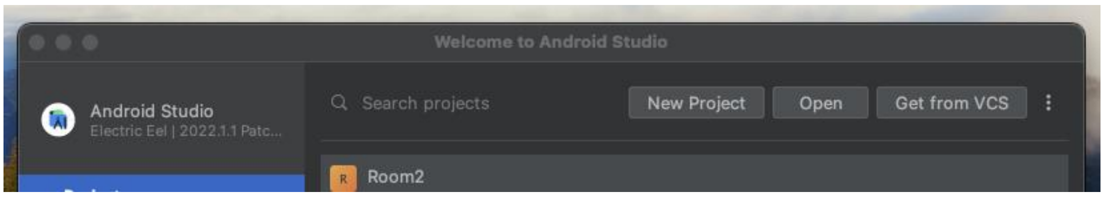
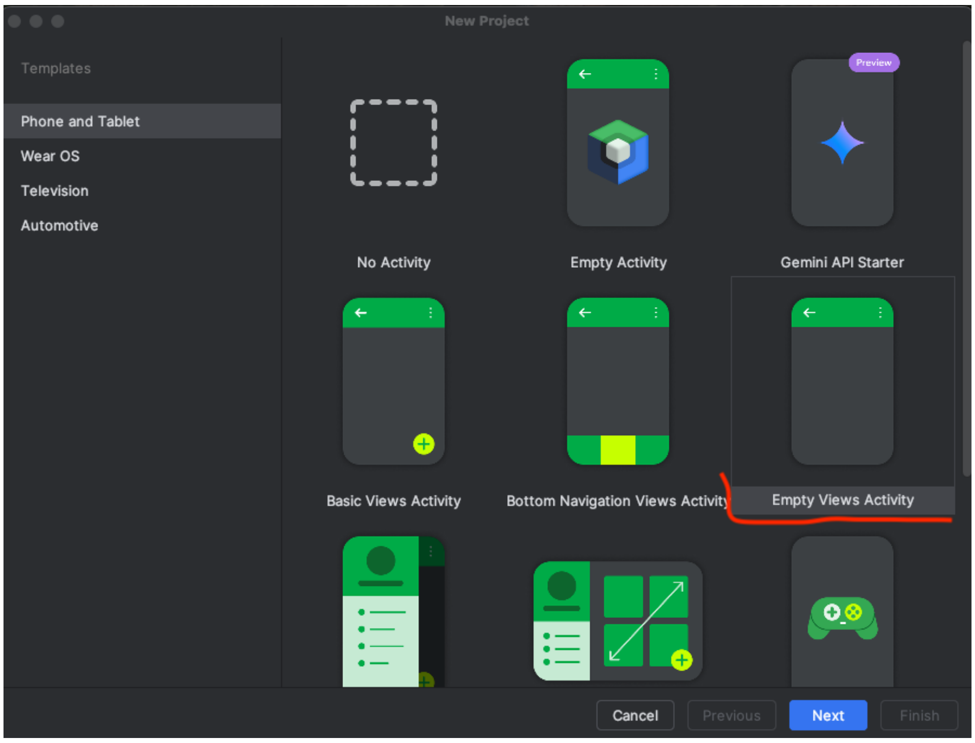
Android Apps – Static Layouts

▪ Launch Android Studio

▪ Create a new project (File -> New Project or New Project from a Startup window)

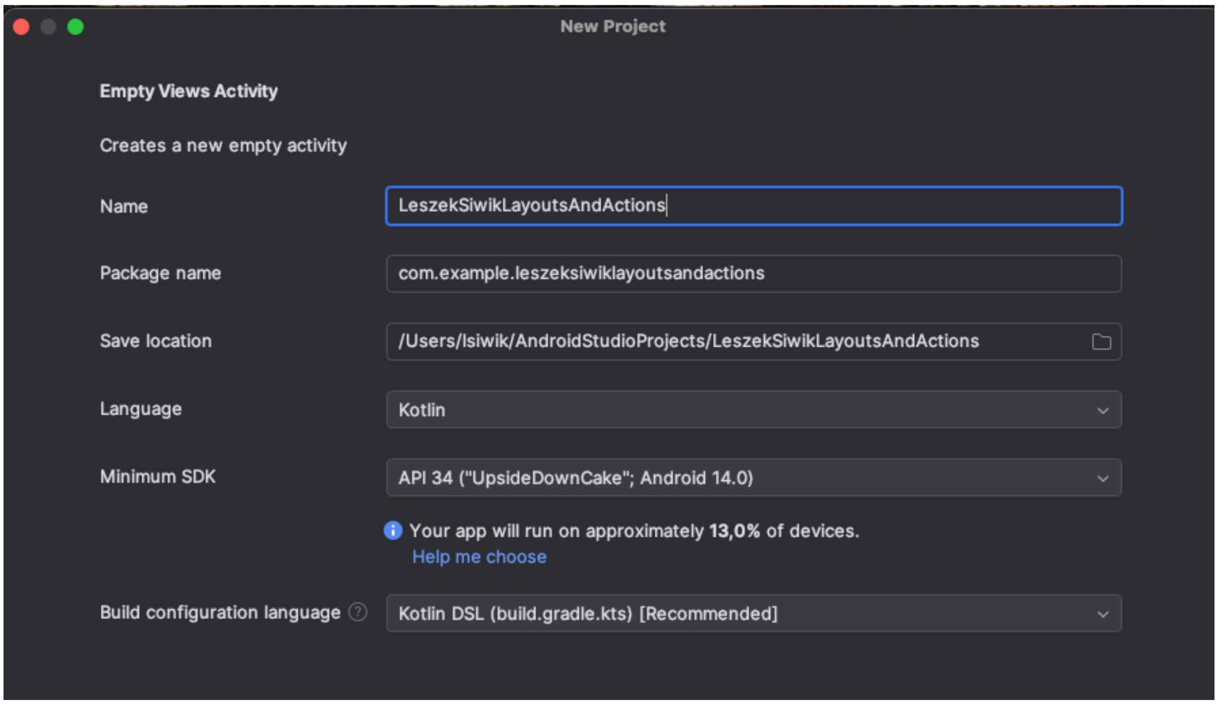


▪Select Phone and Tablet and Empty Views Activity as a project template



▪ On the next screen set:

- FirstnameSurnameLayoutsAndActions as the project name
- Default value as the package name and project location
- Kotlin as project language
- API 34 Android 13.0 (UpsideDownCake) as the Minimum SDK
- And press Finish


New Project

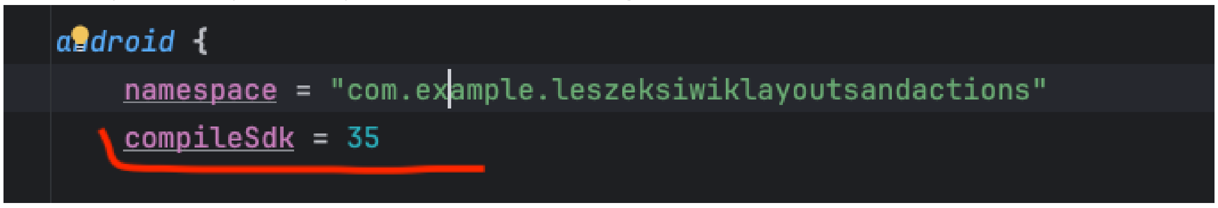
Empty Views Activity

Creates a new empty activity

Name	LeszekSiwikLayoutsAndActions
Package name	com.example.leszeksiwiklayoutsandactions
Save location	/Users/lsiwik/AndroidStudioProjects/LeszekSiwikLayoutsAndActions
Language	Kotlin
Minimum SDK	API 34 ("UpsideDownCake"; Android 14.0)
Build configuration language	Kotlin DSL (build.gradle.kts) [Recommended]

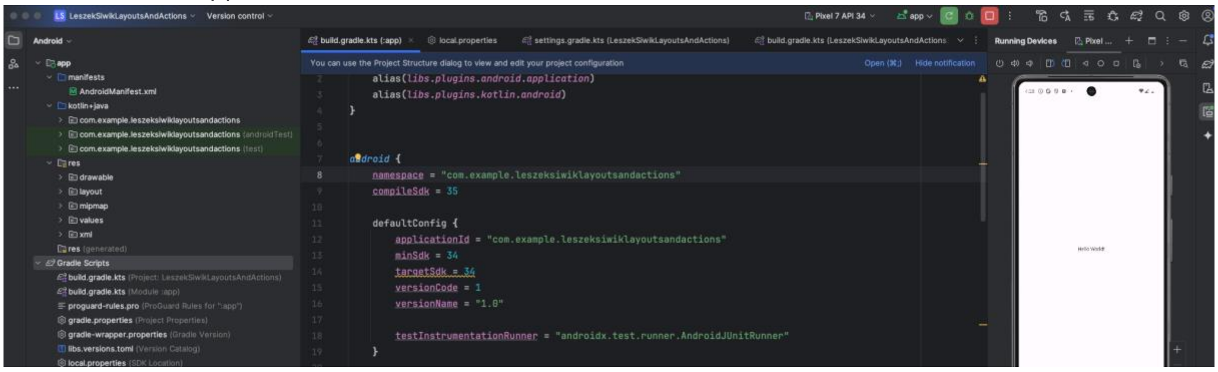
Your app will run on approximately **13,0%** of devices.
Help me choose

- Wait until Gradle finishes his job and let's launch the app to make sure that everything is fine. If you are getting an error saying that Dependency 'androidx.core:core:1.15.0' requires libraries and applications that depend on it to compile against version 35 or later of the Android APIs. Just bump the compileSDK up to 35 version in build.gradle.kts (Module) file



```
android {
    namespace = "com.example.leszeksiwiklayoutsandactions"
    compileSdk = 35
```

-
- And rebuild the app.

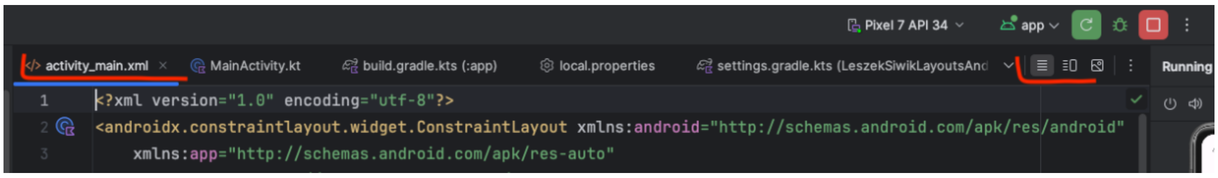


```
    alias(libs.plugins.android.application)
    alias(libs.plugins.kotlin.android)
}


android {
    namespace = "com.example.leszeksiwiklayoutsandactions"
    compileSdk = 35

    defaultConfig {
        applicationId = "com.example.leszeksiwiklayoutsandactions"
        minSdk = 34
        targetSdk = 34
        versionCode = 1
        versionName = "1.0"

        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
    }
```

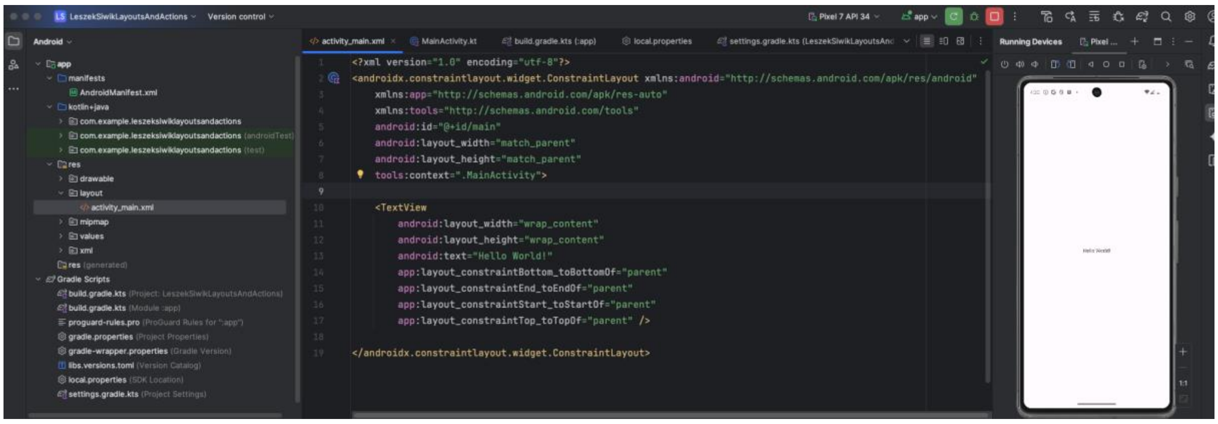
-

- ## **Working with static Layouts**

    - Definition of how our first activity (first screen) looks like is contained in the res -> layout -> activity_main.xml file

