# INDICE



## 1.1. Introducción

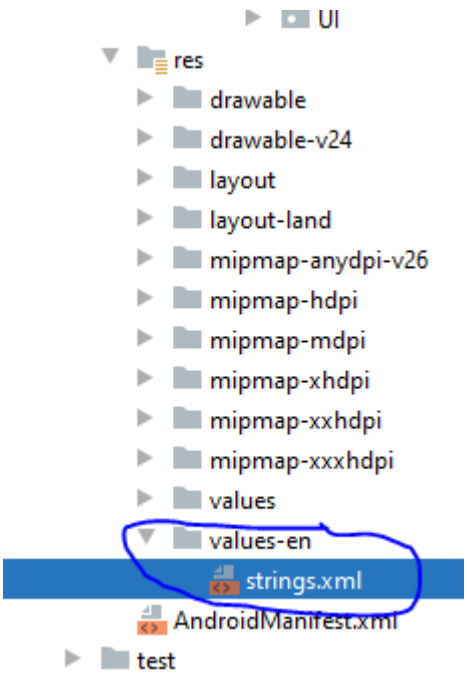
- ✓ También tenemos otras formas de *personalizar* nuestras aplicaciones en función de una serie de sufijos.
- ✓ Si queremos que las cadenas constantes se traduzcan a varios idiomas, solo necesitamos que crear una nueva carpeta llamada **values- ??** donde ?? y el idioma.
- ✓ Para inglés tendríamos **/res/values-en**, **/res/values-es** para español, ....
- ✓ El idioma predeterminado será el almacenado en / res / values.
- ✓ Solo se deben cambiar los valores de las constantes, los identificadores deben ser los mismos.

- ✓ **Referencias:**
  - **Localización**: https://developer.android.com/guide/topics/resources/localization.html

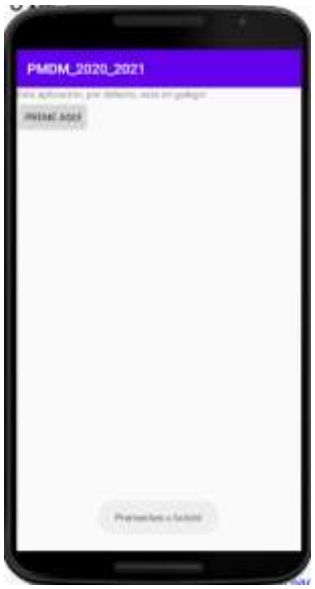
## 1.2. Caso práctico

- ✓ Crea un nuevo proyecto: **U2_41_International**
- ✓ Vayamos a **/res/values** definimos las cadenas de texto en gallego y definimos esas mismas cadenas en inglés en **/res/values-en**.

## Internacionalización




Carpetas de recursos: values e values-en. Cada uno con su fichero de cadena.




La aplicación está en gallego, tanto no layout como en código (Toast).


Y lo mismo ocurre con el inglés. El layout y el código Java es el mismo para ambos casos.

### 1.2.1.strings.xml de /res/values/

```
<resources>
    <string name="action_settings">Settings</string>
    <string name="idioma">Esta aplicación, por defecto, está en galego!</string>
    <string name="boton">Preme aquí</string>
    <string name="mensaxe_toast">Premeches o botón!</string>
</resources>
```

### 1.2.2.strings.xml de /res/values-en

```
<resources>
    <string name="app_name">U2_41_Internacional</string>
    <string name="action_settings">Settings</string>
    <string name="idioma">This app is by default in galician language!</string>
    <string name="boton">Click here</string>
    <string name="mensaxe_toast">You have clicked on the button!</string>
</resources>
```

### 1.2.3.El XML del layout

```
<?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"
    android:orientation="vertical"
    tools:context=".U2_41_Internacional.U2_41_Internacional">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
            android:text="@string/idioma" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/boton"
            android:onClick="onBotonClick" />

    </LinearLayout>
```

**Líneas 10 y 15**: Ahora tenemos que trabajar con constantes, no podemos poner el texto directamente.

### 1.2.4.El código Java

```
package olga100.proyectosandroid.pmdm_2020_2021.U2_41_Internacional;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

import olga100.proyectosandroid.pmdm_2020_2021.R;

public class U2_41_Internacional extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_u2_41__internacional);
    }


    public void onBotonClick(View v) {
        Toast.makeText(this, R.string.mensaxe_toast, Toast.LENGTH_SHORT).show();
    }
}
```

**Línea 25:** observar como el toast también coge la cadena de texto de los recursos tipo string.

## 1.3. Consideraciones a tener en cuenta

- ✓ Cuando queremos utilizar un texto con una función que no acepta un recurso de clase R de tipo R.string.resource_name, debemos hacer uso de la función getResources().getString("R.string.resource_name") la cual devolvería una cadena (String) en el idioma adecuado.

  A través de esta función (getResources ()) podemos recuperar, llamando a diferentes métodos, todo tipo de recursos almacenados en /res/.

Nota: La función getString ("R.string.resource_name") también podría usarse directamente.

Nota: Recordar que hasta el momento las Activities creadas tienen una etiqueta 'puesto a mano' por nosotros en el archivo **AndroidManifiest.xml**.

Deben estar referenciados en un archivo externo de tipo 'values' => 'string'. Recordar cambiarlo en tu proyecto (solo debe llevar una etiqueta a nivel de proyecto, no a nivel de activity).

Estos sufijos que estamos usando (-land; -es) se pueden combinar entre sí y con otros muchos diferentes.

Este orden de prefijos aparece en el siguiente enlace.

Por ejemplo:

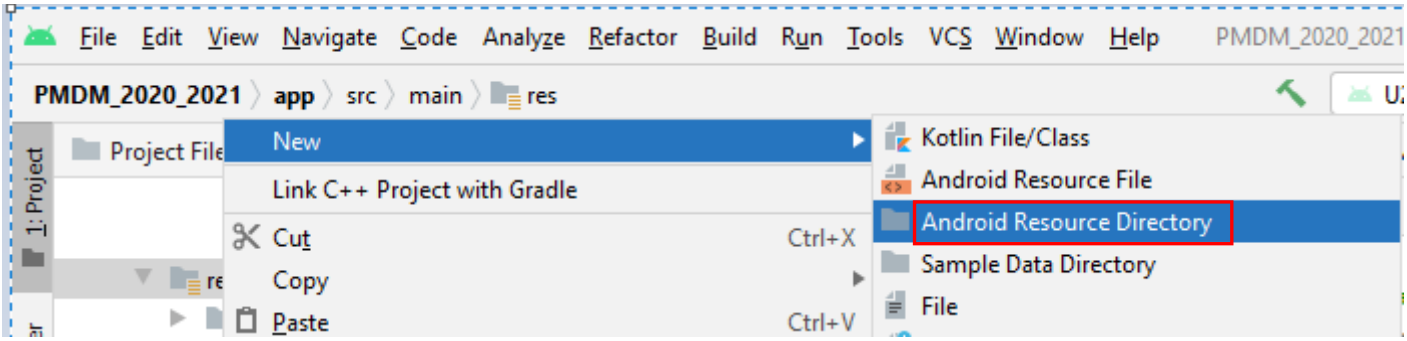
- ✓ res / values-en-rUS / strings.xml
- ✓ res / values-en-rUK / strings.xml

Donde el segundo sufijo se refiere al inglés "americano" (en-rUS) o al "inglés" del Reino Unido (en-rUK).

Nota: La letra r se refiere a la región.

Para facilitar nuestro trabajo, Android Studio cuenta con una pantalla donde podemos indicar qué 'características' queremos que tenga nuestro archivo de recursos y agregar automáticamente los sufijos necesarios para nosotros.

