



Instituto Tecnológico de Ciudad Madero

INGENIERÍA EN SISTEMAS COMPUTACIONALES

EQUIPO

PEREZ ANASTASIO KARLA ZAFIRO 20070574

GARAY HERNANDEZ MIGUEL ENRIQUE 20070600

MATERIA

Programación nativa para móviles

TAREA

Tarea No 8 Unidad 2 Creación de la IU de una app



Conceptos Basicos de kotlin

<https://github.com/Zafirows/Programacion-nativa>

```
1 fun main() { new*
2     println(1 == 1)
3 }
4
5
6 /*
7 fun main() {
8     println(1 < 1)
9 }
10 */
```

Run CondicionalesKt

"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=53115" -Dfile.encoding=UTF-8 -Dsun.java2d.d3d=false

Process finished with exit code 0

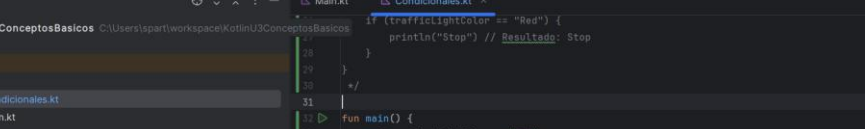
```
16 }
17 /*
18
19 |
20 // Este es el bloque activo que se ejecuta actualmente.
21 fun main() {
22     // Se define una constante con el color del semáforo
23     val trafficLightColor = "Red"
24
25     // Si el color es "Red", imprime "Stop"
26     if (trafficLightColor == "Red") {
27         println("Stop") // Resultado: Stop
28     }
29 }
30
```

Run CondicionalesKt

"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=53363" -Dfile.encoding=UTF-8 -Dsun.java2d.d3d=false

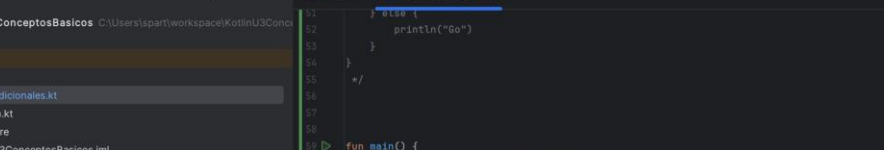
Stop

Process finished with exit code 0



The screenshot shows the IntelliJ IDEA IDE with the following details:

- Project View (Left):** Shows the project structure for 'KotlinU3ConceptosBasicos'. The 'src' directory contains 'Condicionales.kt' and 'Main.kt'. The 'out' directory is also visible.
- Code Editor (Center):** Displays the code in 'Condicionales.kt'. The code defines a variable 'trafficLightColor' and a 'main' function that prints 'Stop' when the color is 'Red'.
- Run Console (Bottom):** Shows the command executed to run the program: `"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=53490" -Dfile.encoding=UTF-8 -Dsun.` The output is 'Stop'.



The screenshot shows the IntelliJ IDEA IDE with the following details:

- Project View:** The project is named "KotlinU3ConceptosBasicos". The source files are "Condicionales.kt" and "Main.kt".
- Code Editor:** The code in "Main.kt" is as follows:

```
51 } else {  
52     println("Go")  
53 }  
54 }  
55 */  
56  
57  
58  
59 fun main() {  
60     val trafficLightColor = "Yellow"  
61  
62     if (trafficLightColor == "Red") {  
63         println("Stop")  
64     } else if (trafficLightColor == "Yellow") {  
65         println("Slow")  
66     } else {  
67         println("Go")  
68     }  
69 }  
70
```
- Run Configuration:** The "Run" button is highlighted, and the "Run" configuration window is open. The configuration is named "CondicionalesKt". The command to run the program is:

```
C:\Program Files\Java\jdk-24\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=53712" -Dfile.encoding=UTF-8 -Dsun.Slow
```
- Run Output:** The "Run" output window shows the message "Process finished with exit code 0".

The screenshot shows the IntelliJ IDEA IDE with a Kotlin file named `Condicionales.kt` open. The file contains the following code:

```

71
72
73
74 fun main() {
75     val trafficLightColor = "Black"
76
77     if (trafficLightColor == "Red") {
78         println("Stop")
79     } else if (trafficLightColor == "Yellow") {
80         println("Slow")
81     } else {
82         println("Go")
83     }
84 }
85

```

The Run tab at the bottom shows the command executed to run the program:

```

C:\Program Files\Java\jdk-24\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=53750" -Dfile.encoding=UTF-8 -Dsun