- Because we are practicing, please complete the ongoing exercises by editing the xml file(s) directly!
- Possibly the most convenient view mode for us will be a split view showing both: the UI source code and the designer where you may follow the effects of your changes made in the source code.   So please change the view mode into the SplitView
- 

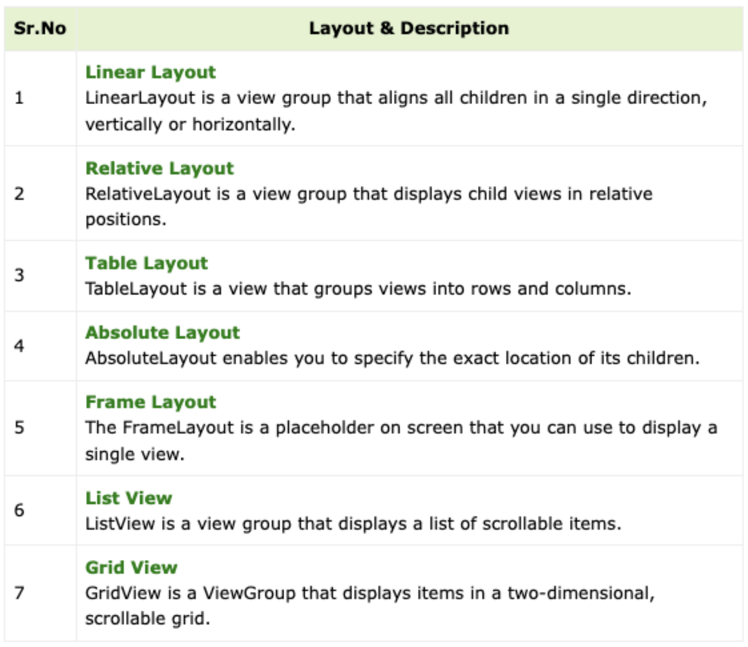
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
```

- And you should get something like:
- 

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout 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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

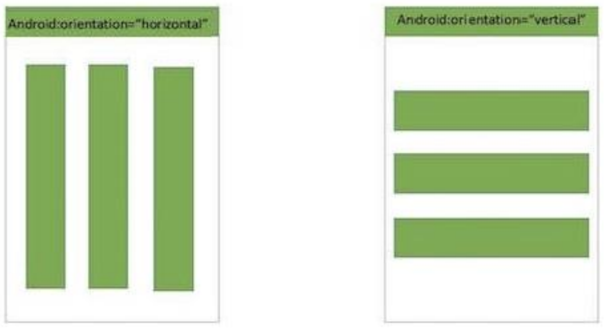
LeszekSiwikLayoutsAndActions
Version control
Android
app
manifests
AndroidManifest.xml
kotlin+java
com.example.leszeksiwiklayoutsandactions
com.example.leszeksiwiklayoutsandactions (androidTest)
com.example.leszeksiwiklayoutsandactions (test)
res
drawable
layout
activity_main.xml
mipmap
values
xml
res (generated)
Gradle Scripts
build.gradle.kts (Project: LeszekSiwikLayoutsAndActions)
build.gradle.kts (Module :app)
proguard-rules.pro (ProGuard Rules for ":app")
gradle.properties (Project Properties)
gradle-wrapper.properties (Gradle Version)
libs.versions.toml (Version Catalog)
local.properties (SDK Location)
settings.gradle.kts (Project Settings)
Running Devices
Pixel ...
Hello World!

- **Layouts**
  - When we are defining the user interface, we may use "static" layouts - a type of containers, defined in the xml files, where we may drop off user interface elements like buttons, lists, text fields, other nested layouts, etc.)
  - Android SDK provides some predefined layouts like:

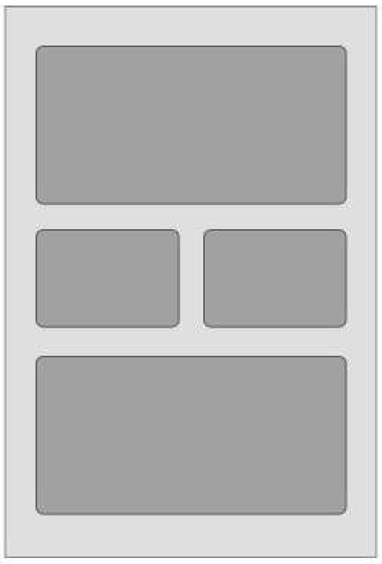


Sr.No	Layout & Description
1	**Linear Layout** LinearLayout is a view group that aligns all children in a single direction, vertically or horizontally.
2	**Relative Layout** RelativeLayout is a view group that displays child views in relative positions.
3	**Table Layout** TableLayout is a view that groups views into rows and columns.
4	**Absolute Layout** AbsoluteLayout enables you to specify the exact location of its children.
5	**Frame Layout** The FrameLayout is a placeholder on screen that you can use to display a single view.
6	**List View** ListView is a view group that displays a list of scrollable items.
7	**Grid View** GridView is a ViewGroup that displays items in a two-dimensional, scrollable grid.

- (https://www.tutorialspoint.com/android/android_user_interface_layouts.htm)
- LinearLayout – aligns all the elements dropped off inside linearly (vertically or horizontally depending on the "orientation") one by one:

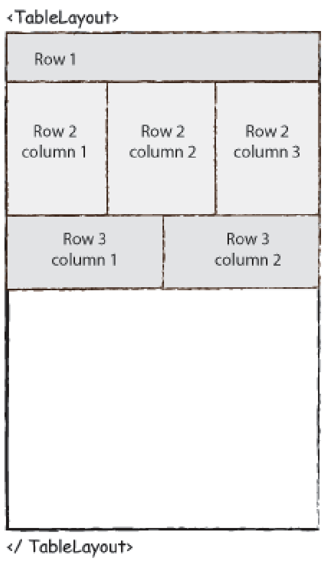

Android:orientation="horizontal"
Android:orientation="vertical"

- More information about LinearLayouts can be found e.g. here: https://www.tutorialspoint.com/android/android_linear_layout.htm
- RelativeLayout - where elements are placed "relatively" to each other - e.g. we place something to the right of something, something else under something, etc.

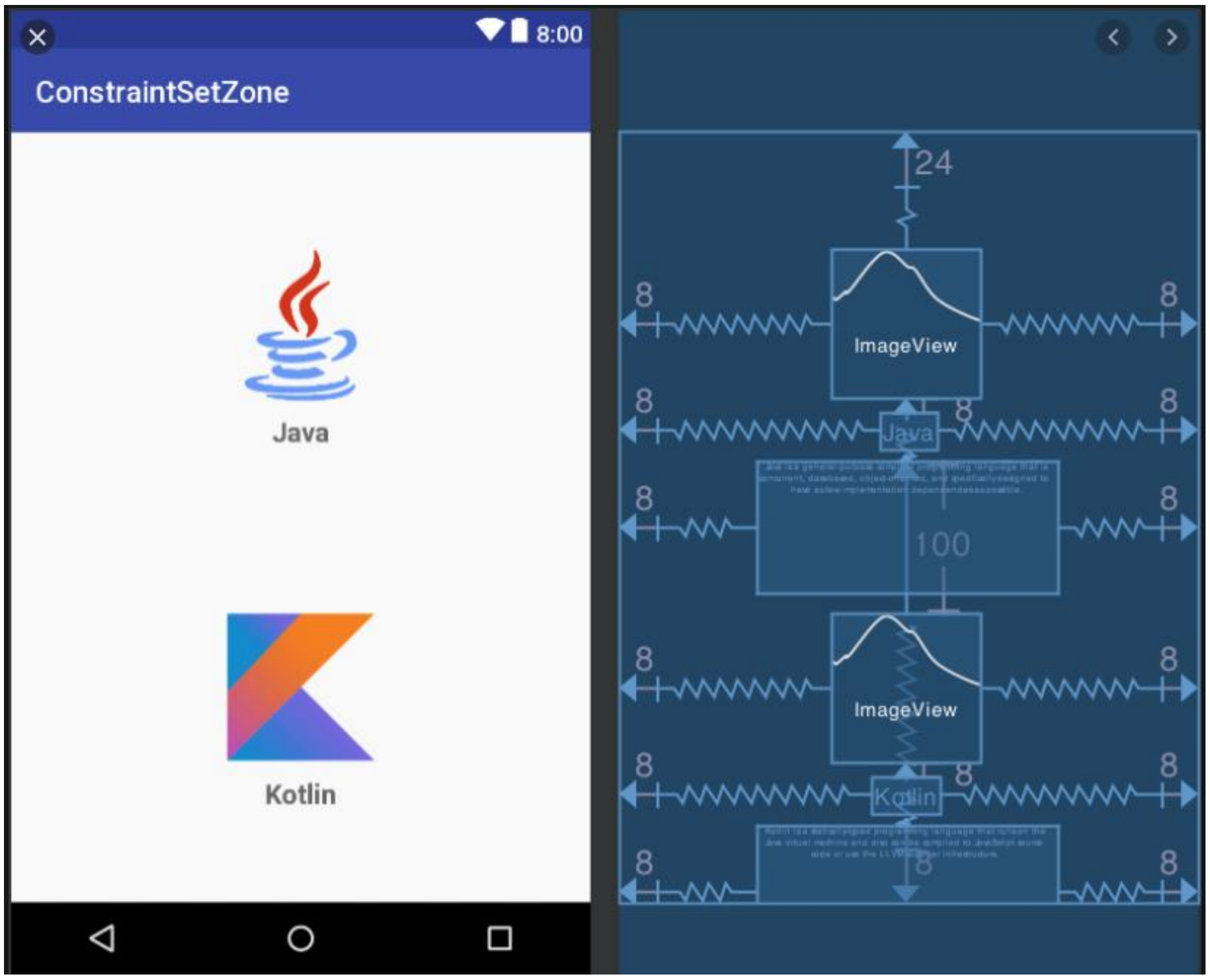


- Relative Layout

- More information about RelativeLayouts can be found e.g. here: https://www.tutorialspoint.com/android/android_relative_layout.htm
- TableLayout – where elements are located in rows and columns

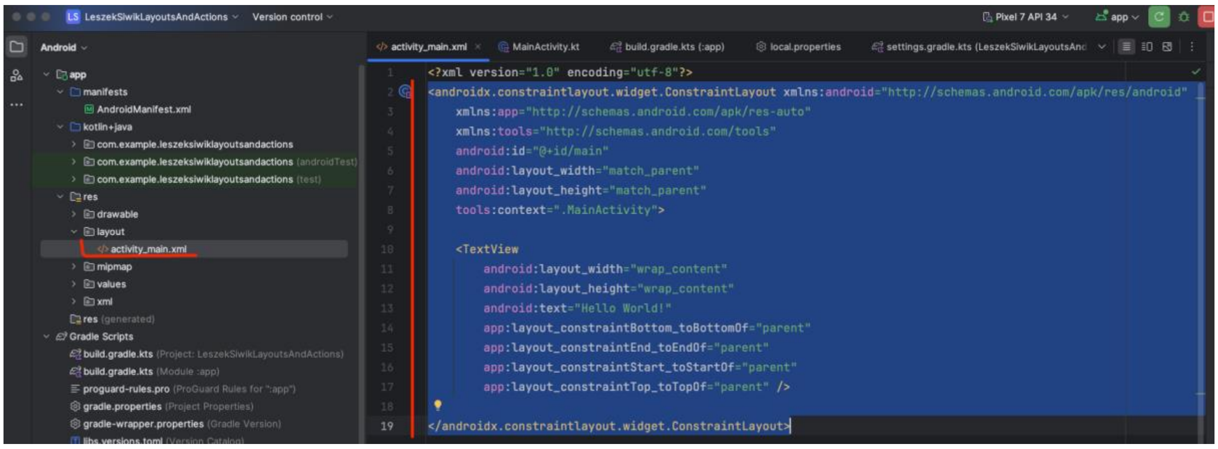


- More information about TableLayouts can be found e.g. here: https://www.tutorialspoint.com/android/android_table_layout.htm
- ConstraintLayout – where elements are "pinned together" with constraits/strings

- 

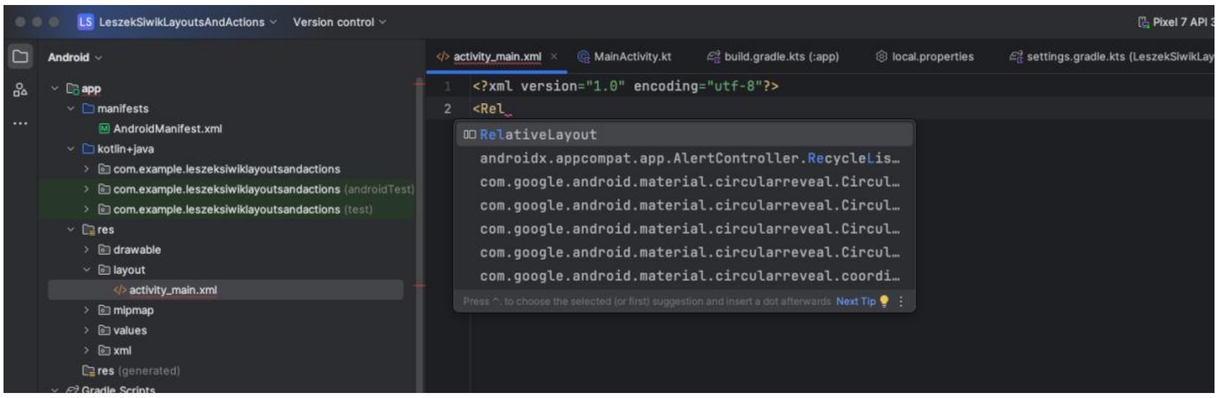
- ## RelativeLayout

  - For the beginning, we start working with the RelativeLayout
  - As it was said, the RelativeLayout is a "container" where elements are placed "relatively" to each other – i.e, we place something to the right of something, something else under something etc. We will work with it for a while now.
  - Please, open the main_activity.xml file. Delete the ConstraintLayout definition added there by the wizard. (So remove lines 2 to 19)

- 

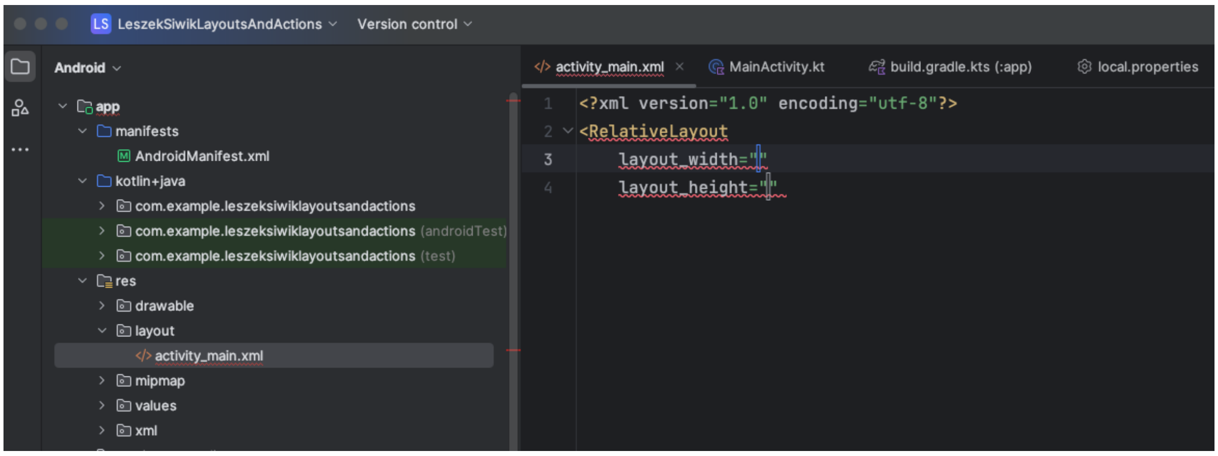
```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