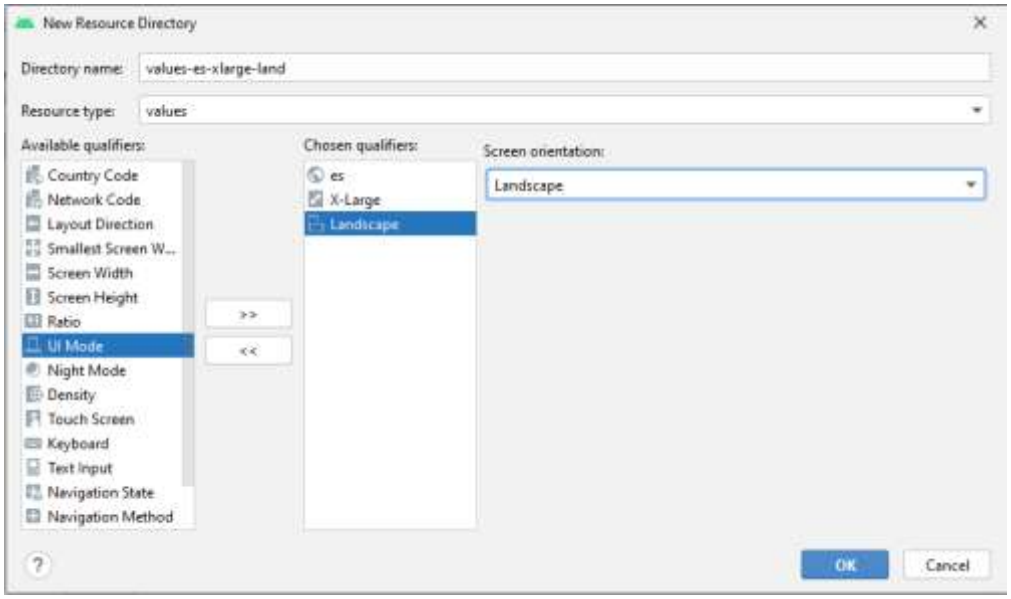
**Creando un nuevo archivo de recursos**

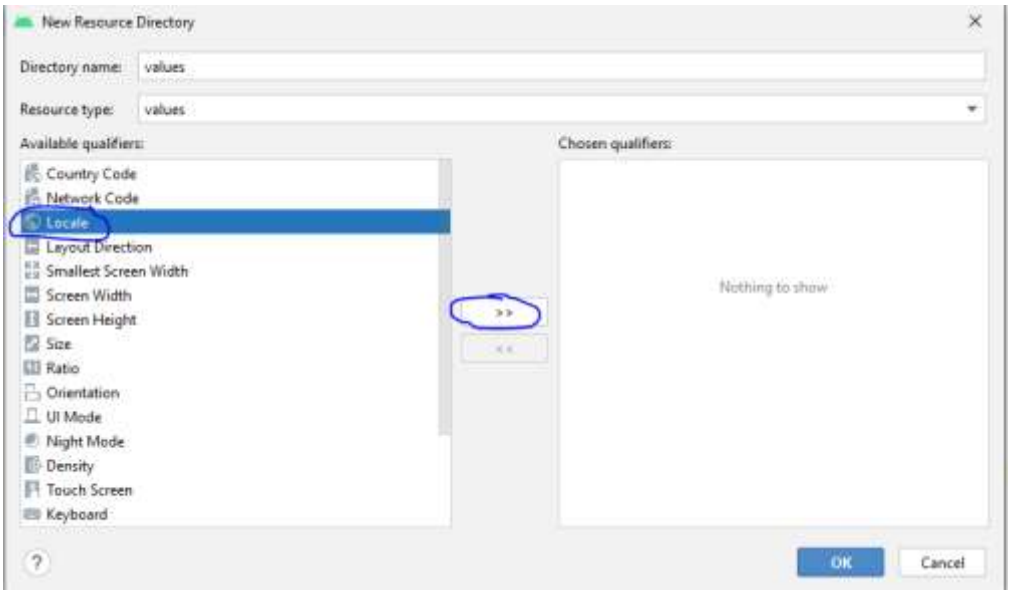



Hacer clic derecho en la carpeta **/res/** o **/res /values /**. Si lo hacemos sobre **/res/** aparecerá un combo extra en el que tendremos que elegir el tipo *'values'*.

