Nizan Díaz

Informe sobre el CodeLab

Contenido

Nizan Díaz

PRIMERA TANDA DE EJERCICIOS

Ejercicios 1.

1

2

3

Ejercicio 2.

```kotlin
fun main() {
    println("I'm")
    println("learning")
    println("Kotlin!")
}
```

```
I'm
learning
Kotlin!
```

Target platform: JVM Running on kotlin v: 1.9.10

Ejercicio 3

```kotlin
fun main() {
    println("Monday")
    println("Tuesday")
    println("Wednesday")
    println("Thursday")
    println("Friday")
}
```

```
Monday
Tuesday
Wednesday
Thursday
Friday
```

Nizan Díaz

Ejercicio 4

```kotlin
fun main() {
    println("Tomorrow is rainy")
}
```

```
Tomorrow is rainy
```

Ejercicio 5

```kotlin
fun main() {
    println("There is a chance of snow")
}
```

```
There is a chance of snow
```

Target platform: JVM Running on kotlin v. 1.9.10

Ejercicio 6

```kotlin
fun main() {
    println("Cloudy")
    println("Partly Cloudy")
    println("Windy")
}
```

```
Cloudy
Partly Cloudy
Windy
```

Target platform: JVM Running on kotlin v. 1.9.10

Nizan Díaz

Ejercicio 7

```
fun main() {
    println("How's the weather today?")
}
```

```
How's the weather today?
```

SEGUNDA TANDA DE EJERCICIOS

PROBLEMAS PRÁCTICOS CONCEPTOS BÁSICOS EJERCICIOS

Ejercicio 1.

```
fun main() {
    println("Use the val keyword when the value doesn't change."
    println("Use the var keyword when the value can change.")
    println("When you define a function, you define the paramete
    println("When you call a function, you pass arguments for th
}
```

```
Use the val keyword when the value doesn't change.
Use the var keyword when the value can change.
When you define a function, you define the parameters that can be
When you call a function, you pass arguments for the parameters.
```

Ejercicio 2.

El println que se nos ofrece a corregir presenta una apertura con (, pero sin embargo, su cierre es con }. Además el tipo de comilla de apertura y cierre son distintas.

```
fun main() {
    println("New chat message from a friend")
}
```

```
New chat message from a friend
```

Nizan Díaz

Ejercicio 3.

Lo que ocurre es que para atribuirle valor a las variables está implementando val cuando esta se emplea cuando esperamos que nuestra variable no cambie, por ello deberíamos de implementar var.

```
fun main() {
    var discountPercentage: Int = 0
    var offer: String = ""
    val item = "Google Chromecast"
    discountPercentage = 20
    offer = "Sale - Up to $discountPercentage% discount on $iter

    println(offer)
}
```

```
Sale - Up to 20% discount on Google Chromecast! Hurry up!
```

Ejercicio 4

No se realiza la suma inicial debido a que nuestras variables están en formato String en vez de en formato int para que puedan ser operables, no obstante se hace una concatenación de las cadenas.

```
fun main() {
    val numberOfAdults = "20"
    val numberOfKids = "30"
    val total = numberOfAdults + numberOfKids
    println("The total party size is: $total")
}
```

```
The total party size is: 2030
```

Para que realice la suma debe de atribuirle valores númericos en vez de valores textuales.

```
fun main() {
    val numberOfAdults = 20
    val numberOfKids = 30
    val total = numberOfAdults + numberOfKids
    println("The total party size is: $total")
}
```

```
The total party size is: 50
```

Nizan Díaz

Ejercicio 5

El código proporcionado nos indicará la suma a realizar, pero sin operar dicha suma debido a que totalSalary está sumando en formato cadena de texto.

```
fun main() {
    val baseSalary = 5000
    val bonusAmount = 1000
    val totalSalary = "$baseSalary + $bonusAmount"
    println("Congratulations for your bonus! You will receive a
}
```

```
ulations for your bonus! You will receive a total of 5000 + 1000 (
```

Target platform: JVM Running on kotlin v. 1.9.10

Para realizar ahora la operación debemos eliminar la cadena de caracteres de totalSalary, consiguiente de esta forma que nos haga la suma de 5000 + 1000, es decir, devolviéndonos 6000.

```
fun main() {
    val baseSalary = 5000
    val bonusAmount = 1000
    val totalSalary = baseSalary + bonusAmount
    println("Congratulations for your bonus! You will receive a
}
```

```
or your bonus! You will receive a total of 6000 (additional bonus)
```

Nizan Díaz

Ejercicio 6

Para arregla el código del primer paso creamos una nueva variable result.

```
fun main() {
    val firstNumber = 10
    val secondNumber = 5
    val result = firstNumber + secondNumber
    println("$firstNumber + $secondNumber = $result")
}
```

```
10 + 5 = 15
```

Paso 2: Implementamos la función add con sus parámetros.

```
fun main() {
    val firstNumber = 10
    val secondNumber = 5
    val thirdNumber = 8

    val result = add(firstNumber, secondNumber)
    val anotherResult = add(firstNumber, thirdNumber)

    println("$firstNumber + $secondNumber = $result")
    println("$firstNumber + $thirdNumber = $anotherResult")
}
fun add(n1 : Int, n2 : Int): Int{
    return n1 + n2
}
// Define add() function below this line
```

```
10 + 5 = 15
10 + 8 = 18
```

Paso 3: Mi función subtract realiza una multiplicación de los números de entrada.

```
fun main() {
    val firstNumber = 10
    val secondNumber = 5
    val thirdNumber = 8

    val result = add(firstNumber, secondNumber)
    val anotherResult = add(firstNumber, thirdNumber)
    val result2 = subtract(firstNumber, secondNumber)
    val anotherResult2 = subtract(firstNumber, thirdNumber)

    println("$firstNumber + $secondNumber = $result")
    println("$firstNumber + $thirdNumber = $anotherResult")
    println("$firstNumber + $secondNumber = $result2")
    println("$firstNumber + $thirdNumber = $anotherResult2")
}
fun add (n1 : Int, n2: Int) : Int{
    return n1+n2
}
fun subtract (n1 : Int, n2: Int) : Int{
    return n1*n2
}
// Define add() function below this line
```

```
10 + 5 = 15
10 + 8 = 18
10 + 5 = 50
10 + 8 = 80
```

Ejercicio 7

Paso 1: Crear la función displayAlertmessage() para que muestre el mensaje que nos indican.

```
fun main() {
    val operatingSystem = "Chrome OS"
    val emailId = "sample@gmail.com"

    println(displayAlertMessage(operatingSystem, emailId))
}
fun displayAlertMessage(n1 : String, n2 : String) : String {
    return "There's a new sign-in request on $n1 for your Google Account $n2"
}

// Define your displayAlertMessage() below this line.
```

```
There's a new sign-in request on Chrome OS for your Google Account sample@gmail
```

Nizan Díaz

Paso 2:

Hacemos una modificación del código introduciendo en la función displayAlertmessage() un String inicial de displayAlertMessage(operatingSystem: String, emailId: String)

```
fun main() {
    val firstUserEmailId = "user_one@gmail.com"

    // The following line of code assumes that you named your parameter as ema
    // If you named it differently, feel free to update the name.
    println(displayAlertMessage(emailId = firstUserEmailId))
    println()

    val secondUserOperatingSystem = "Windows"
    val secondUserEmailId = "user_two@gmail.com"

    println(displayAlertMessage(secondUserOperatingSystem, secondUserEmailId))
    println()

    val thirdUserOperatingSystem = "Mac OS"
    val thirdUserEmailId = "user_three@gmail.com"

    println(displayAlertMessage(thirdUserOperatingSystem, thirdUserEmailId))
    println()
}
fun displayAlertMessage(operatingSystem : String = "Unknow OS", emailId : Stri
    return "There's a new sign-in request on $operatingSystem for your Google
}

// Define your displayAlertMessage() below this line.
```

```
There's a new sign-in request on Unknow OS for your Google Account user_one@gma

There's a new sign-in request on Windows for your Google Account user_two@gmail

There's a new sign-in request on Mac OS for your Google Account user_three@gmai
```

Nizan Díaz

Ejercicio 8

He modificado las nomenclaturas, sustituyendo las mayúsculas y minúsculas en función a lo aprendido en el codelab.

```
fun main() {
    val steps = 4000
    val caloriesBurned = pedometerStepsToCalories(steps);
    println("Walking $steps steps burns $caloriesBurned calories")
}

fun pedometerStepsToCalories(numberOfSteps: Int): Double {
    val caloriesBurnedForEachStep = 0.04
    val totalCaloriesBurned = numberOfSteps * caloriesBurnedForEachStep
    return totalCaloriesBurned
}
```

Ejercicio 9

```
fun main() {
println(comparador(300, 250))
        }

fun comparador(timeSpentToday : Int, timeSpentYesterday: Int) : Boolean {
    return timeSpentToday > timeSpentYesterday
    }
```

```
true
```

Target platform: JVM Running on kotlin v. 1.9.10

Nizan Díaz

EJERCICIO DISEÑO APP TARJETA DE CUMPLEAÑOS

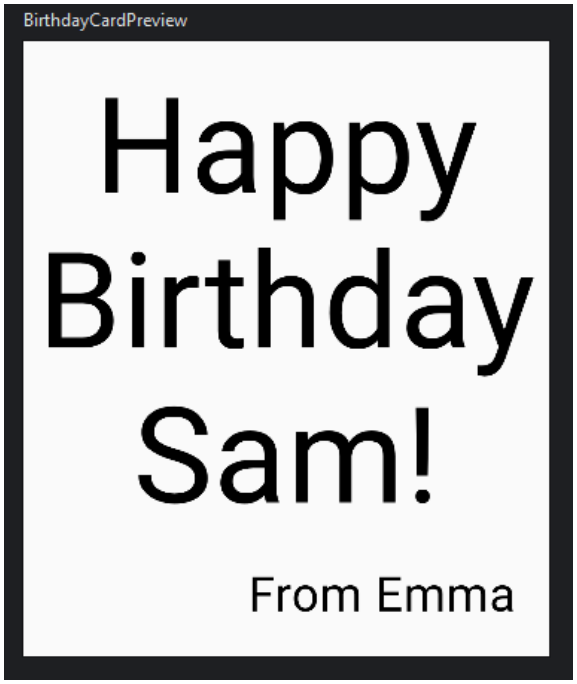
```kotlin
package com.example.happybirthday

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.padding
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.example.happybirthday.ui.theme.HappyBirthdayTheme

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
    }
}

@Composable
fun GreetingText(message : String, from : String, modifier: Modifier = Modifier){
    Column(verticalArrangement = Arrangement.Center,modifier = modifier.padding(8.dp) ) {
        Text(
            text = message,
            fontSize = 100.sp,
            lineHeight = 116.sp,
            textAlign = TextAlign.Center
        )
        Text(
            text = from,
            fontSize = 36.sp,
            modifier = Modifier.padding(16.dp).align(alignment = Alignment.End)
        )
    }
}
@Preview(showBackground = true)
@Composable
fun BirthdayCardPreview() {
    HappyBirthdayTheme {
        GreetingText(message = "Happy Birthday Sam!",from = "From Emma")
    }
}
```

DESIGN



Nizan Díaz

EJERCICIO DISEÑO APP TARJETA DE CUMPLEAÑOS CON IMAGEN

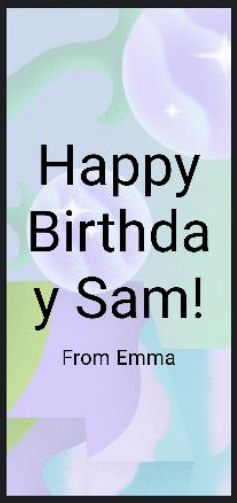
Funcion GreetingImage

```
        .align(alignment = Alignment.CenterHorizontally)
    )
  }
}
@Composable
fun GreetingImage(message : String, from: String, modifier: Modifier = Modifier) {
    val image = painterResource(R.drawable.androidparty)
    Box { this: BoxScope
        Image(
            painter = image,
            contentDescription = null,
            contentScale = ContentScale.Crop,
            alpha = 0.5F
        )
        GreetingText(message = message, from = from,modifier = Modifier
            .fillMaxSize()
            .padding(8.dp))
    }
}
@Preview(showBackground = true)
@Composable
fun BirthdayCardPreview() {
    HappyBirthdayTheme {
        GreetingImage(stringResource(R.string.happy_birthday_text), stringResource(R.strin
    }
}
```

Up-to-date ✔ ⚠ 2

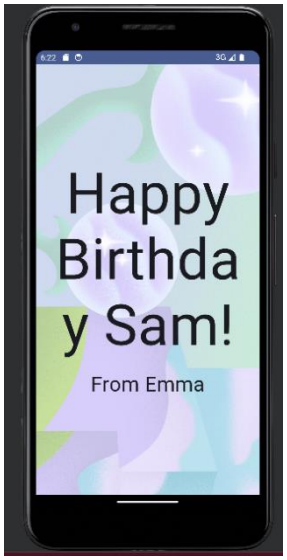
Up-to-date ✔

BirthdayCardPreview



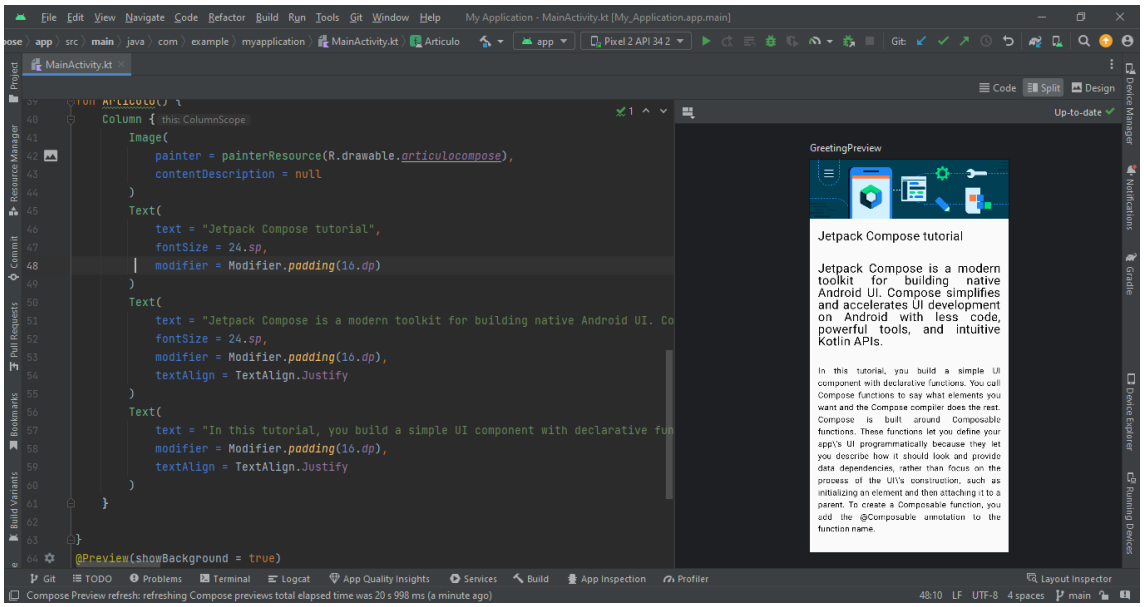
Función Main implementando GreetingImage

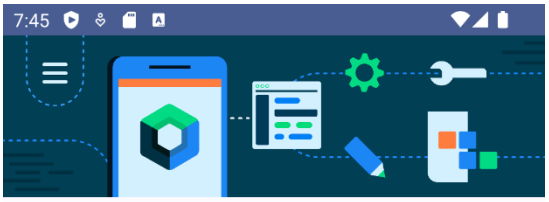
```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            HappyBirthdayTheme {
                // A surface container using the 'background' color from the theme
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
                ) {
                    GreetingImage(
                        stringResource(R.string.happy_birthday_text),
                        stringResource(R.string.signature_text)
                    )
                }
            }
        }
    }
}
```



Nizan Díaz

ARTICULO DE COMPOSE





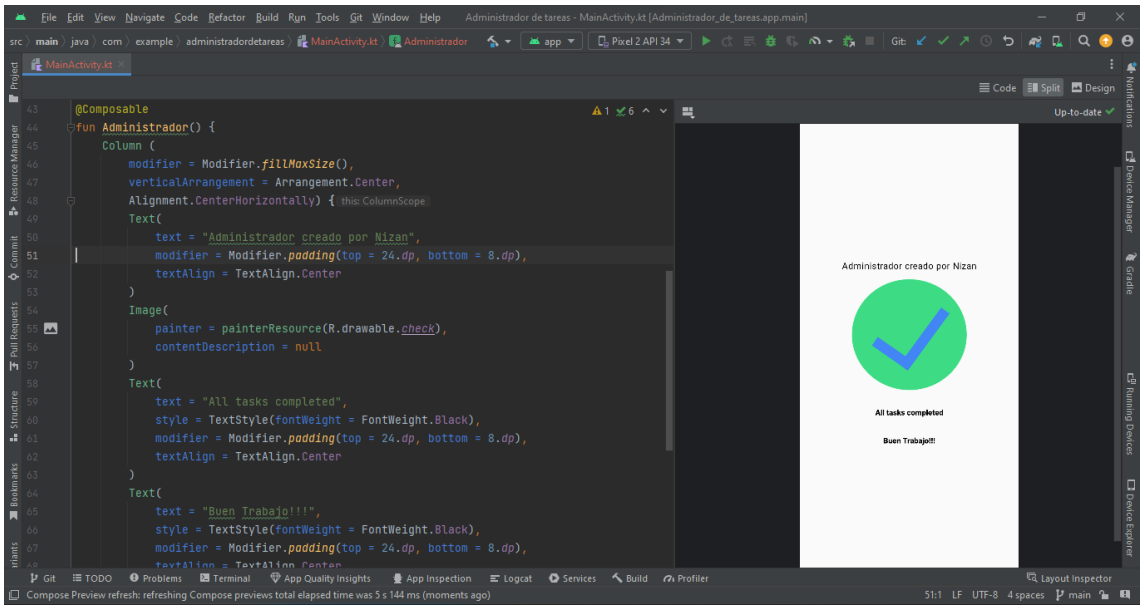
Jetpack Compose realizado por Nizan

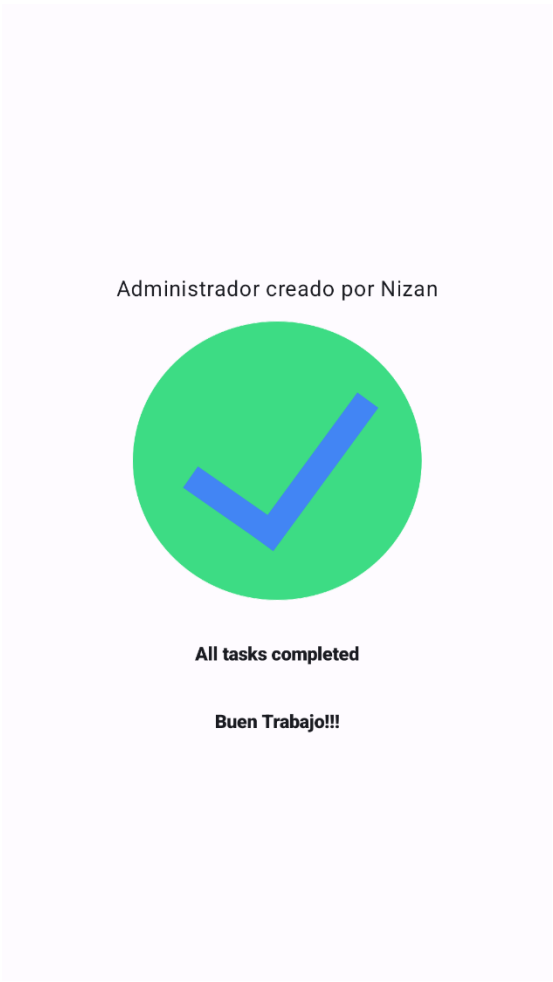
Jetpack Compose is a modern toolkit for building native Android UI. Compose simplifies and accelerates UI development on Android with less code, powerful tools, and intuitive Kotlin APIs.

In this tutorial, you build a simple UI component with declarative functions. You call Compose functions to say what elements you want and the Compose compiler does the rest. Compose is built around Composable functions. These functions let you define your app\'s UI programmatically because they let you describe how it should look and provide data dependencies, rather than focus on the process of the UI\'s construction, such as initializing an element and then attaching it to a parent. To

Nizan Díaz

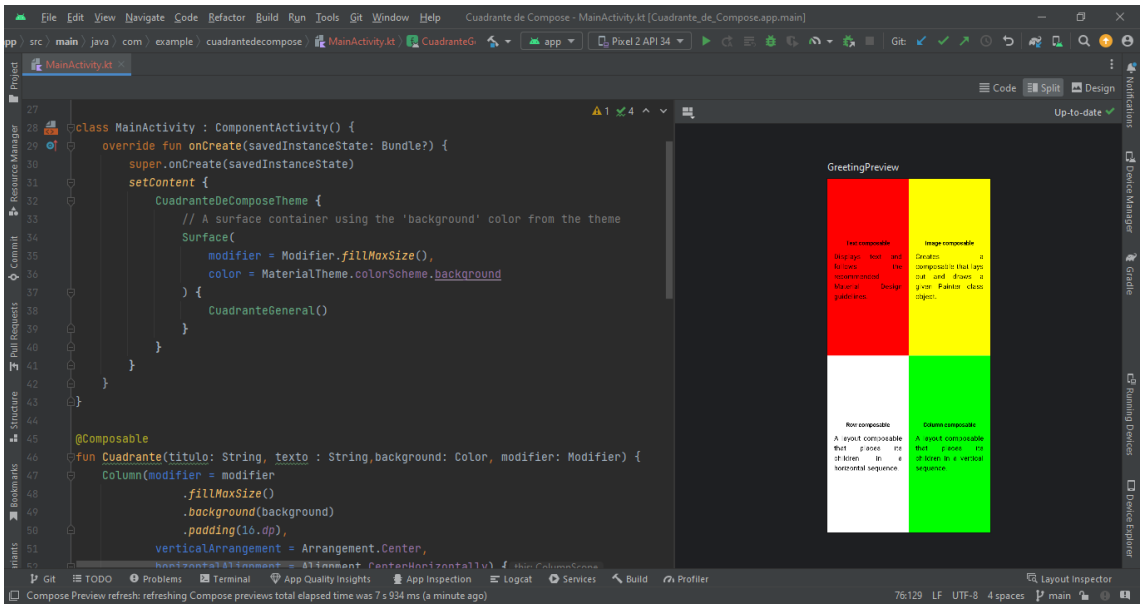
ADMINISTRADOR DE TAREAS

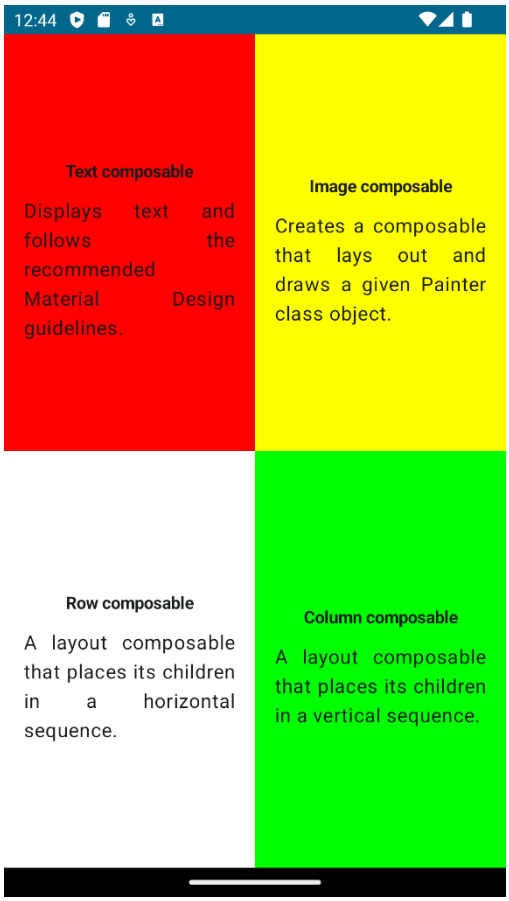




Nizan Díaz

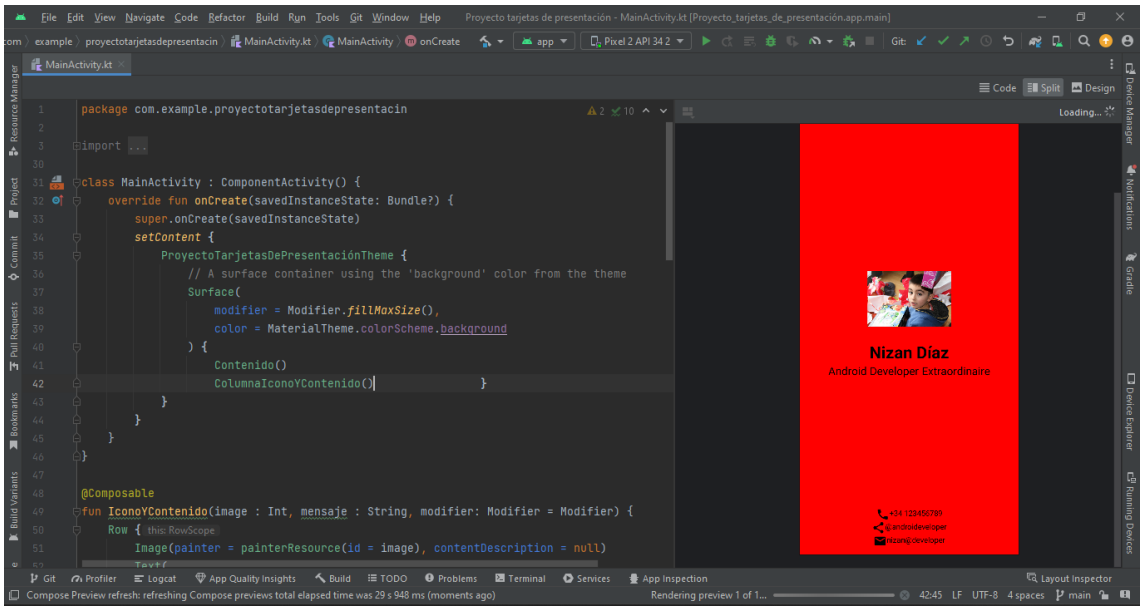
CUADRANTE DE COMPOSE





Nizan Díaz

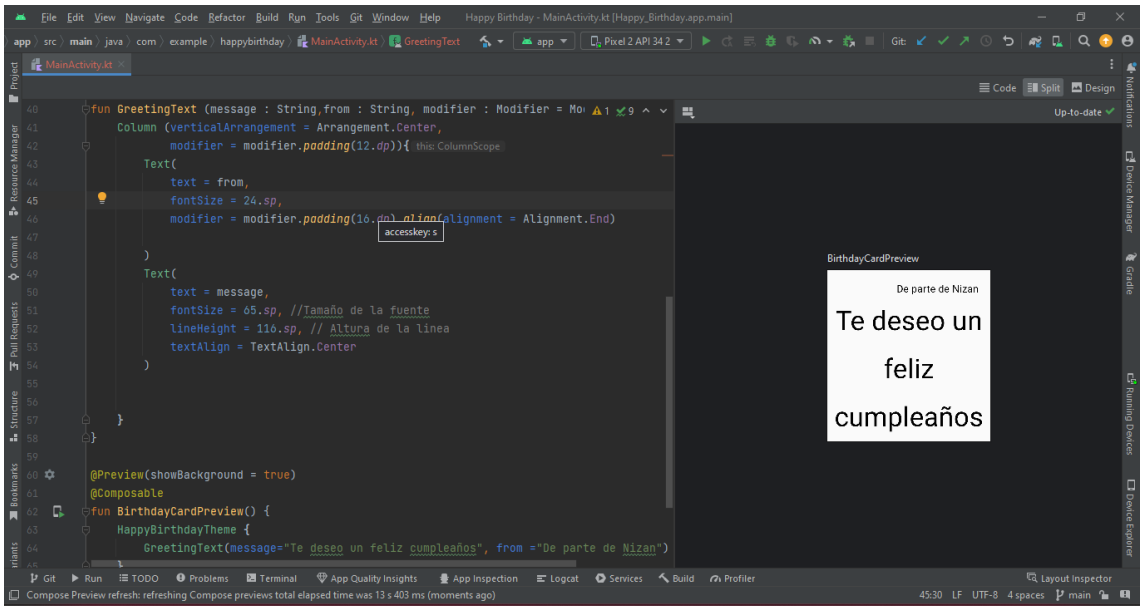
PROYECTO FINALTARJETAS DE PRESENTACION



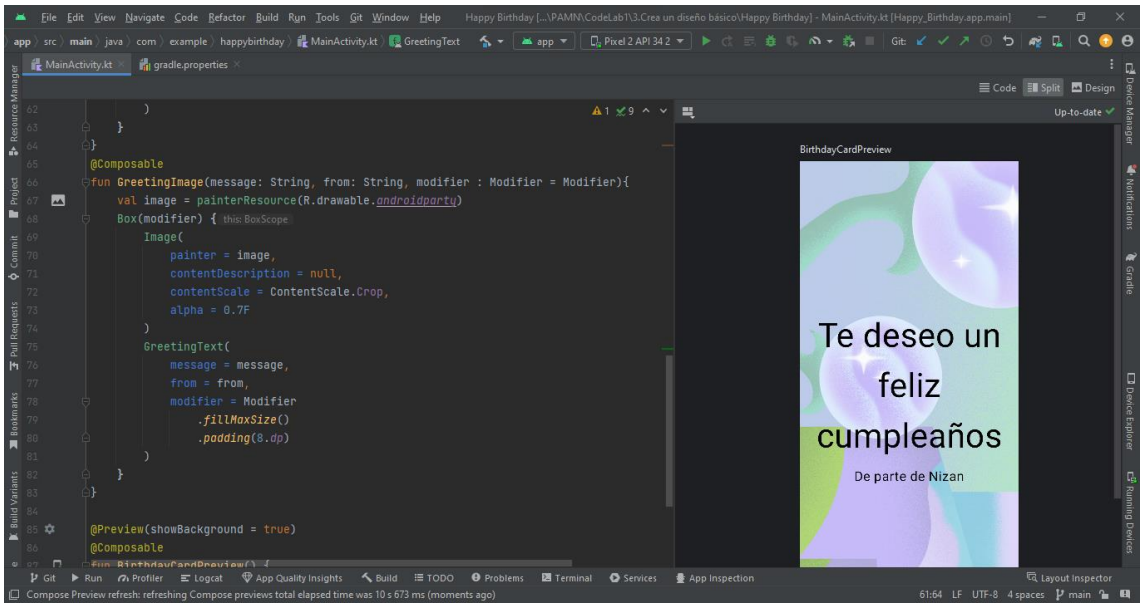
Nizan Díaz

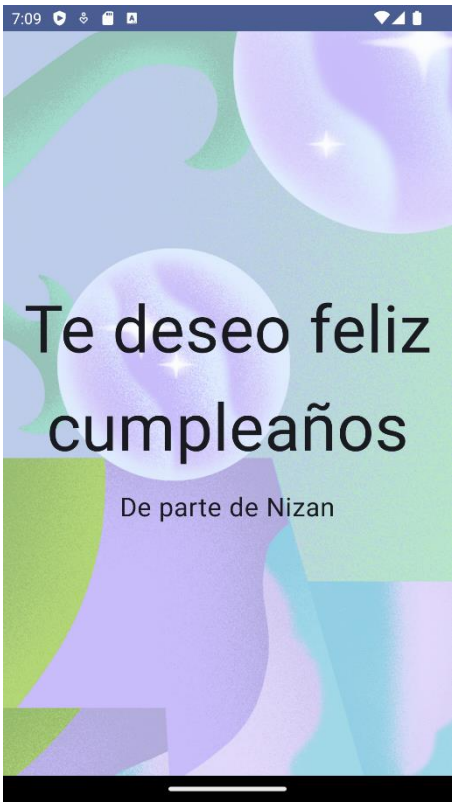
MODIFICAIONES EXTRAS AL CÓDIGO ESTANDAR

Tarjeta de cumpleaños



Nizan Díaz


)
}
}
@Composable
fun GreetingImage(message: String, from: String, modifier : Modifier = Modifier){
val image = painterResource(R.drawable.androidparty)
Box(modifier) { this: BoxScope
Image(
painter = image,
contentDescription = null,
contentScale = ContentScale.Crop,
alpha = 0.7F
)
GreetingText(
message = message,
from = from,
modifier = Modifier
.fillMaxSize()
.padding(8.dp)
)
}
}
@Preview(showBackground = true)
@Composable
BirthdayCardPreview
Te deseo un feliz cumpleaños
De parte de Nizan


7:09
Te deseo feliz cumpleaños
De parte de Nizan