



INICIANDO EN JETPACK COMPOSE



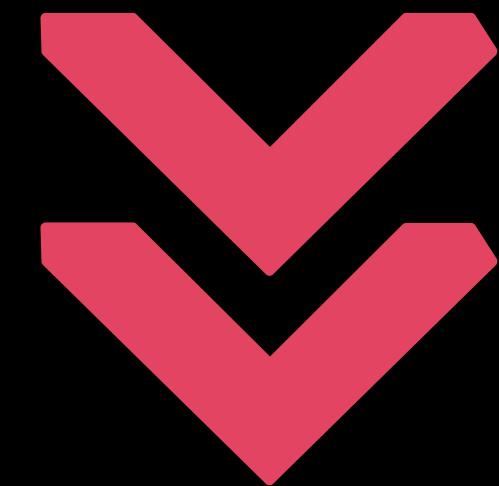
¿QUÉ VAMOS A VER?

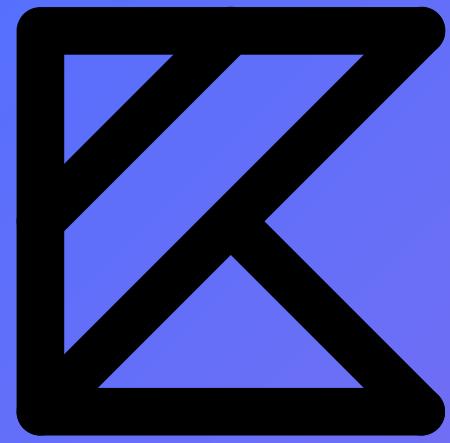
- Estructuras MainActivity.
- Modificadores.
- Columnas.
- Filas.
- LazyColumn & LazyRow.
- Estado de las variables.





¡VAMOS A COMENZAR!





ESTRUCTURAS MAINACTIVITY



ESTRUCTURAS EN MAINACTIVITY

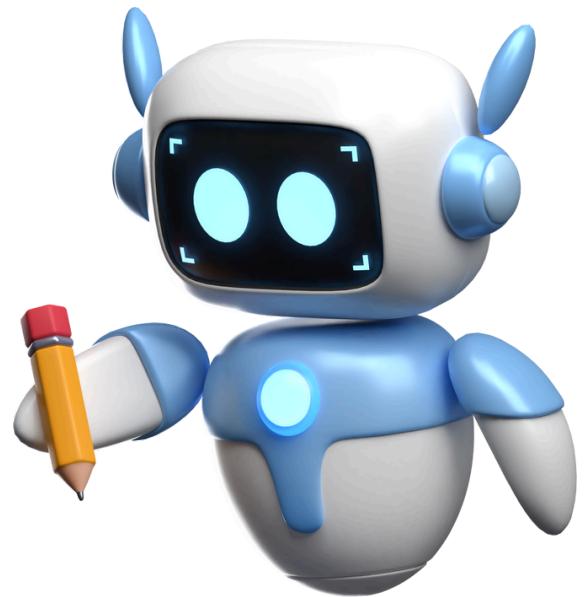
- **setContent:** Es la estructura principal que va a contener el “grueso” de nuestra aplicación. Básicamente es lo que lee el emulador cada vez que lo iniciamos.
- **enableEdgeToEdge():** Función por defecto que nos permite abarcar la totalidad del layout que tenemos disponible en el móvil. **La recomendación** es quitarlo, ya que nos puede traer problemas a la hora de organizar el diseño de nuestra aplicación.
- **@Composable:** Es una etiqueta que va a tener toda función que dentro contenga una estructura de diseño. Puede ser un botón, una imagen o incluso una lista de datos.





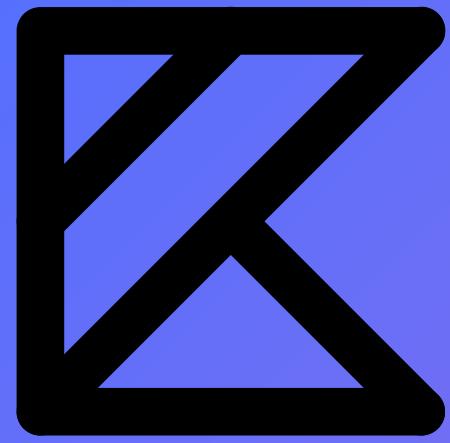
ESTRUCTURAS EN MAINACTIVITY

- **@Preview:** Esta etiqueta nos va a permitir tener a la mano una previsualización del objeto de diseño. Funciona perfecto junto a `@Composable`.
- También vamos a tener la facultad de crear funciones sin etiquetas, o más específicamente, sin `@Composable`, y eso en realidad solo sería una función normal.



```
@Composable
fun Greeting(name: String, modifier: Modifier = Modifier) {
    Button(onClick = {
        hola()
    }) {
        Text(text = name)
    }
}

fun hola() {
    println("Hola Jetpack Compose")
}
```



