I. ANDROID

I.I. INTRODUCCIÓN AL DESARROLLO ANDROID

CARACTERÍSTICAS Y COMPONENTES

Android es un sistema operativo en continuo crecimiento, cuenta con una comunidad de desarrolladores Linux que continuamente hacen sus contribuciones y permiten un mayor crecimiento y consolidación, buscando de forma permanente innovación y crecimiento, ofreciendo:

- Procesamiento en múltiples núcleos
- Pantallas emergentes
- Gráficos de alto rendimiento
- Sensores

Android proporciona lo necesario para diseñar y construir nuevas experiencias a través de aplicaciones que aprovechan las funcionalidades del dispositivo en el cual serán ejecutadas, pues la interfaz de usuario se adapta de forma automática al dispositivo, siempre en busca de un mejor rendimiento, hecho que proporciona un mayor control sobre la aplicación, dependiendo siempre del dispositivo que las implementa, pues, el desarrollador, tiene la posibilidad de definir los recursos generales de la aplicación y los recursos específicos para dispositivos celulares y tabletas, permitiendo a Android la aplicación de los recursos que considera correctos basándose en factores como: tamaño de pantalla, densidad y configuración regional, entre otros. Este hecho permite afirmar que Android optimiza fácilmente **un único formato para múltiples dispositivos**.

¿En qué dispositivos podemos encontrar el sistema operativo Android?

A través de los siguientes enlaces podrá conocer con mayor detalle los alcances de la plataforma Android para otros dispositivos:

- https://www.android.com/intl/es_es/auto/
- https://www.android.com/intl/es_es/tv/

El constante cambio en las necesidades del usuario ha ocasionado una **evolución en sus versiones**, todas basadas en la experiencias y comentarios del usuario final. Desde que Android fue lanzado comercialmente en el año 2008 ha mantenido una relación directa con los terminales a los cuales va dirigido, es decir, su ambiente y estructura están basados en el alcance de los dispositivos.

A través del siguiente video podrá conocer un poco la evolución en las versiones del sistema operativo Android: https://www.youtube.com/watch?v=Yi9q5Q0KX84. Es importante tener en cuenta que el versionamiento en Android maneja: versión comercial (ejemplo: Android Pie), versión del fabricante (Ejemplo: Android 1.6) y versión para desarrollo (se le conoce como el nivel de API, por ejemplo, API 31).

Al momento de iniciar con el desarrollo de aplicaciones, la **compatibilidad** juega un papel importante, debido a que es posible que algunas características que se requiere implementar no apliquen a la totalidad de dispositivos, por ejemplo, la aplicación de lectura de huellas dactilares.

Por lo anterior, **es necesario conocer las versiones del sistema operativo de acuerdo con la funcionalidad de la aplicación**.

CARACTERÍSTICAS DE ANDROID

De acuerdo con la forma en la que son desarrolladas, encontramos dos tipos de aplicaciones que no están destinadas a una sola plataforma, entre ellas, se encuentran las aplicaciones web y las aplicaciones híbridas. En las aplicaciones web, encontramos tecnologías como HTML, Javascript y CSS, permitiendo, por ejemplo, acceder a algunas páginas para encontrar funcionalidades similares a las aplicaciones instaladas sobre el sistema operativo del dispositivo. Para el caso de las aplicaciones híbridas, encontramos aquellas que son desarrolladas en framework específicos, buscando integrar en un mismo código la funcionalidad que será exportada a otras plataformas, mezclando recursos nativos con estructura web, por ejemplo, muchos de los juegos actualmente disponibles, corresponden a aplicaciones híbridas, además, de aplicaciones multimedia.

