



ISEL
INSTITUTO SUPERIOR DE
ENGENHARIA DE LISBOA

Desenvolvimento de Aplicações Móveis Mobile Application Development DAM

Tutorial 1 - Hello World

Pedro Fazenda
CEDET - E.2.16.4
pedro.fazenda@isel.pt

António Teófilo
ADEETC - C.2.4
antonio.teofilo@isel.pt

Abstract

This tutorial presents the Android development and execution environments and the development of a Hello World Android application.

Deadline: 12th of March 2020



Hello Mobile Android World!!

24 of February 2020

1 Introduction

This tutorial starts with an introduction to Android, then it proceeds with the download and installation of the official Android Integrated Development Environment (IDE) that is the Android Studio. The tutorial proceeds with the configuration and update of the necessary tools. Thereupon, it requires the development of a simple “Hello World” application.

Follows a link that should be consulted during the development of Android applications.

Android Developers <https://developer.android.com/guide>

Do not miss the tutorial deadline!!!

All tutorials are expected to be finished and presented to the teacher during class hours. The resulting projects must be completed, presented, evaluated and uploaded to the course page on the Moodle platform in class, until the deadline date shown on the cover page. After each deadline date the final grade, for the project, will have **a penalty value of 5/20 for each passing week/Monday** until the project is presented and evaluated by the teacher in class. This condition applies regardless if the project was submitted or not on the Moodle platform.

2 Introduction to Android

Android OS was originally created inside Android Inc. by Andy Rubin in the early 21st century as a mobile operating system. In 2005, Google acquired Android Inc. and made Andy Rubin the director of Google’s mobile platforms.

Android OS is an operating system supported by the **Linux Kernel**. That’s why Android devices are essentially Linux computers. The project responsible for developing the Android system is called “Android Open Source Project (AOSP)” and is led by Google.

The **Dalvik Virtual Machine (DVM)** is the executor of Android applications. The compilation of Android code generates Dalvik Executable format files with extension **.dex**. These files, plus resources and the manifest files are grouped in **.apk** files. One **.apk** file is one Android application ready to be installed.

There are four (plus one - Layouts) main components in Android applications (see <https://developer.android.com/guide/components/fundamentals>):

- **Activities** - (organizational layer) an Activity supports a screen for user interaction;
- **Layout Containers** - (presentation layer) a Layout is a container for user interface elements in a screen;
- **Services** - (background processing layer) a Service is a component that runs in background, providing services for Activities;

- **Broadcast Receivers** - (notification layer) a Broadcast Receiver is a component that allows the Activities to receive notifications from the system (and react to them);
- **Content Providers** - (storage layer) a Content Provider is a component that store and provide data for Activities.

There are also some important helper classes:

- Android **Fragments** - they are reusable parts of the user interface;
- Android **Intents** - they enable communication between components.

These elements will be introduced in more detail when they are used. For now, we will focus on Android **Activities** and **Layouts**.

An **activity** is a screen, in an application, providing the context for one user interaction. An application may have multiple activities, providing multiple contexts for the user. This is the way used to spread all the information, of the application, in the reduced size screens of mobile devices. An activity has one Layout Container to join the user interface elements. One activity can also change its Layout Container and provide a mutable user interface.

A **Layout Container** groups design elements, called **Widgets**, to build a screen. The design elements can be user interface elements for text, graphics, 3D content, digital video, menus, animation, ...

Layouts Containers are built based on **ViewGroup** class and **Widgets** based on **View** class.

For Android graphics such as images or animations the term **Drawables** is used.

The Event Handling capabilities of Android User Interface elements offers a way to applications react to user actions or other source of events.

3 Preparatory actions

To enable the development and execution of Android applications this tutorial presents the installation of IDE, the development libraries (Software Development Kit (SDK)) and emulation capability.

3.1 Download and installation of Android Studio

1. Download the Android Studio from the following link:

<https://developer.android.com/studio>

The Android Studio package includes all the tools necessary for the development of android applications:

- **Android Studio IDE** - to edit, run, debug Android code
- **Android SDK and Tools Manager** - to install Android SDKs and tools

- **Android Emulator** - to execute Android apps in Android emulated devices
- **Android Virtual Device (AVD) manager** - to create and manage AVDs.

Because the programming language used is Java you must have installed the Java Development Kit (JDK) and the Java Runtime Environment (JRE).

2. Execute the “android-studio-ide-xxx.xxxxxxx-windows.exe” file obtained with the previous download to install the Android Studio package.
3. Run the Android Studio IDE. The first time you execute Android Studio you can choose whether you want to import applications or settings from a previous version as shown in Figure 1.