```

Process finished with exit code 0



```
Project
├── KotlinU3ConceptosBasicos
│   ├── .idea
│   ├── out
│   └── src
│       ├── Condicionales.kt
│       ├── Main.kt
│       ├── .gitignore
│       ├── KotlinU3ConceptosBasicos.iml
│       └── External Libraries
└── Scratches and Consoles

Run
CondicionalesKt
"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=53826" -Dfile.encoding=UTF-8 -Dsun.
Invalid traffic-light color
Process finished with exit code 0
```

```
Project
├── KotlinU3ConceptosBasicos
│   ├── .idea
│   ├── out
│   └── src
│       ├── Condicionales.kt
│       ├── Main.kt
│       ├── .gitignore
│       ├── KotlinU3ConceptosBasicos.iml
│       └── External Libraries
└── Scratches and Consoles

Run
CondicionalesKt
"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=54192" -Dfile.encoding=UTF-8 -Dsun.
Slow
Process finished with exit code 0
```

```
Project
├── KotlinU3ConceptosBasicos
│   ├── .idea
│   ├── out
│   └── src
│       ├── Condicionales.kt
│       ├── Main.kt
│       ├── .gitignore
│       ├── KotlinU3ConceptosBasicos.iml
│       └── External Libraries
└── Scratches and Consoles

Run
CondicionalesKt
"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=54346" -Dfile.encoding=UTF-8 -Dsun.
x is a prime number between 1 and 10.
Process finished with exit code 0
```



```
Project
└─ KotlinU3ConceptosBasicos
   └─ src
      └─ Condicionales.kt
         └─ Main.kt
            └─ .gitignore
               └─ KotlinU3ConceptosBasicos.iml
                  └─ External Libraries
                     └─ Scratches and Consoles
```

```
130 when (x) {
131     2 -> println("x is a prime number between 1 and 10.")
132     3 -> println("x is a prime number between 1 and 10.")
133     5 -> println("x is a prime number between 1 and 10.")
134     7 -> println("x is a prime number between 1 and 10.")
135     else -> println("x isn't a prime number between 1 and 10.")
136 }
137
138 */
139
140
141 fun main() {
142     val x = 3
143
144     when (x) {
145         2, 3, 5, 7 -> println("x is a prime number between 1 and 10.")
146         else -> println("x isn't a prime number between 1 and 10.")
147     }
148 }
149
```

```
Run CondicionalesKt
"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=54394" -Dfile.encoding=UTF-8 -Dsun.
x is a prime number between 1 and 10.
```

```
Project
└─ KotlinU3ConceptosBasicos
   └─ src
      └─ Condicionales.kt
         └─ Main.kt
            └─ .gitignore
               └─ KotlinU3ConceptosBasicos.iml
                  └─ External Libraries
                     └─ Scratches and Consoles
```

```
147 }
148
149 */
150
151
152 fun main() {
153     val x = 4
154
155     when (x) {
156         2, 3, 5, 7 -> println("x is a prime number between 1 and 10.")
157         in 1..10 -> println("x is a number between 1 and 10, but not a prime number.")
158         else -> println("x isn't a prime number between 1 and 10.")
159     }
160 }
```

```
Run CondicionalesKt
"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=54478" -Dfile.encoding=UTF-8 -Dsun.
x is a number between 1 and 10, but not a prime number.
```

```
Project
└─ KotlinU3ConceptosBasicos
   └─ src
      └─ Condicionales.kt
         └─ Main.kt
            └─ .gitignore
               └─ KotlinU3ConceptosBasicos.iml
                  └─ External Libraries
                     └─ Scratches and Consoles
```

```
155 when (x) {
156     2, 3, 5, 7 -> println("x is a prime number between 1 and 10.")
157     in 1..10 -> println("x is a number between 1 and 10, but not a prime number.")
158     else -> println("x isn't a prime number between 1 and 10.")
159 }
160
161 */
162
163
164 fun main() {
165     val x: Any = 20
166
167     when (x) {
168         2, 3, 5, 7 -> println("x is a prime number between 1 and 10.")
169         in 1..10 -> println("x is a number between 1 and 10, but not a prime number.")
170         is Int -> println("x is an integer number, but not between 1 and 10.")
171         else -> println("x isn't an integer number.")
172     }
173 }
174
```

```
Run CondicionalesKt
"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=54580" -Dfile.encoding=UTF-8 -Dsun.
x is an integer number, but not between 1 and 10.
```



```
Project
└─ KotlinU3ConceptosBasicos
   └─ src
      └─ Condicionales.kt
         └─ Main.kt
            └─ KotlinU3ConceptosBasicos.iml
               └─ External Libraries
                  └─ Scratches and Consoles
```

```
169 when (x) {
170     2, 3, 5, 7 -> println("x is a prime number between 1 and 10.")
171     in 1..10 -> println("x is a number between 1 and 10, but not a prime number.")
172     is Int -> println("x is an integer number, but not between 1 and 10.")
173     else -> println("x isn't an integer number.")
174 }
175
176 */
177
178 fun main() {
179     val trafficLightColor = "Amber"
180
181     when (trafficLightColor) {
182         "Red" -> println("Stop")
183         "Yellow", "Amber" -> println("Slow")
184         "Green" -> println("Go")
185         else -> println("Invalid traffic-light color")
186     }
187 }
```

Run CondicionalesKt

"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=54770" -Dfile.encoding=UTF-8 -Dsun.Slow

```
Project
└─ KotlinU3ConceptosBasicos
   └─ src
      └─ Condicionales.kt
         └─ Main.kt
            └─ KotlinU3ConceptosBasicos.iml
               └─ External Libraries
                  └─ Scratches and Consoles
```

```
186 "Red" -> println("Stop")
187 "Yellow", "Amber" -> println("Slow")
188 "Green" -> println("Go")
189 else -> println("Invalid traffic-light color")
190 }
191
192 */
193
194 fun main() {
195     val trafficLightColor = "Black"
196
197     val message =
198         if (trafficLightColor == "Red") "Stop"
199         else if (trafficLightColor == "Yellow") "Slow"
200         else if (trafficLightColor == "Green") "Go"
201         else "Invalid traffic-light color"
202
203     println(message)
204 }
```

Run CondicionalesKt

"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=54843" -Dfile.encoding=UTF-8 -Dsun.Invalid traffic-light color

```
Project
└─ KotlinU3ConceptosBasicos
   └─ src
      └─ Condicionales.kt
         └─ Main.kt
            └─ KotlinU3ConceptosBasicos.iml
               └─ External Libraries
                  └─ Scratches and Consoles
```

```
207
208
209 fun main() {
210     val trafficLightColor = "Amber"
211
212     val message = when(trafficLightColor) {
213         "Red" -> "Stop"
214         "Yellow", "Amber" -> "Slow"
215         "Green" -> "Go"
216         else -> "Invalid traffic-light color"
217     }
218     println(message)
219 }
```

Run CondicionalesKt

"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=55012" -Dfile.encoding=UTF-8 -Dsun.Slow

KotlinU3ConceptosBasicos master

Project

KotlinU3ConceptosBasicos

idea

out

src

Condicionales.kt

Main.kt

Nuladidad.kt

gitignore

KotlinU3ConceptosBasicos.iml

External Libraries

Scratches and Consoles

Main.kt

Condicionales.kt

Nuladidad.kt

```
1 fun main() {
2     val favoriteActor = null
3     println(favoriteActor)
4 }
```

Run NuladidadKt

```
"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=55415" -Dfile.encoding=UTF-8 -Dsun
null
Process finished with exit code 0
```

KotlinU3ConceptosBasicos master

Project

KotlinU3ConceptosBasicos

idea

out

src

Condicionales.kt

Main.kt

Nuladidad.kt

gitignore

KotlinU3ConceptosBasicos.iml

External Libraries

Scratches and Consoles

Main.kt

Condicionales.kt

Nuladidad.kt

```
7
8
9 /*
10 fun main() {
11     var favoriteActor: String? = "Sandra Oh"
12     println(favoriteActor)
13
14     favoriteActor = null
15     println(favoriteActor)
16 }
17 */
18
19 fun main() {
20     var number: Int? = 10
21     println(number)
22
23     number = null
24     println(number)
25 }
```

Run NuladidadKt

```
"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=55935" -Dfile.encoding=UTF-8 -Dsun
10
null
```



The screenshot shows the IntelliJ IDEA IDE with the following details:

- Project Structure:** The project is named "KotlinU3ConceptosBasicos". The source files are located in the "src" directory under "KotlinU3ConceptosBasicos". The files listed are "Condicionales.kt", "Main.kt", and "Nuladidad.kt".
- Code Editor:** The "Nuladidad.kt" file is open. It contains the following Kotlin code:

```
15 println(favoriteActor)
16 }
17 */
18
19 /*
20
21 fun main() {
22     var number: Int? = 10
23     println(number)
24
25     number = null
26     println(number)
27 }
28 */
29
30 fun main() { new *
31     var favoriteActor: String = "Sandra Oh"
32     println(favoriteActor.length)
33 }
```
- Run Console:** The console shows the command used to run the program: `"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=55974" -Dfile.encoding=UTF-8 -Dsun`

The screenshot shows the IntelliJ IDEA IDE interface. On the left, the Project Explorer displays the structure of the 'KotlinU3ConceptosBasicos' project, including files like 'Main.kt', 'Nulidad.kt', and 'Condicionales.kt'. The main editor window shows the code for 'Nulidad.kt', which includes a main function that prints the length of a favorite actor's name. The Run tab at the bottom shows the command used to execute the program.

```
26 println(number)
27 }
28 */
29
30
31 /*
32 fun main() {
33     var FavoriteActor: String = "Sandra Oh"
34     println(favoriteActor.length)
35 }
36 */
37
38
39
40 fun main() { new *
41     var favoriteActor: String? = null
42     println(favoriteActor?.length)
43 }
44
```

Run NulidadKt

"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=56918" -Dfile.encoding=UTF-8 -Dsun.

