DA: Android Introduction

Applications for mobile devices - Lab - Unit 1

Didac Florensa Cazorla

Any: 2021-2022
Curs: 102386
Institut: University of Lleida (Campus Igualada)
Titulació: Bachelor's degree in Digital Interaction and Computing Techniques (GTIDIC)

Agenda

Android Platform Architecture

Android Versions and API Level

Android Studio Tour

Gradle

Homework

Android Platform Architecture

Android Platform

1. System and user apps
2. Java API Framework
   - Android OS is available to you through APIs written in the Java language. (View class hierarchy to create UI screens, Notification Manager and Activity manager for life cycles and navigation).
3. C/C++ Libraries.
   - Give access to core native Android system components and services.
4. Android Runtime.
   - Each app runs in its own process with its own instance of the Android Runtime.
5. Expose device hardware capabilities.
   - Standard interfaces that expose device hardware capabilities as libraries. $\Rightarrow$ since *API level 21*.
6. Linux Kernel
   - Threading and low-level memory management
   - Security features
   - Drivers

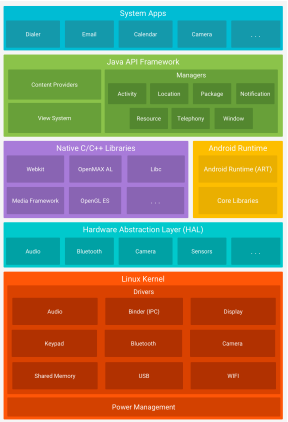


Figure 1: Android Platform from developers

Android Versions and API Level

What is the Android Version?

Android 12 is the latest release. Information from other Android releases

Figure 2: Android version list from temok

What is the problem about Android Versions?

Definition

Fragmentation: It is about different versions of **Android**, the multitude of other devices, the manufacturer skins that sit on top, or even forks.

- **Inconsistency**: Assume an app requires a particular Android version and a specific device feature to function correctly. Multiple Android versions are circulating in the market. Furthermore, there is no guarantee that all Android devices will be upgraded in the market to that version. This restricts the potential users. Many versions also make it challenging to optimize the app for every version.
- **Development**: Every app developer must consider many devices and Android OS versions when creating an app. Imagine the issues in optimizing an app for every possible Android device.
- **Security**: The developer needs to deal with multiple security issues due to the difference between Android versions

⇒ Android fragmentation is one of the most emphatic concerns for developers and testers. They must ensure that it works on the most significant users, which translates to its optimization for various devices and OS versions.

What is the API Level?

Definition

Android platform provides a framework API that applications can use to interact with the Android system.

- A core set of packages and classes.
- A set of XML elements and attributes for declaring a *Manifest file*.
- A set of XML elements and attributes for accessing the resources.
- A set of *Intents*.
- A set of permissions that the app can request.

⇒ To provide the best features and functionality across several Android versions, you should use the Android Support Library in your app, which allows you to use several recent platform APIs on older versions.

How can we define the API Level?

```
<manifest>
    <uses-sdk android:minSdkVersion="5" />
    ...
<\manifest>
```

- **android:minSdkVersion**: Min API level where the app is able to run. By default is 1.
- **android:targetSdkVersion**: API level where the app is designed to run.
- **android:maxSdkVersion**: Max API level where the app is able to run. (*!!!!! It is not recommended to declare this attribute.*)

How to choose the right version

⇒ While the latest versions of Android often provide outstanding APIs for your app, you should continue to support older versions of Android until more devices get updated. This information is alive and needs to be rechecked.

- Versions
- Comparative

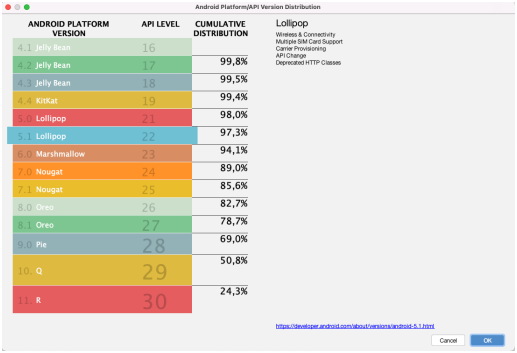


Figure 3: Percentage of Android devices where an application can be installed, according to the minimum version chosen

Question (1)

Imagine that a new Android version appears where no new features have been added (only existing ones have been improved). What do you think is required to change?

- The API level.
- The platform version.
- The commercial name.
- All.

Answer (1)

Imagine that a new Android version appears where no new features have been added (only existing ones have been improved). What do you think is required to change?

- The API level.
- **The platform version**.
- The commercial name.
- All.

Question (2)

True or false:

- Two API levels can correspond to the same version.
- Two commercial names correspond to the same version.
- Two API levels can have the same commercial name.