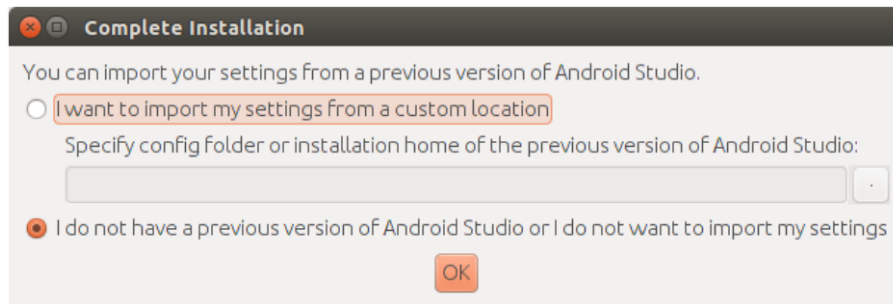


Figure 1: Android Studio after being executed for the first time.

4. After clicking the “OK” button you will go through several configuration steps in order to install Android Studio (see Figures 2 to 5).

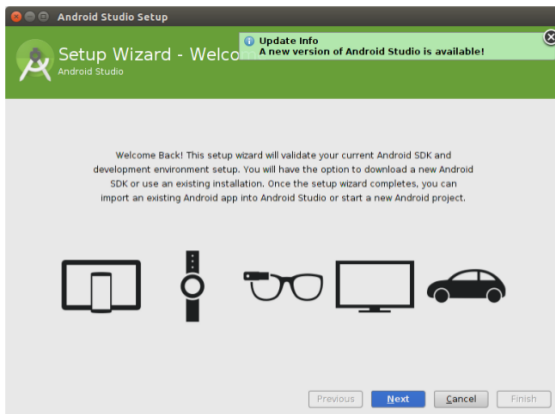


Figure 2: SETUP - Start Setup Wizard.

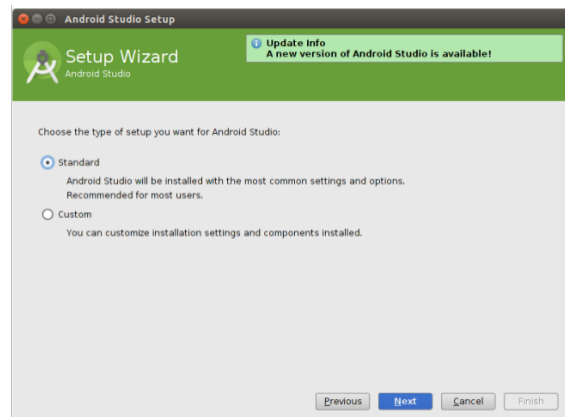


Figure 3: SETUP - Option “Standard”.

3.2 Installation of Android development libraries (SDKs)

There are several versions of the Android platform. They have a version number and name and an API number, like Android 5.0 Lollipop (API 21), Android 8.0 Oreo (API 26) and Android 10 Q (API 29). To develop code for one platform it is needed its libraries, called Software Development kit (SDK). So, to develop code for Android 8.0 it is needed its Android SDK platform and tools. For the management of the SDKs installed, or to install new ones, there is the Android SDK Manager. It is recommended to have installed

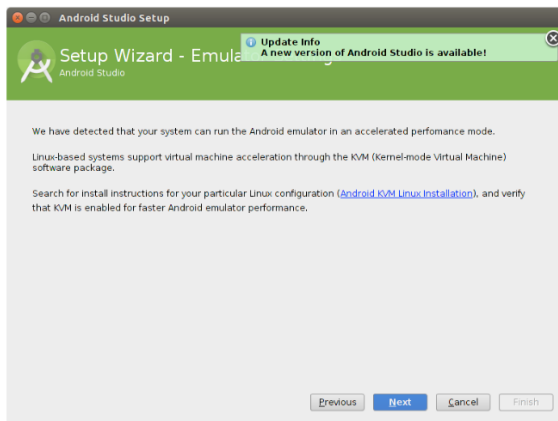


Figure 4: SETUP - Some PC's may have this option to speed up the emulator.