- Instead, please start typing “<Rel .... and when the intellisense shows up the RelativeLayout just press the enter:



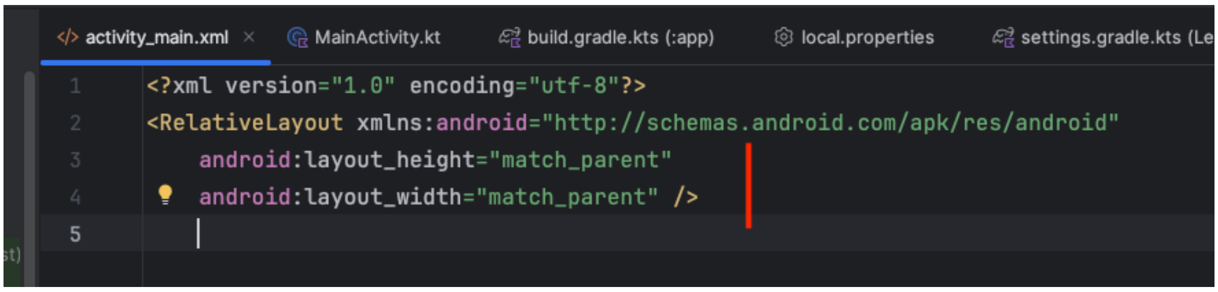
```xml
<?xml version="1.0" encoding="utf-8"?>
<Rel
```

```
RelativeLayout
androidx.appcompat.app.AlertController.RecycleLis…
com.google.android.material.circularreveal.Circul…
com.google.android.material.circularreveal.Circul…
com.google.android.material.circularreveal.Circul…
com.google.android.material.circularreveal.Circul…
com.google.android.material.circularreveal.coordi…
Press ^. to choose the selected (or first) suggestion and insert a dot afterwards Next Tip
```

- 

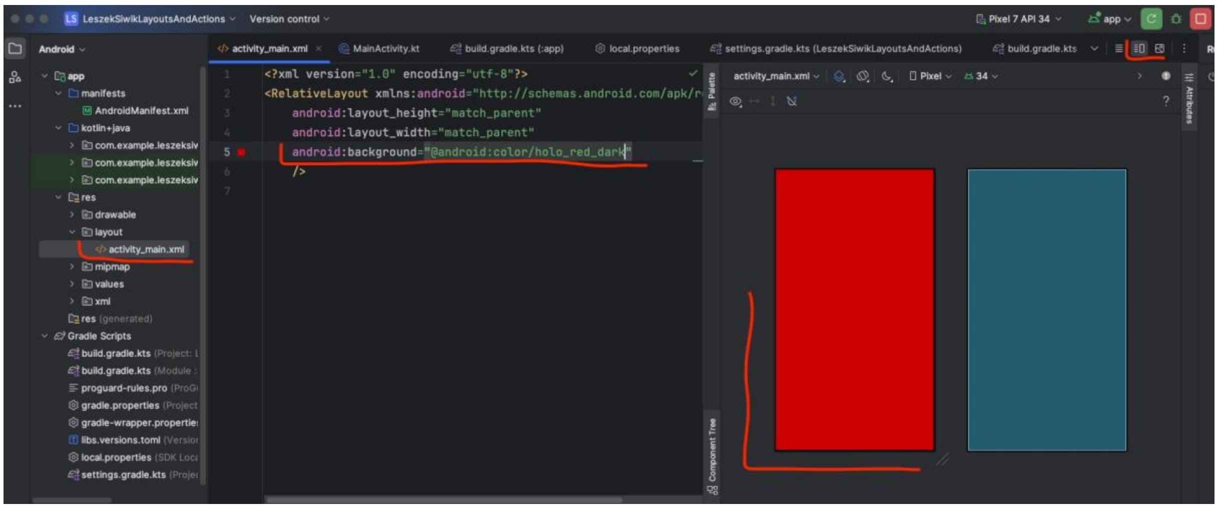
```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    layout_width=""
    layout_height=""
```

- As we can see, the RelativeLayout has two mandatory parameters - layout_width and layout_height. We set both to match_parent - which means that we want to define the layout as width and as height as its "parent" is, and because the created layout is the "top most” layout – (it is not nested in any other layout) - its "parent" is just the screen of the phone on which the application will be launched - so by setting layout_width and layout_height to "match_parent" we say that we want the created layout to be as wide and as high as the screen of the device on which the application will be launched .

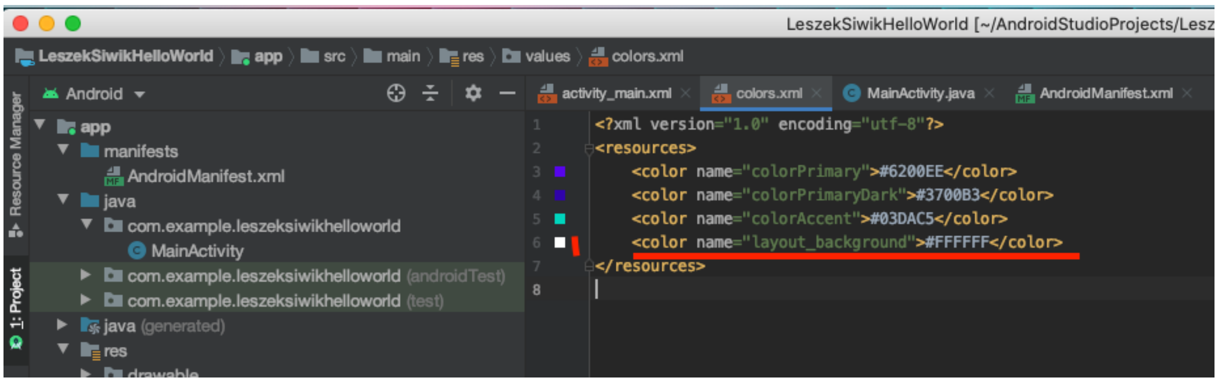
- 

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

- Additionally, let's set the background color of the layout, to red (or any other), what makes it easy to “track” of how the layout is defined. So, being inside the layout definition, we start typing “back” and Intellisense should support us:

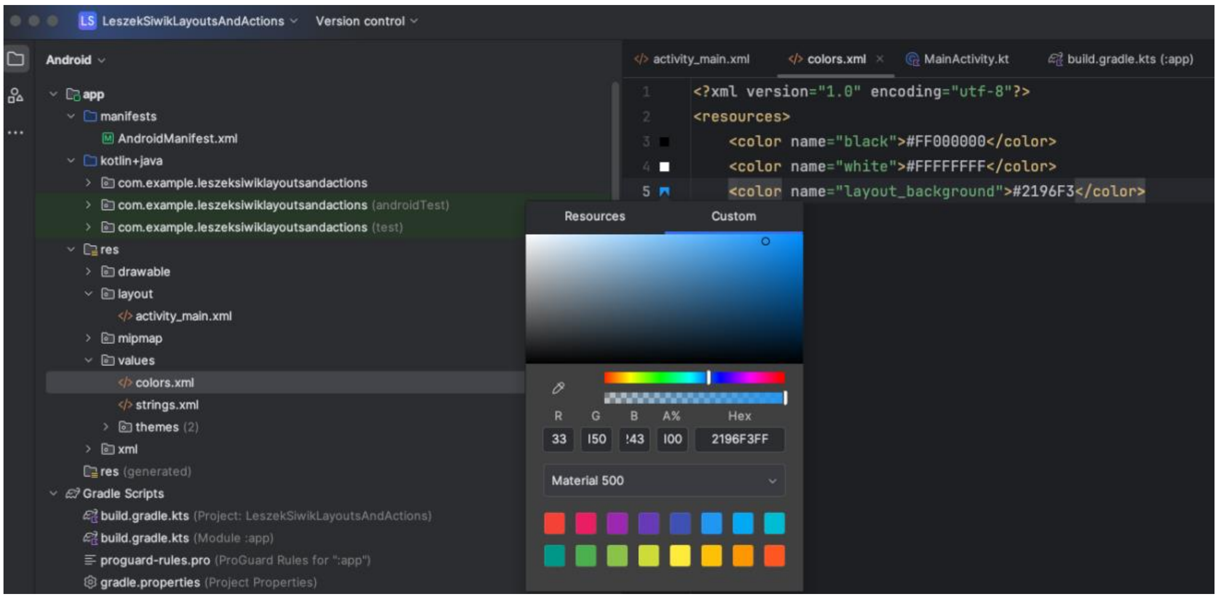


```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/r
    android:layout_height="match_parent"
    android:layout_width="match_parent"
    android:background="@android:color/holo_red_dark"
    />
```

- **Working with custom colors**
  - As we touched the color topic - let's define our own color and set it as the layout background color. For this purpose:
    - Open res/values/colors.xml file
  - Similarly to those already defined there - let's add a new "color item"
  - Set the name of the color to be added (e.g "layout_background") and its value to #FFFFFF
  - 

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#6200EE</color>
    <color name="colorPrimaryDark">#3700B3</color>
    <color name="colorAccent">#03DAC5</color>
    <color name="layout_background">#FFFFFF</color>
</resources>
```

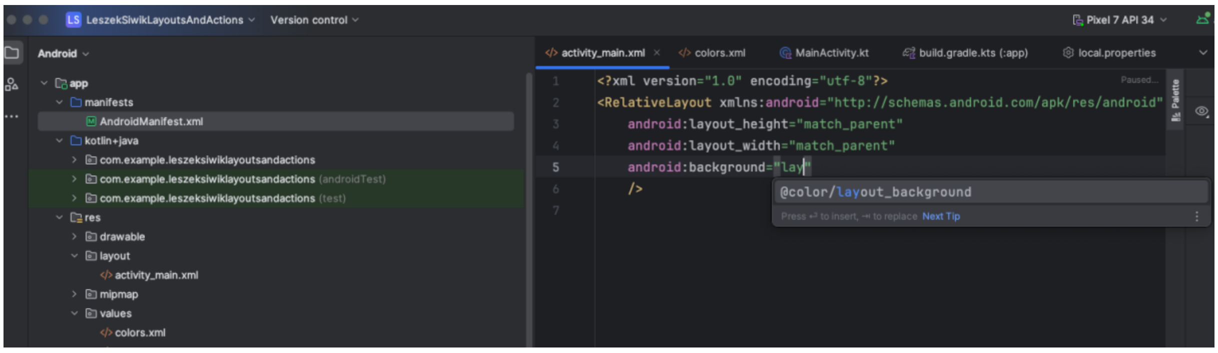
- Click on the white square to the right of the line number (for me to the right of line 6) and choose any color you want to be used

- 

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <resources>
3      <color name="black">#FF000000</color>
4      <color name="white">#FFFFFFFF</color>
5      <color name="layout_background">#2196F3</color>
```

```
R    G    B    A%   Hex
33   150  243  100  2196F3FF
Material 500
```

- Now, let’s go back to activity_main.xml file and set the value of the layout_background property to the color we have just defined.