Project

KotlinU3ConceptosBasicos

idea

out

src

Condicionales.kt

Main.kt

Nuladidad.kt

gitignore

KotlinU3ConceptosBasicos.iml

External Libraries

Scratches and Consoles

Main.kt

Condicionales.kt

Nuladidad.kt

```
31 /*
32 fun main() {
33     var favoriteActor: String = "Sandra Oh"
34     println(favoriteActor.length)
35 }
36 */
37
38 /*
39 fun main() {
40     var favoriteActor: String? = null
41     println(favoriteActor?.length)
42 }
43 */
44
45
46 fun main() { new
47     var favoriteActor: String? = null
48     println(favoriteActor!!.length)
49 }
```

Run NuladidadKt

```
"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=56052" -Dfile.encoding=UTF-8 -Dsun.
Exception in thread "main" java.lang.NullPointerException: Create breakpoint
    at NuladidadKt.main(Nuladidad.kt:48)
    at NuladidadKt.main(Nuladidad.kt)
```

Project

KotlinU3ConceptosBasicos

idea

out

src

Condicionales.kt

Main.kt

Nuladidad.kt

gitignore

KotlinU3ConceptosBasicos.iml

External Libraries

Scratches and Consoles

Main.kt

Condicionales.kt

Nuladidad.kt

```
42     println(favoriteActor?.length)
43 }
44 */
45
46 /*
47 fun main() {
48     var favoriteActor: String? = null
49     println(favoriteActor!!.length)
50 }
51 */
52
53
54 fun main() { new
55     var favoriteActor: String? = "Sandra Oh"
56
57     if (favoriteActor != null) {
58         println("The number of characters in your favorite actor's name is ${favoriteActor.length}.")
59     }
60 }
```

Run NuladidadKt

```
"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=56087" -Dfile.encoding=UTF-8 -Dsun.
The number of characters in your favorite actor's name is 9.
```

Project

KotlinU3ConceptosBasicos

idea

out

src

Condicionales.kt

Main.kt

Nuladidad.kt

gitignore

KotlinU3ConceptosBasicos.iml

External Libraries

Scratches and Consoles

Main.kt

Condicionales.kt

Nuladidad.kt

```
57     var favoriteActor: String? = "Sandra Oh"
58
59     if (favoriteActor != null) {
60         println("The number of characters in your favorite actor's name is ${favoriteActor.length}.")
61     }
62 }
63 */
64
65
66 fun main() { new
67     var favoriteActor: String? = null
68
69     if(favoriteActor != null) {
70         println("The number of characters in your favorite actor's name is ${favoriteActor.length}.")
71     } else {
72         println("You didn't input a name.")
73     }
74 }
```

Run NuladidadKt

```
"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=56147" -Dfile.encoding=UTF-8 -Dsun.
You didn't input a name.
```



```
Project
└─ KotlinU3ConceptosBasicos
   └─ src
      ├── Condicionales.kt
      ├── Main.kt
      └─ Nuladidad.kt
         ├── .gitignore
         ├── KotlinU3ConceptosBasicos.iml
         └─ External Libraries
            └─ Scratches and Consoles

Run
NuladidadKt

C:\Program Files\Java\jdk-24\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=56218" -Dfile.encoding=UTF-8 -Dsun.
The number of characters in your favorite actor's name is 9.
```

```
Project
└─ KotlinU3ConceptosBasicos
   └─ src
      ├── Condicionales.kt
      ├── Main.kt
      └─ Nuladidad.kt
         ├── .gitignore
         ├── KotlinU3ConceptosBasicos.iml
         └─ External Libraries
            └─ Scratches and Consoles

Run
NuladidadKt

C:\Program Files\Java\jdk-24\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=56286" -Dfile.encoding=UTF-8 -Dsun.
The number of characters in your favorite actor's name is 9.
Process finished with exit code 0
```

```
Project
└─ KotlinU3ConceptosBasicos
   └─ src
      ├── ClasesObjetos.kt
      ├── Condicionales.kt
      ├── Main.kt
      └─ Nuladidad.kt
         ├── .gitignore
         ├── KotlinU3ConceptosBasicos.iml
         └─ External Libraries
            └─ Scratches and Consoles

Run
ClasesObjetosKt

C:\Program Files\Java\jdk-24\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=56966" -Dfile.encoding=UTF-8 -Dsun.
Smart device is turned on.
Smart device is turned off.
```



```
Project >
  KotlinU3ConceptosBasicos
    src
      ClasesObjetos.kt
      Condicionales.kt
      Main.kt
      Nuladidad.kt
    External Libraries
    Scratches and Consoles

Run ClasesObjetosKt x

"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=57173" -Dfile.encoding=UTF-8 -Dsun.java2d.c
Device name is: Android TV
Smart device is turned on.
Smart device is turned off.
Process finished with exit code 0
```

```
Project >
  KotlinU3ConceptosBasicos
    src
      ClasesObjetos.kt
      Condicionales.kt
      Main.kt
      Nuladidad.kt
    External Libraries
    Scratches and Consoles

Run ClasesObjetosKt x

"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=57912" -Dfile.encoding=UTF-8 -Dsun.java2d.c
Android TV is turned on. Speaker volume is set to 2 and channel number is set to 1.
Google Light turned on. The brightness level is 2.
```

```
Project >
  KotlinU3ConceptosBasicos
    src
      ClasesObjetos.kt
      Condicionales.kt
      FuncionesExpresionesLamda.kt
      Main.kt
      Nuladidad.kt
    External Libraries
    Scratches and Consoles

Run FuncionesExpresionesLamdaKt x

"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=58297" -Dfile.encoding=UTF-8 -Dsun.java2d.c
Process finished with exit code 0
```

Project

KotlinU3ConceptosBasicos

idea

out

src

ClasesObjetos.kt

Condicionales.kt

FuncionesExpresionesLamda.kt

Main.kt

Nuladidad.kt

gitignore

KotlinU3ConceptosBasicos.iml

External Libraries

Scratches and Consoles

Main.kt

Condicionales.kt

Nuladidad.kt

ClasesObjetos.kt

FuncionesExpresionesLamda.kt

```
6 fun trick() {
7     println("No treats!")
8 }
9
10 */
11
12
13 fun main() { new *
14     val trickFunction = trick
15     trick()
16 }
17
18 val trick = {
19     println("No treats!")
20 }
```

Run

FuncionesExpresionesLamdaKt

Run

Debug

Stop

Exit

C:\Program Files\Java\jdk-24\bin\java.exe

"-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=58587" -Dfile.encoding=UTF-8 -Dsun

No treats!

Project

KotlinU3ConceptosBasicos

idea

out

src

ClasesObjetos.kt

Condicionales.kt

FuncionesExpresionesLamda.kt

Main.kt

Nuladidad.kt

gitignore

KotlinU3ConceptosBasicos.iml

External Libraries

Scratches and Consoles

Main.kt

Condicionales.kt

Nuladidad.kt

ClasesObjetos.kt

FuncionesExpresionesLamda.kt

```
17
18 val trick = {
19     println("No treats!")
20 }
21
22 */
23
24 fun main() { Abraham Ramirez
25     val trickFunction = trick
26     trick()
27     trickFunction()
28 }
29
30 val trick = {
31     println("No treats!")
32 }
```

Run

FuncionesExpresionesLamdaKt

Run

Debug

Stop

Exit

C:\Program Files\Java\jdk-24\bin\java.exe

"-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=58666" -Dfile.encoding=UTF-8 -Dsun

No treats!

No treats!

Project

KotlinU3ConceptosBasicos

idea

out

src

ClasesObjetos.kt

Condicionales.kt

FuncionesExpresionesLamda.kt

Main.kt

Nuladidad.kt

gitignore

KotlinU3ConceptosBasicos.iml

External Libraries

Scratches and Consoles

Main.kt

Condicionales.kt

Nuladidad.kt

ClasesObjetos.kt

FuncionesExpresionesLamda.kt

```
33 }
34 */
35
36
37 val trick = {
38     println("No treats!")
39 }
40
41
42 val treat: () -> Unit = {
43     println("Have a treat!")
44 }
45
46 fun main() { new *
47     val trickFunction = trick
48     trick()
49     trickFunction()
50     treat()
51 }
```

Run

FuncionesExpresionesLamdaKt

Run

Debug

Stop

Exit

C:\Program Files\Java\jdk-24\bin\java.exe

"-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=58898" -Dfile.encoding=UTF-8 -Dsun

No treats!

No treats!

