

Compila la IU de una app

40983 – Programación de Aplicaciones Móviles Nativas

Pablo Santana Susilla

Pr. Laboratorio 01.01

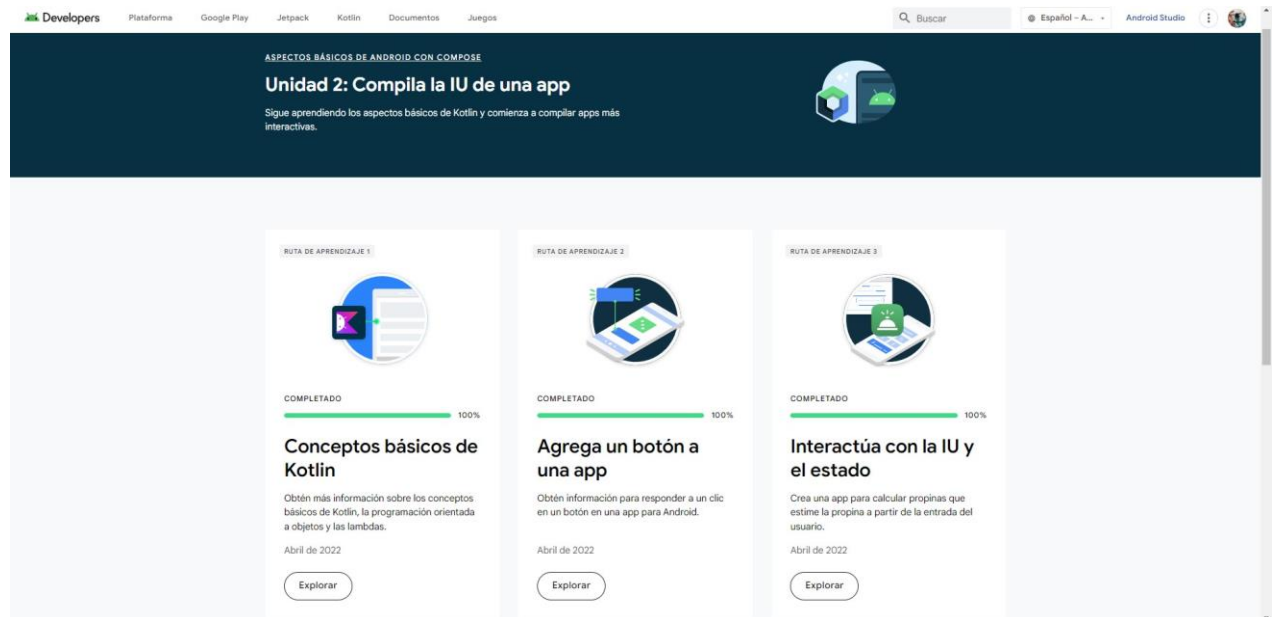
Profesor adjunto: Antonio Iván Hernández Fragiel

Curso: 2023/2024

Contenido

PRUEBA DE FINALIZACIÓN DEL CODELAB.....	3
RUTA DE APRENDIZAJE 1: Conceptos básicos de Kotlin	3
Escribe condicionales en Kotlin.....	3
Usa la nulabilidad en Kotlin.....	5
Usa clases y objetos en Kotlin	6
Precio de la entrada de cine	6
Conversor de temperatura	7
Catálogo de canciones	7
Perfil de internet	8
Subasta Especial.....	8
RUTA DE APRENDIZAJE 2: Agrega un botón a una app.....	9
RUTA DE APRENDIZAJE 3: Interactúa con la UI y el estado	13
OPINIÓN.....	17
ENLACE A GITHUB.....	17
BIBLIOGRAFÍA	18

PRUEBA DE FINALIZACIÓN DEL CODELAB



RUTA DE APRENDIZAJE 1: Conceptos básicos de Kotlin

Escribe condicionales en Kotlin

```
1.9.10 ▼ JVM ▼ Program arguments

fun main() {
    val trafficLightColor = "Black"

    if (trafficLightColor == "Red") {
        println("Stop")
    } else if (trafficLightColor == "Yellow") {
        println("Slow")
    } else if (trafficLightColor == "Green") {
        println("Go")
    } else {
        println("Invalid traffic-light color")
    }
}
```

Invalid traffic-light color

Figura 1. Sentencia *if-else*

1.9.10 ▾ JVM ▾ Program arguments

```
fun main() {  
    val trafficLightColor = "Yellow"  
  
    when (trafficLightColor) {  
        "Red" -> println("Stop")  
        "Yellow" -> println("Slow")  
        "Green" -> println("Go")  
        else -> println("Invalid traffic-light color")  
    }  
}
```

Slow

Figura 2. Sentencia *when*

1.9.10 ▾ JVM ▾ Program arguments Copy link <> Share code Run

```
fun main() {  
    val x: Any = 18  
  
    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 number.")  
        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.

Figura 3. Sentencia *when* avanzada

```
1.9.10 ▼ JVM ▼ Program arguments
```

```
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.

Figura 4. Sentencia *when* como expresión

Usa la nulabilidad en Kotlin

```
1.9.10 ▼ JVM ▼ Program arguments
```

```
fun main() {  
    var number: Int? = 10  
    println(number)  
  
    number = null  
    println(number)  
}
```

10
null

Figura 5. Usar variables anulables

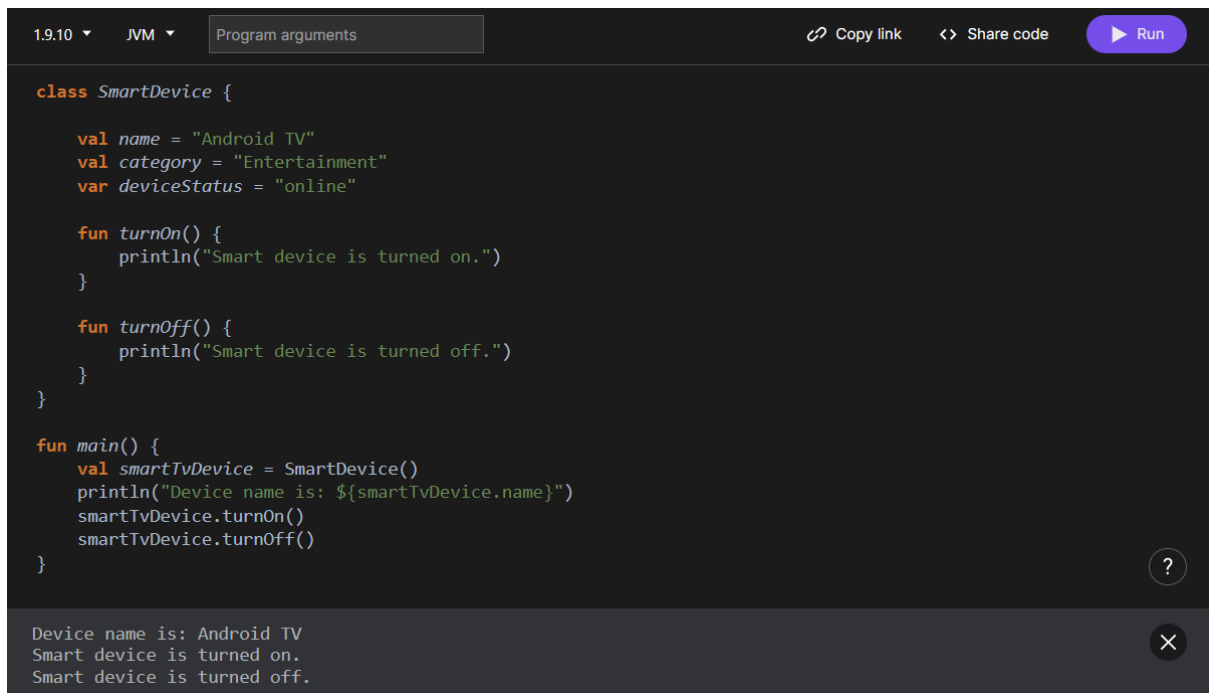
```
1.9.10 ▼ JVM ▼ Program arguments Copy link <> Share code Run
```

```
fun main() {  
    val favoriteActor: String? = "Sandra Oh"  
    val lengthOfName = favoriteActor?.length ?: 0  
    println("The number of characters in your favorite actor's name is $lengthOfName.")  
}
```