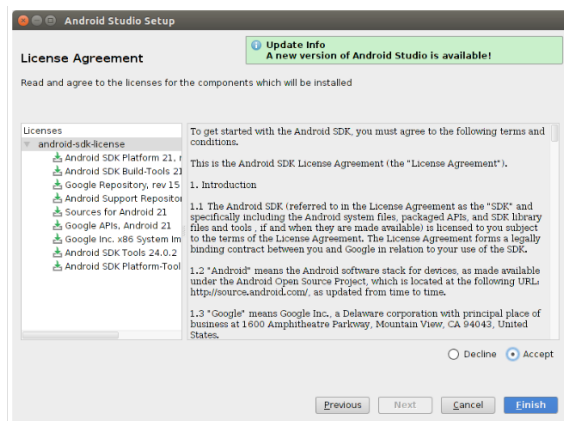


Figure 5: SETUP - Finishing the installation.

the Android 10 Q (API 29) and another one like Android 6.0 Marshmallow (API 23) or the highest one that is supported by the student phone.

1. Open the Android SDK Manager: it can be done in the “Welcome to Android Studio” window (when there is no Project opened), by selecting the “**Configure**” drop down list and choosing **SDK Manager** (see Figure 6) or when we are working on a project by opening the menu **Tools > SDK Manager** or by selecting the **SDK Manager icon** in the toolbar (see Figure 7).

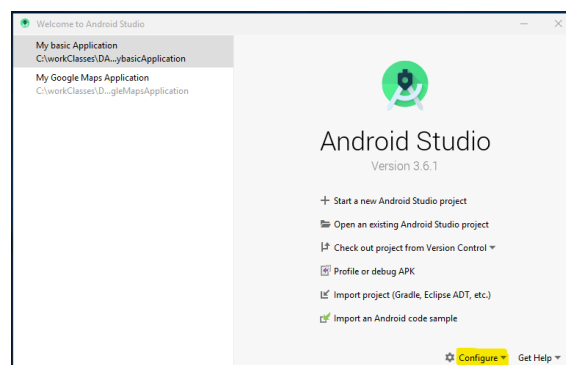


Figure 6: Configure in Welcome window.

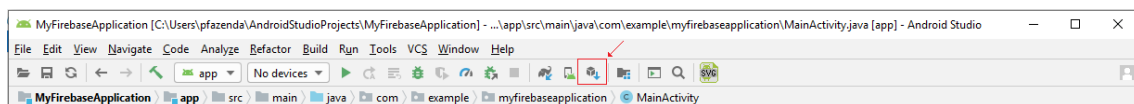


Figure 7: SDK manager icon (red element in the toolbar).

2. In the SKD Manager panel (see Figure 8) select the Android version you wish to develop and are not installed by checking them and then press the “OK” button. Select “Show Package Details” check-box to see in detail the components that are part of each version of the Android API.

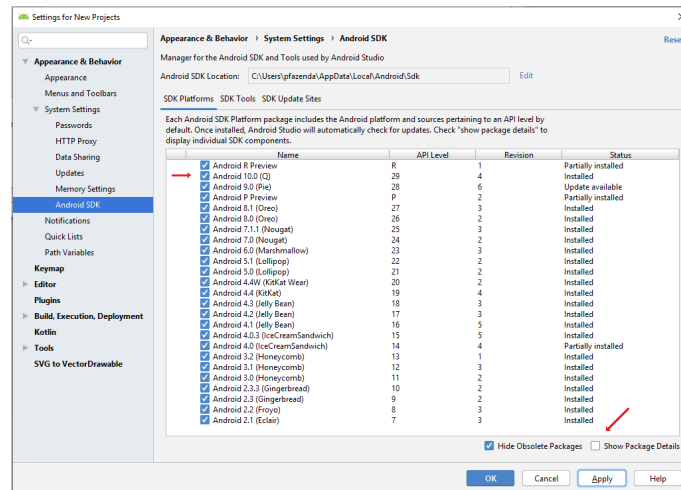


Figure 8: SDK Manager - The “SDK Platforms” tab is used to install Android API versions and the “SDK Tools” tab is used to install development tools.

3.3 Checking for Updates

It is recommended that the Android Studio and all the platforms and tools to be updated. After installation is a good moment to check for the latest updates.

1. Check for updates: in the “Welcome to Android Studio” window, when there is no Project opened, by selecting the “**Configure**” drop down list and choosing **Check for updates** (see Figure 6) or when we are working on a project by opening the menu **Help** > **Check for updates...** (see Figure 9) (do not install Beta versions).

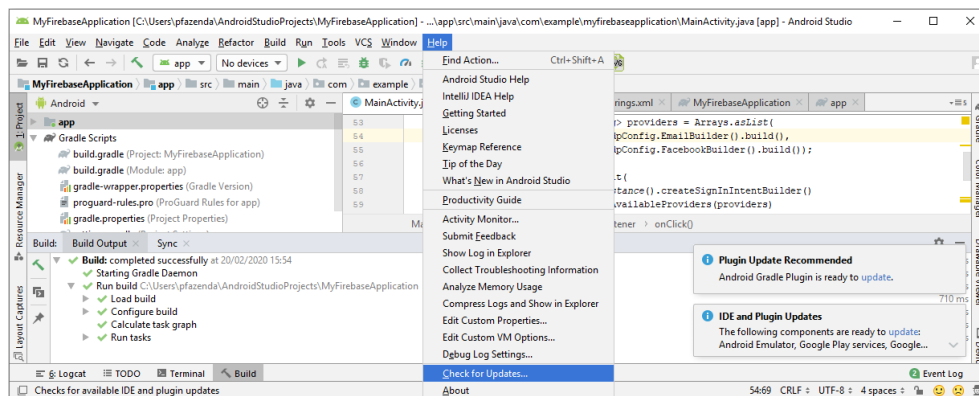


Figure 9: Android Studio - “Android Studio” menu and option “Check for Updates”.

3.4 Creating Android Virtual Devices (AVDs)

An Android Virtual Device (AVD) is a piece of software that behaves similarly to one physical device. AVDs allows to test Android applications in different Android versions and hardware profiles without using physical devices. The AVD Manager is a tool that allows the creation of AVDs of the following types of devices: Phone, Tablet, TV, Automotive and Wear OS. It contains several hardware profiles but it can create or import other ones. You should have at least two AVDs: one **Pixel 3** smartphone and one **Pixel C** tablet. The AVD Manager gives the possibility to create AVDs for a specific hardware profile and for a specific Android platform version. Therefore, to have an AVD we need to

choose one hardware profile and one Android platform system image. You should select the latest Android API version that is compatible with your device so that you can run applications on the virtual device and on your device without having to make changes to the code.

1. Open the AVD Manager either in “Welcome to Android Studio” window (when there is no Project opened) by selecting the “**Configure**” drop down list and choosing **AVD Manager** (see Figure 6) or when we are working on a project by opening the menu **Tools > AVD Manager** or by selecting the **AVD Manager icon** in the toolbar (the one at the left of SDK Manager in Figure 7).

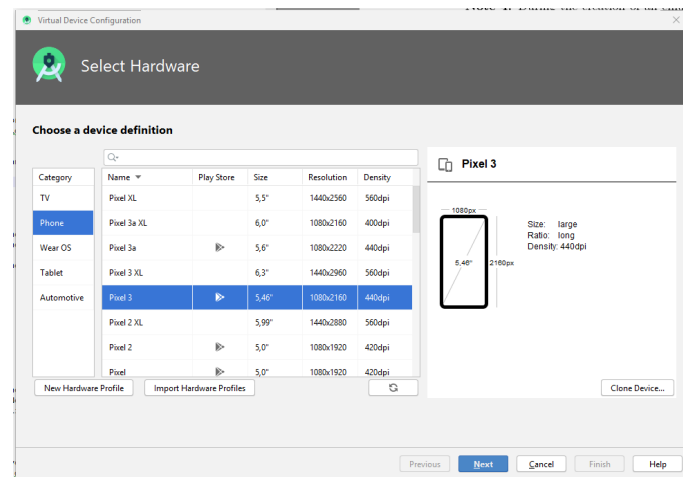


Figure 10: Virtual Device: choosing hardware profile.

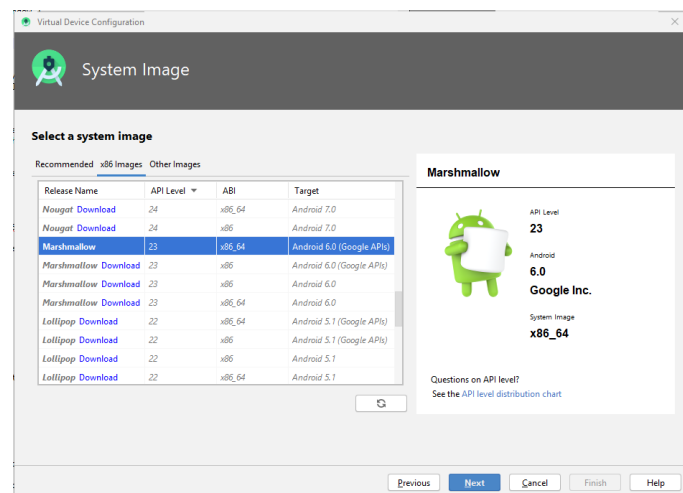


Figure 11: Virtual Device: choosing system image.

2. Create two AVDs, one for a **Pixel 3** smartphone and another for a **Pixel C** tablet. For each one, start by clicking in the “Create Virtual Device” button. In the AVD hardware profile panel, see Figure 10, select the hardware profile for the desired AVD and then click the “Next” button. In the AVD system image panel, see Figure 11, select the Android platform version, hardware architecture (x86, ...) and the supported Google services (Google Play, ...), download it if necessary and then click the “Next” button. Leave the remaining parameters, for now, set to “default” and finish.

3. Start the Pixel 3 AVD. After you create the emulators, you can see them in the AVD Manager panel (see Figure 12). On that list click the “play” button on the **Pixel 3** smartphone emulator to launch it.

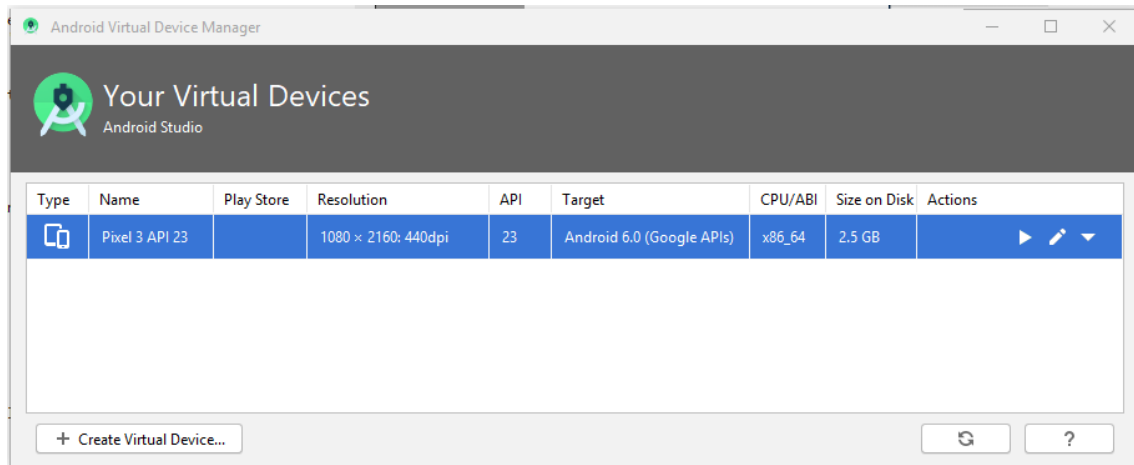


Figure 12: AVD Manager: list of existing AVDs.

Note 1: The first launch of an AVD can take up to 10 minutes, depending on the computing power of the PC. If you stop an AVD during the launch process, the AVD may become corrupted.

Note 2: The AVD after being started should not be stopped/closed during development. If you make changes to the code, just do the “re-deploy” of the application in the AVD.

Note 3: During the creation of an emulator you can choose between the options: “Snapshot” or “Host GPU”. If you choose the first option, the emulator, when stopped, is like hibernated (saved a copy of its memory state), and a subsequent start will be fast. If you choose the second option, rendering is faster because the PC graphics card is used.

Note 4: When choosing the Android API version, when creating the emulator, you can select a version where the AVD image is based on an ARM CPU architecture or an Intel CPU architecture. An AVD that is based on an Intel system is much faster if it runs on Intel hardware, because in that case it is not necessary to convert the ARM CPU instructions to the Intel architecture of the PC.

3.5 Prepare your phone for your Apps

1. **Enable non-Play Store Apps instalation.** Go to **Settings > Security** and then select **Unknown sources**. Selecting this option will allow you to install apps outside of the Google Play store. From Android 8.0 on, we must go to **Settings > Apps & notifications > Advanced > Special app access > Install unknown apps** and then select **Allow from this source**.
2. **Enable “Developer options”.** To install and debug applications in your phone you must activate its **Developer options**. To activate it, in your phone, go to

Settings > **About phone** and tap **Build number** seven times.

3. **Enable debugging.** App installation and debugging is done by **Android Debug Bridge (ADB)**. The ADB lets you to install and debug applications in the devices and provides you with a command line shell that you can use to send commands to the device. The ADB can connect with the device by an USB cable or by WiFi. After **Developer options** activation, you must enable debugging by Android Debug Bridge (ADB). To do it, go to **Settings** > **Developer options** and enable **USB debugging**.

See <https://developer.android.com/studio/command-line/adb>.

