Clase 3: Recursos de diseño

Orientación del dispositivo

A través del **AndroidManifest.xml podremos barajar el comportamiento de los Activity**.

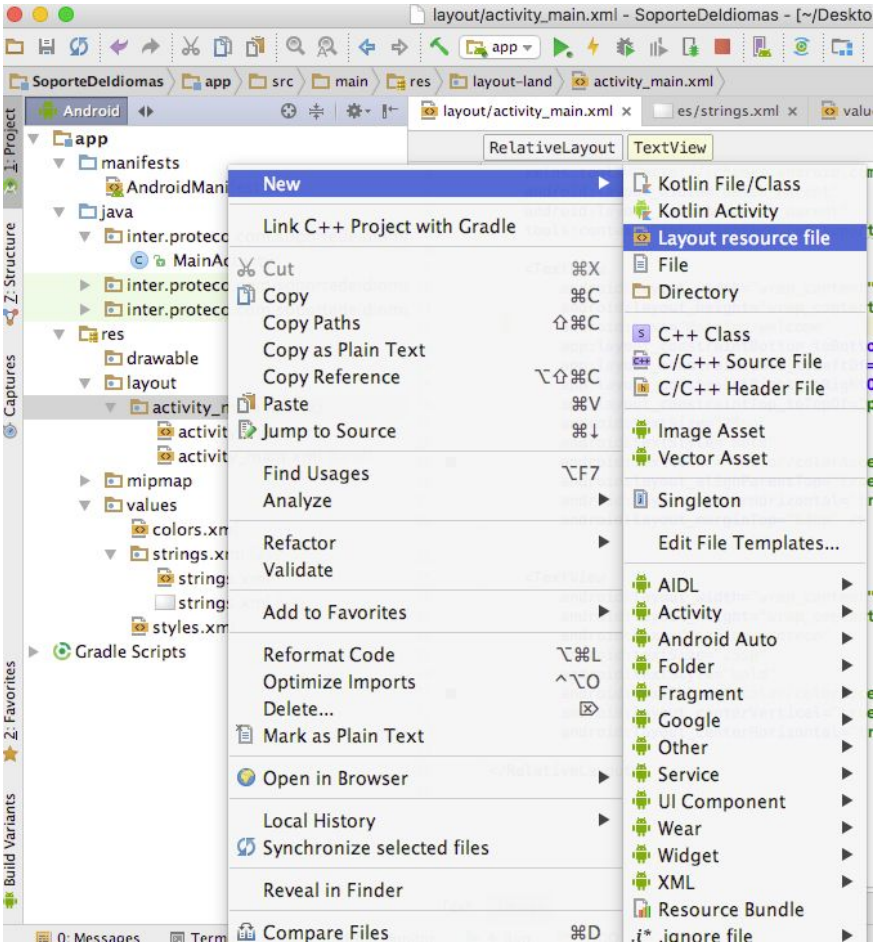
```
<activity
   android:name="android:name=".MainActivity"
   android:label="@string/song_detail"
   android:screenOrientation="portrait">
</activity>
```

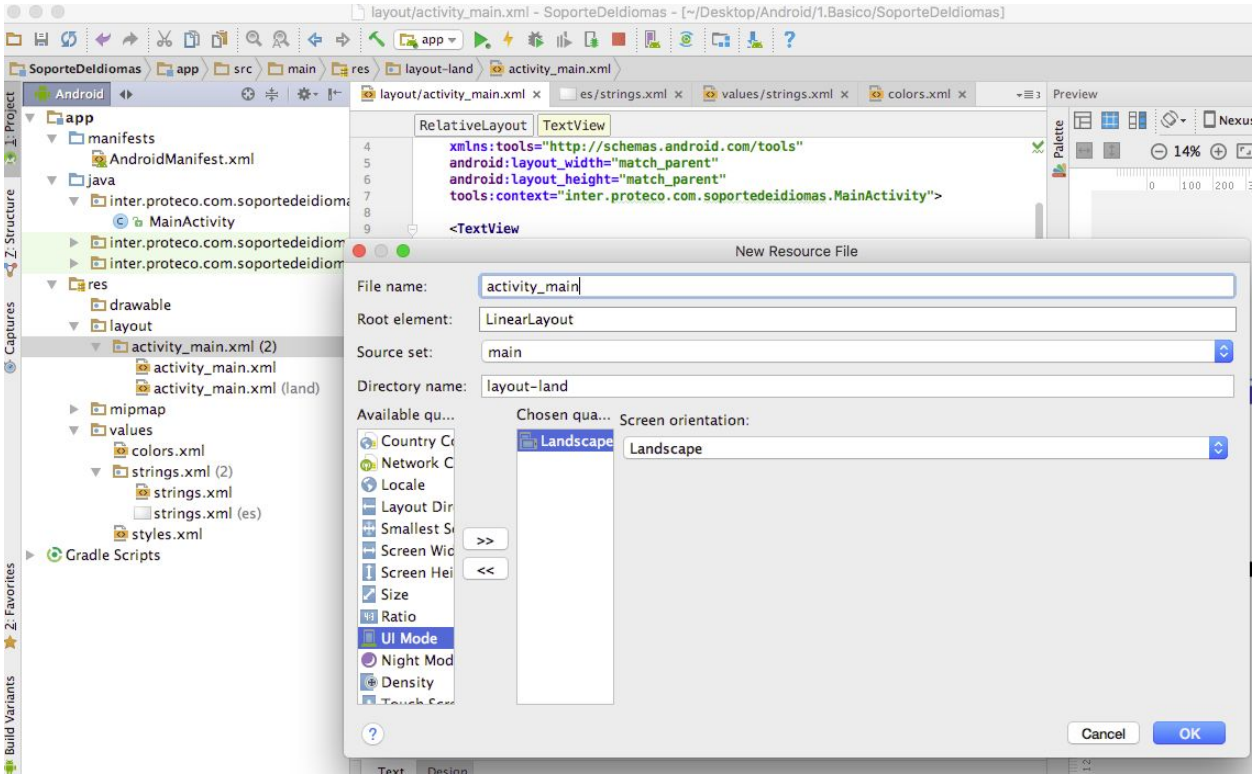
android:screenOrientation="portrait //Solo vertical (evito que se vea en horizontal)
android:screenOrientation="landscape" //Solo horizontal (evito que se vea en vertical)

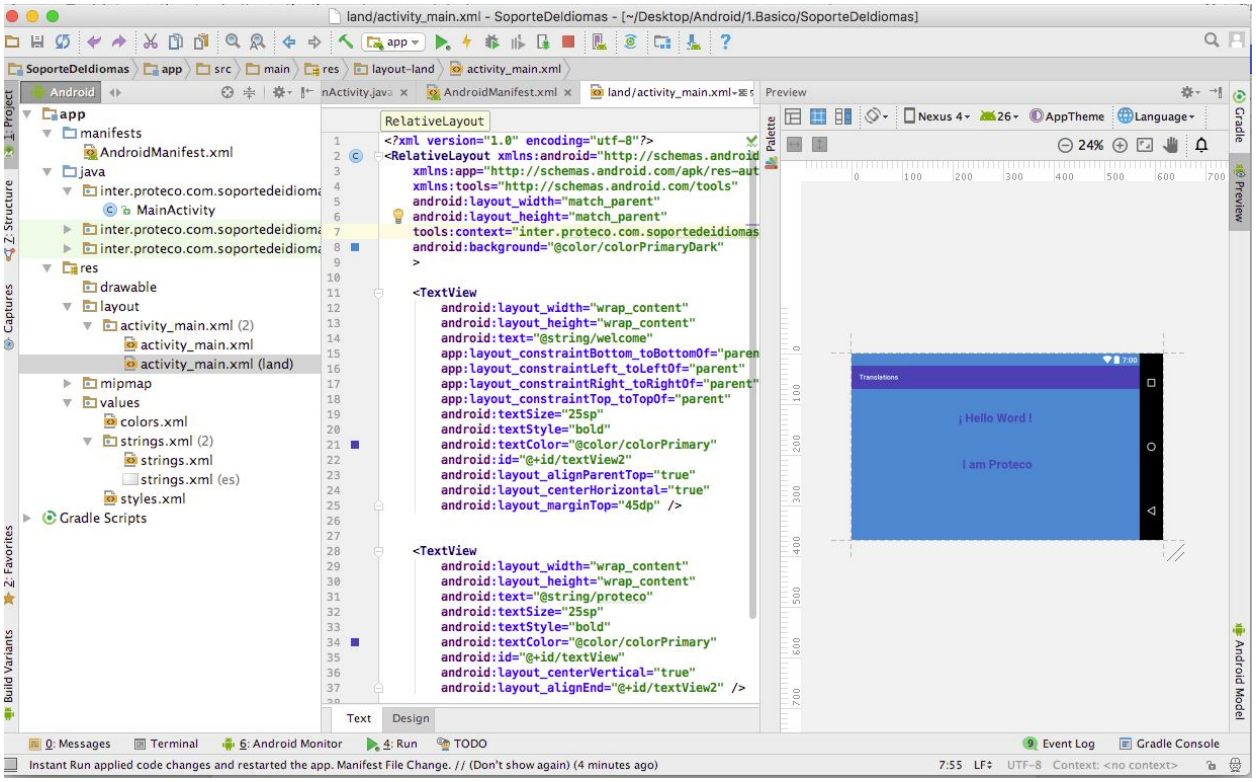
Tamaño de pantalla:

1. Usa wrap_content, match_parent o unidades dp al especificar las dimensiones en un archivo de diseño XML.
2. No uses valores de píxeles codificados en el código de tu aplicación.
3. No uses AbsoluteLayout (dejó de estar disponible).
4. Proporciona diferentes elementos de diseño de mapa de bits para diferentes densidades de pantalla.

Crear un nuevo Layout con una vista Horizontal.




layout/activity_main.xml - SoporteDeldiomas - [~/Desktop/Android/1.Basico/SoporteDeldiomas]
New Resource File
File name: activity_main
Root element: LinearLayout
Source set: main
Directory name: layout-land
Available qu...
Chosen qua...
Screen orientation:
Landscape
Cancel
OK


land/activity_main.xml - SoporteDeldiomas - [~/Desktop/Android/1.Basico/SoporteDeldiomas]
¡ Hello Word !
I am Proteco

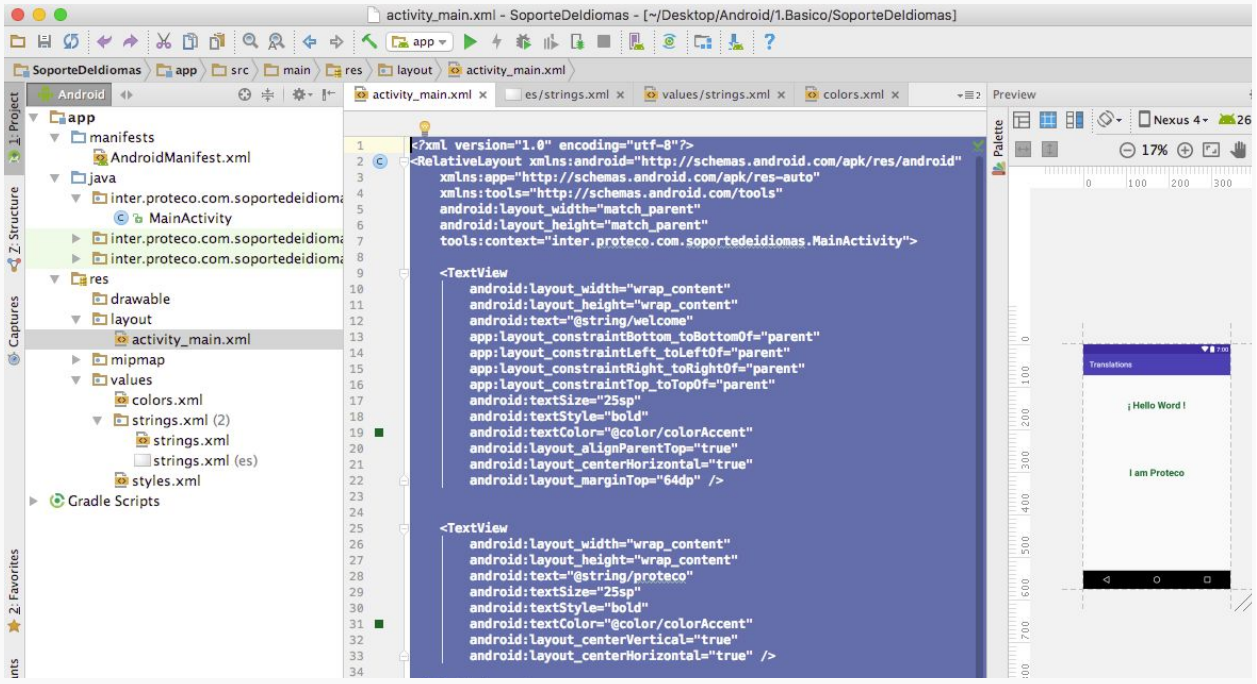
Soporte de idiomas

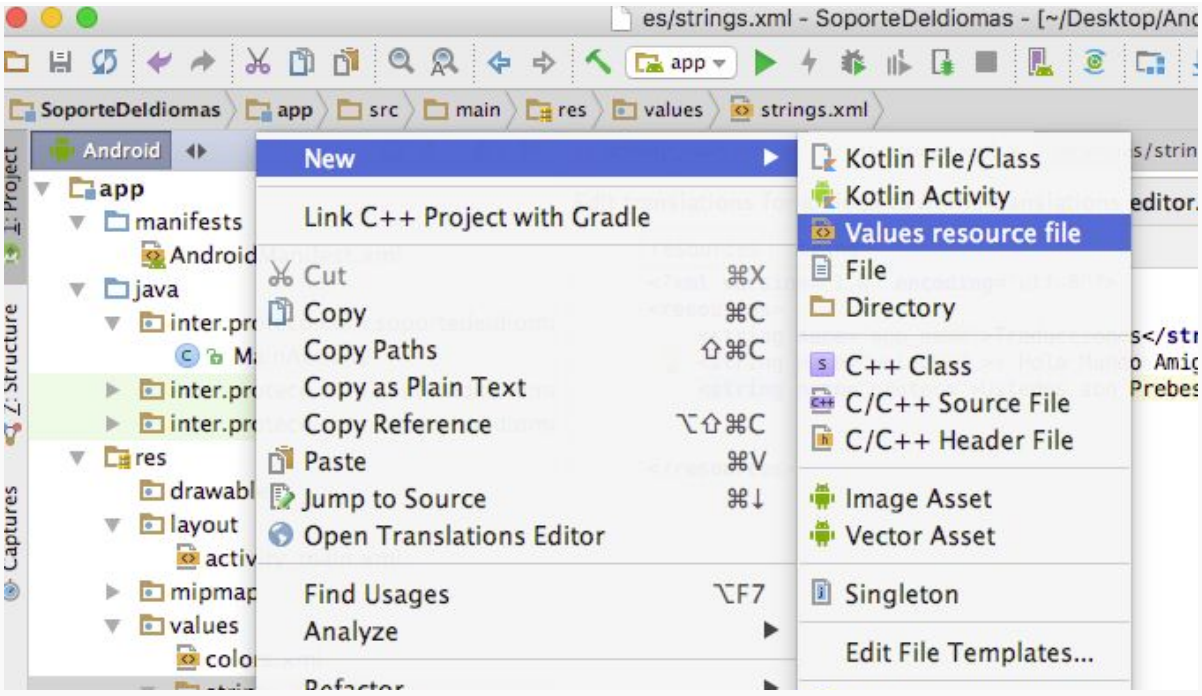
archivo externo, string.xml. -----> res/values/strings.xml