MODIFICADORES



MODIFICADORES

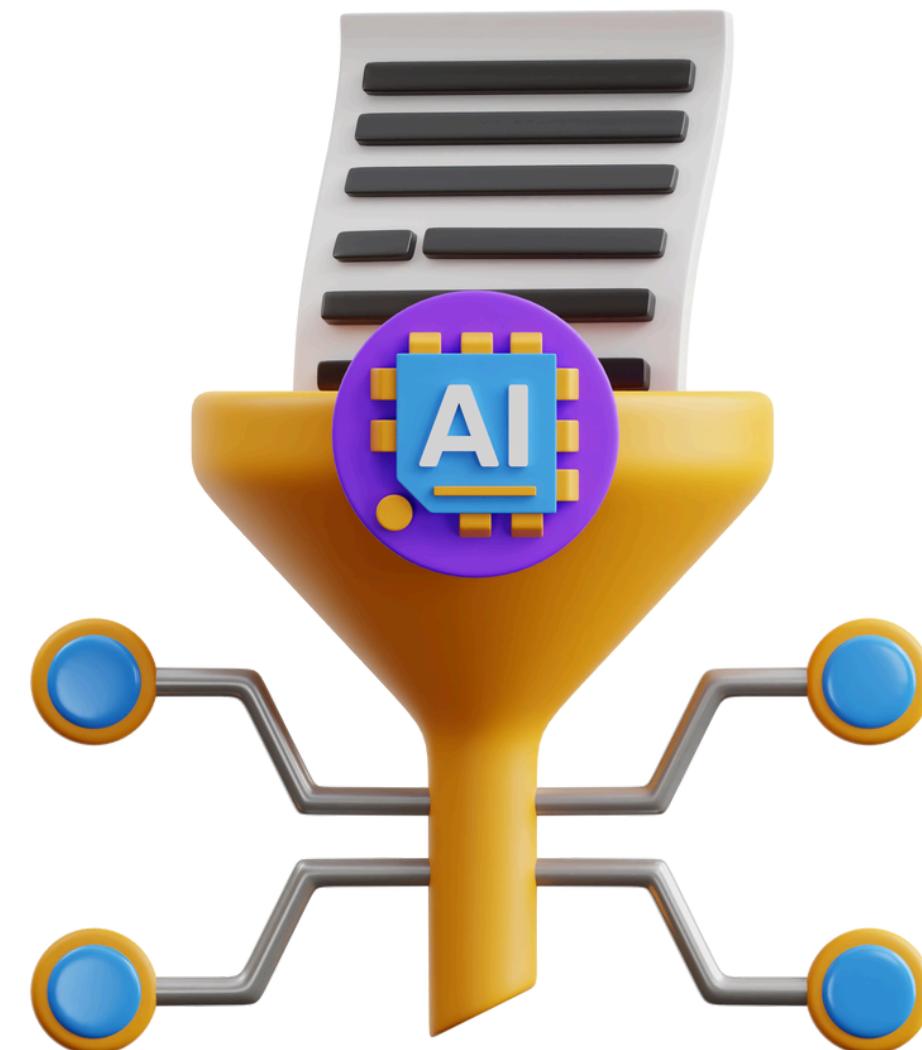
- Nos ayudan a personalizar la apariencia y el comportamiento de nuestros componentes.
- Pero, es importante que tengamos en cuenta que los modificadores no siempre afectaran a todos los elementos que tengamos en nuestro proyecto.
- Sin ir más lejos, muchas veces el Texto va a ser uno de esos elementos que no se verá afectado por los modificadores.

```
@Preview(showBackground = true)
@Composable
fun Content() {
    Text(
        text = "Hola Jetpack",
        color = Color.Red,
        fontWeight = FontWeight.Bold,
        fontSize = 40.sp,
        textAlign = TextAlign.Center
    )
}
```



