|  |
| --- |
| http://www.becas.sep.gob.mx/images/logo.png  TECNOLÓGICO NACIONAL DE MÉXICO  Instituto Tecnológico de Chihuahua II |
| Estructura de las Arquitecturas Móviles |
|  |
|  |
|  |
| 330/09/2016  Unidad 1  Ulises Manuel De la Rosa Contreras  13550362  Sist.Op.Dis.Moviles |

Contenido

[**Introducción.- 4**](#_Toc463034190)

[**Desarrollo.- 4**](#_Toc463034191)

[**Windows Phone 4**](#_Toc463034192)

[**Estructura. 4**](#_Toc463034193)

[**Núcleo. 5**](#_Toc463034194)

[**Procesos. 5**](#_Toc463034195)

[**Virtualización. 5**](#_Toc463034196)

[**Almacenamiento y sistemas de archivos. 6**](#_Toc463034197)

[**Lenguajes de programación soportados. 6**](#_Toc463034198)

[**IOS 6**](#_Toc463034199)

[**Estructura. 6**](#_Toc463034200)

[**Núcleo. 7**](#_Toc463034201)

[**Procesos. 7**](#_Toc463034202)

[**Virtualización. 8**](#_Toc463034203)

[**Almacenamiento y sistemas de archivos. 8**](#_Toc463034204)

[**Lenguajes de programación soportados. 8**](#_Toc463034205)

[**Web OS 8**](#_Toc463034206)

[**Estructura. 8**](#_Toc463034207)

[**Núcleo. 9**](#_Toc463034208)

[**Procesos. 9**](#_Toc463034209)

[**Virtualización. 10**](#_Toc463034210)

[**Almacenamiento y sistemas de archivos. 10**](#_Toc463034211)

[**Lenguajes de programación soportados. 10**](#_Toc463034212)

[**Firefox OS 10**](#_Toc463034213)

[**Estructura. 10**](#_Toc463034214)

[**Núcleo. 12**](#_Toc463034215)

[**Procesos. 12**](#_Toc463034216)

[**Virtualización. 13**](#_Toc463034217)

[**Almacenamiento y sistemas de archivos. 13**](#_Toc463034218)

[**Lenguajes de programación soportados. 13**](#_Toc463034219)

[**MeeGo 14**](#_Toc463034220)

[**Estructura. 14**](#_Toc463034221)

[**Núcleo. 15**](#_Toc463034222)

[**Procesos. 15**](#_Toc463034223)

[**Virtualización. 15**](#_Toc463034224)

[**Almacenamiento y sistemas de archivos. 15**](#_Toc463034225)

[**Lenguajes de programación soportados. 15**](#_Toc463034226)

[**Android 16**](#_Toc463034227)

[**Estructura. 16**](#_Toc463034228)

[**Núcleo. 17**](#_Toc463034229)

[**Procesos. 17**](#_Toc463034230)

[**Virtualización. 18**](#_Toc463034231)

[**Almacenamiento y sistemas de archivos. 18**](#_Toc463034232)

[**Lenguajes de programación soportados. 19**](#_Toc463034233)

[**Conclusión.- 19**](#_Toc463034234)

# Introducción.-

Actualmente hay una gran cantidad de sistemas operativos móviles, muchos de ellos ya desaparecieron pero aun así lograron aportar algo diferente y gracias a ellos se pudieron desarrollar tecnologías que se utilizan actualmente.

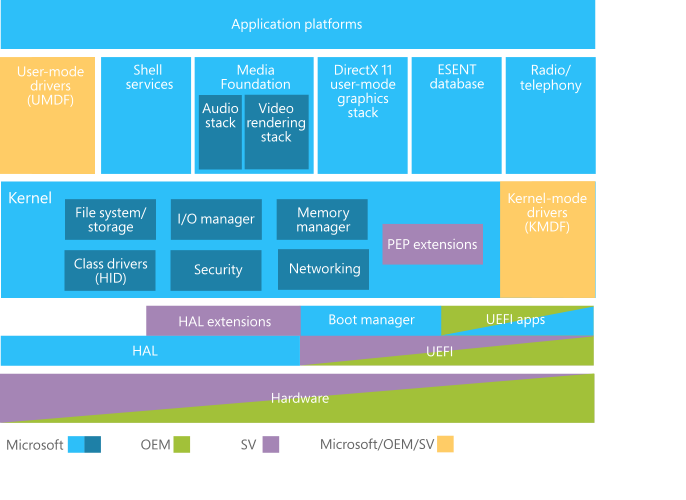
A continuación exploraremos un poco algunos de los sistemas operativos móviles más importantes, sus características, estructura , la manera de desarrollar para ellos, entre otras cosas.

# Desarrollo.-

## Windows Phone

**Estructura.**

El sistema operativo móvil Windows phone de Microsoft, cuenta con una estructura como se muestra en la imagen.



Primero tenemos la capa de aplicación, en la cual es donde el usuario interactúa directamente con el sistema mediante la interfaz gráfica, la cual actualmente es muy similar a la del sistema Windows 10 que utilizan las computadoras. El middleware permite la comunicación entre aplicaciones, es donde se encuentran los controladores y es la capa que proporciona los servicios que requiere el sistema.

El kernel de Windows phone, está basado en el kernel de Windows que utiliza Microsoft para su sistema operativo Windows actual. El kernel cuenta con seguridad, manejo de entrada-salida, administrador de memoria y sistema de almacenamiento entre otros.

**Núcleo**.

El núcleo es el Windows NT, el cual comparte con el sistema Windows 10 y por lo tanto funcionan de forma muy similar, haciendo que el sistema Windows 10 sea multiplataforma y que tenga sincronización una plataforma con la otra. Windows NT cuenta con seguridad, administrador de memoria y sistema de almacenamiento entre otras cosas.

**Procesos.**

Al igual que en Windows 10, Windows phone también cuenta con procesos innecesarios que se pueden desactivar. Para desactivar un proceso en Windows phone se debe desplazar la pantalla hacia la izquierda o pulsar la flecha que nos aparece en la parte final de la pantalla de inicio, nos desplazamos hacia abajo entre las aplicaciones y pulsamos en configuración, dentro de Sistema nos desplazamos hasta Ahorro de batería y pulsamos. Se nos muestran dos pantallas, la de Configuración que nos muestra información sobre la duración restante (en porcentaje y en tiempo estimado) y el tiempo desde la última carga, pasando a la pantalla de USO veremos un listado con las aplicaciones usadas recientemente y una información en cada una de ellas del estado respecto a su ejecución en segundo plano, y desde aquí podemos cambiar el estado del proceso.

**Virtualización.**

Hyper-V es un Virtual Box, un VmWare Cliente o un Windows XP Emulator, pero con potentes ventajas del sistema Hyper-V. Entre ellas está el crear una SAN Virtual, es decir una cabina de discos duros que se comportan como uno solo y que da servicio de almacenamiento a todas la máquinas virtuales; también la facilidad de creación, modificación y eliminación de las máquinas virtuales; y el poder crear múltiples snapshot o fotografías del estado actual para, llegado el caso, volver a un momento temporal anterior.

Otra ventaja de tener un cliente Hyper-V es que me permite importar, exportar y mover máquinas virtuales a otros clientes o a un servidor con facilidad.

**Almacenamiento y sistemas de archivos.**

Debido a que Windows phone 10 esta sincronizado con Windows 10, el sistema de archivos que ambos sistemas es similar.

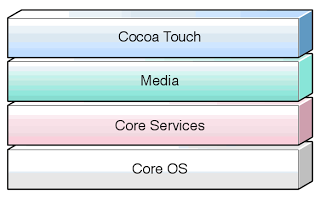
**Lenguajes de programación soportados.**

Windows phone soporta para su desarrollo los lenguajes C#,C++, vb.Net y JavaScript, principalmente los del framework .NET.

## IOS

**Estructura**.

iOS es el sistema operativo de Apple para dispositivos móviles que permite utilizar HTML5. La arquitectura iOS está basada en capas, donde las capas más altas contienen los servicios y tecnologías más importantes para el desarrollo de aplicaciones, y las capas más bajas controlan los servicios básicos.



* Cocoa Touch

Cocoa Touch es la capa más importante para el desarrollo de aplicaciones iOS. Posee un conjunto de Frameworks que proporciona el API de Cocoa para desarrollar aplicaciones. Esta capa está formada por dos Frameworks fundamentales:

* UIKit: contiene todas las clases que se necesitan para el desarrollo de una interfaz de usuario.
* Foundation Framework: define las clases básicas, acceso y manejo de objetos, servicios del sistema operativo.
* Media

Provee los servicios de gráficos y multimedia a la capa superior.

* Core Services

Contiene los servicios fundamentales del sistema que usan todas las aplicaciones.

* Core OS

Contiene las características de bajo nivel: ficheros del sistema, manejo de memoria, seguridad, drivers del dispositivo.

**Núcleo.**

El nivel del sistema abarca el entorno del núcleo, los controladores y las interfaces de bajo nivel del sistema operativo UNIX. El Kernel está basado en “Mach” y es responsable de todos los aspectos del sistema operativo.

Se encarga de tareas básicas de bajo nivel como:

·Gestión de memoria virtual

·Gestión procesos, hilos, el manejo de la memoria y comunicación entre procesos.

·Gestión del sistema de archivos.

·Gestión del acceso a red de bajo nivel.

·Gestión de procesos que interactúan directo con el hardware.

**Procesos.**

El sistema que utiliza el sistema operativo iOS es a través de la cola de prioridades, los niveles se en las que se dividen con base a sus características:

* Normales: Migrar otras prioridades.
* Alta prioridad: Donde se pone la prioridad de los comportamientos y los otros procesos del sistema.
* Modo Kernel.

Los objetivos de la planificación de proceso son:

Eficacia: mantener ocupada la CPU un 100% del tiempo.

Tiempo de regreso: minimizar el tiempo que deben esperar los usuarios por lotes para obtener sus resultados.

Rendimiento: maximizar el número de tareas procesadas por hora.

**Virtualización.**

La única forma de virtualizar un sistema iOS es mediante la virtualización de Mac OS en algún sistema con Windows.

**Almacenamiento y sistemas de archivos.**

El sistema de archivos de Apple o APFS, se enfoca en la seguridad de la información. El sistema jerárquico HFS+, el cual ha sido utilizado por Apple durante los últimos 18 años, será jubilado por APFS. El nuevo sistema operativo es capaz de ofrecer un método de cifrado unificado para todos los dispositivos de Apple, incluido el cifrado multi-circuito que hará que introducirse en estos dispositivos sea algo bastante más complejo. E cifradol es uno de los fundamentos básicos de los sistemas operativos con APFS, y no una característica que se agregue a ellos.

**Lenguajes de programación soportados.**

Actualmente el lenguaje más utilizado para programar en iOS es Swift el lenguaje desarrollado por Apple exclusivo para iOS y MAC.

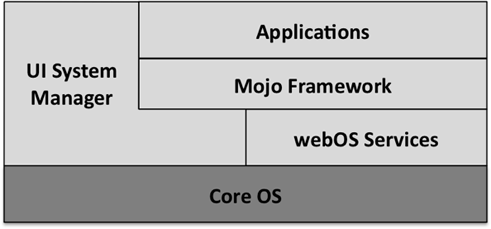
## Web OS

**Estructura.**

Web OS está basado en el kernel 2.6 de Linux, con una combinación de código abierto y componentes de servicios del plano de usuario de Palm, denominado Core OS.

No se tiene ninguna interacción directa con el sistema operativo básico, ni tampoco los usuarios finales. El acceso se proporciona a través de Mojo y de distintos servicios. Los usuarios interactúan con las distintas aplicaciones y el sistema de gestión de la interfaz de usuario. Esto se conoce como el "Entorno de Aplicación".

La siguiente figura muestra una visión simplificada de la arquitectura web OS. Este esquema general se incluye para poder hacerse a la idea de cómo funciona web Os.



**Núcleo.**

Core Os se basa en una versión del kernel 2.6 de Linux con la arquitectura estándar de driver administrado por udev, con un gestor de arranque propio. Soporta un sistema de archivos ext3 para particiones internas y particiones FAT32 para los archivos multimedia, que puede ser montado externamente a través de USB para transferir archivos multimedia desde y hacia el dispositivo.

**Procesos.**

Aquí  los  procesos tienden a ejecutarse en el servidor de redes. Con  este  avance,   se  tiene  que  parte  de  la  lógica  del sistema  operativo  no  se hace de forma   nativa sino a través de la conexión HTTP,  donde  se  pueden ceder  derechos  al  servidor de   forma  asíncrona para que este haga también trabajo  de procesamiento. Esto  para  que  el  usuario  tenga  una  experiencia  más enriquecedora.

En este caso, los hilos de ejecución a su vez, se corren de forma nativa como de forma exterior dando pie a que la capacidad de procesamiento se pueda dar ya sea en el mismo dispositivo o  bien, en un agente externo. Esta paridad se logra a través de Mojo, aplicación encargada de preveer los cambios de contexto y aminorar tiempos de ejecución.

**Virtualización.**

Aunque el sistema operativo de HP WebOS haya muerto con el fracasado touchpad, todavía es posible echarle un vistazo y jugar con él desde OS X (u otra plataforma) mediante un emulador como Virtual Box.

Basta con bajarse la SDK que Palm pone a disposición de los desarrolladores y bueno, cómo no, un software de virtualización como Virtual Box para OS X o Windows.

**Almacenamiento y sistemas de archivos.**

Web OS proponía el método de almacenamiento de información desde la nube y de esta forma se podía acceder a la información desde otros sistemas como Windows u OS.X.

**Lenguajes de programación soportados.**

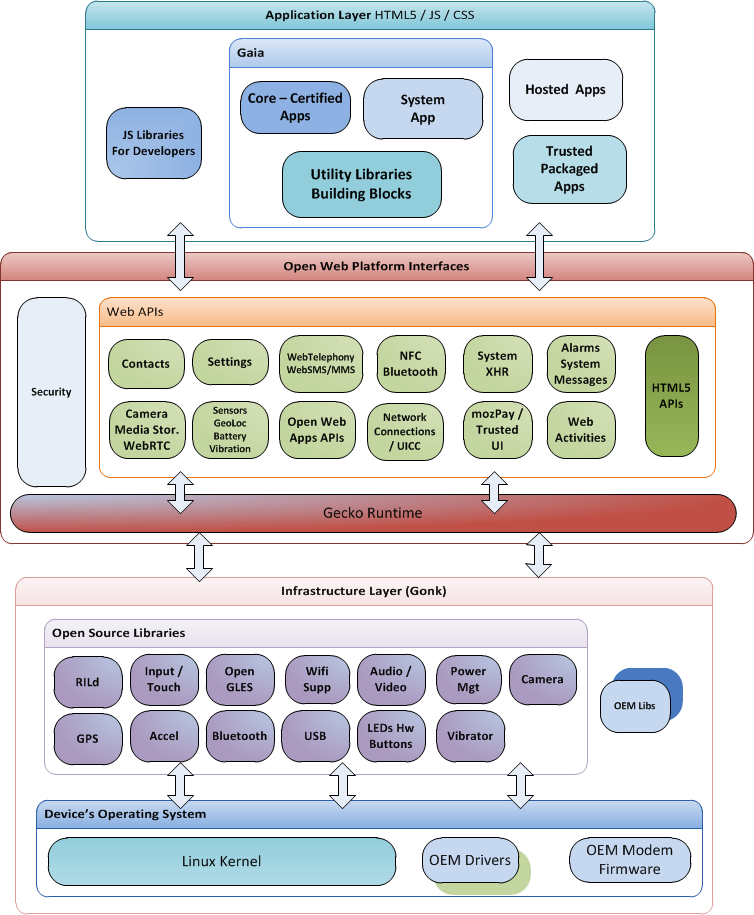
Palm Web Os utiliza estándares como HTML, CSS, JavaScript como base principal. Su entorno de desarrollo AJAX de Mojo está basado en Dojo. También utiliza WebKit como su navegador predeterminado, con sqlite como almacenamiento local, parte de los estándares de HTML5.

Utiliza JavaScript como intermediario entre las funciones del dispositivo, como el GPS, el acelerómetro, etc.

## Firefox OS

**Estructura.**

La arquitectura de Firefox OS es por 3 capas (Gaia, Gecko y Gonk) tal y como se muestra en la imagen.



Gaia

Es la capa de mayor nivel y la interfaz de usuario de Firefox OS. Cualquier cosa que se dibuja en la pantalla una vez que Firefox OS ha arrancado es responsabilidad de Gaia. Esto incluye la pantalla de bloqueo, la pantalla principal y todas las aplicaciones estándar preinstaladas que suelen incorporar los teléfonos inteligentes. El funcionamiento de Gaia se basa exclusivamente en HTML, CSS y JavaScript. La comunicación con las capas inferiores del sistema operativo se realiza a través de Web APIs implementadas en la capa Gecko. La capa Gaia es también donde se instalan las aplicaciones de terceros.

Gecko

Esta es la capa de ejecución de aplicaciones de Firefox OS. Ofrece el soporte para la utilización de los tres estándares abiertos en los que se basa: HTML, CSS y JavaScript. Además incluye las Web APIs que hacen de interfaz entre las aplicaciones de la capa superior (Gaia) y del sistema operativo y hardware de la capa inferior (Gonk).

Gonk

Consiste en un núcleo Linux más una capa de abstracción del hardware (HAL). Todas las peculiaridades de hardware de un dispositivo en particular se abordan en esta capa a través de HAL. De esta forma, por ejemplo, si Gecko tiene que obtener información de geo localización, la solicita a Gonk sin preocuparse del hardware disponible para obtener la información. Es Gonk quien deberá disponer de los drivers necesarios del fabricante del GPS del dispositivo. Se puede decir que Gonk es un distribución muy sencilla de Linux con su kernel, bibliotecas de software, drivers y demás componentes.

**Núcleo.**

El núcleo consiste de un kernel Linux basado sobre el Android Open Source Project (AOSP) y una capa de abstracción de hardware (HAL). Gran parte del código aquí presente proviene de Android, encontrándose también los drivers y componentes esenciales de Linux para controlar el hardware. El proceso b2g en esta capa hace de puente con Gonk.

**Procesos.**

En Firefox OS el encargado de planificar los procesos del sistema es el proceso Init que hace parte del Gonk (kernel), este se encarga de cargar los procesos fundamentales del sistema, luego que da activo como gestor de procesos, este es el mismo Init de Linux (solo que modificado para cargar los respectivos procesos de este sistema), por lo tanto la planificación de los procesos es la misma en ambos sistemas operativos.

El algoritmo de planificación que usa Linux desde su versión 2.6.23 se llama Planificador Completamente Justo, en genera se puede decir que es un modelo de CPU multi-tarea ideal, este algoritmo no tiene una lista de procesos encolados sino que forma un árbol de procesos y escoge el mejor camino a seguir según el vruntime (mide el tiempo del proceso normalizado por la cantidad de tareas).

**Virtualización.**

El complemento Simulador de Firefox OS es una herramienta que le permite probar y depurar aplicaciones para Firefox OS en el escritorio.

En esencia, el complemento del Simulador consiste de:

* El Simulador en sí: éste incluye el cliente de escritorio de Firefox OS, que es una versión de las capas superiores de Firefox OS que se ejecuta en su sistema operativo de escritorio. El Simulator también incluye algunas funciones de emulación adicionales que no están disponibles en las compilaciones de Firefox OS para escritorio estándares.
* El tablero: una herramienta ubicada en el navegador Firefox que permite iniciar y detener el Simulador; e instalar, desinstalar y depurar aplicaciones ejecutadas en él. El tablero también le permite instalar aplicaciones en un dispositivo real, y realiza comprobaciones a los archivos manifest para detectar problemas comunes.

**Almacenamiento y sistemas de archivos.**

Todas las operaciones de directorios y archivos pasan por una capa del kernel llamada "virtual file system" cada sistema de archivos es una implementación de VFS. Cada sistema de archivo es un módulo separado dentro del kernel que contiene las operaciones que son soportadas por este y por VFS.

Los sistemas de archivos soportados varían según dispositivo y según la versión del sistema pero se mantienen comunes los sistemas comunes para memorias flash (el estándar usado para almacenamiento en cuestiones móviles).

* exFAT - la extended File Allocation Table es un sistema de archivo de microsoft.No es muy usado debido a requerimientos de licencia.
* JFFS2 - El Journal Flash File System version 2 es el sistema de archivos por defecto en AOSP (Android Open Source Project).

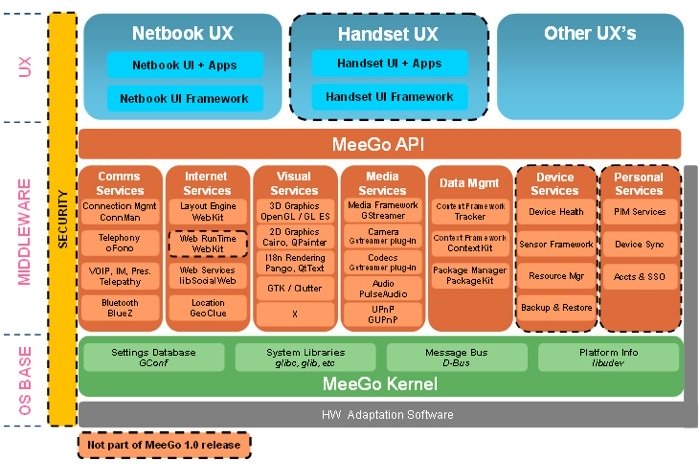
**Lenguajes de programación soportados.**

El proyecto Firefox OS como no podría ser de otra manera, es libre, basado en el kernel de Linux, y funciona a nivel de frontend con lenguajes web (HTML5, CSS3, JavaScript).

## MeeGo

**Estructura.**

El sistema operativo MeeGo, lanzado en 2010 por parte de Intel y Nokia, estaba estructurado de la forma en la que se muestra en la imagen



Tal y como se muestra, el sistema estaba construido por 3 capas: UX, Middleware y OS Base. Estas capas tenían implementado un buen nivel seguridad.

El UX o la experiencia de usuario, representa el nivel de satisfacción del usuario al interactuar con el sistema. Esta cubre el diseño, la interfaz gráfica, las funciones y el tipo de aplicaciones con las que cuenta el sistema.

El middleware es la capa que permite la intercomunicación del sistema con las aplicaciones y las aplicaciones con otras aplicaciones que tengan servicios en común. Esta capa abarca los servicios de internet, los servicios comunes, los servicios como la cámara, el audio y los codecs que utilizan.

Finalmente el OS Base representa lo más básico del sistema, la forma en la que está compuesto y principalmente el núcleo, que en el caso de MeeGo, era un kernel de Linux.

**Núcleo.**

El núcleo de MeeGo era un Kernel de Linux, el cual tenía implementada una función que hacía que el sistema se actualizara automáticamente cada vez que saliera una nueva versión.

**Procesos.**

No hay suficiente información.

**Virtualización.**

MeeGo no tuvo el tiempo de vida suficiente para llegar a virtualizarse.

**Almacenamiento y sistemas de archivos.**

VoIP.

Mensajería instantánea.

Correo electrónico y calendario.

Integración con redes sociales como Facebook y Twitter.

Servicios de localización.

Sincronización de datos en la nube o cloud computing.

Aplicaciones de medios.

Soporte de gestos.

**Lenguajes de programación soportados.**

MeeGo utiliza el framework Qt y este permite varios lenguajes como:

C++

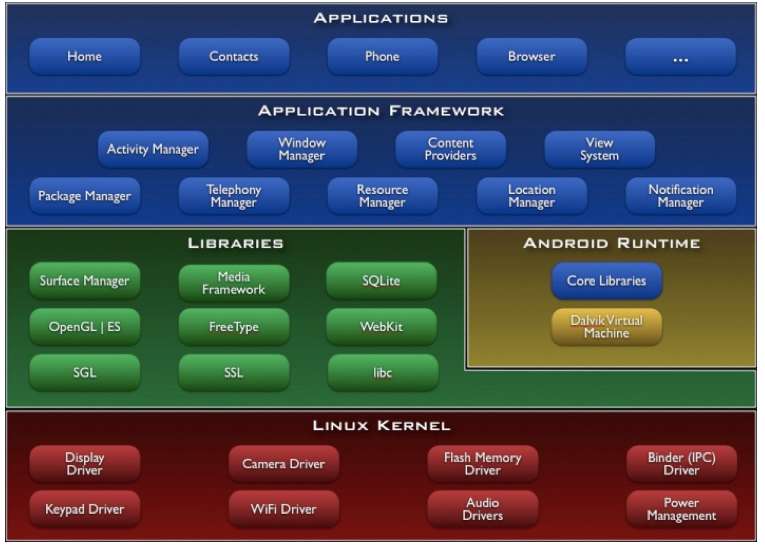
Python

C#

## Android

**Estructura.**

La estructura de Android también se divide en capas como se muestra en la siguiente imagen.



El núcleo proporciona servicios como la seguridad, el manejo de la memoria, el multiproceso, la pila de protocolos y el soporte de drivers para dispositivo.

La capa Runtime está basada en el concepto de máquina virtual utilizada en Java. Debido a las limitaciones de los dispositivos donde se ejecuta Android no fue posible utilizar la máquina virtual de Java estándar. Google tomó la decisión de crear una nueva, la máquina virtual Dalvik.

La siguiente capa son las librerías nativas en C/C++ usadas en varios componentes de Android. Están compiladas en el código nativo del procesador. Muchas de las librerías utilizan proyectos de código abierto:

* WebKit: Soporta un moderno navegador web utilizado en el navegador Android y en la vista webview. Se trata de la misma librería que utiliza Google Chrome y Safari de Apple.
* System C Library: Una derivación de las librerías BSD de C estándar, adaptados para dispositivos embebidos basaos en Linux.
* Media Framework: Librería basada en PacketVideo's OpenCORE; soporta codecs de reproducción y grabación de multitud de formatos de audio, video e imagenes MPEG4, H.264, MP3, AAC, AMR, JPG y PNG.
* Surface Manger: Maneja el acceso al subsistema de representación gráfica en 2D y 3D.
* SGL: Motor de gráficos 2D.
* SQLite: Potente y ligero motor de bases de datos relacionales disponible para todas las aplicaciones.
* SSL: Proporciona servicios de encriptación Secure Socket Layer.

El entorno de aplicación proporciona una plataforma de desarrollo libre para aplicaciones con gran riqueza e innovaciones..

Una de las mayores fortalezas del entorno de aplicación de Android es que se aprovecha el lenguaje de programación JAVA. El SDK de Android no acaba de ofrecer todo lo disponible para su estándar del entorno de ejecución JAVA (JRE), pero es compatible con una fracción muy significativa del sistema.

El nivel de Aplicaciones está formado por el conjunto de aplicaciones instaladas en una máquina virtual Android. Todas las aplicaciones han de ser ejecutadas en la máquina virtual de Dalvik para garantizar la seguridad del sistema.

**Núcleo.**

El núcleo de Android está formado por el sistema operativo Linux, versión 2.6. Esta capa proporciona servicios como la seguridad, el manejo de la memoria, el multiproceso, la pila de protocolos y el soporte de drivers para dispositivos. Esta capa de del modelo actúa como capa de abstracción entre el hardware y el resto de la pila, por lo tanto, es la única que es dependiente del hardware.

**Procesos.**

Android ha sido diseñado para sistemas con poca memoria y un procesador que no es tan rápido como los procesadores de escritorio. Manteniendo las limitaciones en mente, la visión de Google para Android es que tiene un robusto conjunto de APIs de programación y una interfaz de usuario muy sensible. A fin de facilitar esta visión, se creó una capa de abstracción que permite a los desarrolladores de aplicaciones a ser agnóstico hardware en su diseño.

**Virtualización.**

Uno de los elementos clave de Android es la máquina virtual de Dalvik. En lugar de utilizar una tradicional máquina virtual Java (VM), tales como Java ME (Java Mobile Edition), Android utiliza su propia máquina virtual personalizado diseñado para asegurar que la multitarea se ejecutan de manera eficiente en un único dispositivo.

La máquina virtual Dalvik utiliza el dispositivo del kernel de Linux subyacente para manejar bajo nivel de funcionalidad, incluyendo la seguridad, la planificación de procesos, y la gestión de la memoria.

Todo el hardware de Android y acceso a los servicios del sistema se gestiona mediante Dalvik como un nivel intermedio. La máquina virtual Dalvik lanza procesos ejecutables Dalvik, un formato optimizado para asegurar la mínima huella en la memoria . Los ejecutables. Dex se crean mediante la transformación de las clases de Java lenguaje, compilado utilizando las herramientas proporcionadas en el SDK.

**Almacenamiento y sistemas de archivos.**

Cada dispositivo compatible con Android soporta dos tipos de memoria, por un lado tenemos "memoria interna" en esta memoria está instalada la ROM o versión de Android que corre nuestro dispositivo, y suele quedar parte libre para instalar las distintas aplicaciones además esta es la memoria que se borra cuando restablecemos los valores de fábrica, y por otro lado tenemos una "memoria externa" que se puede utilizar para guardar archivos y también para mover esas aplicaciones que ocupan más.

La memoria externa, puede ser un medio de almacenamiento extraíble (como una tarjeta SD) o una memoria interna (no extraíble). Los archivos guardados en el almacenamiento externo son de lectura global y pueden ser modificados por el usuario cuando permiten el almacenamiento masivo USB para transferir archivos de un ordenador.

Es posible que el fabricante del dispositivo particione la memoria interna con una partición de almacenamiento externo o "almacenamiento USB" es decir que cuando compremos un teléfono que nos decían que tenia 8Gb de memoria interna, veamos 2 particiones, una por ejemplo de 1,5Gb que se llama "memoria interna" y otra de 5,5Gb a la que llaman "memoria USB o memoria externa".

* EXT4 en Android

Cuarto sistema de archivos extendido, o en inglés, "fourth extended filesystem", es un sistema de archivos transaccional que fue creado como mejora a EXT3.Este sistema de archivos es capaz de trabajar con tamaños mucho mayores, ya que puede mover archivos de hasta 16 TB. También existe la posibilidad de crear hasta 64.000 subdirectorios, el doble que con EXT3.

La desfragmentación también es algo que este sistema de archivos cuida, y es que ya es posible desfragmentar archivos individualmente. Ya no es necesario desmontar el disco para proceder al desfragmentado del sistema de archivos entero. Este sistema de archivos es el que la gran mayoría de terminales Android utiliza por defecto.

* F2FS en Android

Las siglas significan en inglés "Flash-Friendly File System". Se trata del sistema de archivos creado por Kim Jaegeuk en Samsung para el núcleo Linux (en lo que Android se basa). Fue creado de forma específica por y para que tuviera muy en cuenta las características de los dispositivos de almacenamiento Flash, es decir, la forma de memoria que tienen los smartphones en su grandísima mayoría. Nuestros teléfonos traen una memoria de estado sólido, o SSD, junto a tarjetas SD, en ambos casos con tecnología flash.

**Lenguajes de programación soportados.**

El principal lenguaje que se utiliza para desarrollar aplicaciones es Android es Java. En programación web soporta CSS y JavaScript.

# Conclusión.-

En conclusión ha habido muchos sistemas operativos móviles muy interesantes y que gracias a ellos nuestras vidas se han facilitado bastante, ya que la comunicación móvil vino a revolucionar completamente nuestro estilo de vida.

Actualmente los sistemas móviles más importantes son Android, iOS y en menor medida Windows phone, algunos como Firefox OS prometían cosas bastante interesantes, pero debido a ciertas circunstancias no tuvieron el éxito esperado.