Android Basics

Table of Contents

1 General Facts 2

1.1 Android Versions 2

2 Platform Architecture 3

2.1 Linux Kernel 4

2.2 Hardware Abstraction Layer 4

2.3 Android Runtime 4

2.3.1 Core Libraries 5

2.4 Native C/C++ Libraries 5

2.5 Java API Framework 5

2.6 System Apps 5

3 Basics of Android Apps 6

3.1 Application Components 6

3.1.1 Activities 6

3.1.2 Services 6

3.1.3 Broadcast receivers 6

3.1.4 Content providers 6

3.2 Android Manifest 7

3.3 Resources 7

3.4 Intents 7

3.5 Fragments 7

1 General Facts

Android is an Operating System for mobile Devices such as Smartphones, Tablets and Smartwatches

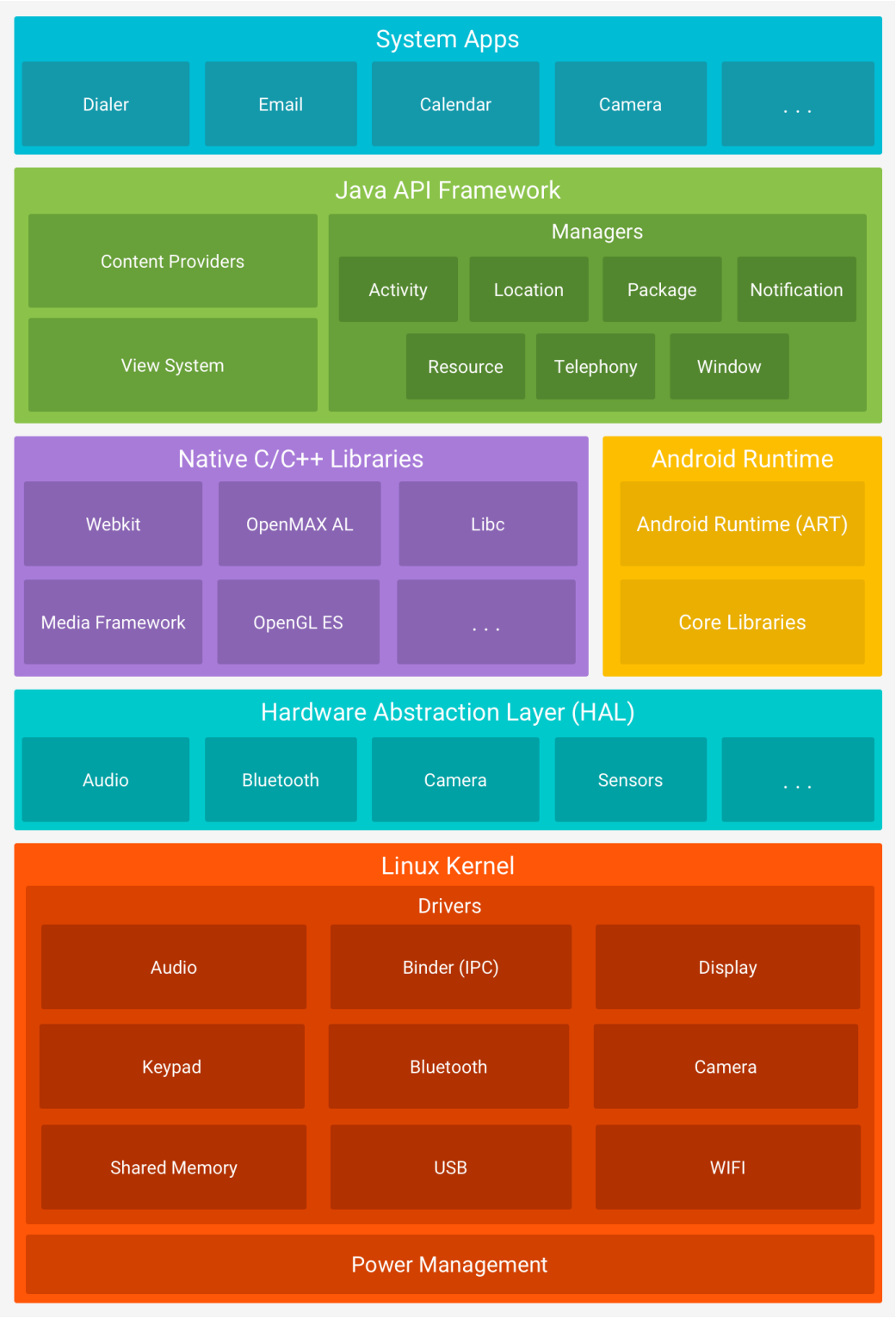
It offers a unified approach for developing applications. An app developed for Android should work on different devices which run Android

1.1 Android Versions

Latest version as of 2017

Oreo
Android 8.0

2 Platform Architecture



2.1 Linux Kernel

The Android Operating System based on the Linux Kernel, which includes the device specific Drivers.

2.2 Hardware Abstraction Layer

The Hardware Abstraction Layer consists of multiple Libraries modules, which provide standard interfaces for specific device hardware. These interfaces are called upon by the Java API Framework, if the Hardware is accessed.