MODIFICADORES

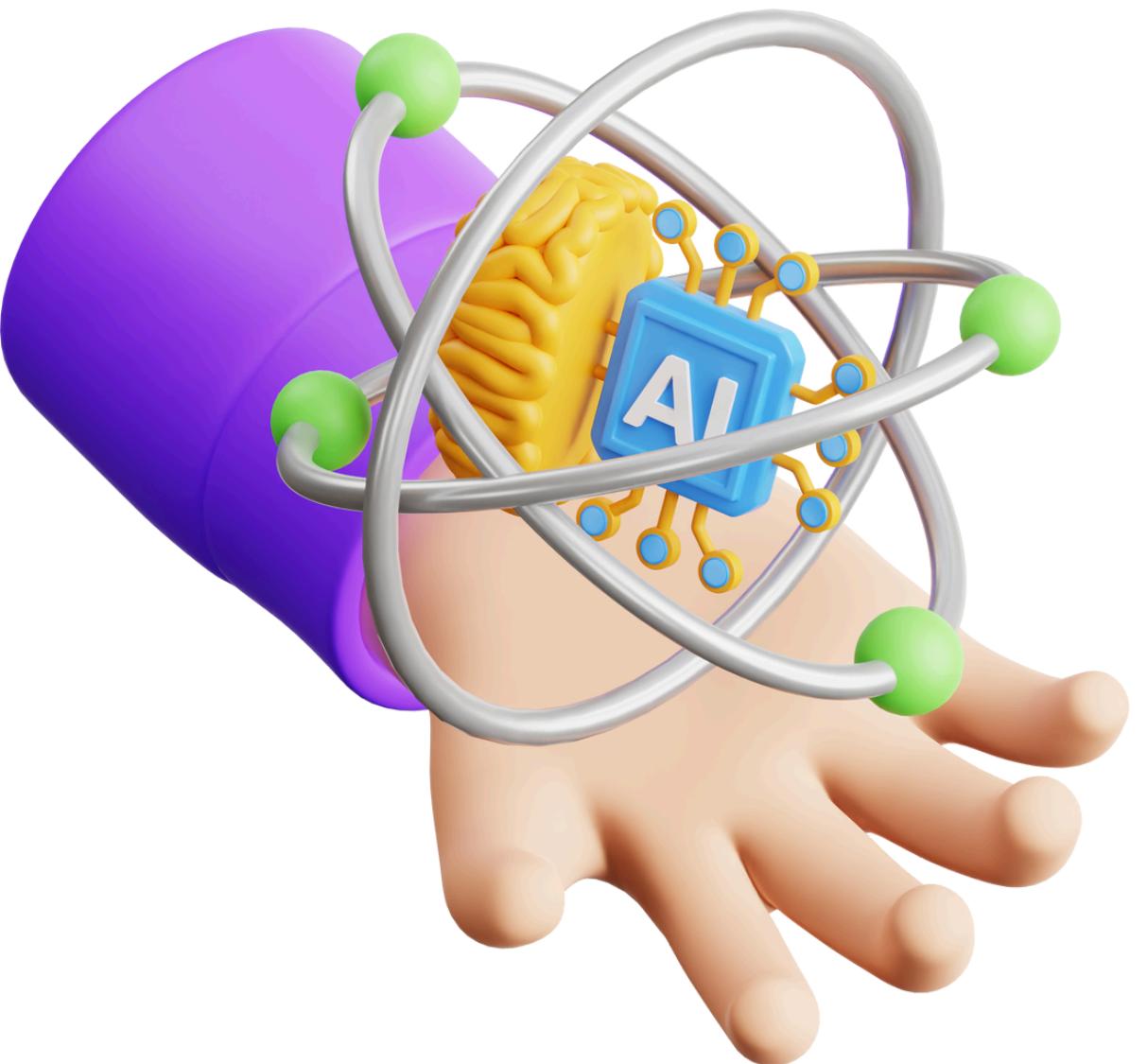
- Dentro de los modificadores nos vamos a encontrar a cuatro tipos diferentes.
- **De posicionamiento:** Nos va a permitir modificar principalmente el ancho (width) y alto (height) de nuestros elementos. También el espacio efectivo que utilicen sobre nuestros layout.
- **De funcionalidad:** Modificará las funcionalidades que pueden tener algunos elementos, como por ejemplo los clics o scrolls que necesitemos.





MODIFICADORES

- **De apariencia:** Si bien, este tipo no afectará con normalidad a los textos, se centra mucho más en backgrounds, paddings, margins, borders, etc.
- **De escucha:** Recibe y modifica eventos que puedan estar escuchando diferentes acciones dentro de nuestra aplicación, como por ejemplo un onKeyEvent, que “reaccionará” cuando se presione una tecla en específico.



