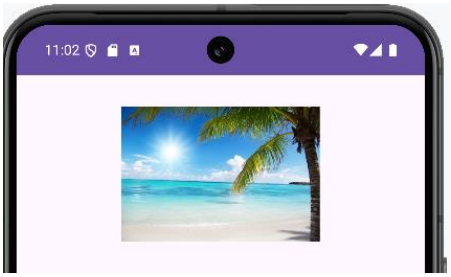
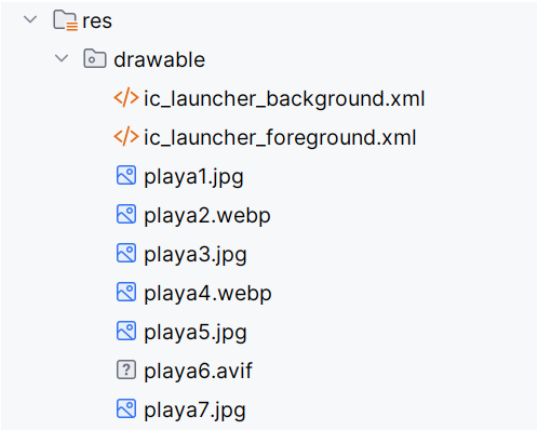
RECYCLERVIEW: Carrusel de Imágenes.



En esta app, veremos como crear un carrusel de imágenes usando un RecyclerView.

Crea un proyecto: File→New Project → Empty Views Activity, lo llamaremos, por ejemplo, **RecyclerViewCarrusel**.

1. Vamos a buscar las imágenes que queramos que aparezcan en la galeria de imágenes o carrusel y las pondremos en el directorio ***res/drawable***



2. Crearemos la object class **ImagenesProvider** dentro de la cual declararemos un array con los nombres de las imágenes que hemos pues en el directorio **res/drawable** (sin incluir la extensión, solo el nombre).

```
object ImagenesProvider {
    val imagenesList = arrayListOf(
        "playa1",
        "playa2",
        "playa3",
        "playa4",
        "playa5",
        "playa6",
        "playa7"
    )
}
```

3. En el layout de nuestra activity (activity_main.xml), añadiremos un RecyclerView, y crearemos el layout (imagen_item.xml), que servirá para dar aspecto a cada ítem de mi RecyclerView. Dentro de la carpeta layout (**New→XML→LayoutXML**), donde imagen_item.xml simplemente tiene un ImageView

4. Vamos a crear el **ViewHolder,** donde se suelen crear variables asociadas a las Views de cada elemento de la la lista, y en este caso solo tendremos la referencia a la ImageView

```
class ImagenesViewHolder(itemView: View) :
RecyclerView.ViewHolder(itemView) {
    private val binding = ImagenItemBinding.bind(itemView)

    val imagenView: ImageView = binding.idImagenView
}
```

5. Creamos el adaptador: **AdaptadorImagenes,** el cual recibe una lista de String con el nombre de las imágenes, con la cual inicializamos los datos con los que vamos a cargar el RecyclerView. En la función onBindViewHolder, vamos a acceder al ImageView usando el nombre de la imagen.

```
class AdaptadorImagenes(private val imageNames: List<String>) :
RecyclerView.Adapter<ImagenesViewHolder>() {
    private var data: List<String>
    init {
        data = imageNames
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
ImagenesViewHolder {
        // Inflamos el layout de cada elemento
        val layoutInflater = LayoutInflater.from(parent.context)
        return
ImagenesViewHolder(layoutInflater.inflate(R.layout.imagen_item,
parent, false))
    }

    override fun onBindViewHolder(holder: ImagenesViewHolder,
position: Int) {
        // Inicializamos la lista de imagenes
        val imagenName = data[position]
        //accedo al imageView, por el nombre
        val imageResourceId =
holder.itemView.context.resources.getIdentifier(
imagenName,"drawable",holder.itemView.context.packageName)
        holder.imagenView.setImageResource(imageResourceId)
    }

    override fun getItemCount(): Int {
        return data.size
    }

}
```

6. Ya solo nos falta inicializar el RecyclerView en la función onCreate de **MainActivity**. Usaremos **LinearSnapHelper**, para hacer el efecto de "snap", es decir, ese efecto de carrusel donde los elementos se centran automáticamente al visualizarlos. Además, crearemos **LinearLayoutManager**, indicando que use una alineación horizontal.

```
class MainActivity : AppCompatActivity() {
    lateinit var binding: ActivityMainBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        // Configuramos el RecyclerView
        initRecyclerView()
    }

    private fun initRecyclerView() {
        val manager = LinearLayoutManager(this,
LinearLayoutManager.HORIZONTAL, false)
        binding.rvImages.layoutManager = manager
        // Para el efecto de "snap"
        val snapHelper = LinearSnapHelper()
        snapHelper.attachToRecyclerView(binding.rvImages)

        binding.rvImages.adapter =
AdaptadorImagenes(ImagenesProvider.imagenesList)
    }
}
```