Instituto Superior de Engenharia do Porto
(ISEP)

Licenciatura em Engenharia de Telecomunicações e Informática (LETI)

Desenvolvimento de Software e Sistemas Móveis (DSSMV)

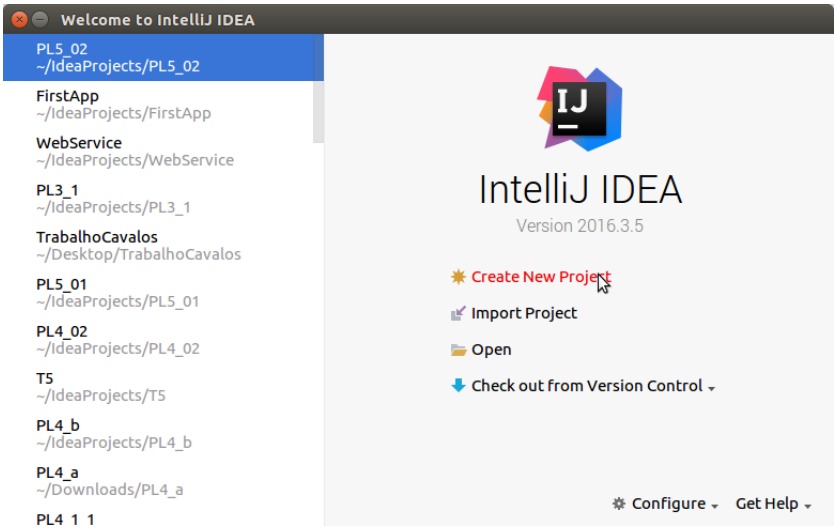
DSSMV: PL: Week 4

Basics on Android Programming

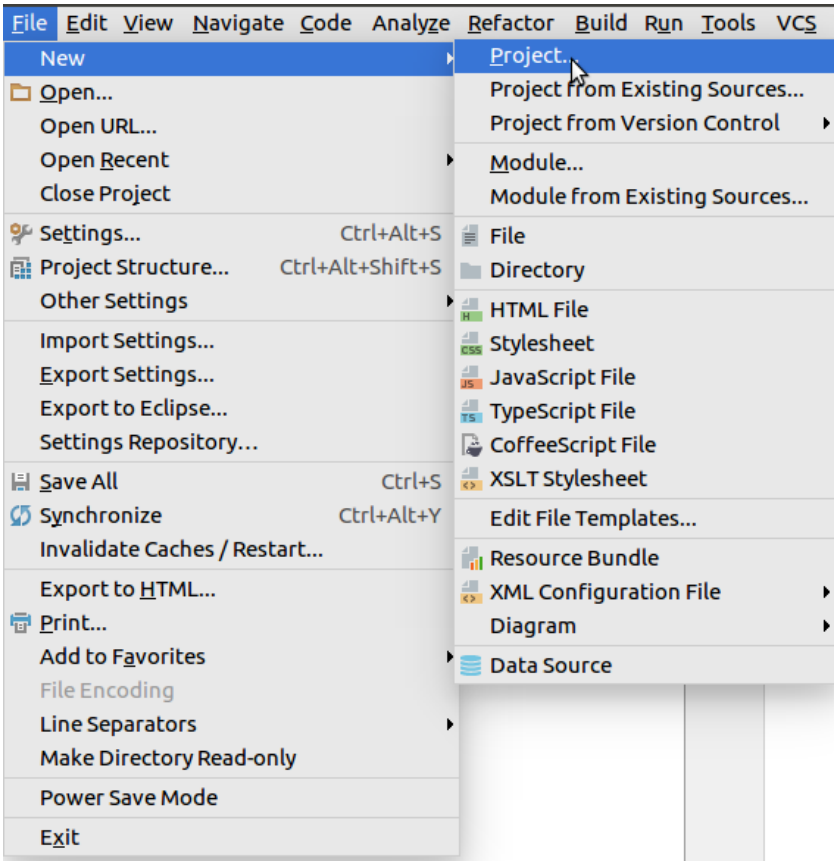
Paulo Baltarejo Sousa and Carlos Filipe Freitas
{pbs,caf}@isep.ipp.pt
2025/26