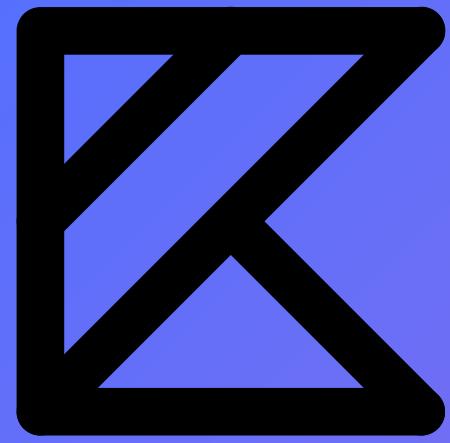


MODIFICADORES

- **Importante:** Cuando ocupemos modificadores dentro de un elemento, no podemos anexar las funcionalidades en el orden que nos de la gana, ya que eso puede afectar el comportamiento que estamos esperando.
- Es por eso que es muy útil previsualizar los cambios mientras los estamos anexando. Sobre todo cuando se trata de funcionalidades de diseño.

```
modifier = Modifier
    .background(Color.Black)
    .padding(horizontal = 30.dp)
    .clickable {
        println("Hola Jetpack")
    }
```

```
modifier = Modifier
    .padding(horizontal = 30.dp)
    .background(Color.Black)
    .clickable {
        println("Hola Jetpack")
    }
```

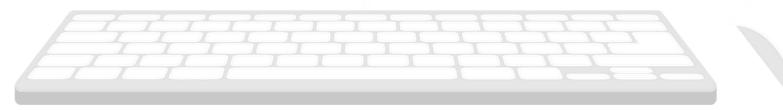
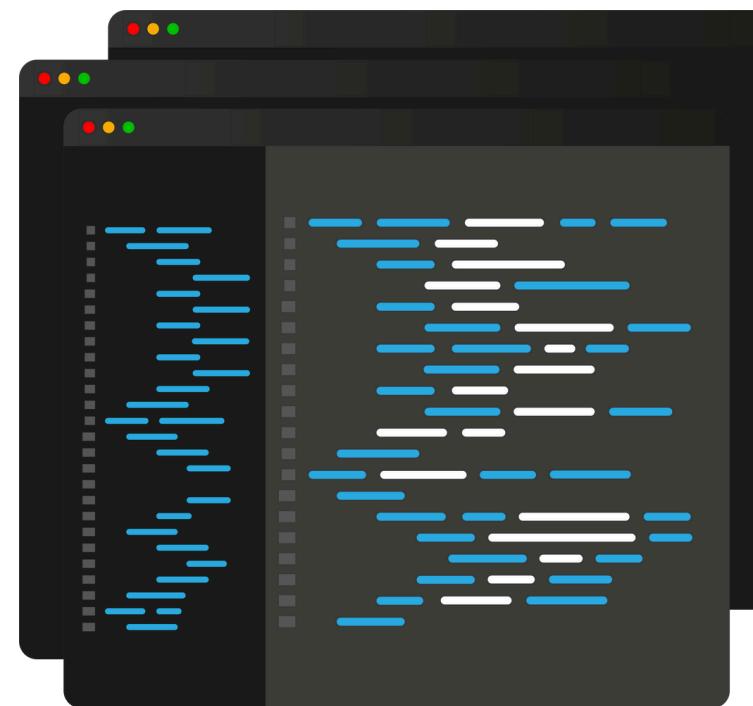


COLUMNAS



COLUMNAS

- La base del diseño en Jetpack Compose se guía principalmente por filas y columnas.
- En este caso, la columna va a ser la estructura que nos permite organizar nuestros componentes de manera vertical dentro del layout, y así evitamos que se sobrepongan unos con otros.
- También, dentro de la misma columna aplicaremos los modificadores necesarios para aprovechar mucho mejor el espacio disponible.