The number of characters in your favorite actor's name is 9.

Figura 6. Procesa variables anulables

Usa clases y objetos en Kotlin



```
1.9.10 JVM Program arguments Copy link Share code Run

class SmartDevice {

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

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

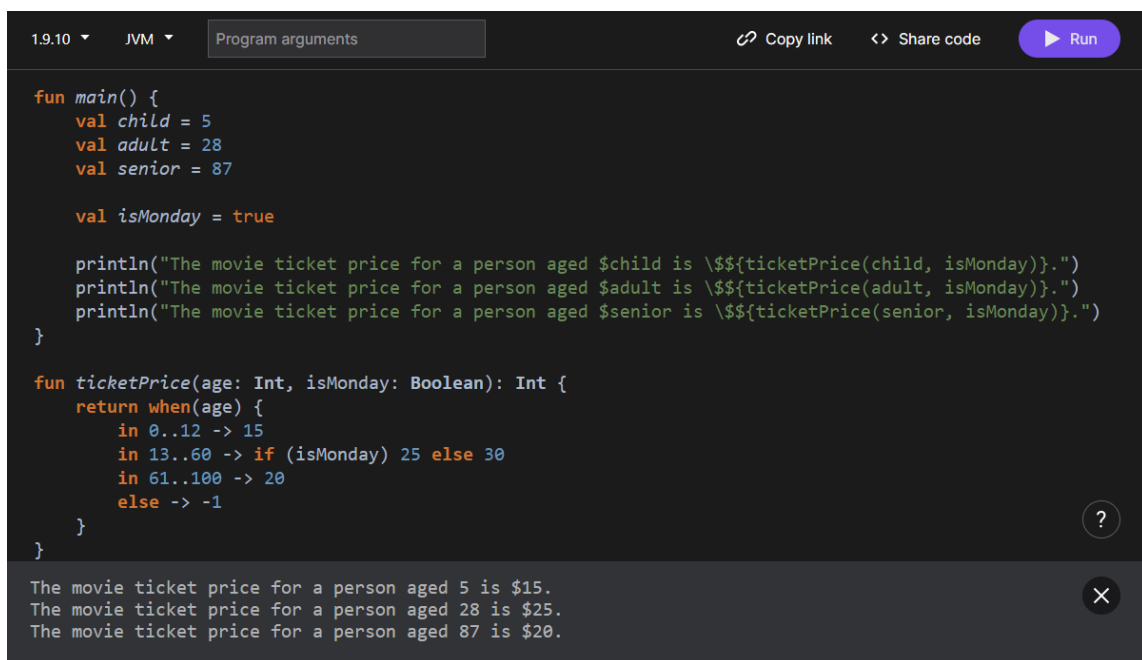
    fun turnOff() {
        println("Smart device is turned off.")
    }
}

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.
```

Figura 7. Utilización de variables de una función y variables en otra.

Precio de la entrada de cine



```
1.9.10 JVM Program arguments Copy link Share code Run

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

    val isMonday = true

    println("The movie ticket price for a person aged $child is \${ticketPrice(child, isMonday)}.")
    println("The movie ticket price for a person aged $adult is \${ticketPrice(adult, isMonday)}.")
    println("The movie ticket price for a person aged $senior is \${ticketPrice(senior, isMonday)}.")
}

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
    }
}

The movie ticket price for a person aged 5 is $15.
The movie ticket price for a person aged 28 is $25.
The movie ticket price for a person aged 87 is $20.
```

Figura 8. Código ejercicio precio entrada de cine

Conversor de temperatura

```
1.9.10 ▼ JVM ▼ Program arguments

fun main() {
    printFinalTemperature(27.0, "Celsius", "Fahrenheit") { celsiusToFahrenheit(it) }
    printFinalTemperature(350.0, "Kelvin", "Celsius") { kelvinToCelsius(it) }
    printFinalTemperature(10.0, "Fahrenheit", "Kelvin") { fahrenheitToKelvin(it) }
}

fun celsiusToFahrenheit(celsius: Double): Double {
    return 9/5 * celsius + 32
}

fun kelvinToCelsius(kelvin: Double): Double {
    return kelvin - 273.15
}

fun fahrenheitToKelvin(fahrenheit: Double): Double {
    return 5/9 * (fahrenheit - 32) + 273.15
}

fun printFinalTemperature(
    initialMeasurement: Double,
    initialUnit: String,
    finalUnit: String,
    conversionFormula: (Double) -> Double
) {
    val finalMeasurement = String.format("%.2f", conversionFormula(initialMeasurement)) // dos decimales
    println("$initialMeasurement degrees $initialUnit is $finalMeasurement degrees $finalUnit.")
}

27.0 degrees Celsius is 59.00 degrees Fahrenheit.
350.0 degrees Kelvin is 76.85 degrees Celsius.
10.0 degrees Fahrenheit is 273.15 degrees Kelvin.
```

Figura 8. Resultado de rellenar el código 'Conversor de temperatura'