1 Creating an Android Project

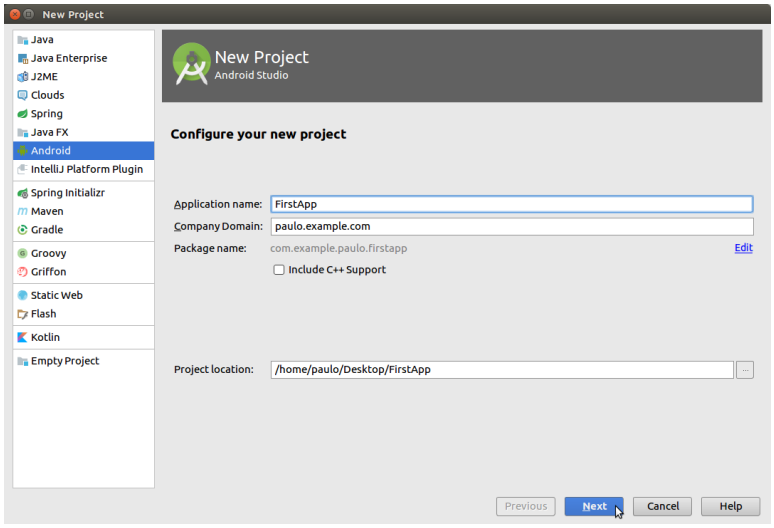
If you did not have a project opened, IntelliJ IDEA shows the Welcome screen. To create a new project, click `Create New Project`.



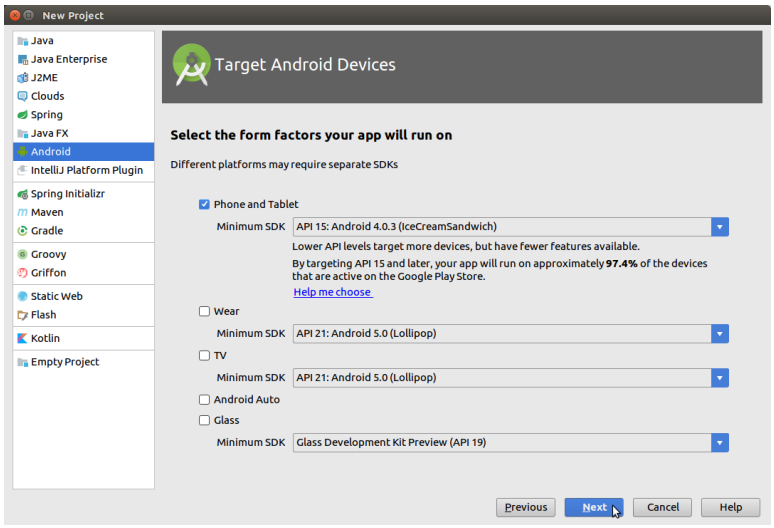
If you had a project already opened, IntelliJ IDEA shows the development environment. To create a new project, click `File > New Project`.



1. The next window lets you configure the name of your app, the package name, and the location of your project. Click `Next`.

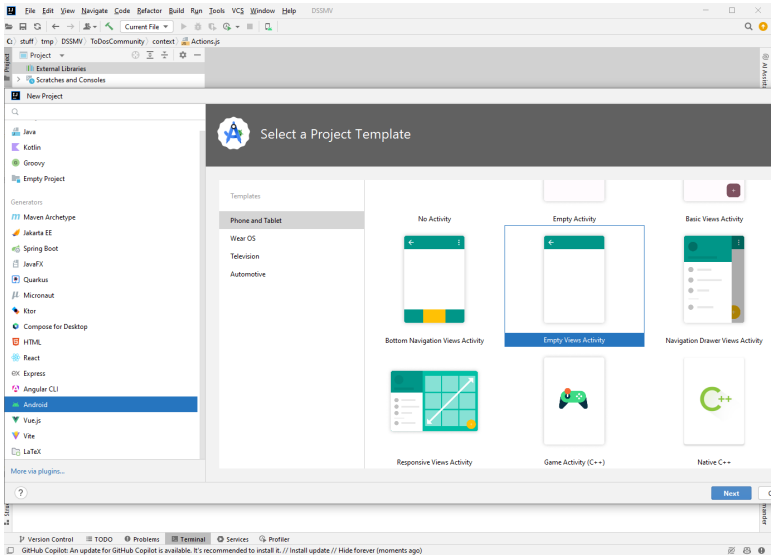


2. Select Form Factors and API Level. The next window lets you select the form factors supported by your app, such as phone, tablet, TV, Wear, and Google Glass. The selected form factors become the app modules within the project. For each form factor, you can also select the API Level for that app. To get more information, click `Help me choose`.