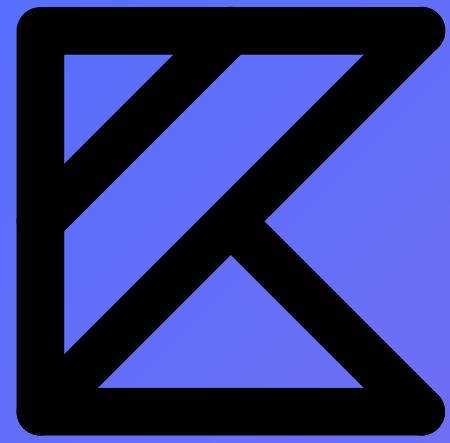
```
Column(modifier = Modifier
    .fillMaxSize()
    .wrapContentSize(Alignment.Center)
) {
```



COLUMNAS

- **.fillMaxSize()**: Nos permitirá aprovechar todo el espacio del layout, tanto en ancho como en alto.
- **.wrapContentSize(Alignment.Center)**: Podremos establecer nuestro objeto en el centro del layout.
- **.fillMaxWith()**: Nos permite aprovechar todo el ancho que tenemos disponible, respecto al contenedor padre.





FILAS

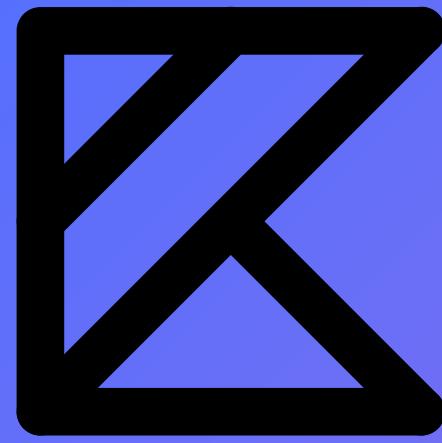


FILAS

- Dentro de nuestro diseño, vamos a tener que elegir cual estructura será la base de nuestra maqueta. Puede ser una fila o una columna sin problema, es más, dentro de cada estructura podemos agregar otras para comenzar a construir poco a poco nuestras vistas.
- Tengamos en cuenta que con las filas solo tendremos el ancho para disponer nuestro contenido, por eso, tendremos siempre a la mano diferentes métodos para organizar los objetos y que nuestra aplicación quede bien estructurada.



```
Row(modifier = Modifier  
    .fillMaxWidth(),  
    horizontalArrangement = Arrangement.SpaceEvenly  
) {
```



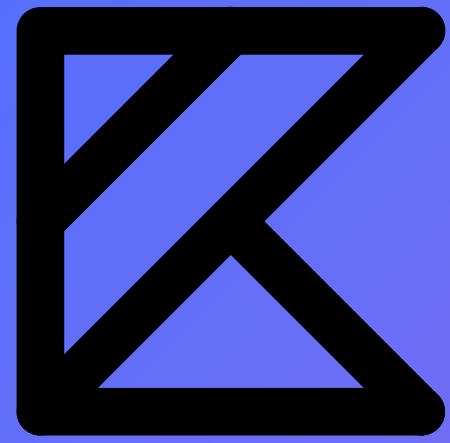
LAZYCOLUMN & LAZYROW



LAZYCOLUMN & LAZYROW

- Cumplen exactamente con el mismo trabajo que las columnas y filas normales, con la salvedad muy importante de que cuando nos quedemos sin espacio en nuestra pantalla, van a generar un scroll.
- **¿Recuerdan lo importante del RecyclerView?**
- Las estructuras Lazy van a cumplir con el **mismo principio**, cuando inicializamos nuestra aplicación, los objetos que no se ven, no van a cargar hasta que el scroll nos permita acceder a ellos.

```
LazyRow(  
    modifier = Modifier  
        .fillMaxWidth(),  
    horizontalArrangement = Arrangement.SpaceEvenly  
) {  
    items(items) { item ->  
        Circulo(color = item)  
        Spacer(modifier = Modifier.width(10.dp))  
    }  
}
```



ESTADO DE LAS VARIABLES



ESTADO DE LAS VARIABLES

- Dentro de Jetpack Compose también tenemos la oportunidad de **modificar en tiempo real** las variables que tengan nuestras vistas.
- Siempre vamos a capturar este cambio por medio de **un evento o un clic**.
- **Es importante** no olvidar que la variable que se modificará, debe estar declarada dentro de una función que tenga la etiqueta de `@Composable`, además de que llevará la palabra reservada `remember` para leer los cambios en tiempo real.

```
@Composable
fun Content() {
    var likes by remember {
        mutableStateOf( value: 0 )
    }
    Row(
        modifier = Modifier
            .fillMaxWidth(),
        horizontalArrangement = Arrangement.Center,
        verticalAlignment = Alignment.CenterVertically
    ) {
        Button(onClick = {
            likes++
        }) {
            Text(text = nombre)
        }
        Spacer(modifier = Modifier.width(10.dp))
        Resultado(likes = likes)
    }
}
```



**GRACIAS POR
LA ATENCIÓN**