Answer (2)

True or false:

- Two API levels can correspond to the same version. **False**.*Impossible,if API level increases, the version must increase too.*
- Two commercial names correspond to the same version. **False** *The name is changed when important changes are introduced.*
- Two API levels can have the same commercial name. **True**. *For example, API 16 and 17 belons to Jelly Bean*

Question (3)

We create a new virtual android device (AVD) with min version: Android 1.6, and API level 4. Which apps we can run:

- Only the apps implemented in API 4.
- API 4 and less.
- API 4 and more.

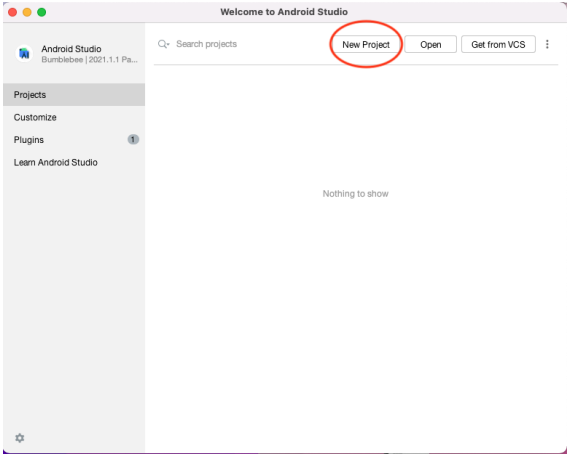
Answer (3)

We create a new virtual android device (AVD) with min version: Android 1.6, and API level 4. Which apps we can run:

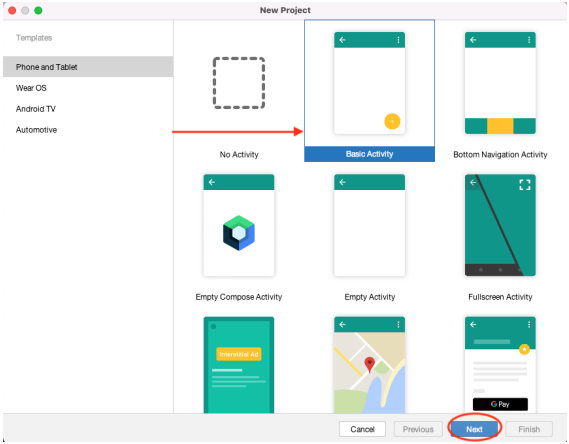
- Only the apps implemented in API 4.
- API 4 and less.
- **API 4 and more**.

