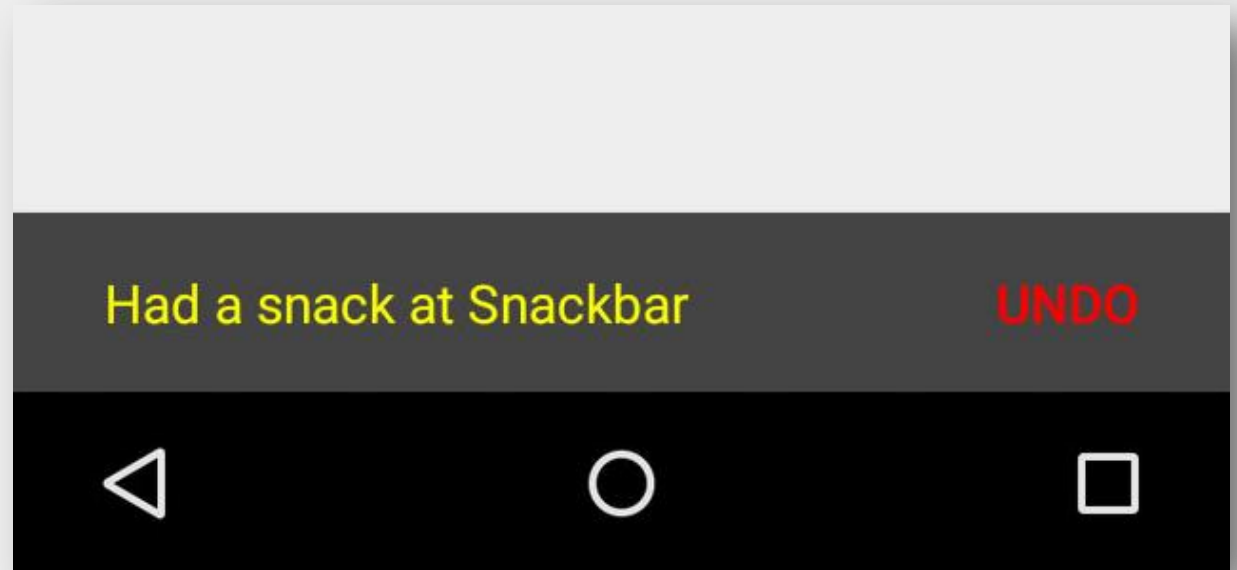




Interfaces de usuario avanzadas

# SnackBar



- En **build.gradle** (Module:app):

compile 'com.android.support:design:24.2.1'

- Nota: El número de versión puede variar.

# Mostrar SnackBar

- SnackBar básica:

```
Snackbar.make(findViewById(R.id.activity_main), "Mensaje",  
Snackbar.LENGTH_LONG).show();
```

- SnackBar con acción:

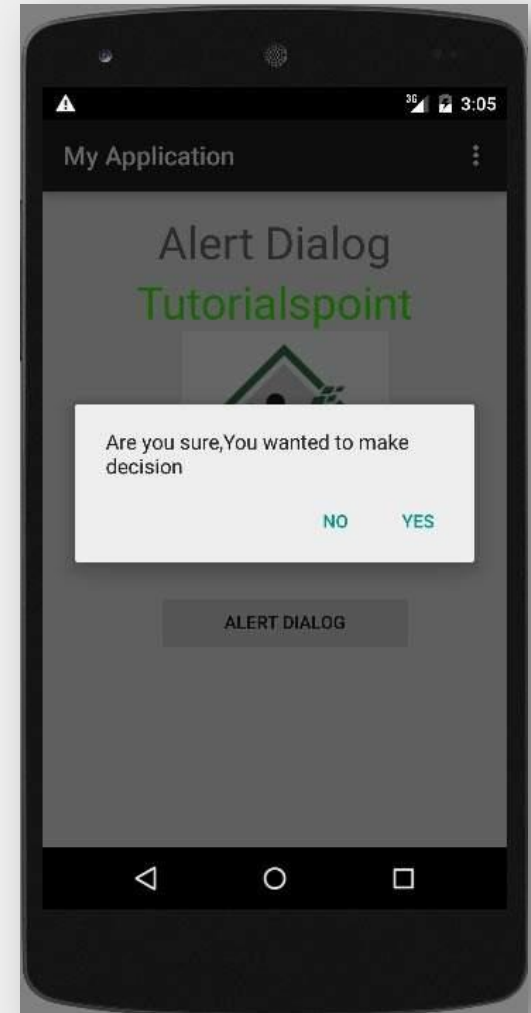
```
Snackbar.make(findViewById(R.id.activity_main), "Mensaje",  
Snackbar.LENGTH_LONG)  
    .setAction("Acción", new View.OnClickListener() {  
        @Override  
        public void onClick(View view) {  
            // Si no ponemos nada simplemente la SnackBar se oculta  
        }  
    }).show();
```

