Imagenes

Compose

Icon() **Image()** **AsyncImage()**

Icon()

Está diseñado para mostrar íconos. Normalmente se utiliza junto a colecciones de íconos (como Icons.Default) o vectores que tienen un estilo predefinido.

```
@Composable
fun Icon(painter: Painter,
        contentDescription: String?,
        modifier: Modifier = Modifier,
        tint: Color = LocalContentColor.current
       )

@Composable
fun Icon(imageVector: ImageVector,
        contentDescription: String?,
        modifier: Modifier = Modifier,
        tint: Color = LocalContentColor.current
       )

@Composable
fun Icon(bitmap: ImageBitmap,
        contentDescription: String?,
        modifier: Modifier = Modifier,
        tint: Color = LocalContentColor.current
       )
```

Image()

Se utiliza para mostrar imágenes en general.

```
@Composable
fun Image(painter: Painter,
        contentDescription: String?,
        modifier: Modifier = Modifier,
        alignment: Alignment = Alignment.Center,
        contentScale: ContentScale = ContentScale.Fit,
        alpha: Float = DefaultAlpha,
        colorFilter: ColorFilter? = null)

@Composable
fun Image(imageVector: ImageVector,
        contentDescription: String?,
        modifier: Modifier = Modifier,
        alignment: Alignment = Alignment.Center,
        contentScale: ContentScale = ContentScale.Fit,
        alpha: Float = DefaultAlpha,
        colorFilter: ColorFilter? = null)

@Composable
fun Image(bitmap: ImageBitmap,
        contentDescription: String?,
        modifier: Modifier = Modifier,
        alignment: Alignment = Alignment.Center,
        contentScale: ContentScale = ContentScale.Fit,
        alpha: Float = DefaultAlpha,
        colorFilter: ColorFilter? = null,
        filterQuality: FilterQuality=DefaultFilterQuality)
```

Icon()

Image()

painterResource(id: Int)

Devuelve un Painter que se adapta al tipo del recurso (bitmap o vector). Es la opción más versátil para usar en Image.

```
Image(
    painter = painterResource(id = R.drawable.my_png_image),
    contentDescription = "Imagen PNG de ejemplo",
    modifier = Modifier.size(200.dp)
  )
```

ImageVector.vectorResource(id: Int)

Específico para recursos vectoriales, ideal para Icons.

```
Icon(imageVector = Icons.Filled.MailOutline,
    contentDescription = null,
    tint = Color.Red,
    modifier = Modifier.size(100.dp)
  )
```

```
Icon(painter = painterResource(android.R.drawable.ic_menu_edit),
    contentDescription = null,
    tint = Color.Red,
    modifier = Modifier.size(100.dp)
  )
```

Icon()

Image()

```
Image(
    painter = painterResource(id = R.drawable.visibility_on),
    contentDescription = "Visivilidad activada",
    modifier = Modifier.size(200.dp),
    contentScale = ContentScale.Inside,
    filterQuality = FilterQuality.High,
  )
```

contentScale =

El parámetro contentScale *en el composable Image permite aplicar transformaciones en cuanto al escalado de la imagen:*

ContentScale.Fit
La imagen se ajusta para caber dentro del contenedor sin recortar ninguna parte, manteniendo la relación de aspecto. Puede dejar espacios vacíos si las proporciones no coinciden.

ContentScale.Crop
La imagen se escala de manera que el contenedor quede completamente cubierto, pero se recortarán las partes que excedan el contenedor.

ContentScale.FillBounds
La imagen se estira para llenar el contenedor, lo que puede distorsionar la imagen si las proporciones difieren.

ContentScale.Inside
La imagen se escala solo si es más grande que el contenedor, pero se mantiene sin recorte y sin distorsión.

filterQuality =

El parámetro filterQuality *determina la suavidad y calidad de la imagen cuando se escala (a mayor nivel más calidad de la imagen pero menor rendimiento).*

FilterQuality.None *No aplica ninguna interpolación*. La imagen se dibuja tal cual

FilterQuality.Low

FilterQuality.Medium

FilterQuality.High

Icon()

Image()

```
Image(
    painter= painterResource(id= R.drawable.visibility_on),
    contentDescription = null,
    modifier = Modifier.size(200.dp),
    contentScale = ContentScale.Crop,
    alpha = 0.2f,
    colorFilter = ColorFilter.tint(Color.Red)
   )
```

alpha

El parámetro alpha controla la opacidad de la imagen, permitiéndote definir un nivel de transparencia. Su valor es un Float que va de 0.0f (completamente transparente) a 1.0f (completamente opaco).

colorFilter =

El parámetro colorFilter en el composable Image permite aplicar transformaciones de color. Por ejemplo, convertir la imagen a escala de grises, ajustar el brillo, la saturación o el contraste.

ColorFilter.tint(Color.Red)

La transformación tint() tiñe la imagen de un color.

AsyncImage()

Permite cargar una imagen de forma asíncrona fundamentalmente a partir de una URL

libs.versions.toml

```
[versions]
  : : :
#Añadido para AsyncImage
coilCompose = "3.1.0"
coilNetwork = "3.1.0"
```

```
[libraries]
 : : :
#Añadido para AsyncImage
coil-compose = {group = "io.coil-kt.coil3", name = "coil-compose", version.ref = "coilCompose"}
coil-network = {group = "io.coil-kt.coil3", name = "coil-network-okhttp", version.ref = "coilNetwork"}
```

build.gradle.kts

```
dependencies {
   : : :
  implementation(libs.coil.compose)   //Añadido para AsyncImage
  implementation(libs.coil.network)   //Añadido para AsyncImage
}
```

AsyncImage()

```
@Composable
@NonRestartableComposable
fun AsyncImage(model: Any?,
        contentDescription: String?,
        modifier: Modifier = Modifier,
        alignment: Alignment = Alignment.Center,

        contentScale: ContentScale = ContentScale.Fit,
        alpha: Float = DefaultAlpha,
        colorFilter: ColorFilter? = null,
        filterQuality: FilterQuality = DefaultFilterQuality,

        placeholder: Painter? = null,
        error: Painter?  = null,
        fallback: Painter? = error,
)
```

AsyncImage()

model

El parámetro model *define la ubicación de la imagen*

"https://miro.medium.com/v2/resize:fit:1400/1*-1elwQ9eKqBI-q6cGngDKg.png"
String que representa la URL de la imagen.

Res.getUri("drawable/sample.jpg")
Recurso de la aplicación desde el que obtener la imagen

contentScale

El parámetro contentScale *en el composable Image permite aplicar transformaciones en cuanto al escalado de la imagen:*

ContentScale.Fit
La imagen se ajusta para caber dentro del contenedor sin recortar ninguna parte, manteniendo la relación de aspecto. Puede dejar espacios vacíos si las proporciones no coinciden.

ContentScale.Crop
La imagen se escala de manera que el contenedor quede completamente cubierto, pero se recortarán las partes que excedan el contenedor.

ContentScale.FillBounds
La imagen se estira para llenar el contenedor, lo que puede distorsionar la imagen si las proporciones difieren.

ContentScale.Inside
La imagen se escala solo si es más grande que el contenedor, pero se mantiene sin recorte y sin distorsión.

AsyncImage()

filterQuality

El parámetro filterQuality *determina la suavidad y calidad de la imagen cuando se escala (a mayor nivel más calidad de la imagen pero menor rendimiento).*

FilterQuality.None *No aplica ninguna interpolación*. La imagen se dibuja tal cual

FilterQuality.Low

FilterQuality.Medium

FilterQuality.High

alpha

El parámetro alpha controla la opacidad de la imagen, permitiéndote definir un nivel de transparencia. Su valor es un Float que va de 0.0f (completamente transparente) a 1.0f (completamente opaco).

colorFilter

El parámetro colorFilter en el composable Image permite aplicar transformaciones de color. Por ejemplo, convertir la imagen a escala de grises, ajustar el brillo, la saturación o el contraste.

```
ColorFilter.tint(Color.Red)
```

La transformación tint() tiñe la imagen de un color.

AsyncImage()

placeholder

Establece un painter que será visualizado mientras se carga la imagen.

```
AsyncImage(model= "https://miro.medium.com/v2/resize:fit:1400/1*-1elwQ9eKqBI-q6cGngDKg.png",
        placeHolder = painterResource(id = R.drawable.my_png_image),
```

error

Establece un painter que será visualizado si se produce un error durante la carga de la imagen

```
AsyncImage(model= "https://miro.medium.com/v2/resize:fit:1400/1*-1elwQ9eKqBI-q6cGngDKg.png",
        placeHolder = painterResource(id = R.drawable.my_placeholder_image),
      error = painterResource(id = R.drawable.my_error_image),
```

fallback

Establece un painter que será visualizado si model es null

```
AsyncImage(model= producto.imagen,
        placeHolder = painterResource(id = R.drawable.my_placeholder_image),
      error = painterResource(id = R.drawable.my_error_image),
```