ESTILOS

En este documento trataremos de cómo maneja los estilos Android.
Para ello tendremos presentes los siguientes enlaces:

Enlaces sobre estilos:

I. https://m3.material.io/

II. https://m3.material.io/theme-builder

III. https://developer.android.com/jetpack/compose/designsystems/material3

IV. https://github.com/material-components/material-components-android/tree/master/docs/components/assets

¿Qué es Material Design?

Material Design es un lenguaje de diseño desarrollado por Google, que busca crear interfaces visuales coherentes, modernas y funcionales, imitando cómo los objetos físicos (materiales) se comportan en el mundo real. Se enfoca en el uso de sombras, animaciones, capas y movimientos naturales para simular el comportamiento del papel y la tinta, permitiendo que las interfaces sean intuitivas y atractivas.

Cabe remarcar que Material no es solamente de Android. **Material Design** se puede usar de varias formas y en diferentes plataformas, tanto para aplicaciones web como móviles. Google ha desarrollado herramientas, bibliotecas y frameworks que permiten implementar el lenguaje de diseño en una amplia variedad de entornos. Aquí te explico algunas de las principales formas de utilizarlo:

1. Web

- **Material Design Lite (MDL)**: Es una biblioteca de CSS y JavaScript que proporciona componentes basados en Material Design. Puedes incluirla en tu proyecto web para crear sitios que sigan las guías de diseño de Material.
- **Materialize CSS**: Un framework CSS basado en Material Design, que facilita la creación de interfaces visualmente atractivas con componentes listos para usar.
- **Angular Material**: Librería oficial para Angular que implementa Material Design. Ofrece componentes como botones, formularios, diálogos, tablas, etc., diseñados según las directrices de Material Design.
- **Vue Material**: Librería para Vue.js que implementa componentes de Material Design.

2. Aplicaciones móviles

- **Android**: Material Design es nativo de Android y se integra en el desarrollo de aplicaciones a través del framework **Android SDK**. Las aplicaciones Android utilizan el lenguaje de Material Design por defecto, y los desarrolladores pueden personalizarlo usando XML o Kotlin/Java.
- **React Native (con Paper o UI Kitten)**: Puedes utilizar bibliotecas como **React Native Paper**, que implementa Material Design para aplicaciones móviles usando React Native.

3. Aplicaciones multiplataforma

- **Flutter**: Es un framework de Google para construir aplicaciones nativas multiplataforma (iOS, Android, Web) usando un único código base. Flutter tiene soporte integrado para Material Design, por lo que puedes crear aplicaciones con este estilo visual de forma sencilla.

4. Extensiones y herramientas de diseño

- **Material Design Icons**: Hay conjuntos de íconos basados en Material Design que puedes utilizar en tus proyectos, disponibles en formato web o para su uso en aplicaciones móviles.
- **Google Fonts**: Parte de Material Design incluye recomendaciones tipográficas, como el uso de fuentes disponibles en Google Fonts (e.g., **Roboto**, **Montserrat**), que se integran fácilmente en proyectos web.

5. Diseño gráfico y prototipado

- **Figma, Adobe XD, Sketch**: Estas herramientas de diseño tienen kits y recursos para diseñar interfaces según las guías de Material Design, lo que facilita el prototipado y la creación de interfaces.

MATERIAL EN APLICACIONES ANDROID

Una vez sabido esto vamos a centrarnos en su uso en aplicaciones para Android. Android por defecto comienza las aplicaciones utilizando estilos de Material. Las librerías necesarias se pueden ver que están en el archivo graddle y son descargadas al realizar un proyecto.

Para ello abramos el primer enlace facilitado anteriormente.

1. Haremos una descripción, pasando por cada menú.
    a. Home y Get Started: Tutorial de todo (Material Design, Fuentes etc.)
    b. Develop: Puedes entrar en la plataforma en la que vayas a trabajar, para ver tutoriales y crear estilos.
    c. Styles: Puedes ir en los diferentes apartados, viendo y obteniendo estilos, iconos etc.
    d. Components: Puedes ir componente a componente viendo cómo podrias aplicar ese estilo en tu proyecto.
2. Más adelante veremos cómo aplicarlo a mi app, pero antes cabe recordar algunos conceptos que hemos ido viendo días atrás.
    a. Recordad que, en el Manifest, está vinculado del estilo que va a tirar mi app, que está en Themes. (ahí también está el nombre de la app, que, si lo cambio en Strings, cambia automáticamente en todos los sitios dónde se referenciaba)
    b. En Themes (dentro de res→values), tenemos los estilos de los que "tira" mi app, tanto para tema claro cómo oscuro.

```
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Base.Theme.Estilos" parent="Theme.Material3.DayNight.NoActionBar">
        <!-- Customize your light theme here. -->
        <!-- <item name="colorPrimary">@color/my_light_primary</item> -->
    </style>

    <style name="Theme.Estilos" parent="Base.Theme.Estilos" />
</resources>
```

Como veis, se hereda de esos estilos Theme…..NoActionBar.

Lanzaremos una app, y cambiar el estilo por los siguientes para que veáis que la app cambia:

```
-   Theme.Material3.Light
-   Theme.Material3.Dark
-   Theme.Material3.DayNight
```

Vamos a crear un proyecto nuevo, que llamaremos PruebasConEstilos, solamente para ir haciendo pruebecillas y que luego apliquéis en algunas apps que hemos realizado.

1. Añade 2 botones y otro TextView además del inicial y ve probando todo lo que voy indicando.
2. Podemos cambiar el fondo fácilmente, con backgroundTint (es diferente a background, sólo cambia el color sin afectar al resto del estilo que al venIr de Material es más complejo y por eso background, no hace caso, es más si lo aplicas, te 'destroza' el resto de comportamiento del botón)
3. Juega con la propiedad backTintMode.
4. ¿cómo podrías quitar el redondeo de las esquinas? Efectivamente, prueba con cornerRadius.
5. Juega ahora con strokeWidth y strokeColor.
6. Prueba con ripleColor. (mola ¿eh?).
7. Así ve probando un rato con otras propiedades como padding, Fontfamily y FontSize, Style, Alignment, etc. Si descubres alguna chula, indícalo y la probamos todos en clase.

Lo correcto no es ir modificando elemento a elemento, sino crear estilos para que todas las views tengan el mismo aspecto. Para ello crearemos un estilo, por ejemplo, para el color de fondo de los botones y se lo aplicaremos a todos los botones.

8. Crea el siguiente estilo (previamente añade los colores en el archivo "colors".

```
<style name="misBotones">
    <item name="backgroundTint">@color/fondoVerde</item>
    <item name="android:textColor">@color/textoAzul</item>
</style>
```

Así ya podrás aplicar esos estilos a todos los botones uno a uno. Puedes cambiar las propiedades anteriormente vistas, que salen al escribir name="" en la ayuda textual (que sale pulsando CTRL+espacio.

He completado algunos estilos pero libremente prueba otros:

```
<style name="misBotones">
    <item name="backgroundTint">@color/fondoVerde</item>
    <item name="android:textColor">@color/textoAzul</item>
    <item name="android:textSize">16sp</item>
    <item name="android:textAlignment">center</item>
    <item name="android:padding">3dp</item>
    <item name="android:gravity">center</item>
    <item name="rippleColor">@color/fondoAmarillo</item>
    <item name="cornerRadius">10dp</item>

</style>
```

Ejecutar y veréis cómo quedan. Qué chula la propiedad rippleColor, ¿verdad?

Ahora una vez sabido cómo podemos cambiar los estilos a mano, pasaremos a ver cómo cambiar estilos utilizando librerías o herramientas que ofrece Material.

9. Entrad en el segundo enlace de los de arriba.
10. Personalizad tema, colores, fuentes…y descargar el archivo. (como veis se puede exportar para diferentes tecnologías).
11. Ahora con los ficheros generados, id añadiendo a los de la app. Es mejor abrir los ficheros y añadir sin tocar las cabeceras, a sobrescribir todo, por ese motivo. Ejemplo, abres colors.xml, y le añades todos los colores generados. (puedes pegarlos desde Android Studio). Comprueba cómo ha cambiado.
12. Cambiad el fondo de la app a alguno de los colores añadidos, o añadid algún elemento y veréis como ya sale con los nuevos estilos.

Ahora aplicar un estilo a una de las aplicaciones anteriores, por ejemplo, la de formulario, pizza o encuesta y enseñádmelo.

Además de aplicar estilos generales, podemos utilizar componentes (views), que nos da Material con un aspecto más moderno y agradable. Veremos algún ejemplo, pero tenerlos en cuenta para otros elementos que queráis aplicar desde ahora.

13. Para ello vamos a añadir un par de iconos, ya que podemos decorar ciertas views.
14. Abrid el cuarto enlace de los de arriba. (se puede acceder también desde el primer enlace, en el menú de la izquierda en Components y eligiendo el componente y la tecnología que necesitas.
15. Mirad el ejemplo de un botón elevado. Simplemente es añadirle un estilo una vez que añades el botón:

```
style="@style/Widget.Material3.Button.ElevatedButton"
```

16. Si quieres añadir un icono hay que seguir las instrucciones, añadamos .Icon y luego el icono que tengas.

```
style="@style/Widget.Material3.Button.ElevatedButton.Icon"
app:icon="@drawable/ic_add_24dp"
```

17. Ahora, probemos algún campo de texto que los hay muy chulos. Lo suyo es ya añadir estos en vez de los del IDE. Ponedles las restricciones.
    Comprobad que ahora está la entrada de texto dentro de un layout para hacer efectos chulos.

```
<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/textField"
    android:layout_width="292dp"
    android:layout_height="73dp"
    android:hint="TextoChulo"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.496"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.821">

    <com.google.android.material.textfield.TextInputEditText
        android:layout_width="match_parent"
        android:layout_height="60dp" />

</com.google.android.material.textfield.TextInputLayout>
```

18. Si queremos añadir un icono, se haría así:

```
app:startIconContentDescription="@string/descripcion_texto"
app:startIconDrawable="@drawable/ic_avion_rojo">
```

19. Ahora hacer vosotros una prueba con el password.
20. Así investigad otros componentes, por ejemplos los diálogos quedan muy chulos. Añadid un diálogo al pulsar un botón.