ANDROID EN 20 CONCEPTOS:

La semana pasada anunciábamos la nueva sección ***Aprende Android en 20 conceptos***, sección orientada a **introducirnos en los conceptos más básicos para empezar a programar nuestra propia aplicación Android**.

De hecho, aparte de los *20 conceptos*, incluimos la semana pasada un punto 0 de introducción, donde podíamos **instalarnos el entorno de desarrollo y tener nuestra máquina lista para empezar a programar**.

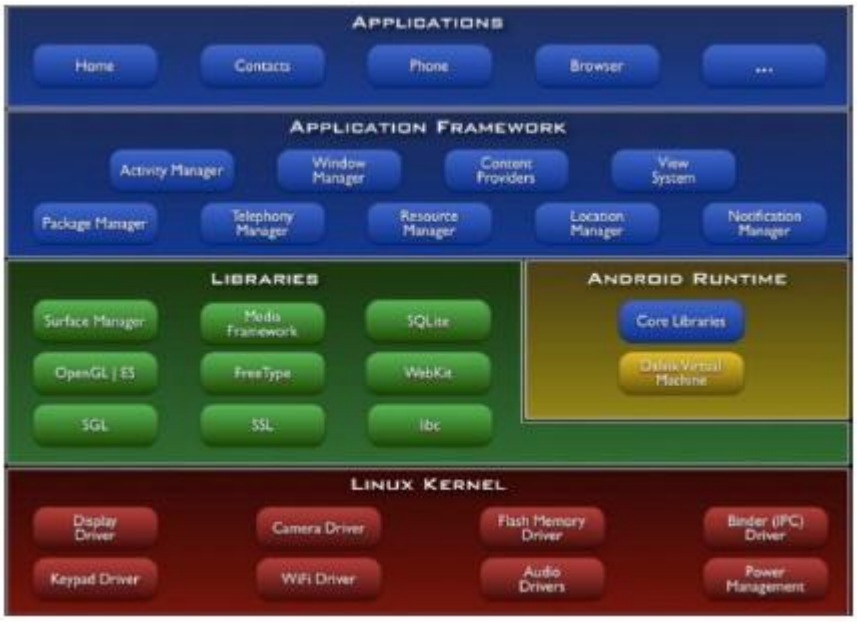
Hoy toca hablar ya de los dos primeros conceptos: los **Fundamentos de una aplicación** y los **Recursos.**

1. Fundamentos de una aplicación

Lo primero que tenemos que mencionar es que las aplicaciones Android están escritas en el **lenguaje de programación orientado a objetos Java**. El SDK de Android tiene una serie de herramientas que permitirán compilar el código, incluyendo los datos y los recursos (de los que hablaremos a continuación), y lo meterá todo en un fichero **APK**, o 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 virtua**l, 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, por ejemplo para compartir datos entre diferentes aplicaciones (un ejemplo perfecto son los Contactos).

Una vez conocido como funciona Android, es hora de pasar a definir los **componentes de una aplicación**. Éstos son los bloques básicos que podemos construir. 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). Entraremos en más detalle en esta clase cuando lleguemos al concepto 3.

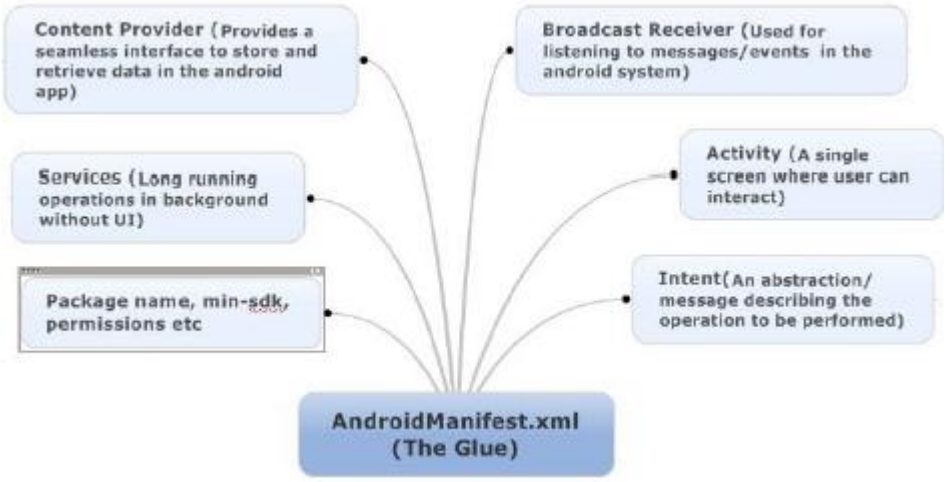
- **Service**: Es un **componente que corre de fondo para hacer operaciones de larga duración o trabajo en procesos remotos**. Contrario a la actividad, no dispone de interfaz gráfica. Veremos más detalles al llegar al concepto 11.
- **Content Provider:** Este componente nos permite **gestionar un conjunto de datos de la aplicación para compartir**. Los Contactos son el ejemplo perfecto para este componente: datos que podemos compartir entre diferentes aplicaciones. Pero podemos crear nuestro propio conjunto de datos (más detalle en el concepto 13).
- **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*), aunque podemos diseñar nuestros propios mensajes (más detalles en el concepto 8).

