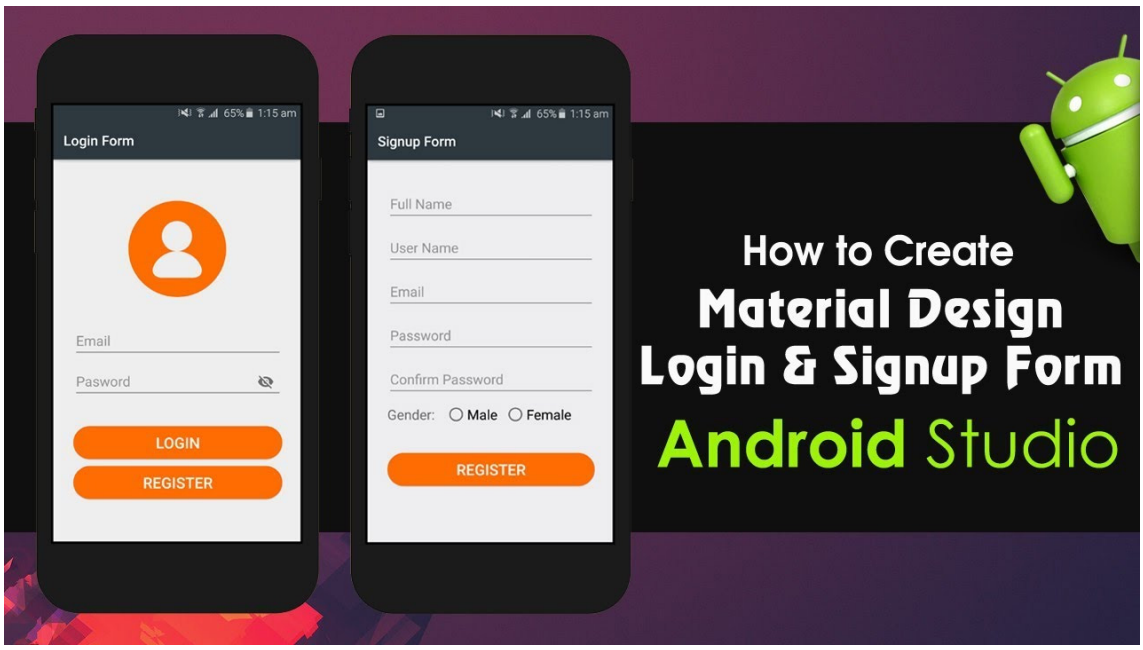
UNIDAD 3: Estilos y Plantillas

Podemos mejorar la apariencia de nuestra aplicación por medio de formas, estilos y plantillas.



FORMA (SHAPE)

Una forma (*shape*) como su nombre indica, permite establecer la apariencia de una vista (formato, borde, color de fondo, etc.).

Las formas se definen mediante un elemento "shape" dentro de archivo xml de la carpeta de recursos "*drawable*".
Sobre este elemento "shape" se establecerá como propiedad la forma (*shape*) del elemento (*line*, *rectangle*, *ring*, *oval*), y en su interior se podrán definir las características de cada uno de sus componentes: borde (*stroke*), esquinas (*corners*), color de fondo (*solid* o *gradient*), tamaño (*size*) y margen interno (*padding*).

Ejemplo:
Vamos a definir una forma para los EditText, y para ello crearemos dentro de la carpeta "drawable" el archivo "edit_style.xml" con las siguientes propiedades: forma rectangular, esquinas redondeadas, bordes de 3dp de color azul oscuro, y color de fondo azul claro.

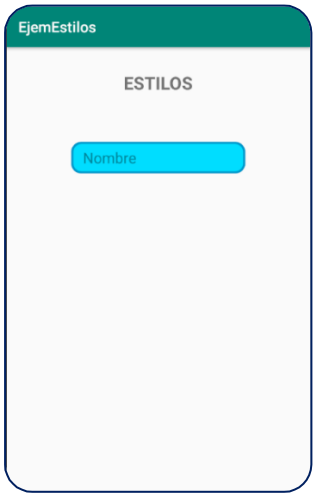
```
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">

    <stroke
        android:width="3dp"
        android:color="@android:color/holo_blue_dark"/>

    <padding
        android:top="8dp"
        android:bottom="8dp"
        android:left="16dp"
        android:right="16dp"/>

    <corners
        android:radius="12dp"/>

    <solid
        android:color="@android:color/holo_blue_bright"/>

    <!--
    <gradient
        android:startColor="@android:color/white"
        android:endColor="@android:color/darker_gray"
        android:type="linear"
        android:angle="270"/>
    -->

</shape>
```

Nota: se ha comentado un ejemplo de cómo se haría para utilizar un gradiente como fondo en lugar de un color plano.

Tras definir la forma, sólo tendríamos que añadirla como fondo (*background*) en todos los controles en los que queramos utilizarlo:

```
<EditText
    android:id="@+id/editNombre"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="64dp"
    android:ems="10"
    android:hint="Nombre"
    android:inputType="textPersonName"
    android:textSize="20sp"
    android:background="@drawable/edit_style"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/etiTitulo" />
```

Esta sería la apariencia del *EditText* tras establecer la forma como fondo(*background*):



ESTILO (STYLE)

Un estilo (*style*) es un conjunto de propiedades de formato al que damos un nombre para que sean aplicadas de forma conjunta ya sea en una vista o en una ventana de nuestra app.

Los estilos de definen en un archivo xml dentro de la carpeta de recursos "*values*".
El elemento raíz debe ser obligatoriamente "*resources*" y cada estilo se definirá mediante un elemento "*style*" que tendrá que ir acompañado de un nombre (*name*), y opcionalmente de un estilo padre del que heredar características (*parent*).

Cada una de las propiedades del estilo se definirán mediante un elemento "ítem" dentro del estilo.

Ejemplo:

```
<?xml version="1.0" encoding="utf-8"?>
<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>

    <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>
```

Más información sobre estilos y plantillas:
https://developer.android.com/guide/topics/ui/themes.html?hl=es-419

MATERIAL DESIGN

La creación de estilos y temas permite personalizar los diferentes elementos visuales de la aplicación para darle un aspecto más profesional, pero puede ser una labor muy compleja y que consuma un tiempo excesivo.

Material Design pone a disposición de los desarrolladores un gran abanico de componentes y plantillas de gran calidad, que luego podremos personalizar a nuestro gusto sin tener que realizar todo el proceso desde cero.

https://material.io/develop/android/

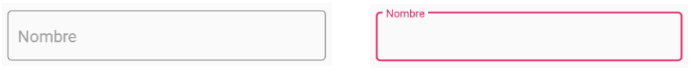
Para poder utilizar los componentes de Material Design, en primer lugar tendremos que importar la dependencia a nuestro proyecto, añadiendo a Gradle:
implementation 'com.google.android.material:material:<version>'

```
implementation 'com.google.android.material:material:1.0.0'
```

Las versiones actualizadas de *Material Design* suelen requerir los últimos niveles de API. Por ejemplo, para utilizar la versión 1.2.0 de *Material Design* se requiere Android 10 (API 29).

A continuación se muestran algunos ejemplos:

Caja de texto cuyo *hint* se muestra como etiqueta al recibir el foco



Son necesarios dos controles:

com.google.android.material.textfield.TextInputLayout

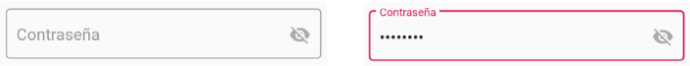
Hace la función de contenedor y que albergará el *hint* cuando la caja de texto tiene el foco

com.google.android.material.textfield.TextInputEditText

Es la caja de texto en la que se introduce el dato y mientras no tiene el foco muestra la ayuda o *hint*.

```
<com.google.android.material.textfield.TextInputLayout
  style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"
  android:id="@+id/layoutUser"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:layout_marginTop="64dp"
  android:layout_marginHorizontal="32dp"
  app:layout_constraintEnd_toEndOf="parent"
  app:layout_constraintStart_toStartOf="parent"
  app:layout_constraintTop_toBottomOf="parent">

  <com.google.android.material.textfield.TextInputEditText
    android:id="@+id/editUser"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Nombre"/>

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

Caja de texto para contraseña con botón de mostrar/ocultar contraseña



Se crea de forma idéntica al anterior, pero al contenedor se le añade la propiedad "*passwordToogleEnabled*" para que permita mostrar y ocultar el texto y "*passwordToogleTint*" para establecer el color del botón, y en la caja de texto se establece el valor del "*inputType*" como "*textPassword*" para indicar que es una contraseña y que por tanto los caracteres se deben ocultar.

```
<com.google.android.material.textfield.TextInputLayout
  style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"
  android:id="@+id/layoutPass"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:layout_marginTop="64dp"
  android:layout_marginHorizontal="32dp"
  app:layout_constraintEnd_toEndOf="@+id/layoutUser"
  app:layout_constraintStart_toStartOf="@+id/layoutUser"
  app:layout_constraintTop_toBottomOf="@+id/layoutUser"
  app:passwordToggleEnabled="true"
  app:passwordToggleTint="@android:color/darker_gray">

  <com.google.android.material.textfield.TextInputEditText
    android:id="@+id/editPass"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Contraseña"
    android:inputType="textPassword"/>

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