Have a treat!



```
Project
└─ KotlinU3ConceptosBasicos
   └─ src
      ├── ClasesObjetos.kt
      ├── Condicionales.kt
      └── FuncionesExpresionesLamda.kt
         ├── Main.kt
         └── Nuladidad.kt
      └─ .gitignore
      └─ KotlinU3ConceptosBasicos.iml
      └─ External Libraries
      └─ Scratches and Consoles

Run
  FuncionesExpresionesLamda.kt

  "C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=59157" -Dfile.encoding=UTF-8 -Dsun
  Have a treat!
  No treats!
```

```
Project
└─ KotlinU3ConceptosBasicos
   └─ src
      ├── ClasesObjetos.kt
      ├── Condicionales.kt
      └── FuncionesExpresionesLamda.kt
         ├── Main.kt
         └── Nuladidad.kt
      └─ .gitignore
      └─ KotlinU3ConceptosBasicos.iml
      └─ External Libraries
      └─ Scratches and Consoles

Run
  FuncionesExpresionesLamda.kt

  "C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=60333" -Dfile.encoding=UTF-8 -Dsun
  5 quarters
  Have a treat!
  No treats!
```

```
Project
└─ KotlinU3ConceptosBasicos
   └─ src
      ├── ClasesObjetos.kt
      ├── Condicionales.kt
      └── FuncionesExpresionesLamda.kt
         ├── Main.kt
         └── Nuladidad.kt
      └─ .gitignore
      └─ KotlinU3ConceptosBasicos.iml
      └─ External Libraries
      └─ Scratches and Consoles

Run
  FuncionesExpresionesLamda.kt

  "C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=60873" -Dfile.encoding=UTF-8
  5 quarters
  Have a treat!
  Have a treat!
  Have a treat!
  Have a treat!
  No treats!
```



```
Project
└─ KotlinU3ConceptosBasicos
   └─ src
      ├── ClasesObjetos.kt
      ├── Condicionales.kt
      ├── FuncionesExpresionesLamda.kt
      ├── Main.kt
      ├── Nuladidad.kt
      └── Practicas.kt
  .gitignore
  KotlinU3ConceptosBasicos.iml
  External Libraries
  Scratches and Consoles

Run PracticasKt
C:\Program Files\Java\jdk-24\bin\java.exe -javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=61257 -Dfile.encoding=UTF-8 -Dsun
You have 51 notifications.
Your phone is blowing up! You have 99+ notifications.
```

```
Project
└─ KotlinU3ConceptosBasicos
   └─ src
      ├── ClasesObjetos.kt
      ├── Condicionales.kt
      ├── FuncionesExpresionesLamda.kt
      ├── Main.kt
      ├── Nuladidad.kt
      └── Practicas.kt
  .gitignore
  KotlinU3ConceptosBasicos.iml
  External Libraries
  Scratches and Consoles

Run PracticasKt
C:\Program Files\Java\jdk-24\bin\java.exe -javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=61257 -Dfile.encoding=UTF-8 -Dsun
You have 51 notifications.
Your phone is blowing up! You have 99+ notifications.
```

```
Project
└─ KotlinU3ConceptosBasicos
   └─ src
      ├── ClasesObjetos.kt
      ├── Condicionales.kt
      ├── FuncionesExpresionesLamda.kt
      ├── Main.kt
      ├── Nuladidad.kt
      └── Practicas.kt
  .gitignore
  KotlinU3ConceptosBasicos.iml
  External Libraries
  Scratches and Consoles

Run PracticasKt
C:\Program Files\Java\jdk-24\bin\java.exe -javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=61343 -Dfile.encoding=UTF-8 -Dsun
27.0 degrees Celsius is 80.60 degrees Fahrenheit.
350.0 degrees Kelvin is 76.85 degrees Celsius.
10.0 degrees Fahrenheit is 260.93 degrees Kelvin.
```




```
Project
└─ KotlinU3ConceptosBasicos
   └─ src
      ├── ClasesObjetos.kt
      ├── Condicionales.kt
      ├── FuncionesExpresionesLamda.kt
      ├── Main.kt
      ├── Nuladidad.kt
      └── Practicas.kt
  └─ External Libraries
     └─ Scratches and Consoles

Run
PracticasKt

"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=61380" -Dfile.encoding=UTF-8 -Dsun...
We Don't Talk About Bruno, performed by Encanto Cast, was released in 2022.
true
```

```
Project
└─ KotlinU3ConceptosBasicos
   └─ src
      ├── ClasesObjetos.kt
      ├── Condicionales.kt
      ├── FuncionesExpresionesLamda.kt
      ├── Main.kt
      ├── Nuladidad.kt
      └── Practicas.kt
  └─ External Libraries
     └─ Scratches and Consoles

Run
PracticasKt

"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=61540" -Dfile.encoding=UTF-8 -Dsun...
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.
```

```
Project
└─ KotlinU3ConceptosBasicos
   └─ src
      ├── ClasesObjetos.kt
      ├── Condicionales.kt
      ├── FuncionesExpresionesLamda.kt
      ├── Main.kt
      ├── Nuladidad.kt
      └── Practicas.kt
  └─ External Libraries
     └─ Scratches and Consoles

Run
PracticasKt

"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=50135" -Dfile.encoding=UTF-8 -Dsun...
The phone screen's light is off.
The phone screen's light is on.
```



```
158 newFoldablePhone.unfold()
159 newFoldablePhone.switchOn()
160 newFoldablePhone.checkPhoneScreenLight()
161 }
162 */
163
164
165 fun main() { new *
166     val winningBid = Bid( amount: 5000, bidder: "Private Collector")
167
168     println("Item A is sold at ${auctionPrice(winningBid, minimumPrice: 2000)}.")
169     println("Item B is sold at ${auctionPrice( Bid: null, minimumPrice: 3000)}.")
170 }
171
172 class Bid(val amount: Int, val bidder: String) new *
173
174 fun auctionPrice(bid: Bid?, minimumPrice: Int): Int { new *
175     return bid?.amount ?: minimumPrice
176 }
```

Run PracticasKt

"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=50201" -Dfile.encoding=UTF-8 -Dsun

Item A is sold at 5000.