Android Studio Tour

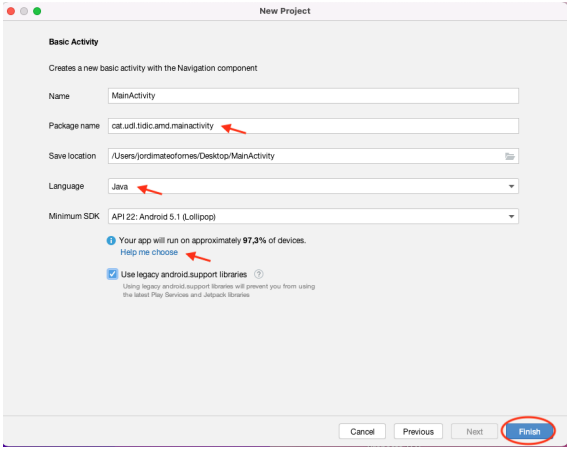
Start Android Studio



Pick a template

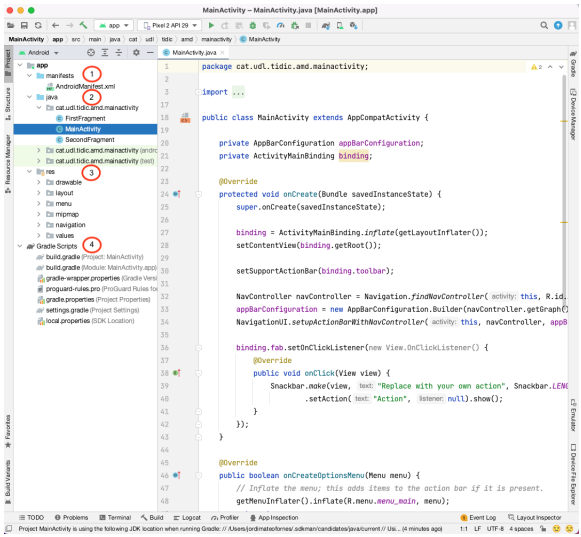


Project settings

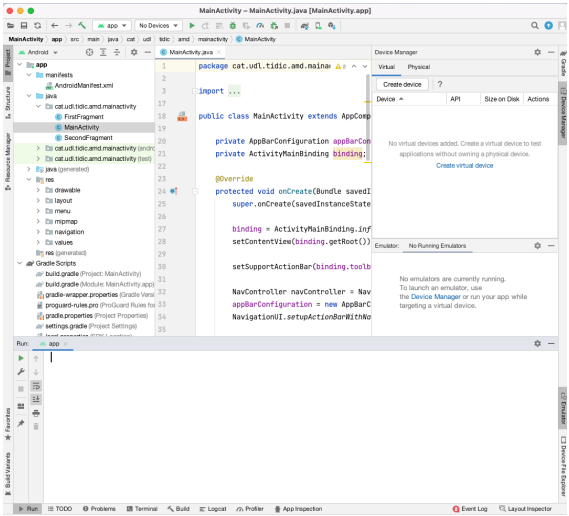


Project folders

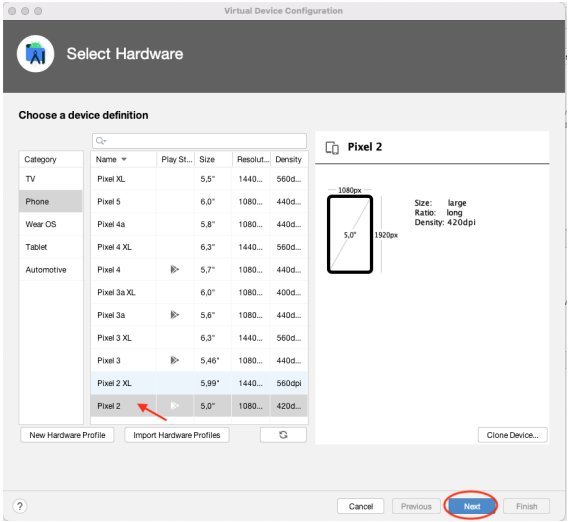
1. **manifests**: Android Manifest file description of app read by the Android runtime.
2. **java** Java source code packages
3. **res**: Resources (XML) - layout, strings, images, dimensions, colors...
4. **build** gradle—Gradle build files



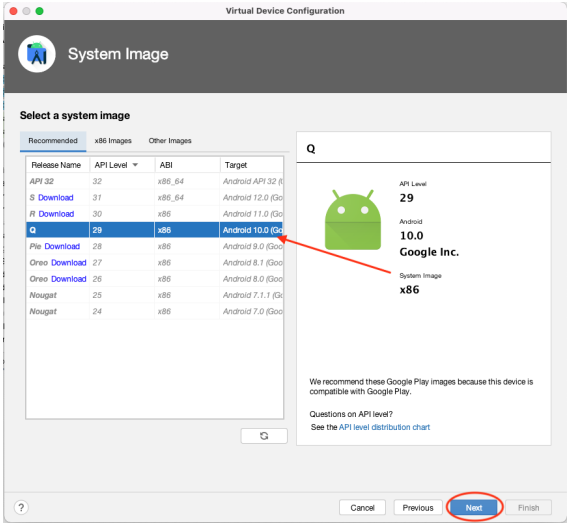
Create a virtual device to run the App



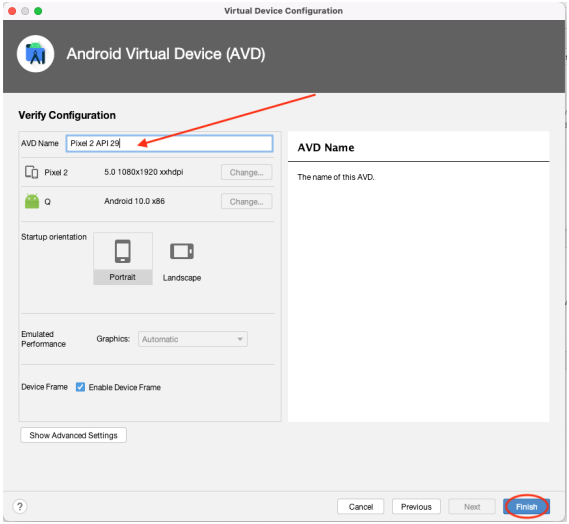
Virtual device configuration



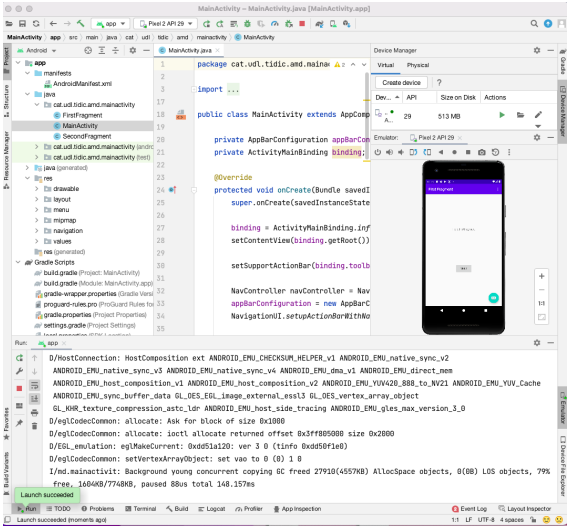
Choose an image



Verify configuration

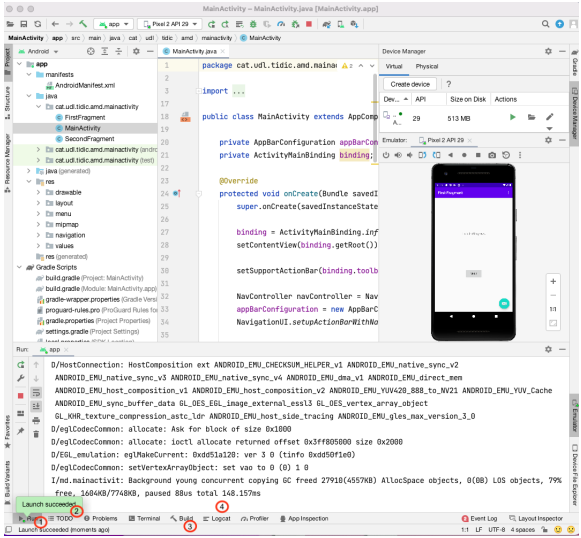


Run on a virtual device



Run feedback

1. **Run**: Emulator running the app. Information of the current execution.
2. **TODO** List of *@ TODO* in code
3. **Build**: Build log.
4. **LogCat** Log information with filters (Log level).

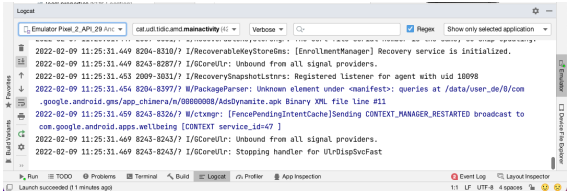


Low to display information in LogCat

```java
import android.util.Log;

// Use class name as tag
private static final String TAG =
    MainActivity.class.getSimpleName();

// Show message in Android Monitor, logcat pane
// Log.<log-level>(TAG, "Message");
Log.d(TAG, "Creating the URI…");
```



Gradle

What is gradle?

Gradle is a build system (open source) used to automate building, testing, deployment... *build.gradle* are scripts where one can automate the tasks. It manages:

- Build
- Dependencies
- Project files Every Android project needs a Gradle for generating an apk from the *.java and* .xml files in the project

⇒ Similar to Ant or Maven

Exemple

For example, the simple mission to copy some files from one directory to another can be performed by the Gradle build script before the actual build process.

How gradle works?

- **BuildScript**: Setup script to add dependecies on the building process. (dependencies).
- **Project**: Element that represents the current project and is the top level item. Contains all important information, plugins, flies and sources directories.
- **Repositories**: Defines which are the sources of the possible dependencies.
- **Tasks**: Pieces of code that are able to perform some action defined. Tasks can be dependent from each other and they can receive actions before and after.
- **Plugins**: Programs that evaluates during the build process and are able to add behaviour and tasks to the current project.

What is the top level build.gradle?

```
plugins {
    id 'com.android.application' version '7.1.1' apply false
    id 'com.android.library' version '7.1.1' apply false
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

What is the module level build.gradle?

```
plugins {
    id 'com.android.application'
}
android {
    compileSdk 31
    defaultConfig {
        applicationId "cat.udl.tidic.amd.mainactivity"
        minSdk 22
        targetSdk 31
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
```

What is the module level build.gradle? (cont.)

```
buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'),
            'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
    buildFeatures {
        viewBinding true
    }
```

What is the module level build.gradle? (cont.)

```
dependencies {
    implementation 'com.android.support:appcompat-v7:28.0.0'
    implementation 'com.android.support:design:28.0.0'
    implementation 'com.android.support.constraint:constraint-layout:2.0.4'
    implementation 'android.arch.navigation:navigation-fragment:1.0.0'
    implementation 'android.arch.navigation:navigation-ui:1.0.0'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'com.android.support.test:runner:1.0.2'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'
}
```

Homework

Task A: Launch the App in an Android Physical Device

1. Turn on Developer Options:
   - Settings > About phone.
   - Tap Build number seven times.
2. Turn on USB Debugging Settings > Developer Options > USB Debugging
3. Connect phone to computer with cable
   - Windows/Linux additional setup: Using Hardware Devices
   - Windows drivers: OEM USB Drivers

Task B: CodeLab 00

Build Your First Android App in Java!!!!

- Do the following codelab 00

That's all

QUESTIONS?

About me

www — jordimateofornes.com
github — github.com/JordiMateo
twitter — @MatForJordi
gdc — Distributed computation group