

Estudio de las arquitecturas de tecnologías móviles

Introducción

Unidad 2

Contenidos

- 1 **Infraestructura de movilidad**
 - Red de comunicaciones
 - Arquitectura de aplicación
- 2 **Descripción detallada de la arquitectura interna de los sistemas**
 - Android
 - iOS
 - Windows y BlackBerry
- 3 **Análisis de funcionalidades y controles de los sistemas**
 - Sandboxing*
 - Sistema de permisos
 - Control de acceso
 - Arranque seguro
 - Cifrado de datos
- 4 **Ejercicios de investigación**
- 5 **Ejercicios prácticos**
 - Test de evaluación

A long-exposure photograph of a multi-lane highway at night. The image shows vibrant light trails from vehicles, with red and orange streaks in the foreground lanes and blue and white streaks in the background lanes. The road curves into the distance under a dark sky. A semi-transparent blue rectangular box is overlaid on the middle of the image, containing the text 'Infraestructura de movilidad' in white.

Infraestructura de movilidad

Red de comunicaciones



Introducción

- Una red de comunicaciones es un sistema distribuido para el envío de información entre localizaciones.
- De forma genérica una red de comunicaciones está compuesta por:
 - **Terminales** (*Endpoints*): el lugar de la red en el que se inicia o finaliza una transmisión de datos.
 - Ejemplos: estaciones de trabajo, teléfonos móviles, servidores, etc.
 - **Nodos**: puntos de la red por donde viajan los datos, pero que no son ni origen ni destino.
 - Ejemplos: *routers*, *switches*, puntos de acceso, etc.
 - **Canales**: elementos que conectan los diferentes nodos y puntos finales de la red.
 - Ejemplos: fibra óptica, Bluetooth, wifi, cable Ethernet, etc.

◆◆◆ Redes inalámbricas

Modos

- Los dispositivos móviles acceden principalmente a la red a través de redes inalámbricas.
- Las redes inalámbricas son aquellas en las que algunos de sus elementos, generalmente los terminales, se conectan sin necesidad de cables, utilizando radiofrecuencia.
- Existen tres tipos principales de redes inalámbricas:
 - **Modo infraestructura:** es una red de tipo centralizado, donde los terminales se conectan mediante un punto de acceso para poder comunicarse entre sí o con otros dispositivos.
 - Ejemplos: router wifi en redes WLAN, estación base en redes de telefonía móvil, etc.
 - **Modo ad-hoc:** no requiere de un punto de acceso centralizado. En su lugar, los dispositivos se conectan entre sí.
 - Ejemplos: Wifi Direct, Bluetooth, NFC, Infrarrojos, etc.
 - **Modo heterogéneo:** combinan características de ambos tipos de red.

Modo infraestructura

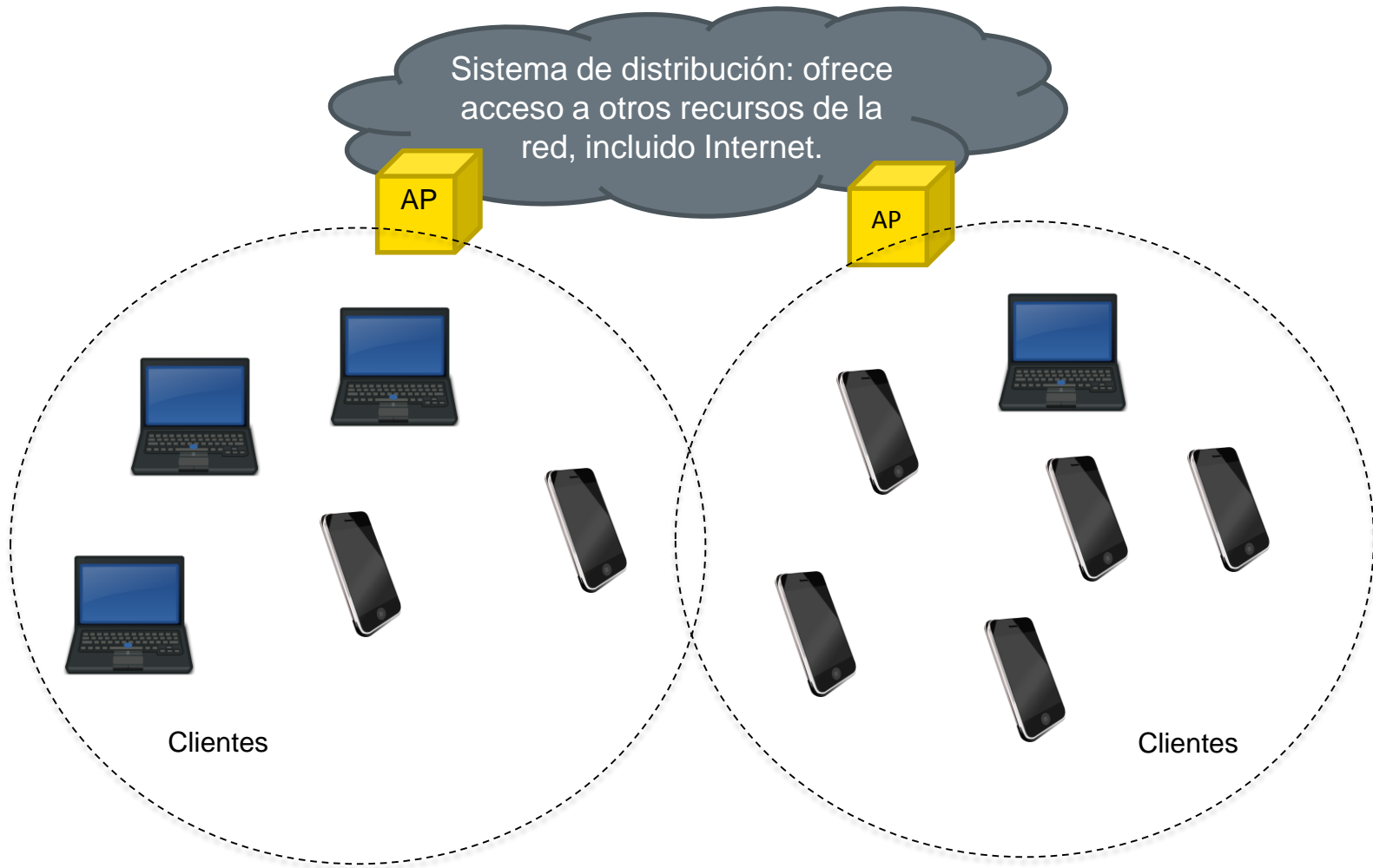
- Dependiendo del tipo de infraestructura, generalmente existen dos tipos de red:
 - **Redes WLAN:**
 - Cobertura limitada a espacios geográficos relativamente pequeños: domicilio, escuela, edificio, etc.
 - En general, están basadas en el estándar IEEE 802.11 (el nombre comercial es el símbolo wifi).
 - Opera en las bandas de radiofrecuencia de 2.4 y 5 GHz.
 - **Redes de telefonía móvil:**
 - Ofrecen conectividad inalámbrica a espacio geográficos de gran tamaño.
 - La red es dividida en celdas. En cada celda la estación base y antenas ofrecen acceso a los dispositivos inalámbricos que se conectan.
 - Múltiples tecnologías: GSM, GPRS, CDMA, UMTS, etc.
 - La banda de frecuencia depende de la tecnología: 0.9, 1.8, 2.1 GHz y más.

Elementos

- **Clientes (estaciones):** terminales que se conectan al punto de acceso a través de su interfaz inalámbrico.
- **Punto de acceso (AP):** ofrece una conexión inalámbrica hacia la red. Hace de puente entre la red inalámbrica y el resto de la infraestructura.
 - Puede conectar a redes cableadas o a redes telefónicas celulares (módems 4G con wifi).
- **Sistema de distribución:** ofrece acceso al resto de servicios.
 - En algunas conexiones puede ser la red de la organización o la red doméstica.
 - En otras es la red telefónica.
 - En ambos casos se ofrece acceso a internet a través de *routers* y *switches*.



Esquema



Protocolos de seguridad en 802.11

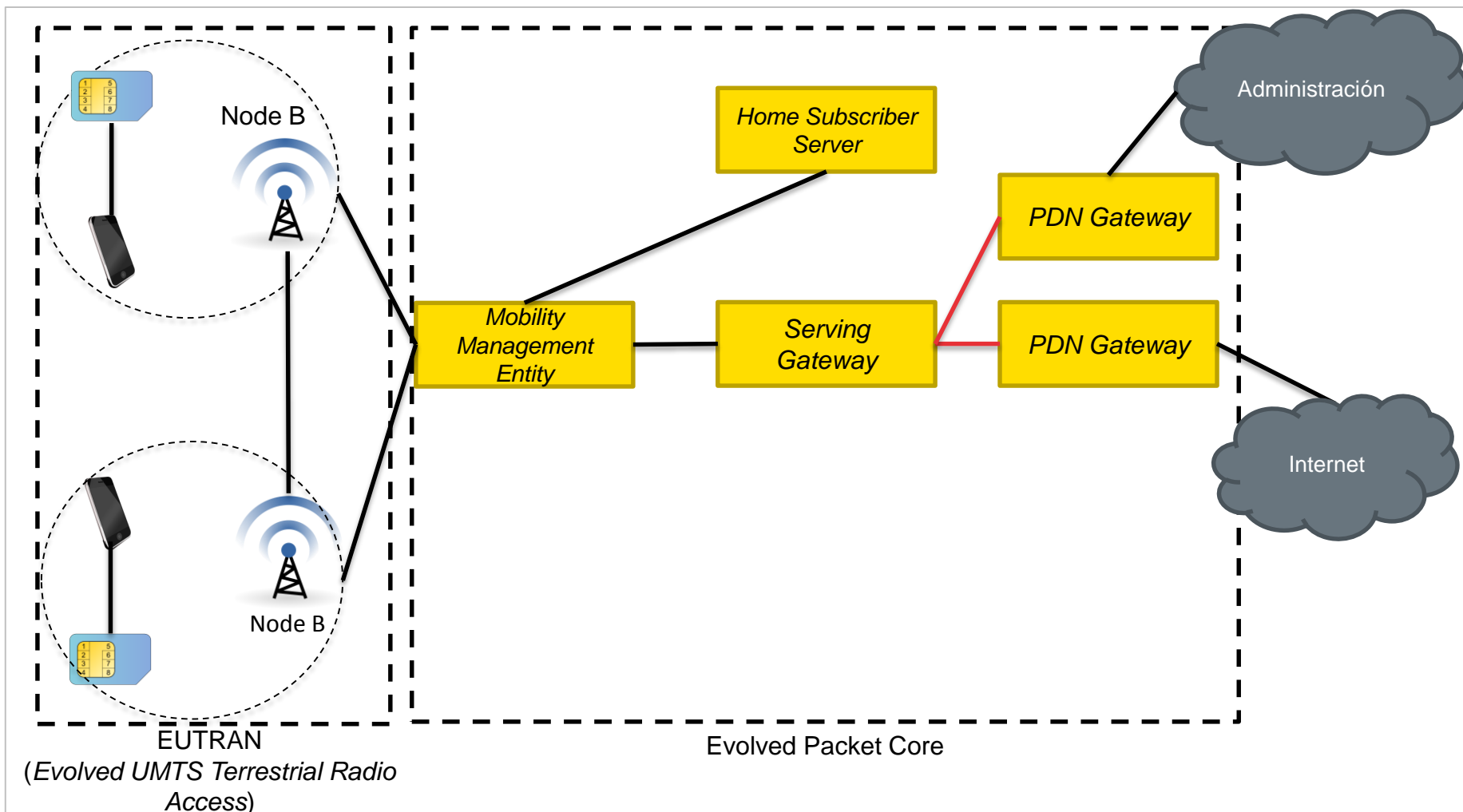
- **WEP (*Wired Equivalent Privacy*):**
 - Prácticamente en desuso.
 - No hay autenticación, solo control de acceso al medio mediante cifrado.
 - Considerado inseguro, su descifrado es trivial utilizando un equipo doméstico.
 - Prohibido su uso en cualquier red que trate información de tarjetas de crédito según PCI-DSS (Estándar de compañías de tarjetas de crédito).
- **WPA y WPA2 (*Wifi Protected Access*):**
 - WPA Mejora el cifrado y control de integridad de WEP con el mismo *hardware* disponible. TKIP (*Temporal Key Integrity Protocol*) para generar una clave de cifrado por paquete y MICHAEL como función de control de integridad.
 - WPA2 añade algoritmos de cifrado y control de integridad basados en AES.
 - Permiten dos tipos de autenticación:
 - Clave pre-compartida: equipos domésticos. Siguen siendo vulnerables a ataques de diccionario para el descifrado de la clave de acceso a la red.
 - Empresa: permite implementar diferentes mecanismos de autenticación basándose en EAP. Utiliza un servidor RADIUS para la autenticación. Ofrece autenticación mutua.

Elementos

- Las redes de telefonía móvil, como las LTE, cuentan con los siguientes elementos:
 - *User Equipment*: el dispositivo móvil y la tarjeta SIM del usuario.
 - *Node B*: ofrece el enlace inalámbrico entre los dispositivos móviles y la red.
 - *Mobility Management Entity* (MME): elemento principal de la red de telefonía.
 - Controla las funciones principales de la red:
 - Seleccionar el *gateway* por el que irán los paquetes IP.
 - Autenticación del usuario.
 - Punto hasta el que se cifra la información desde el dispositivo móvil.
 - *Home Subscriber Server* (HSS): guarda la información de los usuarios de la red para la autenticación, su localización y dirección IP.
 - *PDN Gateway*: ofrece acceso a la red central a servicios como internet o a redes internas de administración (la voz es transmitida en tramas IP).
 - *Serving Gateway*: se encarga del enrutador de los paquetes y al *PDN Gateway*.

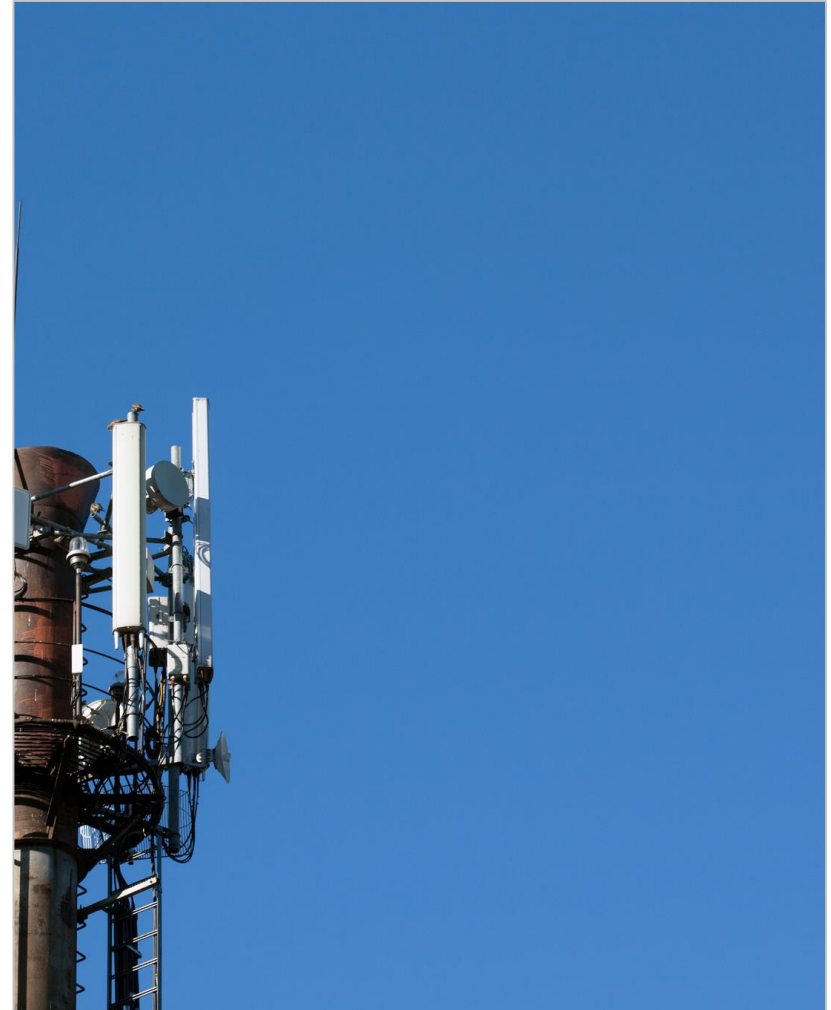
◆◆◆ Redes de telefonía móvil

Esquema de la red LTE (simplificado)



Protocolos de seguridad

- Dependiendo de la tecnología de la red, se utilizan algoritmos y protocolos diferentes, pero en general, la seguridad recae en la clave de autenticación (Ki) que se almacena en la SIM y en el HLR.
 - Es única para cada tarjeta SIM y no se puede acceder a ella directamente (salvo manipulación maliciosa).
 - La Ki no puede abandonar la SIM. Si es necesario firmar algo con la Ki será la SIM la que realice el proceso.
- Respecto a las generaciones de tecnología:
 - En GPRS, cifrado A5, vulnerable, todos los datos pueden ser descifrados.
 - En 3G, cifrado KASUMI, vulnerable, pero con ataques poco prácticos (salvo los llevados a cabo a través de femtocells).
 - En LTE, diferentes algoritmos para mitigar vulnerabilidades (AES entre ellos).



Protocolos de seguridad adicionales

- Los protocolos de seguridad en redes inalámbricas permiten asegurar la confidencialidad e integridad de los datos transmitidos entre el dispositivo móvil y el punto de acceso a la red.
- Para ofrecer seguridad más allá **del punto** de acceso a la red, existen principalmente dos soluciones:
 - **SSL** (nivel de aplicación):
 - Confidencialidad e integridad entre el dispositivo y el servidor (aplicación).
 - Basado en certificados.
 - Utilizado en conexiones HTTPS.
 - Usado habitualmente para asegurar el tráfico de las aplicaciones móviles.
 - **VPN** (nivel IP):
 - Envía todo el tráfico IP a través de un túnel cifrado.
 - Diferentes opciones, siendo la más establecida IPsec.
 - Utilizado generalmente para asegurar el tráfico de un dispositivo para acceder a la red de una organización.

Modo ad-hoc

- Permite la creación de nuevos tipos de red como las *Personal Area Networks* (PAN) en las que interaccionan dispositivos.
- En los dispositivos móviles, los modos ad-hoc utilizan principalmente dos interfaces: wifi y Bluetooth.
- **Wifi:** principalmente a través del estándar Wifi *Direct*.
 - El establecimiento de la clave de sesión se realiza mediante Wifi *Protected Setup* (WPS) y la conexión se cifra mediante WPA.
 - Las principales aplicaciones incluyen la creación de puntos de acceso desde dispositivos móviles, impresión a través de wifi, y acceso a otros servicios como controles remotos, etc. en dispositivos embebidos.
 - La información en las redes de este tipo va cifrada, pero no se realiza ningún tipo de autenticación ni comprobación de credenciales.
 - Esto implica que los dispositivos no pueden controlar quien se conecta a ellos.
 - Muchos de estos dispositivos traen la funcionalidad activada por defecto, lo que genera nuevos vectores de ataque que probablemente no hayan sido considerados por la organización.

Modo ad-hoc

- Bluetooth: permite conexiones ad-hoc con un bajo consumo de energía.
 - Durante los últimos años se ha popularizado debido a la proliferación de dispositivos inteligentes, conocido como el *Internet of Things* (IoT).
 - Se utiliza como método de conexión hacia la mayoría de accesorios electrónicos (relojes, electrodomésticos, coches, etc.).
 - Las conexiones Bluetooth se pueden realizar en dos modos:
 - **Con emparejamiento:** se ejecuta un protocolo de establecimiento de conexión entre los dos dispositivos. La clave de cifrado de la comunicación se intercambia por un canal fuera banda (generalmente el usuario la lee de la pantalla de un dispositivo y la introduce en el otro).
 - **Sin emparejamiento:** utilizado para *beacons* Bluetooth. Un *beacon* es un pequeño dispositivo que emite un identificador único que puede ser leído por otros dispositivos Bluetooth sin necesidad de emparejamiento. Se utiliza principalmente para servicios basados en localización para interiores.
 - Mediante la utilización de dispositivos especiales, se pueden capturar los paquetes Bluetooth, aunque para el descifrado es indispensable haber capturado los paquetes intercambiados durante el proceso de emparejamiento.
 - En muchas ocasiones, permiten el rastreo del usuario pues no modifican la dirección MAC del dispositivo.

Modo ad-hoc

- NFC (*Near Field Communication*): es un tipo de comunicación inalámbrica que permite el intercambio de información a muy corta distancia (pocos centímetros).
 - El principal beneficio de las conexiones NFC es que pueden participar elementos sin una fuente de energía. En esos casos, la misma onda electromagnética que se utiliza para la transmisión se utiliza también como una fuente de energía para el dispositivo.
 - Durante los últimos años su uso se ha masificado debido a su adopción como medio de pago en el transporte público y su uso en las tarjetas de crédito.
 - Sus principales aplicaciones son: transacciones electrónicas enfocadas a los pagos y el canal fuera banda para el intercambio de credenciales (Bluetooth).
 - El riesgo de interceptación de los datos es menor debido a la cercanía necesaria para realizar la captura de los datos.



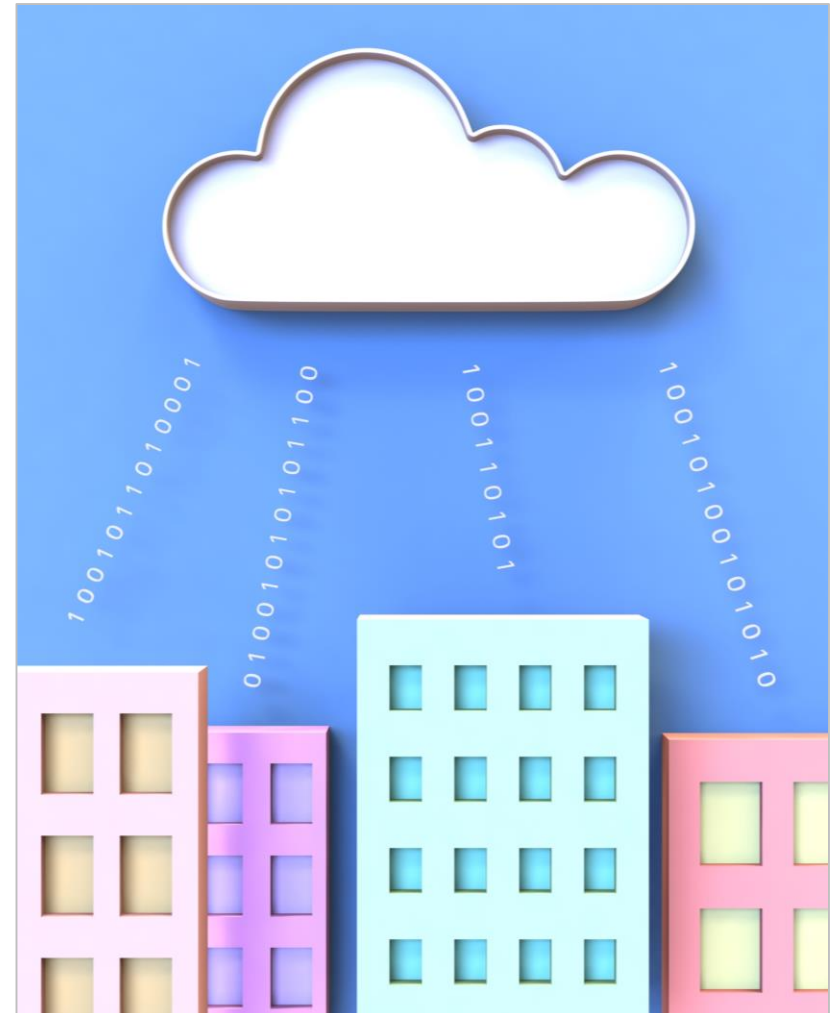


Arquitecturas de aplicación

◆◆◆ Arquitecturas de aplicación

Introducción

- En general, todas las aplicaciones se estructuran según las siguientes arquitecturas:
 - **Cliente-Servidor:** uno de los terminales (servidor) ofrece servicios que son consumidos por diferentes terminales (clientes).
 - **P2P (peer-to-peer):** todos los terminales de la aplicación ofrecen y consumen servicios.
- Los requisitos de los sistemas modernos como escalabilidad, tolerancia a fallos, redundancia, etc. han resultado en arquitecturas de aplicación mucho más complejas.
- En la práctica, un sistema puede ser dividido en varios componentes que interaccionan entre ellos utilizando las dos arquitecturas antes mencionadas.
- A continuación se presentan los principales elementos de estas arquitecturas.



Clientes - *Hardware*

- Los clientes consumen los servicios ofrecidos en la red a través de aplicaciones específicas o el navegador.
- Desde el punto de vista del *hardware* se pueden diferenciar los siguientes clientes:
 - **Estaciones de trabajo:** acceden desde redes cableadas. Equipos completamente funcionales con alta posibilidad de configuración y personalización. Típicamente usados en empresas, negocios y entornos corporativos. En casas cada vez más se ven reemplazados por portátiles.
 - **Portátiles:** igual que la estación de trabajo pero puede conectarse a la aplicación a través de redes no confiables.
 - **Móviles:** similares en capacidades a los portátiles, pero con la posibilidad de personalización y capacidades más limitadas.
 - **Dispositivos IoT (Internet de las Cosas):** capacidades de comunicación muy limitadas. Generalmente acceden a los servicios a través de dispositivos de más capacidad a los que son conectados (teléfono móvil o estación de trabajo).
 - **Servicio de *back-end*:** aplicación de servidor que necesita consumir datos de vía web *services* u otros protocolos.

Clientes - *Software*

- Los **clientes móviles** consumen los servicios ofrecidos por la red utilizando:
 - **Navegadores web:** se accede a una aplicación web utilizando el protocolo HTTP. La aplicación web se crea con HTML5/JavaScript/CSS.
 - **Aplicaciones específicas:** se desarrolla una aplicación, necesaria una por plataforma, para acceder a la web. La comunicación con el servidor se puede realizar utilizando una API (*Application Programming Interface*) del tipo:
 - HTTP (generalmente a través de servicios REST o SOAP).
 - Protocolos específicos de la aplicación.
- Para evitar problemas de seguridad, se recomienda que todos los clientes accedan al servicio a través de la misma API. De esta forma se desacopla la lógica de negocio de las capas de presentación existentes para cada plataforma.
 - Permite aislar los diferentes problemas de seguridad que puedan darse a su plataforma (cada tipo de cliente, incluidos el web y el *back-end* por otro) independientemente del tipo de cliente que acceda al servicio.

Servidores

- Un servidor es una entidad que espera, procesa y responde a las peticiones de los clientes.
- Dependiendo del tipo de aplicación, existen distintos tipos de servidores, entre otros:
 - Servidor de base de datos: para poder trabajar con gran cantidad de información.
 - Servidor de aplicaciones: para ofrecer servicios a través de distintas aplicaciones (generalmente web).
 - Servidor de correo: para el envío y recepción de correo.
- Un servicio se puede desplegar principalmente de tres maneras:
 - Utilizando un hardware e **infraestructura propia**. En estos casos el servidor puede ser:
 - Dedicado: si la aplicación no comparte el *hardware* con ningún otro servicio ofrecido.
 - Compartido: si coexisten varias aplicaciones en un mismo servidor. Generalmente, a través de un servidor de aplicaciones instalado en la propia máquina.
 - Utilizando servicios en la **nube**. Estos servicios se basan principalmente en tecnologías de virtualización para emular la existencia de varios servidores en una única máquina física de grandes prestaciones.
 - Utilizando una arquitectura **híbrida** que mezcla infraestructura propia y servicios en la nube.


Servicios en la nube

- Los servicios en la nube reducen el coste en infraestructura, *hardware* y mantenimiento para mediante la utilización de *hardware* y servicios de terceros.
- La utilización de servicios en la nube debe estudiarse con detenimiento, ya que tienen otras implicaciones como la pérdida del control de los datos, su almacenamiento en servidores externos o la exposición a riesgos como el cese de actividad de la compañía.
- Existen diferentes tipos de servicios:
 - **Infraestructura como servicio** (IaaS): ofrece infraestructura, generalmente a través de virtualización, para construir los servicios. Ejemplo: Amazon EC2.
 - **Plataforma como servicio** (PaaS): añade a la infraestructura un entorno para la ejecución de aplicaciones y bases de datos ya configurados. Ejemplo: Google App Engine.
 - **Software como servicio** (SaaS): ofrece directamente un servicio en particular.
 - Ejemplo: Microsoft Office 365, Dropbox, etc.
- Desde el punto de vista de la seguridad en dispositivos móviles, son de especial interés aquellos que permiten el desarrollo de *back-end* de forma sencilla para aplicaciones móviles. Estos tipos de PaaS se llaman *Mobile back-end as a service* (BaaS o MbaaS).
- Este tipo de servicios eliminan las tareas de administración y actualización y simplifican el desarrollo ofreciendo librerías con componentes previamente auditados.

Mobile Back-end as a Service

- Un Baas o MbaaS ofrece al desarrollador de aplicaciones un entorno pre-configurado para poder desplegar una aplicación móvil de forma rápida.
- Las tareas de administración se simplifican, por lo que el desarrollador de aplicaciones no necesita conocimientos sobre *back-end*.
- Principales proveedores:
 - App Engine (Google): una vez desarrollado el *back-end*, el *framework* genera librerías para incorporar a aplicaciones iOS y Android. *Back-end* en Java, Python, PHP o Go.
 - Parse (Facebook): la mayoría de aplicaciones se realizan desde los clientes a través de las librerías de Parse. En el *back-end* se definen políticas de seguridad y tareas periódicas que se programan en Javascript.
 - Mobile Hub (Amazon): integración de varios servicios de Amazon Web Services para el despliegue de aplicaciones móviles.

Los desarrolladores inexpertos pueden provocar problemas de seguridad en el lado del servidor.



Descripción detallada de la
arquitectura interna de los sistemas

A group of green Android robots standing in a line, with the word "Android" overlaid in white text.

Android

Introducción

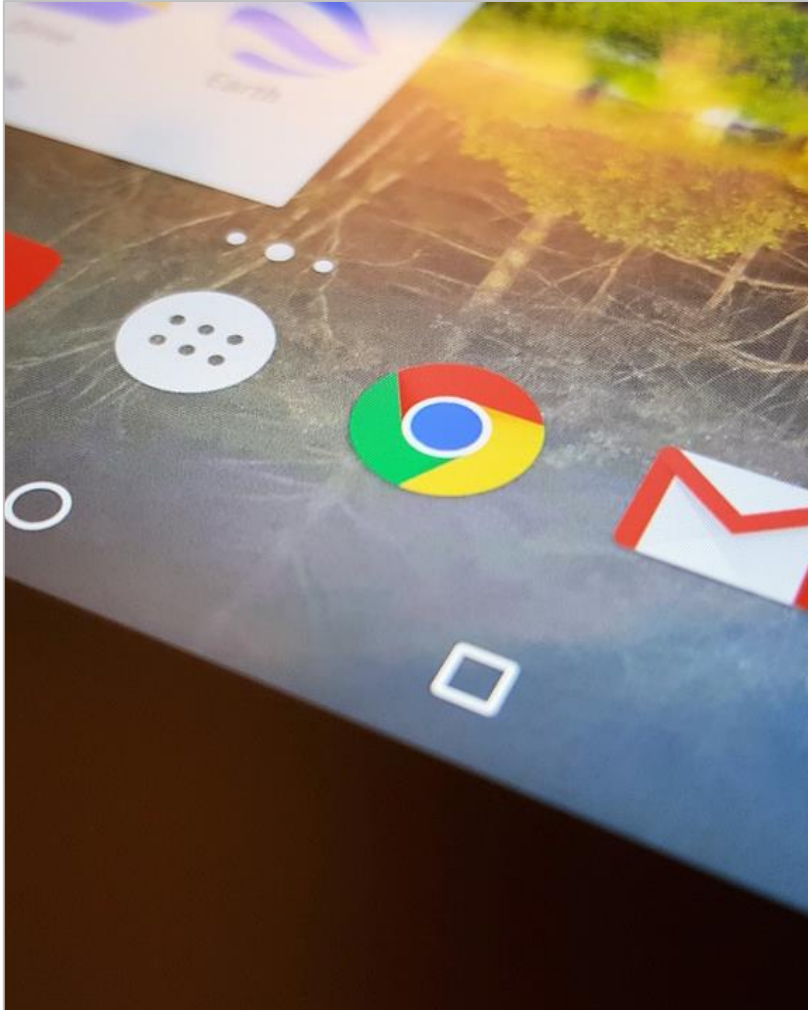


- Esta sección ofrece una versión general del sistema operativo Android.
- En concreto, abordaremos:
 - Arquitectura del sistema operativo.
 - Librerías disponibles.
 - Sistema de ficheros y particiones existentes.
 - Estructura de un binario.
 - Componentes de una aplicación.
 - comunicación entre aplicaciones.
 - Proceso de compilación y creación de una aplicación.
- Para obtener más información sobre el sistema operativo se puede visitar: <http://source.android.com>

Especificaciones técnicas

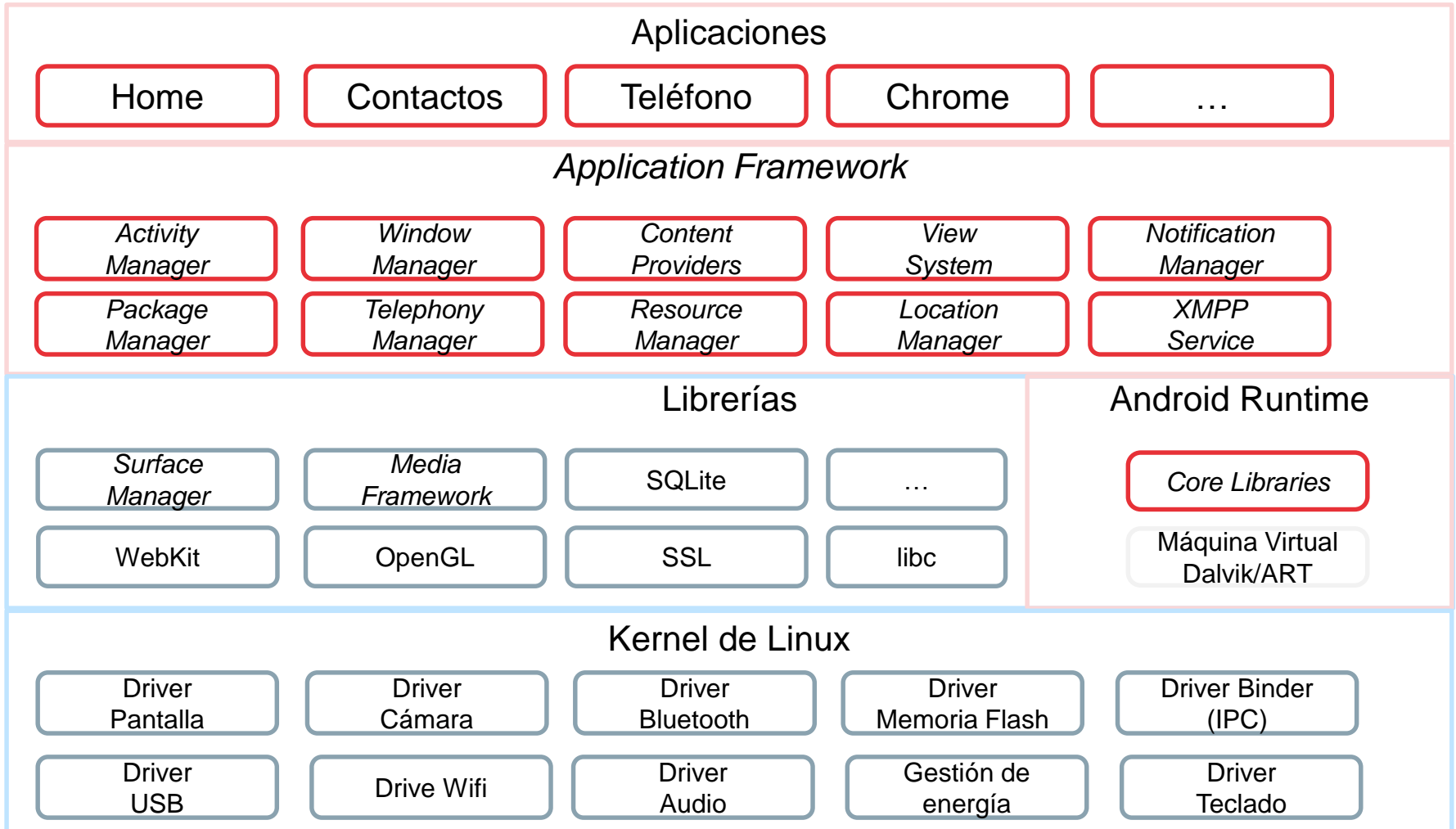
Arquitecturas válidas	ARM 32 y 64 bits, x86 32 y 64 bits, MIPS y NEON.
<i>Kernel</i>	<i>Kernel</i> de Linux 3.X dependiendo de la versión de Android y el dispositivo.
Sistema de ficheros	Soporta varios, pero principalmente utiliza EXT4 o JFFS2 para el sistema de ficheros internos. Acepta tarjetas de memoria formateadas en FAT32. Soporta cifrado de disco desde la versión 4.3.
Ejecutables	<i>Bytecode</i> . Ejecutable por la máquina Dalvik o por el más reciente Android Runtime (ART).
Plataforma del sistema	Basado en Linux.
Licencia	Licencia Apache 2.0. Los fabricantes modifican el código del sistema y lo adaptan a los dispositivos con libertad dentro de unos parámetros mínimos comunes.

Arquitectura del sistema operativo



- El sistema operativo se organiza en las siguiente capas:
 - **Aplicaciones:** aquellas instaladas por el usuario o pre-instaladas en el sistema.
 - ***Application Framework*:** ofrecen servicios a las aplicaciones. Desarrollados en Java.
 - **Librerías:** módulos que ofrecen servicios al *application framework*. Desarrollados en C y compilados a código nativo en cada plataforma.
 - **Android Runtime:** cada aplicación ejecuta su propia instancia de una máquina virtual de Java, específica para Android. Las aplicaciones nativas se ejecutan directamente.
 - **Kernel:** ejecuta código dependiente del dispositivo (ARM, x86, MIPS, etc.). Ofrece servicios de seguridad a las capas superiores.

Arquitectura del sistema operativo



Librerías disponibles

- Las aplicaciones acceden a las librerías a través del *Application Framework*.
 - **Package Manager**: controla la instalación de paquetes.
 - **Activity Manager**: gestiona las actividades que se muestran en pantalla.
 - **Location Manager**: ofrece información sobre la localización del dispositivo.
 - **Notification Manager**: permite gestionar las notificaciones recibidas por una aplicación.
 - **Content Providers**: ofrece acceso a datos almacenados por aplicaciones en bases de datos.
 - **View System**: controla las diferentes vistas que se muestran al usuario.
 - **Bluetooth API**: controla las conexiones Bluetooth del dispositivo.
- Para más información sobre librerías, visitar:
<http://developer.android.com/guide/index.html>

Sistemas de ficheros

- Android, por defecto, utiliza múltiples particiones, generalmente formateadas en *Journal Flash File System 2* (JFFS2).

Cualquier fabricante puede utilizar otro sistema de ficheros o modificar la estructura de ficheros o particiones.

- Principales directorios (se explicarán con mayor detalle en el tema 4):
 - **/system**: directorio donde se guarda la ROM del SO (solo lectura).
 - **/proc**: información de los procesos en ejecución.
 - **/mnt**: punto de montaje para otros tipos de almacenamiento.
 - **/sdcard**: redirige a **/mnt/sdcard**, punto de montaje de la tarjeta SD.
 - **/cache**: guarda la chache de datos de las aplicaciones y el sistema.
 - **/data**: directorio en el que se almacenan las aplicaciones.

Cada aplicación es guardada en un directorio al que solo ella tiene permisos de acceso.

Componentes de una aplicación

Activity

Service

Content
Provider

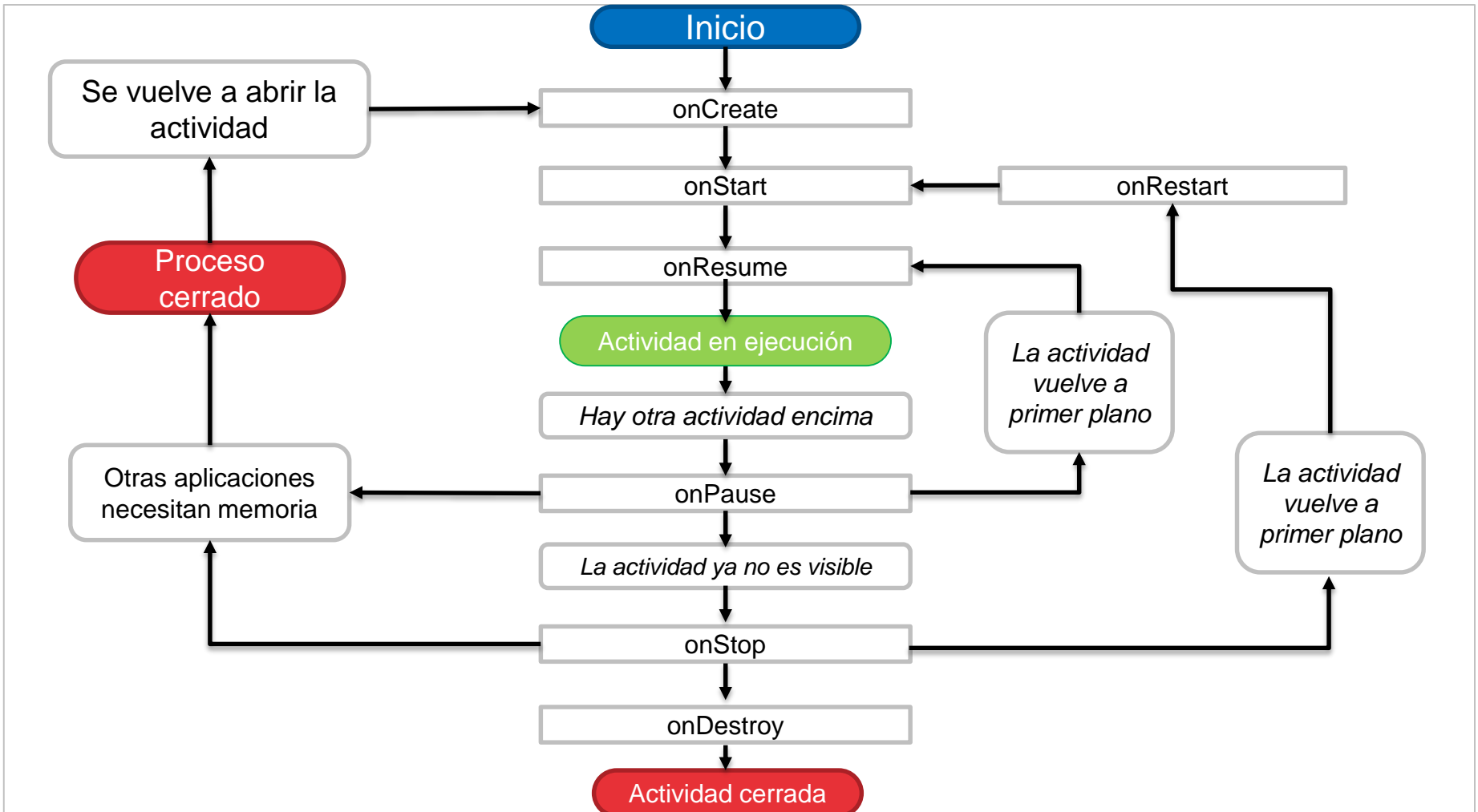
Broadcast
Receiver

- Las **actividades** constituyen la capa de presentación de la aplicación.
- Ofrecen el interfaz visible de la aplicación.
- Ocupa toda la pantalla.

Para mostrar elementos de información que no ocupan toda la pantalla se utilizan los *Fragments*.

- Cada actividad contiene una jerarquía de vistas (*Views*) con las que el usuario puede interactuar.
- Cada aplicación puede tener una Actividad principal que será la que lanzará cuando se toque el icono de la aplicación en la pantalla de apps.
- Cada actividad de la aplicación tiene un ciclo de vida como el mostrado a continuación.

Ciclo de vida de una actividad



Componentes de una aplicación

Activity

Service

Content
Provider

Broadcast
Receiver

- Los **servicios** son los encargados de realizar operaciones en el *background*. Cuando una aplicación deja de estar en primer plano puede seguir ejecutándose a través de servicios.
- Generalmente, son utilizados para realizar cálculos, guardar o recibir datos de internet o disco.
- Una de las formas de ejecutar tareas de servicios es a través de la utilización de *IntentServices*:
 - Por cada *IntentService* el sistema crea un único *thread* separado del de interfaz de usuario.
 - Cada tarea enviada al *IntentService* (por medio de un *Intent*) es ejecutada en el *thread* (no hay ejecución concurrente de varios *Intents*).

Componentes de una aplicación

Activity

Service

Content
Provider

Broadcast
Receiver

- Los **Content Providers** están diseñados para compartir datos estructurados entre aplicaciones.
- Ofrecen un interfaz para que otras aplicaciones puedan acceder a los datos almacenados por la aplicación.
- Las aplicaciones que quieren compartir datos con el resto deben contar con sus propios *providers*.
 - Es una buena práctica de seguridad requerir permisos para restringir las aplicaciones que pueden acceder a los permisos declarados por la aplicación.
- El sistema operativo ofrece por defecto una serie de *Providers* para el acceso a datos del sistema como los contactos, listado de llamadas y SMS. Para acceder a ellos, la aplicación tiene que declarar los permisos correspondientes.

Componentes de una aplicación

Activity

Service

Content
Provider

Broadcast
Receiver

- Los **Broadcast Receivers** son tareas que se ejecutan cuando llegan mensajes (*Intents*) generados por otros componentes de la aplicación o por otras aplicaciones.
- Las aplicaciones pueden crear *Intents* para enviar mensajes a actividades, servicios o *Broadcast Receivers*.
- Cada *Broadcast Receiver* es configurado para escuchar un tipo específico de *Intents*.

Componentes de una aplicación

Activity

Service

**Content
Provider**

**Broadcast
Receiver**

- Los componentes de una aplicación pueden ser.
 - Públicos: otras aplicaciones pueden interactuar con ellos.
 - Privados: solo componentes de la misma aplicación (mismo user ID) pueden interactuar con ellos.
- Todos los componentes dentro de una aplicación se ejecutan dentro del mismo proceso.
 - A no ser que el desarrollador especifique lo contrario.
- El desarrollador puede definir restricciones para el paso de mensajes a través de permisos.
 - Los que envíen mensajes pueden requerir a las aplicaciones que lo reciban de cierto permiso.
 - Los que reciban mensajes pueden aceptar mensajes solo de aplicaciones con ciertos permisos.

Comunicación entre aplicaciones

- Las aplicaciones en Android se pueden comunicar por cualquier mecanismo de comunicación entre procesos heredado de UNIX:
 - Sistema de ficheros, *sockets*, etc.
 - Los permisos de la aplicación siempre se aplican antes de la comunicación.
- Además, Android ofrece dos mecanismos adicionales de comunicación entre aplicaciones.
 - ***Binder***: sistema RPC (*Remote Procedure Call*) implementado sobre Linux con un driver específico. Las aplicaciones lo utilizan a través de objetos *Intent* que son enviados al sistema para lanzar Servicios, Actividades y *Broadcast Receivers*.
 - ***Content Providers***: operaciones de lectura/escritura sobre bases de datos de otras aplicaciones.

```

    . ltrim(preg_replace('/\\\\\\\\/', '/', $image_src), '/')
['_CAPTCHA']['config'] = serialize($captcha_config);
ay(
=> $captcha_config['code'],
src' => $image_src

ists('hex2rgb') ) {
rgb($hex_str, $return_string = false, $separator = ',') {
= preg_replace("/[^\0-9A-Fa-f]/", '', $hex_str); // Gets a prop
= array();
($hex_str) == 6 ) {
_val = hexdec($hex_str);
ray['r'] = 0xFF & ($color_val >> 0x10);
ray['g'] = 0xFF & ($color_val >> 0x08);
ray['b'] = 0xFF & $color_val;
trlen($hex_str) == 3 ) {
ray['r'] = hexdec(str_repeat(substr($hex_str, 0, 1), 2));
ay['g'] = hexdec(str_repeat(substr($hex_str, 1, 1), 2));
ay['b'] = hexdec(str_repeat(substr($hex_str, 2, 1), 2));
lse;
n_string ? implode($separator, $rgb

```

Comunicación entre aplicaciones - *Intents*

- Un ***Intent*** es un objeto para el envío de mensajes que contiene:
 - Información acerca de la operación que se quiere realizar, a través del parámetro *action* o *component*.
 - Datos sobre los que se quiere realizar la operación, a través de extras o una URI de origen de los datos.
 - Información adicional que puede ser de utilidad para el receptor.
- Los *Intents* son enviados al sistema operativo que se encarga de entregarlos al receptor correspondiente, no pueden ser enviados directamente.
- Existen *Intents* de dos tipos:
 - **Explícitos**: especifican el receptor a través del nombre de componente.
 - **Implícitos**: indican una acción genérica a realizar (ej.: mandar mensaje).
 - El sistema se

```
Intent i1 = new Intent(this, ActivityB.class);  
i.putExtra("KEY", "Value");  
startActivity(i);
```

Ejemplo de creación y envío de un *Intent* explícito a la ActividadB de una aplicación.

El fichero *AndroidManifest*

- El *manifest* de una aplicación contiene toda la información necesaria para la instalación y ejecución de la app por parte de Android.
 - Versión mínima de Android con la que la aplicación es compatible.
 - Nombre del paquete de la aplicación y versión.
 - Componentes incluidos.
 - Define las actividades, servicios y *broadcast* receivers que utiliza la aplicación. Los últimos también pueden declararse dinámicamente durante la ejecución de la aplicación.
 - Permisos que la aplicación solicita al usuario para ejecutarse.
 - Antes de Android 6.0.
 - Capacidades del dispositivo necesarias para ejecutar la aplicación.
 - Ejemplos: cámara, acelerómetro, etc.
 - Librerías del sistema que necesita cargar la aplicación para funcionar.

Estructura de un APK

- Las aplicaciones Android se empaquetan en ficheros APK (zip), los cuales contienen los siguientes elementos:
 - **META-INF**: directorio que contiene un listado de ficheros (MANIFEST.MF). El certificado de la app (CERT.RSA), una lista de recursos de la app y hashes SHA-1 de los ficheros indicados en el listado.
 - **lib**: directorio con el código nativo utilizado por la app para plataformas específicas (ARM, x86, MIPS).
 - **assets**: directorio con diferentes elementos utilizados por la app.
 - **res**: directorio con recursos utilizados por la app (iconos, imágenes, constantes, etc.).
 - **resources.arsc**: ficheros XML pre-compilados con definiciones del interfaz de usuario.
 - **classes.dex**: código compilado de la aplicación para el *runtime* de Android.
 - **AndroidManifest.xml**: fichero de información de la aplicación.

META-INF

lib

assets

res

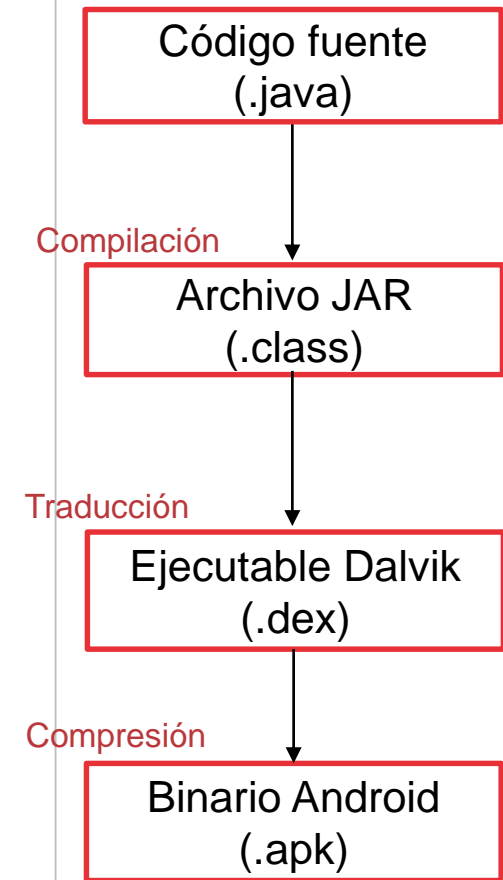
resources.arsc

classes.dex

AndroidManifest

Creación de ficheros APK

- Los diferentes ficheros fuente de java son compilados en un fichero JAR (*Java archive*).
 - Este formato combina múltiples ficheros .class compilados en *bytecode* en un fichero, utilizando la compresión zip.
- El *bytecode* es transformado al formato “Ejecutable Dalvik” para su ejecución en Android.
 - Es ejecutado por Dalvik y ART, el *runtime* introducido en las últimas versiones de Android.
- El fichero dex resultante es comprimido en un fichero zip junto con los recursos, assets y manifest de la aplicación.
- Si la aplicación se quiere publicar en Google Play se firma con un certificado.
 - Puede ser auto firmado, pero todas las versiones de la app deben ser firmadas con el mismo.

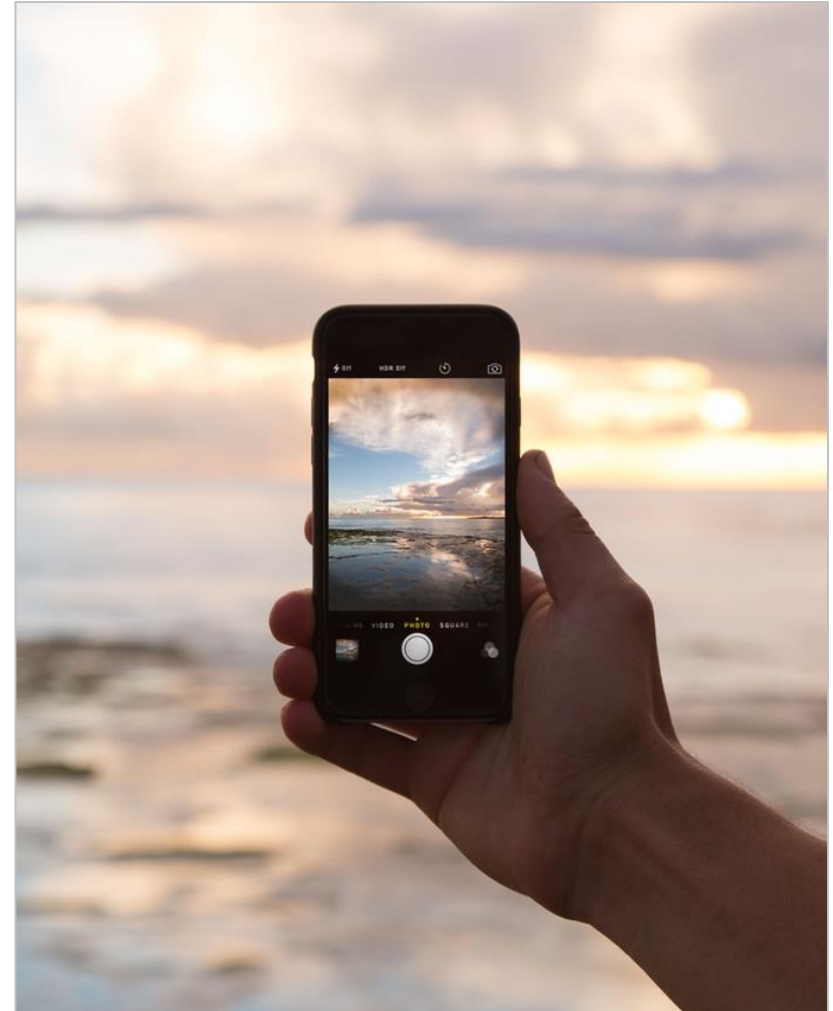




iOS

Introducción

- Esta sección ofrece una visión general del sistema operativo iOS de Apple.
- En concreto, abordaremos:
 - Arquitectura del sistema operativo.
 - Librerías disponibles.
 - Sistema de ficheros y particiones existentes.
 - Estructura de un binario.
 - Componentes de una aplicación.
 - Comunicación entre aplicaciones.
 - Proceso de compilación y creación de una aplicación.
- Para obtener más información sobre el sistema operativo se puede visitar:
 - <https://developer.apple.com/ios/>



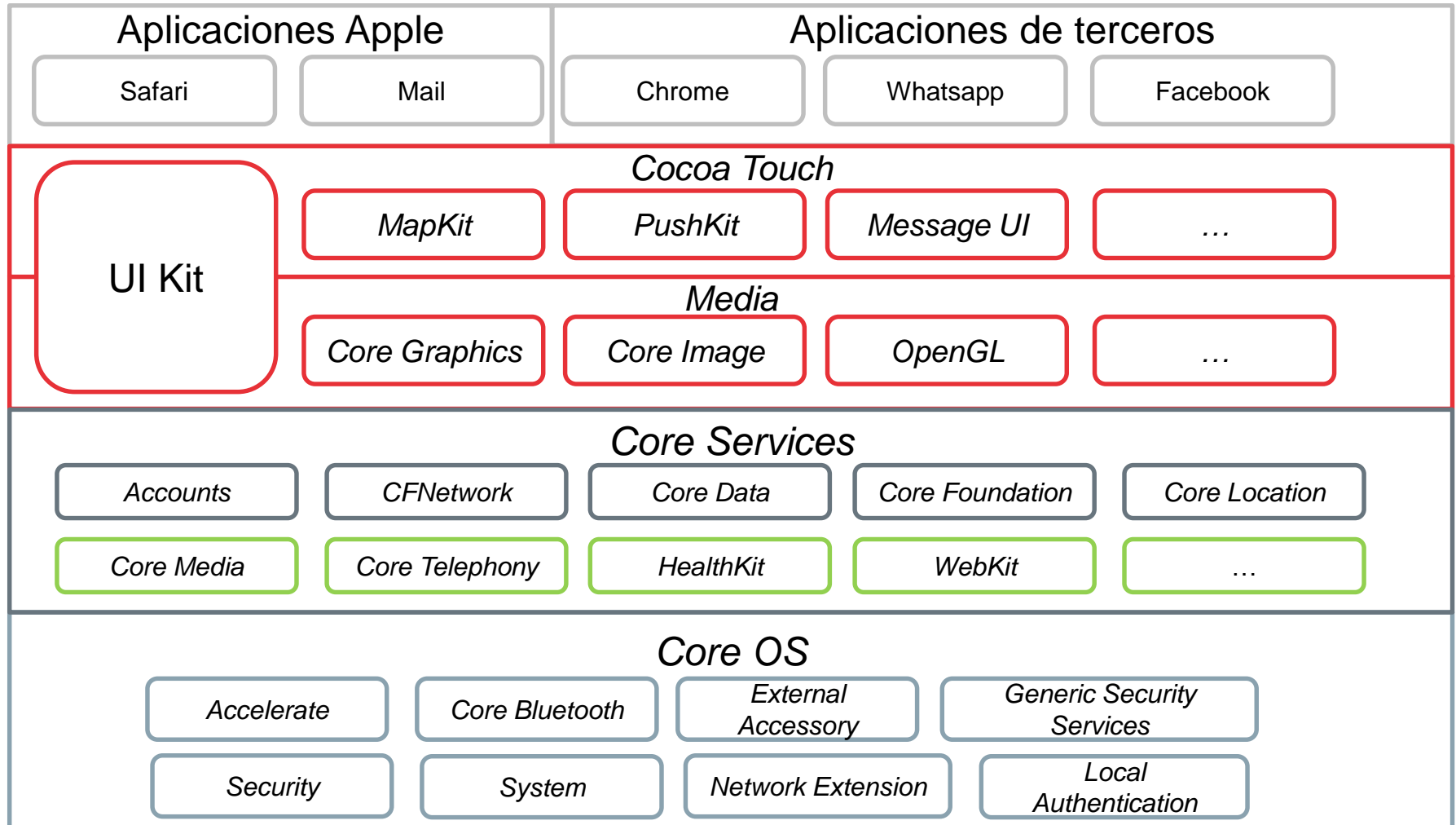
Especificaciones técnicas

Arquitecturas válidas	ARM 32 y 64 bits.
Kernel	XNU. Basado en el kernel Darwin de OS X.
Sistema de ficheros	HFSX. Basado en HFS+ de iOS. Sin almacenamiento extraíble y con cifrado de disco por defecto.
Ejecutables	Formato Match-O, el mismo que el utilizado para aplicaciones de OS X.
Plataforma del sistema	Basado en BSD, igual que OS X.
Licencia	Propietario. No se permite su instalación en equipos que no sean manufacturados por Apple. Su licencia prohíbe la ingeniería inversa del sistema.

Arquitectura del sistema operativo

- El sistema operativo se organiza en las siguientes capas:
 - **Aplicaciones:** incluye las aplicaciones de terceros y las pre-instaladas por Apple. Las últimas tienen acceso a un conjunto de APIs adicional.
 - **Cocoa Touch:** ofrecen la infraestructura básica a todas las aplicaciones (interfaz, *multi-touch*, multitarea, notificaciones, etc.).
 - **Media:** contiene las librerías de bajo nivel que se encargan de mostrar gráficos, audio y vídeo.
 - **Core Services:** ofrece diferentes servicios del sistema a las aplicaciones (redes, localización, iCloud, etc.).
 - **Core OS:** librerías de bajo nivel que son utilizadas por el resto de capas superiores para la ejecución de tareas. Todos los *frameworks* llaman a estas librerías para interactuar con el núcleo del sistema operativo.

Arquitectura del sistema operativo



Librerías disponibles

- iOS ofrece multitud de librerías por defecto para la creación de aplicaciones:
 - **UIKit:** se encarga de gran parte de la gestión de las aplicaciones.
 - Ejemplos: ciclo de vida, creación del interfaz de usuario, *multi-touch*, cámara, etc.
 - **Core Graphics, Animation, Image, OpenGL y Metal:** ofrecen motores para el dibujo de gráficos en 2D y 3D.
 - **CFNetwork:** conjunto de librerías para la creación de conexiones de red (HTTP, SSL, DNS, FTP, etc.).
 - **Core Data:** ofrece persistencia a la aplicación a través de la creación de un modelo de datos.
 - **Core Motion:** permite tratar la información recibida por los sensores del dispositivo.
 - **Core Location:** permite la utilización del GPS del dispositivo por parte de las aplicaciones.



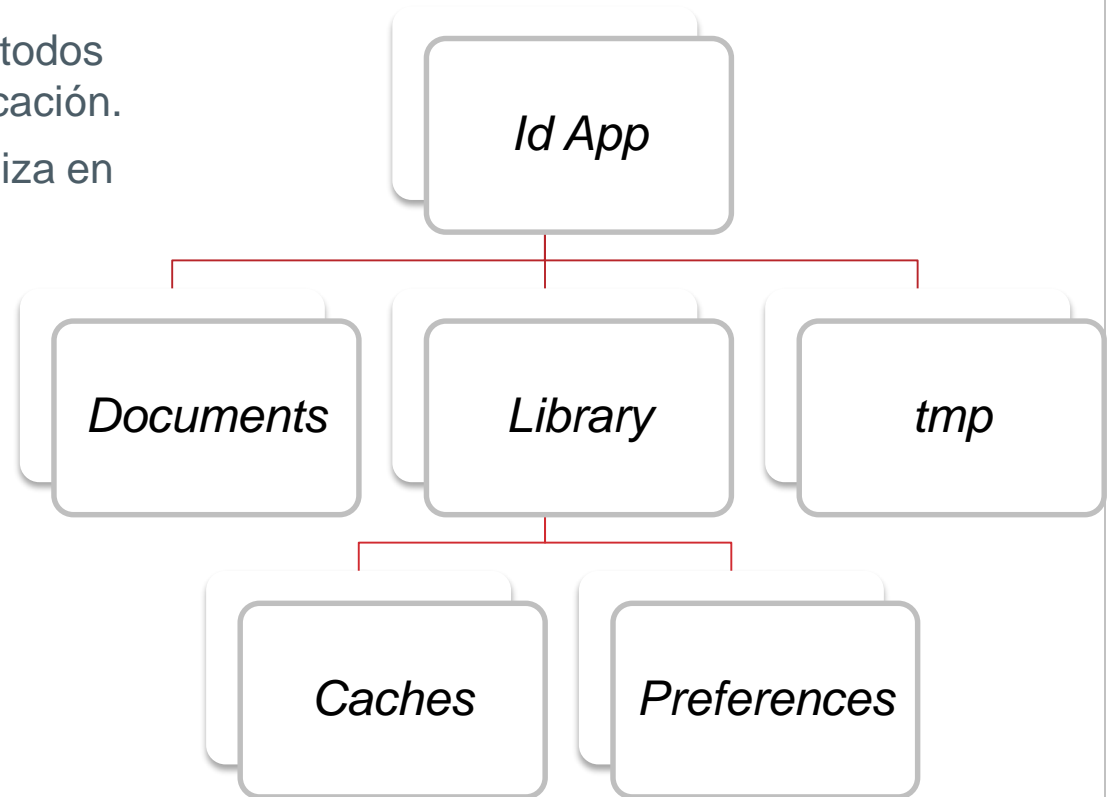
Sistemas de ficheros

- iOS utiliza el sistema de ficheros HFS+ desarrollado por Apple.
- La estructura de directorios está heredada de UNIX con ciertas modificaciones:
 - **/Applications**: directorio en el que se encuentran instaladas las aplicaciones oficiales de Apple.
 - **/private/var**: contiene la partición con todos los datos del usuario.
 - **/private/var/mobile/Applications**: directorio en el que se instalan las aplicaciones de terceros.
 - **/boot**: directorio en el que se guardan los ficheros de actualización.
 - **/private/etc**: contiene ficheros de configuración del sistema operativo.
 - **/Library**: contiene las librerías del sistema utilizadas por las aplicaciones.
 - **/Developer**: utilizado para librerías de desarrollo instaladas por Xcode.

Cada aplicación se almacena en su propio directorio y no puede acceder a la información de otras aplicaciones.

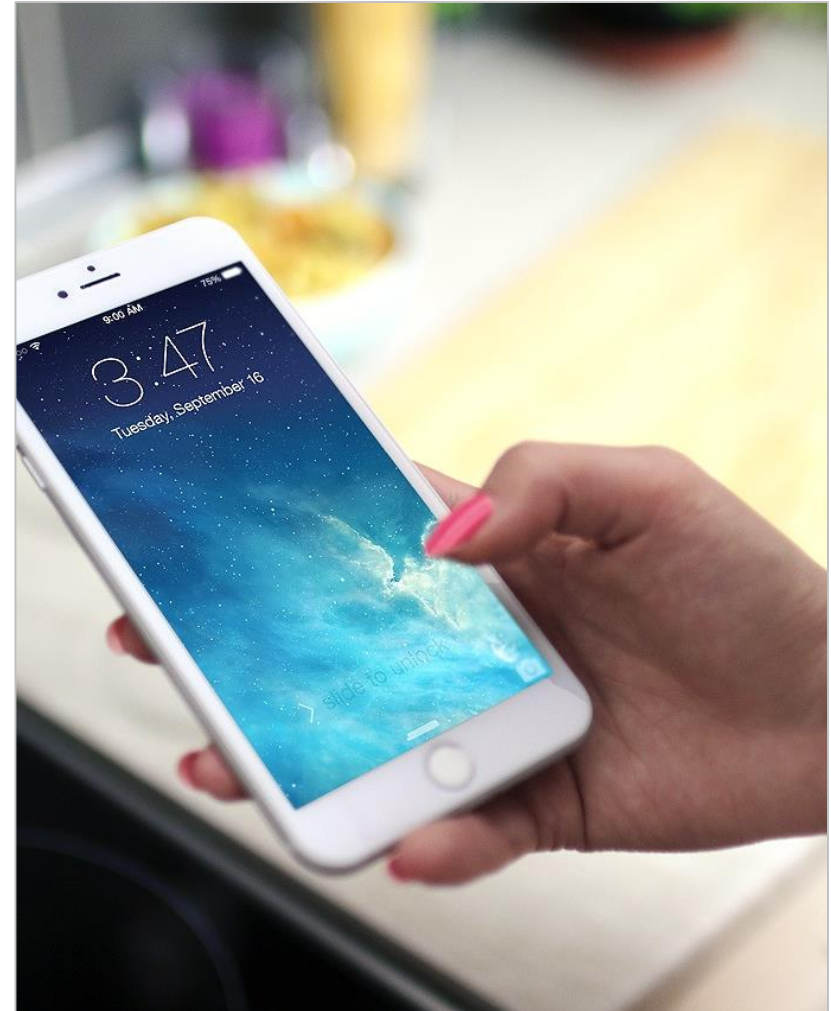
Sistema de ficheros de una aplicación

- Cada aplicación se instala en un directorio con un id único generado en tiempo de instalación.
- El directorio *Documents* contiene todos los ficheros generados por la aplicación.
- El binario de la aplicación se localiza en una carpeta terminada en app en */Containers/Bundle/Application*.
- Los otros directorios almacenan preferencias y datos temporales.

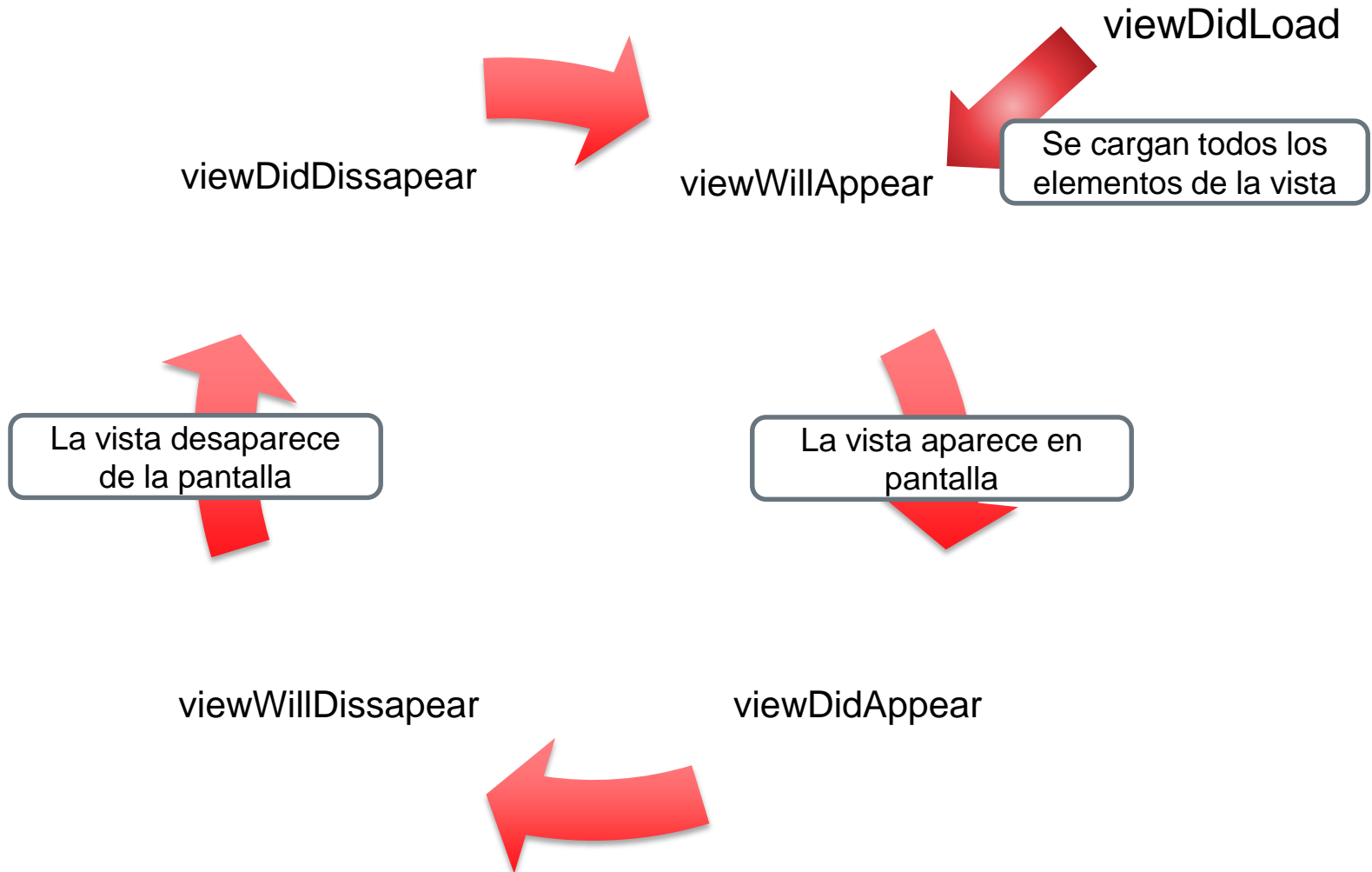


Componentes de una aplicación

- Las aplicaciones en iOS están compuestas por una serie de elementos comunes:
 - **AppDelegate:** es el encargado de mediar entre el sistema operativo y la aplicación. La mayoría de eventos importantes que pueden interferir en el uso de una app (ej.: llamada telefónica) son manejados por él.
 - **ViewControllers:** son los componentes principales de la aplicación. Cada aplicación debe tener al menos uno, aunque suelen tener varios. Un *ViewController* maneja la porción del interfaz mostrado al usuario y gestiona la interacción entre el interfaz de usuario y los datos.
 - **Storyboards:** definen las interfaces de usuario y transiciones entre las diferentes pantallas de la aplicación, aunque también se pueden definir de forma programática.



Ciclo de vida de un controlador



Ciclo de vida de un controlador

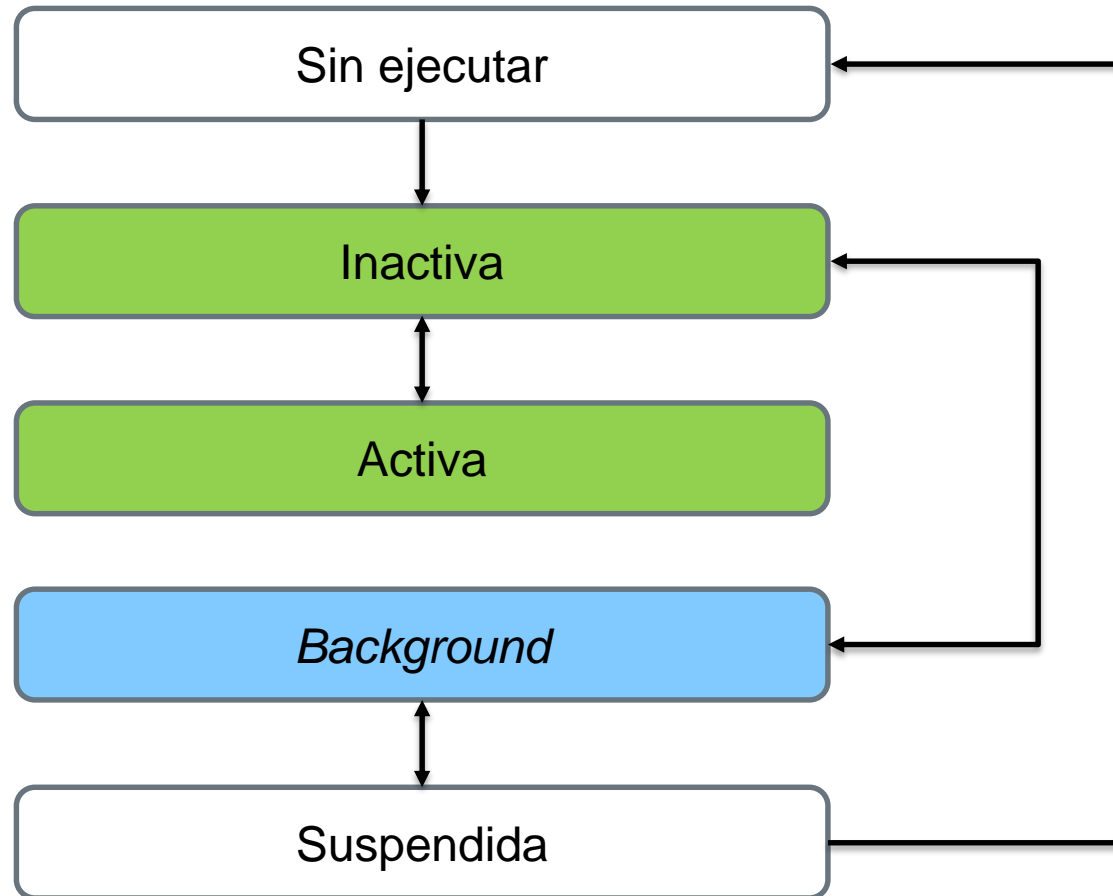
Aún no ha sido ejecutada

En primer plano y sin recibir eventos

En primer plano y recibiendo eventos

Ejecutando tareas antes de ser suspendida

Sin ejecutar código



Comunicación entre aplicaciones

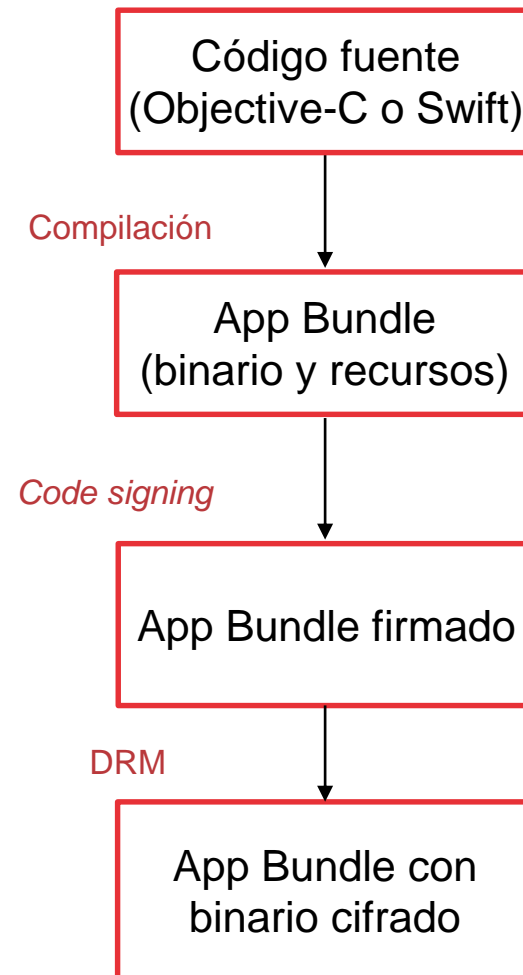
- La comunicación entre aplicaciones en iOS siempre se realiza con la mediación del sistema operativo. El usuario siempre debe iniciar o autorizar la comunicación entre dos aplicaciones.
- Existen cuatro opciones principales:
 - **AirDrop**: solo permite la comunicación con aplicaciones en otros dispositivos, no con aplicaciones dentro del mismo dispositivo.
 - **Esquemas URL**: las aplicaciones pueden registrar tipos de URL que son capaces de abrir (SMS, Phone, HTTP, Spotify, etc.).
 - **App Extensions**: permite extender ciertas funcionalidades del sistema operativo con una aplicación. Sus categorías están restringidas (almacenamiento en la nube, notificaciones, compartir, etc.) y la interacción se realiza a través de APIs específicas.
 - **Document Interaction API**: la aplicación define los documentos que es capaz de abrir. Si la aplicación es seleccionada para abrir el fichero, recibe una URL para leerlo.

Estructura de un binario

- Los binarios descargados de la App Store de Apple son paquetes comprimidos en Zip con la extensión IPA.
- El fichero IPA contiene, entre otros, los siguientes directorios y ficheros:
 - ***/Payload/Application.app***: directorio que incluye:
 - Aplicación compilada: cifrada con DRM, por lo que no se puede acceder directamente al código compilado. Es necesario extraerlo de un dispositivo con *jailbreak*.
 - Recursos utilizados por la app: imágenes, vídeo, audio, etc.
 - *Info.plist*: fichero con información de configuración para la instalación y ejecución de la aplicación.
 - *_CodeSignature*: firma de la aplicación realizada con el certificado de desarrollador emitido por Apple.
 - ***/ITunesArtwork***: contiene las imágenes utilizadas para los iconos de la aplicación.
 - ***/ITunesMetadata.plist***: contiene una ficha con información relevante de la app para su instalación y ejecución: Bundle ID, id del desarrollador, etc.

Creación de ficheros binarios

- Para crear un fichero binario distribuible a través de la App Store es necesario disponer de un certificado emitido por Apple. Desde iOS 9, la instalación de aplicaciones en dispositivos personales a través de Xcode no necesita de certificado.
- Xcode se encarga de compilar la aplicación a la arquitectura ARM, firmar el código y subir el binario a Apple (si el certificado y *provisioning profile* del desarrollador y la aplicación están en regla).
- Apple revisa la aplicación y protege el código compilado mediante DRM, si aprueba la aplicación.



Windows y BlackBerry

A top-down view of a person's hands working on a wooden desk. The right hand holds a black Windows Phone, displaying its home screen with various app tiles. The left hand is positioned near three markers (green, orange, yellow) on the left side of the desk. In the background, there are several sheets of paper with design sketches, including a calendar and a grid. A yellow and a white smartphone are also visible on the desk. A semi-transparent grey box with the text 'Windows y BlackBerry' is overlaid in the center.

Especificaciones técnicas

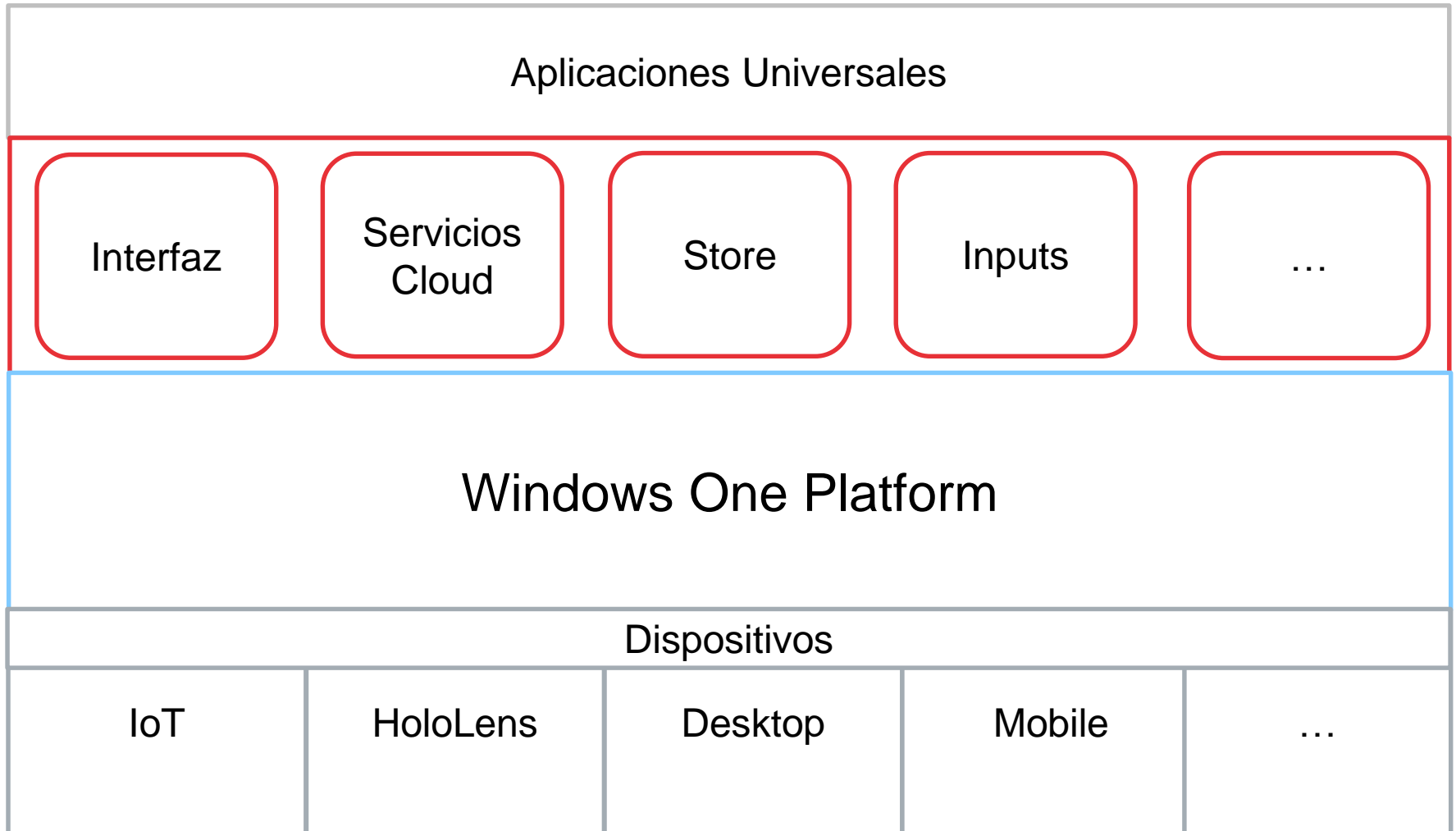
Arquitecturas válidas	ARM 32 y 64 bits, X86 y X64 y otras arquitecturas IoT.
<i>Kernel</i>	Windows One Core para todas las plataformas.
Sistema de ficheros	NFTS por defecto. FAT en tarjetas SD.
Ejecutables	Universal Windows Apps (C#, C, C++, JavaScript).
Plataforma del sistema	Windows. Específica para cada dispositivo en el que se instala (Xbox, móvil, escritorio, etc.).
Licencia	Propietario.

Arquitectura

- Windows 10 trata de unificar la arquitectura de todas las plataformas de Microsoft en una sola plataforma integrada con:
 - Un tipo único de *kernel*.
 - Un único sistema de archivos.
 - Un modelo de aplicación único para todas las plataformas.
- Windows 10 integra las siguientes plataformas:
 - Windows para escritorio.
 - Windows Phone.
 - Xbox One.
 - Windows para IoT.
- De esta manera, una aplicación desarrollada para Windows 10 podrá ser ejecutada en cualquier plataforma de Microsoft (Universal App Platform).



Arquitectura



Aplicaciones

- Las aplicaciones de Windows se desarrollan en base a:
 - Un código común a todas las plataformas que incluye los interfaces de usuario que se van a mostrar. Debido al diseño de los interfaces de Windows, el contenido de los mismos se puede adaptar de forma sencilla a los diferentes dispositivos que ejecutan las aplicaciones.
 - Un código específico en forma de extensión por cada plataforma en la que se quiere ejecutar la aplicación.
- El código común de una aplicación suele contener la lógica de negocio suficiente para ejecutar la aplicación en dispositivos móviles y escritorio.
- Las extensiones permiten a la aplicación acceder a funcionalidad específica de la plataforma en la que se está ejecutando.
- Las aplicaciones deben comprobar de forma activa que se están ejecutando en la plataforma antes de acceder a la funcionalidad de la extensión.

Especificaciones técnicas

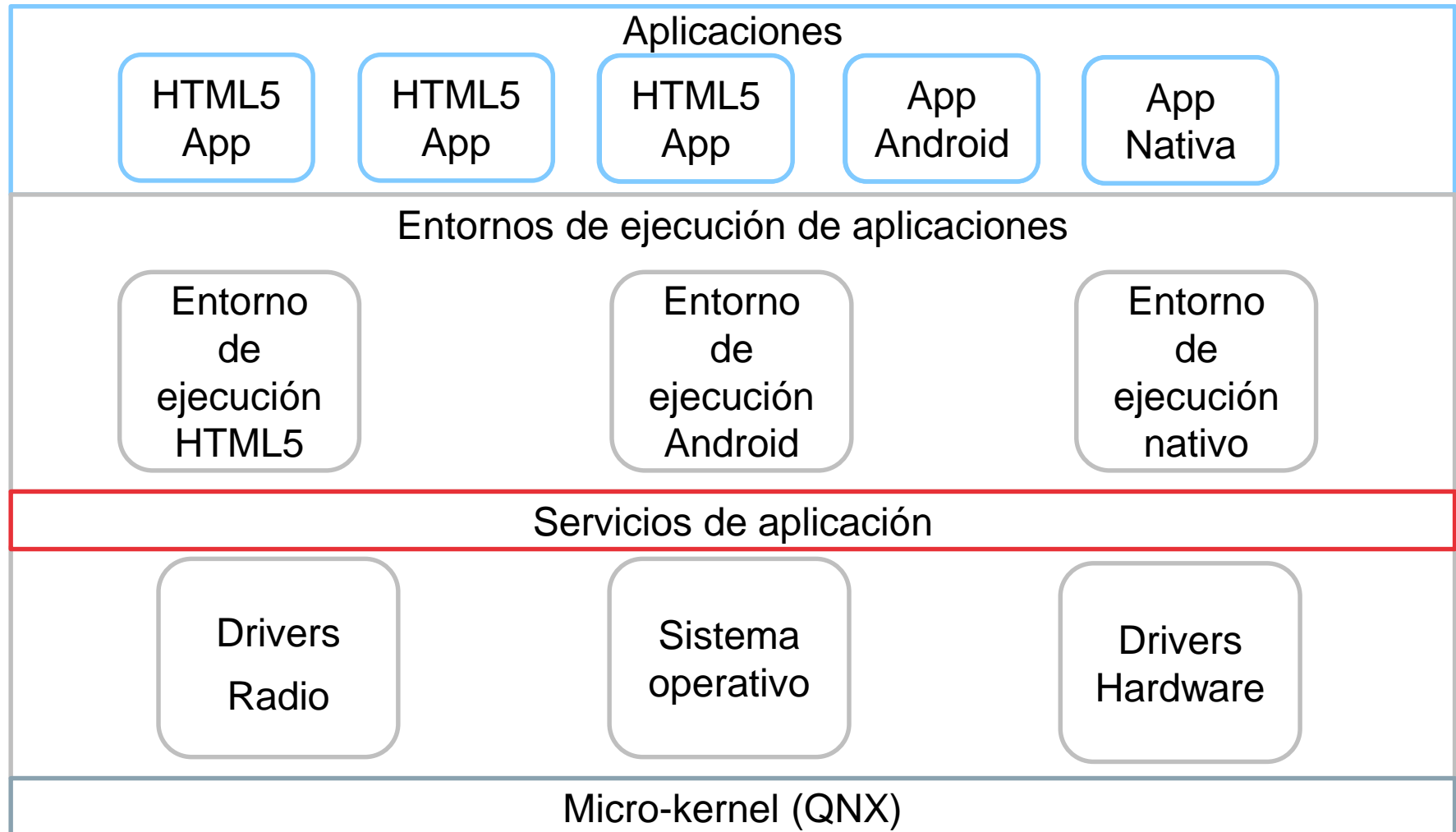
Arquitecturas válidas	ARM.
<i>Kernel</i>	RTOS Microkernel.
Sistema de ficheros	QMX para la memoria interna. FAT en tarjetas SD.
Ejecutables	C++, HTML5, Java para aplicaciones portadas de Android.
Plataforma del sistema	QMX OS (tipo UNIX).
Licencia	Propietario.

Arquitectura

- BlackBerry es un sistema operativo de *micro-kernel* (QNX Neutrino RTOS).
- La mayoría de funciones del sistema se implementan fuera del *kernel*.
- El sistema operativo y los drivers del sistema utilizan el *micro-kernel* para acceder al hardware del dispositivo a bajo nivel.
- Por encima del sistema operativo se sitúan los servicios y plataformas que ofrece el sistema operativo a cada plataforma de ejecución: HTML5, Android y Cascades (Nativas).
- Cada aplicación es ejecutada en su plataforma específica de ejecución.
- El sistema de ficheros de BlackBerry se divide en cuatro particiones:
 - Sistema Operativo: partición de solo lectura con el sistema operativo.
 - *Base File System*: contiene ficheros del sistema y también es de solo lectura.
 - *Work File System*: contiene las aplicaciones y datos del entorno de trabajo. Se cifra por defecto.
 - *Personal File System*: contiene las aplicaciones personales del usuario. No se cifra por defecto.

◆◆◆ BlackBerry 10

Arquitectura



Resumen comparativo

Especificaciones técnicas

iOS	
Arquitecturas	ARM 32/64 bits.
Kernel	XNU. Basado en el kernel Darwin de OS X
Sistema de ficheros	HFSX. Basado en HFS+ de iOS. Sin almacenamiento extraíble y con cifrado de disco por defecto.
Ejecutables	Formato Match-O, el mismo que el utilizado para aplicaciones de OS X.
Plataforma sistema	Basado en BSD, igual que OS X.
Licencia	Propietario. Su licencia prohíbe la ingeniería inversa del sistema.
Windows 10	
Arquitecturas	ARM 32/64 bits, X86, X64 y otras arquitect. IoT.
Kernel	Windows One Core para todas las plataformas.
Sistema de ficheros	NFTS por defecto. FAT en tarjetas SD.
Ejecutables	Universal Windows Apps (C#, C, C++, Javascript).
Plataforma sistema	Windows. Específica para cada dispositivo en el que se instala (Xbox, móvil, escritorio, etc.).
Licencia	Propietario.

Android	
Arquitecturas	ARM 32/64 bits, x86 32/64 bits, MIPS y NEON.
Kernel	Kernel de Linux 3.X.
Sistema de ficheros	Soporta varios, comúnmente utiliza EXT4 para el sistema de ficheros internos. Acepta tarjetas en FAT32. Soporta cifrado (desde versión 4.3).
Ejecutables	Bytecode. Ejecutable por la máquina Dalvik o por Android Runtime (ART).
Plataforma sistema	Basado en Linux.
Licencia	Licencia Apache 2.0.
BlackBerry	
Arquitecturas	ARM.
Kernel	RTOS Microkernel.
Sistema de ficheros	QNX para la memoria interna. FAT en tarjetas SD.
Ejecutables	C++, HTML5, Java para aplicaciones portadas de Android.
Plataforma sistema	QNX OS (tipo UNIX).
Licencia	Propietario.

A close-up photograph of a person's hand moving a white chess piece on a wooden chessboard. The person is wearing glasses and has a focused expression. The background is blurred, showing other chess pieces and the chessboard. A semi-transparent blue rectangular box is overlaid on the center of the image, containing white text.

Análisis de funcionalidades y características de seguridad de los sistemas operativos móviles

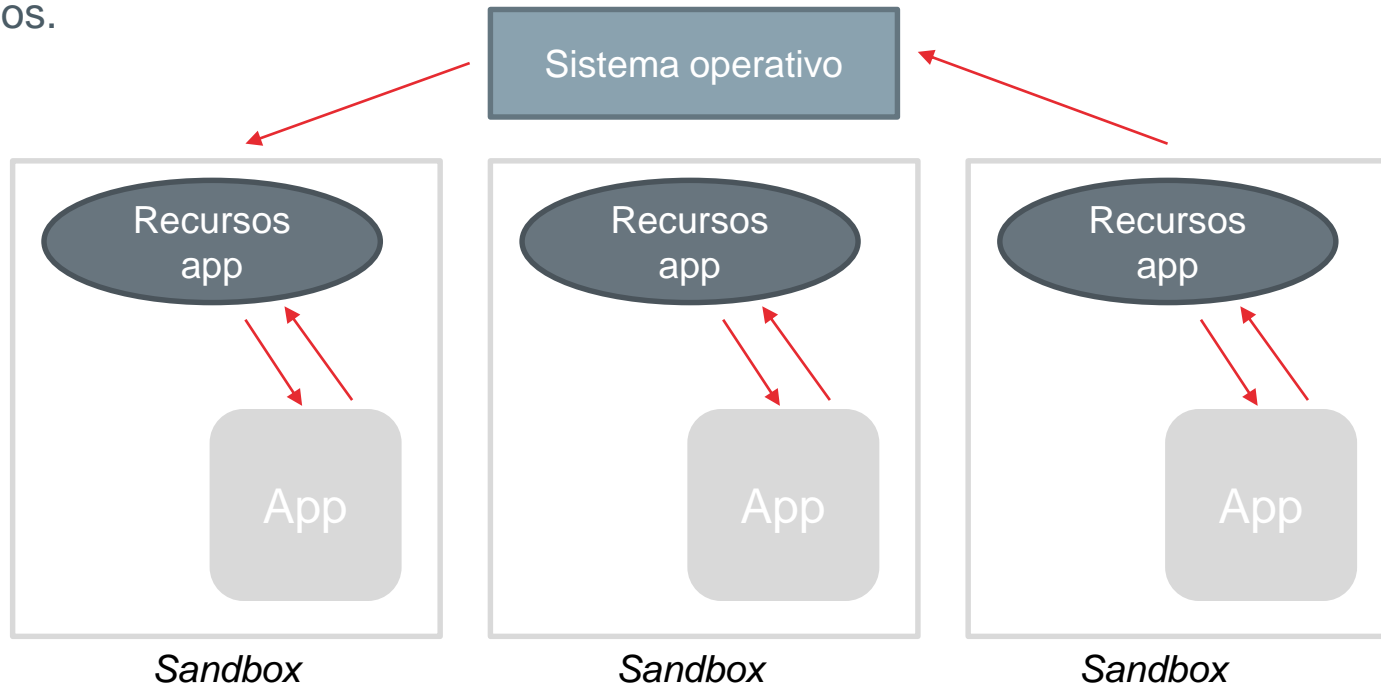
◆◆◆ La seguridad y los *smartphones*

Introducción

- Los sistemas operativos móviles han sido desarrollados con la seguridad como un aspecto transversal a toda su estructura.
- Ofrecen una serie de características de seguridad encaminadas a proteger al usuario frente amenazas internas (*software* malicioso y vulnerabilidades) y externas (pérdida o robo).
 - Aislamiento de aplicaciones (*sandboxing*).
 - Sistema de permisos.
 - Control de acceso.
 - Aplicaciones firmadas.
 - Arranque seguro.
 - Cifrado de datos en aplicaciones.
 - *Kill Switch*.
- Cada sistema operativo tiene sus particularidades en la implementación de cada mecanismo.

Introducción

- En los sistemas operativos móviles, las aplicaciones son consideradas como no confiables y tienen acceso limitado al sistema operativo y otras aplicaciones.
- La interacción entre aplicaciones se lleva a cabo siempre con la mediación del sistema operativo.
- El acceso a recursos sensibles del sistema operativo es controlado por un sistema de permisos.



Introducción

iOS



- Directorio aleatorio para cada aplicación.
- Aplicaciones de terceros se ejecutan con el usuario “*mobile*”.
- La aplicación solo es capaz de ver el sistema de ficheros accesibles desde su propia raíz.
 - El acceso a cualquier API del sistema se realiza mediante la verificación de la firma de la aplicación.
 - Solo aplicaciones firmadas por Apple pueden acceder a ciertos elementos del sistema.



Introducción

Android

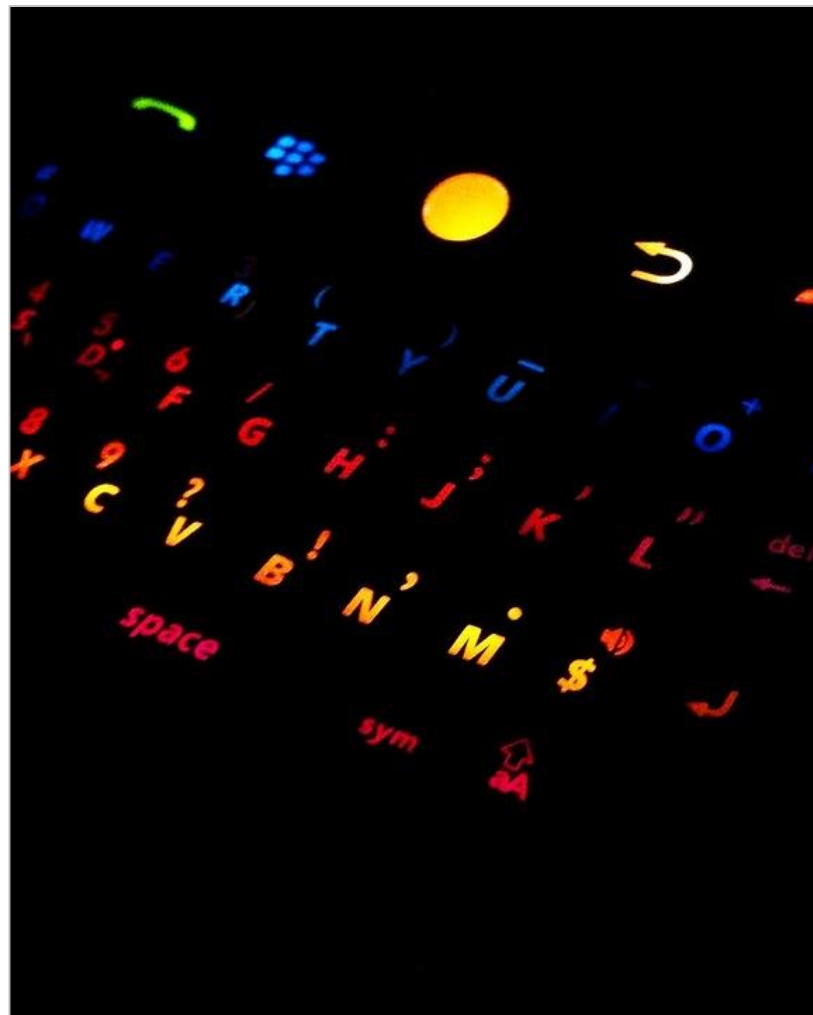


- Cada aplicación se ejecuta en su máquina virtual con su propio user-id de Unix.
 - Ninguna app puede acceder a ficheros de los que es propietaria otra app.
 - Excepto a la tarjeta SD y aplicaciones firmadas por el mismo desarrollador y con el mismo user-id activo.
- SELinux se encarga de la separación efectiva basándose en el uid y permisos de los ficheros (desde Android 4.3).



Otras plataformas

- Blackberry
 - Las *sandbox* pueden ejecutarse en dos espacios diferentes: personal y corporativo.
 - Si una aplicación se puede ejecutar en ambos, se crean dos copias con dos *sandbox* diferentes que no pueden comunicarse.
- Windows Phone
 - La *sandbox* tiene una denominación específica: *App Container*.
 - Cada contenedor tiene asociada una política de seguridad que especifica como interactúa la aplicación con el sistema operativo.
 - El desarrollador especifica la política según el acceso al sistema operativo y los permisos que requiere su aplicación.



Sistema de permisos

- Las aplicaciones móviles, se ejecutan en la *sandbox* siguiendo el esquema de mínimos privilegios.
- El acceso a los recursos sensibles del sistema operativo está protegido por permisos que las aplicaciones deben solicitar. El acceso al recurso depende del sistema operativo.
- En Android suele darse libertad total de uso, mientras que en iOS y otros sistemas el acceso es más limitado, incluso poseyendo el permiso (la realización de llamadas requiere la intervención del usuario).
- Los permisos se utilizan de forma general para acceder a recursos como:
 - La cámara y fotografías del dispositivo.
 - El teléfono.
 - Contactos.
 - Envío y recepción de SMS.
 - Acceso a Internet u otros medios de comunicación (Bluetooth, NFC, etc).

◆◆◆ Sistema de permisos

iOS

- Los permisos son solicitados en tiempo de ejecución cuando se utiliza una API sensible.
- Pueden ser revocados en cualquier momento por el usuario.
- La aplicación debe estar preparada para que le sea denegado el permiso.
- Ciertas tareas requieren la intervención del usuario (llamadas, SMS, etc.).

```
func beginSession() {  
    configureDevice()  
    if err != nil {  
        println("error: \(err?.localizedDescription)")  
    }  
    previewLayer = AVCaptureVideoPreviewLayer(session: captureSession)  
    self.view.layer.addSublayer(previewLayer)  
    previewLayer?.frame = self.view.layer.frame  
    captureSession.startRunning()  
}
```

Código simplificado en Swift que genera una petición para el uso de la cámara del dispositivo para grabar vídeo.

◆◆◆ Sistema de permisos

Android

- Antes de 6.0
 - Los permisos se solicitan en tiempo de instalación.
 - La instalación requiere que el usuario acepte todos los permisos.
 - Los permisos se agrupan por categorías para ser mostrados al usuario.
 - Ejemplo de solicitud en el fichero AndroidManifest.xml.

```
<uses-permission android:name="android.permission.SEND_SMS"/>
```

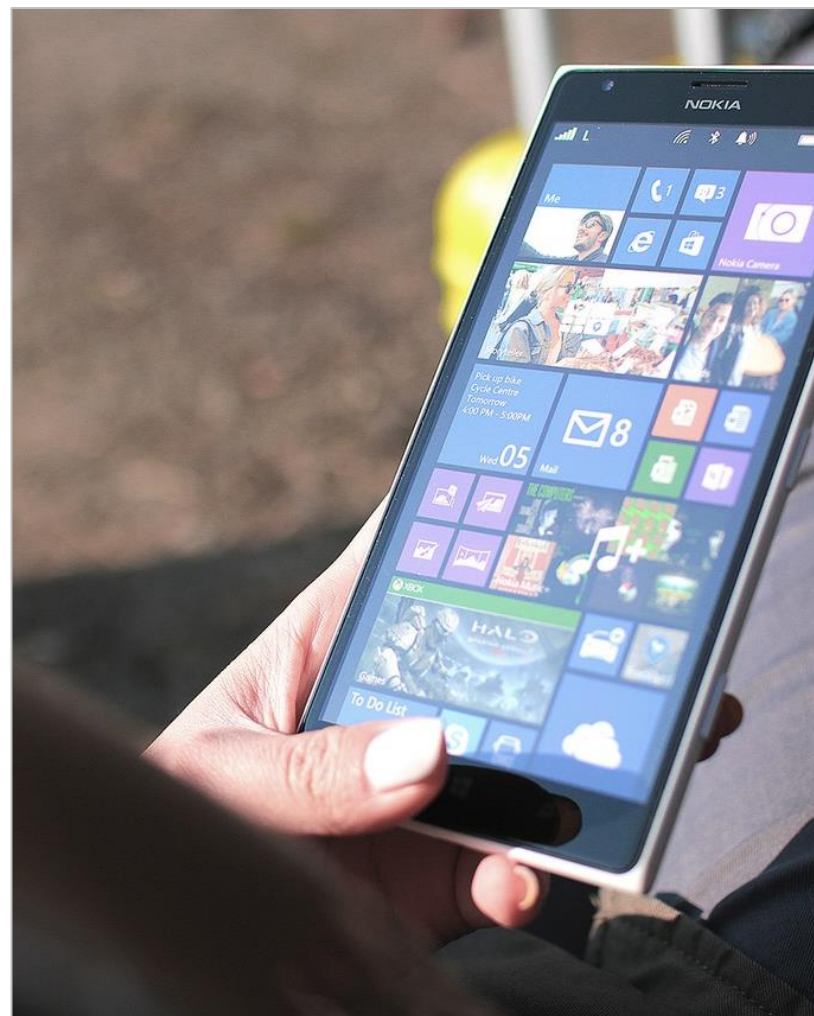
- Después de 6.0
 - Similar a iOS, los permisos son solicitados en tiempo de ejecución.
 - Ejemplo de solicitud de permiso en Android 6.0.

```
ActivityCompat.requestPermissions(thisActivity,  
    new String[]{Manifest.permission.READ_CONTACTS},  
    "Texto Explicativo");
```

◆◆◆ Sistema de permisos

Otras plataformas

- Blackberry 10
 - Los permisos son solicitados la primera vez que se abre la aplicación.
 - El usuario puede seleccionar activar o desactivar los permisos que considere.
- Windows Phone
 - Utiliza un sistema híbrido.
 - Algunos permisos son declarados y aceptados durante la instalación.
 - Con otros, se solicita autorización cuando la aplicación va a utilizar el permiso.
 - Solo pueden revocarse en teléfonos configurados en modo empresarial.



Introducción

- Los dispositivos móviles implementan un sistema de bloqueo para evitar el acceso por parte de terceros.
- Dependiendo de la configuración del sistema, el dispositivo se bloquea tras un cierto tiempo sin ser utilizado.
- Esto evita el acceso al dispositivo (y sistema de ficheros) por parte de terceros.
 - Excepto al almacenamiento extraíble (no afecta a iOS).
- Algunos de los mecanismos de protección del sistema operativo no se activan hasta que no se ha activado un sistema de desbloqueo seguro.
 - El almacén de certificados de Android.
 - La protección adicional de ficheros en iOS.

Métodos de bloqueo

- Código de n dígitos.
 - En caso de repetidos fallos retrasan los nuevos intentos.
 - Se pueden configurar para que se realice un borrado del terminal en caso de superar un límite de intentos máximo.
 - Mediante fuerza bruta, por ejemplo en iOS si el teléfono tiene *jailbreak* se puede conseguir identificar un código de bloqueo de 4 dígitos en 16 horas ([vídeo](#)).
- Patrón de desbloqueo.
 - Muy utilizado en terminales Android, susceptibles a [smudge attacks](#).
 - Son más [predecibles](#) de lo que pueda parecer en un principio.
- Contraseña con teclado completo.
 - Seguridad similar a la existente en los sistemas de escritorio.
- Información biométrica (huella dactilar, rostro).
 - Para poder utilizarlos la primera vez es necesario utilizar un código por desbloqueo la primera vez.
 - Son susceptibles a los ataques de spoofing (copiado de [huellas dactilares](#)).

Importancia de las firmas

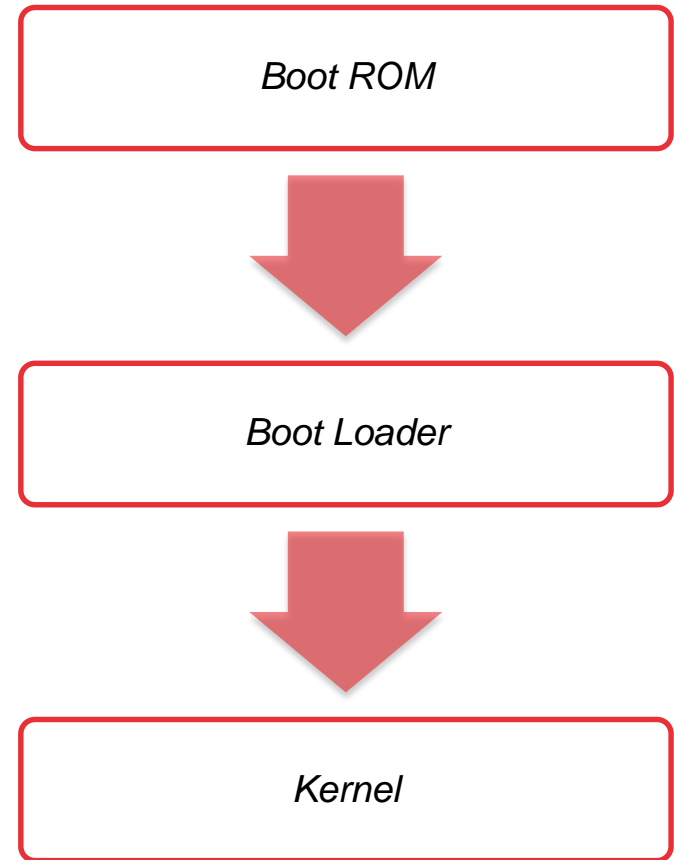
- Todas las plataformas móviles requieren que las aplicaciones ejecutadas en el dispositivo hayan sido firmadas previamente.
- Cada plataforma lo gestiona de una forma diferente:
 - En Android:
 - El certificado utilizado puede ser auto-firmado.
 - Se utiliza simplemente para identificar las aplicaciones creadas por un desarrollador, por lo que no es un mecanismo de seguridad óptimo.
 - No es necesaria la distribución a través de Google Play.
 - En el resto de plataformas:
 - Cada aplicación es firmada con un certificado que debe estar emitido por la autoridad certificadora válida de la plataforma (Apple, Microsoft o RIM).
 - Solo las aplicaciones firmadas con un certificado aprobado por Apple pueden ser ejecutadas por los dispositivos físicos.
 - En iOS, se pueden ejecutar aplicaciones sin firmar siempre y cuando se dispone del código fuente de la misma y se instala a través de XCode 7.
 - Los programas para empresas permiten a organizaciones obtener certificados especiales para firmar aplicaciones sin tener que pasar por la tienda oficial.

Integridad

- Para asegurar la integridad del sistema operativo, aplicaciones y datos del usuario, los dispositivos se tienen que asegurar de que solo ejecutan código que ha sido aprobado por el fabricante.
- Este proceso comienza desde el propio inicio del sistema operativo.
- El dispositivo debe asegurarse de que el código que los diferentes niveles del sistema operativo que está ejecutando no ha sido alterados.
- En general, todos los fabricantes siguen un esquema similar de arranque seguro.
- El proceso se divide en varios pasos. Cada paso verifica la integridad del siguiente elemento a ejecutar y lo carga en caso de que la verificación sea positiva.
- Si la verificación falla, el dispositivo entra en modo recuperación y debe ser restaurado (si no se ha destruido la información de la *Boot ROM*).

Inicio del sistema

- La Boot ROM contiene código de solo lectura que se carga al encender el dispositivo.
- Este código es aceptado implícitamente por el dispositivo, ya que se encuentra guardado en el propio procesador y asume que no puede ser modificado (tamper-proof).
- La Boot ROM contiene un certificado que es utilizado para comprobar la integridad del Boot Loader.
- Si la verificación es correcta se carga el siguiente paso del sistema operativo.
- En caso de que haya más de un procesador (secure enclaves, baseband), cada uno ejecuta un proceso de inicio independiente.



Introducción

- Todos los sistemas operativos móviles escriben a disco los datos cifrados.
- El principal uso de este sistema es doble:
 - Borrado rápido del sistema de forma remota (requiere del olvido de una clave), en vez del borrado activo de los datos).
 - Acceso a datos del dispositivo si se encuentra bloqueado.
 - Las aplicaciones acceden al sistema de ficheros descifrado, por lo que no es útil para proteger un dispositivo que se encuentra desbloqueado.
- A continuación veremos cómo funciona en Android e iOS.



Android

- Los datos son cifrados utilizando AES 128 en modo CBC.
- El cifrado en Android está disponible desde la versión 4.0 y es obligatoria desde Android 5.0.
- La clave de cifrado maestra es aleatoria y se cifra siguiendo el siguiente procedimiento, en dispositivos con *Trusted Execution Environment* (TEE):
 - Se deriva una clave de 32 bytes del código de bloqueo del dispositivo (PIN o contraseña).
 - Se firma la clave anterior con una clave generada (y almacenada) en el TEE.
 - Del resultado anterior se deriva una nueva clave de 32 bytes.
 - El resultado es utilizado para cifrar los datos con AES 128 en modo CBC, siendo los 16 primeros bytes como clave y los segundos como vector de inicialización.
- Los datos en la tarjeta de memoria no son cifrados.
- Independientemente del cifrado de disco, desde la versión 4.0, Android permite cifrar certificados p12.

iOS

- Todos los datos son cifrados en el almacenamiento flash mediante AES 256.
- Existe un motor AES entre la memoria del sistema y la flash que se encarga de cifrar todos los datos.
- La clave de cifrado es única por dispositivo, se guarda dentro del motor de AES y no puede ser accedida por ningún otro interfaz del sistema.
- Además, iOS permite el cifrado de ficheros y credenciales de aplicación:
 - Cada fichero es cifrado con una clave única que se genera aleatoriamente.
 - Estas claves son cifradas con una clave derivada del *passcode*.
 - Cada clave se puede configurar con un nivel diferente de protección de tal forma que esté disponible en diferentes ocasiones (solo con el dispositivo desbloqueado, después del primer desbloqueo, etc.).
 - Las aplicaciones del sistema (Mail, Health, etc.) implementan este mecanismo por defecto.

Otras plataformas

- Blackberry:
 - El espacio de trabajo es cifrado por defecto, aunque no el personal.
 - El espacio personal y la tarjeta de memoria también pueden ser cifrados.
 - El cifrado se realiza utilizando AES 256.
 - Además, las aplicaciones del espacio de trabajo pueden añadir una capa adicional de cifrado para cuando el dispositivo está bloqueado.
 - Además, permite el cifrado de credenciales y claves de acceso.

- Windows Phone:
 - En la versión 8.1, solo los dispositivos gestionados por una organización tienen acceso al cifrado de disco.
 - A partir de Windows 10, todos los usuarios pueden cifrar los datos escritos a disco.
 - También permite el cifrado de credenciales y claves de acceso.

Borrado remoto

- Ante un robo, todos los sistemas operativos ofrecen un mecanismo de bloqueo y borrado remoto.
 - En la mayoría de los casos el proceso es instantáneo, pues requiere únicamente de cambiar la clave de cifrado del sistema.
- Además, todos los sistemas operativos cuentan con un mecanismo para el borrado remoto de aplicaciones desde los propios fabricantes.
 - Su finalidad es ser utilizado únicamente en casos de emergencia.
- En algunos casos esta funcionalidad ha sido utilizada para contrarrestar *malware*, como en caso de DroidDream:
 - Generalmente distribuido a través de *repackaging* de aplicaciones de pago.
 - Configurado para actuar solo mientras el usuario está durmiendo (de ahí su nombre).
 - Intenta utilizar dos *exploits* para ganar permisos de administrador, romper los mecanismos de protección de la *sandbox* y *rootear* el teléfono.

A close-up, slightly blurred photograph of a computer keyboard. The keys are dark grey or black. In the lower-left foreground, a blue Windows logo key is prominent. To its right, a key with a green plus sign is visible. The background shows more keys, some with green symbols, but they are out of focus. A semi-transparent blue rectangular box is overlaid across the middle of the image, containing the text 'Ejercicio de investigación' in white.

Ejercicio de investigación

◆◆◇ Ejercicio de Investigación

Introducción

- Durante este ejercicio deberás realizar una labor de investigación y volcar en el foro de la asignatura los resultados obtenidos para el debate con el resto de alumnos.
- Existen cuatro posibles ejercicios disponibles:
 - Estudio de Samsung Knox.
 - Privacidad y seguridad de las características del entorno móvil.
 - La problemática del almacenamiento extraíble.
 - Problemas de seguridad derivados del *jailbreak* o *rooting* de dispositivos.
- En las siguientes páginas podrás encontrar información más detallada de cada ejercicio.

Samsung Knox

- Samsung Knox es una solución de seguridad en Android para el entorno empresarial.
- Utilizado en un dispositivo Android, crea dos entornos completamente aislados para las aplicaciones. Cada contenedor tiene su propia pantalla de inicio y las aplicaciones de un contexto no pueden comunicarse con las del otro. La finalidad es que el empleado utilice un contenedor para las aplicaciones de la empresa y otro para las aplicaciones personales. El contenedor enfocado al trabajo cuenta con opciones de seguridad adicionales.
- Este entorno dual dificulta la fuga de datos y incrementa el control de la compañía sobre las aplicaciones relacionadas con el negocio. Sin embargo, introduce nuevas variables en las arquitecturas de las tecnologías móviles.
- Deberás analizar en profundidad las características de seguridad que añaden este tipo de arquitecturas y comparar Knox con otras propuestas emergentes que se mueven en la misma dirección (Android for work, etc.).

Estudio teórico

- A continuación se muestran un conjunto no exhaustivo de características típicas de las arquitecturas formadas por dispositivos móviles:
 - Comunicación inalámbrica.
 - Variedad de canales de comunicación.
 - Portabilidad.
 - Limitación en la capacidad de cómputo y consumo de energía.
 - Recolección de información del entorno por medio de sensores.
 - Ecosistema de apps.
- Deberás realizar un estudio riguroso para identificar las consecuencias, a nivel de seguridad y privacidad, que tienen al menos dos de las características anteriores. Las amenazas generadas por cada característica deben ser analizadas por separado, pero también en conjunto si su conjunción supone nuevas consecuencias a nivel de seguridad y privacidad.

Estudio de distintos sistemas operativos

- Algunos de los sistemas operativos móviles actuales permiten la utilización de tarjetas SD para la expansión de la capacidad de almacenamiento del dispositivo.
- Deberás averiguar que mecanismos de expansión de almacenamiento ofrece cada sistema operativo (iOS, Android, Windows Phone y Blackberry), tanto si es oficial como si no lo es.
- Además, por cada sistema operativo deberás detallar qué supone la utilización de estos mecanismos de expansión de espacio, describiendo, entre otras cosas:
 - Si permiten la instalación de aplicaciones en el almacenamiento extraíble.
 - Si aplican alguna medida de seguridad por defecto sobre estas aplicaciones.
 - Si existen mecanismos (oficiales o no) al alcance de los desarrolladores para poder almacenar de forma segura los datos en el almacenamiento extraíble.

Técnicas de *jailbreak* y *rooting*

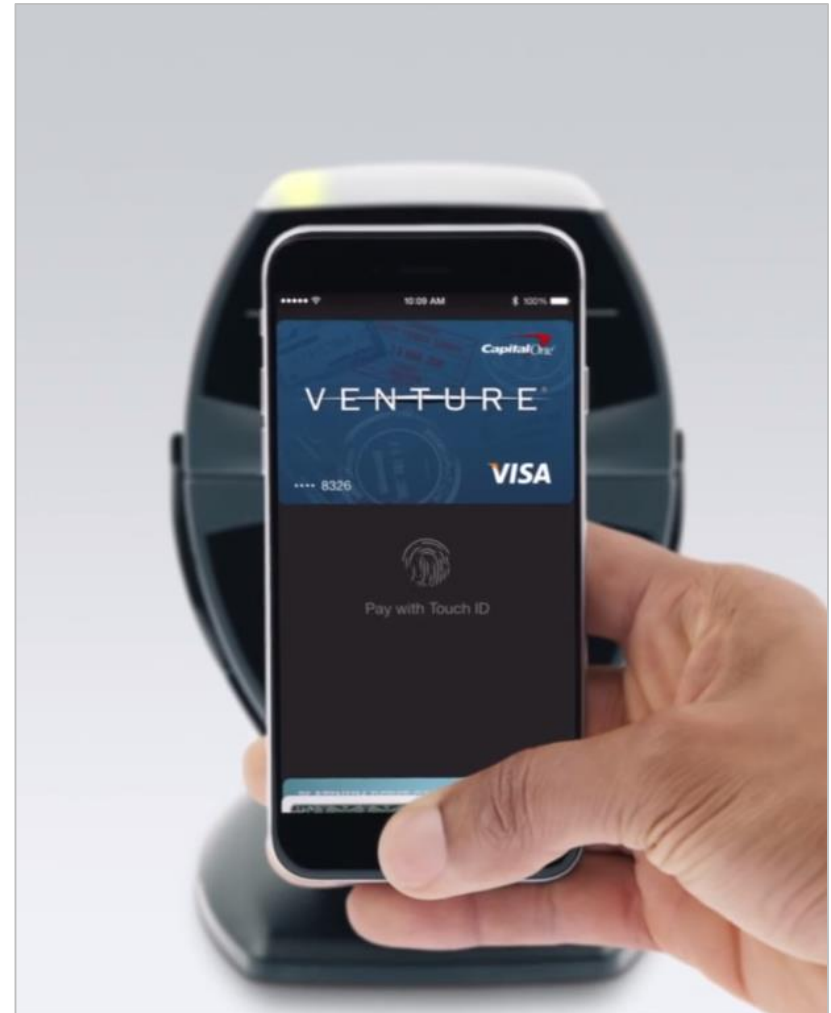
- Como se mencionó en la unidad anterior, el *jailbreak* (en dispositivos iOS) o *rooting* (en dispositivos Android) consiste en la eliminación de las restricciones incluidas en el sistemas operativos para evitar la ejecución de código sin firmar.
- Deberás investigar, para cada plataforma estudiada (iOS, Android, Windows Phone y Blackberry 10):
 - Versiones y dispositivos susceptibles de *jailbreak* o *rooting*.
 - Vulnerabilidad que se aprovecha en cada caso y requisitos para poder ejecutar el *jailbreak*.
 - ¿Podría ser utilizada la misma vulnerabilidad con fines maliciosos?
 - Consecuencias que tiene para la seguridad de las aplicaciones la ejecución en un entorno con *jailbreak* o *rooting*.
 - Mecanismos existentes para la detección de *jailbreak* o *rooting* por parte de aplicaciones.



Ejercicio práctico

Pago *contactless*

- Apple Pay es un sistema de pagos sin contacto implementado por Apple. Permite registrar una tarjeta de crédito en el dispositivo, para su posterior uso en terminales de pago sin contacto.
- Deberás realizar un informe en el que describas el sistema e incluyas la siguiente información:
 - Dispositivos compatibles.
 - Componentes del sistema.
 - Fases y funcionamiento general del sistema.
 - Mecanismos de seguridad utilizados.
 - Niveles y prestaciones de seguridad que ofrece al usuario.
 - Posibles amenazas en su uso.
 - Diferencias con respecto a Android Pay.



◆◆◆ Referencias

Páginas web útiles

- Para la realización del ejercicio puedes utilizar la siguiente documentación como referencia
- https://www.apple.com/business/docs/iOS_Security_Guide.pdf
- <https://developer.apple.com/apple-pay>
- <https://www.android.com/pay/>
- <https://developers.google.com/android>



A hand holding a pen is writing on a notebook. In the foreground, a black calculator and a textbook are visible. The textbook contains math problems, including arithmetic series and sigma notation. A blue semi-transparent banner is overlaid on the image, containing the text "Test de evaluación".

Test de evaluación

Gracias por su atención