# Diálogo

- Heredar de **DialogFragment**:
  - Crear método **newInstance**
  - Sobrecribir **onCreateDialog**:
    - Título
    - Mensaje
    - Botón positivo
    - Botón negativo

**IMPORTANTE:** Para poder usar diálogos dentro de un Fragment hay que importar:

```
import android.app.DialogFragment;
```



# Clase Dialogo

```
public class Dialogo extends DialogFragment {
    public static Dialogo newInstance() {
        Dialogo fragment = new Dialogo();

        return fragment;
    }
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
        builder.setTitle("Título");
        builder.setMessage("Mensaje");
        builder.setPositiveButton("Botón +", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                // Hacer algo
            }
        });
        builder.setNegativeButton("Botón -", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                // Hacer algo o dejarlo vacío para cerrar el diálogo sin hacer nada
            }
        });
        return builder.create();
    }
}
```

# Mostrar diálogo

- En Activity o Fragment:

```
DialogFragment dialogFragment = Dialogo.newInstance();  
dialogFragment.show(getFragmentManager(),"Dialogo");
```

# ProgressBar

- En layout (ProgressBar Circular):

```
<ProgressBar  
    android:id="@+id/progressBarCircular"  
    style="?android:attr/progressBarStyle"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content" />
```

- En layout (ProgressBar Horizontal):

```
<ProgressBar  
    android:id="@+id/progressBarHorizontal"  
    style="?android:attr/progressBarStyleHorizontal"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content" />
```

# ProgressBar

- Obtener ProgressBar:

**ProgressBar** *progressBar*;

*progressBar* = (**ProgressBar**) findViewById(R.id.*progressBarHorizontal*);

- Cambiar progreso:

*progressBar*.setProgress(*progressBar*.getMax() / 2);

- Mostrar / ocultar:

*progressBar*.setVisibility(View.*VISIBLE*);

*progressBar*.setVisibility(View.*GONE*);



# Animaciones

- Crear *res/anim/nombre\_animacion.xml*:

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

```
<set xmlns:android="http://schemas.android.com/apk/res/android">
```

```
  <alpha
```

```
    android:fromAlpha="0"
```

```
    android:toAlpha="1"
```

```
    android:duration="1000">
```

```
  </alpha>
```

```
</set>
```

# Animaciones

- Obtener animación:

Animation animation = AnimationUtils.loadAnimation(**this**, R.anim.*animacion*);

- Asignar animación a una vista:

*vista*.setAnimation(animation);

- Iniciar animación:

*vista*.getAnimation().start();

# Más ejemplos de animaciones

## <translate

```
android:fromXDelta="0%p"  
android:toXDelta="75%p"  
android:duration="800"  
android:startOffset="4000"  
android:interpolator="@android:anim/accelerate_interpolator"/>
```

## <rotate

```
android:fromDegrees="0"  
android:toDegrees="360"  
android:pivotX="50%"  
android:pivotY="50%"  
android:duration="4000" >
```