4 Laboratory Work

4.1 Create a “Hello Mobile World” app

1. Create an Android project selecting **File** > **New** > **New Project** to open a dialogue box with several options (see Figure 13). Choose the **Empty Activity** option and click **Next**. This option will create a basic application with one activity.

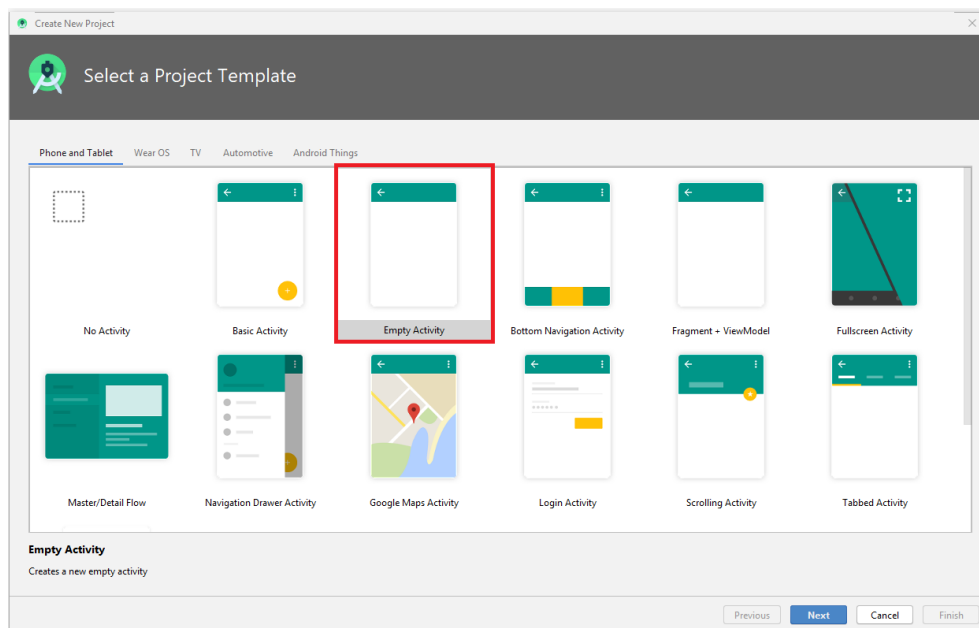


Figure 13: Choose your project - Basic Activity

2. Type the name of the application: **Hello World**. Write the name of the package: “dam_a60000.helloworld” (see Figure 14) (a60000 should be replaced with your student number). **You should use that base package name for all your projects.** Set the path where the project should be saved, as: “dam-path\code\HelloWorld”. Where dam-path should be the directory of DAM. Select Java language and the API 22: Android 5.1 (**Lollipop**) version for the Minimum API level. Click **Finish** to generate the project.

Note 5: In the selection of Minimum API level the configuration wizard will show, for each API version, the estimated percentage of devices that supports the version

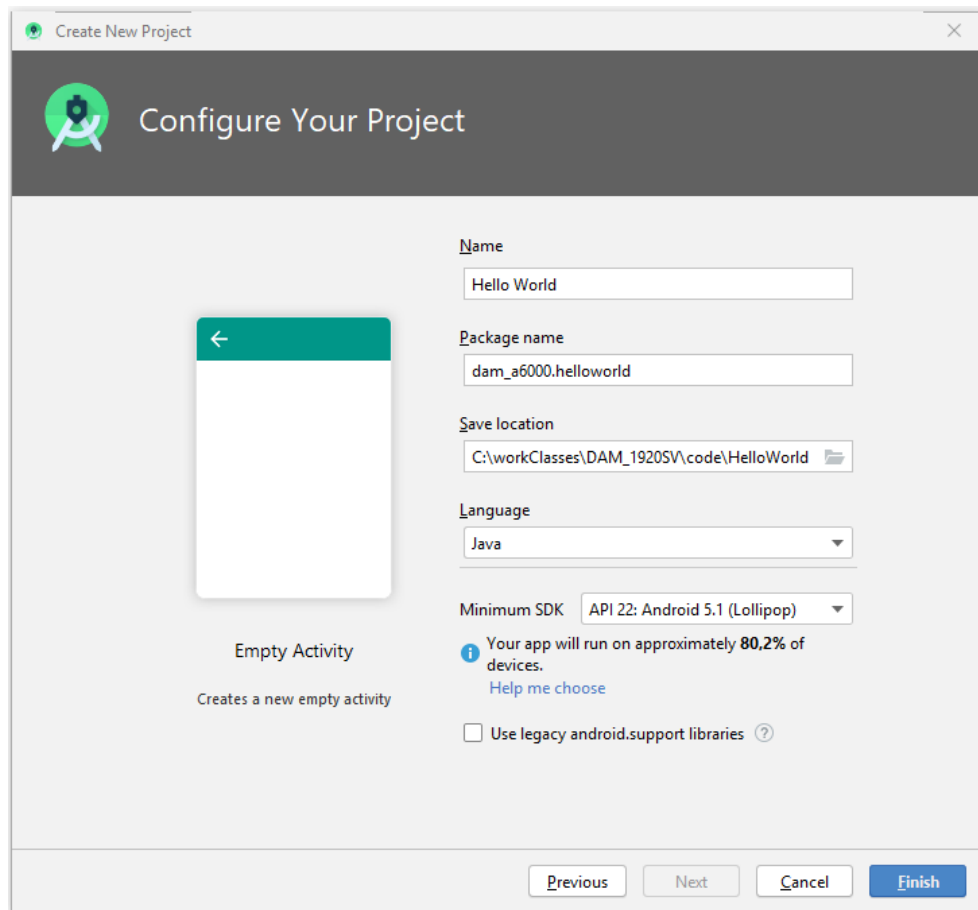


Figure 14: Configure you project.

and the new features introduced by the version. Inspect the new characteristics of each version.

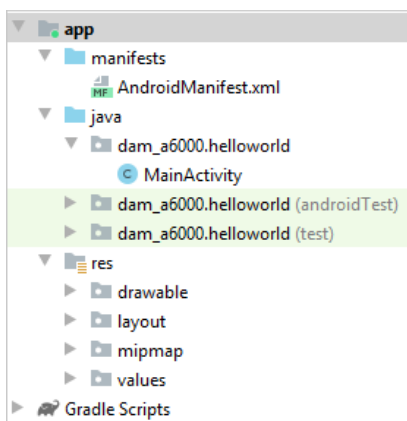


Figure 15: Main files.

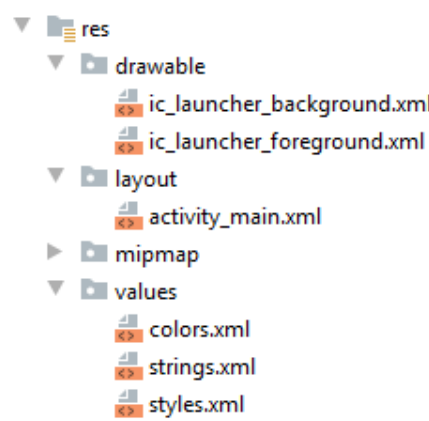


Figure 16: Resource files.

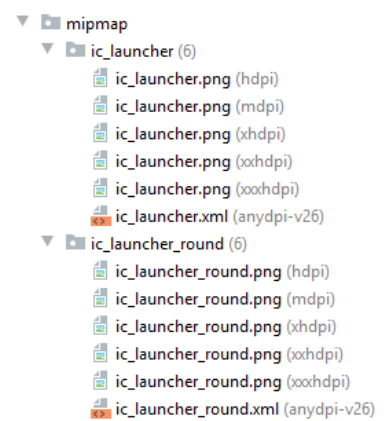


Figure 17: Res. img. files.

3. Inspect generated code. As you can see in Figure 15 the Android Studio creates three sections for files: **manifest**, **java** and **res**.

An app should contain an **manifest** file containing: app package name, components (activities, services, broadcast receivers, and content providers) and their properties, permissions needed and hardware and software features required.

The **java** section contains Java files. It should contain Activity files and all other

Java files. The generated files are set with the defined package. We can see the app contains one Activity.

The **res** (resource) section, as we can see in Figure 16, contains the following resource directories: drawable, layout, mipmap and values. The contents of mipmap are shown in Figure 17.

Inspect the content of all files. View the .xml files in **Code**, **Split** and **Design** mode, by selecting them on the icons on top right side of the editor window.

4. Execute the application in the Pixel 3 AVD. Press **Run** > **Run “app”**. In the window choose your Pixel 3 AVD smartphone and click the “OK” button. The AVD will present the contents of the Figure 18. Check the created elements in the emulator screen.

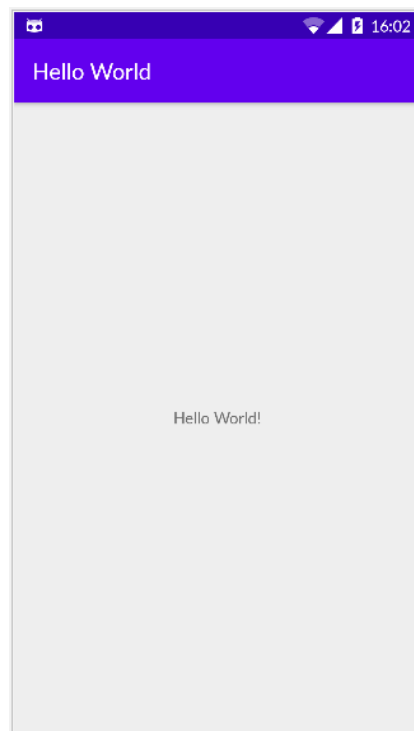


Figure 18: Hello World app.

Note 6: This procedure may take some time. If the execution blocks, try closing the AVD and restarting the execution.

5. Put **strings in strings.xml file**. String should be defined in string.xml file, to enable internationalization (we will see it latter). In strings.xml each string is identified by a name. To get the value of the string we must use `@string/strname` (if named strname). Change the text in the TextView to strings.xml: in activity_main.xml select the existing string “Hello World!” and press ALT + ENTER; select “Extract string resource”; write hello_string in the name and press Ok. Check that the TextView android:text property now has: `@string/hello_string` and in strings.xml there is the new string hello_string declared. Run the app.
6. Change the string hello_string (in strings.xml) to “Hello Android World”. Run again.

7. Change the app name (in strings.xml) to “Hello World V1”. Run again.
8. Run and debug the application in your real device. Use ADB by **USB** cable and by **WiFi**.

See <https://developer.android.com/studio/command-line/adb>.

4.2 Improve your application - Hello World v2

1. Change the app name (in strings.xml) to “Hello World V2”. Run the application (do the same for remaining points).
2. **Change the TextView** to the top and change some properties. You should have the window in Slip mode (select in top right corner), with the **Palette view** at the left, the design editor in the middle and the **Attributes view** at the right. In the design editor elect the TextView and in its bottom constraint give a left mouse click with th CTRL pressed to remove that constraint. To add a constraint of that type just stretch one of that points. Set its attributes: textColor, textSize and textStyle, to look as in 19.
3. **Add a second TextView** with “My First App” text (strings to strings.xml), by dragging one from the palette to the middle of the graphic editor. Connect the top graphical constraint to the bottom of the other TextView. Change the attributes of the TextViews to change: layout_width (set to 0dp to use all width), textSize, textAlignment and background to look as in 19.

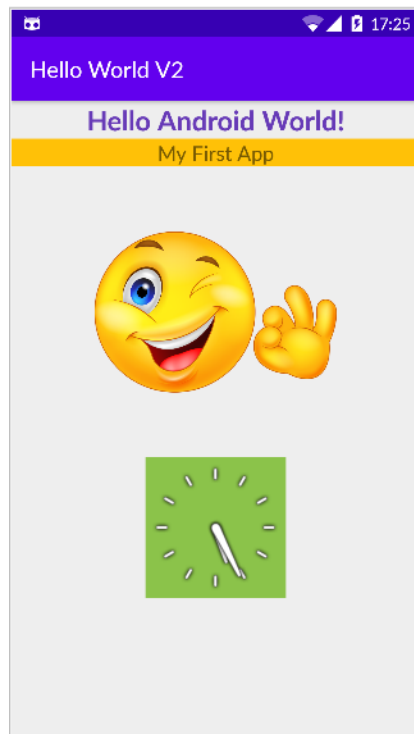


Figure 19: Hello World app v2.

4. **Add an image** to the screen, as shown in Figure 19. Download one small image from the web and add it to the “drawable” folder. The image name should only

have lower-case letters. From the palette view drag an `ImageView` element to the layout of `activity_main.xml`. In “Pick a resource view” select the image. It will add to the `ImageView`: `android:src="@drawable/smile"` (if `smile` is the name of the image file). You will see an yellow warning because the image do not have an description. Add a content Description (alternative text) to the `ImageView`: `android:contentDescription="Just smile"` (strings to `strings.xml`).

5. **Add a watch** to the layout, as shown in Figure 19. The `AnalogClock` is deprecated since API 23 and for that reason it is not available in the components list of the Layouts Editor Interface. However, it can be included in the layout XML file related, using for instance:

```
<AnalogClock
    android:id="@+id/analogClock"
    android:layout_width="124dp"
    android:layout_height="124dp" />
```

The graphical editor gives and error, as we are using the deprecated `AnalogClock`. Nevertheless, we can ignore it and run the app.