



# Introducción a Android

---

LIN - Curso 2015-2016



# Contenido

---



## **1** Introducción

## **2** Arquitectura y Componentes

## **3** Android vs. GNU/Linux



# Contenido

---



## 1 Introducción

## 2 Arquitectura y Componentes

## 3 Android vs. GNU/Linux



# Android: Características

## Todo lo que se puede esperar de un SO actual para móviles

- Ecosistema de aplicaciones
  - permite al usuario añadir y eliminar nuevas aplicaciones
  - exponer/publicar nuevas características por todo el sistema
- Soporte completo para la gran mayoría de tecnologías web
  - Incluye navegador web basado en el motor de renderización Web-kit
- Soporte para múltiples tecnologías de comunicación inalámbrica
  - GSM, CDMA, UMTS, LTE, Bluetooth, WiFi, NFC
- Soporte para aceleración de gráficos por HW vía OpenGL ES



# Android: Historia (I)

---

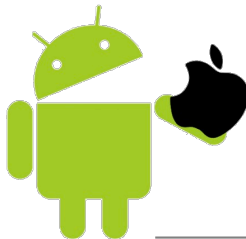
- El desarrollo de Android comenzó en 2003 en una *start-up* (Android Inc.)
  - Sede en Palo Alto, CA
  - La compañía se centró en el desarrollo software para móviles desde sus inicios
  - Los fundadores ya tenían un conocimiento profundo del mercado de las tecnologías móviles
    - Andy Rubin
    - Danger Inc. (Sidekick Phone)
- Google adquirió Android Inc. en 2005



# Android: Historia (II)

---

- En 2007 Google anunció la creación del **OHA** (*Open Handset Alliance*)
  - **Consortio de empresas** del mercado de la tecnología móvil para potenciar el desarrollo de Android
  - *Vendedores y fabricantes de HW*: Intel, Texas Instruments, Qualcomm, Nvidia, Motorola, HTC, Sony Ericsson, Samsung, etc.
  - *Empresas de Software*: Google, eBay, Andago, etc.
  - *Operadores móviles*: T-Mobile, Telefónica, Vodafone, etc
- En 2008 Google publica la primera versión *open source* de Android





# Android Open Source Project (AOSP)

---

- Al finalizar el desarrollo de cada versión de Android, Google libera su código fuente (AOSP)
  - El código del AOSP está disponible en <https://source.android.com>
  - Permite a la comunidad y a los distintos fabricantes adaptar dicha versión a sus productos/necesidades
- Cualquiera puede contribuir al AOSP con sugerencias y *bugfixes* pero...
  - En realidad, proceso de desarrollo cerrado que controla Google
  - Este modelo promueve la existencia de *forks* del AOSP
    - CyanogenMod
    - Android-x86
    - RowBoat
    - Replicant





# Android Open Source Project (AOSP)

---

- Típicamente el código del AOSP requiere modificaciones para poder funcionar en la mayor parte de dispositivos móviles
  - Desarrollo del AOSP se realiza habitualmente sobre los dispositivos móviles de Google (Nexus)
  - Google suele asociarse con un fabricante particular cada 6 meses para trabajar conjuntamente en una nueva versión de Android







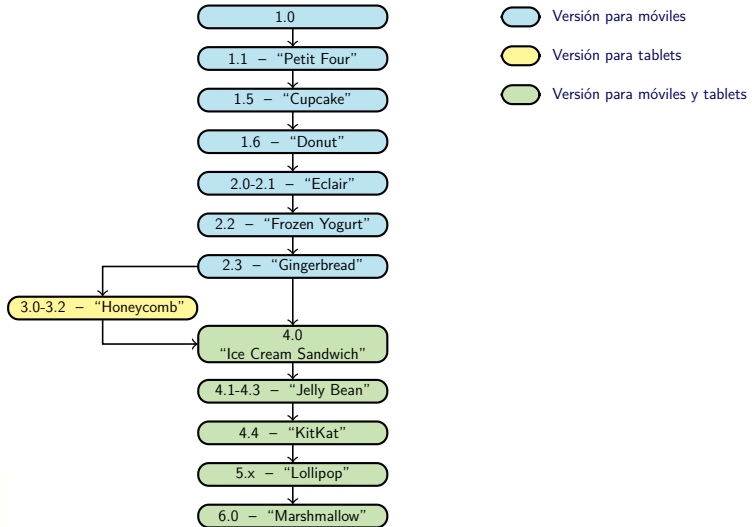
# Versiones de Android (AOSP)

---

- Se usan nombres de *postres/dulces* para cada versión
- Los nombres de versión se asignan en orden alfabético
- Últimas versiones:
  - Android 2.3 “Gingerbread”
  - Android 3.X “Honeycomb”
  - Android 4.0 “Ice Cream Sandwich”
  - Android 4.1/4.2/4.3 “Jelly Bean”
  - Android 4.4 “KitKat”
  - Android 5.0-5.1.X “Lollipop”
  - Android 6.0 “Marshmallow”

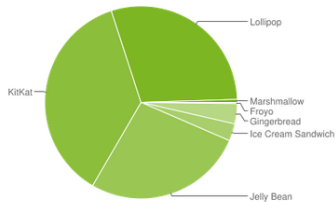


# Versiones de Android (AOSP)



# Uso de las distintas versiones

Version	Codename	API	Distribution
2.2	Froyo	8	0.2%
2.3.3 - 2.3.7	Gingerbread	10	3.4%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	2.9%
4.1.x	Jelly Bean	16	10.0%
4.2.x		17	13.0%
4.3		18	3.9%
4.4	KitKat	19	36.6%
5.0	Lollipop	21	16.3%
5.1		22	13.2%
6.0	Marshmallow	23	0.5%



Data collected during a 7-day period ending on December 7, 2015.

Any versions with less than 0.1% distribution are not shown.

# Contenido

---



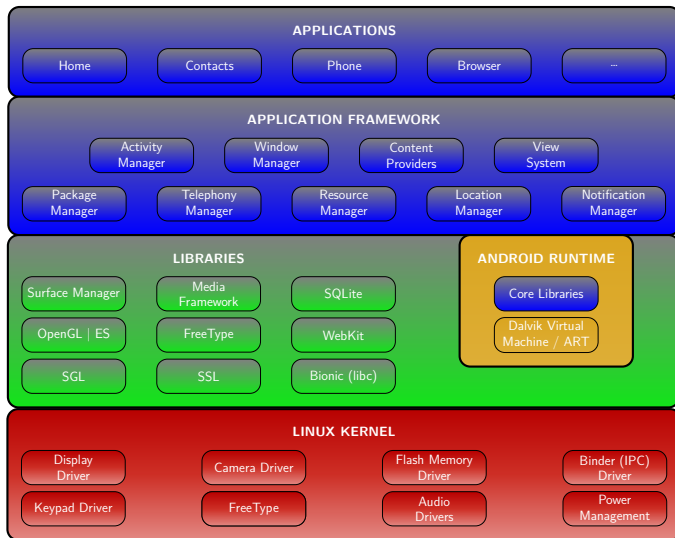
**1** Introducción

**2** Arquitectura y Componentes

**3** Android vs. GNU/Linux



# Arquitectura de Android





# kernel Linux en Android

---

- El kernel Linux constituye la base fundamental del SO Android
  - Clon de UNIX que implementa gran parte del estándar POSIX
- No es la versión *vanilla* de Linux, sino que incluye elementos extra
  - Binder - Comunicación entre procesos
  - Extensiones para gestión de energía en móviles/tablets
  - Drivers específicos
  - ...
- En la actualidad la versión “Androidizada” del kernel constituye un *fork* sustancial de Linux
  - Algunas extensiones se están incorporando en el directorio *staging* de las fuentes de Linux
  - Linaro mantiene una versión “androidizada” del kernel cercana a la versión actual de Linux (*mainline*)





# Librerías de Android

---

- Componentes del SO que permiten interactuar a bajo nivel con el kernel
- En el conjunto de librerías hay algunas desarrolladas por Google y muchas otras (*open source*) desarrolladas por terceros
- **Ejemplos**
  - **Bionic**: Librería de "C" de Android
    - Desarrollada por Google
    - Licencia BSD
  - **WebKit** (Licencia BSD)
  - **SQLite** (Dominio Público)
  - **OpenSSL** (Licencia derivada de BSD)





# Android Runtime

---

- Gestiona la ejecución de aplicaciones Android (escritas en Java)
  - Desarrollado prácticamente en su totalidad por Google
- **Varios componentes:**
  - **Dalvik/ART:** Máquina virtual de Java en Android
    - Reemplaza a la versión de Sun/Oracle
  - **Core Library:** Implementación del API de Java
    - Reemplaza a la versión de Sun/Oracle
    - Basado en el proyecto Apache Harmony
  - **Comandos básicos**
    - Accesibles mediante ADB shell
  - **Demonios/Servicios** nativos del sistema
  - **Init**







# Android Framework

---

- Implementa el API de desarrollo de aplicaciones en Android
  - Desarrollado principalmente en Java aunque hace uso de JNI para acceso a más bajo nivel
- Expone todos los servicios/subsistemas proporcionando las abstracciones necesarias
- Servicios/Componentes:
  - **GUIs en Android:** Activity Manager, Window Manager, View System
  - **Acceso a ficheros y bases de datos SQLite:** Content Providers
  - **Administración de Aplicaciones:** Package Manager
  - ...





# Aplicaciones Android

---

- El AOSP incluye algunas aplicaciones básicas:
  - Phone
  - Browser ( $\neq$  Chrome)
  - Contacts
  - Email ( $\neq$  Gmail)
- Las aplicaciones de Google (Google Play incluida) son propietarias
  - Código fuente NO incluido en el AOSP
  - Para que un fabricante de móviles/tablets pueda obtener y distribuir las aplicaciones de Google su producto ha de cumplir ciertas especificaciones y pasar tests
    - *Android Compatibility Definition Document (CDD)*
    - *Compatibility Test Suite (CTS)*

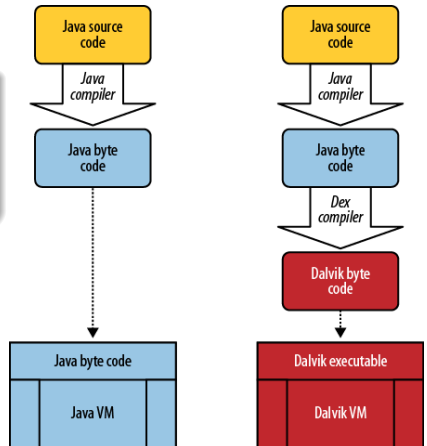


# Java en Android

- Aplicaciones Android están escritas en Java pero **usan Dalvik VM (o ART)** en lugar de Java VM

## Dalvik/ART vs. Java VM

- Basada en registros/Basada en pila
- Ejecución de ficheros .dex vs. .class
- Dalvik/ART poseen optimizaciones para móviles





# ¿Está Google reinventando la rueda?

---

- Google evita utilizar en Android muchos componentes SW ampliamente usados
  - GNU libc (→ Bionic)
  - Java VM (→ Dalvik/ART VM)
  - Librerías de Sun SDK (→ Core Libraries de Apache Harmony)
  - GNU Bash (→ *port* de MirBSD Korn Shell)
  - ...





# Licencias en el AOSP

---

- Problema: Ciertas licencias atentan contra el modelo de negocio/distribución de Android
  - SW con licencia GPL (p.ej., GNU libc) obliga a que el desarrollador de una variante de Android distribuya el código fuente junto con la versión binaria
    - Además código fuente distribuido hereda la licencia GPL
  - Licencia de Java SDK de Oracle no permite distribución de productos derivados
- Google intenta solventar este problema de dos formas:
  - 1 Uso de componentes SW con licencia BSD o ASL (Apache)
    - Si no existe componente open-source NO-GPL, Google desarrolla dicho componente
  - 2 Evitar distribución de componentes del SDK de Oracle en el AOSP
    - Los desarrolladores de Android usan el compilador de java de Oracle SDK para compilar aplicaciones y el propio AOSP pero el compilador no se distribuye en el AOSP



# Contenido

---



**1** Introducción

**2** Arquitectura y Componentes

**3** Android vs. GNU/Linux



# Android vs. GNU/Linux



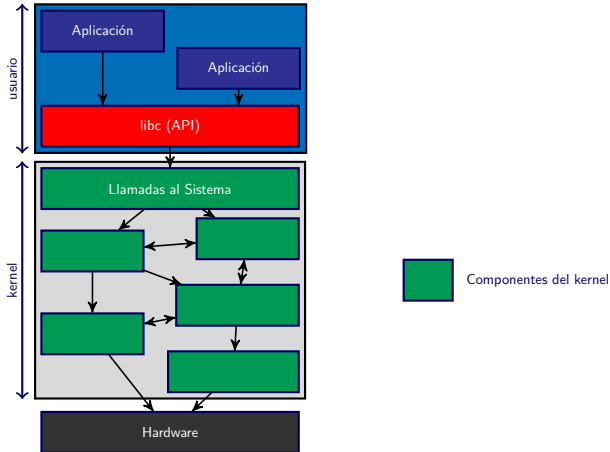
## Principales Diferencias en Arquitectura Interna

- Modelo de aplicaciones
- Modelo de *drivers*
- Estructura del sistema de ficheros
- Diferencias a nivel de kernel (*Androidized kernel*)



# Modelo de aplicaciones: GNU/Linux

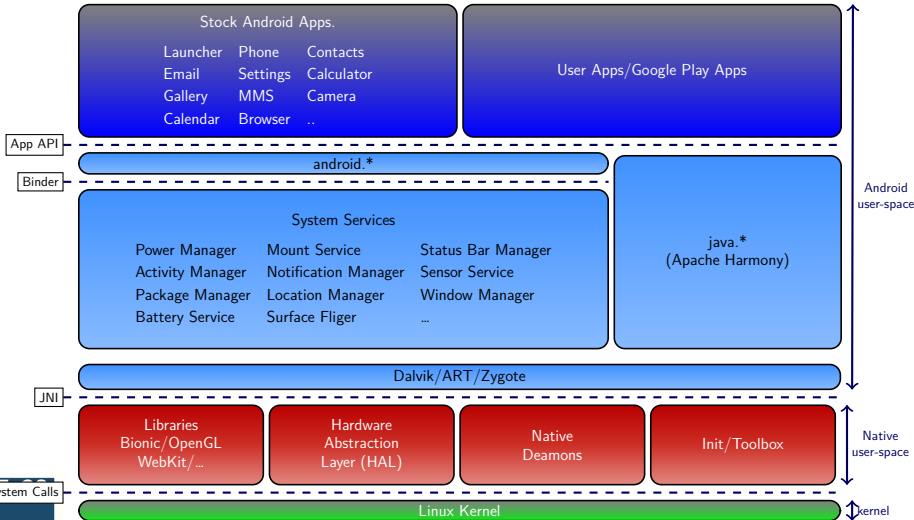
- En GNU/Linux las aplicaciones nativas interactúan con el kernel a través de la libc







# Modelo de aplicaciones: Android

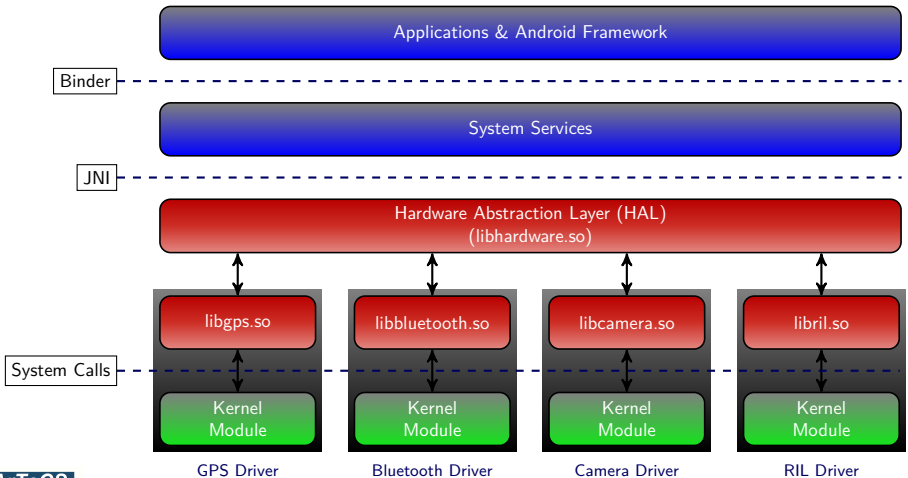


# Modelo de *drivers*

---

- En GNU/Linux la funcionalidad de un driver se suele encapsular completamente en un módulo del kernel
  - Los procesos de usuario acceden a las funciones del driver a través de la libc, del VFS (Virtual File System) o vía ficheros de dispositivo (p.ej.: /dev/sda1)
- Android ofrece un modelo alternativo (HAL) *más adecuado* para drivers propietarios de HW específico (GPS, Audio, Telefonía-RIL,...)
  - Fabricante distribuye módulo del kernel que implementa solamente soporte de bajo nivel (puede ser GPL)
  - La mayor parte de la funcionalidad del driver se encapsula en biblioteca dinámica (.so) propietaria
    - La biblioteca se carga bajo demanda
    - Ej: libaudio.so, libril.so, libgps.so
  - Las aplicaciones Android interactúan con la biblioteca a través de los servicios del sistema

# Modelo de *drivers*: Android





# Estructura del sistema de ficheros: FHS

---

- La mayoría de las distribuciones de Linux usan el estándar FHS (*Filesystem Hierarchy Standard*) para estructurar el sistema de ficheros
- El FHS define el nombre, utilidad y contenidos de los principales directorios del sistema
  - `/bin`: Comandos esenciales
  - `/boot`: Ficheros para el gestor de arranque, imagenes del kernel, ficheros de configuración del kernel
  - `/etc`: Ficheros de configuración del sistema
  - ...
- Android evita explícitamente usar los directorios definidos por el FHS
  - Permite crear (potencialmente) estructuras híbridas del SF
  - GNU/Linux + Android





# Estructura del sistema de ficheros: Android

- Android utiliza dos directorios principales:
  - 1 /system: Directorio inmutable generado durante la compilación del SO
    - Binarios, librerías estáticas y dinámicas
    - Ficheros .jar
    - Ficheros de configuración
    - Aplicaciones Android estándar (*Stock Apps*)
  - 2 /data: Directorio donde se almacena el contenido mutable
    - Aplicaciones de usuario
    - Datos de las aplicaciones y del usuario
- Estos dos directorios se montan en particiones separadas del sistema de ficheros raíz, que se monta como RAM disk.
- Además de los dos directorios aparecen muchos otros con la funcionalidad habitual de HFS
  - /proc, /dev, /sys, /etc, /sbin, /mnt





# Características del kernel en Android

## Abstracciones de Android

- Binder
- Wakelocks
- klogger
- Ashmem
- Low Memory Killer
- Paranoid Network
- Alarm Timers
- RAM Console
- ADB
- YAFFS2
- Timed GPIOs

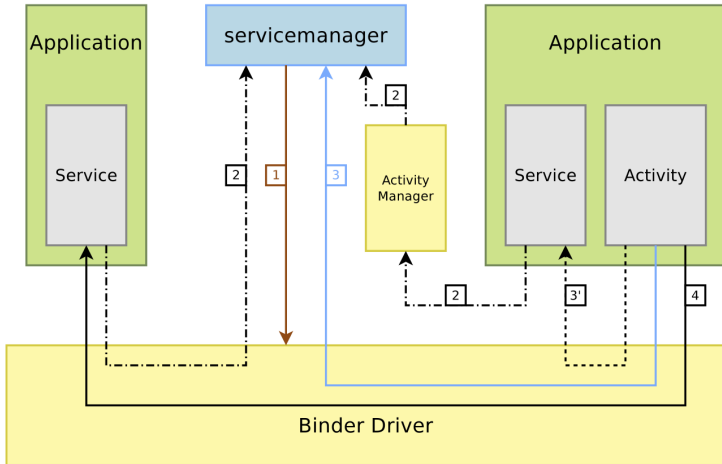






- Binder es un mecanismo de comunicación entre procesos (IPC)
  - Google se basó en el proyecto OpenBinder (BeOS) para crearlo
- Abstracción para permitir invocación de métodos remotos con soporte del kernel
  - RPC orientado a objetos
  - Concepto similar a JAVA RMI pero sin sockets (comunicación dentro de la máquina)
- Es una abstracción clave de Android, sin ella no podría funcionar
  - Llamadas de una aplicación a los servicios del sistema (Android Framework)
  - Comunicación entre aplicaciones
  - Comunicación entre componentes de una misma aplicación



# Binder



 `ioctl(BINDER_SET_CONTEXT_MGR)`  
 service registration

 service lookup  
 service call  
 service direct call





# Wakelocks: Motivación

- Los procesadores actuales tienen distintos modos de consumo de energía
  - Modos *idle*: Estado usado cuando el procesador está totalmente inactivo. No permite la ejecución de instrucciones.
  - Modos de bajo consumo: frecuencia y voltaje bajos
  - Modos de alto rendimiento: alta frecuencia y consumo elevado
- El kernel Linux explota estos estados para ahorrar energía
  - No solo los cores, sino también los discos o la pantalla tienen estados de consumo
- Ejemplo:
  - Al bajar la tapa de un portátil, el kernel activa el modo suspensión:
    - Los cores, la pantalla y los discos se “apagan”
    - La RAM permanece activa





# Wakelocks

- Este modo “suspensión” no es apropiado para un móvil
  - Ej: El usuario no quiere que la música deje de sonar al bloquear la pantalla
- El mecanismo de *Wakelocks* proporciona una solución a este problema en Android
- Idea general
  - 1 El kernel activa el modo suspensión siempre que puede
  - 2 Las aplicaciones, drivers y otros servicios de Android indican al sistema cuándo quieren que éste se mantenga activo
    - Adquieren un lock especial → *wakelock*
- Las aplicaciones y drivers tienen que usar el API de wakelocks
  - Requiere cambios significativos y problemáticos en el código de Linux





# Ashmem

---

- La memoria compartida (MC) es uno de los mecanismos de comunicación entre procesos disponibles en la mayoría de SSOO
- Linux implementa POSIX SHM (Unix System V)
  - Problema: Las regiones de memoria compartida globales pueden no llegar a liberarse nunca
- Ashmem es la alternativa de memoria compartida en Android
  - El kernel usa contadores de referencia para saber si las regiones de MC están en uso o no
  - El kernel puede reducir el tamaño de las regiones de MC en situaciones críticas
- Modo de uso: Comunicación entre ap. A y ap. B
  - A abre una región de memoria compartida, que tiene asociado un descriptor de fichero interno
  - El descriptor de fichero se envía de A a B a través de Binder





# Low Memory Killer

- Cuando el sistema se queda sin memoria, Linux activa el OOM (Out-Of-Memory) Killer
  - Mata los procesos que usan más memoria
- El OOM Puede llegar a matar procesos de usuario críticos en Android (Telephony stack, Graphic subsystem, etc) y dejar vivos procesos menos importantes (Música, Juegos,...)
- Android introduce el Low Memory Killer (LMK)
  - Se activa antes que el OOM killer del kernel
  - Prioridades a la hora de matar procesos
    - 1 Aplicaciones que pueden guardar/restaurar estado
    - 2 Procesos en background no críticos
    - 3 Aplicaciones en primer plano
- La introducción del LMK hace que el OOM killer no se llegue a invocar





# Paranoid Network

---

- El kernel Linux *vanilla* permite que cualquier aplicación abra sockets para comunicarse por la red
- En un SO para móviles es crítico restringir el acceso a la red a las aplicaciones controlar que aplicaciones tienen acceso
  - Implementar este tipo de restricciones involucra cambios en el kernel
- En Android se otorga/deniega permiso de acceso a cada aplicación por cada tipo de protocolo de comunicación (TCP/IP, raw sockets, Bluetooth, RFCOMM,...)





# Referencias

---

- *Embedded Android*

- Cap. 1 *"Introduction"*
- Cap. 2 *"Internals Primer"*

- Presentaciones

- *Android System Development*, Free Electrons
- *Android Internals*, Marko Magenta (Marakana)





## LIN - Introducción a Android Versión 0.2

©J.C. Sáez

*This work is licensed under the Creative Commons **Attribution-Share Alike 3.0 Spain License**. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/3.0/es/> or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.*

*Esta obra está bajo una licencia **Reconocimiento-Compartir Bajo La Misma Licencia 3.0 España de Creative Commons**. Para ver una copia de esta licencia, visite <http://creativecommons.org/licenses/by-sa/3.0/es/> o envíe una carta a Creative Commons, 171 Second Street, Suite 300, San Francisco, California 94105, USA.*

Este documento (o uno muy similar) está disponible en <https://cv4.ucm.es/moodle/course/view.php?id=62472>