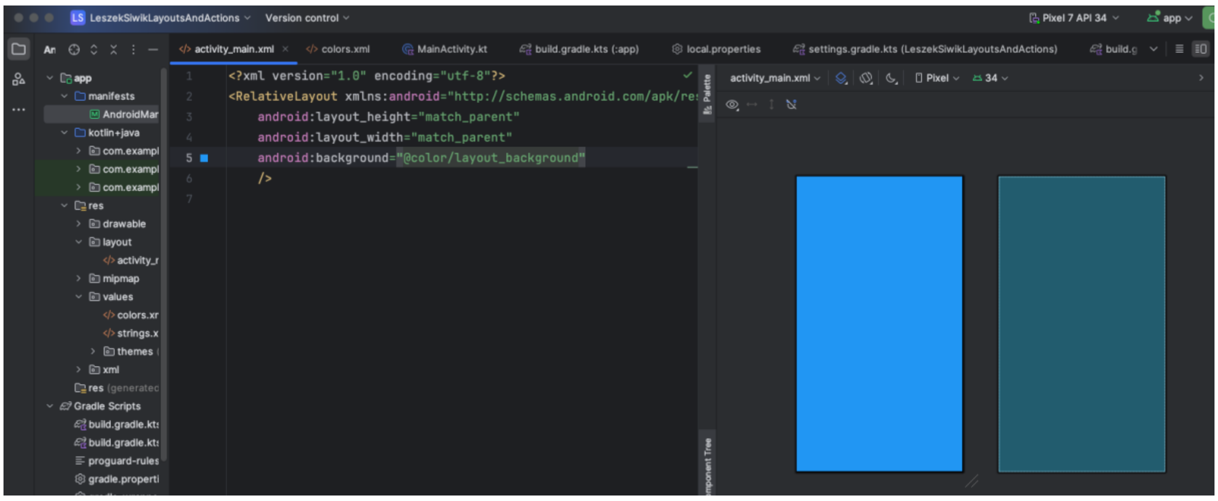
- 

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_height="match_parent"
    android:layout_width="match_parent"
    android:background="lay"
    />
```

```
@color/layout_background
```

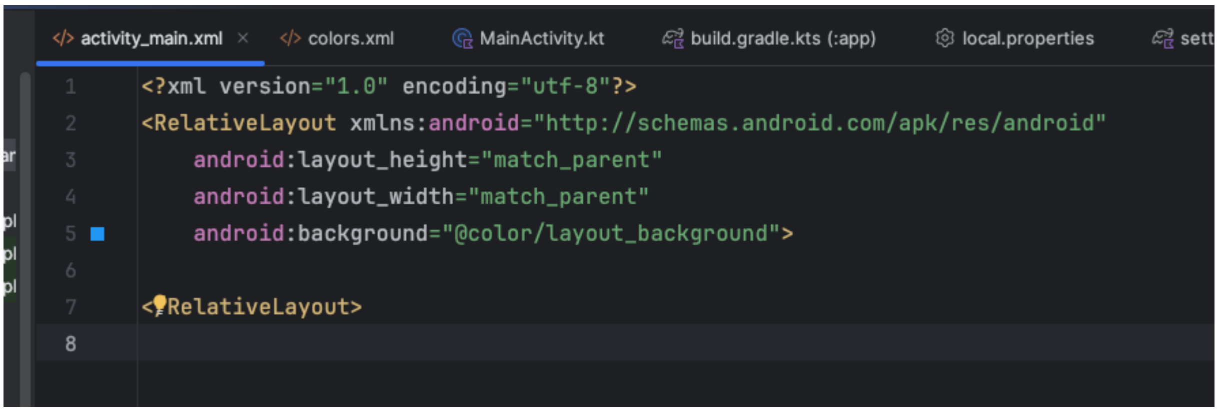
-

- The color change should be visible on the designer / preview

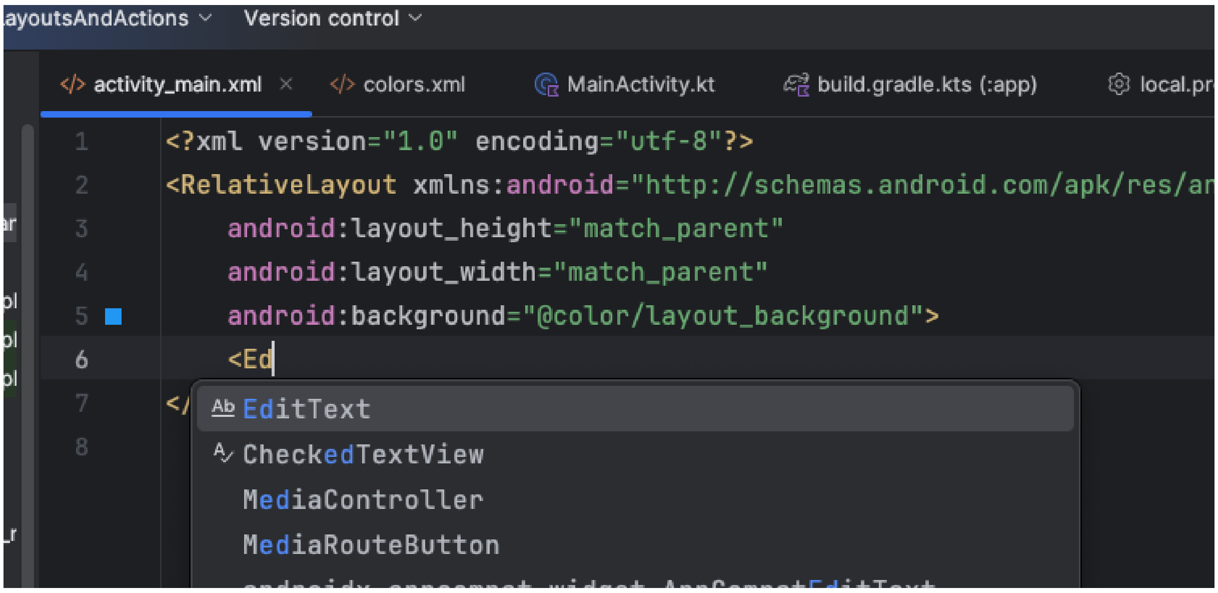
- 

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_height="match_parent"
    android:layout_width="match_parent"
    android:background="@color/layout_background"
    />
```

- ## Working with label

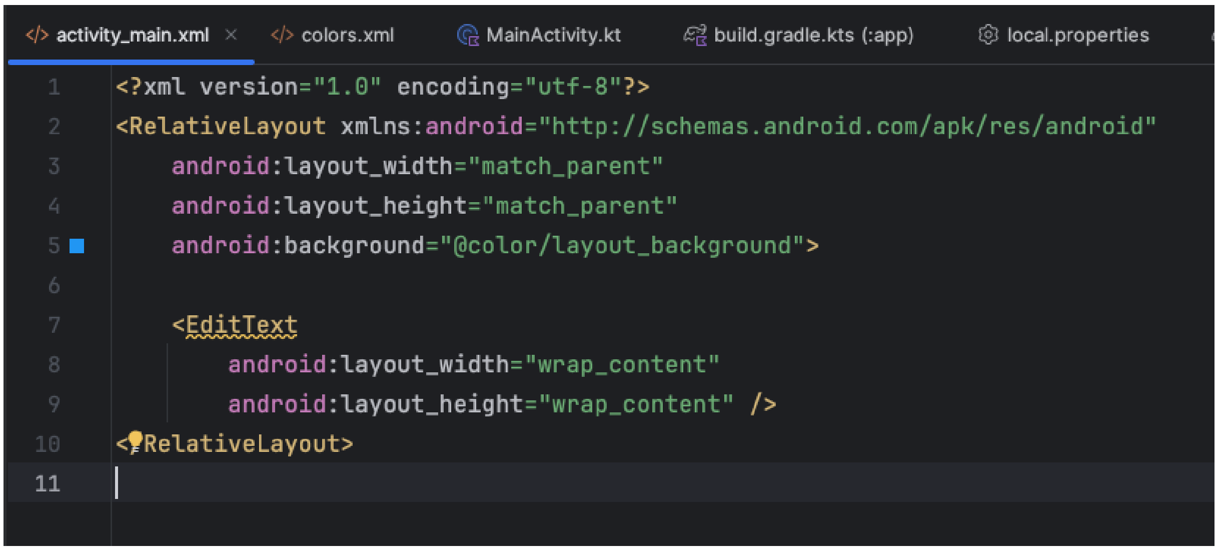
  - Let's add some "label" (static inscription) to the layout, namely:
  - Being in the activity_main.xml file - we split RelativeLayout definition into two separate tags - opening and closing ones.
  - So, we change its definitions of our layout as follows:
  - 

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_height="match_parent"
    android:layout_width="match_parent"
    android:background="@color/layout_background">

</RelativeLayout>
```

  - And now, we may add inside the layout the EditText
  - 

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/ar
    android:layout_height="match_parent"
    android:layout_width="match_parent"
    android:background="@color/layout_background">
    <Ed
</
```

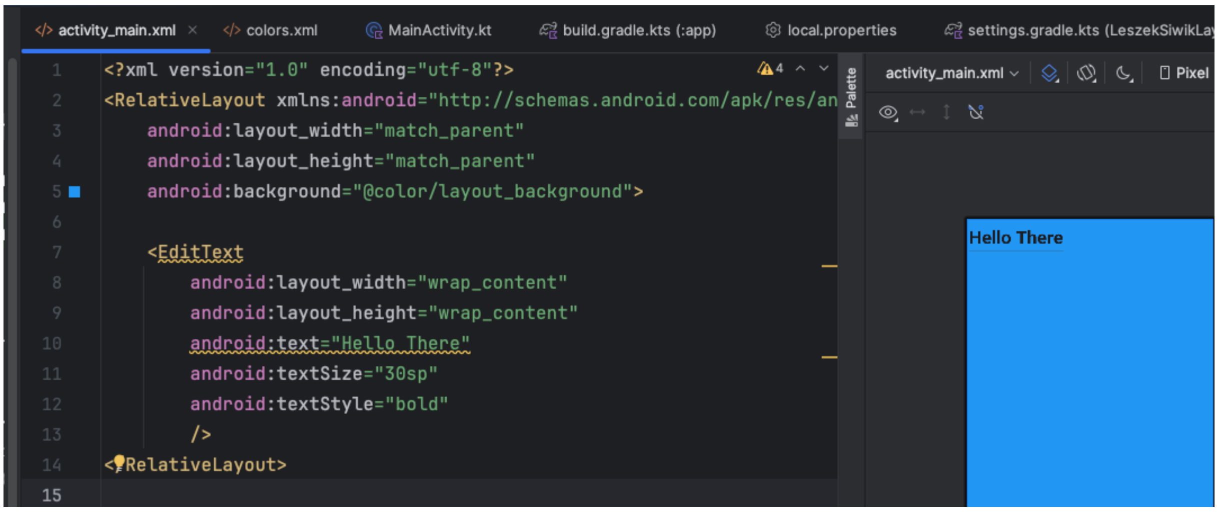
  - This time, let’s set the width and height of this element to "wrap_content" - that is, we want the EditText to be as wide and as high as the inscription we will place inside.



activity_main.xml | colors.xml | MainActivity.kt | build.gradle.kts (:app) | local.properties