Catálogo de canciones

```
1.9.10 ▼ JVM ▼ Program arguments

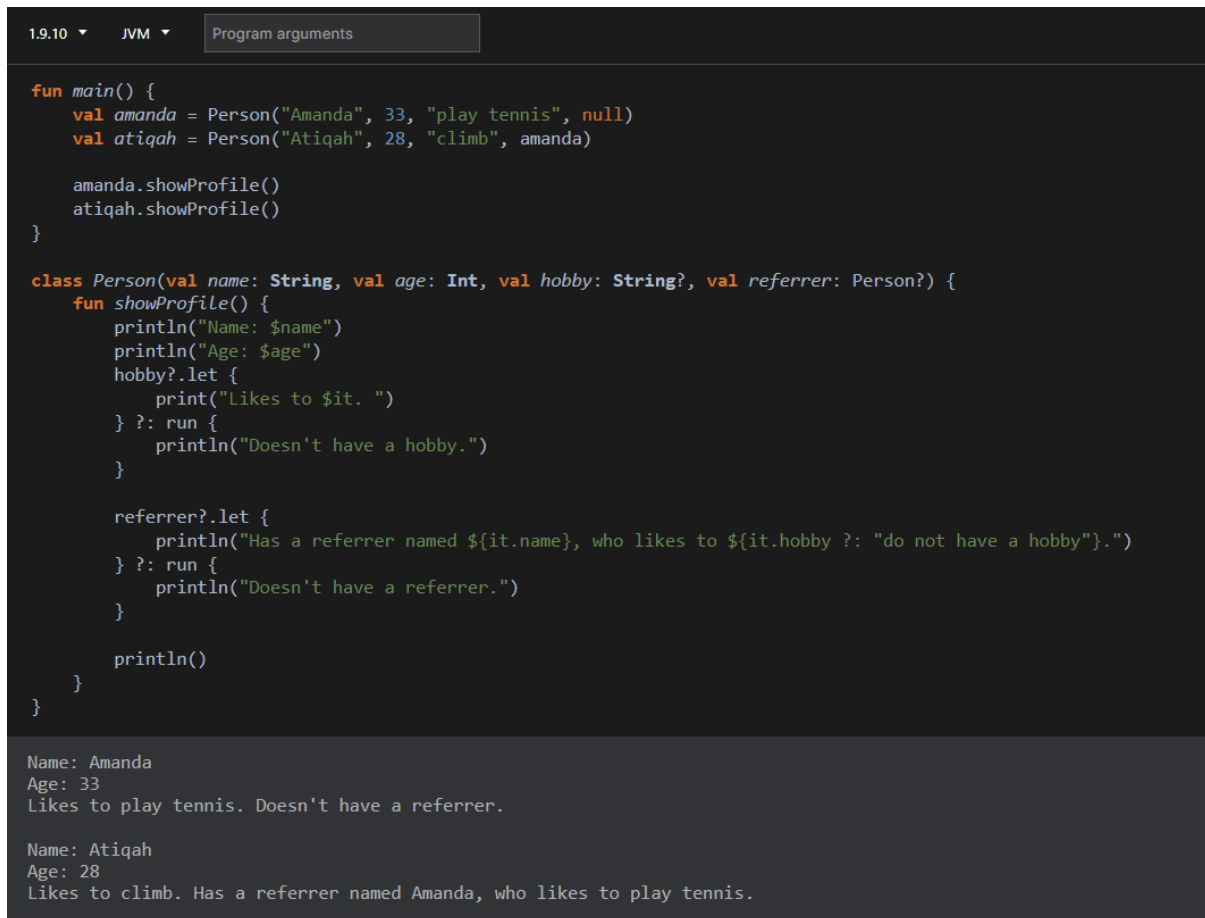
class Song(val title: String, val artist: String, val year: Int) {
    fun getDescription(): String {
        return "$title, interpretada por $artist, se lanzó en $year."
    }
}

fun main() {
    val mySong = Song("Don't You Worry Child", "Swedish House Mafia", 2012)
    println(mySong.getDescription())
}

Don't You Worry Child, interpretada por Swedish House Mafia, se lanzó en 2012.
```

Figura 9. Resultado de rellenar el código 'Catálogo de canciones'

Perfil de internet



```
1.9.10 ▼ JVM ▼ Program arguments

fun main() {
    val amanda = Person("Amanda", 33, "play tennis", null)
    val atiqah = Person("Atiqah", 28, "climb", amanda)

    amanda.showProfile()
    atiqah.showProfile()
}

class Person(val name: String, val age: Int, val hobby: String?, val referrer: Person?) {
    fun showProfile() {
        println("Name: $name")
        println("Age: $age")
        hobby?.let {
            print("Likes to $it. ")
        } ?: run {
            println("Doesn't have a hobby.")
        }

        referrer?.let {
            println("Has a referrer named ${it.name}, who likes to ${it.hobby ?: "do not have a hobby"}.")
        } ?: run {
            println("Doesn't have a referrer.")
        }

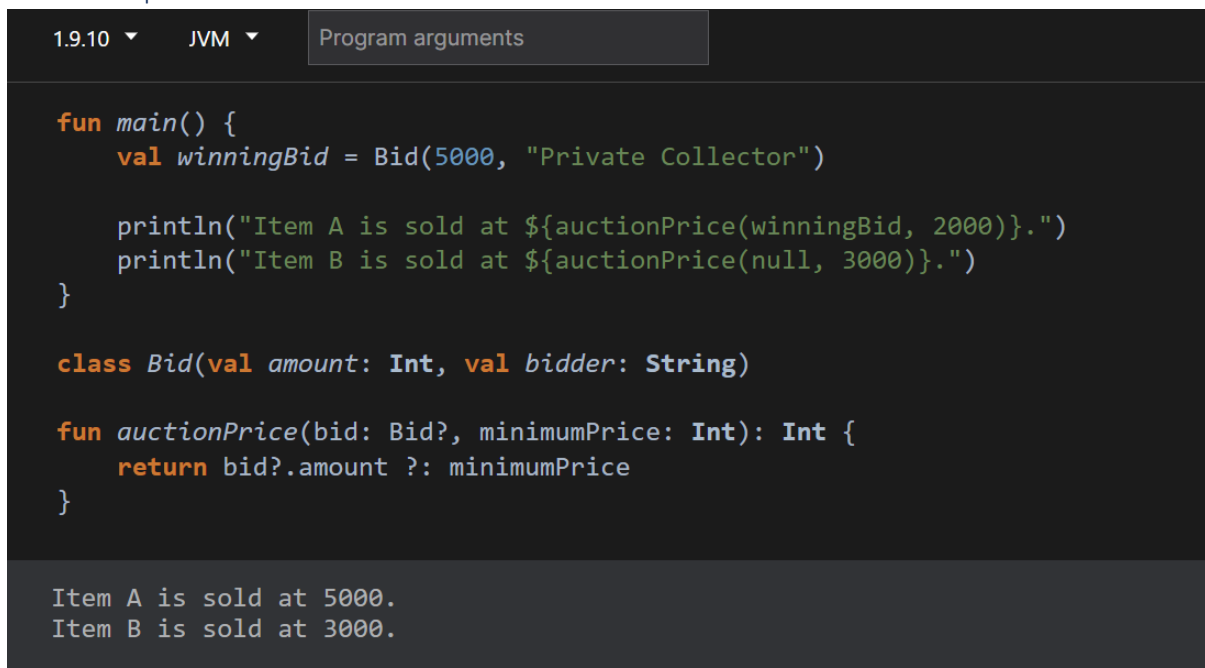
        println()
    }
}

Name: Amanda
Age: 33
Likes to play tennis. Doesn't have a referrer.

Name: Atiqah
Age: 28
Likes to climb. Has a referrer named Amanda, who likes to play tennis.
```

Figura 10. Resultado de rellenar el código 'Perfil de internet'

Subasta Especial



```
1.9.10 ▼ JVM ▼ Program arguments

fun main() {
    val winningBid = Bid(5000, "Private Collector")

    println("Item A is sold at ${auctionPrice(winningBid, 2000)}.")
    println("Item B is sold at ${auctionPrice(null, 3000)}.")
}

class Bid(val amount: Int, val bidder: String)

fun auctionPrice(bid: Bid?, minimumPrice: Int): Int {
    return bid?.amount ?: minimumPrice
}

Item A is sold at 5000.
Item B is sold at 3000.
```

Figura 11. Resultado de rellenar el código 'Subasta especial'

RUTA DE APRENDIZAJE 2: Agrega un botón a una app

En este apartado, el propósito principal fue desarrollar una aplicación interactiva para Android, mediante el uso de Compose, y se incorporó un botón a la interfaz de usuario (UI). Para ello, se definieron funciones de componibilidad y se importaron recursos drawable con el fin de exhibir una imagen mediante con “image”.