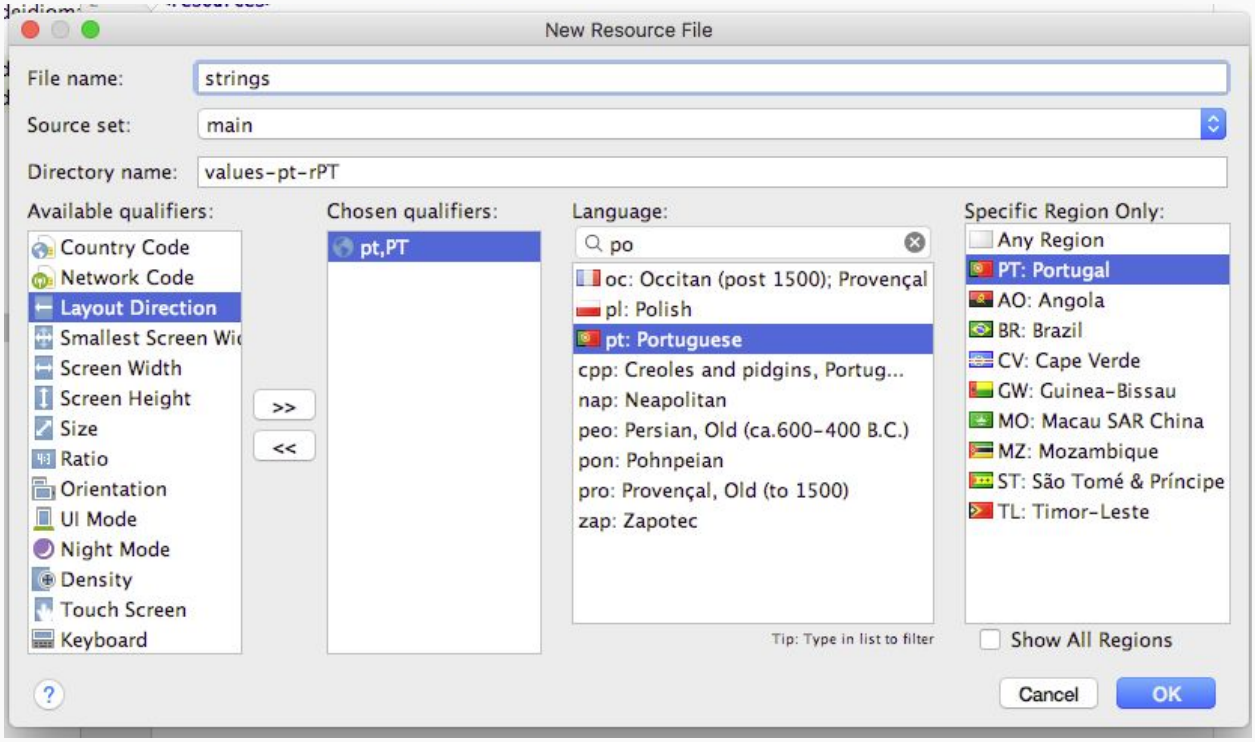
Ejemplo:

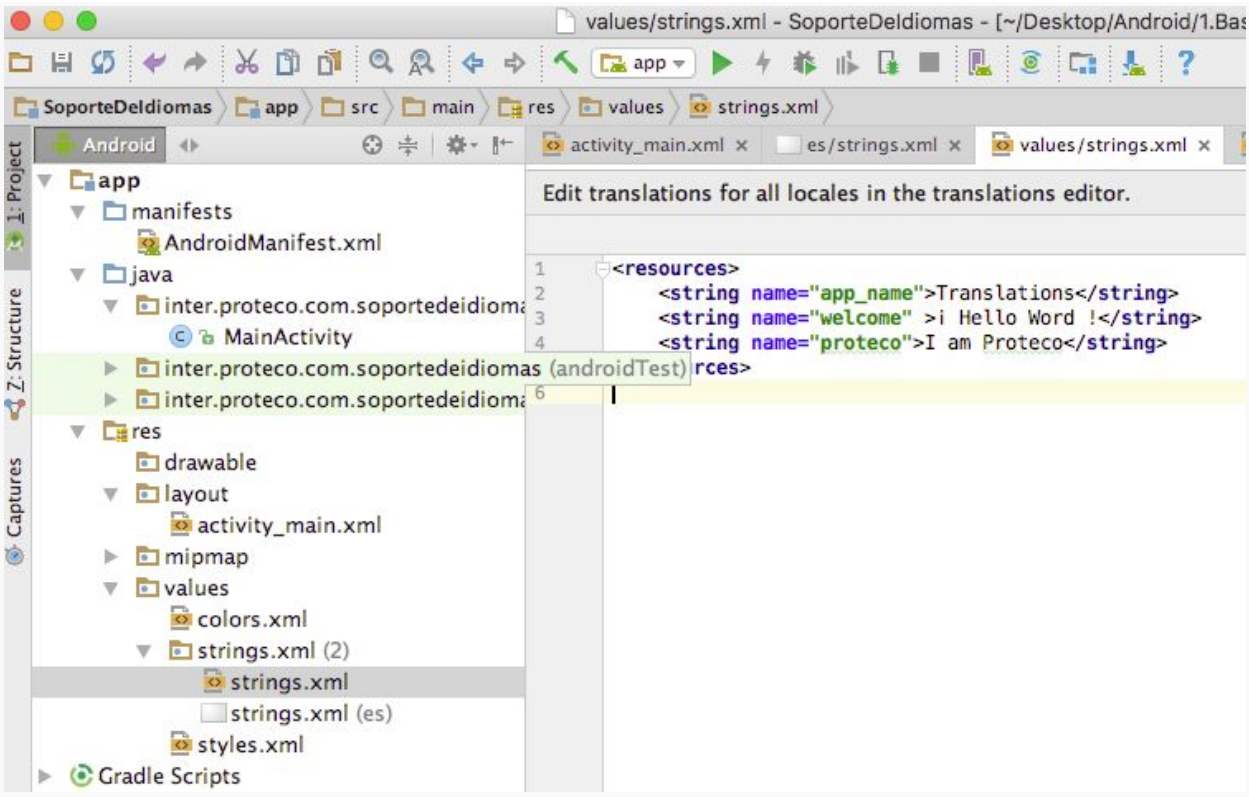
En nuestro activity_main

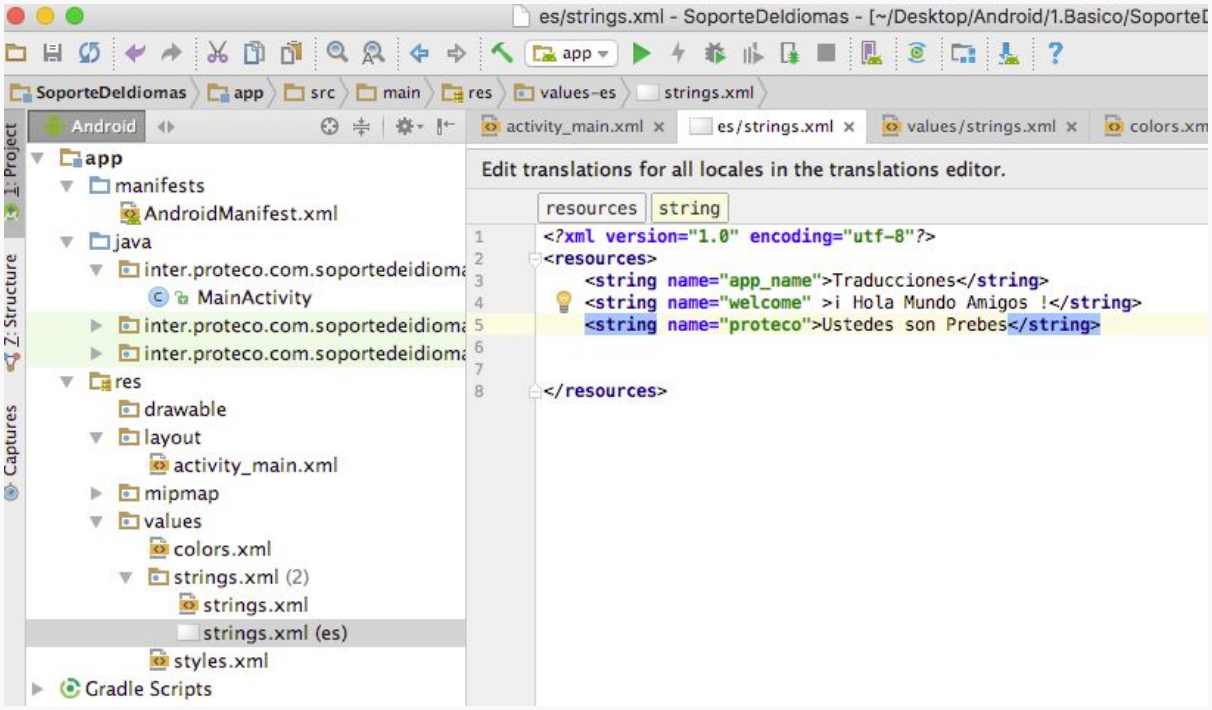
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context="inter.proteco.com.soportedeidiomas.MainActivity">

  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/welcome"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    android:textSize="25sp"
    android:textStyle="bold"
    android:textColor="@color/colorAccent"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="64dp" />

  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/proteco"
    android:textSize="25sp"
    android:textStyle="bold"
    android:textColor="@color/colorAccent"
    android:layout_centerVertical="true"
    android:layout_centerHorizontal="true" />

</RelativeLayout>
```


activity_main.xml - SoporteDeIdiomas - [~/Desktop/Android/1.Basico/SoporteDeIdiomas]
SoporteDeIdiomas
app
src
main
res
layout
activity_main.xml
Android
app
manifests
AndroidManifest.xml
java
inter.proteco.com.soportedeidioma
MainActivity
res
drawable
layout
activity_main.xml
mipmap
values
colors.xml
strings.xml (2)
strings.xml
strings.xml (es)
styles.xml
Gradle Scripts
activity_main.xml
es/strings.xml
values/strings.xml
colors.xml
Preview
Nexus 4
17%
Translations
¡Hello Word !
I am Proteco


es/strings.xml - SoporteDeIdiomas - [~/Desktop/And
SoporteDeIdiomas
app
src
main
res
values
strings.xml
Android
app
manifests
java
res
New
Link C++ Project with Gradle
Cut ⌘X
Copy ⌘C
Copy Paths ⇧⌘C
Copy as Plain Text
Copy Reference ⌥⇧⌘C
Paste ⌘V
Jump to Source ⌘↓
Open Translations Editor
Find Usages ⌥F7
Analyze
Kotlin File/Class
Kotlin Activity
Values resource file
File
Directory
C++ Class
C/C++ Source File
C/C++ Header File
Image Asset
Vector Asset
Singleton
Edit File Templates...


New Resource File
File name: strings
Source set: main
Directory name: values-pt-rPT
Available qualifiers:
Country Code
Network Code
Layout Direction
Smallest Screen Wi
Screen Width
Screen Height
Size
Ratio
Orientation
UI Mode
Night Mode
Density
Touch Screen
Keyboard
>>
<<
Chosen qualifiers:
pt,PT
Language:
po
oc: Occitan (post 1500); Provençal
pl: Polish
pt: Portuguese
cpp: Creoles and pidgins, Portug...
nap: Neapolitan
peo: Persian, Old (ca.600-400 B.C.)
pon: Pohnpeian
pro: Provençal, Old (to 1500)
zap: Zapotec
Tip: Type in list to filter
Specific Region Only:
Any Region
PT: Portugal
AO: Angola
BR: Brazil
CV: Cape Verde
GW: Guinea-Bissau
MO: Macau SAR China
MZ: Mozambique
ST: São Tomé & Príncipe
TL: Timor-Leste
Show All Regions
Cancel
OK


values/strings.xml - SoporteDeIdiomas - [~/Desktop/Android/1.Bas
app
SoporteDeIdiomas
app
src
main
res
values
strings.xml
Android
activity_main.xml
es/strings.xml
values/strings.xml
Edit translations for all locales in the translations editor.
app
manifests
AndroidManifest.xml
java
inter.proteco.com.soportedeidioma
MainActivity
inter.proteco.com.soportedeidiomas (androidTest)
inter.proteco.com.soportedeidioma
res
drawable
layout
activity_main.xml
mipmap
values
colors.xml
strings.xml (2)
strings.xml
strings.xml (es)
styles.xml
Gradle Scripts
<resources>
<string name="app_name">Translations</string>
<string name="welcome" >i Hello Word !</string>
<string name="proteco">I am Proteco</string>
rces>
1: Project
Z: Structure
Captures



Soporte de versiones

Para proporcionar las mejores características y funcionalidades a lo largo de varias versiones Android, deberías utilizar la [Android Support Library](#) en tu app, la cual te permite utilizar las APIs de la plataforma más recientes en las versiones antiguas.

Especificar los Niveles de API Objetivo y Mínimo

El archivo `AndroidManifest.xml` describe los detalles sobre tu app e identidades sobre cuales versiones de Android soporta. Específicamente, el atributo `minSdkVersion` dentro del elemento `<uses-sdk` identifica la API de menor nivel con la que tu app es compatible y la API de mayor nivel para la cual tu app ha sido diseñada y testeada.

AndroidManifest.xml

```
0
1 <manifest xmlns:android="http://schemas.android.com/apk/res/android" ... >
2     <uses-sdk android:minSdkVersion="4" android:targetSdkVersion="15" />
3     ...
4 </manifest>
5
```

Las nuevas versiones de Android que son publicadas pueden modificar el comportamiento y los estilos. Permite que tu app tome las ventajas de dichos cambios y asegúrate de que tu app encaja con el estilo de cada dispositivo de los usuarios, por tanto deberías utilizar el valor del atributo `targetSdkVersion` para que sea el de la última versión Android disponible.

Comprobar la Versión del Sistema en Tiempo de Ejecución

Android proporciona un código único para cada versión de plataforma en la clase `Build`. Utiliza dichos códigos dentro de tu app para crear condiciones que aseguren que tu código que depende de los niveles de API superiores es ejecutado únicamente cuando dichas APIs se encuentren disponibles en el sistema.

MainActivity.java

```
0
1 private void setUpActionBar() {
2     // Make sure we're running on Honeycomb or higher to use ActionBar APIs
3     if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB) {
4         ActionBar actionBar = getActionBar();
5         actionBar.setDisplayHomeAsUpEnabled(true);
6     }
7 }
8
```

Tambien de esta forma.

```
0
1 private void setUpActionBar() {
2     // Make sure we're running on Honeycomb or higher to use ActionBar APIs
3     if (Build.VERSION.SDK_INT >= 11 {
4         ActionBar actionBar = getActionBar();
5         actionBar.setDisplayHomeAsUpEnabled(true);
6     }
7 }
8
```

Nota: Cuando analiza recursos XML, Android ignora los atributos XML que no son soportados por el dispositivo actual. Así puedes utilizar atributos XML de forma segura que sean soportados únicamente por versiones superiores sin preocuparte sobre si las versiones antiguas dejarían de funcionar cuando encontrasen dicho código. Por ejemplo, si tu seleccionas targetSdkVersion)"11", tu app incluye la ActionBar por defecto en Android 3.0 o superior. Entonces añadir elementos de menú a la action bar, necesitas utilizar android:showAsAction="ifRoom" en tu recurso de menú XML. Esto es seguro en archivos XML de diferentes versiones, ya que las versiones más antiguas de Android simplemente ignorarán el atributo showAsAction (es decir, no necesitas crear una versión separada en res/menu-v11/).
https://developer.android.com/about/dashboards/index.html

Estilos y temas

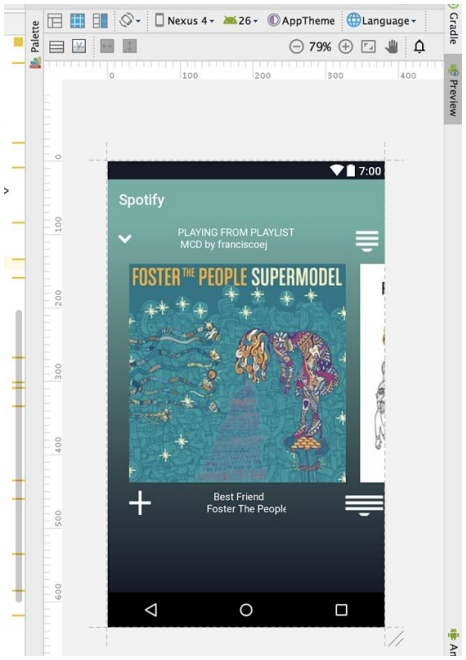
Android proporciona temas para la experiencia de usuario que dan a las apps el aspecto y las sensaciones que transmite el mismo sistema operativo. Esos temas pueden aplicarse a tu app dentro del archivo manifest. Utilizando esos estilos y temas, que ya se encuentran listos para utilizarse, tu app seguirá el último aspecto de forma natural de Android con cada nueva versión publicada.
Para que tu actividad tenga el aspecto de una caja de diálogo:
<activity android:theme="@android:style/Theme.Dialog">
Para que tu actividad tenga un fondo transparente:
<activity android:theme="@android:style/Theme.Translucent">
Para aplicar tu propio tema personalizado definido en res/values/styles.xml:
<activity android:theme="@style/CustomTheme">

Para aplicar un tema a tu app al completo (todas las actividades), añade el atributo android:theme al elemento <application>:
<application android:theme="@style/CustomTheme">

Tarea:

Ver el siguiente video:

1. https://www.youtube.com/watch?v=bA2wDt7MznE **(Muy importante)**
2. **Para la tarea de Spotify completar su tarea de tal forma que se vea de la siguiente forma.**



3. **Ademas deberan agregar un nuevo Layout con orientación horizontal(landscape) de tal forma que al rotar su celular se vea la siguiente vista.**

4. **Agregar soporte de idiomas**

Pd. Reviso el Lunes 31 de Octubre

Estimados Amigos PREBECARIOS debido al poco tiempo del curso se removió esta clase y algunas otras. Por lo que me apena decirles que nos vemos el viernes 3 de Nov para continuar con la diversión. Cualquier duda mandar correo a barcenas.proteco@gmail.com. Saludos y que la fuerza esté con ustedes. Si quieren una clase de Swift, Kotlin, Material Design o algun repaso de algún tema, nos ponemos de acuerdo para programar alguna clase algun sabado despues de C Linux. Hasta Pronto.