Android Parte II

Índice

1. Scaffold y Navegación

Scaffold

El componente Scaffold nos da una estructura predeterminada para todas nuestras pantallas, de forma que podremos definir una única vez como será nuestro TopBar y nuestro BottomBar y se extenderá por toda la aplicación. Además, nos facilitará muchísimo la navegación al crear aplicaciones donde los botones para movernos entre pantallas están en la parte inferior y únicamente cambia el contenido de la vista central.

Al crear un proyecto el "cascarón" que tenemos sobre nuestra pantalla, es el Scaffold.

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            ScaffoldTheme {
                Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding ->
                    Greeting(
                        name = "Android",
                        modifier = Modifier.padding(innerPadding)
                    )
                }
            }
        }
    }
}

@Composable
fun Greeting(name: String, modifier: Modifier = Modifier) {
    Text(
        text = "Hello $name!",
        modifier = modifier
    )
}

@Preview(showBackground = true)
@Composable
fun GreetingPreview() {
    ScaffoldTheme {
        Greeting("Android")
    }
}
```

Lo primero que debería de llamar la atención es el "innerPadding". Este es un valor calculado automáticamente por el Scaffold, que tendremos que aplicar al contenido que haya dentro, ya que según lo que añadamos en el Scaffold, nuestra pantalla "real" empezará más arriba o más abajo, si lo tenemos en cuenta desde la parte superior izquierda. El componente Scaffold se puede utilizar con varios componentes para hacerlo funcionar correctamente.

- **TopAppBar**. Componente que podremos añadir a nuestro Scaffold, para mostrar una barra superior con títulos o navegación.

- **BottomAppBar.** Componente que podremos añadir a nuestro Scaffold para mostrar una barra inferior, sobre todo lo utilizaremos como menú de navegación.

- **NavHost.** Componente utilizado para crear la lógica de navegación entre pantallas

Por tanto, la estructura de un Scaffold tendrá **tres** argumentos, que por simplicidad, aparecen vacíos en este ejemplo.

- **topBar = {}**
- **bottomBar = {}**
- **Modifier.fillMaxSize().** Mediante este modificador conseguimos que nuestro Scaffold ocupe todo el espacio de la pantalla.
- Dentro del cuerpo de nuestro Scaffold (entre las llaves {}), tendremos normalmente un contenedor que nos facilite la disposición de los elementos en la pantalla. Este contenedor, tendrá que aplicar el

```
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.enableEdgeToEdge
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.padding
import androidx.compose.material3.Scaffold
import androidx.compose.material3.Text
import androidx.compose.ui.Modifier


class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            Tema2Theme {
                Scaffold(
                    topBar = {},
                    bottomBar = {},
                    modifier = Modifier.fillMaxSize()) { innerPadding ->
                       Box(Modifier.padding(innerPadding)){
                               Text("Texto de ejemplo", Modifier.padding())
                       }

                }

            }
        }
    }
}
```

“innerPadding” calculado por el Scaffold para que el contenido se vea bien ajustado en la pantalla.

TopAppBar

Como se comenta anteriormente, podemos utilizar el Componente TopAppBar como header, dentro de Scaffold.

```
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.enableEdgeToEdge
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.padding
import androidx.compose.material3.ExperimentalMaterial3Api
import androidx.compose.material3.Scaffold
import androidx.compose.material3.Text
import androidx.compose.material3.TopAppBar
import androidx.compose.ui.Modifier


class MainActivity : ComponentActivity() {
    @OptIn(ExperimentalMaterial3Api::class)
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            Tema2Theme {
                Scaffold(
                    topBar = {
                        TopAppBar(
                            title = { Text("Tíituto") }
                        )
                    },
                    bottomBar = {},
                    modifier = Modifier.fillMaxSize()) { innerPadding ->
                        Box(Modifier.padding(innerPadding)){
                            Text("Texto de ejemplo", Modifier.padding())
                        }

                }

            }
        }
    }
}
```

En el argumento **topBar = {},** hemos añadido el Componente TopAppBar y dentro de este componente hemos añadido el título como { Text("Título") }.

¿Por qué añadimos las llaves a la asignación del título? El argumento 'título', dentro del componente TopAppBar es de tipo **Composable lambda,** lo que nos da flexibilidad para añadir cualquier tipo de Componente al título.

Vamos a añadir algo de color a nuestra TopBar. Otro argumento que puede tener TopAppBar es *colors*. Le asignaremos la clase *TopAppBarDefaults*, que nos dejará sobreescribir el color del contenedor como se muestra en el siguiente fragmento de código.

```
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.enableEdgeToEdge
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.padding
import androidx.compose.material3.ExperimentalMaterial3Api
import androidx.compose.material3.Scaffold
import androidx.compose.material3.Text
import androidx.compose.material3.TopAppBar
import androidx.compose.material3.TopAppBarDefaults
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color


class MainActivity : ComponentActivity() {
    @OptIn(ExperimentalMaterial3Api::class)
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            Tema2Theme {
                Scaffold(
                    topBar = {
                        TopAppBar(
                            colors = TopAppBarDefaults.topAppBarColors(
                                containerColor = Color(0xFFCDE7B0)
                            ),
                            title = { Text("Título") }
                        )
                    },
                    bottomBar = {},
                    modifier = Modifier.fillMaxSize()) { innerPadding ->
                        Box(Modifier.padding(innerPadding)){
                            Text("Texto de ejemplo", Modifier.padding())
                        }

                }

            }
        }
    }
}
```


8:06
Título
Texto de ejemplo
1:1

CenterAlignedTopAppBar

Otro tipo de TopBar que podemos utilizar es *CenterAlignedTopAppBar* el cual nos dejará ordenar las opciones de la barra de mejor manera. Igual que en el anterior, tendremos *title* y *colors*, que podemos modificar como en el caso anterior.

Añadimos dos argumentos más.

- **navigationIcon**. Donde podremos especificar el componente que queramos para que aparezca a la *izquierda* de nuestra barra. En este caso, se ha añadido un Box circular que modificaremos más adelante.

- **Actions.** Donde podremos especificar varios componentes, que su comportamiento por defecto será como si estuvieran dentro de una Row, por lo que se alinearán horizontalmente. En este caso, se han añadido dos Box circulares que modificaremos más adelante.

```
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.enableEdgeToEdge
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.width
import androidx.compose.foundation.shape.CircleShape
import androidx.compose.material3.CenterAlignedTopAppBar
import androidx.compose.material3.ExperimentalMaterial3Api
import androidx.compose.material3.Scaffold
import androidx.compose.material3.Text
import androidx.compose.material3.TopAppBarDefaults
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.unit.dp


class MainActivity : ComponentActivity() {
    @OptIn(ExperimentalMaterial3Api::class)
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            Tema2Theme {
                Scaffold(
                    topBar = {
                        CenterAlignedTopAppBar(
                            colors = TopAppBarDefaults.topAppBarColors(
                                containerColor = Color(0xFFCDE7B0)
                            ),
                            title = { Text("Título") },
                            navigationIcon = {
                                Box(Modifier
                                    .clip(shape = CircleShape)
                                .height(35.dp)
                                .width(35.dp)
                                .background(Color.Black)
                                ){}
                            },
                            actions = {
                                Box(Modifier
                                    .clip(shape = CircleShape)
                                    .height(35.dp)
                                    .width(35.dp)
                                    .background(Color.Black)
                                ){}
                                Box(Modifier
                                    .clip(shape = CircleShape)
                                    .height(35.dp)
                                    .width(35.dp)
                                    .background(Color.Black)
                                ){}
                            }

                        )
                    },
                    bottomBar = {},
                    modifier = Modifier.fillMaxSize()) { innerPadding ->
                        Box(Modifier.padding(innerPadding)){
                            Text("Texto de ejemplo", Modifier.padding())
                        }

                }

            }
        }
    }
}
```

BottomAppBar

El siguiente componente que veremos es BottomAppBar que, como su nombre indica, nos facilitará la tarea de añadir una barra en la parte inferior de la pantalla. Podemos crearla directamente sin ningún argumento, pero no veremos nada en la pantalla, por lo que como mínimo añadiremos el color con el argumento **containerColor.**