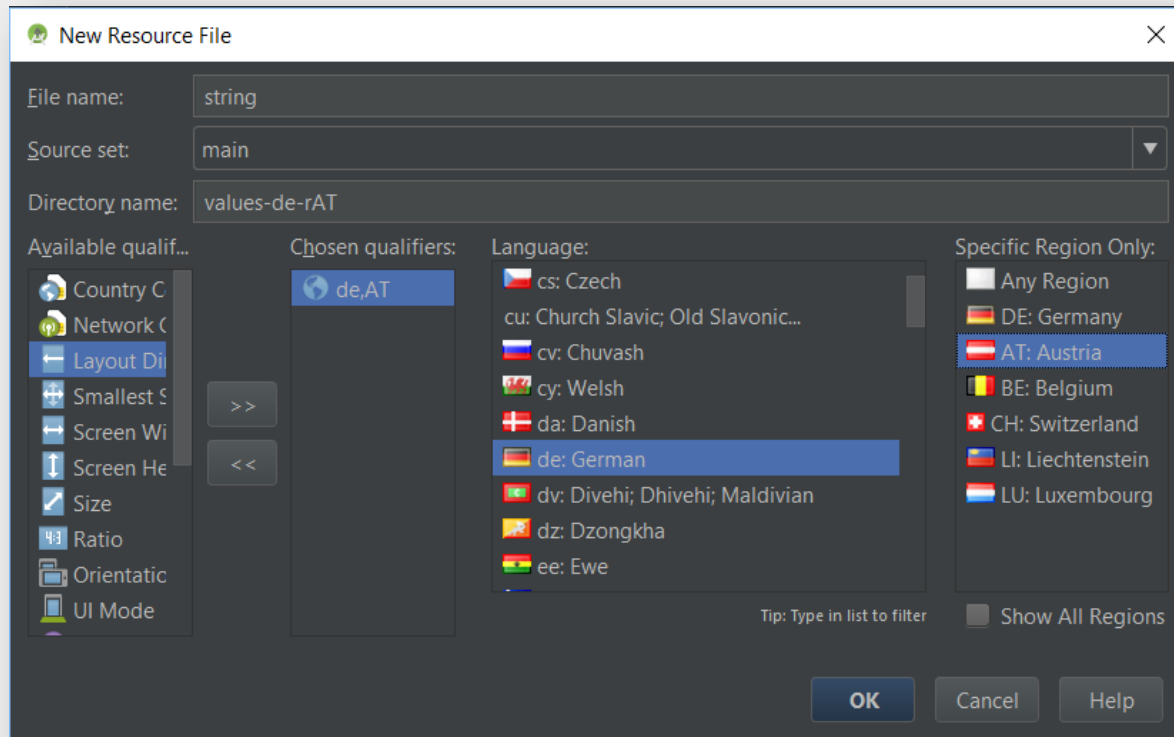
## </rotate>

## <scale

```
android:duration="500"  
android:fromXScale="1.0"  
android:fromYScale="1.0"  
android:interpolator="@android:anim/linear_interpolator"  
android:toXScale="1.0"  
android:toYScale="0.0"  
android:repeatCount="3"/>
```