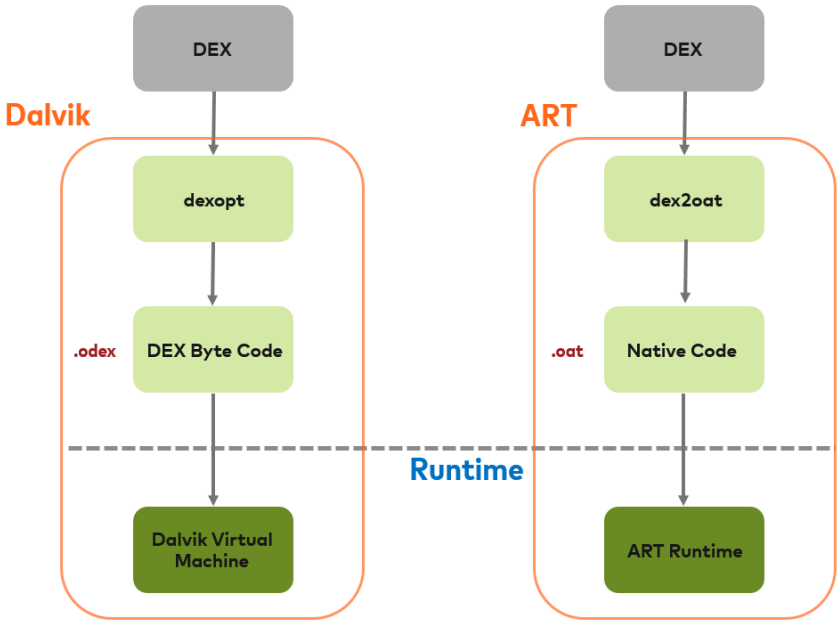
2.3 Android Runtime

Android uses a Virtual Machine as the Runtime to run Apps. The Java Sources of the App is converted into DEX Bytecode. Then the DEX Bytecode is translated into native machine code by the Virtual Machine.

Android has two Virtual Machines. The Dalvik Virtual Machine and the Android Runtime (ART)

The Dalvik Virtual Machine uses a JIT (Just in Time) Compiler which converts the DEX Byte Code into native machine code while running the App. The ART uses an AOT (Ahead of Time) Compiler which compiles the DEX Bytecode into native machine code, while installing the App. This machine code is saved in the device's storage and called upon when the App is running.

The Dalvik Virtual machine was used until Android 5.0 (Lollipop). The ART was introduced in Android 4.4 (KitKat) and replaced the Dalvik Virtual Machine in Android 5.0.



Dalvik vs ART

2.3.1 Core Libraries

The Core Libraries mainly consist of the VM Specific Libraries, Java Interoperability Libraries and Android Libraries

2.3.1.1 VM Specific Libraries

Libraries for interacting directly with an instance of the Virtual Machine. (Unlikely to be used for developing Applications)

2.3.1.2 Java Interoperability Libraries

Adapted and transformed open source implementations of standard Core Java Libraries.

2.3.1.3 Android Libraries

Java based Libraries for Android development.

2.4 Native C/C++ Libraries

These Libraries fulfill a diverse and wide range of function, which are used by systems and components, like the HAL and ART, and the Core Libraries. The Core Libraries are in fact just Java wrappers of the C/C++ Libraries. Android NDK can be used to write Apps in C/C++ by accessing native platform libraries

2.5 Java API Framework

The Java API Framework offers APIs as tools for building Android Apps. They simplify the reuse of core, modular system components and services.

2.6 System Apps

Pre-installed Apps to ensure basic Functionality. There isn't any special difference between these and third-party apps.

3 Basics of Android Apps

Java Apps can be written in Kotlin, Java and C/C++. The contents of the App (code with any data and resource files) are packaged into an “Android package” (.apk). The APK is used to install the app on Android devices.

Activities services and broadcast receivers are entry points of an app.

3.1 Application Components

3.1.1 Activities

An activity represents a single screen with a user interface. It is the part of the app that the user interacts with. An activity can call another activity and working together they fulfill the functions of the App. But they are independent of each other. So, a single Activity of an App can be called by another App.

Activites are implemented by a subclass of the Activity Class.

3.1.2 Services

Services are components which run in the background and perform their intended task. It doesn’t provide a user Interface.

There are two types of Services: Started services and bounded services.

Started services inform the Android system to keep them running until their task is done.

Bounded services are services called by an app or the system which want to use it. The service is practically offering an API and can be used by multiple apps. It keeps running until there isn’t a user of its APIs anymore.

A Service is implemented as a subclass of the Service class.

3.1.3 Broadcast receivers

The Android system and apps can send broadcasts, which are events. There a lot of different types of events. A broadcast receiver is a listener which perceives those and can initiate the suitable action.

A BroadcastReceiver is implemented as a subclass of the BroadcastReceiver class. Broadcasts are sent as an Intent object.

3.1.4 Content providers

A Content provider manages data which can be stored in the file system, a SQLite database, the internet or any other accessible storage. Apps can query and modify data through it.

A ContentProvider is implemented as a subclass of the ContentProvider class.

3.2 Android Manifest

The Android Manifest File is in the root of the app project directory. In this file all components of the app must be declared. In addition, it can declare other things such as the minimum API Level, required user permissions, required or used hardware and software features and other API libraries the app must be linked against.

3.3 Resources

An Android requires other resources besides the source code. Such as images, audio files or any other data for the visual representation of the app. These types of resources are saved in separate sub directories of the global resource directory. They can be differentiated by qualifiers such as language, country, screen size, screen density and many more. These qualifiers are appended on the resources directory name and the Android system checks those qualifiers and picks the corresponding resource. With this ability the App can react to the given circumstances accordingly if programmed right. Such as different App languages based on the system language or different layouts based on the device size.

3.4 Intents

An Intent is an object which is used to request an action from another App component.
There are two types of intents: Explicit intents and implicit intents.

An explicit Intent starts a specific component by its class name.

An implicit intent describes an action that must be performed, which is then performed by a capable app.

3.5 Fragments

A Fragment represents a portion of a user Interface in Activities. They are modular components with their own lifecycle. Fragments can be added and removed while the Activity is running. Their intention is to provide reusable code which can be used in multiple Activities.