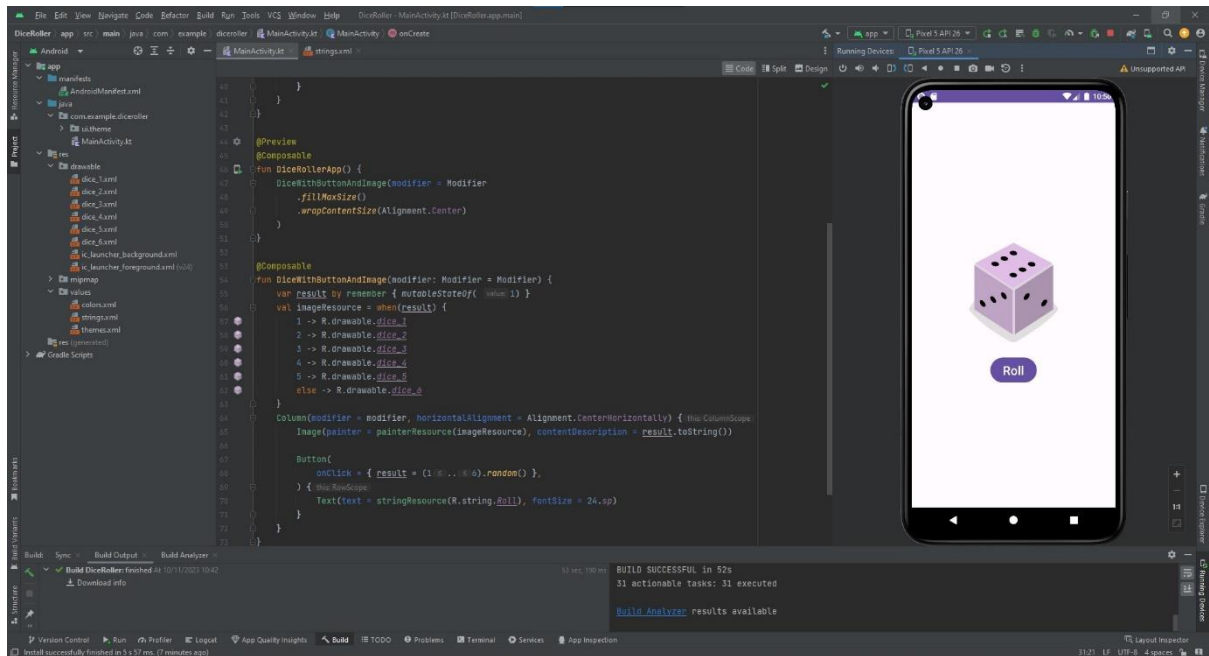


Figura 12. Ejecución de Dice Roller

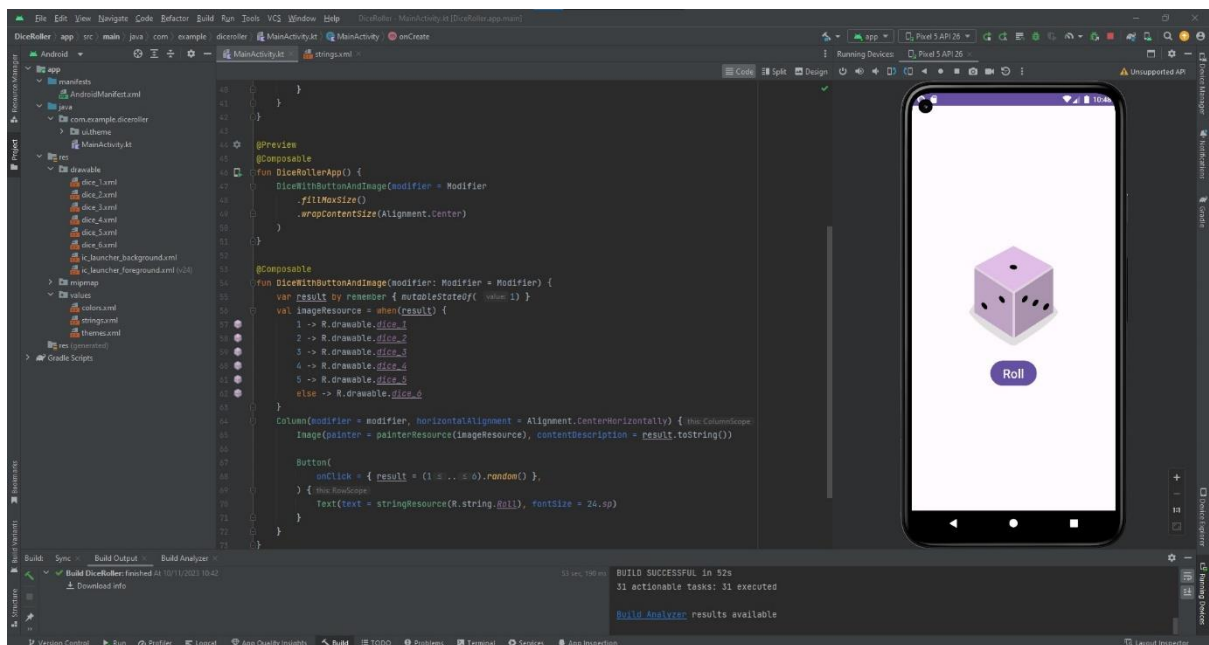


Figura 13. Ejecución (2) Dice Roller

Además, con este mismo código, se ha puesto en práctica el uso del depurador en Android Studio.