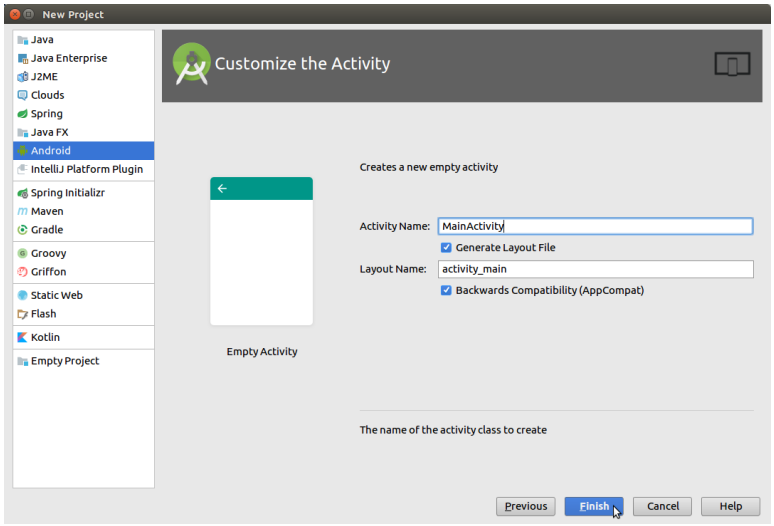
Click `Next`.

3. Add an Activity.



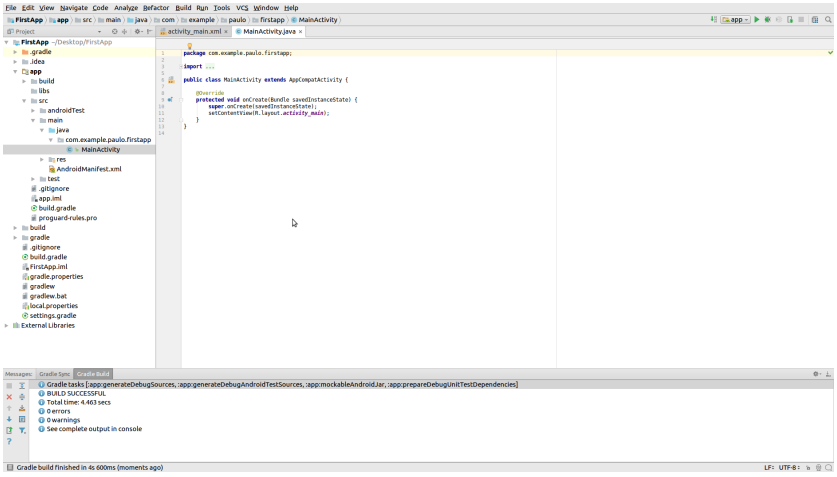
Choose an activity type (in this case select `Empty Views Activity`) then click `Next`.

4. Configure Your Activity.



Enter the activity name, the layout name, and the activity title (it suggests some default names). Then click `Finish`.

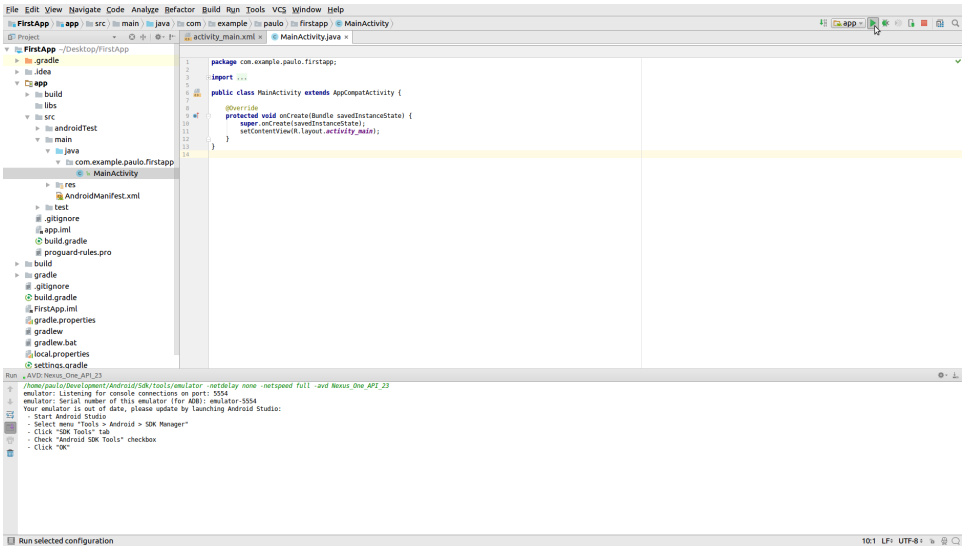
5. Develop your app.



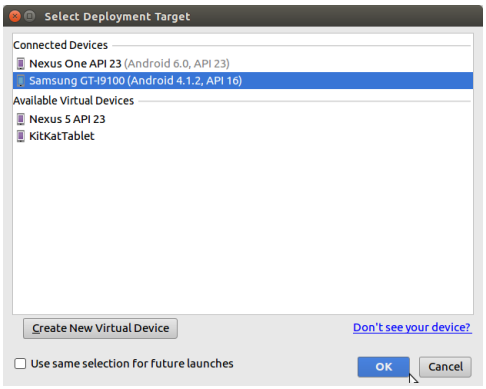
The project creation is done and the development environment is like that.

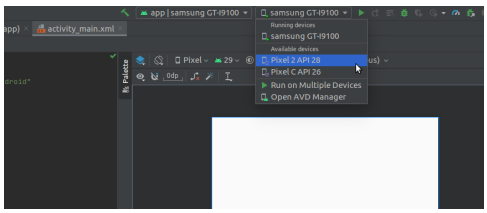
2 Running "First App"

1. In the IntelliJ IDEA, click `Run` from the toolbar.



2. if you have a physical Android device connected and a suitable AVD the `Choose Device` window will appear providing the choice between running within the emulator or on a connected physical Android device:





2.1 Run on a Real Device

If you have a physical Android device, here's how you can install and run your app:

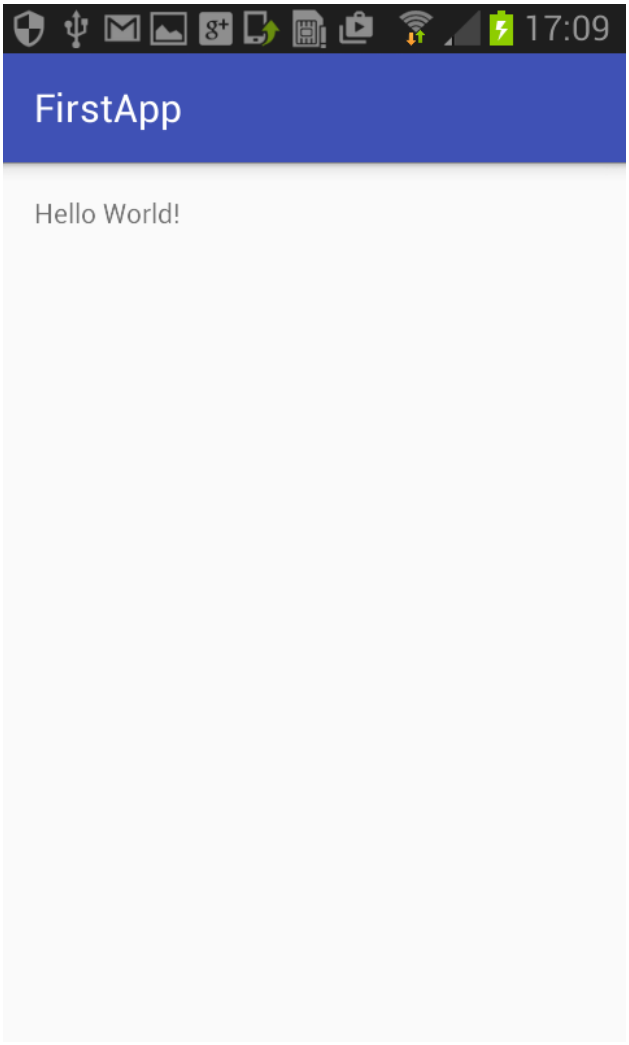
1. Plug in your device to your development machine with a USB cable. You might need to install the appropriate USB driver for your device.