```
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.enableEdgeToEdge
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.width
import androidx.compose.foundation.shape.CircleShape
import androidx.compose.material3.BottomAppBar
import androidx.compose.material3.CenterAlignedTopAppBar
import androidx.compose.material3.ExperimentalMaterial3Api
import androidx.compose.material3.Scaffold
import androidx.compose.material3.Text
import androidx.compose.material3.TopAppBarDefaults
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.unit.dp


class MainActivity : ComponentActivity() {
    @OptIn(ExperimentalMaterial3Api::class)
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            Tema2Theme {
                Scaffold(
                    topBar = {
                        CenterAlignedTopAppBar(
                            colors = TopAppBarDefaults.topAppBarColors(
                                containerColor = Color(0xFFCDE7B0)
                            ),
                            title = { Text("Título") },
                            navigationIcon = {
                                Box(Modifier
                                    .clip(shape = CircleShape)
                                .height(35.dp)
                                .width(35.dp)
                                .background(Color.Black)
                                ){}
                            },
                            actions = {
                                Box(Modifier
                                    .clip(shape = CircleShape)
                                    .height(35.dp)
                                    .width(35.dp)
                                    .background(Color.Black)
                                ){}
```

```
                                Box(Modifier
                                    .clip(shape = CircleShape)
                                    .height(35.dp)
                                    .width(35.dp)
                                    .background(Color.Black)
                                ){}
                            }

                        )
                    },
                    bottomBar = {
                        BottomAppBar(
                            containerColor = Color(0xFFCDE7B0)
                        ) {}

                    },
                    modifier = Modifier.fillMaxSize()) { innerPadding ->
                        Box(Modifier.padding(innerPadding)){
                            Text("Texto de ejemplo", Modifier.padding())
                        }

                }

            }
        }
    }
}
```

En el cuerpo de nuestro *BottomAppBar*, podremos añadir los componentes que queramos y se dispondrán como si estuvieran en una *Row*, de izquierda a derecha. En las *Row*, si no indicamos lo contrario, los elementos aparecen uno al lado del otro, por lo que haremos una pequeña modificación añadiendo un **horizontalArrangement** y que los elementos que añadamos ocupen todo el espacio.

El resto del código sigue igual, se añade sólo esta parte por simplificar.

```
bottomBar = {
    BottomAppBar(
        containerColor = Color(0xFFCDE7B0)
    )
    {
        Row (Modifier.fillMaxWidth(),
            horizontalArrangement = Arrangement.SpaceEvenly)
        {
            Box(Modifier
                .clip(shape = CircleShape)
                .height(35.dp)
                .width(35.dp)
                .background(Color.Black)
            ){}
            Box(Modifier
                .clip(shape = CircleShape)
                .height(35.dp)
                .width(35.dp)
                .background(Color.Black)
            ){}
            Box(Modifier
                .clip(shape = CircleShape)
                .height(35.dp)
                .width(35.dp)
                .background(Color.Black)
            ){}
        }
    }
},
```


8:06
Título
Texto de ejemplo
1:1

Navegación

En este apartado vamos a utilizar un componente muy importante utilizado para navegar entre pantallas. Primero, crearemos tres componentes nuevos que actuarán como nuestras pantallas y que cambiarán según el Tab que pulsemos dentro de nuestra aplicación.

Por lo tanto, creamos en nuestra ruta **ui/components** las tres pantallas.

Screen1.kt

```
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier

@Composable
fun Screen1(){
    Box(Modifier.fillMaxSize(),
        contentAlignment = Alignment.Center){
        Text("Pantalla 1")
    }
}
```

Screen2.kt

```
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier

@Composable
fun Screen2(){
    Box(Modifier.fillMaxSize(),
        contentAlignment = Alignment.Center){
        Text("Pantalla 2")
    }
}
```

Screen3.kt

```
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier

@Composable
fun Screen3(){
    Box(Modifier.fillMaxSize(),
```

```
            contentAlignment = Alignment.Center){
            Text("Pantalla 3")
        }
    }
```

Los componentes que utilizaremos para controlar la navegación serán **NavHost** y **NavHostController.** NavHost será el componente en el que definamos las rutas de nuestras pantallas y NavController el que se encargará de movernos entre ellas.

NavHost tendrá dos argumentos.

- **navController**, que utilizará para gestionar la navegación.
- **startDestination,** que definirá cual será la pantalla por defecto cuando llamemos a NavHost.

Para gestionar la navegación, crearemos un componente nuevo en nuestra **MainActivity**

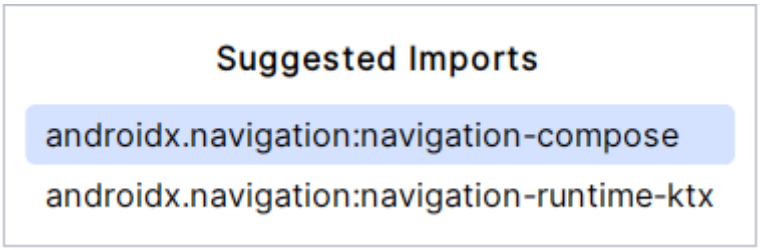
Creamos el componente en **MyNavControllerCustom,**