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/layout_background">

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
<RelativeLayout>
```

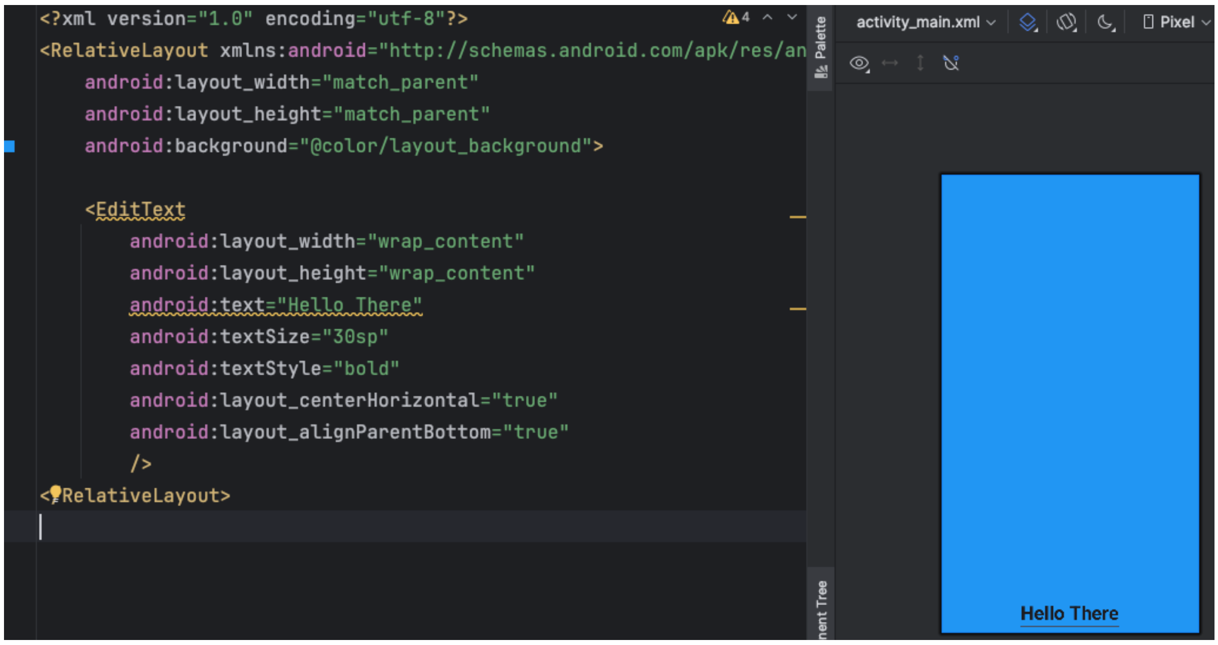
- Next, let's set the text property of our EditText to "Hello There",
- Additionally, lets set textSize to "30sp" and textStyle to Bold. The definition and effect should be as below:



activity_main.xml | colors.xml | MainActivity.kt | build.gradle.kts (:app) | local.properties | settings.gradle.kts (LeszekSiwikLa

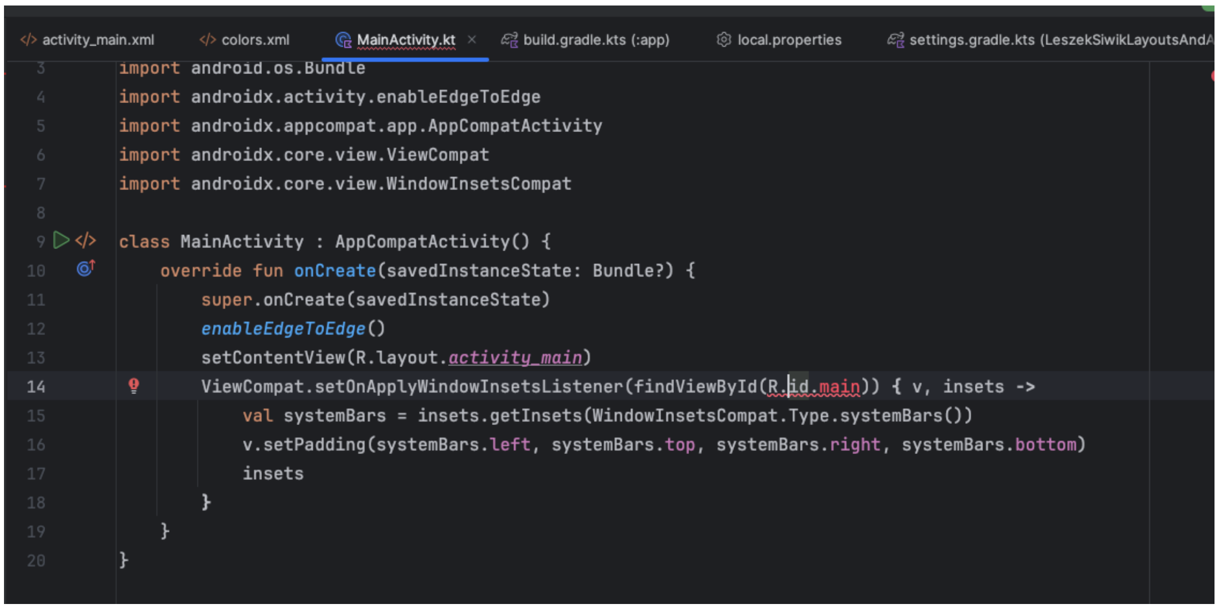
```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/an
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/layout_background">

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello There"
        android:textSize="30sp"
        android:textStyle="bold"
        />
<RelativeLayout>
```

- The “sp” unit that we specified is so-called scale-independent-pixels, which means the abstract pixel size taking into account the physical "density" of the screen of the device on which the application is launched and user preferences (fonts size) as well. More information can be found, e.g. here: https://stackoverflow.com/questions/2025282/what-is-the-difference-between-px-dip-dp-and-sp
- Let's make our label to be centered horizontally (property: android: layout_centerHorizontal = "true") and let’s place it at the bottom of the screen (property: android: layout_alignParentBottom = "true"). So the definition and the effect should be as follows:

- 

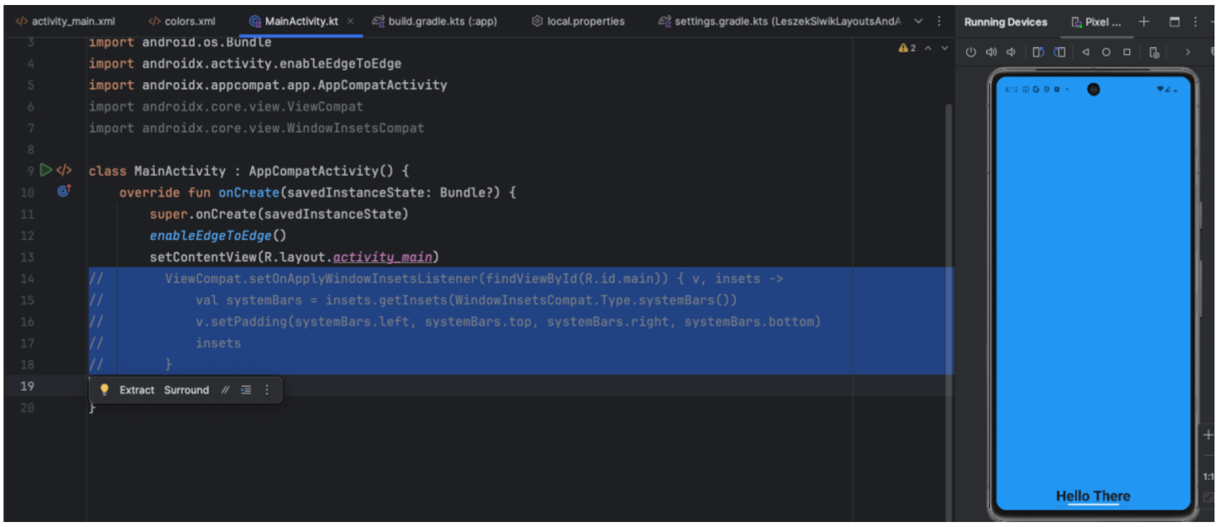
```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/an
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/layout_background">

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello There"
        android:textSize="30sp"
        android:textStyle="bold"
        android:layout_centerHorizontal="true"
        android:layout_alignParentBottom="true"
        />
</RelativeLayout>
```

- Let’s re-build and re-run the app to make sure that everything is fine. If you are getting the error saying that R.id.main is not recognized in MainActivity.kt

- 

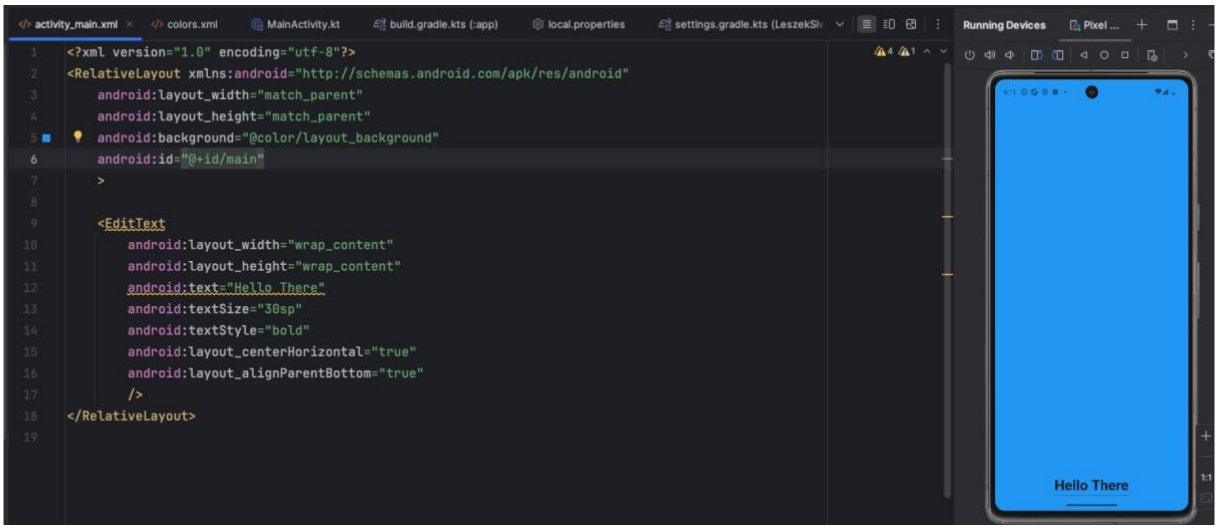
```kotlin
import android.os.Bundle
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_main)
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets ->
            val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)
            insets
        }
    }
}
```

- You may either remove / comment out this insets part:

- 

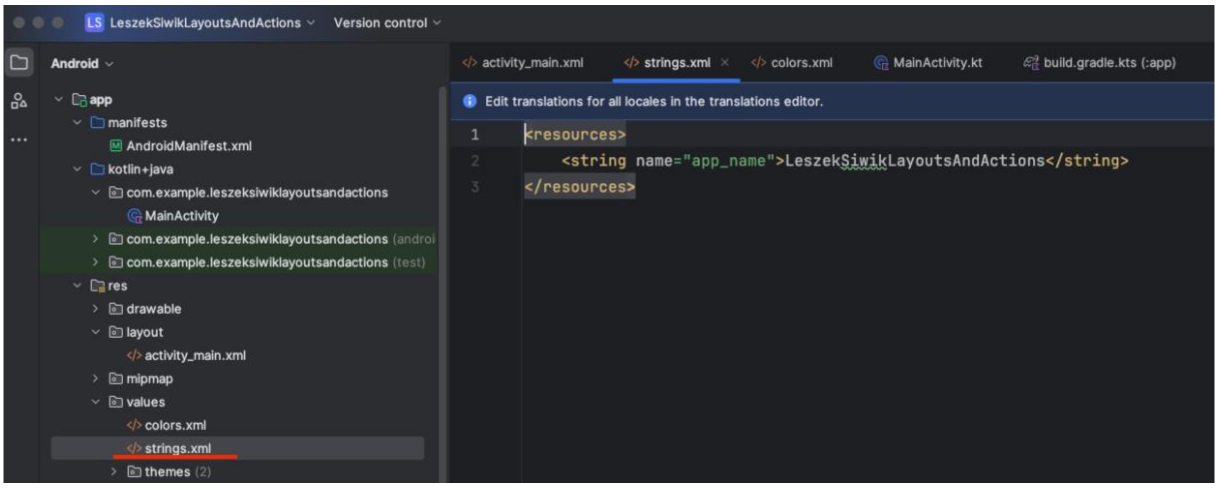
```
import android.os.Bundle
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_main)
//        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets ->
//            val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
//            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)
//            insets
//        }

