ATC AND-802

ANDROID SECURITY ESSENTIALS V8

Certificación "Android Application Engineer"

TEMA 1
PERMISOS

- INTRODUCCIÓN
- ARQUITECTURA DE LA PLATAFORMA ANDROID
- ARQUITECTURA DE LA SEGURIDAD ANDROID
- PERMISOS

INTRODUCCIÓN

- Filosofía LINUX (la hereda y la amplía)
  - Protección de ficheros rwxr-xr-- UGO
  - UID

- Cada APP está aislada una de otra y a su vez del Sistema base Android

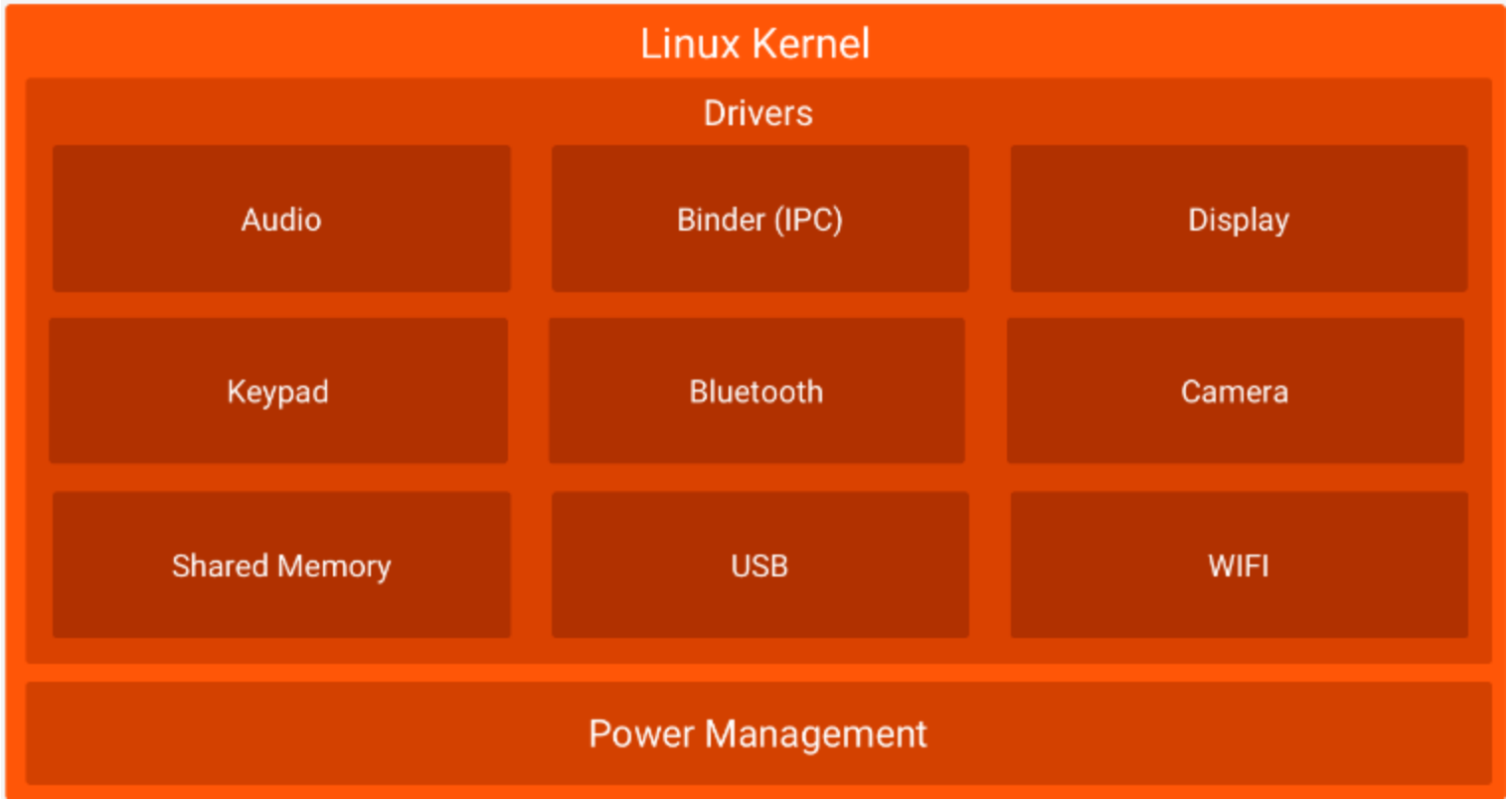
INTRODUCCIÓN

- Android protege algunos recursos con Permisos

(ej: Cámara)

- Si la APP solicita y obtiene el permiso adecuado, puede usar ciertas APIS protegidas

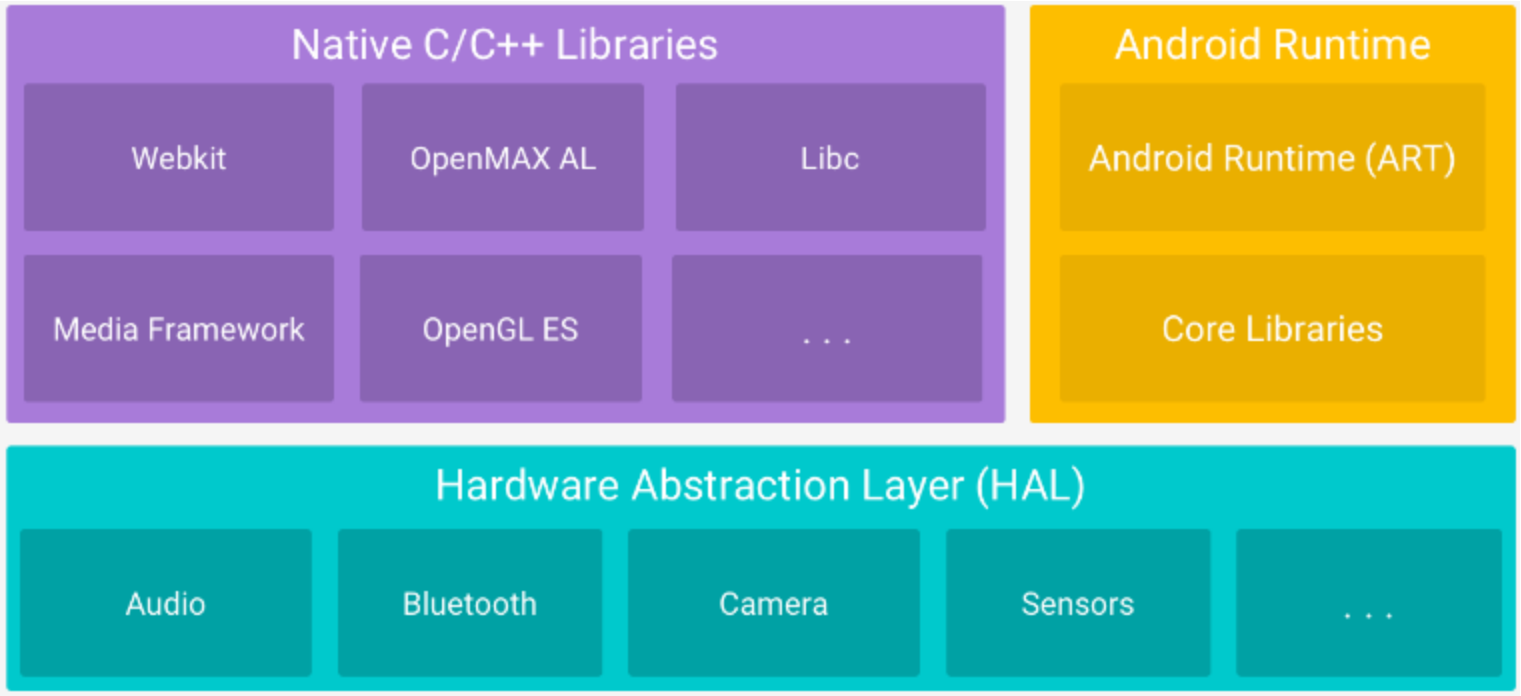
ARQUITECTURA SW/HW



ARQUITECTURA SW/HW
KERNEL

- Kernel: núcleo del Sistema Operativo
- Gestión de Memoria, Procesos e hilos
- Encendido, pantalla, interfaz de red, etc.
- Capa entre el software y el hardware de la máquina

ARQUITECTURA SW/HW



ARQUITECTURA SW/HW
HAL

- Hardware Abstraction Layer

- Ofrece un API para desde Java/Kotlin usar el HW
- Implementa controladores/librerías para cada módulo
- Enlaza alto nivel con bajo nivel

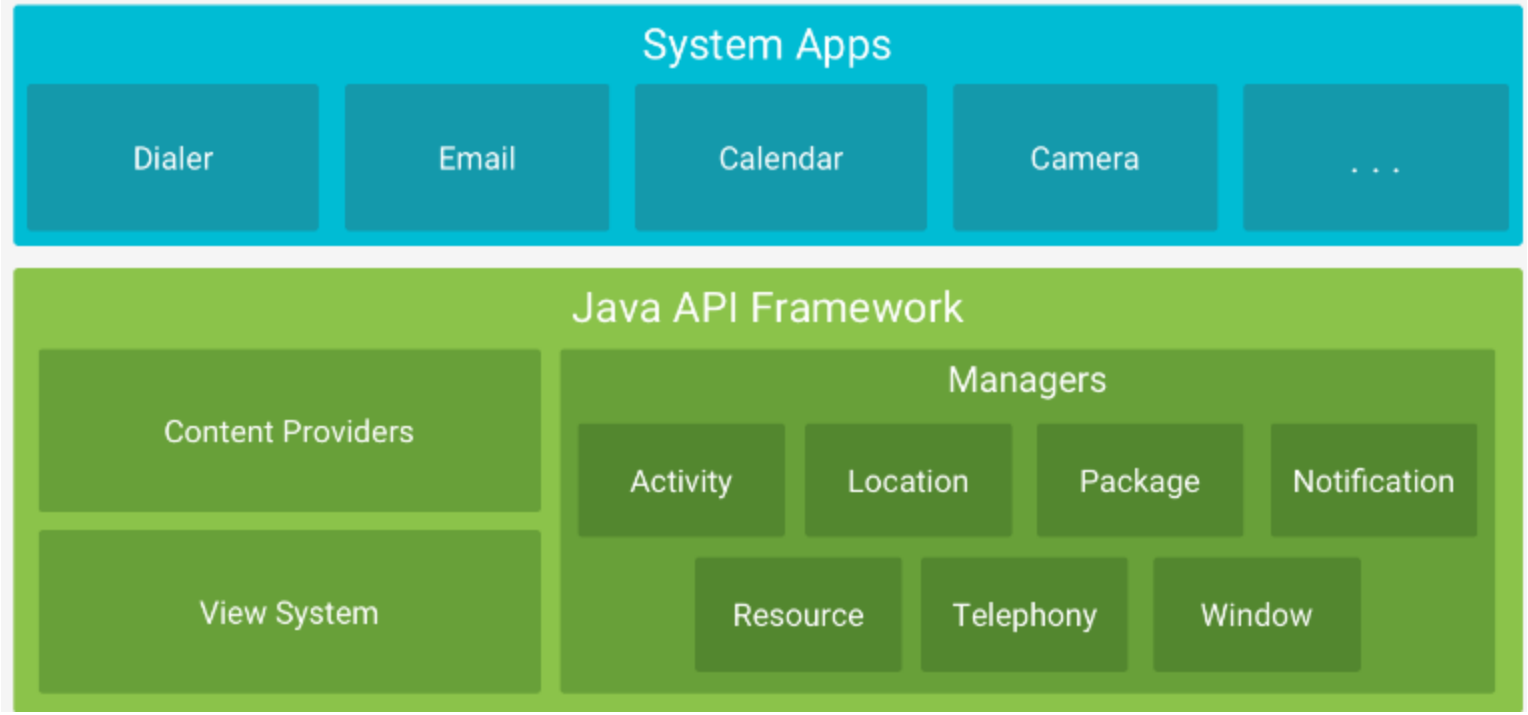
ARQUITECTURA SW/HW
ART

- Android RunTime: una Java Virtual Machine para cada App en ejecución
- Compilación, interpretación del código fuente, recolector de basura
- Hasta API 21 (5) Dalvik (DEX - class -)

ARQUITECTURA SW/HW NATIVE C/C++

- Librerías de bajo nivel usadas en la implementación del HAL y ART
- El API de Java/Kotlin usa estas librerías por detrás
- Para gráficos, almacenamiento, navegador, etc.

ARQUITECTURA SW/HW



ARQUITECTURA SW/HW
JAVA API Framework

- Todo lo que podemos hacer como programadores queda expuesto en este nivel

- Gestión de Vistas, Ciclo de vida de Actividades, Gestor de Notificaciones, Gestión de recursos (res)

ARQUITECTURA SW/HW
System Apps

- Aplicaciones que viene por defecto instaladas
- Tales como el Calendario, Mensajes, Agenda, Cámara, etc.
- Podemos usar estas aplicaciones desde la nuestra (vía Intents implícitos)

ARQUITECTURA DE LA SEGURIDAD

- Cada app está aislada en un *sandbox* o ámbito
- Por defecto, sólo puede usar lo de su ámbito
- Sólo vía permisos, la aplicación podrá acceder a recursos fuera de su ámbito

ARQUITECTURA DE LA SEGURIDAD
Firma de aplicaciones

- Cada app debe firmarse antes de publicarse
- La firma es un mecanismo de autenticación basado en la infraestructura de clave pública (PKI)
- Se debe emplear la misma firma para el sucesivo lanzamiento de las versiones de una misma app

ARQUITECTURA DE LA SEGURIDAD
Firma de aplicaciones

- Es esencial que la firma está protegida con una clave privada que debe mantenerse en el tiempo

- Sólo aplicaciones con el mismo atributo android:sharedUserId y con la misma firma, podrán compartir datos

ARQUITECTURA DE LA SEGURIDAD
Instalación de aplicaciones

- Las aplicaciones declaran una serie de permisos para acceder a determinados recursos del entorno (normalmente de Sistema Android base)

- Desde la versión 6 (API 23) se exigen a los programadores una gestión dinámica de los mismos, que permita al usuario revisar y retirar los permisos otorgados a una app.

PERMISOS ¿para qué?

- Los permisos, conceden a la aplicación el privilegio de acceder a determinados recursos protegidos del Sistema

- De modo que antes de usar ciertas API's, deberé obtener ese permiso

PERMISOS ¿para qué?

- Cada app tiene un Id único, de modo que Android maneja la concurrencia de varios procesos/apps y bloquea los recursos bajo demanda
- Si un permiso se deniega, en realidad se deniega el uso del recurso asociado a ese permiso
- Los permisos pueden ser concedidos bien por Android o por otra aplicación

PERMISOS - Ejecución

- Se pueden requerir permisos en las siguientes circunstancias:
  - Llamadas al sistema
  - Lanzamiento de una actividad
  - Envío y recepción de señales *broadcast*

PERMISOS - Ejecución

- Uso de un *Content Provider*
- Ejecución de Servicios

PERMISOS – Niveles de protección

- Los permisos pertenecen a una de estas cuatro categorías o niveles de protección:
  - Normal o Cero
  - Peligroso o Uno
  - Firma (Apps) o Dos
  - Firma (Sistema) o Tres

PERMISOS – Nivel Cero

- Permisos otorgados automáticamente, sin la intervención del usuario. Su aprobación no puede causar daños al dispositivo o al usuario

  - ACCESS_LOCATION_EXTRA_COMMANDS
  - ACCESS_NETWORK_STATE
  - ACCESS_NOTIFICATION_POLICY
  - ACCESS_WIFI_STATE
  - BLUETOOTH
  - BLUETOOTH ADMIN

PERMISOS – Nivel Cero

- BROADCAST_STICKY
- CHANGE_NETWORK_STATE
- CHANGE_WIFI_MULTICAST_STATE
- CHANGE_WIFI_STATE
- DISABLE_KEYGUARD
- EXPAND_STATUS_BAR
- GET_PACKAGE_SIZE
- INSTALL_SHORTCUT
- INTERNET
- KILL_BACKGROUND_PROCESSES
- MODIFY_AUDIO_SETTINGS
- NFC

PERMISOS – Nivel Cero

- USE_FINGERPRINT
- VIBRATE
- WRITE_SYNC_SETTINGS
- WAVE_LOCK

PERMISOS – Nivel Uno

- Los permisos peligrosos o de nivel uno, requieren la aprobación expresa del usuario. Así que además de declararlos, hay que gestionarlo programáticamente. Forman parte de esta categoría, los siguientes:
  - READ_CALENDAR
  - WRITE_CALENDAR
  - CAMERA

PERMISOS – Nivel Uno

- READ_CONTACTS
- WRITE_CONTACTS
- GET_ACCOUNTS
- ACCESS_FINE_LOCATION
- ACCESS_COARSE_LOCATION
- RECORD_AUDIO
- READ_PHONE_STATE
- CALL_PHONE
- READ_CALL_LOG
- WRITE_CALL_LOG
- ADD_VOICEMAIL
- USE_SIP

PERMISOS – Nivel Uno

- PROCESS_OUTGOING_CALLS
- BODY_SENSORS
- SEND_SMS
- RECEIVE_SMS
- READ_SMS
- RECEIVE_WAP_PUSH
- RECEIVE_MMS
- READ_EXTERNAL_STORAGE
- WRITE_EXTERNAL_STORAGE

PERMISOS – Nivel Dos

- Este permiso lo concede Android automáticamente a aquellas aplicaciones que tienen la misma firma que la aplicación que protege / declara ese permiso. Ejemplos de permisos de este nivel, son:
  - BIND_ACCESSIBILITY_SERVICE
  - ACCESS_INPUT_FLINGER
  - CRYPT_KEEPER

PERMISOS – Nivel Dos

- BIND_TEXT_SERVICE
- BIND_VPN_SERVICE
- INSTALL_AS_USER
- BIND_WALLPAPER
- MANAGE_MEDIA_PROJECTION
- DELETE_PACKAGES
- ACCESS_NETWORK_CONDITIONS
- REBOOT
- BIND_DREAM_SERVICE
- ALLOW_ANY_CODEC_FOR_PLAYBACK
- BIND_CONDITION_PROVIDER_SERVICE
- BIND_JOB_SERVICE

PERMISOS – Nivel Dos

- CONFIRM_FULL_BACKUP
- ACCESS_ALL_PRINT_JOBS
- ACCESS_BLUETOOTH_SHARE
- C2D_MESSAGE
- SEND_DOWNLOAD_COMPLETED_INTENTS
- BIND_PRINT_SPOOLER_SERVICE
- MODIFY_AUDIO_ROUTING
- CAPTURE_SECURE_VIDEO_OUTPUT
- ACCESS_KEYGUARD_SECURE_STORAGE
- FILTER_EVENTS
- BIND_REMOTE_DISPLAY
- CAPTURE_TV_INPUT

PERMISOS – Nivel Dos

- SET_ORIENTATION
- MODIFY_NETWORK_ACCOUNTING
- REMOVE_DRM_CERTIFICATES
- TV_INPUT_HARDWARE
- SET_POINTER_SPEED
- MOVE_PACKAGE
- CONFIGURE_WIFI_DISPLAY
- REQUEST_SUPERUSER
- CALL_PRIVILEGED
- BIND_DEVICE_ADMIN
- ACCESS_CONTENT_PROVIDERS_EXTERNALLY
- PACKAGE_USAGE_STATS

PERMISOS – Nivel Dos

- PERFORM_CDMA_PROVISIONING
- RETRIEVE_WINDOW_TOKEN
- DELETE_CACHE_FILES
- MEDIA_CONTENT_CONTROL
- COPY_PROTECTED_DATA
- START_PRINT_SERVICE_CONFIG_ACTIVITY

PERMISOS – Nivel Tres

- Permisos especiales. Sólo las aplicaciones con la misma firma que la imagen del Sistema Android pueden obtenerlos.

- El máximo nivel que un programador usaría es el dos

PERMISOS - Contexto

- Según donde son requeridos los permisos, pueden declararse en dos niveles o contextos
  - A nivel de Aplicación
  - A nivel de Componente

PERMISOS - Aplicación

- Los permisos a este nivel se declaran dentro de la etiqueta <application> en el fichero de Manifiesto

- Son los permisos que la aplicación necesita para ejecutar sus operaciones.

PERMISOS - Aplicación

- A su vez, los permisos de la aplicación se dividen en:
  - Permisos del Sistema: qué permisos necesita la aplicación fuera de su *sandbox*
  - Permisos definidos en la aplicación: el programador los define para proteger el acceso a lo que él desarrolla

PERMISOS - Componente

- El usuario puede proteger componentes (clases) individuales. Los componentes son clases especiales, declaradas en el Manifiesto y gestionadas en parte por el Sistema. Hay 4 tipos de componentes:
  - Actividades
  - Servicios
  - *Broadcast Receivers*
  - *Content Providers*

PERMISOS - Componente

- Cada uno de estos componentes declarados en el Manifest, añaden sus etiquetas propias de permiso, dentro de sus elementos XML

```
<activity
android:name=”AndroidATC”
android:permission=”com.Android.ATC.start_Activity”
```

PERMISOS - Componente

```
<service
   android:enabled=”true”
   android:name=”.ServiceExample”
   android:permission=”android.permission.REQUEST_DELETE_
   PACKAGES”/>
```

PERMISOS - Componente

```
<provider
android:name=”AndroidATC”
android:authorities=”com.Android.ATC.SMS.
contentprovider”
android:grantUriPermission=”true”
android:readPermission=”android.permission.READ_SMS”
android:writePermission=”android.permission.WRITE_SMS”
/>
```

PERMISOS - Componente

```
<receiver
   android:name=".AndroidATC"
   android:permission="android.permission.READ_CALL_LOG">
<intent-filter>
   <action android:name="Android.ATC.CALL_LOG" />
</intent-filter>
</receiver>
```

PERMISOS - Componente

- Los Broadcast Receivers pueden gestionar los permisos en los dos sentidos:
  - El emisor o Broadcast: Qué apps pueden recibir el mensaje (boot)
  - El Receptor: Qué emisor o broadcast recibo/escucho

PERMISOS - Componente

- Por ejemplo así, se emite una señal de broadcast

```
val broadcastIntent = Intent(CALL_LOG)
context.sendBroadcast(broadcastIntent, “Android.ATC.
CALL_LOG”) //
```

PERMISOS - Componente

- Sólo el receptor configurado así tendrá acceso a la señal

```
val intentFilter = IntentFilter(CALL_LOG)
var ATC = ATCreceiver()
context.registerReceiver(ATC, intentFilter,”Android.
ATC.CALL_LOG”, null)
```

PERMISOS - Personalizados

- Además de los permisos del Sistema, que ya están agrupados, el usuario puede declarar sus propios permisos e incluso sus propios grupos.

PERMISOS - Personalizados

- Para crear un nuevo permiso, declaro esta etiqueta en el fichero de Manifiesto. El permiso tiene un nombre, una descripción, un icono y un nivel, entre otros.

```
<permission
android:name=”@string/androidatcname”
android:description=”@string/androidatcdesc”
android:icon=”@drawable/Androidatcicon”
android:protectionLevel=”normal|dangerous|signature
|signatureOrSystem|system|development”/>
```

PERMISOS – Grupos Personalizados

- Los permisos, están agrupados en una clasificación de Android.
- Igualmente, puedo crear un grupo de permisos, declarando la etiqueta <permission-group> en el fichero de Manifiesto y aportando los atributos:

PERMISOS - Grupos Personalizados

- Nombre: El nombre del permiso
- Label: El nombre para el usuario en tiempo de instlación
- Icono, logo: La imagen personalizada del grupo de permisos
- Descripción: breve descripción
- Flags Destacar Información personal, privacidad económica
- Priority : La prioridad

PERMISOS – Árbol de permisos

- Igualmente, puedo crear un árbol de permisos o <permission-tree>, que simplemente sirva como paquete o aglutinador de permisos, para su posterior reutilización o referencia por otra aplicación

- Los árbol de permisos se describen con
  - Nombre
  - Etiqueta
  - Logo e Icono

TEMA 2
GESTIÓN DE LA SEGURIDAD

- INTRODUCCIÓN
- EL FICHERO DE MANIFIESTO
- MODIFICANDO LA POLÍTICA DE SEGURIDAD

INTRODUCCIÓN

- La configuración y la gestión de la seguridad se hace mayormente desde el fichero de manifiesto - AndroidManifest.xml -

- Los componentes (clases principales de Android) así como otros aspectos de la seguridad y permisos se describen en este fichero.

EL FICHERO DE MANIFIESTO

- La gramática o secciones del documento son:

<?xml version=”1.0” encoding=”utf-8”?>

<manifest>

<uses-permission />

<permission />

<permission-tree />

<permission-group />

<instrumentation />

<uses-sdk />

EL FICHERO DE MANIFIESTO

<uses-confguration />

<uses-feature />

<supports-screens />

<compatible-screens />

<supports-gl-texture />

<application>

EL FICHERO DE MANIFIESTO

<activity>

<intent-filter>

<action />

<category />

<data />

</intent-filter>

<meta-data />

</activity>

EL FICHERO DE MANIFIESTO

<activity-alias>

<intent-flter>.....</intent-flter>

<meta-data />

</activity-alias>

EL FICHERO DE MANIFIESTO

<service>

<intent-flter>.....</intent-flter>

<meta-data />

</service>

<receiver>

<intent-flter>.....</intent-flter>

<meta-data />

</receiver>

EL FICHERO DE MANIFIESTO

<provider>

<grant-uri-permission />

<meta-data />

<path-permission />

</provider>

<uses-library />

</application>

</manifest>

EL FICHERO DE MANIFIESTO - <manifest>

- Los atributos del elemento <manifest> son:
  - package Nombre del paquete e Id de la app
  - android:sharedUserId El id de usuario. Android, por defecto, usa un id distinto para cada app. De modo que sólo un usuario con ese id, puede acceder a los recursos de esa app. Si dos apps comparten el valor de este atributo, ambas pueden compartir recursos (vistas, ficheros, etc.). Obsoleto desde el API 29

EL FICHERO DE MANIFIESTO - <manifest>

Esta constante dejó de estar disponible a partir del nivel de API 29.

Los IDs de usuario compartidos generan comportamientos no determinístico en el administrador de paquetes. Por lo tanto, no se recomienda para nada su uso, y es posible que se quiten en una versión futura de Android. En su lugar, usa mecanismos de comunicación adecuados, como servicios y proveedores de contenido, para facilitar la interoperabilidad entre componentes compartidos. Las apps existentes no pueden quitar este valor, ya que no se admite la migración desde un ID del usuario compartido. En estas apps, agrega `android:sharedUserMaxSdkVersion="32"` para evitar el uso del ID del usuario compartido en instalaciones de usuarios nuevos.

EL FICHERO DE MANIFIESTO - <manifest>

- versionCode Número natural que identifica cada versión
- versionName Código alfanumético que identifica cada versión
- InstallLocation La ubicación donde el ejecutable APK se instala. Puede asumir los valores internalOnly|auto|preferExternal

EL FICHERO DE MANIFIESTO - <application>

- Los atributos del elemento <application> son:
  - allowTaskReparenting: Si es true la actividad puede relanzarse con otra tarea
  - backupAgent: La clase que realiza la copia de seguridad (nombre canónico o abreviado)
  - debuggable: Determina si la app puede depurarse en el móvil del usuario final. False por defecto
  - description: Texto descriptivo de la aplicación

EL FICHERO DE MANIFIESTO - <application>

- enabled: Por defecto a true, permite que Android ejecute la aplicación y sus componentes. Los componentes heredan este valor de este elemento/etiqueta
- hasCode Por defecto a true, indica si la aplicación contiene código java / bytecodes Si sólo tuviera código C, debería estar a false
- hardwareAccelerated: Por defecto a true. Versiones previas a la API 14, podrían carecer de este motor Open GL implementado apartir de entonces.
- icon: enlace al recurso gráfico que será el icono de la aplicación

EL FICHERO DE MANIFIESTO - <application>

- killAfterRestore: Por defecto a true, la app se apaga tras configurar el teléfono o reestablecer la configuración original
- largeHeap: solicita más memoria en el proceso Java que una app normal (aunque tampoco garantiza que el Sistema se la dé)
- label: Literal usado para toda la app. Cada componente lo hereda si no lo redefine: Ej: título de la actividad
- logo: logo predeterminado para todos las actividades de la aplicación NOTA: el logo es un símbolo usado en el Action Bar , no es el icono de la aplicación
- manageSpaceActivity: Enlace a la actividad que permite gestionar el almacenamiento

EL FICHERO DE MANIFIESTO - <application>

- name: Nombre de la clase Application que se instancia antes que cualquier componente. La mayoría de apps no lo necesitan. Parecido a Object de Java
- permission: nombre del permiso requerido para ejecutar la aplicación Se aplica a todos los componentes (si estos no lo redefinen)
- persistent: Indica con un boolean si la ejecución de la aplicación es permanente. Sólo alguna del sistema lo son. True si lo requiere.
- process: El nombre del proceso en el que se ejecutan los componentes. Cada componente puede redefinirlo. Por defecto, es el nombre del paquete de la etiqueta manifest

EL FICHERO DE MANIFIESTO - <application>

- restoreAnyVersion Por defecto a false. Usado a true, permite restaurar a copias de seguridad de versiones anteriores o nuevas.
- requiredAccountType Si la app necesita alguna cuenta de whatsapp, telegram o google para su funcionamiento. Por defecto a null, con lo que se puede usar la app sin ninguna cuenta (Account)
- restrictedAccountType Indica a qué tipo de cuentas de usuario y su información se puede acceder desde la aplicación
- SupportsRtl: Indica si soporta mostrar diseños de derecha a izquierda Desde Api 17. Por defecto a false

EL FICHERO DE MANIFIESTO - <application>

- taskAffinity Un nombre que apela a todas las actividades y que sirve para finalizar todas a la vez, salvo que redefinan esta propiedad
- theme : El conjunto de estilos por defecto aplicable a la apariencia de la app. Igualmente, puede redefinirse por cada actividad.
- uiOptions Por defecto valor none, pero admite splitActionBarWhenNarrow, que añade una barra de acciones en la parte baja de la actividad
- vmSafeMode: Indica si el proceso que lanza la actividad interpreta o ejecuta el código compilado -false por defecto-

MODIFICANDO LA POLÍTICA DE SEGURIDAD

- COMPARTIENDO EL MISMO ID DE USUARIO
- ESTABLECIENDO PERMISOS
- ESTABLECIENDO PERMISOS PARA APP EXTERNAS
- MEMORIA EXTERNA
- MODO DEBUG
- BAKCUP COPIA SEGURIDAD

MODIFICANDO LA POLÍTICA DE SEGURIDAD
COMPARTIENDO EL MISMO ID DE USUARIO

- Si dos apps tienen la misma firma y además comparten el atributo sharedUserId del manifest, pueden ejecutarse el mismo proceso y acceder a los mismos datos
- El atributo de usuario compartido sigue la nomenclatura de los paquetes java "com.usuario.ejemplo"
- El gestor de aplicaciones de Android /gestor de paquetes, no garantiza el correcto comportamiento de aplicaciones con esta configuración, por lo que desde el Api 29 se desaconseja su uso

MODIFICANDO LA POLÍTICA DE SEGURIDAD
ESTABLECIENDO PERMISOS

- Con la etiqueta <uses-permission> la aplicación describe y solicita acceso a otros recursos del entorno
- Los permisos descritos en esta sección son mostrados en el proceso de instalación

MODIFICANDO LA POLÍTICA DE SEGURIDAD
ESTABLECIENDO PERMISOS PARA APPS EXTERNAS

- Para restringir el uso de componentes de mi aplicación, debo establecer permisos en la etiqueta application (de modo que lo heredan todos) o a nivel de componente individual : activity, provider, receiver o service.

```
<application
android:icon=”@drawable/ic_launcher”
android:label=”@string/app_name”
android:permission=”android.permission.WRITE_
EXTERNAL_STORAGE”>
```

MODIFICANDO LA POLÍTICA DE SEGURIDAD
MEMORIA EXTERNA

- Con el atributo installLocation del elemento manifest, especifico el lugar de instalación de la aplicaicón.
- El valor puede ser internalOnly preferExternal o auto
- Por defecto el Sistema intentará instalar la aplicación en la memoria interna

/data/app/<nombre_paquete>-<número_aleatorio>/ es la memoria interna y /mnt/ la ubicación externa

MODIFICANDO LA POLÍTICA DE SEGURIDAD
MODO DEBUG

- Con el atributo debuggable del manifest a true bajo el umbral del log a verbose, de modo que puedo ver con todo detalle el registro de eventos de la aplicación

- Cuando la aplicación se publique y cambie su modo a release, es muy importante establecer este valor a falso

MODIFICANDO LA POLÍTICA DE SEGURIDAD
BACKUP

- Desde el API 23 (versión 6) de Android, las aplicaciones de Android disponen de una copia de seguridad automática, de modo que los datos del usuario se guardan en la cuenta de Google Drive
- El límite de almacenamiento es 25MB
- Con el atributo allowBackup de application y backUp Agent, podemos configurar el destino de la copia en la nube

TEMA 3
PRIVACIDAD Y PROTECCIÓN DE DATOS

- PRINCIPIOS DE SEGURIDAD
- ENTORNO DEL DISPOSITIVO
- ESTADOS DE LOS DATOS
- VULNERABILIDADES Y ATAQUES
- PRINCIPOS DE PROTECCIÓN
- CONSEJOS DE CODIFICACIÓN

PRIVACIDAD Y PROTECCIÓN DE DATOS
PRINCIPIOS DE SEGURIDAD

- CONFIDENCIALIDAD
- INTEGRIDAD
- DISPONIBILIDAD

PRIVACIDAD Y PROTECCIÓN DE DATOS
PRINCIPIOS DE SEGURIDAD

- CONFIDENCIALIDAD
  - Los datos están a salvo de cualquier acceso no deseado. Sólo disponibles para el usuario legítimo
  - Los datos de un componente son sólo accesibles desde componente u otros con los debidos permisos

PRIVACIDAD Y PROTECCIÓN DE DATOS
PRINCIPIOS DE SEGURIDAD

- CONFIDENCIALIDAD
  - Los datos además deben estar encriptados

PRIVACIDAD Y PROTECCIÓN DE DATOS
PRINCIPIOS DE SEGURIDAD

- INTEGRIDAD
  - Los datos de tu aplicación no son alterados ni modificados durante su transmisión o almacenamiento. SSL previene el ataque "Man In the Middle"

PRIVACIDAD Y PROTECCIÓN DE DATOS
PRINCIPIOS DE SEGURIDAD

- DISPONIBILIDAD
  - Los datos están disponibles cuando se requieren
  - Consultar ajuestes Wi-Fi en el emulador: Ajutes -> Network -> Wi-Fi -> On -> AndroidWifi. Observar IP
  - Denegación de servicio (DDoS)

PRIVACIDAD Y PROTECCIÓN DE DATOS
ENTORNO DEL DISPOSITIVO

- Además de preocuparte por la seguridad como programador, debes también ser consciente del complejo ecosistema que rodea el desarrollo y la explotación del software.
- Los principales miembros de este entorno a tener en cuenta son:

PRIVACIDAD Y PROTECCIÓN DE DATOS
ENTORNO DEL DISPOSITIVO

- Consumers: Los usuarios finales, que interactúan con el resto de partes

- Services: Servicios de terceros que interactúan con tu dispositivo ej: Gmail, nube, etc

PRIVACIDAD Y PROTECCIÓN DE DATOS
ENTORNO DEL DISPOSITIVO

- Application Developers: Otros programadores
- OEM (Device Manufacturers) Los distintos fabricantes de hardware que corren Android. Hay muchas diferencias que testar
- Carriers: Las compañías que proveen la infraestructura de comunicación

PRIVACIDAD Y PROTECCIÓN DE DATOS
ENTORNO DEL DISPOSITIVO

- Infraestructura: En Protocolos como GSM o CDMA y sus protocolos recaen gran parte de la seguridad de nuestras aplicaciones

PRIVACIDAD Y PROTECCIÓN DE DATOS
ESTADOS DE LOS DATOS

- Los datos vistos como una entidad, atraviesan un flujo de estados o ciclo de vida:
  - Datos almacenados : Datos en la memoria del dispositivo, servidores o bases de datos. También llamados datos en descanso

PRIVACIDAD Y PROTECCIÓN DE DATOS
ESTADOS DE LOS DATOS

- Datos en procesamiento: Datos siendo procesados. Por ejemplo, leyéndose de disco o enviándose de una actividad a otra

- Datos en tránsito: Por ejemplo, entre el dispositivo y el servidor de autenticación

PRIVACIDAD Y PROTECCIÓN DE DATOS
VULNERABILIDADES Y ATAQUES

Con independencia de su estado, los datos están siempre ante un riesgo potencial. Los permisos nos ayudan a protegerlos pero no garantizan siempre su seguridad

Al gestionar los datos, hay que estar atento a dos aspectos: su vulnerabilidad y sus amenazas

PRIVACIDAD Y PROTECCIÓN DE DATOS
VULNERABILIDADES Y ATAQUES

VULENARIBILIDADES:

- Malware
- SuperUsuario Root
- Mismo almacenamiento
- Sin protección adicional

PRIVACIDAD Y PROTECCIÓN DE DATOS
VULNERABILIDADES Y ATAQUES

AMENAZAS Debemos preguntarnos ¿qué ocurre si un ataque accede a unas credenciales? Por cada dato debemos valorar:

- El tipo de dato a proteger / su valor
- El tipo de amenaza que supone su acceso
- Cómo puede ser atacado ese dato

PRIVACIDAD Y PROTECCIÓN DE DATOS
PRINCIPIOS DE PROTECCIÓN

Los datos sensibles puede protegerse por una política de permisos y además con encriptación

Debemos considerar diferentes niveles de protección, estrategia conocida como "Defensa en profundidad"

PRIVACIDAD Y PROTECCIÓN DE DATOS
PRINCIPIOS DE PROTECCIÓN

Como desarrolladores, debemos considerar:

- ¿Qué info necesitamos realmente guardar para llevar a cabo nuestra app? A menor volumen de datos almacenados, menos necesidad de protección.

PRIVACIDAD Y PROTECCIÓN DE DATOS
PRINCIPIOS DE PROTECCIÓN

Y además:

- ¿Dónde guardamos esa información? En el dispositivo, en un servidor, en la nube, en otra app...

PRIVACIDAD Y PROTECCIÓN DE DATOS
CONSEJOS DE PROTECCIÓN

1 No exportes un componente si no es necesario

2 Si lo haces, asegúrate de protegerlo con los permisos adecuados

3 Reduce el riesgo añadiendo *Intent Filters*

PRIVACIDAD Y PROTECCIÓN DE DATOS
CONSEJOS DE PROTECCIÓN

4 Evita guardar datos en los intents que lanzan una Actividad interesante para un atacante

5 Si envías datos a un servicio, asegúrate que es el correcto

6 No delegues en los Intent Filters para lanzar tus actividades. Puede ser una fuente de fallos

TEMA 4
SEGURIDAD Y ALMACENAMIENTO

- INTRODUCCIÓN
- DECISIONES SOBRE EL ALMACENAMIENTO
- PREFERENCES Y JAVA.IO
- MEMORIA CACHÉ
- BASES DE DATOS SQLITE

SEGURIDAD Y ALMACENAMIENTO
INTRODUCCIÓN

EXISTEN DISTINTOS MODOS APIS DE ALMACENAMIENTO EN MEMORIA SECUNDARIA

QUÉ TIPO ELEGIR SEGÚN LA IMPORTANCIA O CRITICIDAD DE LOS DATOS ES UNA CUESTIÓN QUE DEBEMOS RESOLVER

SEGURIDAD Y ALMACENAMIENTO
DECISIONES DE ALMACENAMIENTO

LOS ASPECTOS QUE INFLUYEN EN QUÉ DATOS Y CÓMO LOS GUARDAMOS SON:

- PRIVACIDAD
- PERIODICIDAD

SEGURIDAD Y ALMACENAMIENTO
DECISIONES DE ALMACENAMIENTO

SOBRE LA PRIVACIDAD DEBEMOS TENER EN CUENTA:

- ASPECTOS LEGALES (país)
- CONSENTIMIENTO
- ASPECTOS MORALES

SEGURIDAD Y ALMACENAMIENTO
DECISIONES DE ALMACENAMIENTO

SOBRE LA PERIODICIDAD DEBEMOS TENER EN CUENTA:

- VALIDEZ EN EL TIEMPO
- ID AL DISPOSITIVO
- ID AL USUARIO
- UBICACIONES

SEGURIDAD Y ALMACENAMIENTO
MECANISMOS DE PERSISTENCIA

ANDROID NOS OFRECE DISTITAS APIS COMO:

- SharedPreferences
- API java.io / java.nio
- SQLite
- Vectores Tipados (XML)

SEGURIDAD Y ALMACENAMIENTO
MECANISMOS DE PERSISTENCIA

Y DISTINTOS ESPACIOS DE MEMORIA COMO:

- Memoria Interna
- Memoria Externa
- Caché