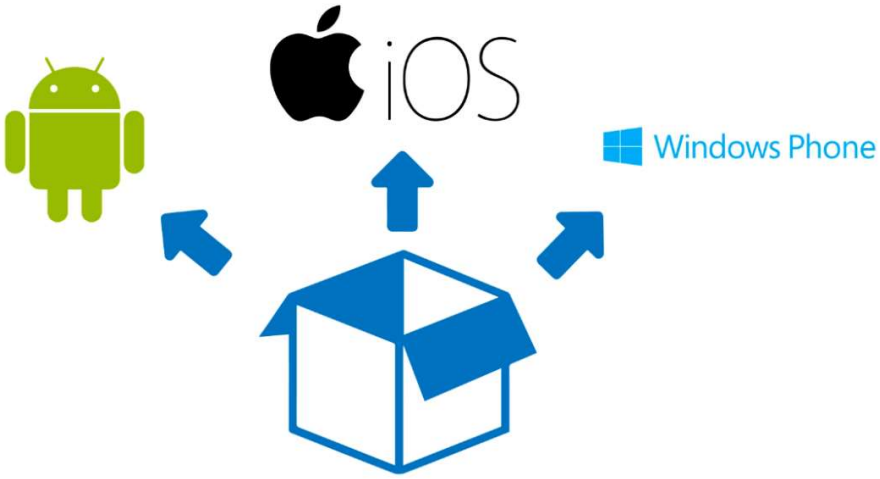

Imagen 1. Aplicaciones híbridas

A las aplicaciones que son desarrolladas bajo una sola plataforma, se les conoce con el nombre de **aplicaciones nativas**, impidiendo que sea exportada o distribuida en tiendas de otras plataformas, por ejemplo, al desarrollar una aplicación Android, ésta será distribuida únicamente en Google Play. Dentro de las ventajas que brindan las aplicaciones nativas, encontramos el acceso completo a los recursos del dispositivo, haciendo más sencilla la comunicación entre los componentes de software y hardware, por ejemplo, al querer utilizar el esquema de notificaciones de Android, la mejor opción es construir una aplicación nativas.

ARQUITECTURA ANDROID

Conocer la estructura del sistema operativo es fundamental para el desarrollo de aplicaciones pues evita programar a bajo nivel y facilita el acceso de manera abstracta a los recursos hardware del dispositivo. Su estructura interna se puede organizar en capas, que separan al usuario final, los recursos internos y a los desarrolladores.

La siguiente imagen proporciona información sobre la distribución por capas.

