Aplicaciones Android

Fundamentos de una aplicación

- Las aplicaciones Android están escritas en el lenguaje de programación orientado a objetos **Java (o Kotlin** desde agosto 2017) . El SDK de Android tiene una serie de herramientas que permitirán compilar el código, incluyendo los datos y los recursos y lo meterá todo en un fichero **APK** también conocido como paquete Android. Este fichero será nuestro instalador.
- Una vez instalada una aplicación, cada una de ellas tiene su propio sistema de seguridad, de tal modo que:

  - Cada aplicación será un **usuario diferente** dentro de Android como Sistema Operativo basado en un sistema Linux multiusuario. Este usuario será un ID de usuario Linux único.
  - Android dará **permisos** para todos los ficheros de una aplicación **únicamente para el usuario** que identifica dicha app.
  - Cada proceso tiene su **propia máquina virtual**, por lo que la ejecución de aplicaciones es totalmente independiente.
  - Por defecto, cada aplicación **corre en su propio proceso** Linux, el cual se gestiona a nivel de Sistema Operativo
- Con todas estas reglas, Android consigue implementar lo que se conoce como Principio de menor privilegio, consistente en otorgar los permisos justos a cada aplicación, de modo que **el sistema sea lo más seguro posible**.
- Pero todo esto es el funcionamiento **por defecto**, pues podremos gestionarlo según nos interese.

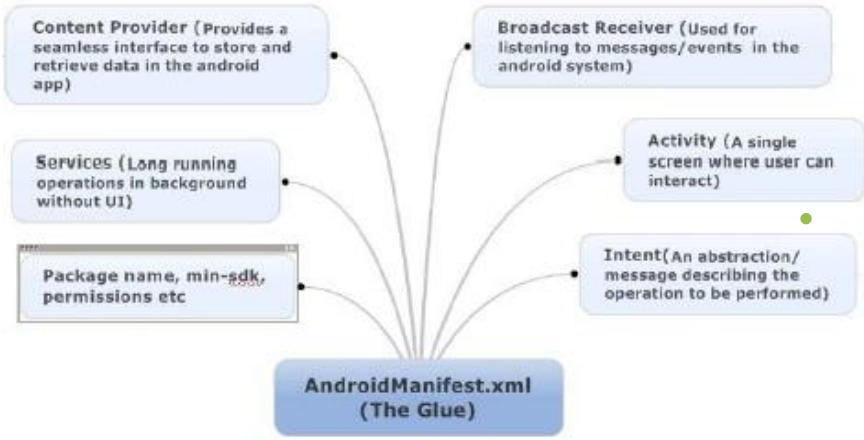
Componentes de una aplicación

- Hay 4 diferentes tipos de componentes:
  - **Activity**: Representa una pantalla independiente con una interfaz de usuario. A pesar de que nuestra aplicación dispondrá de múltiples pantallas interconectadas entre sí, nosotros deberemos generarlas individual e independientemente (pudiendo pasar datos entre ellas, en caso de ser necesario).
  - **Service**: Es un componente que corre de fondo para hacer operaciones de larga duración o trabajo en procesos remotos. Contrario a la actividad, no dispone de interfaz gráfica.
  - **Content Provider**: Este componente nos permite gestionar un conjunto de datos de la aplicación para compartir. Los Contactos son el ejemplo perfecto para este componente: datos que podemos compartir entre diferentes aplicaciones. Podemos crear nuestro propio conjunto de datos a compartir entre aplicaciones distintas.
  - **Broadcast Receiver**: El cuarto de los componentes nos permite responder a anuncios broadcast del sistema. Un buen ejemplo es si queremos gestionar cuando tengamos el aviso de batería baja (el cual enviará un mensaje broadcast). Podemos diseñar nuestros propios anuncios a los que responder.

- Un aspecto interesante de diseño de Android es que una aplicación A podría abrir un componente de una aplicación B. El ejemplo ideal es cuando queremos usar la cámara en nuestra app, podemos abrir el componente de la cámara que viene ya instalada por defecto en el sistema operativo. Para ello utilizamos un componente llamado **Intent**, que sirve para activar 3 de los 4 componentes de una app (todos excepto el Content Provider).

Fichero AndroidManifest.xml

- Será el encargado de comunicarle al sistema operativo:
  - los componentes de las que dispone la aplicación (actividades, servicios,…)
  - los permisos necesarios para la aplicación (acceso a Internet, uso de GPS,…)
  - la versión de Android mínima necesaria
  - el hardware y software requerido y/o usado (cámara, servicios de bluetooth,…)
  - las librerias externas que utiliza (como Google Maps…)
- Para ello, utilizaremos etiquetas, que en el caso de los componentes serán: *<activity>* *<service>* *<receiver>* *<provider>*



- Cada una de estas etiquetas tendrán una serie de atributos disponibles, donde indicaremos qué componente en cuestión será de todos los disponibles, icono o un sinfín de opciones disponibles.
- Además, si queremos indicar las capacidades de uno de nuestros componentes, podemos hacer uso de la etiqueta *<intent-filter>*.

Recursos de una aplicación

- Una aplicación se compone de algo más que de código Java, como **imágenes**, archivos de **audio**, **textos** y todo lo relacionados con la **presentación visual** de la aplicación.
- Siempre podemos especificar un recurso genérico o por defecto.
- Pero también podremos especificar diferentes recursos dependiendo del tipo de dispositivo en el que estemos, sin necesidad de modificar el código.
- Además también tendremos la opción de especificar que una versión concreta de un recurso es para una configuración específica. Para detallar la configuración específica podemos basarnos en idiomas, resolución, orientación del dispositivo…

Reglas básicas para el desarrollo de aplicaciones

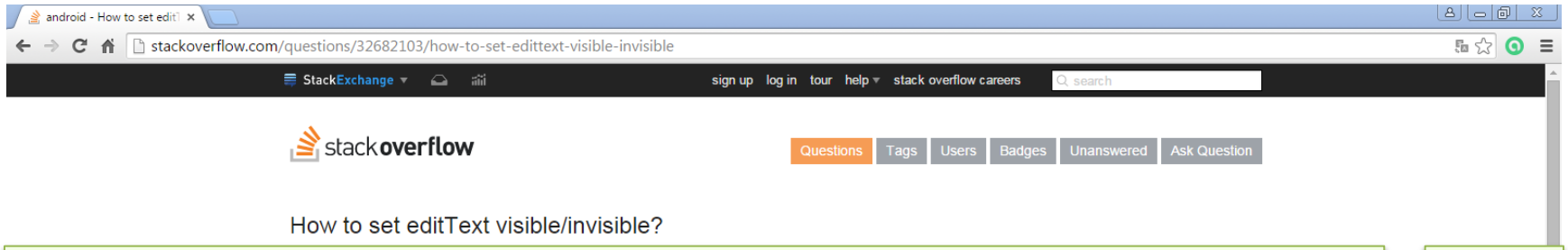
- Crear un código eficiente es una tarea prioritaria para garantizar el mejor rendimiento posible en los dispositivos móviles.
- Las dos reglas básicas para sistemas con recursos limitados son:
  - **No realizar trabajos que no sean indispensables.**
  - **No utilizar memoria si puede evitarlo.**

Consejos Java

- **Evitar la creación de objetos** (Supongamos que deseamos extraer cadenas de una entrada que hizo el usuario; en este caso resulta mejor devolver una subcadena en lugar de crear una copia)
- **Utilizar métodos nativos** (Cuando trabajemos con cadenas, resulta una buena práctica, utilizar métodos como *String.indexOf()* y *String.lastIndexOf()* que ya están implementados y que son más rápidos que utilizar algún método propio que tenga por objetivo reinventar la rueda)
- **Declarar constantes como final** (Cuando declaramos variables de tipo *final*, son gestionadas directamente por la máquina virtual).
- **Evitar el uso de enums** (Su uso no se recomienda cuando el tamaño y la velocidad son limitadas)
- **Evitar el uso de variables tipo float** (Por cuestiones de falta de soporte en Hardware para el punto flotante, se recomienda cambiar las operaciones que utilicen valores de tipo *float* a tipo *double*)
- **Uso prudente de la sintaxis en bucles for**

Enlaces interesantes

- Android Developers
- Android Developers blog
- Curso de programación Android de sgoliver
- Stack Overflow
- Tutorial Android UPV
- Tutoriales Vogella
- …



Esquema del flujo de trabajo