Item B is sold at 3000.

Agrega un Boton a una app

<https://github.com/Zafirows/Programacion-nativa>

// Paquete principal de la app

package com.example.diceroller

// Importaciones necesarias para la actividad y UI con Jetpack Compose

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.Column

import androidx.compose.foundation.layout.fillMaxSize

import androidx.compose.foundation.layout.padding

import androidx.compose.foundation.layout.wrapContentSize

import androidx.compose.material3.Button

import androidx.compose.material3.MaterialTheme

import androidx.compose.material3.Scaffold



```
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.runtime.setValue
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.res.stringResource
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.sp
import com.example.diceroller.ui.theme.DiceRollerTheme

// Actividad principal de la aplicación
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        // Establece el contenido de la UI usando Jetpack Compose
        setContent {
            // Aplica el tema de la app
            DiceRollerTheme {
                // Surface actúa como contenedor base con fondo temático
                Surface(
                    modifier = Modifier.fillMaxSize(), // Ocupa toda la pantalla
```



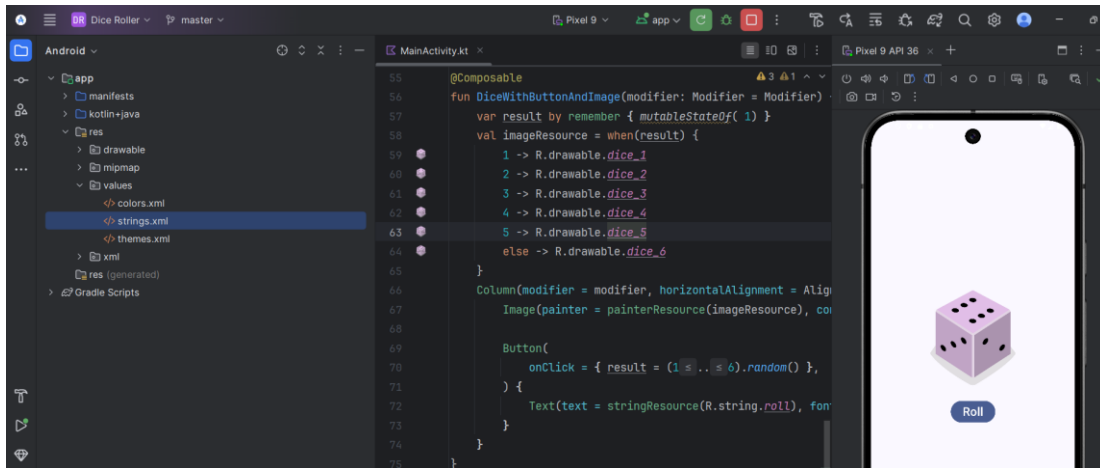
```
        color = MaterialTheme.colorScheme.background // Usa el color de  
fondo del tema
```

```
    ) {  
        // Llama al componente principal de la app  
        DiceRollerApp()  
    }  
}  
}  
}  
}  
}  
}  
// Anotación para mostrar vista previa en el editor de Android Studio  
@Preview  
@Composable  
fun DiceRollerApp() {  
    // Llama a la función composable que contiene la lógica de dado y botón  
    DiceWithButtonAndImage(  
        modifier = Modifier  
            .fillMaxSize() // Ocupa todo el tamaño del contenedor  
            .wrapContentSize(Alignment.Center) // Centra el contenido  
    )  
}  
// Función composable que muestra el dado y un botón para tirarlo  
@Composable  
fun DiceWithButtonAndImage(modifier: Modifier = Modifier) {  
    // Variable de estado para almacenar el resultado actual del dado  
    var result by remember { mutableStateOf(1) }  
    // Selecciona la imagen correspondiente al valor del dado
```



```
val imageResource = when(result) {  
    1 -> R.drawable.dice_1  
    2 -> R.drawable.dice_2  
    3 -> R.drawable.dice_3  
    4 -> R.drawable.dice_4  
    5 -> R.drawable.dice_5  
    else -> R.drawable.dice_6 // Para el caso de valor 6  
}  
  
// Organiza los elementos en una columna, centrados horizontalmente  
Column(modifier = modifier, horizontalAlignment =  
Alignment.CenterHorizontally) {  
    // Muestra la imagen del dado según el valor actual  
    Image(  
        painter = painterResource(imageResource), // Imagen del recurso  
        contentDescription = result.toString() // Descripción para accesibilidad  
    )  
  
    // Botón que, al hacer clic, genera un número aleatorio del 1 al 6  
    Button(  
        onClick = { result = (1..6).random() }, // Al hacer clic, cambia el valor del  
        dado  
    ) {  
        // Texto del botón (por ejemplo: "Roll")  
        Text(  
            text = stringResource(R.string.roll), // Usa el recurso de string definido  
            fontSize = 24.sp // Tamaño de fuente  
        )  
    }  
}
```

```
}
}
```