2. Enable USB debugging on your device.

   - On most devices running Android 3.2 or older, you can find the option under `Settings > Applications > Development`.
   - On Android 4.0 and newer, it's in `Settings > Developer` options. Note: On Android 4.2 and newer, `Developer` options is hidden by default. To make it available, go to `Settings > About` phone and tap `Build` number **seven** times. Return to the previous screen to find Developer options.

3. From `Choose Device` window select the real device and click `OK`.

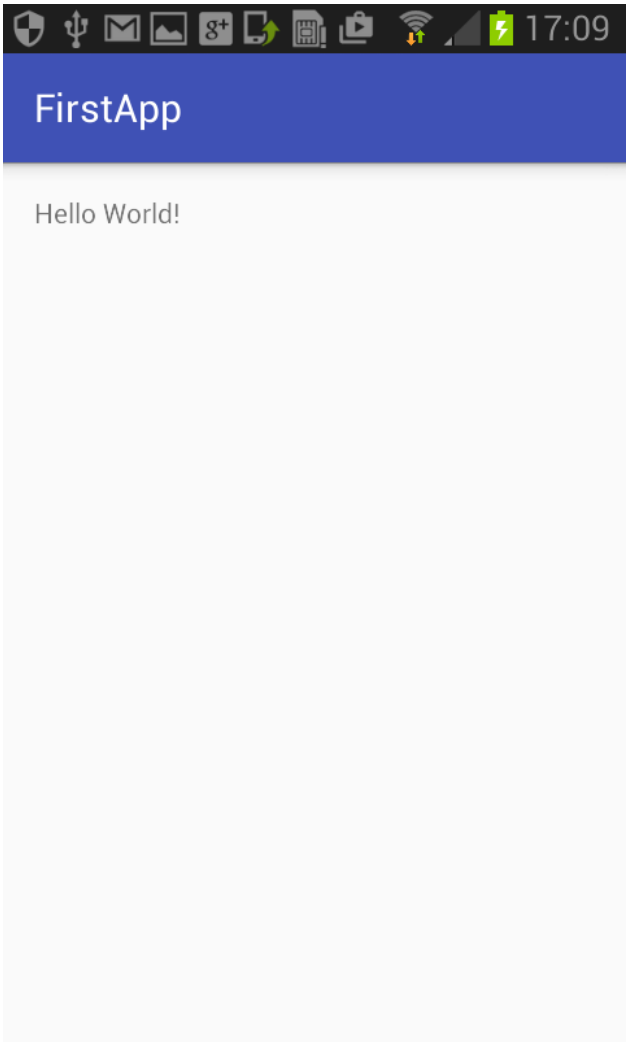
4. IntelliJ IDEA installs the app on your connected device and starts it.



2.2 Run on the Emulator

To run your app on the emulator you need to first create an Android Virtual Device (AVD). An AVD is a device configuration for the Android emulator that allows you to model different devices.

1. From `Choose Device` window select the AVD device and click `OK`.
2. IntelliJ IDEA installs the app on your connected device and starts it.



3 Linear Layout

A Linear layout is a layout that arranges its children in a single column or a single row. The direction of the row is specified by the orientation field.

3.1 Create a Linear Layout

Open the `activity_main.xml` file from the `res/layout/` directory.

The `EmptyActivity` template you chose when you created this project includes the `activity_main.xml` file with a `ConstraintLayout` root view and a `TextView` child view.

First, delete the <`TextView`> element and change the <`ConstraintLayout`> element to <`LinearLayout`>. Then add the `android:orientation` at-

tribute and set it to "vertical". The result looks like this:

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

</LinearLayout>
```

`LinearLayout` is a view group (a subclass of `ViewGroup`) that lays out child views in either a vertical or horizontal orientation, as specified by the `android:orientation` attribute. Each child of a `LinearLayout` appears on the screen in the order in which it appears in the XML.

The other two attributes, `android:layout_width` and `android:layout_height`, are required for all views in order to specify their size.

Because the `LinearLayout` is the root view in the layout, it should fill the entire screen area that's available to the app by setting the width and height to `match_parent`. This value declares that the view should expand its width or height to match the width or height of the parent view.

Add two more `LinearLayout` elements inside of the previous one. The content of `activity_main.xml` file will be like this:

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

    <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal">
    </LinearLayout>

    <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical">
    </LinearLayout>
</LinearLayout>
```

Attribute values explanation:

`wrap_content` : The view should be only big enough to enclose its content (plus padding).

`match_parent` : The view should be as big as its parent.

3.2 Adding Text Fields

To create text fields, add two `Text` elements inside the each element `LinearLayout` elements like this:

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

   <LinearLayout
          android:layout_width="match_parent"
          android:layout_height="wrap_content"
          android:orientation="horizontal">

       <TextView android:id="@+id/textview_label1"
                 android:layout_width="0dp"
                 android:layout_height="wrap_content"
                 android:layout_weight="1"
                 android:paddingLeft="10dp"
                 android:paddingRight="10dp"
                 android:paddingBottom="20dp"
                 android:paddingTop="20dp"
                 android:textColor="#ffffff"
                 android:background="#ff0000"
                 android:text="@string/text_message" />
       <TextView android:id="@+id/textview_label2"
                 android:layout_width="0dp"
                 android:layout_height="wrap_content"
                 android:layout_weight="1"
                 android:paddingLeft="10dp"
                 android:paddingRight="10dp"
                 android:paddingBottom="20dp"
                 android:paddingTop="20dp"
                 android:textColor="#ffffff"
                 android:background="#00ff00"
                 android:text="@string/text_message" />


   </LinearLayout>
   <LinearLayout
          android:layout_width="match_parent"
          android:layout_height="match_parent"
          android:orientation="vertical">
       <TextView android:id="@+id/textview_label3"
                 android:layout_width="match_parent"
                 android:layout_height="wrap_content"
                 android:paddingLeft="10dp"
                 android:paddingRight="10dp"
                 android:paddingBottom="20dp"
                 android:paddingTop="20dp"
                 android:textColor="#000000"
                 android:background="#0000ff"
                 android:text="@string/text_message"/>
       <TextView android:id="@+id/textview_label4"
                 android:layout_width="match_parent"
                 android:layout_height="wrap_content"
                 android:paddingLeft="10dp"
                 android:paddingRight="10dp"
                 android:paddingBottom="20dp"
                 android:paddingTop="20dp"
                 android:textColor="#000000"
                 android:background="#ffff00"
```

```
                android:text="@string/text_message" />
    </LinearLayout>

</LinearLayout>
```

Pay attention: You will get some errors concerning to `@string/text_message`. These errors will be fixed in the next subsection.

Like every `View` object, you must define attributes to specify the `TextView` object's properties.

`android:id` : This provides a unique identifier for the `View`, which you can use to reference the object from your app code, such as to read and manipulate the object.

The at sign (@) is required for referring to any resource object from XML. It is followed by the resource type (id in this case), a slash, then the resource name (`text_message`).

The plus sign (+) before the resource type is needed for defining a resource ID for the first time.

`android:layout_width` **and** `android:layout_height` : The `wrap_content` value specifies that the view should be only as big as needed to fit the contents of the view. The `match_parent` value specifies that the element should be match the size of its parent.

`android:layout_weight` : This attribute assigns an "importance" value to a view in terms of how much space it should occupy on the screen. A larger weight value allows it to expand to fill any remaining space in the parent view. Child views can specify a weight value, and then any remaining space in the view group is assigned to children in the proportion of their declared weight. Default weight is zero.

`android:textColor` : Text color.

`android:background` : Background color.

`android:text` : Text to display. Instead of using a hard-coded string as the value, the `"@string/text_message"` value refers to a string resource defined in a separate file. Because this refers to a concrete resource (not just an identifier), it does not need the plus sign.

3.3 Add String Resource

When you need to add text in the UI, you should always specify each string as a resource. String resources allow you to manage all UI text in a single location, which makes it easier to find and update text.

By default, your Android project includes a string resource file at `res/values/strings.xml`. Add a new string named `"text_message"` and set the value to "Text message."

The result for `strings.xml` looks like this:

```
<resources>
    <string name="app_name">Linear Layout</string>
    <string name="text_message">Text message</string>
</resources>
```

3.4 Running App

1. To run the app from Android: Click `Run` from the toolbar. In the `Choose Device` window that appears, select a device/avd and click `OK`.