```
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.navigation.compose.NavHost
import androidx.navigation.compose.composable
import androidx.navigation.compose.rememberNavController
[…]
class MainActivity : ComponentActivity() {

    @OptIn(ExperimentalMaterial3Api::class)
    override fun onCreate(savedInstanceState: Bundle?) {

        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            Tema2Theme {
                val navController = rememberNavController()
                Scaffold(
                    topBar = {
                       /* Añadir código de topBar*/
                    },
                    bottomBar = {
                       /* Añadir código de bottomBar*/
                    },
                    modifier = Modifier.fillMaxSize()) { innerPadding ->
                        MyNavControllerCustom(navController,
Modifier.padding(innerPadding))
                }

            }
        }
    }
}
```

```
@Composable
fun MyNavControllerCustom(navController: NavHostController, modifier: Modifier){

    NavHost(navController = navController, startDestination = "screen1",
        modifier = modifier){
        composable("screen1"){ Screen1() }
        composable("screen2"){ Screen2() }
        composable("screen3"){ Screen3() }
    }
}
```

composable es la forma que tenemos de definir las rutas dentro de nuestro NavHost. Definiremos la ruta como un String ("screen1","screen2"…) y en el cuerpo añadiremos el componente al que queremos navegar (Screen1(),Screen2()…)

Al importar *composable* habrá que elegir la marcada a continuación, además, se añadirán dependencias al proyecto de forma automática.



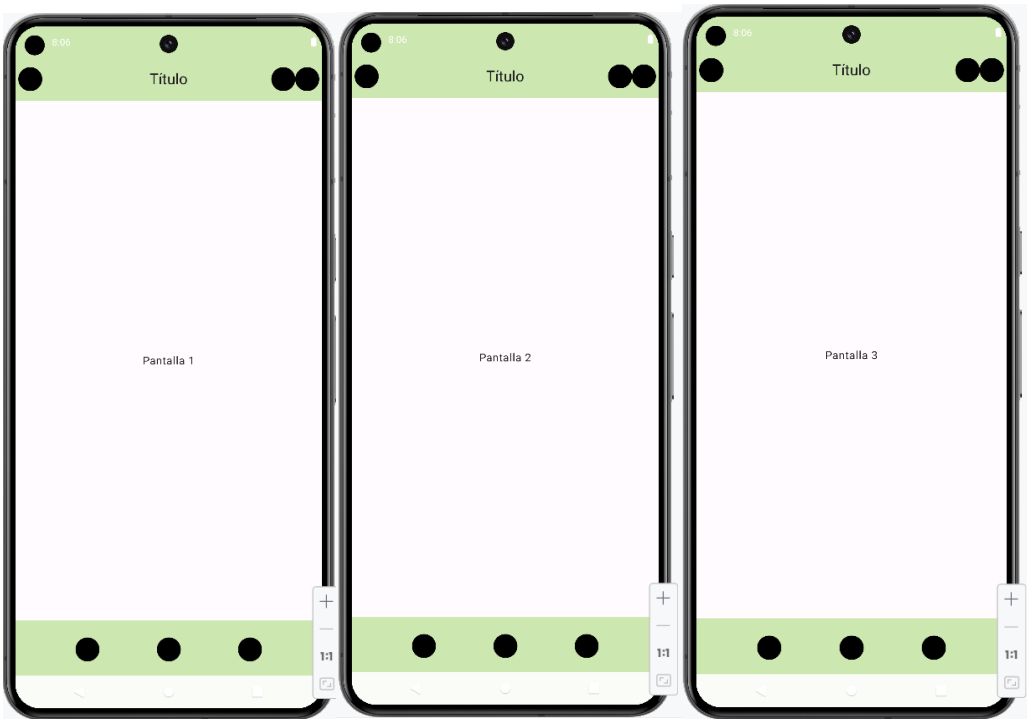
Una vez creado el NavController mediante *rememberNavController()*, creado el NavHost y definido las rutas crearemos especificaremos a donde queremos navegar.

En el código de nuestro BottomAppBar teníamos tres *Box* que, por defecto, no se pueden clickar ya que no tienen evento asociado. Podemos cambiar este comportamiento mediante el modificador *clickable* y añadiéndole el código necesario.

Aquí, añadiremos **navController.navigate("screenX")** y al hacer click, cambiará la pantalla.

```
bottomBar = {
    BottomAppBar(
        containerColor = Color(0xFFCDE7B0)
    )
    {

        Row (Modifier.fillMaxWidth(),
            horizontalArrangement = Arrangement.SpaceEvenly)
        {
            Box(Modifier
                .clip(shape = CircleShape)
                .height(35.dp)
                .width(35.dp)
                .background(Color.Black)
                .clickable {
```

```
                    navController.navigate("screen1")
                }
            ){}
            Box(Modifier
                .clip(shape = CircleShape)
                .height(35.dp)
                .width(35.dp)
                .background(Color.Black)
                .clickable {
                    navController.navigate("screen2")
                }
            ){}
            Box(
                Modifier
                .clip(shape = CircleShape)
                .height(35.dp)
                .width(35.dp)
                .background(Color.Black)
                .clickable {
                        navController.navigate("screen3")
                    }
            ){}
        }
```



2. Iconos

Material Icons

Estos ejemplos sobre iconos se van a hacer en un nuevo proyecto

Compose trae incorporado soporte para iconos de **Material Design.** Es decir, podrás utilizar iconos importándolos directamente, sin necesidad de añadirlos directamente en tu proyecto.

Creamos una *Row* con varios iconos en su interior utilizando el Componente *Icon* que deberá tener tres argumentos.

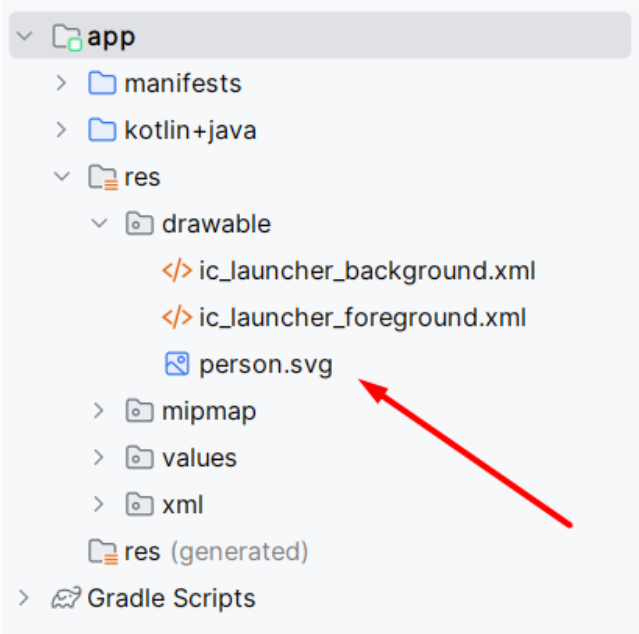
- **imageVector.** Será el icono que queramos mostrar. Utilizaremos el componente Icons.Filled/Outlined/TwoTone.NombreIcono. La lista de iconos está disponible aquí.
- **contentDescription.** Descripción del icono que mostremos.
- **tint.** Color que queremos para nuestro icono, por defecto, negro.

```
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.enableEdgeToEdge
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.size
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.AccountCircle
import androidx.compose.material.icons.outlined.AccountCircle
import androidx.compose.material.icons.twotone.AccountCircle
import androidx.compose.material3.Icon
import androidx.compose.material3.Scaffold
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.unit.dp

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            IconosTheme {
                Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding ->
                    MyIcon(Modifier.padding(innerPadding))
                }
            }
        }
    }
}

@Composable
fun MyIcon(modifier:Modifier){
    Row(){
        Icon(
            imageVector = Icons.Filled.AccountCircle,
```

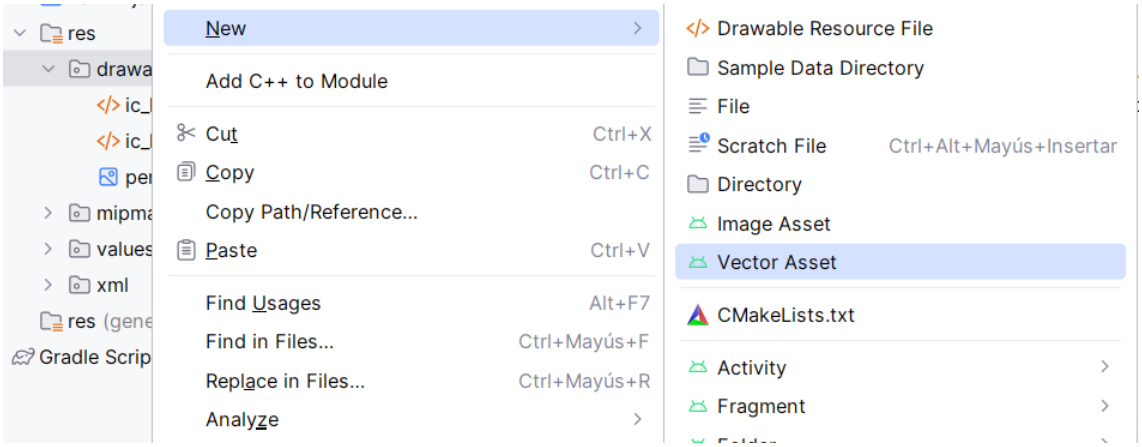
```
                contentDescription = "favorite",
                tint = Color.Red,
                modifier = modifier.size(48.dp)
            )
g
            Icon(
                imageVector = Icons.Outlined.AccountCircle,
                contentDescription = "favorite",
                tint = Color.Red,
                modifier = modifier.size(48.dp)
            )
            Icon(
                imageVector = Icons.TwoTone.AccountCircle,
                contentDescription = "favorite",
                tint = Color.Red,
                modifier = modifier.size(48.dp)
            )
        }
}
```

Iconos personalizables

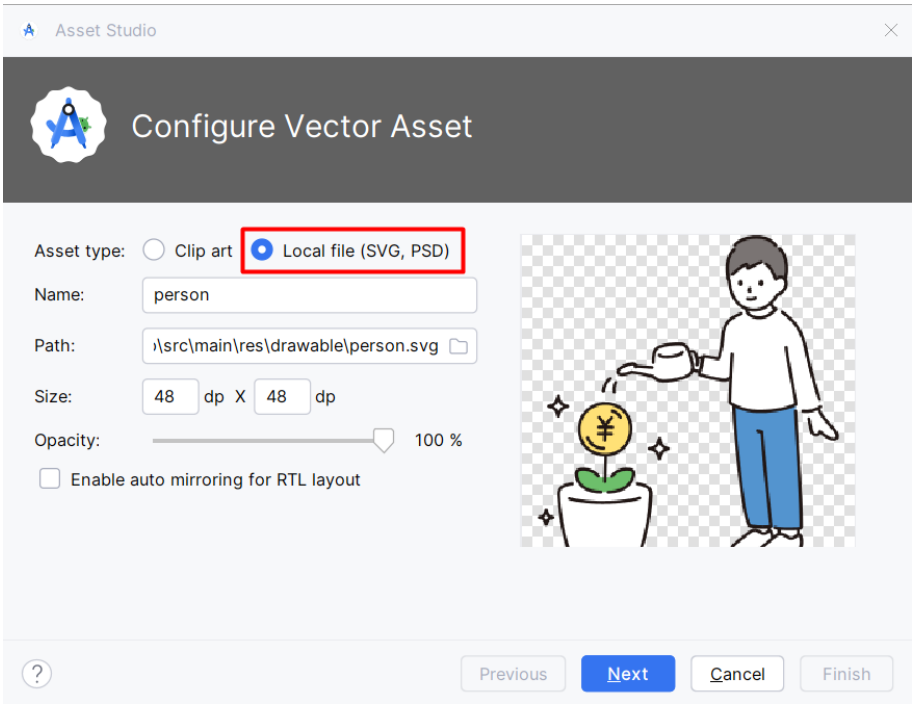
A parte de los iconos que nos da Jetpack Compose podemos añadir los que queramos a nuestro proyecto. Descargamos el icono que queramos desde svgrepo (o desde donde queramos) y lo añadimos a la ruta **app/res/drawable.**

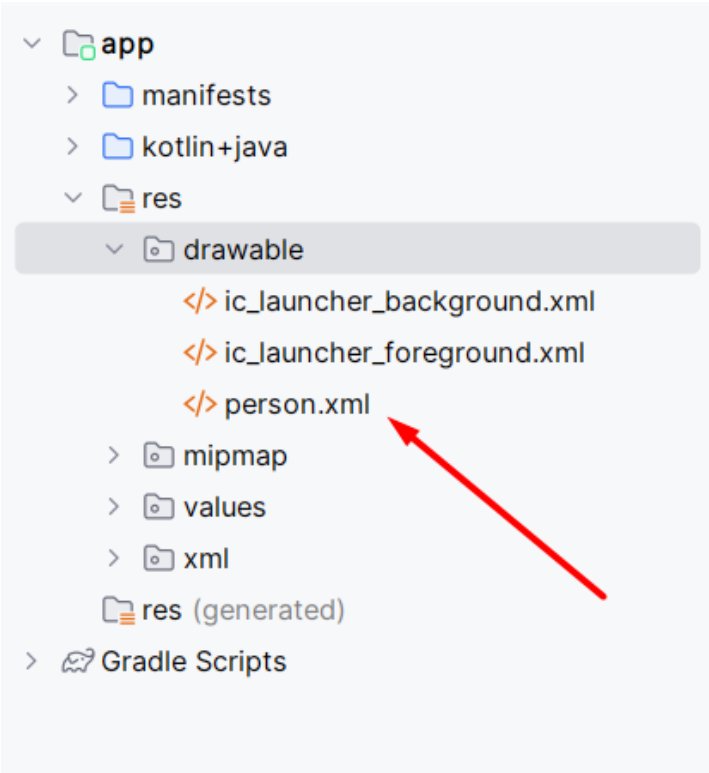


Hacemos click derecho en *drawable* y vamos a *New > Vector Asset*



Elegimos *Local File* y damos a *Siguiente* y luego *Finish*. Nos habrá creado un fichero .xml que será el que utilizaremos como icono, por lo que podemos eliminar el .svg.


Asset Studio
Configure Vector Asset
Asset type:
Clip art
Local file (SVG, PSD)
Name:
person
Path:
\src\main\res\drawable\person.svg
Size:
48
dp X
48
dp
Opacity:
100 %
Enable auto mirroring for RTL layout
Previous
Next
Cancel
Finish


app
manifests
kotlin+java
res
drawable
ic_launcher_background.xml
ic_launcher_foreground.xml
person.xml
mipmap
values
xml
res (generated)
Gradle Scripts

```
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.enableEdgeToEdge
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.size
import androidx.compose.material3.Icon
import androidx.compose.material3.Scaffold
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.unit.dp
import com.example.iconos.ui.theme.IconosTheme

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            IconosTheme {
                Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding ->
                    MyIcon(Modifier.padding(innerPadding))
                }
            }
        }
    }
}

@Composable
fun MyIcon(modifier:Modifier){
    Row(){
        Icon(
            painter = painterResource(R.drawable.person),
            contentDescription = "favorite",
            modifier = modifier.size(48.dp)
        )
    }
}
```

El argumento de *Icon* en este caso es:

- **painter**. Recibe la función painterResource que se utiliza para cargar imágenes vectoriales. Dentro, R. es una clase que tiene referencia a los recursos del proyecto. Por tanto, painterResource carga la imagen que está en drawable/person.

3. Imágenes

El uso de imágenes es muy parecido al de iconos personalizables. Añadimos nuestra imagen a la ruta **app/res/drawable.**

Lo que cambia respecto a los iconos es el argumento contentScale.

- **ContentScale.Fit**. La imagen se redimensionará para ocupar todo el espacio de su contenedor padre, pero sin perder su proporción de aspecto. Esta es la opción por defecto.
- **ContentScale.Crop.** La imagen se redimensionará para ocupar todo el espacio, pero se recortarán las partes que no quepan en el contenedor.

```
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.enableEdgeToEdge
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.padding
import androidx.compose.material3.Scaffold
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import com.example.iconos.ui.theme.IconosTheme

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            IconosTheme {
                Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding ->
                    MyImage(Modifier.padding(innerPadding))
                }
            }
        }
    }
}

@Composable
fun MyImage(modifier:Modifier){
    Box(modifier)
        Image(
            painter = painterResource(R.drawable.space),
            contentDescription = "",
            contentScale = ContentScale.Fit
        )
}
```

4. LazyColumn

Este componente es una versión optimizada de una columna que permite crear listas de elementos de manera eficiente ya que cargará los elementos sólo cuando sean visibles en la pantalla.

Para utilizar *LazyColumn*, tendremos que definir sus elementos internos como *item { }* y dentro de ellos, los componentes que queramos. Un ejemplo sencillo con un único items sería el siguiente.

```
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.enableEdgeToEdge
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material3.Scaffold
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.unit.dp
import com.example.lazycolumn.ui.theme.LazyColumnTheme

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            LazyColumnTheme {
                Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding ->
                    MyLazyColumn(Modifier.padding(innerPadding))
                }
            }
        }
    }
}
@Composable
fun MyLazyColumn(modifier: Modifier){

    LazyColumn (modifier = modifier.padding(horizontal = 15.dp).fillMaxWidth()) {
        item{
            Column(Modifier.fillMaxWidth()){
                Text("Historia", Modifier.align(Alignment.CenterHorizontally))
                Box(
                    Modifier
                        .fillMaxWidth()
                        .height(150.dp)
                        .clip(shape = RoundedCornerShape(20.dp))
```

```
                ){
                    Image(
                        painter = painterResource(R.drawable.history),
                        contentDescription = "",
                        contentScale = ContentScale.Crop

                    )

                }
            }
        }

    }
}
```

La idea de los *LazyColumn* es que sea óptimo en la carga de muchos datos por lo que utilizarlo con un único dato carece de sentido.

Vamos a crear un ejemplo con diez items. La idea será la misma, tener una LazyColumn que dentro tendrá *items* pero esta vez crearemos una lista de donde cogeremos los datos.

Lo primero que haremos será crear un *data class,* clase que utilizaremos para almacenar los datos de cada uno de los items que veremos en pantalla.

Añadimos en **MainActivity.kt** el data class **Asignatura**, que tendrá un String y un Int, que utilizaremos para acceder a las imágenes.

Además, en nuestro componente **MyLazyColumn** crearemos una lista con varios objetos de tipo **Asignatura** de forma que podremos iterar la lista dentro del LazyColumn y mostrarlas.

```
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.enableEdgeToEdge
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material3.Scaffold
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.unit.dp
import com.example.lazycolumn.ui.theme.LazyColumnTheme

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            LazyColumnTheme {
                Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding ->
                    MyLazyColumn(Modifier.padding(innerPadding))
                }
            }
        }
    }
}
@Composable
fun MyLazyColumn(modifier: Modifier){
```

```
    val itemList = listOf(
        Asignatura("Historia", R.drawable.history),
        Asignatura("Matematicas", R.drawable.matematicas),

    )
    LazyColumn (modifier = modifier.padding(horizontal = 15.dp).fillMaxWidth()) {
        items(itemList) { asign ->

            Column(Modifier.fillMaxWidth()){
                Text(asign.asignatura,
Modifier.align(Alignment.CenterHorizontally))
                Box(
                    Modifier
                        .fillMaxWidth()
                        .height(150.dp)
                        .clip(shape = RoundedCornerShape(20.dp))
                ){
                    Image(
                        painter = painterResource(asign.imageId),
                        contentDescription = "",
                        contentScale = ContentScale.Crop,
                    )

                }
            }
            Spacer(Modifier.height(20.dp))
        }

    }
}

data class Asignatura(val asignatura: String, val imageId: Int)
```

- Creamos una lista de tipo Asignatura en la variable itemList = listOf()…
- Dentro del LazyColumn utilizamos *items* que tendrá como argumento nuestra lista sobre la que iteraremos.
- En cada iteración, *asign* cogerá el valor de una asignatura, que dentro de la columna la utilizamos mostrando el texto y la imagen, accediendo a sus atributos como **asign.asignatura** y **asign.imageId**


Historia
Matematicas
1:1