

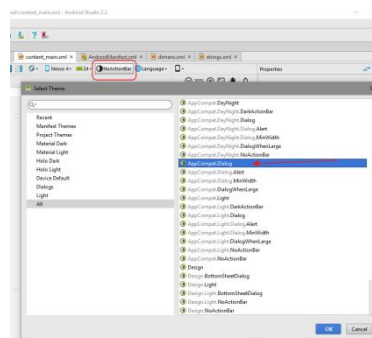


Ejemplo 1: Uso de Material Design

1. Crea un nuevo proyecto con nombre P_07_Lugares_01 con plantilla Scrolling Activity. La actividad principal debe llamarse MainActivity (por conservar el convenio utilizado hasta ahora).
2. Observa que la plantilla sigue estando basada en Material Design y que aparece un mensaje  Fidelity warnings [\[show\]](#) que nos advierte que lo mostrado puede no parecerse a la realidad.
3. Es debido a que estamos usando un tipo de layout propio de Material Design, el NestedScrollView para que el contenido principal pueda desplazarse. **Ejecuta la aplicación** para comprobarlo y observa la animación al desplazar el texto (el botón flotante desaparece y la toolbar se hace más pequeña!)
4. Reemplazar el TextView por un LinearLayout que contenga cuatro Button. Un NestedScrollView solo puede contener dentro un elemento, por lo que no puedes introducir directamente los cuatro botones. Usando un layout que los contenga se resuelve el problema.

	<p>Debes crear recursos en el proyecto:</p> <ul style="list-style-type: none"> • Para los textos de los botones • Para la separación de los botones
--	---

5. Ejecuta la aplicación y cambia la orientación del dispositivo. Podemos ver los botones que no "caben" al usar el desplazamiento, pero es un mal diseño porque no es intuitivo para el usuario dicho desplazamiento.
6. Aplica a la actividad (desde el fichero AndroidManifest.xml), el tema `@style/Theme.AppCompat.Dialog`. Vuelve a ejecutar. Este tema es utilizado en cuadros de diálogo. No parece muy adecuado para nuestra actividad.
7. Observa que el Editor de Layout, la previsualización no se ha cambiado sola, debemos hacerlo (además no nos deja escoger para previsualizar cualquier tema, debe heredar de AppCompatActivity!)



8. Deshaz el cambio realizado en el Manifest.

Ejercicio 2: Estilos

1. En el proyecto P_05_Creciente_01, crea un nuevo estilo con nombre MiEstilo que defina la propiedad color del texto del TextView como naranja, haciendo las modificaciones oportunas en el fichero res/values/styles.xml
2. Aplícalo al TextView que aparece.
3. Crea un nuevo estilo con nombre MiEstilo.Botones. Tiene que modificar el android:padding (tamaño en dp) y android:textSize (el tamaño en sp).
4. Aplícalo a algunos de los botones.
5. Visualiza el resultado.

Ejercicio 3: Creando un Tema para la actividad

6. En el proyecto P_05_Creciente_01 y usando el editor de temas, crea un tema con nombre TemaMio que herede de AppTheme.NoActionBar.
7. Fija la propiedad android:windowFullscreen con valor a true (la actividad ocupa toda la pantalla)
8. Aplica este tema a la actividad principal.
9. Instala el resultado.

Ejercicio 4: Creando un Tema para la aplicación

10. Quita el tema anteriormente aplicado a la actividad principal.
11. Cambia la paleta de colores a una que sea de tu gusto.
12. Crea un tema con nombre TemaMio2 que herede de AppTheme.NoActionBar.
13. Fija el ítem android:windowBackground con valor a una imagen que hayas copiado a la carpeta drawable y que tenga como nombre fondo.jpg (o fondo.png)

```
<style name="TemaMio2" parent="AppTheme.NoActionBar" >  
    <item name="android:windowBackground">@drawable/fondo</item>  
</style>
```

14. Aplica este tema a la aplicación e instala la aplicación.
15. Prueba en dispositivos de distintos tamaños y/o densidades. Se ve bien siempre? No. Recuerda guardar la imagen en distintos tamaños en recursos alternativos de drawables.