}
```

- Or generate/set the id to our layout like this:

- 

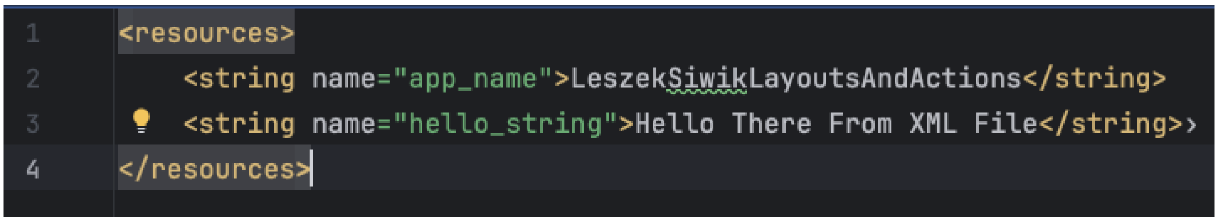
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/layout_background"
    android:id="@+id/main"
    >

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello There"
        android:textSize="30sp"
        android:textStyle="bold"
        android:layout_centerHorizontal="true"
        android:layout_alignParentBottom="true"
        />
</RelativeLayout>
```

- The text appearing in our EditText has been "hardcoded" in the definition of this label. Generally, when creating an Android application the convention is that strings are defined rather as application resources in the res -> values -> strings.xml file:

- 

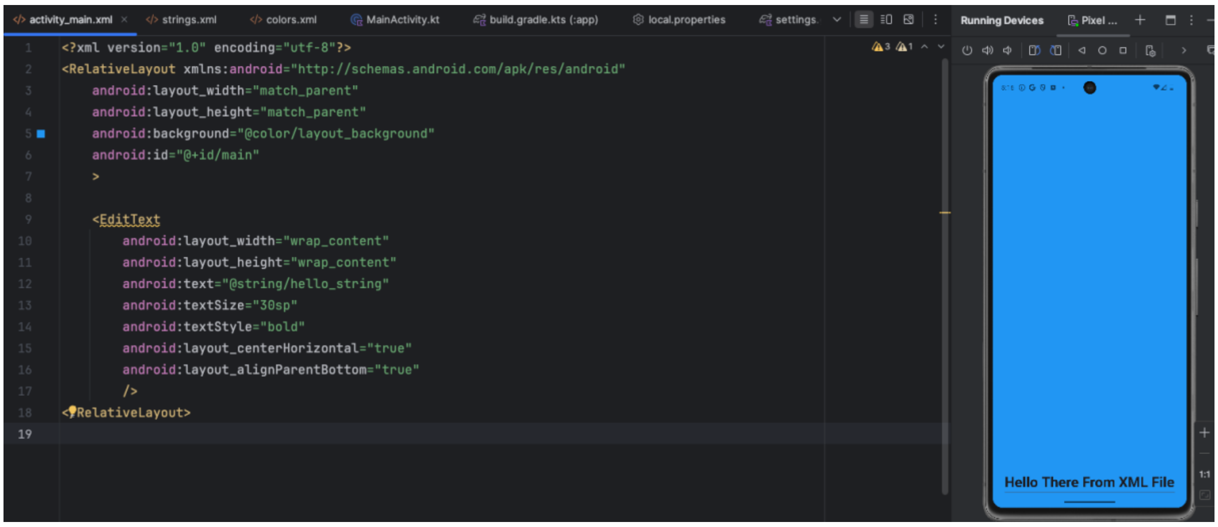
```
<resources>
    <string name="app_name">LeszekSiwikLayoutsAndActions</string>
</resources>
```

- This is to make the application maintenance easier (if we want to change some strings, or facilitate translating the application into various languages etc)
- So, to define our inscription as the app resource let's add a new "string item in stings.xml file.
- Let it be called hello_string and let's set its value to "Hello There from XML file"



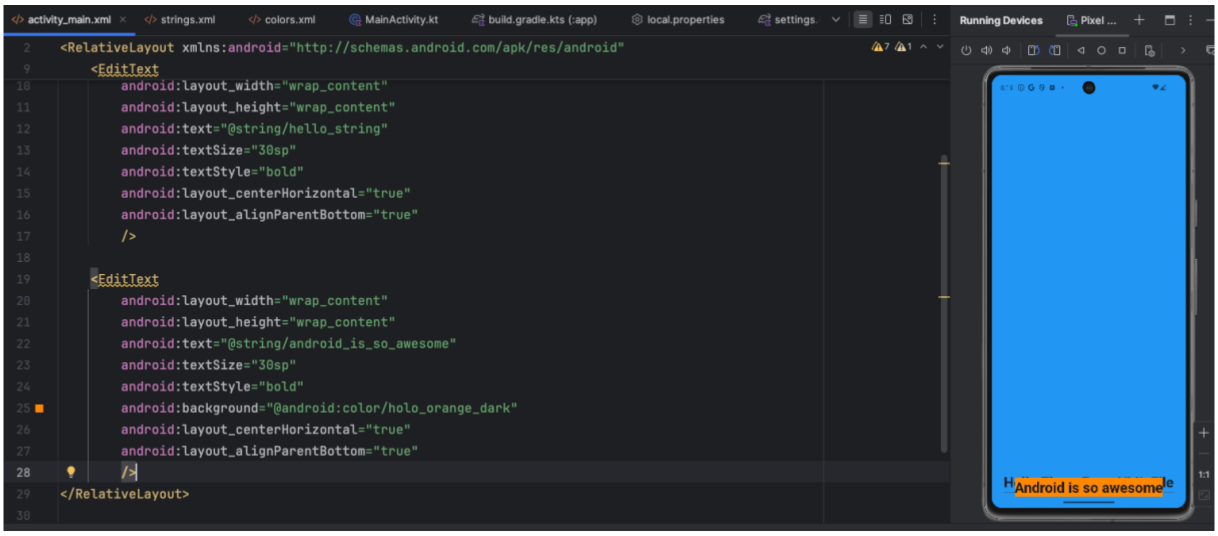
```
<resources>
    <string name="app_name">LeszekSiwikLayoutsAndActions</string>
    <string name="hello_string">Hello There From XML File</string>
</resources>
```

- Now, let's go back to activity_main.xml and let's change the value for the text property from the hardcoded value to @string/hello_string. So the definition and effect should be as below:

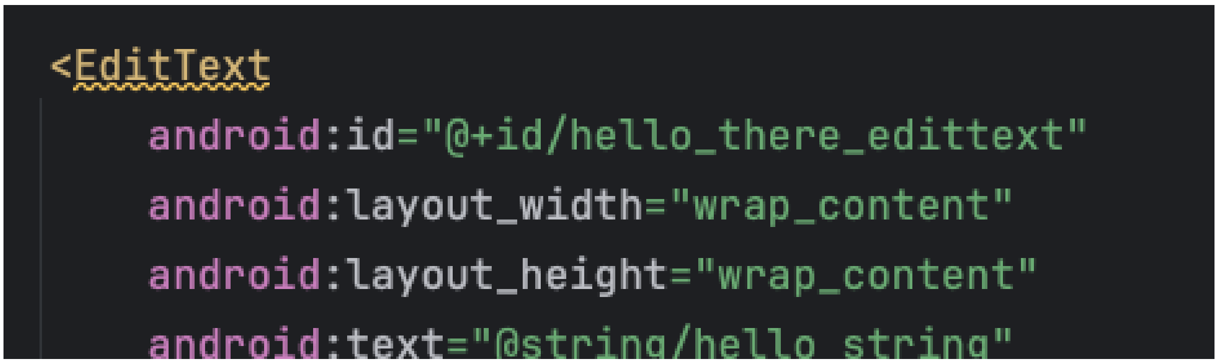
- 

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/layout_background"
    android:id="@+id/main"
    >

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_string"
        android:textSize="30sp"
        android:textStyle="bold"
        android:layout_centerHorizontal="true"
        android:layout_alignParentBottom="true"
        />
</RelativeLayout>
```

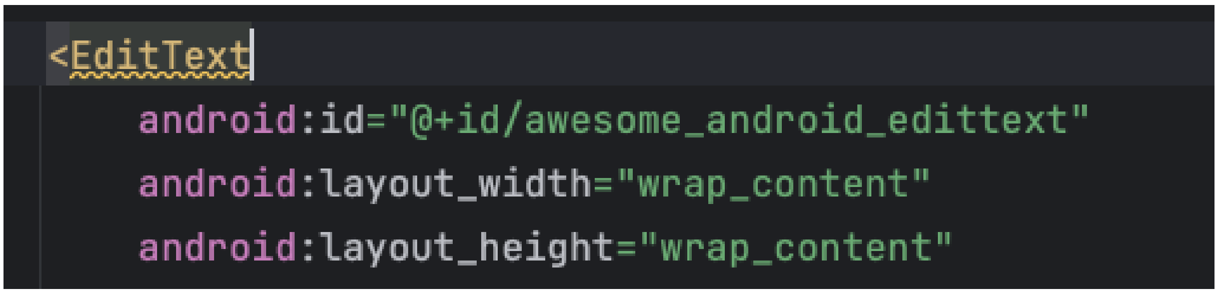
- Let's us now add a second EditText to the layout. Let's set the text inside the new label to "Android is so awesome". Additionally, let's set the background color to holo_orange_dark. We set the rest of the properties as in the previous case
- So now, without any further modifications, the effect will be like

- 

```xml
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_string"
        android:textSize="30sp"
        android:textStyle="bold"
        android:layout_centerHorizontal="true"
        android:layout_alignParentBottom="true"
        />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/android_is_so_awesome"
        android:textSize="30sp"
        android:textStyle="bold"
        android:background="@android:color/holo_orange_dark"
        android:layout_centerHorizontal="true"
        android:layout_alignParentBottom="true"
        />
</RelativeLayout>
```

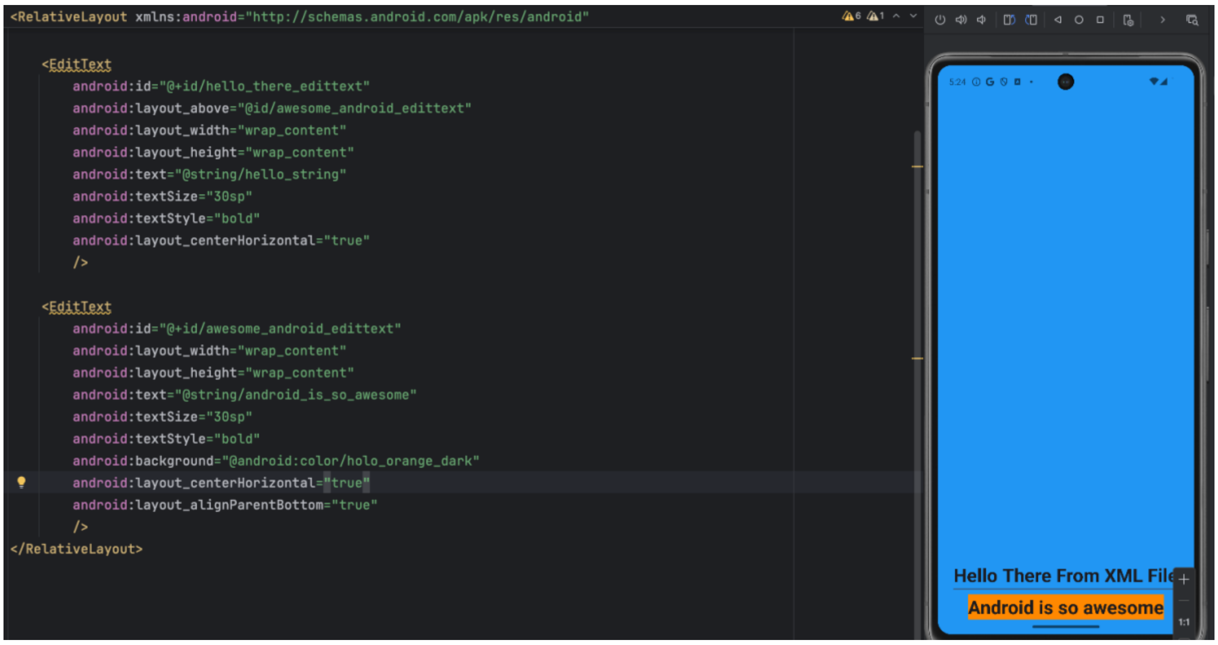
- So, the labels overlap since both are aligned to the bottom of the "parent"
- Let’s fix it now and try to put the labels one under/above the other
- So, first, let’s ad “ids” to both edit texts like
- 

```xml
<EditText
    android:id="@+id/hello_there_edittext"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/hello_string"
```

- 

```xml
<EditText
    android:id="@+id/awesome_android_edittext"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

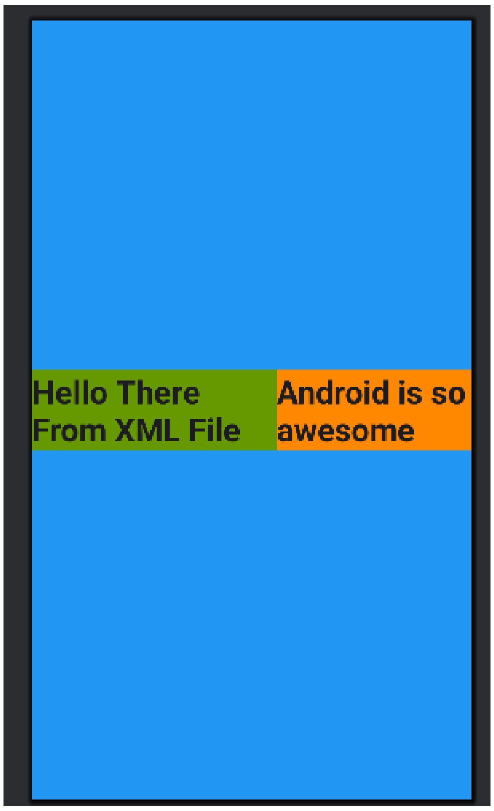
- And now we may put the one above / below the other, e.g. like:



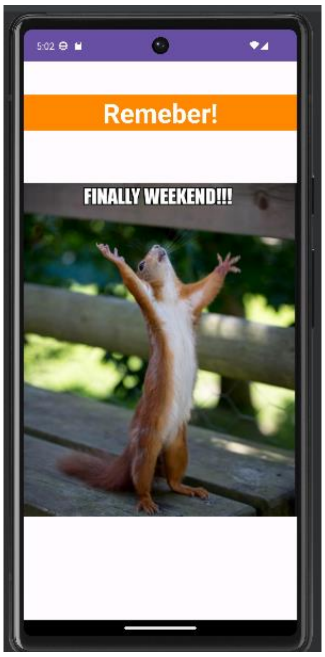
```xml
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

    <EditText
        android:id="@+id/hello_there_edittext"
        android:layout_above="@id/awesome_android_edittext"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_string"
        android:textSize="30sp"
        android:textStyle="bold"
        android:layout_centerHorizontal="true"
        />

    <EditText
        android:id="@+id/awesome_android_edittext"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/android_is_so_awesome"
        android:textSize="30sp"
        android:textStyle="bold"
        android:background="@android:color/holo_orange_dark"
        android:layout_centerHorizontal="true"
        android:layout_alignParentBottom="true"
        />
</RelativeLayout>
```

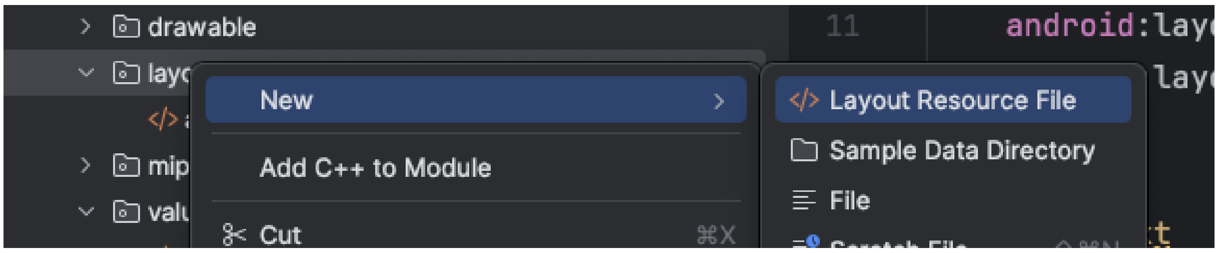
•

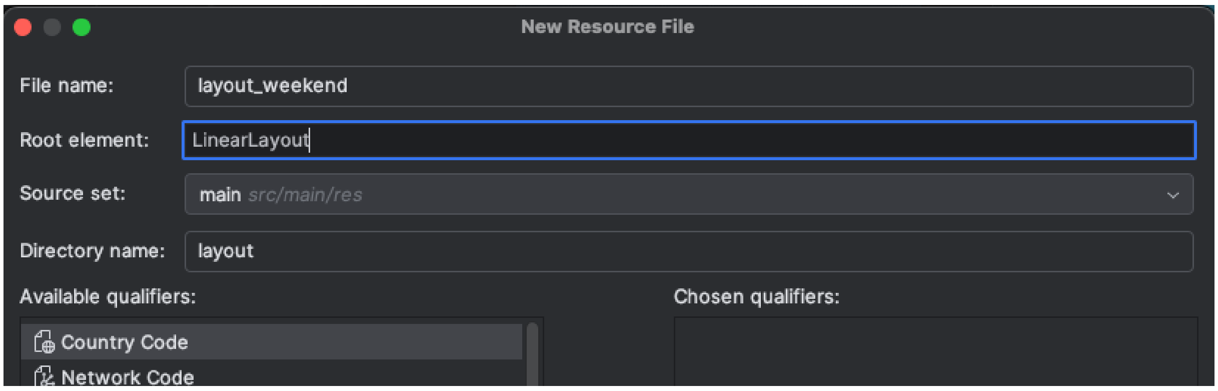
- For practicing, please place both EditTexts in the layout as below:



•

- **Working with new layout and (image) assets/resources**
  - Now, let's play with new resources, and let's prepare the new layout like the one below.
  - 

  - But we will do it in a new/separate layout file. So, right click on the layout folder and then choose New -> Layout Resource File
  - 

  - In the wizard, set the file name to layout_weekend and the root element to LinearLayout
  - 

    New Resource File

    File name: layout_weekend

    Root element: LinearLayout

    Source set: main src/main/res

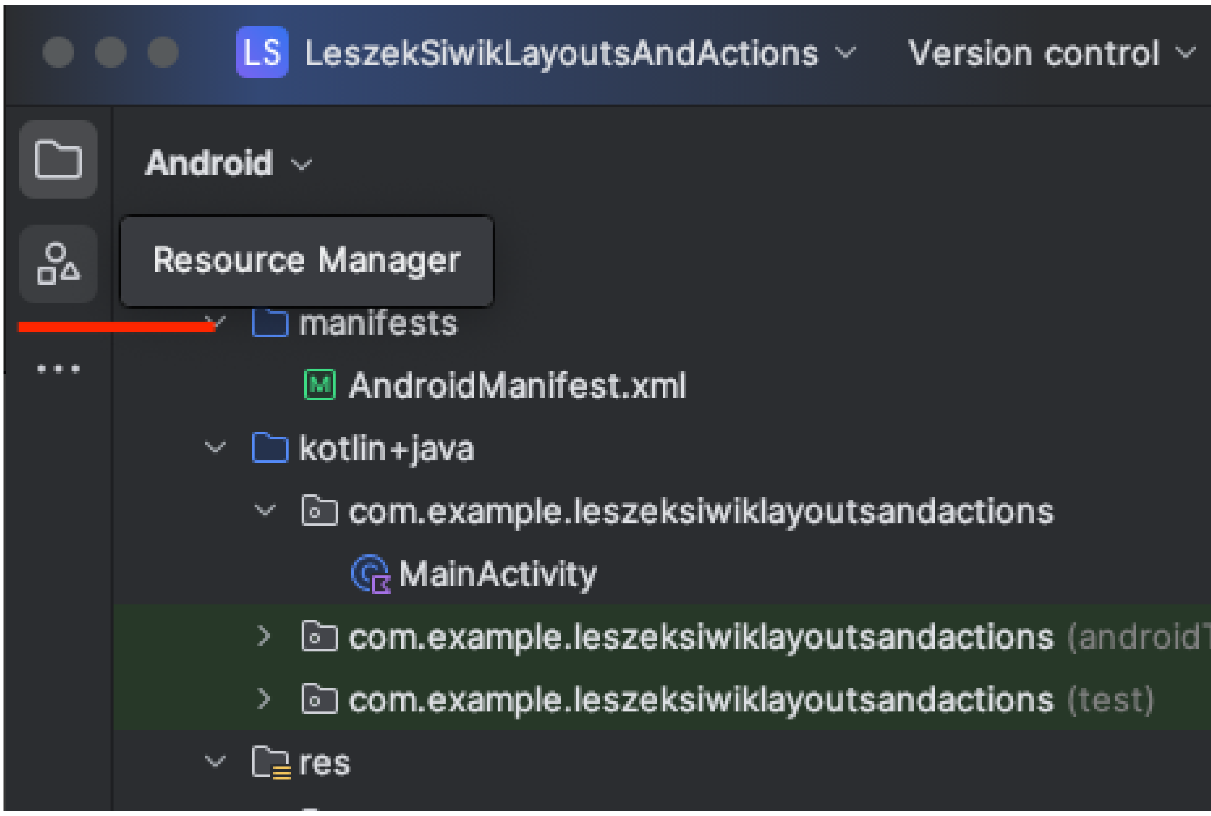
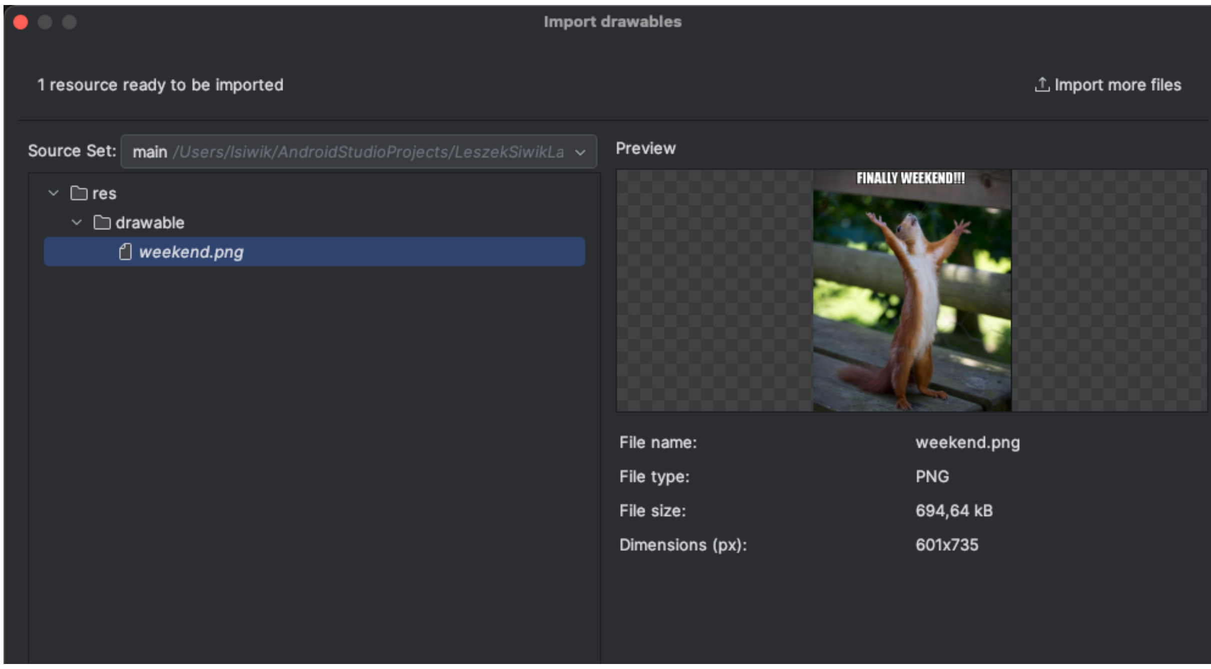
    Directory name: layout

    Available qualifiers:

    Chosen qualifiers:

    Country Code

    Network Code
  - Now, let's compose the layout out of two elements - EditText and ImageView.
  - To set the ImageView content:

- If you want to use the same picture as I did – you can use the following file: https://drive.google.com/file/d/1THpELo2nZM1JGQYVey06hR5diBz-c0KI/view?usp=sharing
- We need to add a picture to the project - so let's go to Resource Manager, Click on "+" and select "Import Drawables" option
- 
LS LeszekSiwikLayoutsAndActions
Version control
Android
Resource Manager
manifests
AndroidManifest.xml
kotlin+java
com.example.leszeksiwiklayoutsandactions
MainActivity
com.example.leszeksiwiklayoutsandactions (android
com.example.leszeksiwiklayoutsandactions (test)
res
- Lets choose the file we want to import ,Click ok, then Next
- 