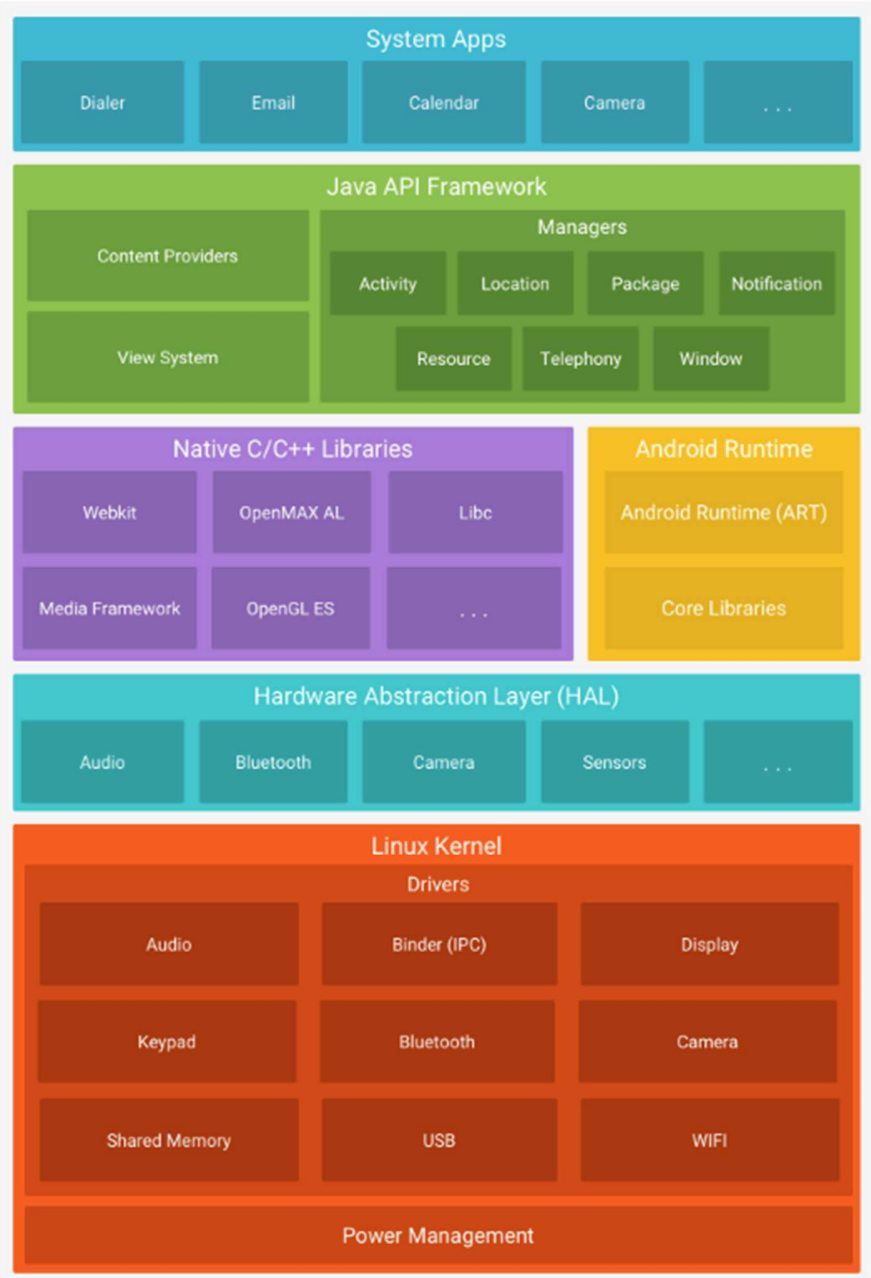

Imagen 2. Distribución por capas[1]

Como se observa, las capas que integran la arquitectura mencionada en forma descendente son: aplicaciones, API Framework, librerías, entorno de ejecución, HAL y kernel de Linux. Cada capa utiliza recursos de la capa siguiente, generando una completa dependencia entre los niveles y conformando una arquitectura de tipo pila.

Haga lectura de la siguiente descripción sobre la arquitectura Android que le proporcionará una visión más específica: https://developer.android.com/guide/platform?hl=es-419#art.

DESCRIPCIÓN DE LA ARQUITECTURA ANDROID

KERNEL DE LINUX: Es la primera dentro del esquema que compone el sistema operativo. El sistema Android centra todo su potencial en el núcleo, el cual es optimizado para dispositivos móviles. El kernel es el corazón del sistema operativo, sirve de conexión entre el hardware y el software, es decir, recibe las órdenes del software para enviarlas al hardware, lo cual lo constituye en el alojador de los drivers del dispositivo, por lo que cada elemento hardware cuenta con su controlador en el kernel.

CAPA DE ABSTRACCIÓN DE HARDWARE HAL: Esta capa se constituye en un elemento estándar para permitir a los proveedores de hardware, implementar sus módulos sin afectar el kernel de Android, haciéndolo independiente. En esta capa, los proveedores encuentran una interfaz genérica que permite hacer sus implementaciones de manera independiente al sistema operativo. Cuando se requiere el uso de hardware a través de la invocación de un componente del marco de trabajo API Java, es la HAL quien proporciona el acceso.

LIBRERÍAS DE ANDROID: está compuesta por las bibliotecas nativas de Android, compiladas de acuerdo a la arquitectura del dispositivo, están elaboradas por el fabricante y generalmente hechas en C o C++. Corresponden a funciones que habitualmente se hacen de forma repetitiva, evitando implementar las mismas funciones varias veces. Dentro de su estructura, se encuentran librerías que permiten el funcionamiento del ART y la HAL. Entre las librerías que generalmente se encuentran en todas las versiones de Android están:

- OpenGL: soporta animaciones y gráficos
- Multimedia: soporta formatos de audio, video e imágenes
- WebKit: soporta navegadores
- FreeType: soporta las fuentes de texto
- SQLite: soporta conexiones a bases de datos

ANDROID RUNTIME: entorno de ejecución, utiliza librerías para su funcionamiento y con esta capa se quiso simular el ambiente de máquina virtual de Java, pero en su implementación se encontraron con que los dispositivos no estaban dotados de buena memoria y procesador para soportarla, por lo que fue necesario disponer de una nueva máquina virtual llamada ***Dalvik*** (hasta Android 5)**,** la cual sería más eficiente. De esta forma, las aplicaciones son desarrolladas en Java/Kotlin, pero al

[1] Developers Android. Recuperado el 29 de agosto de 2024. https://developer.android.com/guide/platform?hl=es-419#art

momento de ser compiladas pasan a un formato de capas interpretado por Dalvik, lo cual se convierte en una ventaja porque garantiza su ejecución en cualquier dispositivo que cuente con la versión mínima especificada. Debe tener en cuenta que la compilación no acepta el bytecode de Java, lo que significa que en lugar de generar un archivo .java, genera un archivo .dex que no puede ejecutarse en Java.

Desde la versión 5, fue implementado el ART (Android Runtime), e cual ejecuta archivo en formato Dalvik (DEX) generando una instancia por cada aplicación.

Java API Framework: aloja diferentes clases y servicios utilizados por las aplicaciones al momento de realizar sus funciones a manera de API escritas en lenguaje Java, a partir de este nivel, se acceden algunos de los recursos de capas inferiores. Dentro de los componentes más importantes de esta capa se encuentran los siguientes:

- Activity manager: encargado de gestionar la pila y ciclo de vida de las actividades de las aplicaciones.
- Windows manager: su función radica en adecuar el espacio en donde se mostrarán las actividades.
- Content provider: librería que contiene el conjunto de información a distribuir entre las diferentes actividades, es muy importante porque controla el acceso a dicha información.
- Views: contiene los recursos para la construcción de las interfaces.
- Notification Manager: se encarga de administrar las notificaciones, esta librería accede a otros recursos, por ejemplo: sonidos, vibraciones y alertas led.
- Package manager: gestiona los paquetes instalados en el dispositivo y administra la instalación de nuevos, un paquete hace referencia a los recursos comprimidos en el .apk, el cual, a su vez, incluye los .dex.
- Telephony manager: permite hacer llamadas, enviar y recibir mensajes de texto.
- Location manager: permite acceder a la posición geográfica del dispositivo, brinda la posibilidad de utilizar recursos de mapas
- Resource manager: esta librería administra recursos adicionales al código fuente, como imágenes, sonidos o cadenas de texto en otros idiomas.
- Sensor manager: permite utilizar los recursos de hardware asociados a sensores como: brújula, giroscopio, sensor de presión, etc.

Cada API que proporciona esta capa está disponible para el sistema operativo y para los desarrolladores.

CAPA DE APLICACIONES: hace referencia a las diferentes aplicaciones del dispositivo, ya sean las que contienen o no interfaz gráfica, las que son nativas, las que son administradas, las preinstaladas y las que el usuario descarga e instala. Sobre esta capa se encuentra el Home o Launcher, el cual se encarga de presentar las aplicaciones al usuario final permitiendo su ejecución. A través de esta capa es posible genera los accesos directos y utilizar los widgets.

Además, esta capa permite el uso de funcionalidades de otras aplicaciones, con algunas excepciones, de esta manera, si se requiere enviar un SMS, no es necesario construir la funcionalidad, se puede hacer uso de otra App que lo realice.

COMPONENTES DE LAS APLICACIONES ANDROID

Un componente es un elemento fundamental para la correcta ejecución de una aplicación Android. Cada componente es único y en combinación con los demás, permiten definir el comportamiento general de una aplicación.

Las aplicaciones Android tienen 4 componentes cada uno con funcionalidades puntuales para el desarrollo de procesos en el dispositivo, esto son:

- **Actividades**: Es el que más percibe el usuario cuando abre una aplicación en dispositivo, una actividad es un espacio de pantalla que le permite al usuario interactuar con una tarea en específico. Por ejemplo: realizar una llamada. Generalmente la actividad cubre toda la pantalla, aunque en algunas cosas, ocupa porciones de la pantalla o la encontramos de forma flotante.

  En ocasiones encontramos un conjunto de actividades que interactúan en sí para lograr una completa experiencia al usuario, dentro de ese grupo de actividades, hay una que se denomina actividad principal, la cual se mostrará inicialmente al usuario cuando abra la aplicación.

  Cuando el usuario está navegando en una actividad y abre una nueva, la actividad anterior se detiene, pero no se pierde, el sistema la conserva en una pila (la cual se rige por el orden LIFO). Las actividades pueden manejar CALLBACKS, como métodos de recepción, lo cual permite distribuir las diferentes acciones que una actividad puede tomar (crear, pausar, reanudar o destruir). Por ejemplo: una aplicación que recibe información de una base de datos, en el momento en el que el usuario decida pausar una de las actividades, dicha actividad debe redefinir su estado, evitando hacer procesos largos o muy pesados, mientras este estado de pausa se mantenga. Una vez se regrese a la actividad, estos procesos son redefinidos para activar nuevamente proceso de conexión a bases de datos.

- **Servicios:** son un componente que realiza procesos de ejecución sin necesidad de proporcionar una interfaz gráfica, generalmente, se implementan en segundo plano, es decir, cuando la aplicación está enfocada en otros procesos, por esto, cuando un usuario cambia de aplicación, los servicios de la aplicación anterior seguirán ejecutándose.

  Los servicios pueden presentarse de dos formas diferentes: iniciado y vinculado.

El servicio es *iniciado* cuando desde otro componente se inicia de manera directa su ejecución, incluso, si el componente que lo inició se destruye, el servicio continuará su funcionamiento. Por ejemplo: si un servicio hace referencia a descargar un archivo, este se detiene cuando finaliza la descarga y no es necesario que esté activo el elemento que lo creó.

El servicio es *vinculado* cuando un componente de una aplicación se une con un determinado servicio, este servicio actúa como un modelo: cliente – servidor, donde los componentes pueden hacer peticiones y recibir información. El tiempo de vida de un servicio vinculado es determinado por el tiempo en que un componente esté consumiendo este servicio, de esta forma, cuando deje de consumirse, éste será destruido, incluso, varios componentes pueden unirse a un mismo servicio y cuando estos finalicen, el servicio también será destruido.

Los servicios son implementados **en el mismo hilo de ejecución** de la aplicación, por lo que es importante tener en cuenta que, si el servicio es demasiado pesado, debe generarse un hilo independiente para no generar bloqueos o largos tiempos de respuesta.

- **Proveedores de contenido:** es el encargado de gestionar la información que puede ser compartida entre aplicaciones, la cual puede encontrarse en los archivos del sistema, en un base de datos o en internet, en todo caso, *el acceso a la información está determinado por los permisos del proveedor de contenido.*

Por ejemplo: seguramente hemos utilizado aplicaciones que tienen acceso a información de contactos del teléfono, como facebook, Whatsapp o Snapchat, lo cual indica que una aplicación con los permisos de acceso puede acceder y modificarla si es necesario.

Pero también existen aplicaciones cuya información no es compartida con otras. Por ejemplo: la aplicación de notas de Android.

- **Receptores de difusión:** este componente hace referencia a los mensajes de difusión que son enviados a todo el sistema. Se caracterizan por no presentar interfaz gráfica al usuario, además, no generan mucho gasto en el consumo de recursos al dispositivo debido a que su

uso está enfocado en convertirse en una puerta de entrada hacia otro componente al que se requiera conducir a usuario.

Para este tipo de componente hay un recurso comúnmente utilizado para enviar mensaje a todo el sistema: *la barra de estado*, mediante este recurso, la información necesaria al usuario es notificada como una alerta indicando que existe un elemento que requiere su inmediata atención, redirigiéndolo si es necesario, a otro componente.

Por ejemplo: cuando la batería del dispositivo se está agotando o existe una nueva versión de alguna aplicación instalada se generan unos mensajes que son gestionados por el receptor de difusión.