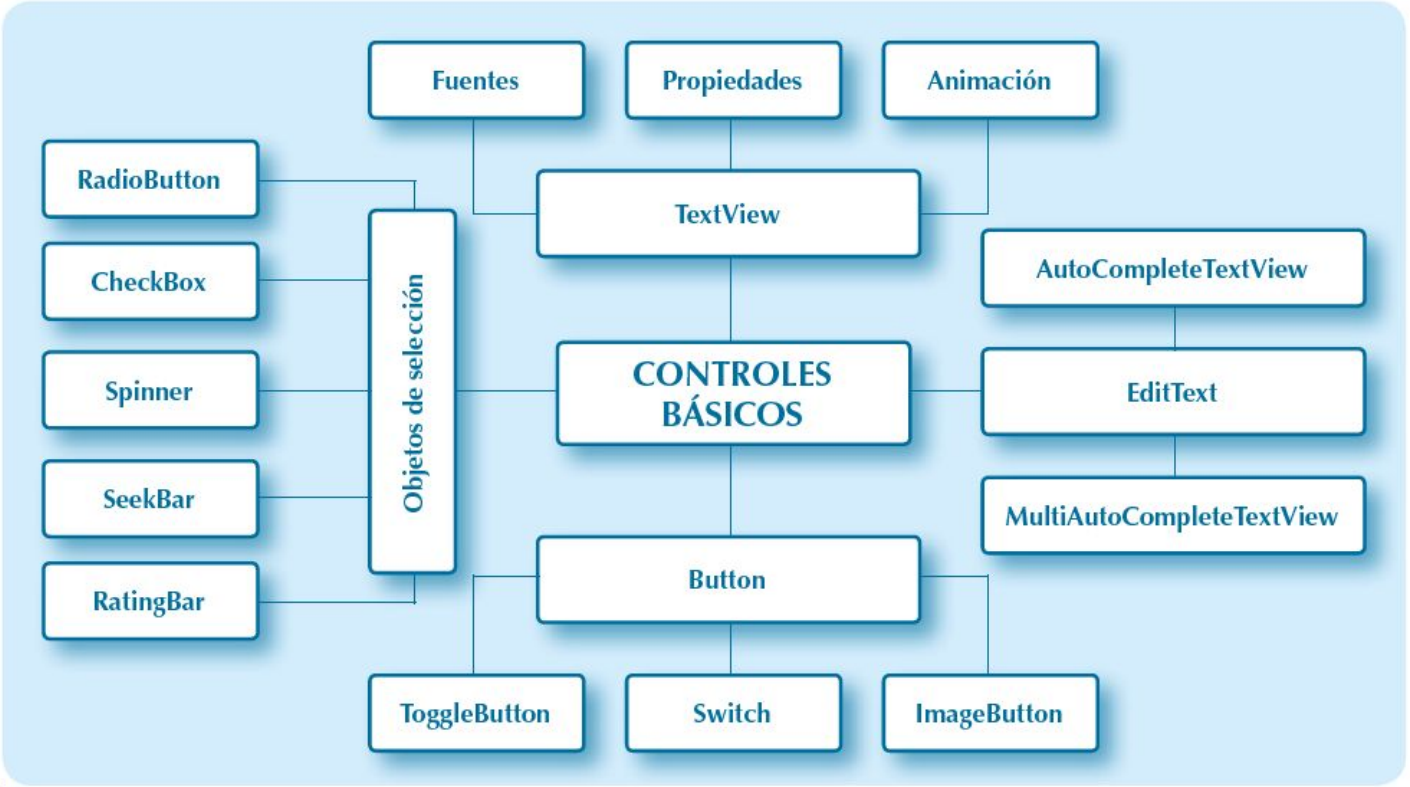
Programación multimedia y dispositivos móviles

UT-2. Controles de entrada

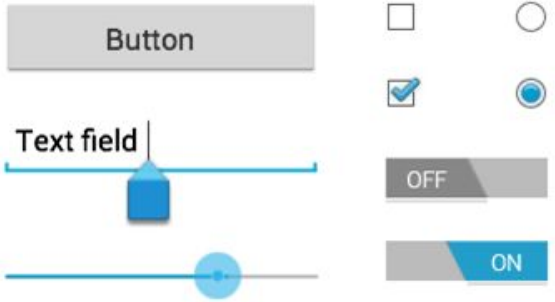
https://developer.android.com/guide/
https://www.sgoliver.net/blog/curso-de-programacion-android/indice-de-contenidos/


Fuentes
Propiedades
Animación
RadioButton
CheckBox
Spinner
SeekBar
RatingBar
Objetos de selección
TextView
AutoCompleteTextView
CONTROLES BÁSICOS
EditText
MultiAutoCompleteTextView
Button
ToggleButton
Switch
ImageButton

Introducción

Los controles de entrada son los componentes interactivos de la interfaz de usuario de tu app. Android ofrece una gran variedad de controles que puedes usar en tu IU, como botones, campos de texto, barras de búsqueda, casillas de verificación, botones de zoom, botones de activación o desactivación y muchos más.

Cada control de entrada admite un conjunto específico de eventos de entrada de modo que puedas gestionar eventos, como cuando el usuario introduce texto o pulsa un botón.



Controles básicos

Tipo de control	Clases relacionadas	Descripción
Botón	Button	Botón que el usuario puede presionar o en el que puede hacer clic para realizar una acción
Campo de texto	EditText AutoCompleteTextView	Campo de texto editable. Puedes usar el control AutoCompleteTextView para permitir al usuario introducir texto con sugerencias de autocompletado
Casilla de verificación	CheckBox	Conmutador de selección y deselección que el usuario puede activar o desactivar. Debes usar casillas de verificación (*checkboxes*) cuando presentes a los usuarios un grupo de opciones seleccionables que no sean mutuamente exclusivas.
Botón de selección	RadioGroup RadioButton	Similar a las casillas de verificación, excepto porque solo se puede seleccionar una opción en el grupo.
Botón para activar o desactivar	ToggleButton	Botón de activación y desactivación con un indicador luminoso.
Control de número	Spinner	Una lista desplegable que permite a los usuarios seleccionar un valor de un conjunto.
Selectores	DatePicker TimePicker	Un cuadro de diálogo para que los usuarios seleccionen un valor individual para un conjunto con los botones de arriba y abajo o mediante un gesto de deslizamiento. Usa un control DatePicker para leer valores de fecha (mes, día, año) o un control TimePicker para leer valores de hora (hora, minuto) en modo 24 horas o AM/PM, a los que se les dará formato automáticamente de acuerdo con la configuración regional del usuario.

Eventos de entrada

De forma general, para añadir lógica a nuestra aplicación a partir de la interacción con el usuario, debemos interceptar los eventos provocados por una interacción del usuario con tu aplicación.

Al considerar los eventos dentro de tu interfaz de usuario, el enfoque consiste en capturar los eventos desde el objeto de vista específico con el que el usuario interactúa. La clase View proporciona los medios para hacerlo.

Eventos de entrada

Dentro de las diversas clases de objetos View que usarás para componer tu diseño, hay varios métodos de retrollamada (callback) públicos que pueden ser útiles para eventos de IU.

La infraestructura (framework) de Android llama a estos métodos cuando la acción respectiva ocurre en el objeto vista. Por ejemplo, cuando el usuario interacciona con un objeto View (por ejemplo, un botón), se produce una llamada al método onTouchEvent() en ese objeto.

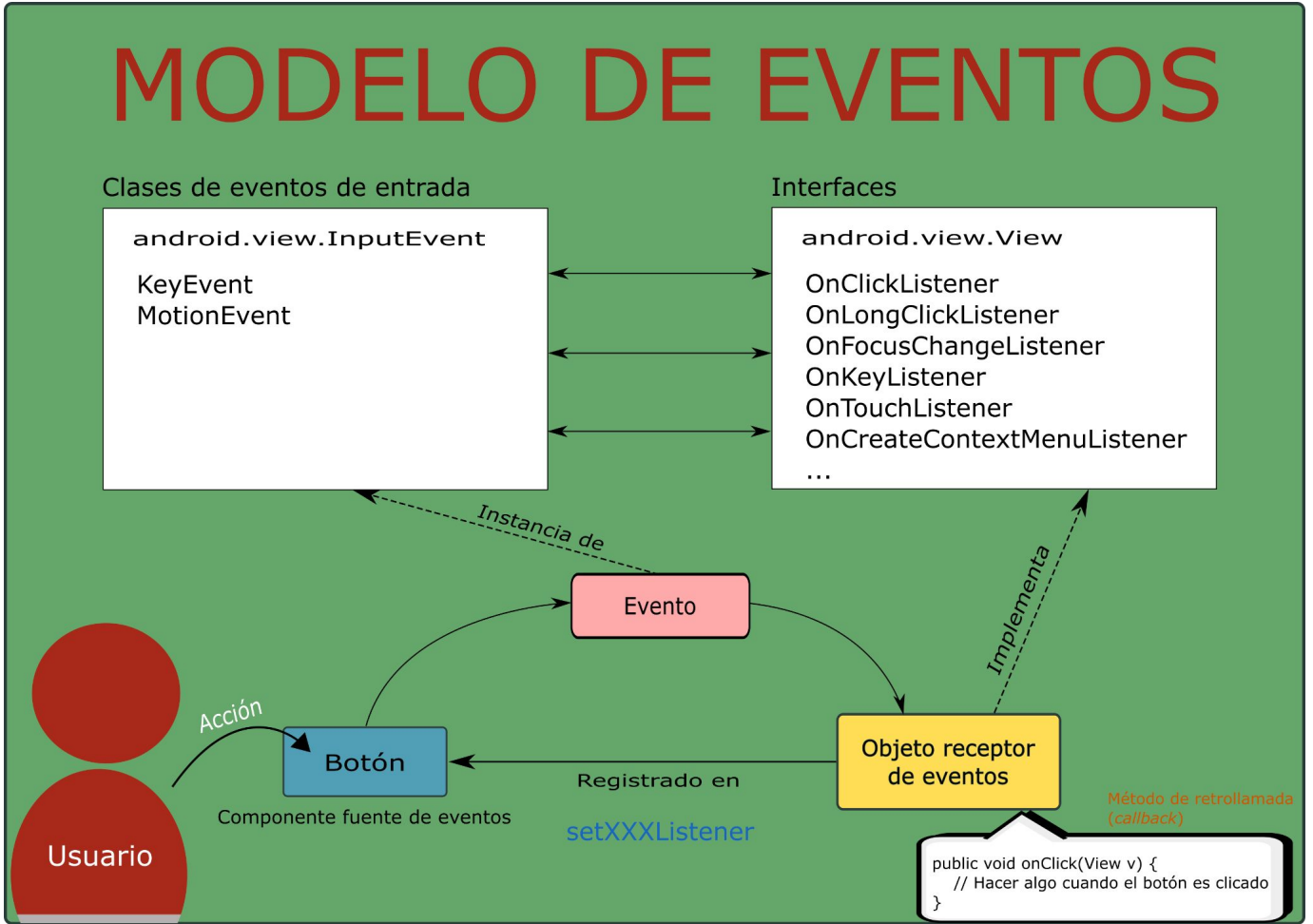
Sin embargo, para interceptar esto, debes extender la clase y sobreescribir el método. No obstante, extender todas las subclases vista para manejar tal evento no sería práctico. Es por ello que la clase de vista también contiene una colección de interfaces anidadas con retrollamadas (callbacks) que puedes definir más fácilmente. Estas interfaces, llamadas receptores de eventos (event listeners), te permiten capturar la interacción del usuario con tu IU.

Eventos de entrada

Un receptor de eventos (event listener) es objeto de una clase que implementa una interface de la clase View la cual contiene un solo método de retrollamada (callback).

Estos métodos serán llamados por la infraestructura (framework) de Android cuando en la vista en la cual se ha registrado el receptor de eventos se produzca un evento provocado por una interacción del usuario con un elemento de esa vista (en la IU).

Eventos de entrada



Eventos de entrada

En dichas interfaces, se incluyen los siguientes métodos de retrollamada (callback):

- **onClick**(): de View.OnClickListener. El método de retrollamada onClick es llamado cuando el usuario toca el elemento (en el modo táctil), o selecciona el elemento con las teclas de navegación o la bola de seguimiento y presiona la tecla “Entrar” adecuada o la bola de seguimiento (trackball).
- **onLongClick**(): de View.OnLongClickListener. El método de retrollamada onLongClick es llamado cuando el usuario toca y mantiene presionado el elemento (en el modo táctil), o selecciona el elemento con las teclas de navegación o la bola de seguimiento y mantiene presionada la tecla “Entrar” adecuada o la bola de seguimiento (durante un segundo).
- **onFocusChange**(): de View.OnFocusChangeListener. El método de retrollamada onFocusChange es llamado cuando el usuario navega hacia el elemento o sale de este utilizando las teclas de navegación o la bola de seguimiento.

Eventos de entrada

- **onKey**(): de View.OnKeyListener. El método de retrollamada onKey es llamado cuando el foco de entrada está situado en el elemento y presiona o suelta una tecla física en el dispositivo.
- **onTouch**(): de View.OnTouchListener. El método de retrollamada onTouch es llamado cuando el usuario realiza una acción calificada como un evento táctil como presionar, soltar o cualquier gesto de movimiento en la pantalla (dentro de los límites del elemento).
- **onCreateContextMenu**(): de View.OnCreateContextMenuListener. El método de retrollamada onCreateContextMenu es llamado cuando se crea un menú contextual (como resultado de un "clic largo" sostenido).

Ten en cuenta que el método de retrollamada (callback) onClick() no tiene valor de retorno, pero algunos otros métodos de receptores de eventos deben devolver un valor lógico (de tipo boolean). Cada caso dependerá del tipo de evento. A continuación, se explican los motivos de los pocos casos que lo hacen:

Eventos de entrada

- **onLongClick**(): este método devuelve un valor booleano para indicar si has consumido el evento y si no debe continuar. Es decir, devuelve true para indicar que has usado el evento y que debe detenerse aquí; devuelve false si no has usado el evento, o si el evento debe continuar para otros receptores de clic.
- **onKey**(): este método devuelve un valor booleano para indicar si has consumido el evento y si no debe continuar. Es decir, devuelve true para indicar que has usado el evento y que debe detenerse aquí; devuelve false si no has usado el evento, o si el evento debe continuar para otros receptores de tecla.
- **onTouch**(): este método devuelve un valor booleano para indicar si tu receptor consume este evento. Lo importante es que este evento puede tener múltiples acciones una después de la otra. Por lo tanto, si devuelve false cuando se recibe el evento de acción de abajo, tú indicas que no has consumido el evento y también que no estás interesado en las acciones subsiguientes de este evento. Por ende, no se te llamará para otras acciones dentro del evento, como un gesto del dedo, o el evento de acción de arriba final.

Controladores de eventos

Los controladores/gestores de eventos (*event handlers*) son los métodos de retrollamada (callback) que son invocados por los receptores de eventos (*event listeners*) cuando se produce un evento asociado al receptor de eventos que se registró en el componente fuente del evento.

Por ejemplo, **onClick**() es el método de retrollamada que controla/gestiona el evento de clicado en un componente como por ejemplo un botón. Dicho botón es el componente fuente del evento.

Recordemos que los componentes con los cuales puede interactuar el usuario en la interfaz gráfica se denominan controles (widgets). El botón es un control (widget).

Controladores de eventos

En el modelo de eventos vimos como el proceso de generación, recepción y gestión de un evento es el siguiente:

1) Registro del receptor de eventos en el componente fuente
2) Interacción del usuario
3) Generación del evento
4) Envío del evento al receptor de eventos (event listener)
5) Ejecución del método de retrollamada (callback) <----- Control/gestión del evento

En el paso 5 se ejecuta alguna acción como reacción al evento que se ha producido en el componente fuente del evento ante la interacción del usuario.

Sobre los eventos que se pueden capturar, tenemos abundante documentación en la página oficial de desarrollo sobre Android Studio: https://developer.android.com/guide/topics/ui/ui-events?hl=es-419

Interfaz de usuario en Android: Controles básicos

En este primer apartado nos vamos a centrar en los diferentes tipos de botones y cómo podemos personalizarlos. El SDK de Android nos proporciona diferentes tipos de botones que iremos viendo: los clásicos de texto (**Button**), los que pueden contener una imagen (**ImageButton**), y los de tipo on/off (**ToggleButton** y **Switch**).

No vamos a comentar mucho sobre ellos dado que son controles similares a los que hay en otros IDEs, ni vamos a enumerar todas sus propiedades porque existen decenas. A modo de referencia, a medida que los vayamos comentando iré poniendo enlaces a su página de la documentación oficial de Android para poder consultar todas sus propiedades en caso de necesidad.

Interfaz de usuario en Android: Controles básicos

- **Button**: en control de tipo Button es el botón más básico que podemos utilizar y normalmente contiene un simple texto. Definimos el texto del botón asignando su propiedad "**text**". Además de esta propiedad podríamos utilizar muchas otras como el color de fondo (**background**), estilo de fuente (**typeface**), color de fuente (**textColor**), tamaño de fuente (**textSize**), etc.
- **ToggleButton**: es un tipo de botón que puede permanecer en dos posibles estados, pulsado o no pulsado. En este caso, en vez de definir un sólo texto para el control definiremos dos, dependiendo de su estado. Así, podremos asignar las propiedades "**textOn**" y "**textoOff**" para definir ambos textos.

Interfaz de usuario en Android: Controles básicos

- **Switch**: un control Switch es muy similar al ToggleButton anterior, donde tan sólo cambia su aspecto visual, que en vez de mostrar un estado u otro sobre el mismo espacio, se muestra en forma de deslizador o interruptor. Además de las propiedades "**textOn**" y "**textoOff**", también podemos asignar un texto (**Text**) o por ejemplo la propiedad "**switchPadding**" para definir el espacio entre el texto del control y su representación visual.

Tanto para los Switch como para los Toogles, además de tener los eventos del resto de controles, hay una propiedad importante (y que luego también utilizaremos en checks y radio botonoes): IsChecked

```
if (miSwitch.isChecked) miTexto.setTextColor(Color.parseColor( colorString: "#0000ff"))
```

Interfaz de usuario en Android: Controles básicos

- **ImageButton**: En un control de tipo ImageButton podremos definir una imagen a mostrar en vez de un texto, para lo que deberemos asignar la propiedad "**srcCompat**". Normalmente asignaremos esta propiedad con el descriptor de algún recurso que hayamos incluido en las carpetas /res/drawable. Adicionalmente, al tratarse de un control de tipo imagen también deberíamos acostumbrarnos a asignar la propiedad "**contentDescription**" con una descripción textual de la imagen, de forma que nuestra aplicación sea lo más accesible posible. Otra propiedad que podemos asignar, también común a casi todos los tipos de botones, sería "**padding**" para definir el espacio entre el borde del botón y su contenido.

Si tu imagen es más "grande" que tu imagebutton puedes ajustarla usando la propiedad scaleType = centerInside.

Interfaz de usuario en Android: Controles básicos

Cabe decir además, que aunque existe este tipo específico de botón para imágenes, también es posible añadir una imagen a un botón normal de tipo Button, a modo de elemento suplementario al texto (compound drawable). Por ejemplo, si quisiéramos añadir un icono a la izquierda del texto de un botón utilizamos la propiedad "**drawableLeft**" y si fuera necesario podríamos indicar también el espacio entre la imagen y el texto mediante la propiedad "**drawablePadding**".

En el caso de un botón de tipo **ToggleButton** o **Switch** suele ser de utilidad reaccionar a un evento que indica que se ha cambiado el valor. Para lo que implementaremos dicho evento llamando al método setOnCheckedChangeListener(). Este evento recibe dos parámetros, el primero de ellos una referencia al propio botón que se ha pulsado y el segundo (isChecked) que indica el estado resultante del botón.

Interfaz de usuario en Android: Controles básicos

- **TextView**: el control TextView es otro de los clásicos y se utiliza para mostrar un determinado texto al usuario. Al igual que en el caso de los botones, el texto del control se establece mediante la propiedad "**text**". A parte de esta propiedad, la naturaleza del control hace que las más interesantes sean las que establecen el formato del texto mostrado, que al igual que en el caso de los botones son las siguientes: "**background**" (color de fondo), "**textColor**"(color del texto), "**textSize**" (tamaño de la fuente), "**typeface**" (fuente) y "**textStyle**" (estilo del texto: normal, negrita, cursiva).

  De igual forma, también podemos manipular estas propiedades desde nuestro código. Como ejemplo le podemos cambiar su color de fondo con "**setBackgroundColor()**"

Interfaz de usuario en Android: Controles básicos

- **EditText**: el control EditText es el componente de edición de texto que proporciona la plataforma Android. Permite la introducción y edición de texto por parte del usuario, por lo que en tiempo de diseño la propiedad más interesante a establecer, además de su posición/tamaño y formato, es el texto a mostrar, atributo "**text**". Por supuesto si no queremos que el cuadro de texto aparezca inicializado con ningún texto, no es necesario incluir esta propiedad. Lo que sí deberemos establecer será la propiedad "**inputType**". Esta propiedad indica el tipo de contenido que se va a introducir en el cuadro de texto, como por ejemplo una dirección de correo electrónico (textEmailAddress), un número genérico (number), un número de teléfono (phone), una dirección web (textUri), o un texto genérico (text).

  El valor que establezcamos para esta propiedad tendrá además efecto en el tipo de teclado que mostrará Android para editar dicho campo. Así, por ejemplo, si hemos indicado "text" mostrará el teclado completo alfanumérico, si hemos indicado "phone" mostrará el teclado numérico del teléfono, etc.

Interfaz de usuario en Android: Controles básicos

Al igual que ocurría con los botones, donde podíamos indicar una imagen que acompañara al texto del mismo, con los controles de texto podemos hacer lo mismo. Las propiedades "**drawableLeft**" o "**drawableRight**" nos permite especificar una imagen, a izquierda o derecha, que permanecerá fija en el cuadro de texto.

Otra opción adicional será indicar un texto de ayuda o descripción (**hint**), que aparecerá en el cuadro de texto mientras el usuario no haya escrito nada (en cuanto se escribe algo este texto desaparece). Con la propiedad "**textColorHint**" podemos indicar su color.

Interfaz de usuario en Android: Controles básicos (II)

Atributos de posicionamiento	layout_width	ancho
	layout_height	alto
Atributos para los márgenes	layout_margin	cuatro márgenes
	layout_marginBotton	margen inferior
	layout_marginLeft	margen izquierdo
	layout_marginRight	margen derecho
	layout_marginTop	margen superior
Atributos para el espaciado	android:padding	espaciado a los cuatro lados
	android:paddingTop	espaciado superior
	android:paddingBottom	espaciado inferior
	android:paddingLeft	espaciado izquierdo
	android:paddingRight	espaciado derecho

Interfaz de usuario en Android: Controles básicos (II)

Dentro de los textos podemos encontrar el **TextInputLayout**. En realidad añades un Layout y dentro un control de texto. Este control responde a una especificación de Material Design para en los campos de texto recoger la utilización de las denominadas «etiquetas flotantes» (floating labels). Este elemento de usabilidad consiste en ubicar las etiquetas de los campos de texto dentro del mismo para posteriormente mostrarlas en la parte superior del campo mientras este tenga el foco o bien contenga algún valor.

El objetivo es optimizar el espacio vertical de la pantalla al mostrarse la etiqueta dentro del campo del texto pero evitar a la vez que el usuario pierda la noción del contenido que corresponde al campo del texto. Se puede marcar la propiedad de “password” para que lo trate como un campo en el que se introduce la password.

Para poder trabajar con este control (**TextInputLayout)** en la parte programática, basta simplemente con castear el control del texto con un objeto EditText y utilizarlo como cualquier EditText.

Depuración en Android: Logging

A la hora de depurar Android, como es habitual en estos frameworks y vimos en la primera unidad, ofrece la posibilidad de ejecutar la aplicación en modo debug, poniendo puntos de parada y continuar paso a paso, o saltar hasta el siguiente punto,....

Una de las técnicas más útiles a la hora de depurar y/o realizar el seguimiento de aplicaciones sobre cualquier plataforma es la creación de logs de ejecución. Android por supuesto no se queda atrás y nos proporciona también su propio servicio y API de logging a través de la clase android.util.Log.

En Android, todos los mensajes de log llevarán asociada la siguiente información:

- Fecha/Hora del mensaje.
- Criticidad. Nivel de gravedad del mensaje (se detalla más adelante).
- PID. Código interno del proceso que ha introducido el mensaje.
- Tag. Etiqueta identificativa del mensaje (se detalla más adelante).
- Mensaje. El texto completo del mensaje.

Depuración en Android: Logging

De forma similar a como ocurre con otros frameworks de logging, en Android los mensajes de log se van a clasificar por su criticidad, existiendo así varias categorias (ordenadas de mayor a menor criticidad):

- ➔ Error
- ➔ Warning
- ➔ Info
- ➔ Debug
- ➔ Verbose

Para cada uno de estos tipos de mensaje existe un método estático independiente que permite añadirlo al log de la aplicación. Así, para cada una de las categorías anteriores tenemos disponibles los métodos e(), w(), i(), d() y v() respectivamente.

Depuración en Android: Logging

Cada uno de estos métodos recibe como parámetros la etiqueta (tag) y el texto en sí del mensaje.

Como etiqueta de los mensajes, aunque es un campo al que podemos pasar cualquier valor, suele utilizarse el nombre de la aplicación o de la actividad concreta que genera el mensaje. Esto nos permitirá identificar y poder visualizar únicamente los mensajes de log que nos interesan, entre todos los generados por Android [que son muchos] durante la ejecución de la aplicación.

- Log.e(LOGTAG, "Mensaje de error");
- Log.w(LOGTAG, "Mensaje de warning");
- Log.i(LOGTAG, "Mensaje de información");
- Log.d(LOGTAG, "Mensaje de depuración");
- Log.v(LOGTAG, "Mensaje de verbose");