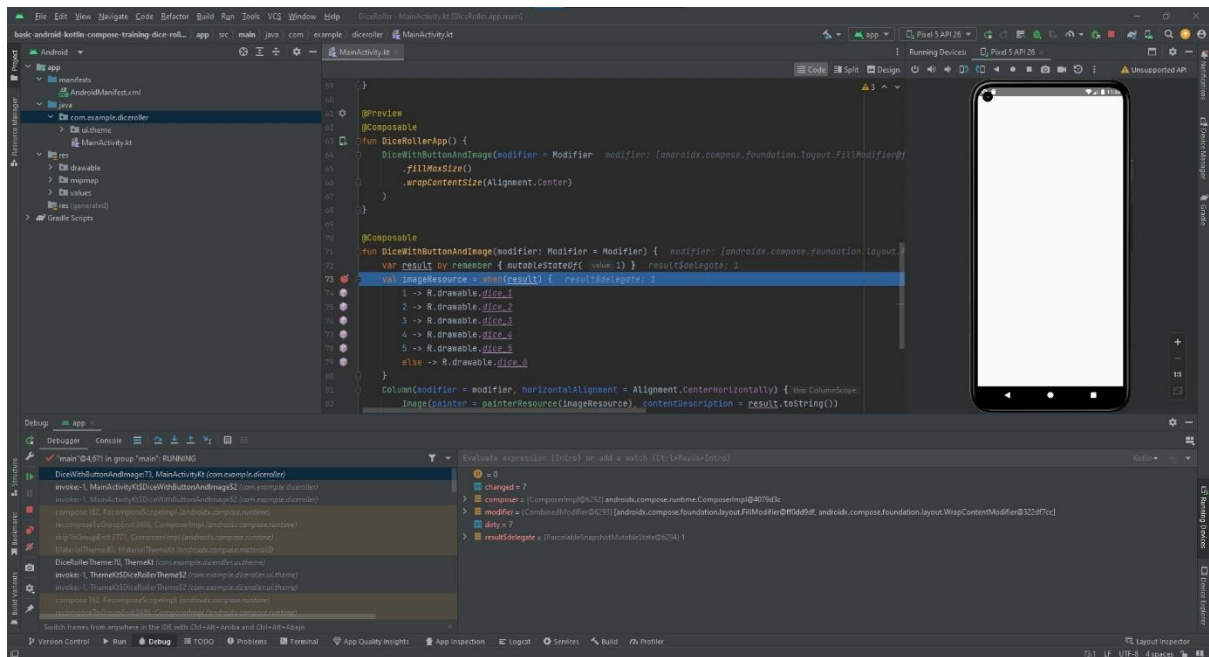


Figura 14. Uso del depurador de Android Studio

En el último apartado de esta ruta de aprendizaje, se ha procedido a desarrollar una aplicación interactiva, que, al igual que Dice Roller, ofrece a los usuarios la posibilidad de “exprimir” un limón y preparar una limonada. Se va a indicar cómo funciona la aplicación:

1. Se inicia la aplicación y se invita al usuario a presionar la pantalla para “coger” un limón.

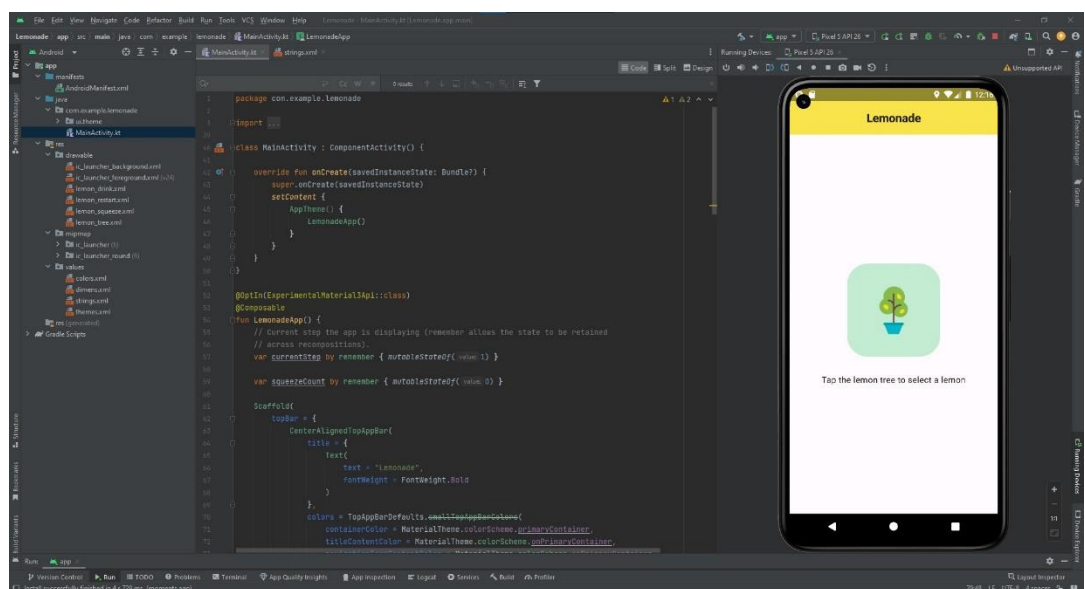


Figura 15. Bienvenida de LemonadeApp

2. Tras pulsar en la pantalla, nos aparecerá la siguiente figura invitando al usuario a volver a pulsar para “exprimir” el limón

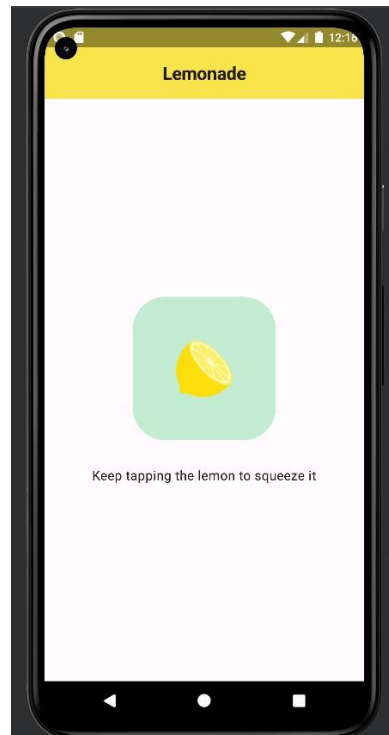


Figura 16. Segunda pantalla de “LemonadeApp”

3. Una vez se ha exprimido el limón, podemos ver que aparece una imagen de un vaso de limonada y se invita al usuario a volver a presionar sobre la pantalla para “beberse” la limonada.

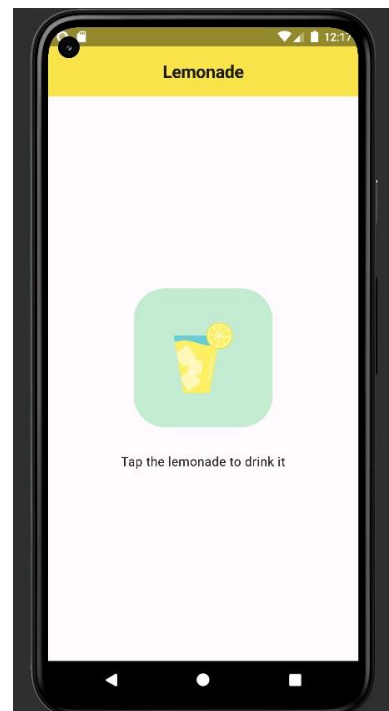


Figura 17. Tercera pantalla de “LemonadeApp”

4. Tras haberse bebido la limonada, aparece la imagen de un vaso vacío y se invita al usuario a volver a pulsar la pantalla con el fin de repetir el proceso descrito.