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 hacer una Activity con la cámara, o abrir el componente de la cámara que viene ya instalada por defecto en el sistema operativo.

Para ello utilizamos un mensaje llamado **Intent**, el cual también sirve para **activar 3 de los 4 componentes de una app** (todos excepto el Content Provider), Más adelante veremos cómo hay métodos específicos para abrir cualquier componente a través de un Intent (concepto 7).

Pero, **¿cómo sabe nuestra aplicación qué componentes tiene disponibles?** Para ello, existe el fichero **AndroidManifest.xml**. Este fichero será el encargado de comunicarle al sistema operativo:

- las componentes de las que dispone la aplicación
- los permisos necesarios para la aplicación (cámara, GPS…)
- la versión de Android mínima necesaria
- el hardware y software requerido y/o usado
- las librerias externas que utiliza (como Google Maps…)



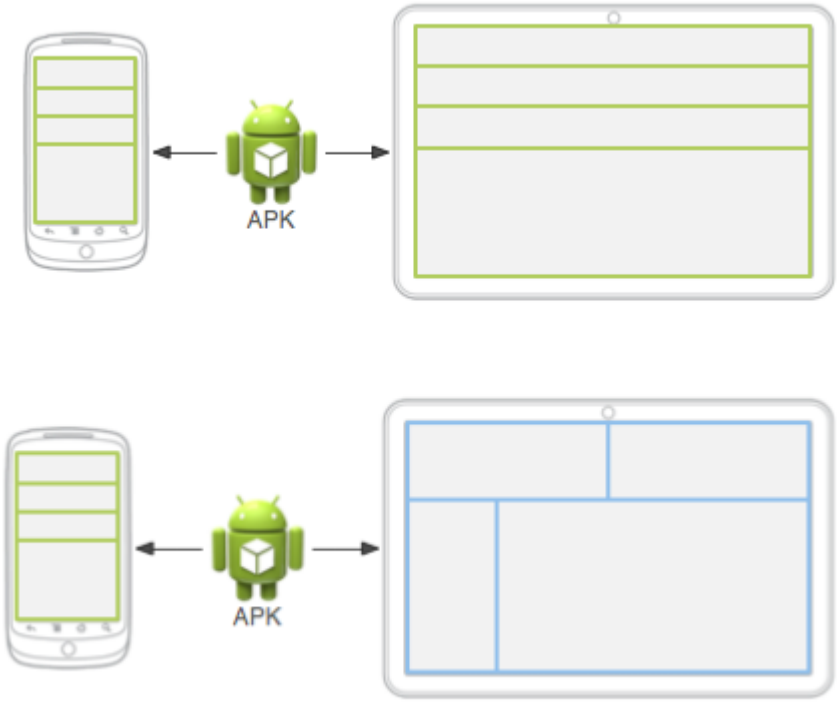
Para ello, utilizaremos etiquetas, que en el caso de los componentes serán:

<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>***.

2. Recursos de una app

A la hora de hacer un buen programa, siempre **hay que externalizar los recursos del código**, entendiendo por recursos imágenes, textos, estilos… De esta forma, también podremos **especificar diferentes recursos dependiendo del tipo de dispositivo en el que estemos, sin necesidad de modificar el código**. Para esto, el ejemplo perfecto es la versión móvil y tablet de una misma pantalla (o Activity, para ir entrando en la *jerga*): creamos una única Activity la cual utilizará una distribución de su contenido diferente según el tipo de dispositivo que usemos.



Siempre podemos especificar **un recurso genérico o por defecto.** En contraposición a éste, tendremos la opción de **especificar que una versión concreta de un recurso es para una configuración específica**.

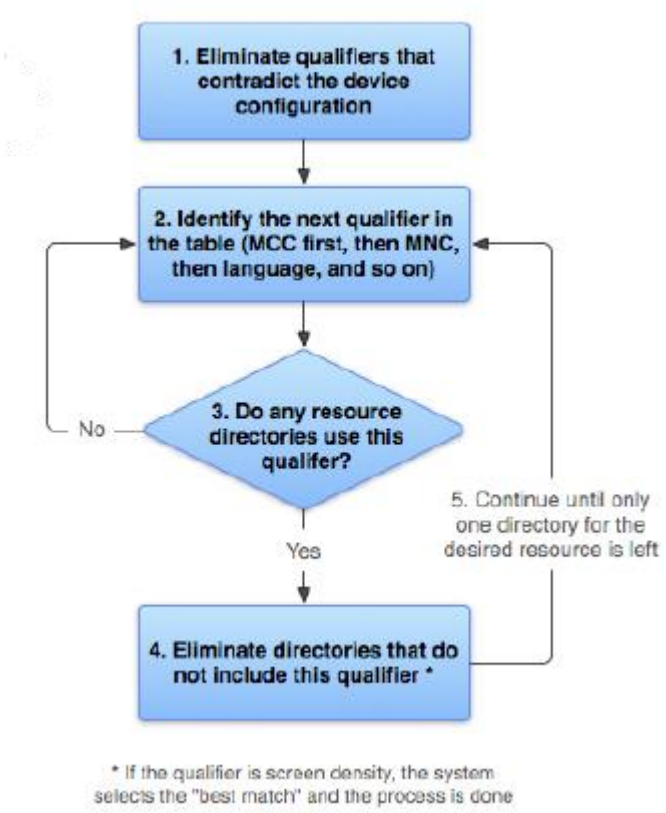
Para detallar la configuración específica podemos basarnos en idiomas, resolución, orientación del dispositivo… Para ello, basta ver las posibilidades en esta **página**. Básicamente radica en **añadir unas terminaciones a las carpetas donde almacenaremos los recursos, acordes a la configuración específica**.

Todos los recursos irán bajo la carpeta ***/res.*** Pero, ¿qué recursos son los que podemos incluir? Los **siguientes**:

- Animaciones
- Colores
- **Imágenes** (*Drawable*)
- **Layouts** (Disposición de elementos gráficos)
- **Menús**
- **Cadenas de texto** (*String*)
- **Estilos**
- **Otros** (booleanos, dimensiones…)

Para ello, deben ir en una **estructura de carpetas específica**, de forma que por ejemplo para añadir cadenas de texto en español utilizaríamos la carpeta /res/values-es o /res/drawable-xxhdpi para *Drawables* para pantallas de alta resolución.

A continuación podéis ver un diagrama de flujo de cómo Android elige el recurso adecuado:



Teniendo claro cómo se gestionan los recursos, **¿cómo creamos algunos recursos específicos?** Veamos a continuación algunos de ellos: layouts, menus y estilos.

Un **layout** define la estructura visual de una interfaz de usuario. A pesar de que podríamos crearla dinámicamente por código, **lo ideal es declarar los elementos de la interfaz en un XML**.

Para crear un layout, disponemos de muchos componentes gráficos ya en la API, aunque podemos crear los nuestros propios. Tenemos layouts donde insertar múltiples componentes, vistas de texto, botones… A continuación, podéis ver un ejemplo de un layout que nos pondrá un texto y justo debajo un botón:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="fill_parent"
              android:layout_height="fill_parent"
              android:orientation="vertical" >
    <TextView android:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello, I am a Button" />
</LinearLayout>
```

En este caso, un LinearLayout nos pondrá elementos uno detrás de otro (en este caso al ser su orientación vertical, uno debajo de otro). A continuación un TextView que de ancho y alto ocupa *lo que necesite (wrap_content)*, con el texto *Hello, I am a TextView*. Y similar para el botón. Cada uno con su identificador único.

Si queremos hacernos buenos a la hora de hacer layouts, lo ideal es que **empecemos trabajando con el editor gráfico de eclipse o Android Studio**, pero vayamos comprobando cómo queda el XML. Conforme pase el tiempo, os daréis cuenta que a veces será más rápido escribir directamente en el XML.

Cuando vamos a definir un **menú** en una de nuestras Actividades, éste también se define a través de un XML. Para más información, os recomiendo visitar el **link**. Aquí os dejo un ejemplo:

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/new_game"
          android:icon="@drawable/ic_new_game"
          android:title="@string/new_game"
          android:showAsAction="ifRoom"/>
    <item android:id="@+id/help"
          android:icon="@drawable/ic_help"
          android:title="@string/help" />
</menu>
```

Por último, cuando hablamos de **estilos**, nos estamos referiendo al concepto más parecido a lo que es CSS para una web: externalizar estilos para poder ser reutilizados. Podremos definir estilos para asignarlos a entidades gráficas, así como crear un **tema**para asignarlo a toda la aplicación.

A continuación podéis ver como mostrar un texto con un formato específico:

```xml
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textColor="#00FF00"
    android:typeface="monospace"
    android:text="@string/hello" />
```

Y como queda tras utilizarlo con estilos, donde el estilo CodeFont podríamos reutilizarlo en otras Views, o si decidiéramos cambiarlo, podríamos cambiarlo a todos a la vez:

```
<TextView
    style="@style/CodeFont"
    android:text="@string/hello" />
```

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="CodeFont" parent="@android:style/TextAppearance.Medium">
        <item name="android:layout_width">fill_parent</item>
        <item name="android:layout_height">wrap_content</item>
        <item name="android:textColor">#00FF00</item>
        <item name="android:typeface">monospace</item>
    </style>
</resources>
```

Llegados este punto, creo que es el momento de mencionaros la **guía de diseño de interfaces para Android**, una web donde podremos ver las tendencias y consejos sobre cómo montar una buena interfaz de usuario.

Por último, para jugar un poco con la asignación de recursos para diferentes configuraciones, estaría bien que sigáis **este ejemplo de Google**.

Con esto damos por terminados los dos primeros conceptos de esta sección. Es cierto que **estos dos conceptos son muy densos, pues son conceptos donde pretendo explicar la estructura de la aplicación**, pero muchos de las cosas que nombramos son las que iremos analizando a posteriori.

¿Preparados para los siguientes conceptos?