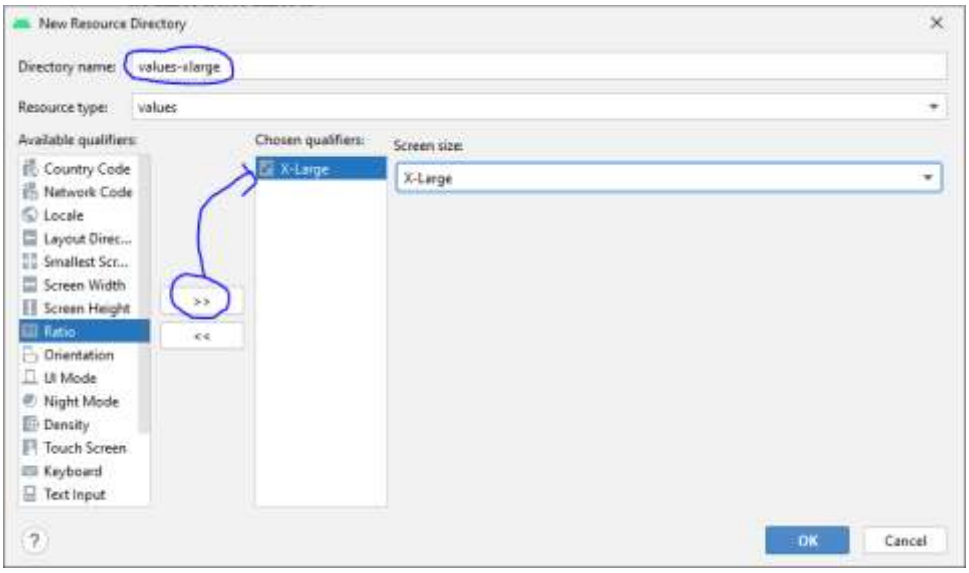
Crearemos un recurso (archivo xml) cuando el dispositivo sea x-large, el idioma este en español y el dispositivo tenga una orien-

tación horizontal.



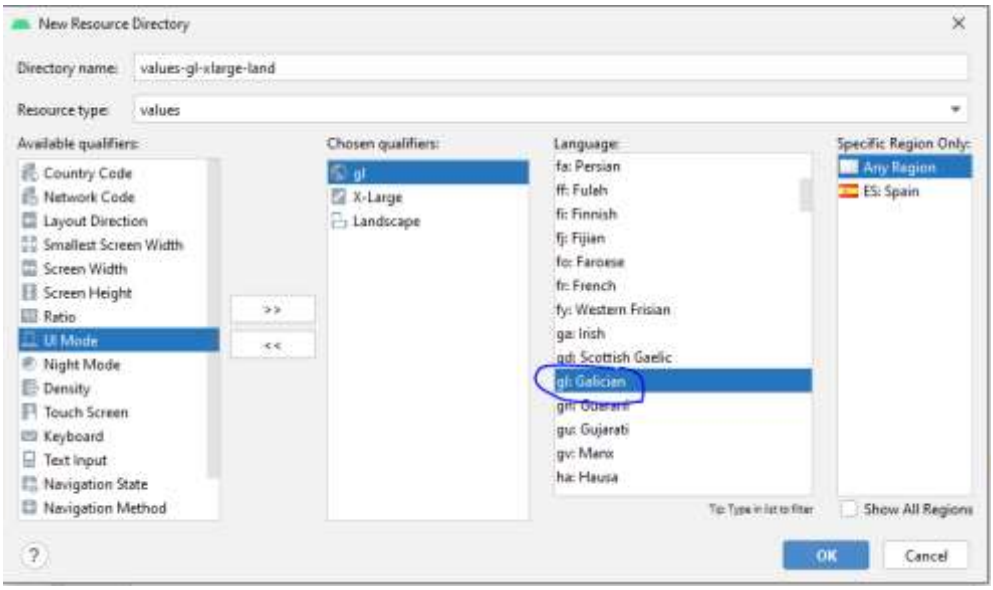




Debemos pasar la propiedad 'SIZE' desde la lista de la izquierda a la derecha (haciendo clic en el botón con la dirección derecha).




En este ejemplo, estamos creando un recurso de tipo de layout para pantallas de 7 a 10 pulgadas (tamaño x-large), por lo que tendríamos que crear un layout en una carpeta "layout-xlarge". Al hacerlo, puede ver cómo aparece el nombre con el sufijo correcto en la parte inferior. Además, se creará una carpeta física con el sufijo.

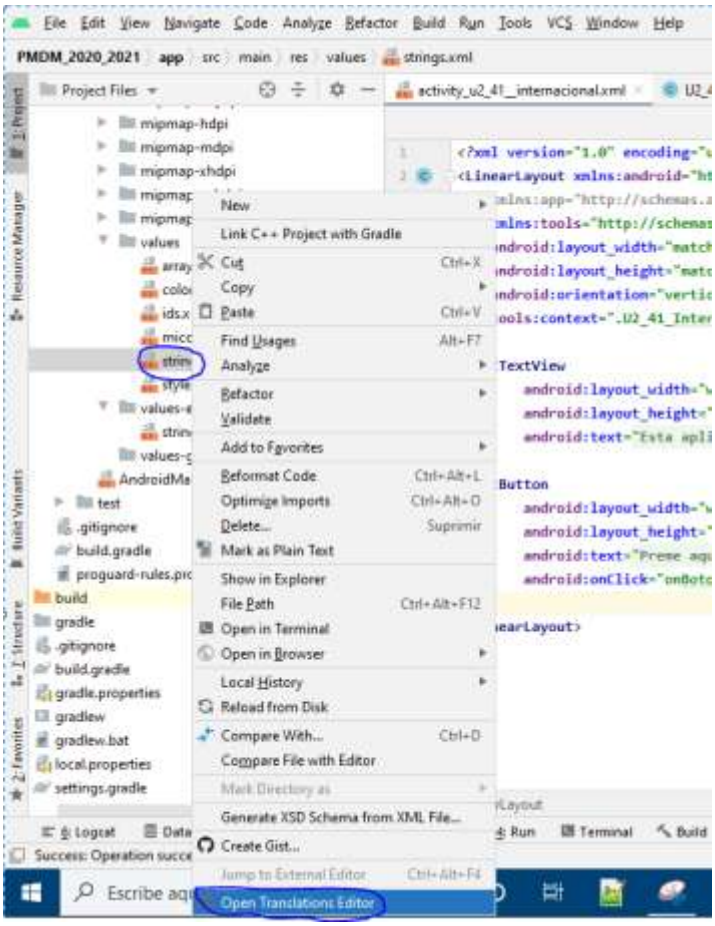
✓ A partir de la versión Lollipop (API 21, android 5.0) ya incluye el idioma gallego.

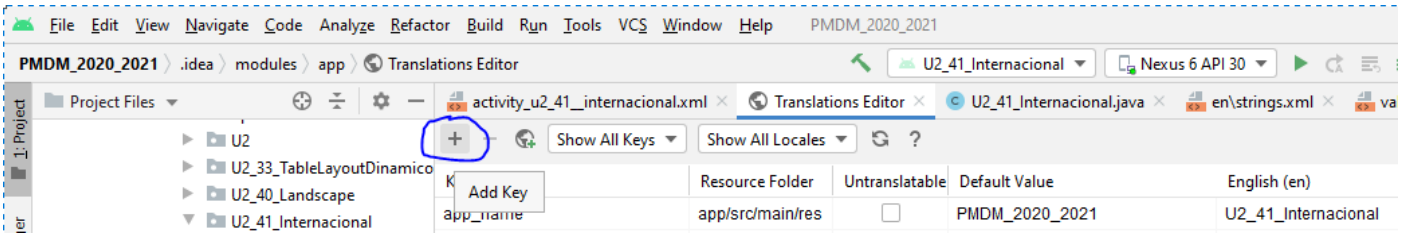



✓ También tenemos la opción de crear estos archivos de idioma directamente en un editor.

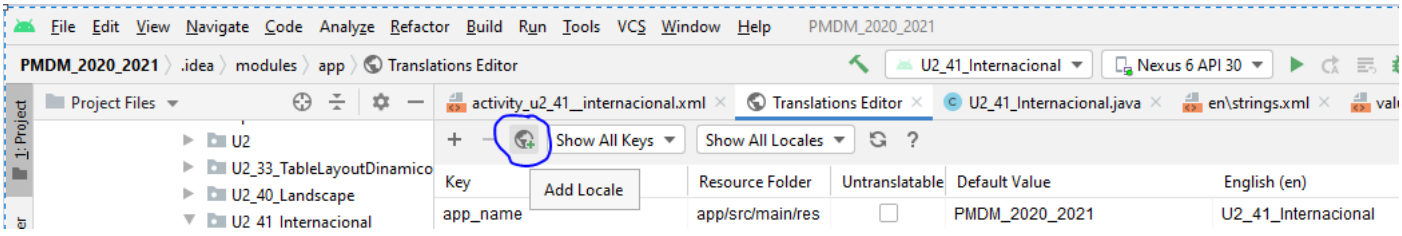
A través de esta herramienta podemos editar cualquier archivo en **/res/values** y asociar gráficamente diferentes idiomas e ingresar su traducción:

## Creando nuevos archivos de idiomas con Translation Editor

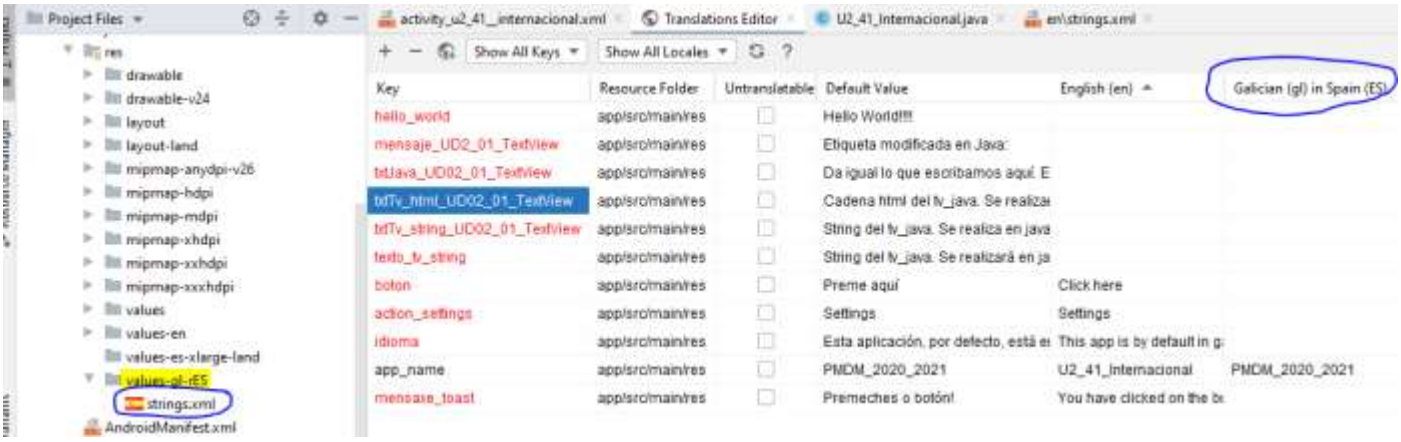



Al hacer clic en el botón **+** creamos una nueva entrada en el archivo XML.




Si queremos crear un nuevo archivo con un idioma diferente debemos hacer clic en el botón **Add Locale**.




Una vez agregado el idioma, aparecerá una nueva columna para cada una de las entradas en el archivo. El archivo se creará en la carpeta del idioma agregado y podemos indicar si alguna de las entradas no tiene traducción (opción Untranslatable) lo que indicará que esta entrada tomará el valor del archivo predeterminado en cualquier idioma que estemos.