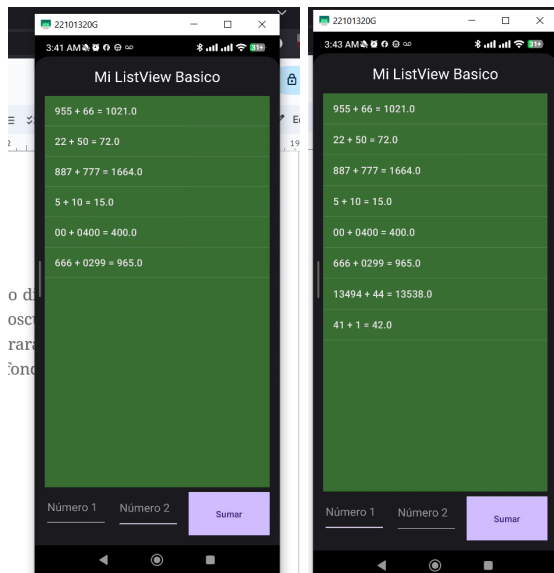
```
1 usage  joseju
private void hideKeyboard() {
    View view = this.getCurrentFocus();
    if (view != null) {
        InputMethodManager imm = (InputMethodManager) getSystemService(Context.INPUT_METHOD_SERVICE);
        imm.hideSoftInputFromWindow(view.getWindowToken(), flags);
    }
}
```

- Oculta el teclado virtual cuando se llama.

EJECUCIÓN

Inicio

Lo primero que se notará al abrir la aplicación es el encabezado distintivo con el título "Mi ListView Basico". Este encabezado, ubicado sobre un fondo oscuro, brinda un toque visualmente atractivo a nuestra aplicación. Justo debajo, encontrarán un `ListView` que mostrará todas las sumas realizadas hasta el momento, con un fondo verde que lo hace destacar.



CONCLUSIÓN

La creación de una aplicación de suma en Android, combinando XML y Java, proporciona una introducción práctica al desarrollo de aplicaciones móviles. A través del diseño de la interfaz con elementos como `ListView`, `EditText`, y `Button`, se facilita la interacción del usuario.