En pantalla se presenta una interfaz compuesta por la imagen de un dado mostrando inicialmente la cara correspondiente al número uno y un botón. Cada vez que el usuario presiona el botón, se genera un número aleatorio del 1 al 6, y la imagen del dado se actualiza para reflejar la cara correspondiente a dicho número.

La lógica de la aplicación está organizada en varios componentes. La clase MainActivity actúa como punto de entrada y establece la interfaz de usuario mediante setContent, aplicando el tema DiceRollerTheme. Dentro de esta configuración, se invoca la función DiceRollerApp, la cual a su vez llama a la función composable DiceWithButtonAndImage.

En DiceWithButtonAndImage se define una variable de estado denominada result, que mantiene el valor actual del dado. A partir de este valor, y mediante una estructura when, se selecciona dinámicamente el recurso gráfico correspondiente (por ejemplo, dice 1, dice 2, etc.). La interfaz se construye utilizando una columna centrada, donde se muestran tanto la imagen del dado como el botón. Al presionar el botón, se actualiza el estado de result con un nuevo número aleatorio entre 1 y 6, lo que desencadena una recomposición del componente y actualiza automáticamente la imagen mostrada.

Limonada

<https://github.com/Zafirows/Programacion-nativa>

```
// Paquete principal de la app
package com.example.lemonade
```



```
// Importaciones necesarias para el funcionamiento de la app con Jetpack
Compose
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.background
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.res.dimensionResource
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.res.stringResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.tooling.preview.Preview
import com.example.lemonade.ui.theme.LemonadeTheme

// Actividad principal de la aplicación
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        enableEdgeToEdge() // Habilita contenido detrás de barras del sistema
        super.onCreate(savedInstanceState)
        // Establece el contenido de la UI con Jetpack Compose
        setContent {
            LemonadeTheme {
                LemonadeApp() // Llama a la función principal de la UI
            }
        }
    }
}

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun LemonadeApp() {

    // Estado que controla el paso actual del flujo de la limonada
```



```
var currentStep by remember { mutableStateOf(1) }

// Contador de exprimidas necesarias (entre 2 y 4 aleatorio)
var squeezeCount by remember { mutableStateOf(0) }

// Scaffold para tener una barra superior y contenido principal
Scaffold(
  topBar = {
    CenterAlignedTopAppBar(
      title = {
        Text(
          text = "Lemonade",
          fontWeight = FontWeight.Bold
        )
      },
      colors = TopAppBarDefaults.largeTopAppBarColors(
        containerColor = MaterialTheme.colorScheme.primaryContainer
      )
    )
  }
) { innerPadding ->
  // Contenedor principal con Surface
  Surface(
    modifier = Modifier
      .fillMaxSize()
      .padding(innerPadding)
      .background(MaterialTheme.colorScheme.tertiaryContainer),
    color = MaterialTheme.colorScheme.background
  ) {
    // Flujo del proceso de hacer limonada
    when (currentStep) {
      1 -> {
        // Paso 1: seleccionar un limón del árbol
        LemonTextAndImage(
          textLabelResourceId = R.string.lemon_select,
          drawableResourceId = R.drawable.lemon_tree,
          contentDescriptionResourceId =
R.string.lemon_tree_content_description,
          onClick = {
            currentStep = 2 // Avanza al paso 2
          }
        )
      }
    }
  }
}
```



```
squeezeCount = (2..4).random() // Define cuántas veces hay que
exprimir
    }
  )
}
2 -> {
  // Paso 2: exprimir el limón
  LemonTextAndImage(
    textLabelResourceId = R.string.lemon_squeeze,
    drawableResourceId = R.drawable.lemon_squeeze,
    contentDescriptionResourceId =
R.string.lemon_content_description,
    onClick = {
      squeezeCount-- // Resta una exprimida
      if (squeezeCount == 0) {
        currentStep = 3 // Cuando se acaba, pasa al paso 3
      }
    }
  )
}
3 -> {
  // Paso 3: beber la limonada
  LemonTextAndImage(
    textLabelResourceId = R.string.lemon_drink,
    drawableResourceId = R.drawable.lemon_drink,
    contentDescriptionResourceId =
R.string.lemonade_content_description,
    onClick = {
      currentStep = 4 // