

# Definiendo Componente Imagen en la Agenda

[Descargar estos apuntes](#)

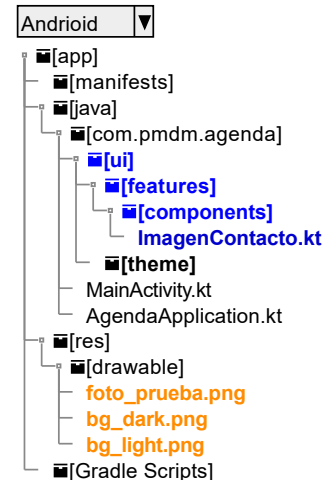
## Ejercicio

Partiremos del **ejercicio 4 del bloque 2**. Donde definimos nuestra propia clase Aplicación a la **Agenda**.

Vamos a crear un **composable** personalizado a partir de **Image** para mostrar la foto de cada contacto de forma homogénea en diferentes contenedores. Cómo va a ser una **feature** reutilizable, lo crearemos en un paquete de componentes re-utilizables denominado **com.pmdm.agenda.ui.features.components** tal y como se muestra en el diagrama. Además, el fuente se llamará **ImagenContacto.kt**.

Además arrastraremos los siguientes recursos, para realizar las pruebas de previsualización, a la carpeta **res/drawable**:

1. **foto\_prueba.png**: Una foto de prueba de un contacto.
2. **bg\_dark.png**: Un fondo para probar el componente con tema oscuro.
3. **bg\_light.png**: Un fondo para probar el componente con tema claro.



Nuestro componente se denominará **ImagenContacto** y tendrá el siguiente interfaz y aspecto:

```
@Composable
fun ImagenContacto(
    modifier: Modifier = Modifier,
    foto : ImageBitmap?,
    anchoBorde : Dp = 4.dp,
)
```



La especificación de la función **ImagenContacto** es la siguiente:

1. Definirá una variable **painterFoto** que será un **Painter** que se inicializará con la **foto** del contacto si es distinta de null o con un icono de cara de Material (**Icons.Filled.Face2**), si no la tiene.
2. Definirá un **composable Image** con los siguientes parámetros:
  - **painter**: Será el **painterFoto** definido anteriormente.
  - **contentScale**: Será **ContentScale.Crop**.
  - **contentDescription**: Será **"Imagen contacto"**.
  - **modifier**: Será el modificador que se le pase como parámetro a la función seguido de:
    - **clip(CircleShape)**: Para que la imagen tenga forma circular.
    - **aspectRatio(ratio = 1f)**: Para que la imagen tenga un aspecto cuadrado.
    - **background(MaterialTheme.colorScheme.surface)**: Para que el fondo de la imagen sea el color de fondo de la superficie en el tema.
    - **border(width = anchoBorde, color = MaterialTheme.colorScheme.inversePrimary, shape = CircleShape)**: Para que la imagen tenga un borde de color primario inverso de ancho **anchoBorde** y forma circular como el clip de la foto.

Vamos ahora a probar nuestro componente en **un primer @Preview**. Para ello, definiremos el composable **ImagenContactoPreviewSinFoto** que aplicará el tema **AgendaTheme** un **Box** con un fondo de color primario y un tamaño de **300dp** de ancho y **200dp** de alto. Dentro del **Box** definiremos un **ImagenContacto** con un modificador que ocupe **todo el alto** del **Box** y que no tenga foto.

Definiremos **dos @Preview** más, una con tema **claro** y otra con tema **oscuro**. Sobre un composable denominado **ImagenContactoPreviewConFotoYFondo** que aplicará el tema **AgendaTheme** un **Box** con un fondo de color primario y un tamaño de **300dp** de ancho y **200dp** de alto. Dentro del **Box** definiremos:

1. Un **Image** con una imagen de fondo que dependerá de si el tema es claro ( **bg\_light.png** ) u oscuro ( **bg\_dark.png** ). Para decidir cual aplicamos, usaremos el método **isSystemInDarkTheme() : Boolean** que está definido en **foundation**. Además, el parámetro **contentScale** será **ContentScale.FillBounds** y un modificador **matchParentSize()**.
2. Un **ImagenContacto** con un modificador que ocupe **todo el alto** del **Box** y que tenga como foto la imagen de prueba **foto\_prueba.png**.

Esta última función con 2 'previews' me deberá producir una salida similar a la siguiente:

