Programación de Aplicaciones Móviles Nativas

Codelab: Compila la IU de una app

Laura González Suárez

Introducción

Este informe tiene como objetivo proporcionar una visión general de la Unidad 2 del codelab, que se enfoca en la compilación de la interfaz de usuario (IU) de una aplicación. Esta unidad se centra en la creación de interfaces de usuario efectivas para aplicaciones Android. Comenzamos con conceptos básicos de Kotlin, avanzamos a la incorporación de botones interactivos en la IU y finalmente exploramos la interacción con la interfaz y el manejo del estado de la aplicación.

Este informe presentará ejemplos concretos de cómo hemos aplicado los conocimientos adquiridos durante el codelab.

Ruta 1: Conceptos básicos de Kotlin

En esta primera parte, exploraremos los fundamentos esenciales del lenguaje Kotlin, desde sus conceptos más básicos hasta la complejidad de las expresiones lambda.
A continuación, se muestran algunas de las funciones desarrolladas en el playground durante esta sección del codelab:

Ilustración 1: Sentencias if-else

```kotlin
fun main() {
    val trafficLightColor = "Amber"

    when (trafficLightColor) {
        "Red" -> println("Stop")
        "Yellow", "Amber" -> println("Slow")
        "Green" -> println("Go")
        else -> println("Invalid traffic-light color")
    }
}
```

```
Slow
```

Target platform: JVM Running on kotlin v. 1.9.10

Ilustración 2: sentencia when

```kotlin
fun main() {
    val x: Any = 15

    when (x) {
        2, 3, 5, 7 -> println("x is a prime number between 1 and 10.")
        in 1..10 -> println("x is a number between 1 and 10, but not a prime numb
        is Int -> println("x is an integer number, but not between 1 and 10.")
        else -> println("x isn't an integer number.")
    }
}
```

```
x is an integer number, but not between 1 and 10.
```

Target platform: JVM Running on kotlin v. 1.9.10

Ilustración 3: sentencia when avanzada

```kotlin
fun main() {
    val trafficLightColor = "Amber"

    val message = when(trafficLightColor) {
        "Red" -> "Stop"
        "Yellow", "Amber" -> "Proceed with caution."
        "Green" -> "Go"
        else -> "Invalid traffic-light color"
    }
    println(message)
}
```

```
Proceed with caution.
```

Target platform: JVM Running on kotlin v. 1.9.10

Ilustración 4: sentencia when como expresión

```kotlin
class SmartDevice {

    val name = "Android TV"
    val category = "Entertainment"
    var deviceStatus = "online"

    fun turnOn(){
        println("Smart device is turned on.")
    }

    fun turnOff(){
        println("Smart device is turned off.")
    }
}

fun main(){
    val smartTvDevice = SmartDevice()
    println("Device name is: ${smartTvDevice.name}")
    smartTvDevice.turnOn()
    smartTvDevice.turnOff()
}
```

```
Device name is: Android TV
Smart device is turned on.
Smart device is turned off.
```

Target platform: JVM Running on kotlin v. 1.9.10

Ilustración 5: Utilizar variables de una función/clase en otra

```kotlin
fun main() {
    val child = 5
    val adult = 28
    val senior = 87

    val isMonday = true

    println("Movie ticket price for a $child years old = \$${ticketPrice(child, isMonday)}")
    println("Movie ticket price for a $adult years old = \$${ticketPrice(adult, isMonday)}")
    println("Movie ticket price for a $senior years old = \$${ticketPrice(senior, isMonda
}

fun ticketPrice(age: Int, isMonday: Boolean): Int {
    return when(age) {
        in 0..12 -> 15
        in 13..60 -> if (isMonday) 25 else 30
        in 61..100 -> 20
        else -> -1
    }
}
```

```
Movie ticket price for a 5 years old = $15.
Movie ticket price for a 28 years old = $25.
Movie ticket price for a 87 years old = $20.
```

Target platform: JVM Running on kotlin v. 1.9.10

Ilustración 6: Precio de las entradas de cine según edad

Ruta 2: Agrega un botón a una App

En esta etapa, el objetivo ha sido crear una aplicación interactiva de Dice Roller para Android utilizando Compose y agregar un botón a la interfaz de usuario. Para lograrlo, hemos definido funciones de componibilidad e importado recursos drawable para mostrar una imagen usando el elemento "Image". Además, se ha utilizado el elemento remember de componibilidad para almacenar objetos en una composición en la memoria y actualizado la IU con la función mutableStateOf() para convertirla en un elemento observable.

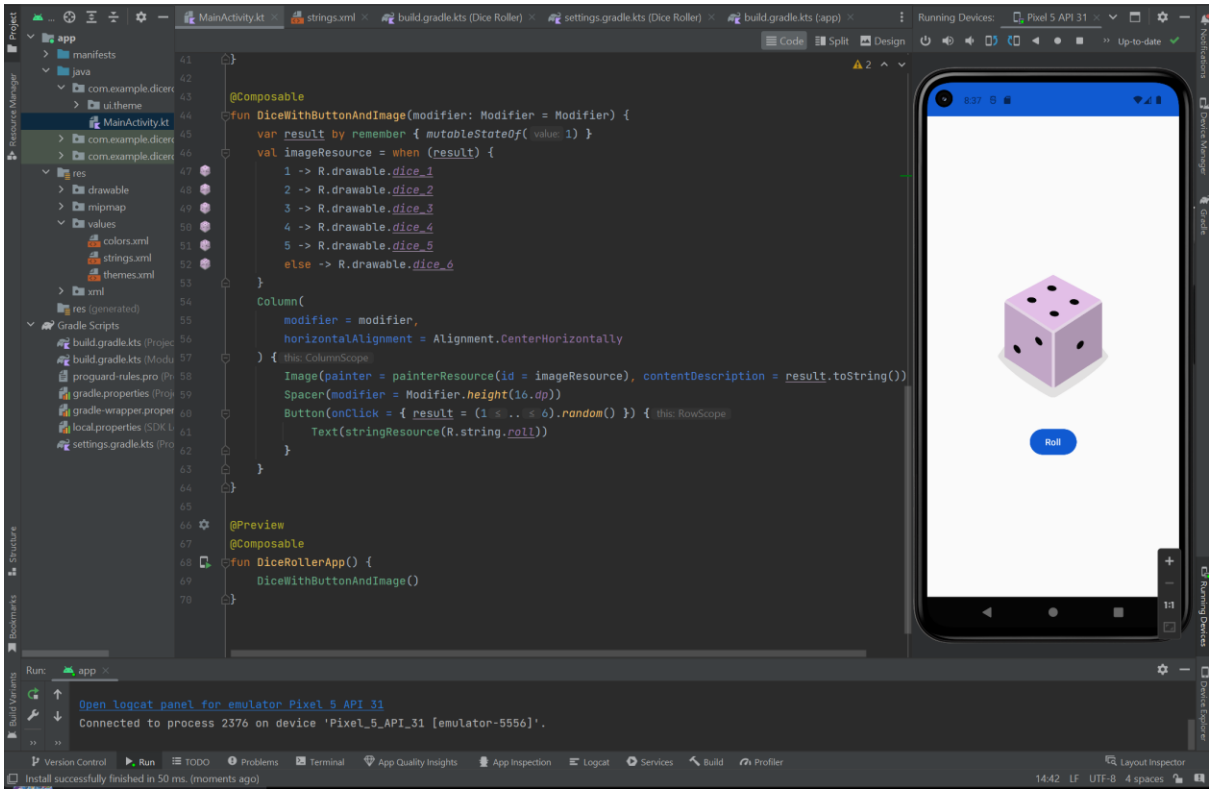


Ilustración 7: Aplicación Dice roller

Además, en esta sección del Codelab, se ha abordado el tema del depurado en Android Studio. La siguiente imagen ilustra el depurador con la ejecución de la aplicación pausada en un punto de interrupción (breakpoint):

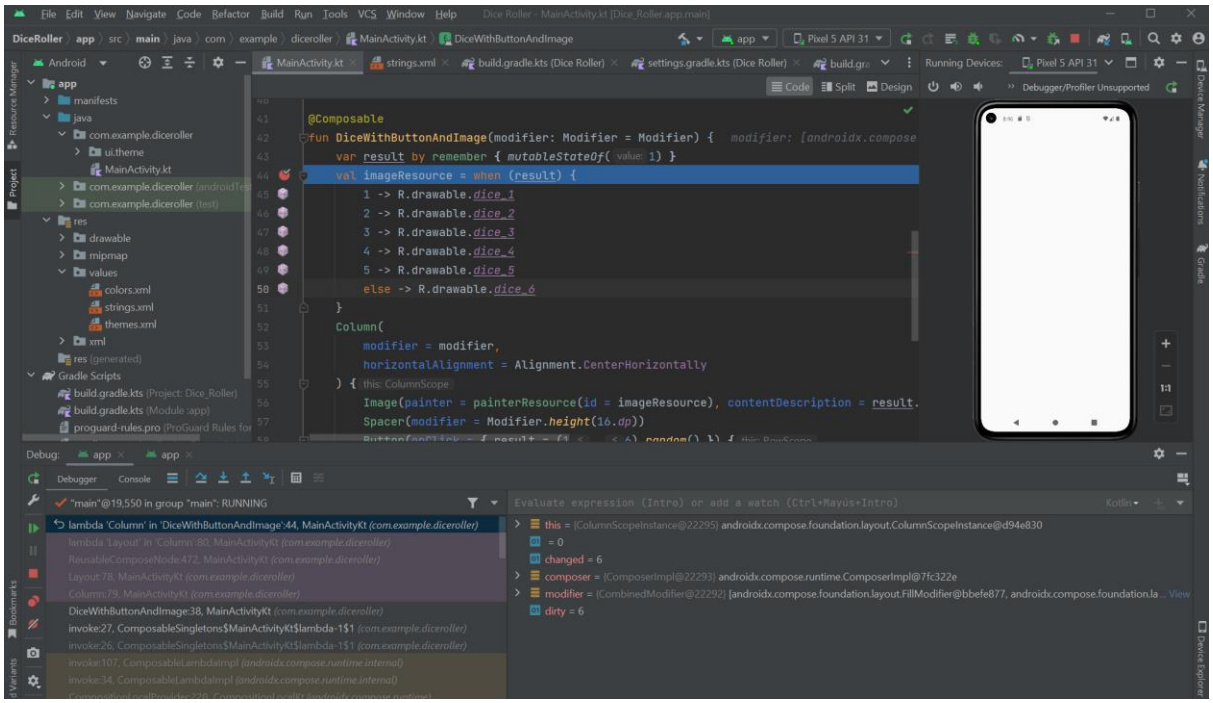


Ilustración 8: Uso del Depurador

En cuanto al siguiente proyecto, se ha desarrollado una aplicación interactiva "LemonadeApp" que posibilita a los usuarios la experiencia de exprimir limones y preparar una limonada. La aplicación sigue un proceso sencillo:

1. Al iniciar la aplicación, los usuarios son recibidos por un limonero y una indicación que les invita a presionar la imagen del limonero para "seleccionar" un limón.
2. Tras pulsar el limonero, los usuarios ven un limón en pantalla y se les solicita que lo presionen para "exprimirlo" y así preparar la limonada.
3. Una vez que han exprimido el limón, se les presenta un refrescante vaso de limonada y se les invita a presionarlo para "beber" la limonada.
4. Después de beber la limonada, aparece un vaso vacío, y los usuarios deben presionarlo para comenzar de nuevo y repetir el proceso.

La aplicación incorpora imágenes y etiquetas de texto que cambian en cada clic.

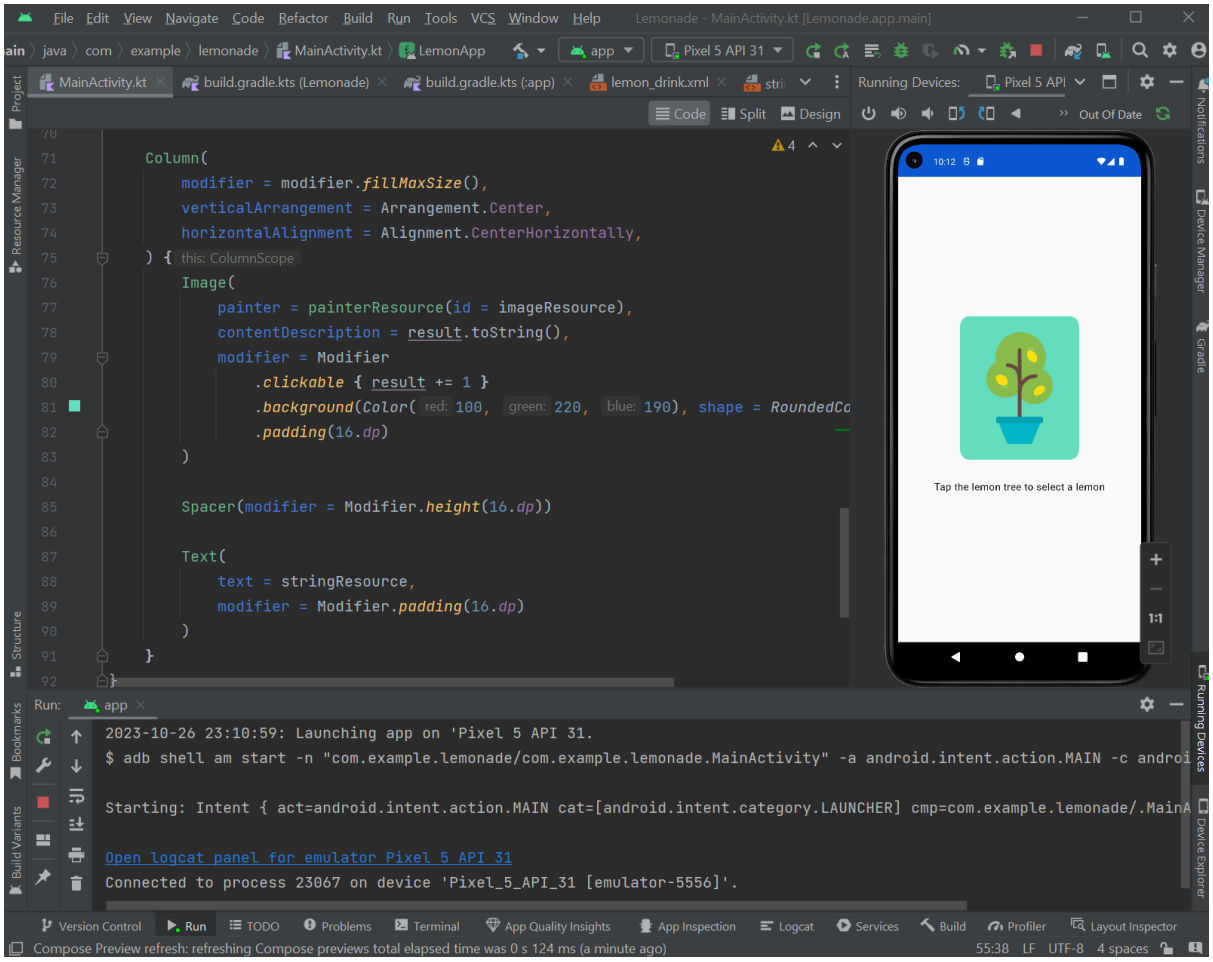


Ilustración 9: Lemonade App (pantalla 1) en AndroidStudio

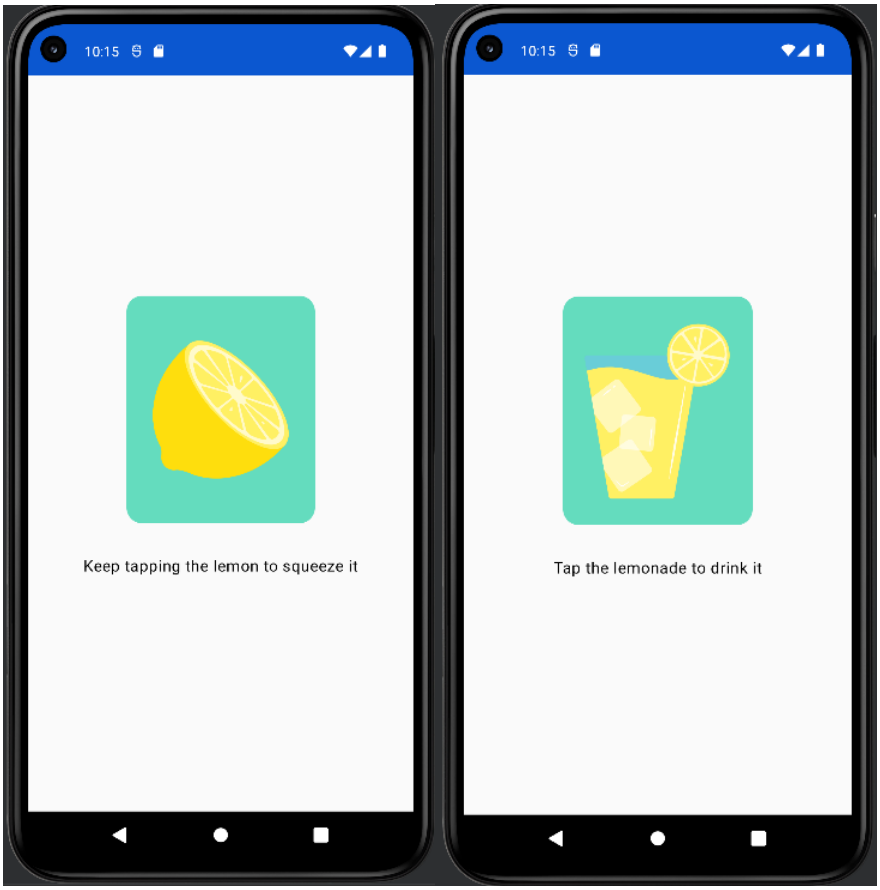


Ilustración 10: Lemonade App (pantallas 2 y 3)

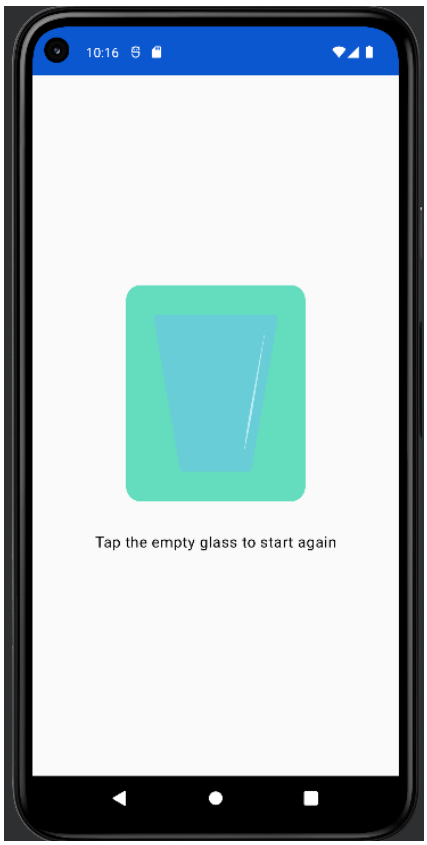


Ilustración 11: Lemonade App (pantalla 4)

Ruta 3: Interactúa con la IU y el estado

En el contexto de esta ruta de aprendizaje, se ha llevado a cabo la compilación de una aplicación diseñada para calcular propinas de manera sencilla. La funcionalidad principal de esta aplicación se basa en permitir al usuario ingresar una cantidad correspondiente a la factura, tras lo cual la aplicación proporcionará una sugerencia de propina. Inicialmente, el porcentaje de propina se encuentra en un 15% del importe total.

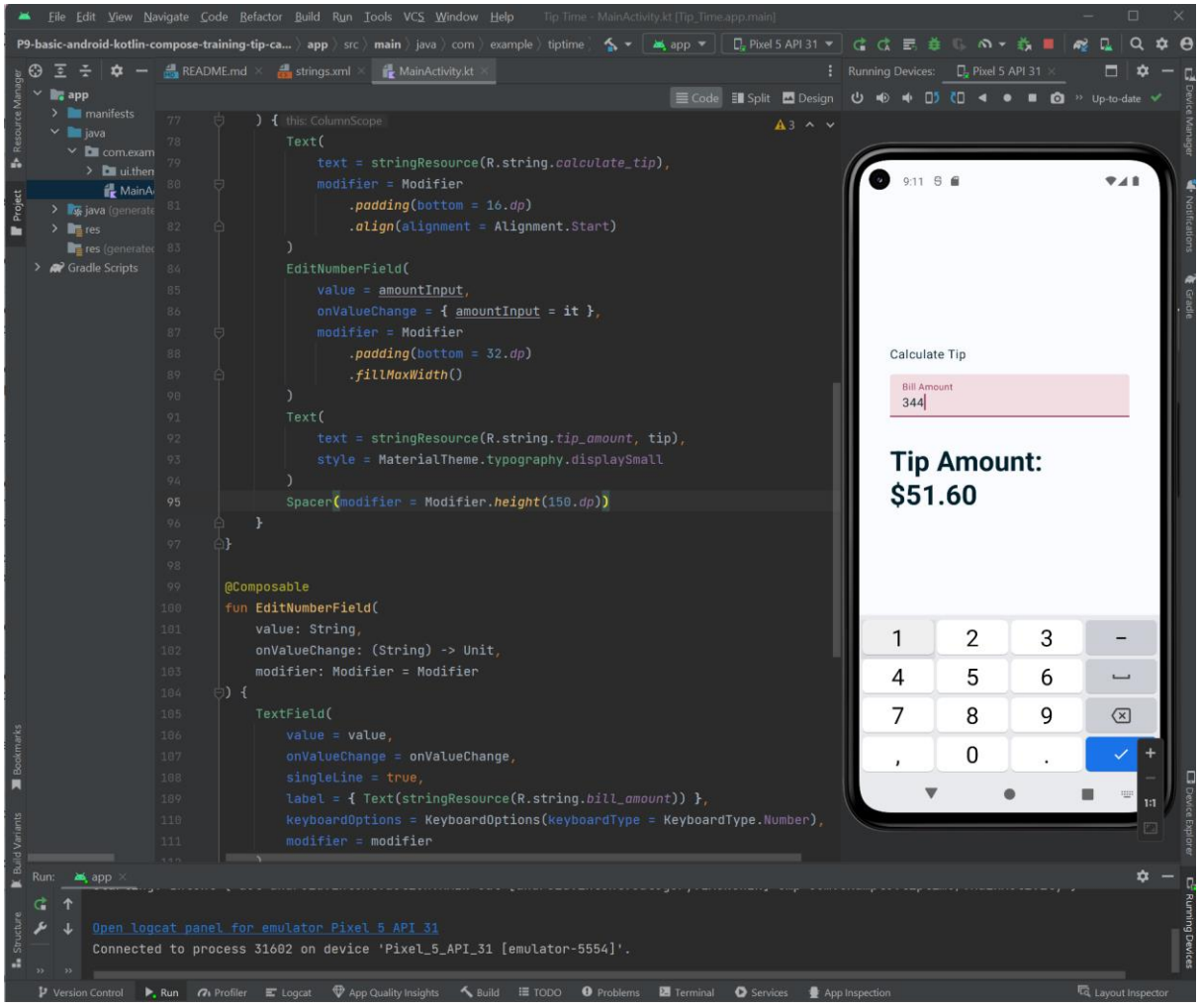


Ilustración 12: Tip Time (estado inicial)

Posteriormente se ha añadido la funcionalidad de calcular el valor final de la propina preguntándole al usuario que porcentaje de propina desea utilizar en un nuevo campo de texto.

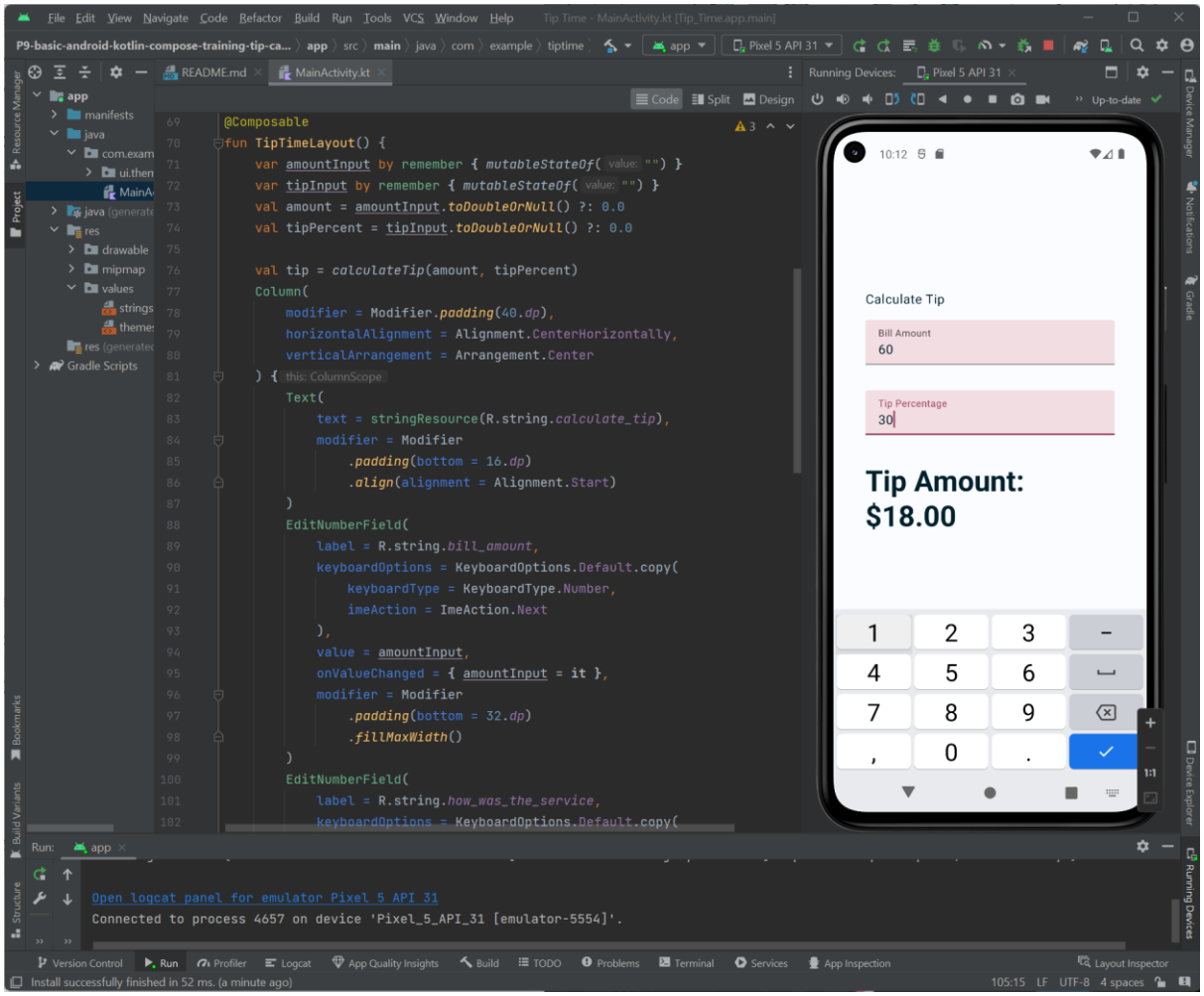


Ilustración 13: Tip Time (versión 1)

En la siguiente versión de nuestra aplicación, como nueva funcionalidad se le ha añadido un botón tipo "toggle" con el que seleccionaremos si queremos redondear el valor final de la propina o no. Además, ahora podemos utilizar la app con la pantalla en horizontal.

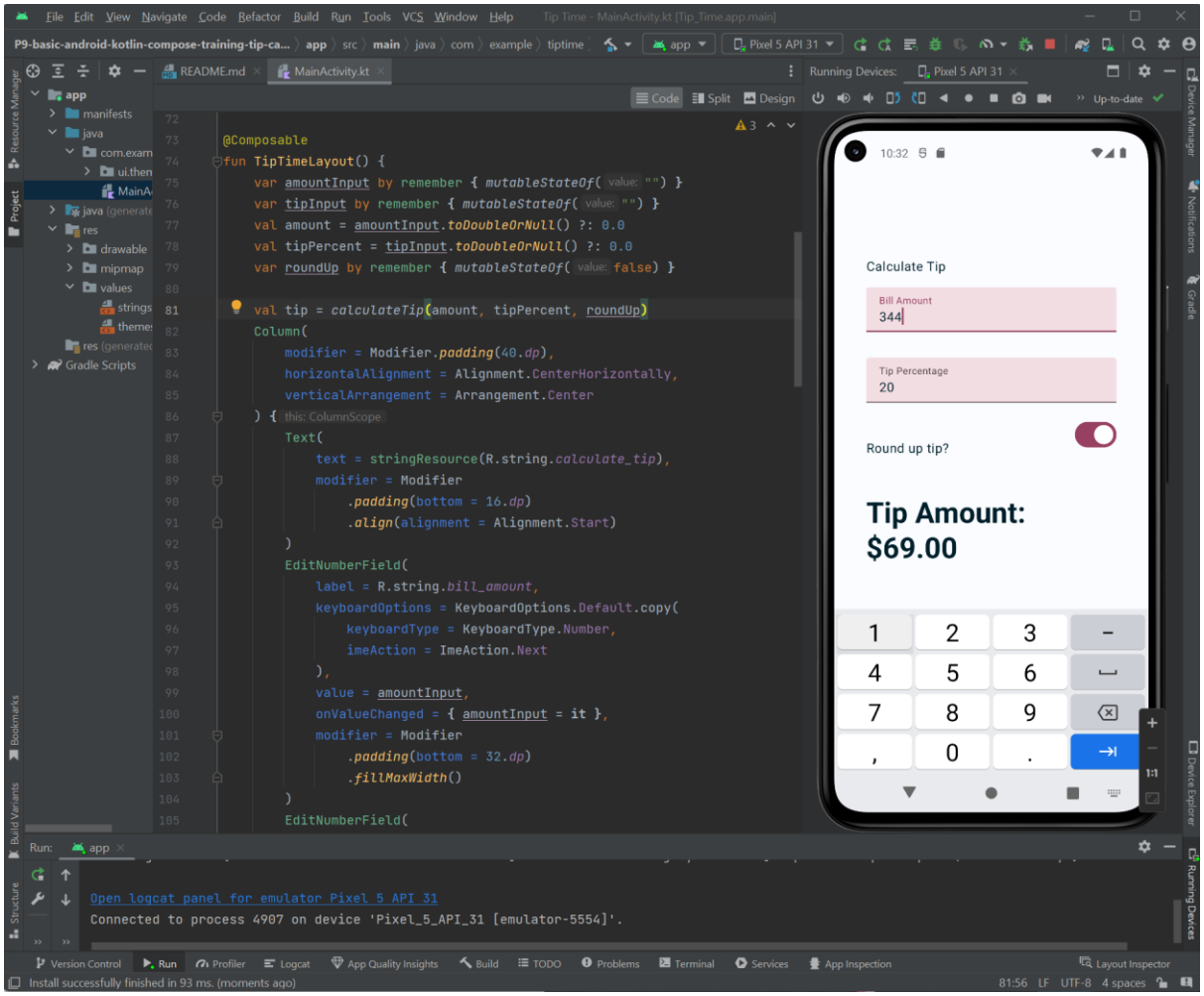


Ilustración 14: Tip Time (versión 2)

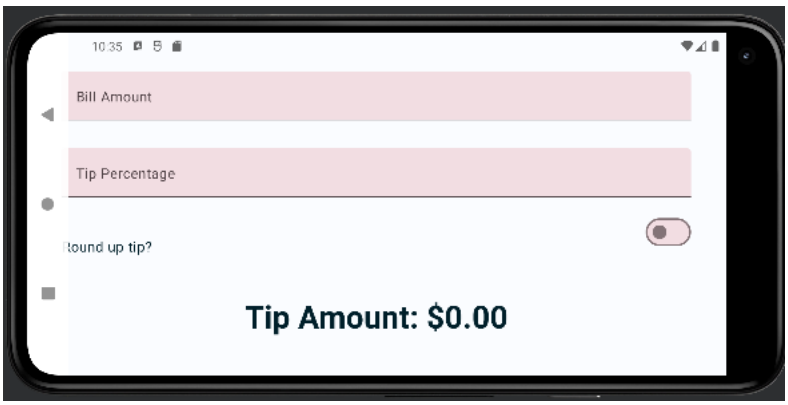


Ilustración 14: Tip Time (versión 2 - horizontal)

Para la versión final de la aplicación se ha agregado un botón de acción a un teclado virtual que permite pasar directamente al siguiente campo de texto. Además, se han agregado íconos iniciales a los campos de texto.

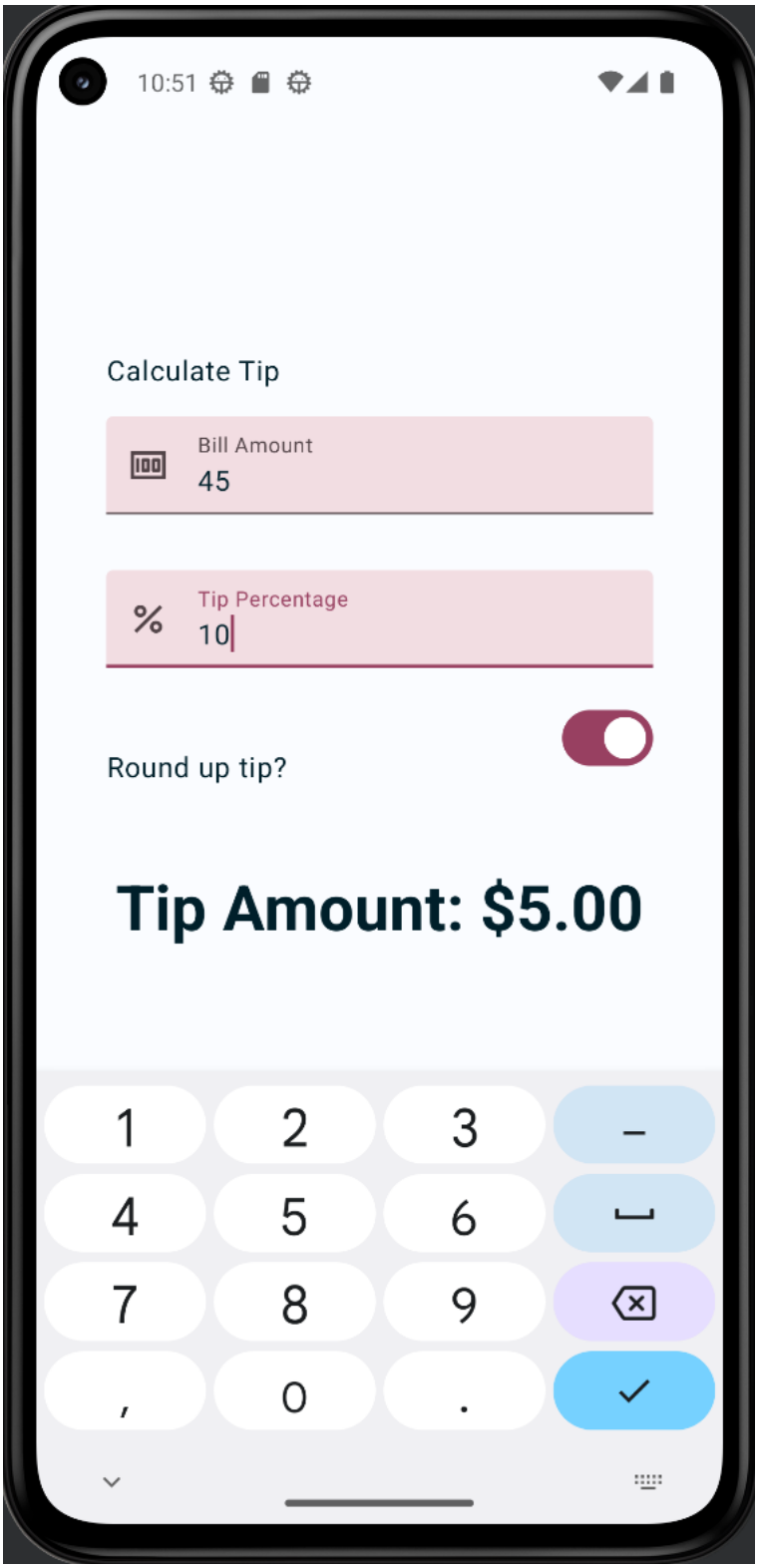


Ilustración 15: Tip Time (versión final)

En esta misma ruta de aprendizaje, se han abordado las pruebas locales y de instrumentación, aplicándolas de manera específica a nuestra aplicación "Tip Time".

Las pruebas locales implican la evaluación directa de métodos a partir del código de la aplicación. Por lo tanto, es fundamental que los métodos que se desean probar estén accesibles para las clases y métodos de prueba. Esto se hace cambiando la visibilidad del método de private a internal.

Un ejemplo concreto de una prueba local se encuentra en el siguiente fragmento de código, donde se garantiza la correcta funcionalidad del método "calculateTip()".

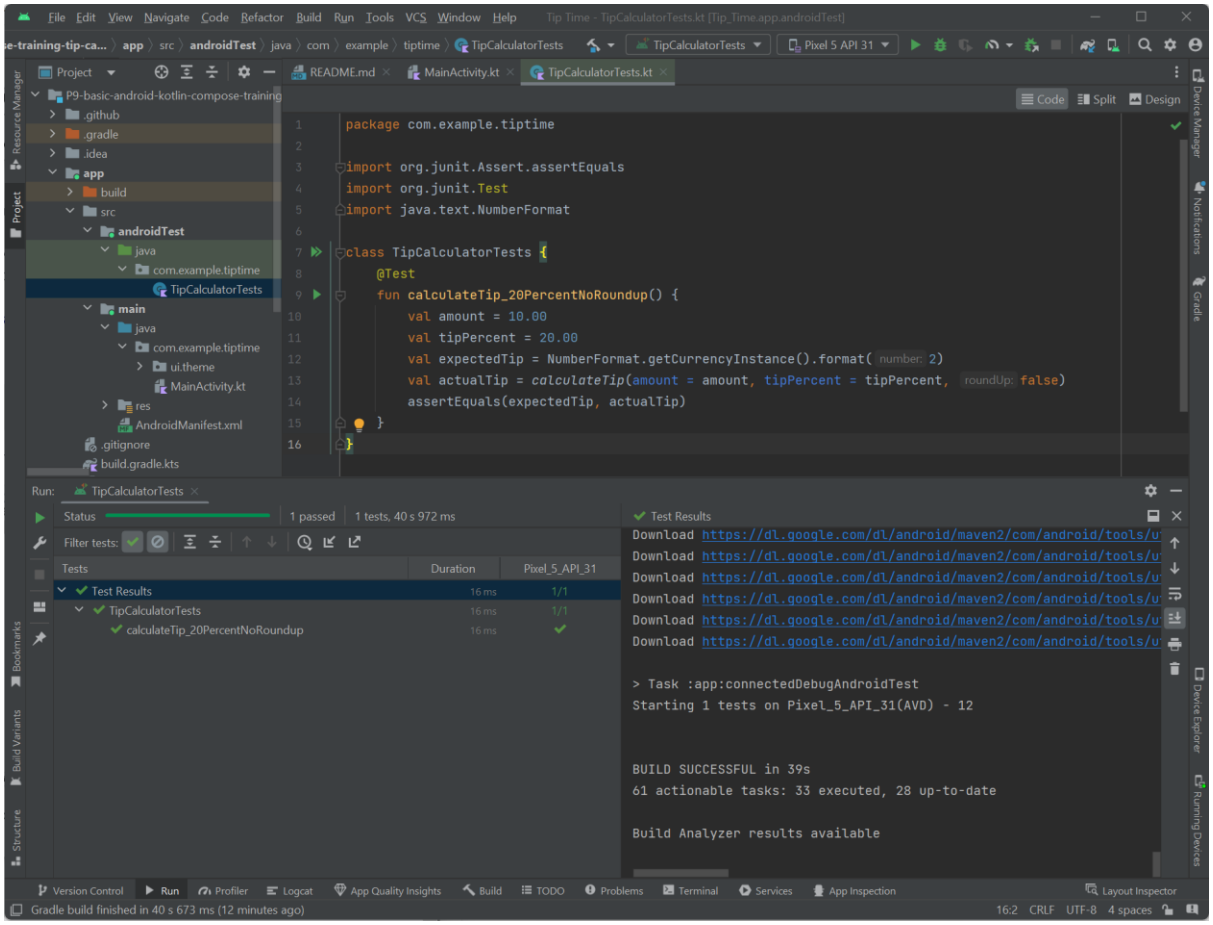


Ilustración 16: Prueba local

Las pruebas de instrumentación, por otro lado, se enfocan en evaluar una instancia real de la aplicación junto con su interfaz de usuario (IU). En este tipo de pruebas, es esencial configurar el contenido de la IU de manera análoga a como se realiza en el método "onCreate()" del archivo "MainActivity.kt" al escribir el código de la aplicación "Tip Time".

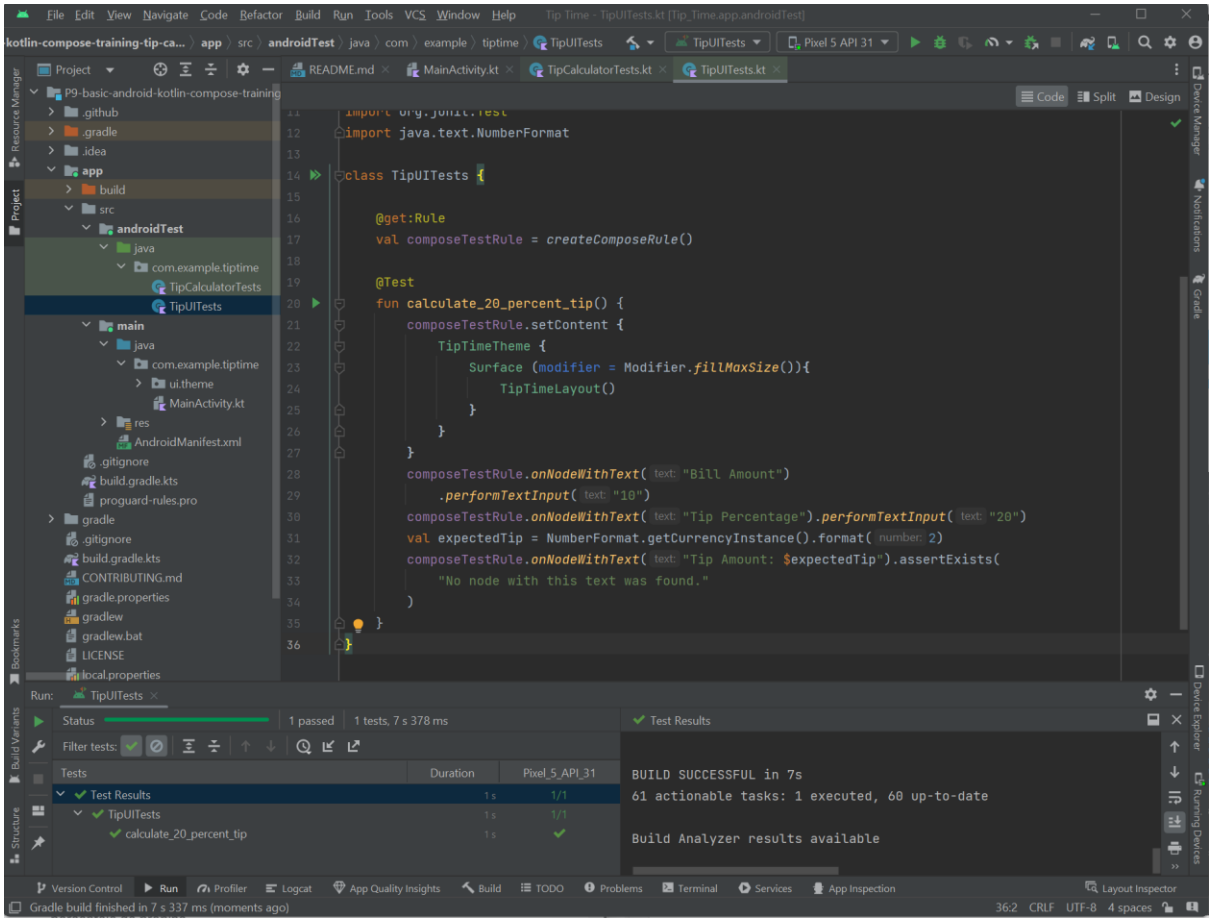


Ilustración 17: Prueba de instrumentación

Proyecto ArtSpace

La aplicación ArtSpace hace de galería de arte, permitiendo a los usuarios navegar entre diferentes obras. Cada obra de arte está acompañada de información detallada sobre su nombre, autor y período histórico. La navegación entre las obras se facilita mediante botones "Previous" y "Next".

La lógica de programación permite una transición entre las piezas. Concretamente, las funciones onPrevClick() y onNextClick() con las que permiten a los usuarios explorar las distintas obras, manteniendo siempre un índice dentro de los límites de la lista.

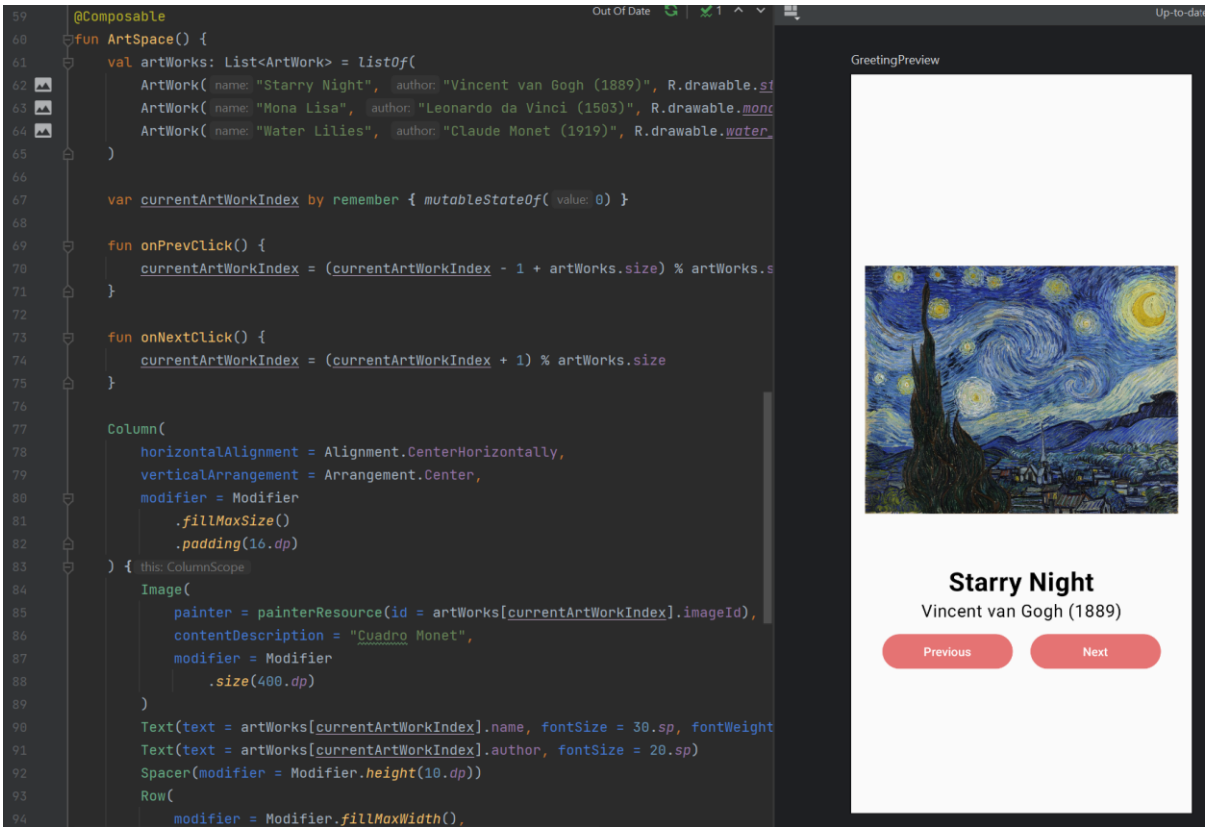
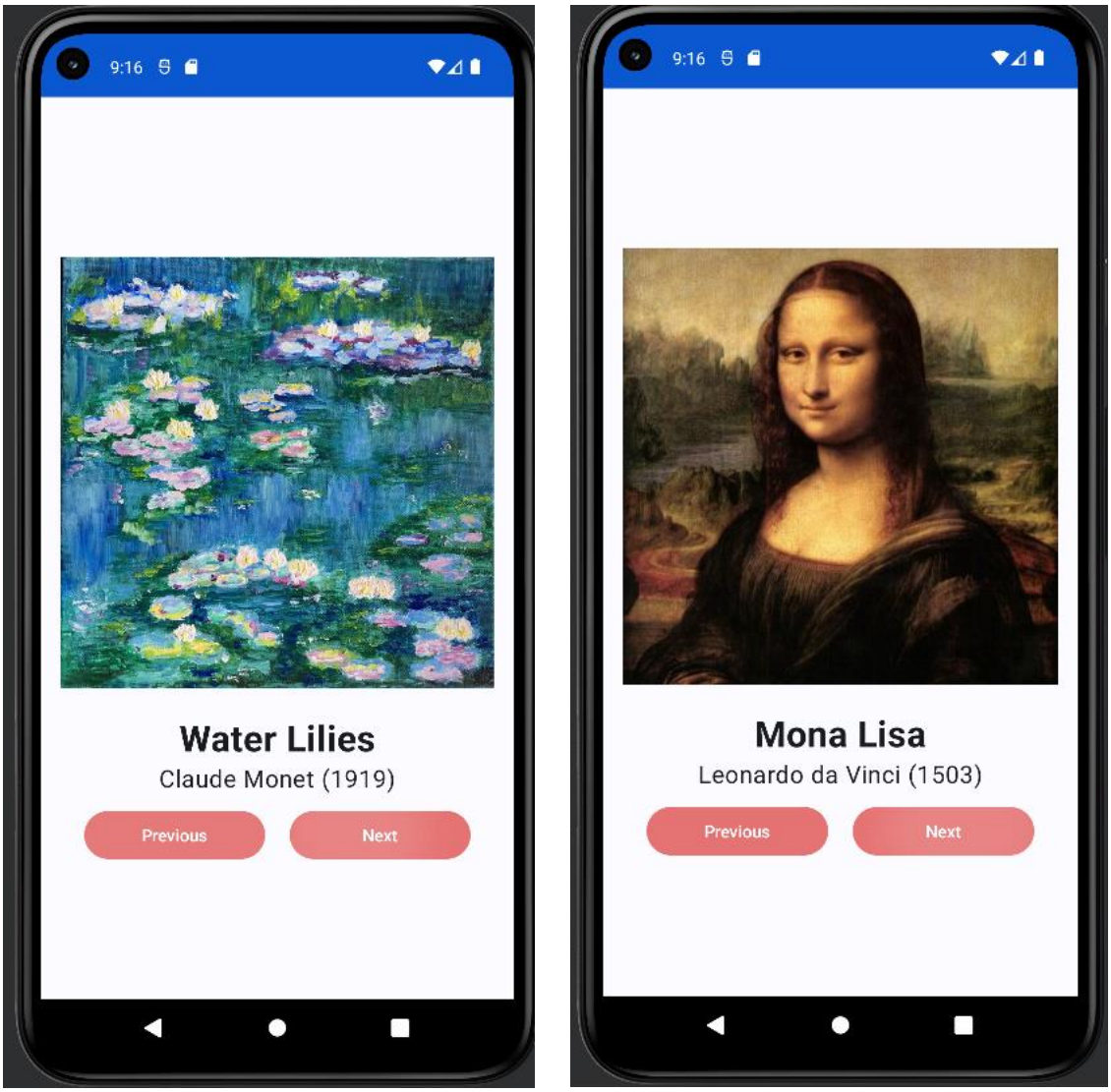


Ilustración 18: Proyecto ArtSpace



Ilustraciones 18: Proyecto ArtSpace en el emulador

Bibliografía

Unidad 2: Compila la IU de una app. Google for Developers.

https://developer.android.com/courses/android-basics-compose/unit-2?hl=es-419 [en línea] 30/10/23