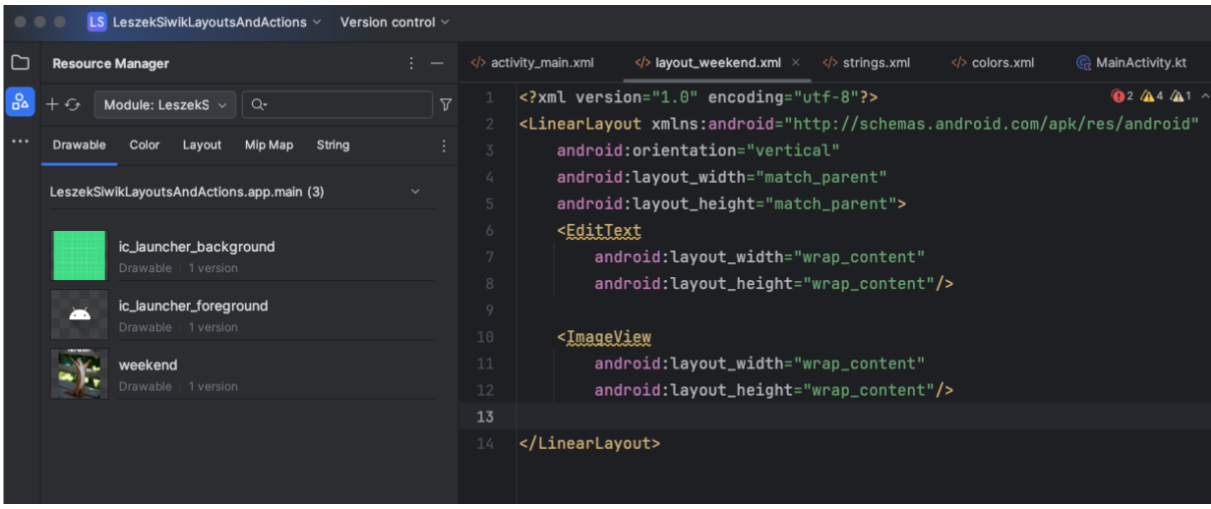
Import drawables

1 resource ready to be imported

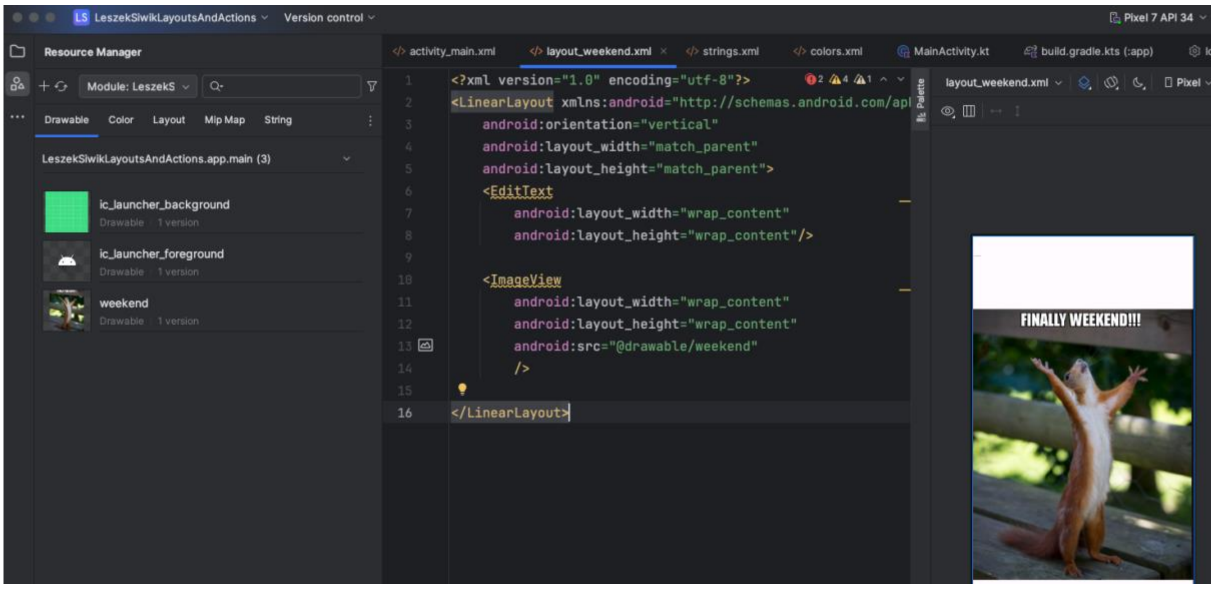
Source Set: main /Users/lsiwik/AndroidStudioProjects/LeszekSiwikLa

res
drawable
weekend.png

File name:	weekend.png
File type:	PNG
File size:	694,64 kB
Dimensions (px):	601x735

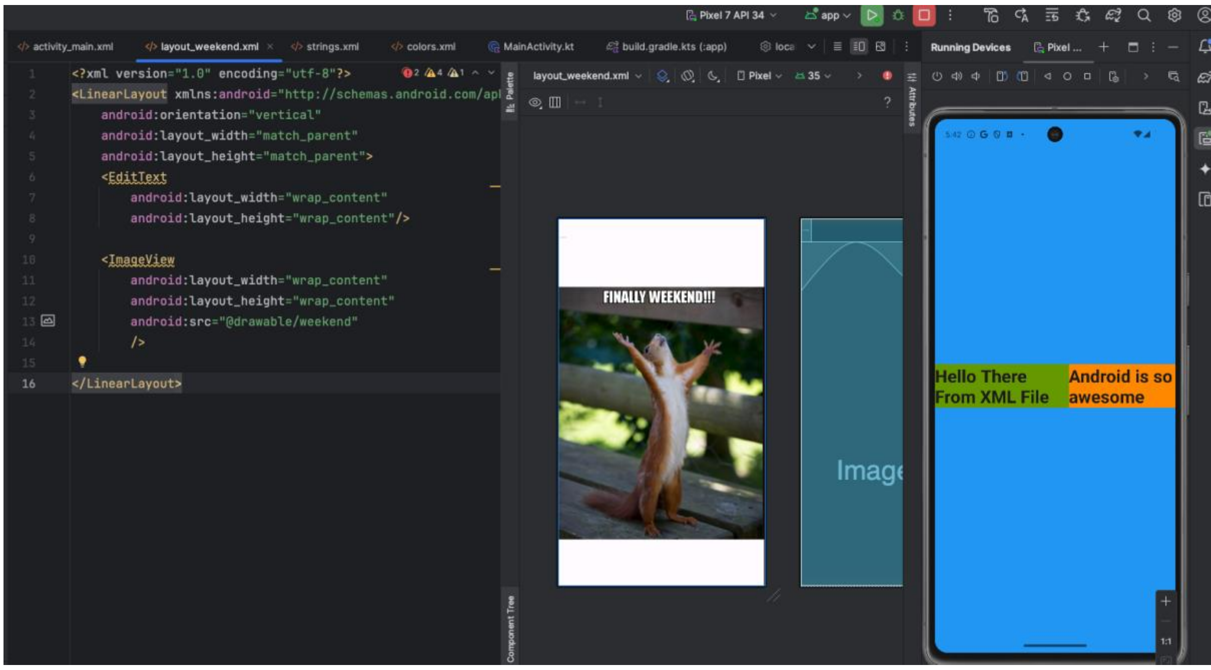
- And finally Import, and now the file should be listed as the Drawable in the Resource Manager
- 

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>

</LinearLayout>
```

- Now, let’s add to our new layout the ImageView element, lets set width to match_parent, height to wrap_content, src to @drawable/weekend and we should see our image added to the layout
- 

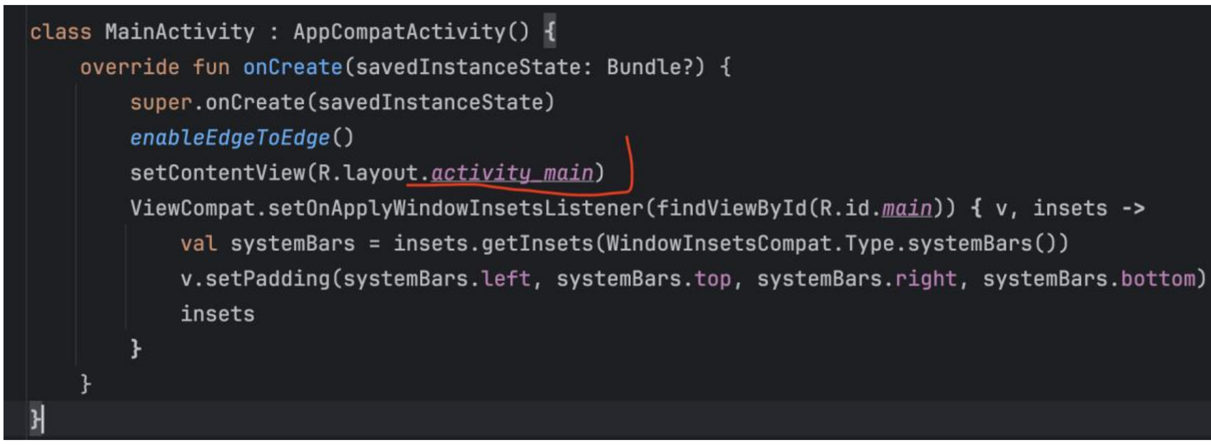
```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apl
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/weekend"
        />

</LinearLayout>
```

- The problem is that when we run the app we see:



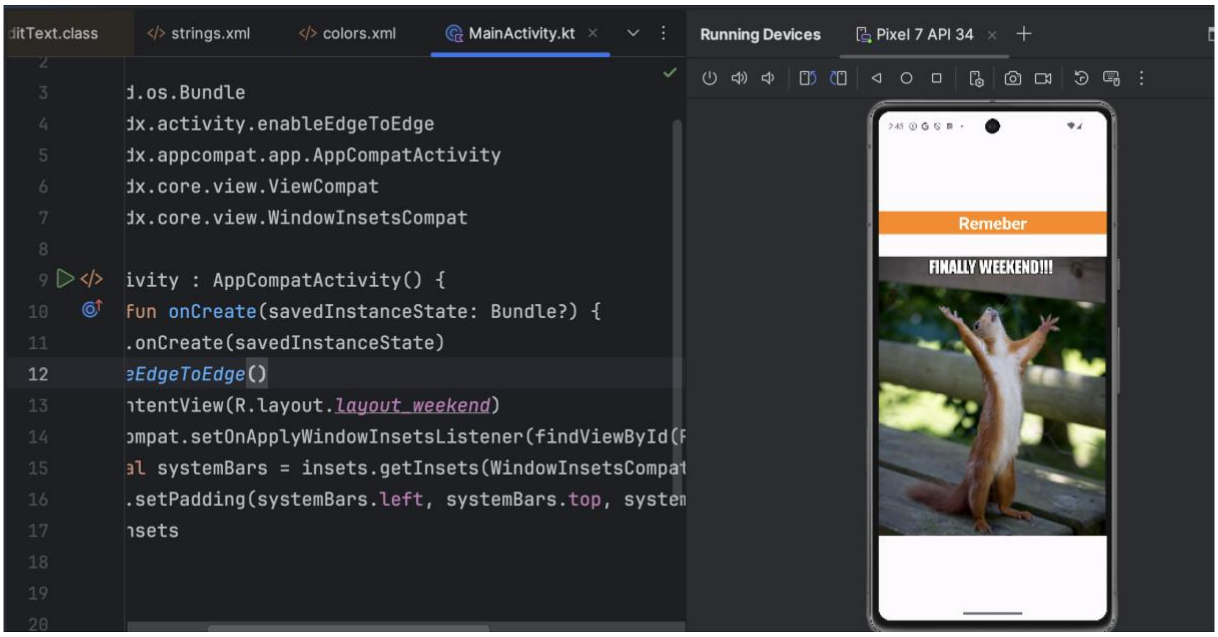
```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/weekend"
        />

</LinearLayout>
```

- which is our activity_main layout.
- This is because we didn't say the application to use the new layout. We have to do it in the MainActivity class:



```kotlin
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_main)
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets ->
            val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)
            insets
        }
    }
}
```

- Here in setContentView method we define which layout should be used – and we need to change it. Also we need to change the layout used for insets accordingly. And after that, it should work as expected, i.e.:

- 

```
d.os.Bundle
dx.activity.enableEdgeToEdge
dx.appcompat.app.AppCompatActivity
dx.core.view.ViewCompat
dx.core.view.WindowInsetsCompat

ivity : AppCompatActivity() {
fun onCreate(savedInstanceState: Bundle?) {
.onCreate(savedInstanceState)
eEdgeToEdge()
ntentView(R.layout.layout_weekend)
ompat.setOnApplyWindowInsetsListener(findViewById(R
al systemBars = insets.getInsets(WindowInsetsCompat
.setPadding(systemBars.left, systemBars.top, system
nsets
```

- When completed pls take code and app screenshots and upload it as of the Layouts task solution on UPEL platform