Figura 18. Cuarta pantalla de "LemonadeApp"

RUTA DE APRENDIZAJE 3: Interactúa con la UI y el estado

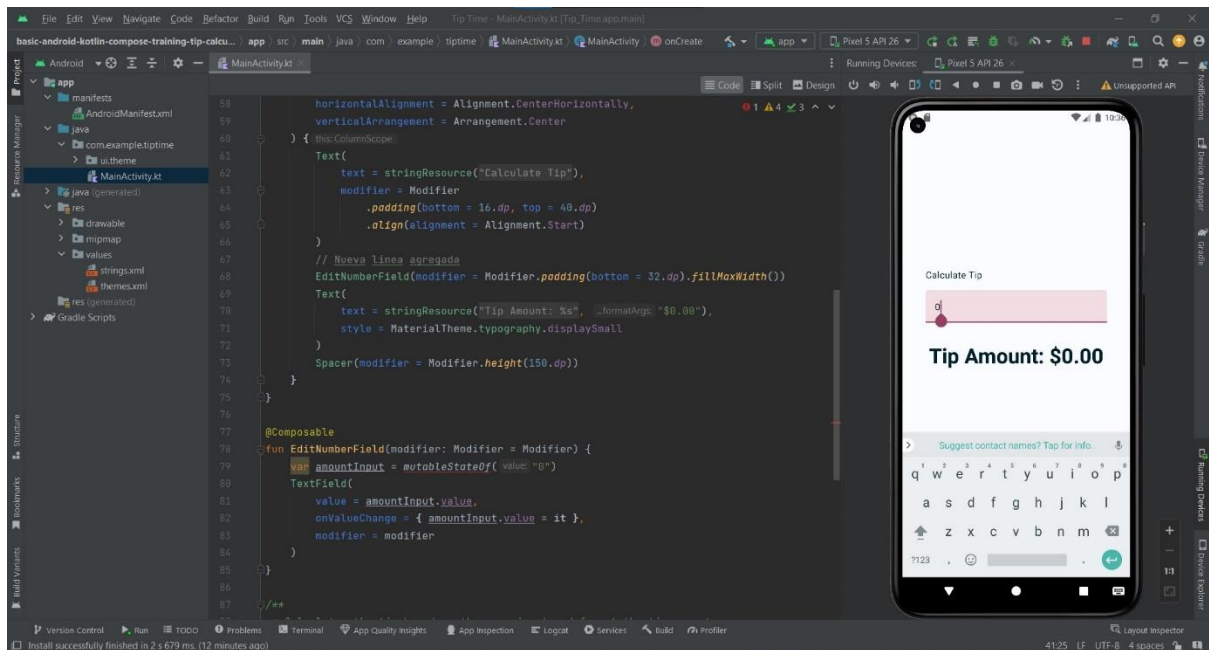


Figura 19. Estado inicial de la aplicación

En este apartado, se ha desarrollado una pequeña aplicación que ayuda al cálculo de propinas de un restaurante. La función principal de esta aplicación es permitir al usuario ingresar la cuenta total para posteriormente, calcular un 15% de esa cuenta.

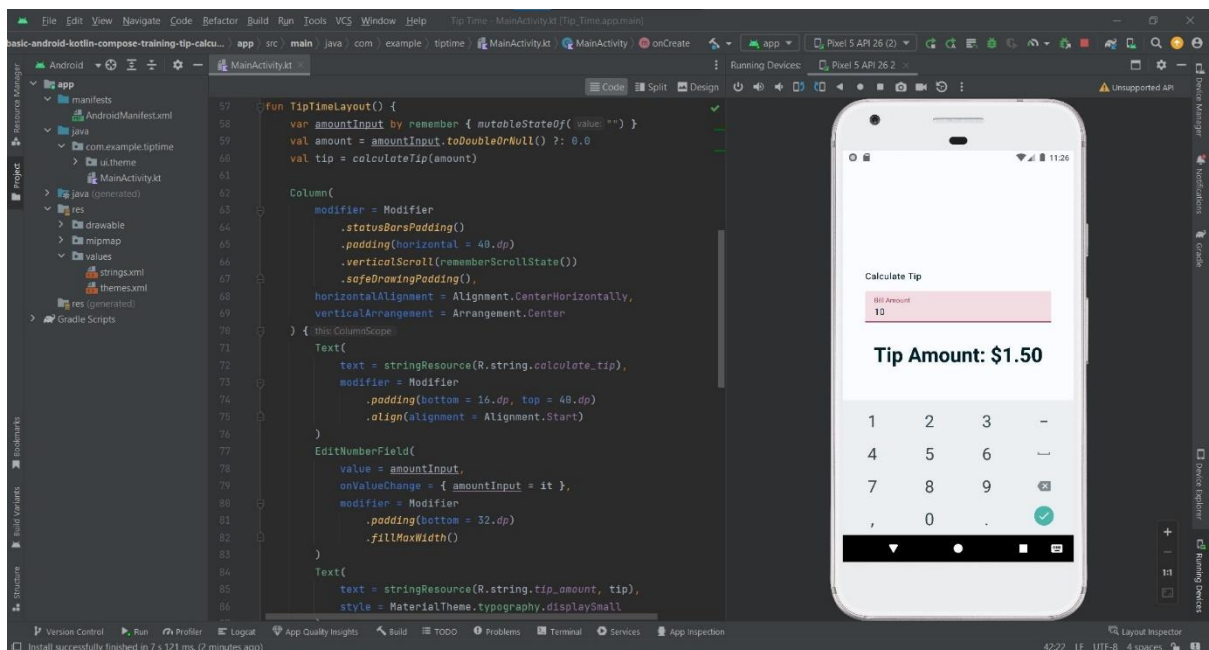


Figura 20. Cálculo del 15% de la cuenta

En el segundo apartado, se ha añadido la funcionalidad de calcular el monto final de la cuenta tras consultar al usuario qué porcentaje desea aplicar a su cuenta, mediante un nuevo campo de texto.