# Internacionalización (i18n)

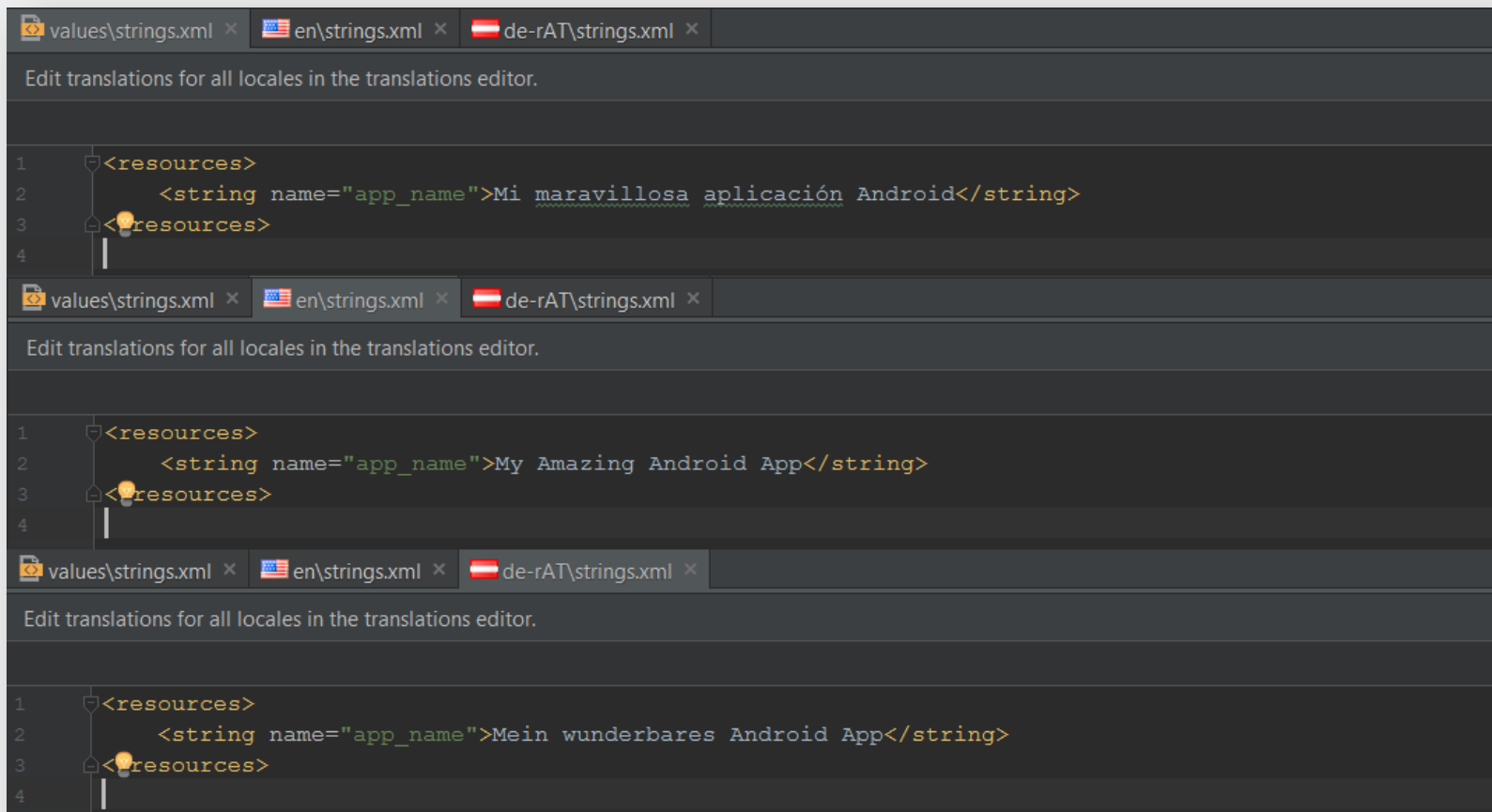
- Crear fichero **strings.xml** con código del idioma:



Nota: Si el idioma no está se usará el string.xml por defecto

# Internacionalización (i18n)

- Rellenar los textos en los idiomas soportados:



The image displays three sequential screenshots of the Android Studio 'translations editor' interface, illustrating the process of internationalizing an application by adding German text for the 'de-rAT' locale.

**Top Screenshot:** The editor shows the initial state with the English string: `<string name="app_name">Mi maravillosa aplicación Android</string>`. The tabs at the top are `values\strings.xml`, `en\strings.xml`, and `de-rAT\strings.xml`. The interface includes a header 'Edit translations for all locales in the translations editor.' and a tree view on the left.

**Middle Screenshot:** The German text 'My Amazing Android App' has been entered for the 'de-rAT' locale. The string in the editor is now: `<string name="app_name">My Amazing Android App</string>`. The tabs and header remain the same.

**Bottom Screenshot:** The German text 'Mein wunderbares Android App' has been entered for the 'de-rAT' locale. The string in the editor is now: `<string name="app_name">Mein wunderbares Android App</string>`. The tabs and header remain the same.