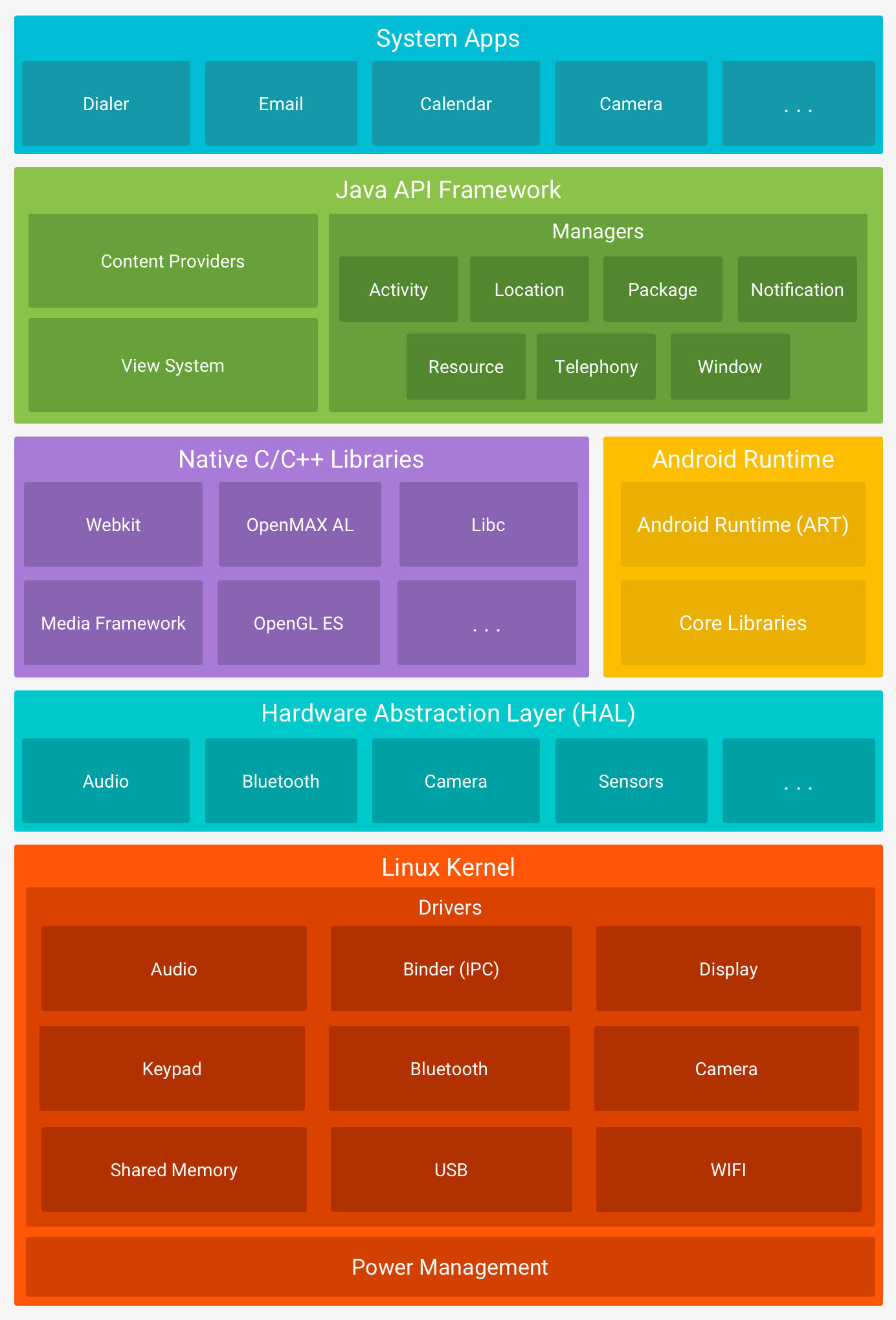
**Android OS Architecture**

The Android software stack generally consists of a Linux kernel and a collection of C/C++ libraries that are exposed through an application framework that provides services, and management of the applications and run time.

Kotlin, Flutter

# Platform Architecture

Android is an open source, Linux-based software stack created for a wide array of devices and form factors. The following diagram shows the major components of the Android platform.



****Figure 1.**** The Android software stack.

## The Linux Kernel

The foundation of the Android platform is the Linux kernel. For example, [the Android Runtime (ART)](https://developer.android.com/guide/platform" \l "art) relies on the Linux kernel for underlying functionalities such as threading and low-level memory management.

Using a Linux kernel allows Android to take advantage of [key security features](https://source.android.com/security/overview/kernel-security.html) and allows device manufacturers to develop hardware drivers for a well-known kernel.

## Hardware Abstraction Layer (HAL)

The [hardware abstraction layer (HAL)](https://source.android.com/devices/architecture/hal) provides standard interfaces that expose device hardware capabilities to the higher-level [Java API framework](https://developer.android.com/guide/platform" \l "api-framework). The HAL consists of multiple library modules, each of which implements an interface for a specific type of hardware component, such as the [camera](https://source.android.com/devices/camera/index.html) or [bluetooth](https://source.android.com/devices/bluetooth.html) module. When a framework API makes a call to access device hardware, the Android system loads the library module for that hardware component.

## Android Runtime

For devices running Android version 5.0 (API level 21) or higher, each app runs in its own process and with its own instance of the [Android Runtime (ART)](https://source.android.com/devices/tech/dalvik/index.html). ART is written to run multiple virtual machines on low-memory devices by executing DEX files, a bytecode format designed specially for Android that's optimized for minimal memory footprint. Build tools, such as [d8](https://developer.android.com/studio/command-line/d8), compile Java sources into DEX bytecode, which can run on the Android platform.

Some of the major features of ART include the following:

* Ahead-of-time (AOT) and just-in-time (JIT) compilation
* Optimized garbage collection (GC)
* On Android 9 (API level 28) and higher, conversion of an app package's [Dalvik Executable format (DEX) files to more compact machine code](https://developer.android.com/about/versions/pie/android-9.0" \l "art-aot-dex).
* Better debugging support, including a dedicated sampling profiler, detailed diagnostic exceptions and crash reporting, and the ability to set watchpoints to monitor specific fields

Prior to Android version 5.0 (API level 21), Dalvik was the Android runtime. If your app runs well on ART, then it should work on Dalvik as well, but [the reverse may not be true](https://developer.android.com/guide/practices/verifying-apps-art).

Android also includes a set of core runtime libraries that provide most of the functionality of the Java programming language, including some [Java 8 language features](https://developer.android.com/guide/platform/j8-jack), that the Java API framework uses.

## Native C/C++ Libraries

Many core Android system components and services, such as ART and HAL, are built from native code that require native libraries written in C and C++. The Android platform provides Java framework APIs to expose the functionality of some of these native libraries to apps. For example, you can access [OpenGL ES](https://developer.android.com/develop/ui/views/graphics/opengl/about-opengl) through the Android framework’s [Java OpenGL API](https://developer.android.com/reference/android/opengl/package-summary) to add support for drawing and manipulating 2D and 3D graphics in your app.

If you are developing an app that requires C or C++ code, you can use the [Android NDK](https://developer.android.com/ndk) to access some of these [native platform libraries](https://developer.android.com/ndk/guides/stable_apis) directly from your native code.

## Java API Framework

The entire feature-set of the Android OS is available to you through APIs written in the Java language. These APIs form the building blocks you need to create Android apps by simplifying the reuse of core, modular system components and services, which include the following:

* A rich and extensible [View System](https://developer.android.com/guide/topics/ui/overview) you can use to build an app’s UI, including lists, grids, text boxes, buttons, and even an embeddable web browser
* A [Resource Manager](https://developer.android.com/guide/topics/resources/overview), providing access to non-code resources such as localized strings, graphics, and layout files
* A [Notification Manager](https://developer.android.com/guide/topics/ui/notifiers/notifications) that enables all apps to display custom alerts in the status bar
* An [Activity Manager](https://developer.android.com/guide/components/activities/intro-activities) that manages the lifecycle of apps and provides a common [navigation back stack](https://developer.android.com/guide/components/tasks-and-back-stack)
* [Content Providers](https://developer.android.com/guide/topics/providers/content-providers) that enable apps to access data from other apps, such as the Contacts app, or to share their own data

Developers have full access to the same [framework APIs](https://developer.android.com/reference/packages) that Android system apps use.

## System Apps

Android comes with a set of core apps for email, SMS messaging, calendars, internet browsing, contacts, and more. Apps included with the platform have no special status among the apps the user chooses to install. So a third-party app can become the user's default web browser, SMS messenger, or even the default keyboard (some exceptions apply, such as the system's Settings app).

The system apps function both as apps for users and to provide key capabilities that developers can access from their own app. For example, if your app would like to deliver an SMS message, you don't need to build that functionality yourself—you can instead invoke whichever SMS app is already installed to deliver a message to the recipient you specify.

## Libraries:

Running on top of the kernel, the Android framework was developed with various features. It consists of various C/C++ core libraries with numerous open-source tools. Some of these are:

**1. The Android Runtime:**The Android runtime consists of core libraries of Java and ART(the Android RunTime). Older versions of Android (4.x and earlier) had Dalvik runtime.

**2. Open GL(graphics library):**This cross-language, cross-platform application program interface (API) is used to produce 2D and 3D computer graphics.

**3. WebKit:**This open-source web browser engine provides all the functionality to display web content and simplify page loading.

**4. Media frameworks:**These libraries allow you to play and record audio and video.

**5. Secure Socket Layer (SSL):**These libraries are there for Internet security.

## Android Runtime

It is the third section of the architecture. It provides one of the key components which is called Dalvik Virtual Machine. It acts like Java Virtual Machine which is designed especially for Android. Android uses its own custom VM designed to ensure that multiple instances run efficiently on a single device.

The Dalvik VM uses the device’s underlying Linux kernel to handle low-level functionality, including security, threading, and memory management.

## Application Framework

The Android team has built on a known set of proven libraries, built in the background, and all of it is exposed through Android interfaces. These interfaces wrap up all the various libraries and make them useful for the Developer. They don’t have to build any of the functionality provided by the android. Some of these interfaces include:

**1. Activity Manager:**It manages the activity lifecycle and the activity stack.

**2. Telephony Manager:**It provides access to telephony services as related subscriber information, such as phone numbers.

**3. View System:**It builds the user interface by handling the views and layouts.

**4. Location manager:**It finds the device’s geographic location.

## Applications:

Android applications can be found at the topmost layer. At the application layer, we write our application to be installed on this layer only. Examples of applications are Games, Messages, Contacts, etc.

developer.android.com