Jetpack Compose

Jetpack Compose fue anunciado por Google en 2019 como un nuevo framework para la construcción de interfaces de usuario en Android.

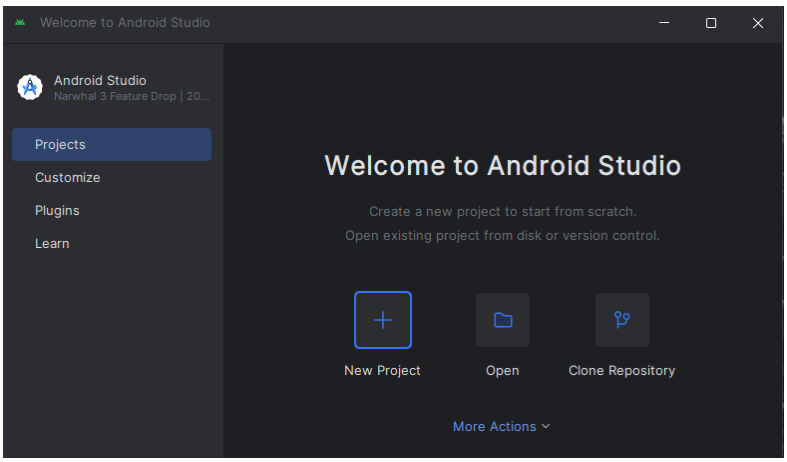
La primera versión estable, Jetpack Compose 1.0, se lanzó en julio de 2021, marcando un hito importante en la evolución del desarrollo de aplicaciones Android.

Jetpack Compose es un API desarrollado en Kotlin y para ser consumido solo desde Kotlin, para el desarrollo de interfaces visuales.

Para sacar partida de Jetpack Compose se requiere tener los conceptos claros del lenguaje de programación Kotlin, si no es así deberá aprender los conceptos fundamentales del lenguaje Kotlin.

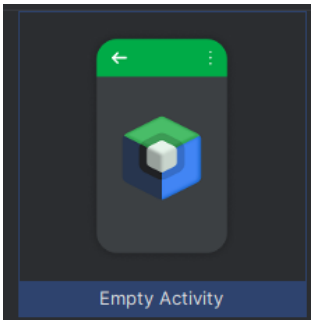
Creación de un Hola Mundo con Compose.

Las últimas versiones de Android Studio ya nos permiten crear un esqueleto básico de una aplicación que utilice el API de Compose, para ello seleccionamos:

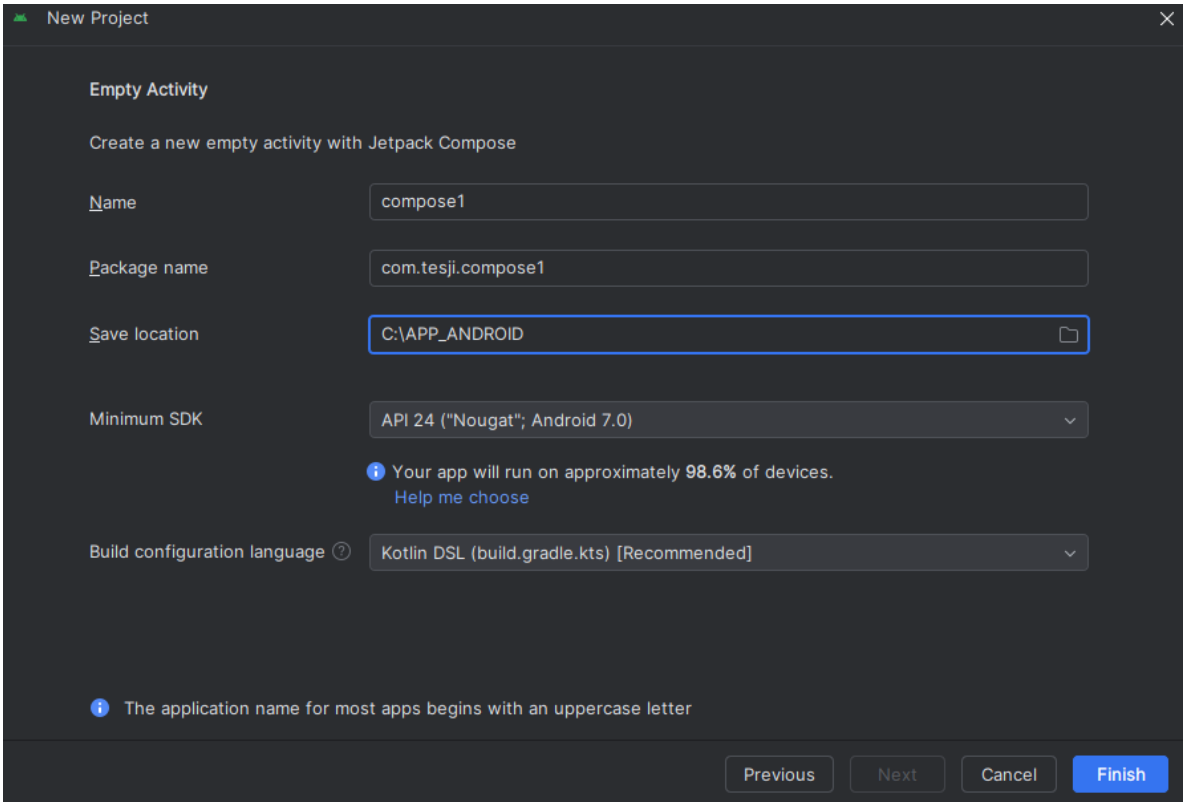


Project\new Project

Selecciónar:

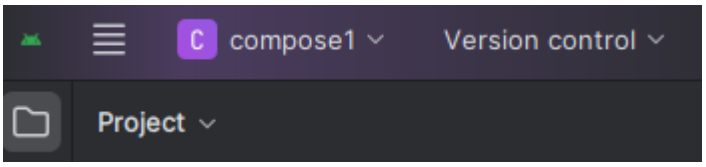


Crear una carpeta en C:\APP_ANDROID que permita identificar los proyectos realizados de manera más eficiente y rápida, agregar los siguientes datos:

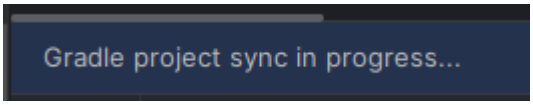


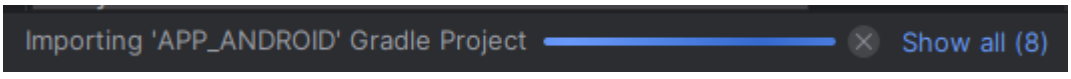
Dar clic en Finish.

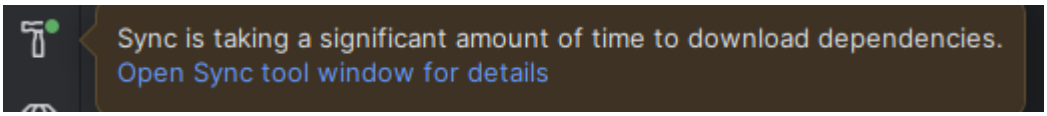
Comenzará a crearse el proyecto:



Así también comenzara el proceso de sincronización de Gradle:

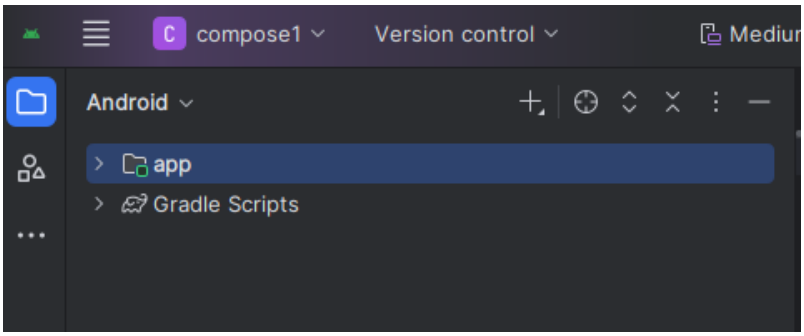






Cuando termine finalmente nos mostrara nuestro directorio de APP

Al mostrar la app estaremos listos para crear nuestro primer aplicación:



1. Se nos genera un esqueleto básico con la siguiente sintaxis:

```
package com.tesji.compose1

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.enableEdgeToEdge
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.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import com.tesji.compose1.ui.theme.Compose1Theme

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            Compose1Theme {
                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() {
    Compose1Theme {
        Greeting("Android")
    }
}
```

2. Procedemos a borrar código que no es necesario, luego el código mínimo debe ser:

```
package com.tesji.compose1

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.enableEdgeToEdge
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.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import com.tesji.compose1.ui.theme.Compose1Theme

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {

        }
    }
}
```

3. Jetpack Compose se basa en funciones que admiten composición. Estas funciones permiten definir la interfaz de usuario de manera programática y no visual como el modelo anterior.

Por ejemplo, para mostrar un texto en pantalla debemos llamar a la función "Text" y pasar el parámetro "text". También es importante hacer dicha llamada dentro de la función 'setContent':

```
package com.tesji.compose1

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.enableEdgeToEdge
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.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import com.tesji.compose1.ui.theme.Compose1Theme

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            Text(text = "Hola Mundo")
        }
    }
}
```

Es decir, llamamos a la función 'Text' (que es una función '@Composable', luego veremos que hay muchas de estas funciones y también podremos crear las nuestras):

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    enableEdgeToEdge()
    setContent {
        Text(text = "Hola Mundo")
    }
```

4. Si ejecutamos ya tenemos nuestra etiqueta impresa en pantalla, solo llamando la función 'Text' y pasando el mensaje a mostrar:

```
package com.tesji.compose1

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.enableEdgeToEdge
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.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import com.tesji.compose1.ui.theme.Compose1Theme

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            Text(text = "Hola Mundo")
        }
    }
}
```

Agrega un dispositivo virtual, y corre la aplicación.

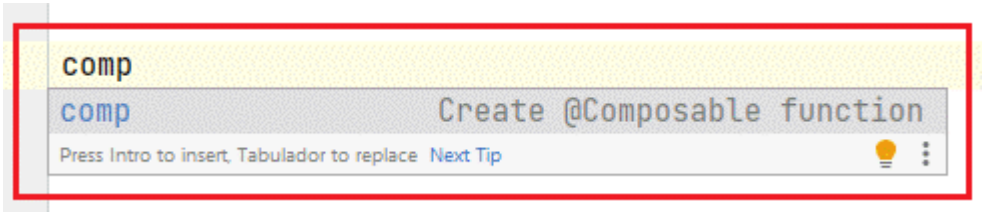
5. Lo más común es que evitemos implementar el código todo dentro de la llamada a 'setContent' y procedamos a crear una función @Composable:

```
setContent {
        HolaMundo("Hola mundo")
    }
```

Para crear una función @Composable debemos anteceder dicha anotación:

```
@Composable
fun HolaMundo(mensaje:String) {
    Text(text = "$mensaje")
}
```

Como será muy común la creación de funciones composables, Android Studio nos permite su creación escribiendo en el editor 'comp':



Código final:

```
package com.tesji.compose1

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.enableEdgeToEdge
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.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import com.tesji.compose1.ui.theme.Compose1Theme

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            HolaMundo("Hola mundo")
        }
    }
}

@Composable
fun HolaMundo(mensaje:String) {
    Text(text = "$mensaje")
}
```