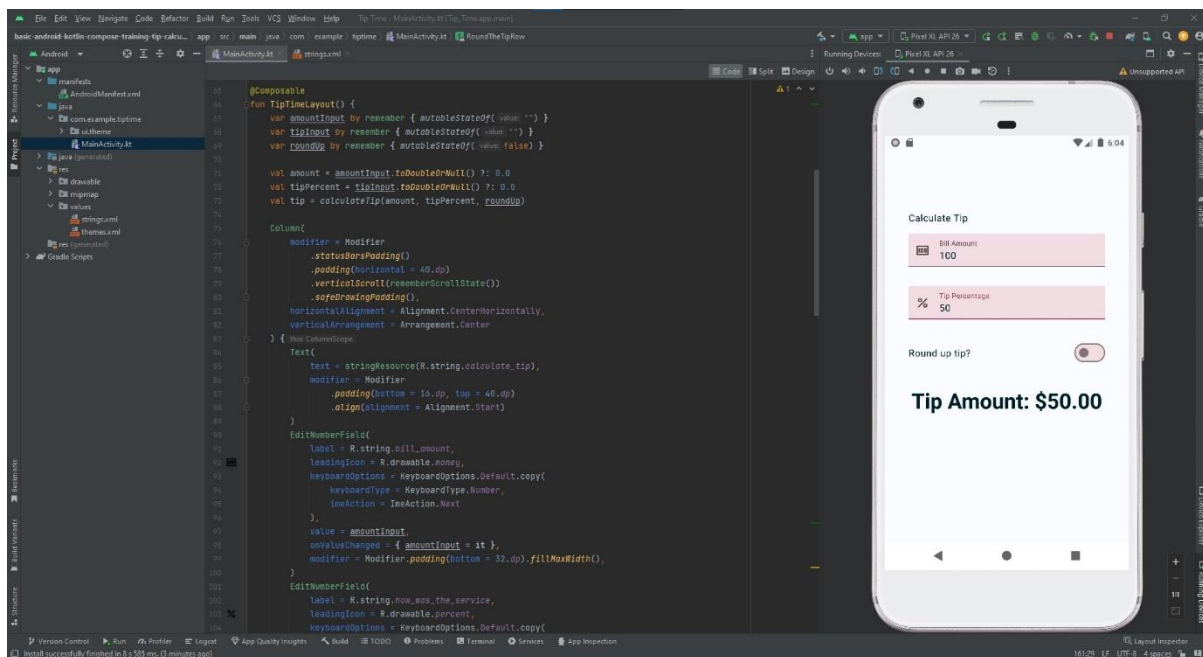


Figura 21. Cálculo de propina tras preguntar al cliente

En el tercer apartado, se añadió un interruptor que redondea hacia arriba la propina calculada tras preguntar al usuario. En la figura 22, se puede observar como al no estar activado el interruptor, la propina no se redondea, sin embargo, en la figura 23 se observa cómo al activar el mismo, se redondea la cifra.

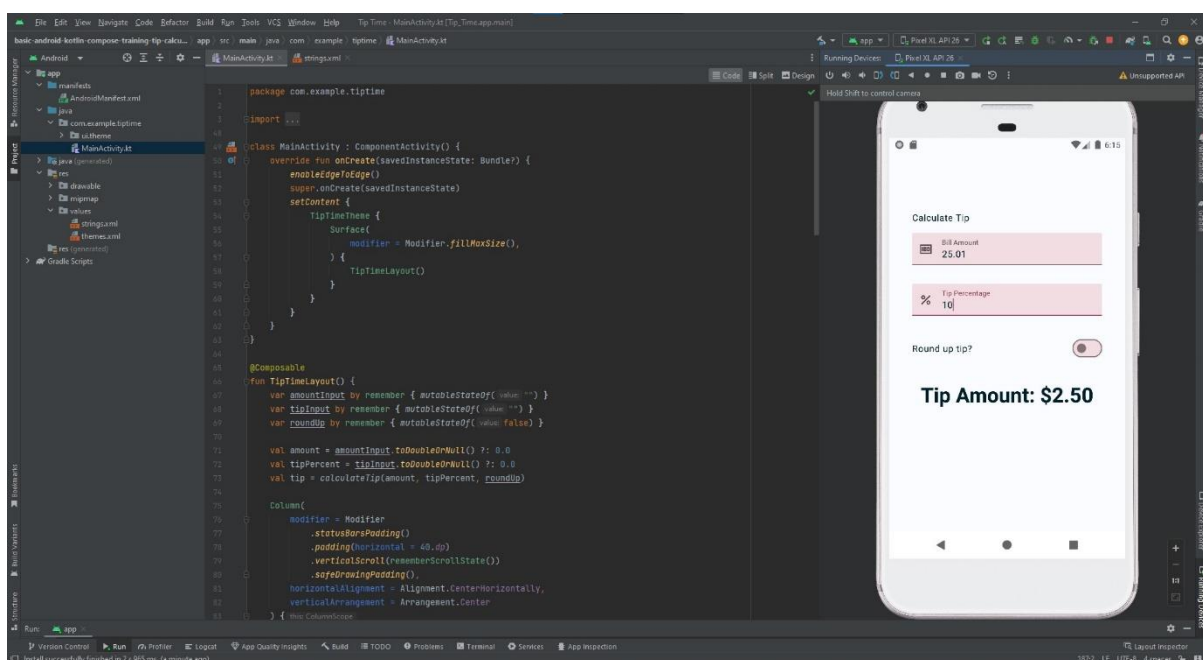


Figura 22. Cálculo de propina sin redondear

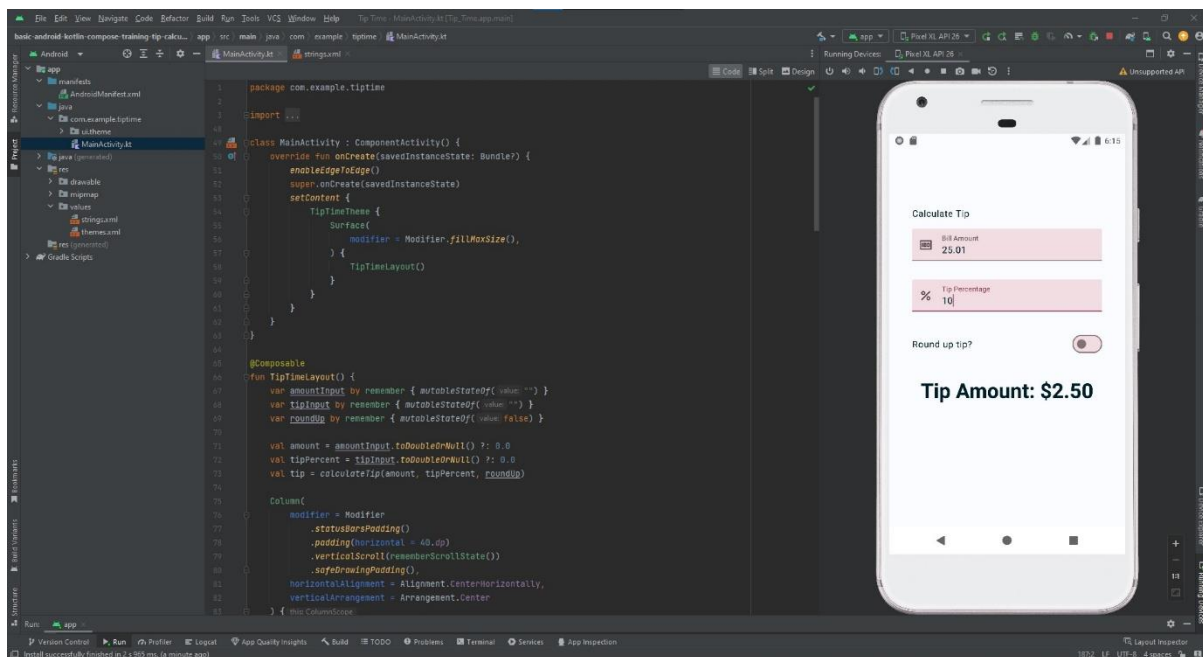


Figura 23. Cálculo de propina redondeando

En el último apartado de esta ruta de aprendizaje, se han abordado pruebas locales y de instrumentación, aplicándolas a esta misma aplicación. Las pruebas locales consisten en evaluar directamente los métodos desde el código de la aplicación. Por lo tanto, es crucial que los métodos que se deseen probar sean accesibles para las clases y métodos de prueba.

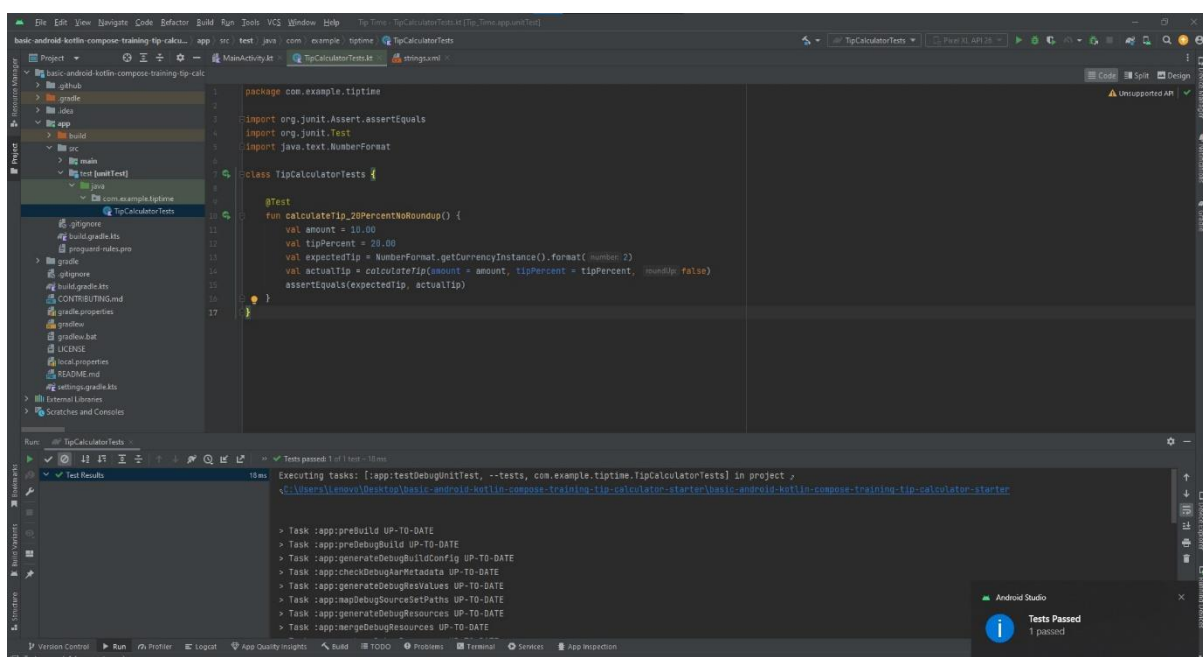


Figura 24. Ejemplo de prueba local.

Las pruebas de instrumentación se centran en evaluar una instancia real de la aplicación junto con su interfaz de usuario (IU). En este tipo de pruebas, es crucial configurar el contenido de la IU de manera similar a como se hace en el método "onCreate()" del archivo "MainActivity.kt" al programar el código de la aplicación.

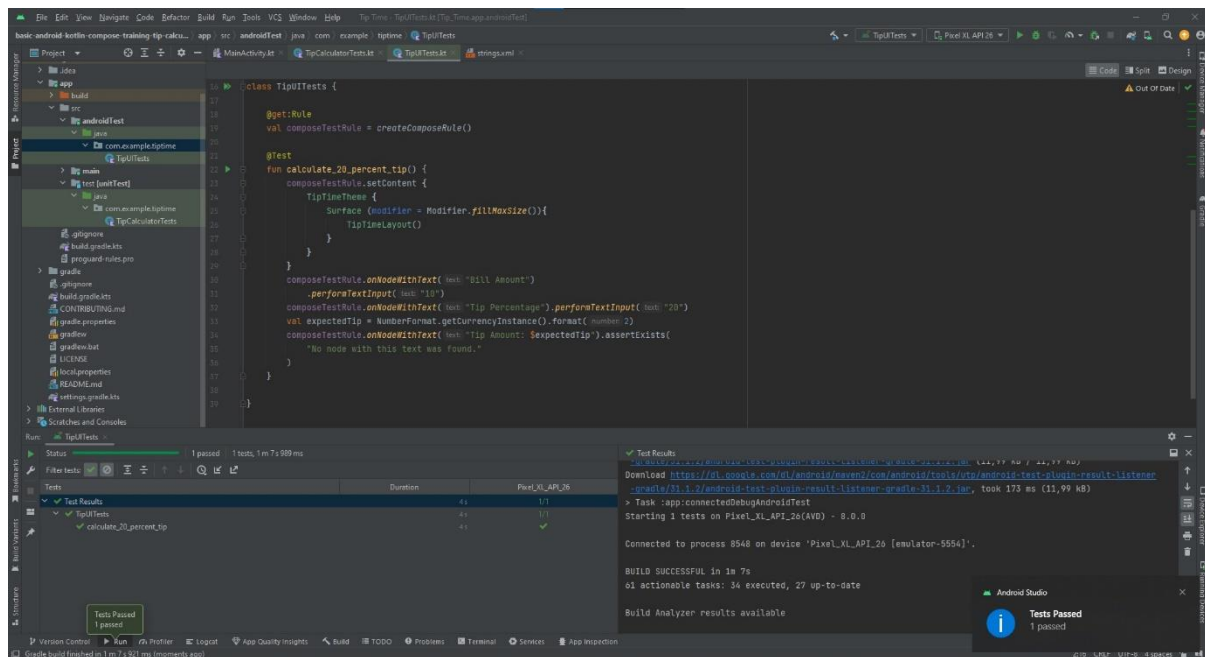


Figura 25. Ejemplo de prueba de instrumentación

OPINIÓN

Mi participación en este CodeLab ha sido una experiencia realmente positiva, ya que me ha permitido seguir aprendiendo sobre el desarrollo de aplicaciones móviles en el entorno Android.

En cuanto a la primera ruta de aprendizaje, me ha resultado interesante el conocer algunas formas de declarar distintas sentencias, como por ejemplo el 'when'. Y en lo que a la tercera ruta de aprendizaje, me ha resultado muy interesante la implementación de un interruptor para activar o desactivar funciones, en el caso del ejemplo trabajado, activar o desactivar el redondeo de la propina. Así como, me ha resultado muy útil el aprender a implementar test del código desarrollado.

ENLACE A GITHUB

<https://github.com/pablosanttanaa/PAMN.git>

BIBLIOGRAFÍA

[1] Unidad 2: Compila la UI de una app. Google for Developers. [Recurso en línea]. Disponible en: <https://developer.android.com/courses/android-basics-compose/unit-2?hl=es-419>