

Material Design



Contenidos

Proyecto	3
Introducción	3
Material Design	3
Vistas incorporadas con Material Design en la API 21	4
Paleta de colores	5
Colores.....	5
Temas y estilos	7
Uso de estilos	7
Uso de temas.....	9
Temas del sistema.....	9
Editor de temas en AS	10
Recursos del sistema	10
Dimensiones	11
Práctica	11

Proyecto

Crea en AS un proyecto llamado P_06_Material_01 que sea para dispositivo con Android como mínimo 5.0 y que esté basado en la plantilla Basic Activity. Conecta un dispositivo (real o emulado) e instala la aplicación.



Introducción

Al usar la plantilla Basic Activity, ya hemos comentado en un ejercicio propuesto anteriormente, que el asistente nos ayuda a que la aplicación tenga un diseño inicial basado en Material Design, de tal manera que nos proporciona:

- dos ficheros en res/layout: activity_main.xml y content_main.xml
- las vistas parece que tienen más propiedades configurables
- además en la sección dependencies del fichero build.gradle del módulo se han incluido la librería necesaria para poder utilizar los componentes recomendados en "Material Design":

Plantilla Empty	Plantilla Basic
<pre>dependencies { implementation fileTree(dir: 'libs', include: ['*.jar']) implementation 'androidx.appcompat:appcompat:1.1.0' implementation 'androidx.constraintlayout:constraintlayout:1.1.3' testImplementation 'junit:junit:4.12' androidTestImplementation 'androidx.test:runner:1.2.0' androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0' }</pre>	<pre>dependencies { implementation fileTree(dir: 'libs', include: ['*.jar']) implementation 'androidx.appcompat:appcompat:1.1.0' implementation 'androidx.constraintlayout:constraintlayout:1.1.3' implementation 'com.google.android.material:material:1.0.0' testImplementation 'junit:junit:4.12' androidTestImplementation 'androidx.test:runner:1.2.0' androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0' }</pre>

Las librerías androidx.appcompat y androidx.constraintlayout que aparecen en ambas plantillas ya hemos comentado que nos van a permitir usar las nuevas clases en dispositivos anteriores a su aparición (la mayoría han sido introducidas en la versión 5.0 de Android).

Material Design

A partir de la versión 5.0 de Android (API 21), se introduce Material Design. Es una guía de Google enfocada al diseño utilizado en Android, pero también en la web y en cualquier plataforma.

Recibe su nombre por estar basado en objetos materiales. La idea es que la UI parezca que está construida de material físico que ocupa un espacio en un tiempo determinado y donde las animaciones de las piezas son lógicas, los objetos se superponen pero no puedan atravesarse el uno al otro,...

Material Design es un lenguaje visual que sintetiza los principios clásicos del buen diseño con la innovación de la tecnología y la ciencia.

¿Cómo se traslada esto a Android? Pues básicamente delimitando claramente el tipo de menús, los botones, los colores y los tipos de imágenes a elegir:

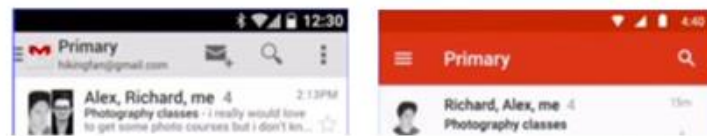
- Elementos ordenados e imágenes claras: Material Design es un diseño con una tipografía clara, casillas bien ordenadas, colores e imágenes llamativos para no perder el foco.
- La luz y las sombras dan sensación de jerarquía y profundidad.
- El movimiento es la mejor forma de guiar al usuario, por ejemplo un objeto que parpadea significa que está llamando tu atención, un elemento que se expande es que se acaba de abrir.

Paleta de colores

Uno de los principios que se definen en Material Design es el uso del color. Google propone usar colores vivos y alegres.

Cada aplicación tiene que definir su propia paleta de colores que la diferencie del resto de aplicaciones. Incluso la barra de estado de Android cambiará para que combine con los colores de la aplicación.

Puedes observar en la imagen inferior cómo ha cambiado el uso del color en la aplicación Gmail. A la izquierda se muestra en una versión 4.x y a la derecha utilizando Material Design. Esta aplicación ha decidido usar una tonalidad rojiza como color primario o característico. Observa como la barra de estado también se muestra en un color similar algo más oscuro.



Por lo tanto, si tu aplicación va a seguir las especificaciones de Material Design (que es lo deseable), lo primero que has de hacer es escoger la paleta de colores que va a utilizar.

En la web de Material Design se nos proponen [algunas paletas de ejemplo](#).

Colores

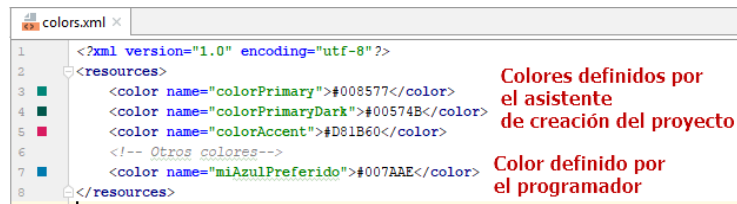
Debemos escoger muy bien, al menos, dos colores:

- **colorPrimary** (el color característico de la aplicación) y sus derivados Dark (más oscuro) y Light (más claro) de la misma paleta. Utilizados como fondo de la toolbar, de la barra de estado, en los efectos de borde, etc.
- **colorAccent** (el color de realce o secundario): es recomendable que sea de otra paleta distinta a la escogida para Primary. Aplicado a elementos flotantes, edición de texto, enlaces, botones, elementos de selección, ProgressBar, etc.



- Existen otros colores que proporcionan un control más fino sobre ciertas vistas: colorControlNormal, colorControlActivated, colorControlHighlight, colorButtonNormal, colorSwitchThumbNormal, colorEdgeEffect, statusBarColor y navigationBarColor.

Los nombres de estos colores con sus valores se guardan en el fichero de recursos res/values/colors.xml. Podremos también añadir otros colores y valores:



Para definir los colores de nuestra aplicación (y que los colores derivados sean los correctos "según Google y su guía de diseño") puedes usar la herramienta de Google [Color Tool](#) o utilizar la herramienta [Material Palette](#); ambas son intuitivas y permiten descargar ficheros en xml con los recursos de colores elegidos.

Aunque en la segunda herramienta web citada solo se nos permite seleccionar el color primario (colorPrimary) y el de resalte o secundario (colorAccent), realmente nos suministran 8 colores para la paleta de la aplicación:

- primaryDarkColor y primaryLightColor son los colores derivados del primario (los que combinan bien!).
- El resto se utilizan para textos e iconos y no es recomendable modificar los propuestos (salvo que seas un poco "hortera" o muy artista 😊), Google recomienda "jugar" en ellos con la opacidad en vez de con el color, por eso son propuestos siempre blancos/negros/grises.

Puedes realizar varias pruebas hasta obtener unos colores de tu gusto. Cuando los tengas, puedes descargar un fichero XML de recursos Android donde se definen estos colores.



Pero no te lo aconsejo, porque los nombres de los colores no coinciden exactamente con los que usa AS y además como solo vamos a cambiar tres valores, va a ser más



sencillo que abras el fichero res/values/colors.xml de la aplicación y reemplaces lo tres valores definidos, con los valores en hexadecimal que has seleccionado (la segunda herramienta te ayuda en Firefox: al picar en el nombre del color, se copia su valor al portapapeles).

Observa como cambia la previsualización de activity_main.xml en el editor de layouts o si ejecutas la aplicación.

Por qué el simple hecho de definir unos colores ha logrado ese cambio? Porque dichos colores aparecen en el tema de la aplicación.



Temas y estilos

Si abres el fichero `res/values/styles.xml` podrás observar como estos tres colores son utilizados para configurar los colores del tema aplicado por defecto a tu aplicación:

```
<resources>

<!-- Base application theme. -->
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
    <!-- Customize your theme here. -->
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>
</style>
...
```

El resto de colores de la paleta no son definidos ya que se utilizarán los colores por defecto (los blancos/grises/negros ya citados).

Si abres el fichero `AndroidManifest.xml` podrás observar como este tema es asignado a la aplicación.

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportRtl="true"
    android:theme="@style/AppTheme">
    ...
</application>
```

Pero, ¿qué son los estilos y temas? Habrás advertido grandes similitudes entre HTML y el diseño de layouts. En los dos casos se utiliza un lenguaje de marcado y se trata de crear diseños independientes del tamaño de la pantalla donde se visualizarán. En el diseño web resultan clave las hojas de estilo en cascada (CSS), que permiten crear un patrón de diseño y aplicarlo a varias páginas. Cuando diseñas los layouts de tu aplicación, vas a poder utilizar unas herramientas similares conocidas como estilos y temas.

Un estilo es una colección de propiedades que especifican el aspecto y el formato que puede tener una Vista en concreto.

Un estilo puede definir propiedades tales como la altura, relleno, color y tamaño de la fuente, color de fondo, etc. y resultan útiles definirlos para usarlos en vistas distintas (concepto similar a los CSS del diseño web).

Un tema es un estilo aplicado a toda una actividad o aplicación, en lugar de a una vista individual.

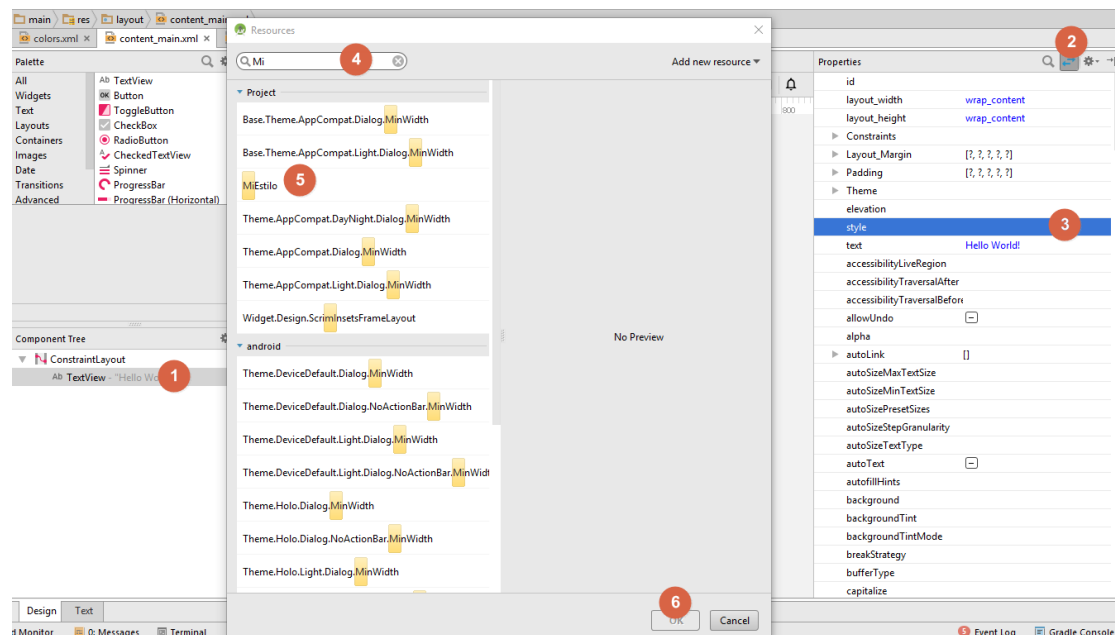
Ambos se definen en el fichero `res/values/styles.xml` y no tienen que referirse solo a colores

Uso de estilos

Imagina que muchos de los `TextView` de los layouts de tu aplicación, tienen el texto en negrita y de color verde. Podríamos fijar esas características en cada una de las vistas que lo necesiten pero es más rápido crear un estilo, dándole un nombre y fijando mediante ítem las citadas propiedades que se repiten:

```
<style name="MiEstilo">
    <item name="android:textStyle">bold</item>
    <item name="android:textColor">@color/miVerdePreferido</item>
</style>
```

A partir de ese momento podemos aplicarlo en las vistas que lo necesitemos:



Un estilo puede heredar propiedades de un padre y a partir de ellas realizar las modificaciones necesarias.

Podemos heredar de estilos que nosotros mismos hayamos creado o de los estilos incluidos en la plataforma.

- Herencia de estilo propio

Utilizamos el nombre del estilo del que queremos heredar como prefijo al nombre que definimos en el atributo name del estilo que estamos creando, separando el prefijo con un punto.

Ejemplo:

```
<style name="MiEstilo.grande">
    <item name="android:textSize">50sp</item>
</style>
```

El nuevo estilo sería igual a MiEstilo más la nueva propiedad indicada. Si quisiéramos hacer referencia a este estilo en alguno de nuestros controles lo haríamos a través de la sintaxis `style="@style/MiEstilo.grande"`.

Es posible también realizar una cadena de herencias. Ejemplo:

```
<style name="MiEstilo.grande.otroMas">
    <item name="android:background">@color/colorAccent</item>
</style>
```

- Herencia de estilo de la plataforma Android

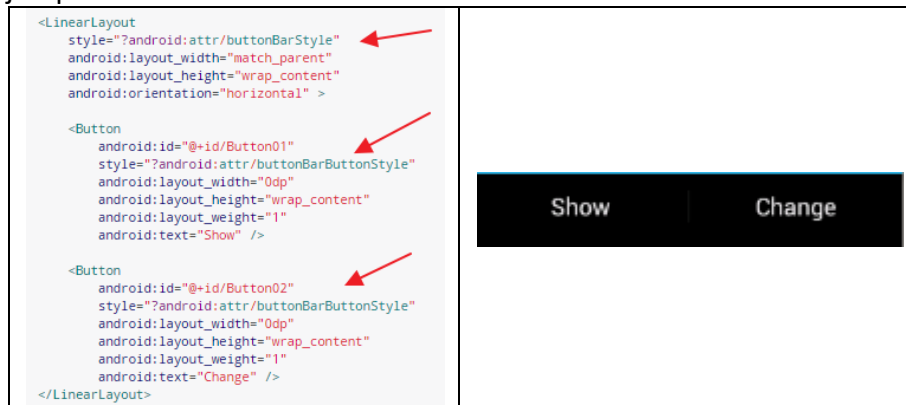
La plataforma Android ofrece una gran colección de estilos y temas que podemos utilizar con el prefijo `@android` en el atributo parent. La lista de estilos disponibles está en [R.style.html](https://developer.android.com/reference/android/R.style.html)

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="MiEstilo"
        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">#00AAFF</item>
        <item name="android:typeface">sans</item>
    </style>
</resources>
```

- La lista de valores de atributos estándar del tema está fijada por la plataforma en [R.attr.htm](https://developer.android.com/reference/android/R.attr.html) y para acceder a un determinado valor se usa `?android:attr/notación-atributo`.

Ejemplo 1.: Usar el valor `?android:attr/listPreferredItemHeight` significa "utilizar el valor definido para el atributo llamado `listPreferredItemHeight` en el tema actual".

Ejemplo 2:



Uso de temas

Para aplicar un tema a toda una aplicación se edita el fichero `AndroidManifest.xml` y se modifica el parámetro `android:theme` en la etiqueta `application`:

`<application android:theme="@style/MiTema">`

También se puede aplicar un tema a una actividad en concreto:

`<activity ... android:theme="@style/MiTema">`

o

`<activity ... android:theme="@android:style/Theme.Translucent">`

Tema propio

Tema de la plataforma

Temas del sistema

La plataforma cuenta con temas propios de los que debemos heredar en nuestros temas propios.

Es recomendable usar los temas de Material Design, ya que es el tema por defecto desde la API 21, permiten aplicar un tono consistente a la app y pueden escogerse entre temas claro y oscuro:

- `@android:style/Theme.Material` (fondo oscuro)
- `@android:style/Theme.Material.Light` (fondo o `windowBackground` claro)
- `@android:style/Theme.Material.Light.DarkActionBar` (fondo claro con barra oscura, es el que aplica AS en sus plantillas)

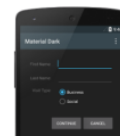


Figure 1: Dark material theme

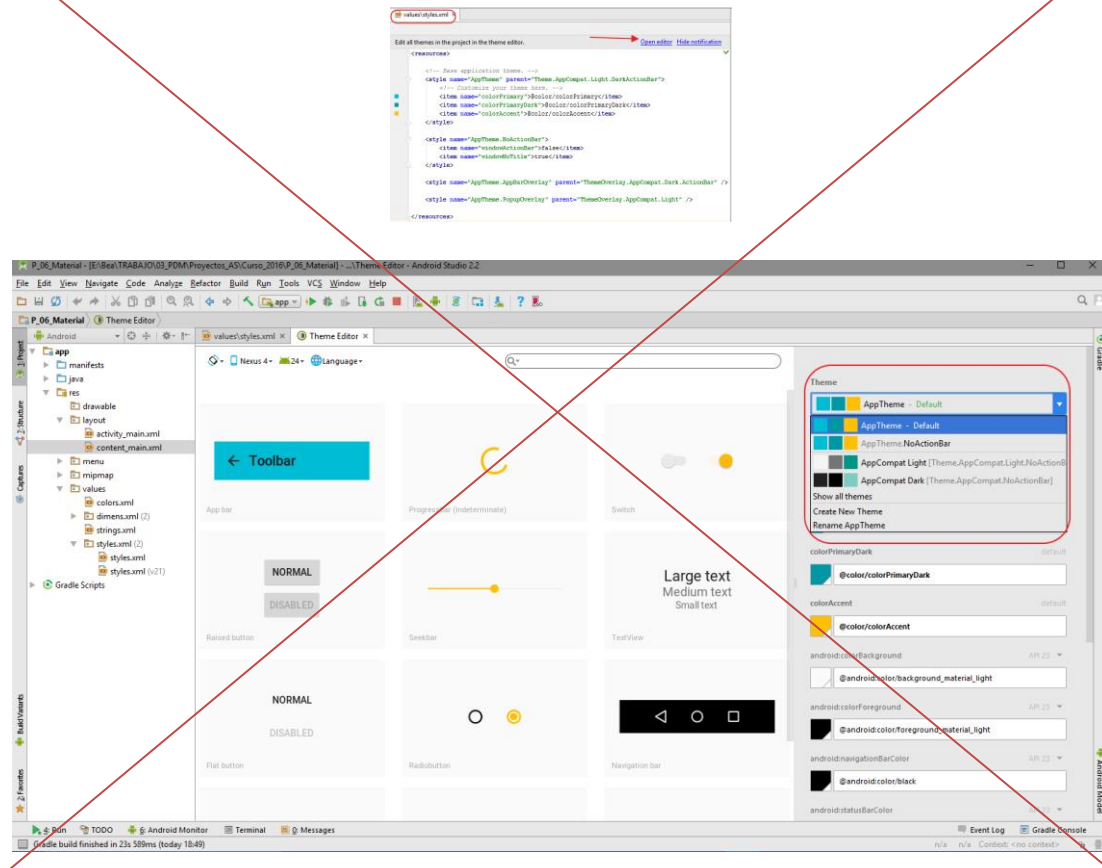


Figure 2: Light material theme

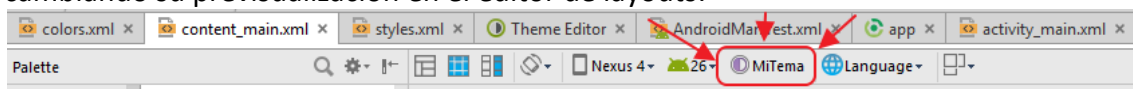
Para que nuestros diseños puedan usar las nuevas características necesitamos la librería de soporte de diseño y que las actividades sean descendiente de `AppCompatActivity`, por eso, tenemos que de aplicar siempre un tema que sea descendiente de `Theme.AppCompat`.

Editor de temas en AS **Desaparecido en AS 3.5**

Definir correctamente los temas puede llegar a ser un trabajo muy laborioso, nuestro IDE AS nos lo facilita con el editor de temas:



Si se aplica un tema a una actividad, podemos probar su uso ejecutándola o cambiando su previsualización en el editor de layouts:




Recursos del sistema

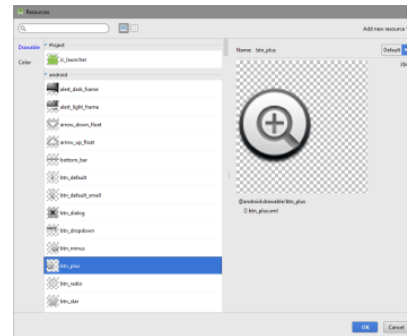
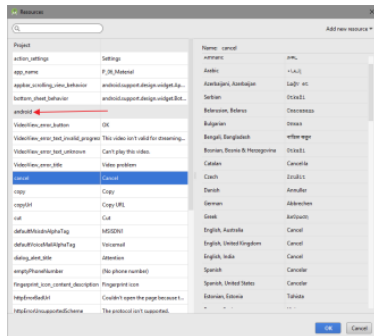
Además de los recursos que podamos añadir a nuestra aplicación, también podemos utilizar una serie de recursos que han sido incluidos en el sistema (ya hemos hablado de estilos del sistema!).

Para acceder a ellos haremos uso de la clase **android.R**

Usar recursos del sistema tiene muchas ventajas:

- No consumen memoria en nuestra aplicación, al estar ya incorporados al sistema.
- Los hay de tipo string, drawable, style, etc.
- Además los usuarios están familiarizados con ellos: Por ejemplo, si utilizamos el recurso **android.R.drawable.ic_menu_edit** se mostrará al usuario el icono . Muy posiblemente el usuario ya lo asocie a la acción de editar.
- Se adaptan a las diferentes versiones de Android: El icono citado es ligeramente diferente en algunas versiones, pero al usar un recurso del sistema nos aseguramos que se mostrará el adecuado a la versión del usuario.

- Se adaptan siempre a las configuraciones locales. Si yo utilizo el recurso **android.R.string.cancel** este será "Cancelar", "Cancel", "取消",... según el idioma escogido por el usuario.
- AS nos proporciona asistentes para acceder a ellos desde el Editor de Layouts:



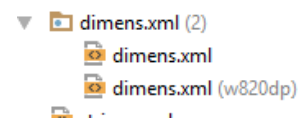
- Existen apps descargables de Play Store como .R o R.Explorer que permiten ver los recursos del sistema en el dispositivo en el que se instalen.

Dimensiones

Del mismo modo como se definen strings, colores, estilos, etc. se definen dimensiones en el fichero `res/values/dimens.xml`. Se pueden definir varias dimensiones, como márgenes, rellenos, altura, anchura, radio, tamaño del texto, etc.

La unidad más recomendable es "dp" (pixel independiente de la densidad, a unidad abstracta basada en la densidad física de una pantalla. [Buena explicación](#))

Una forma de implementar correctamente los márgenes de las vistas es utilizar `res/values/dimens.xml` y definir una dimensión de los márgenes. Posteriormente usar recursos alternativos para el mismo archivo para definir diferentes márgenes para dispositivos más/menos grandes.



Práctica

Realiza los ejercicios propuestos en Ejercicios_02_Material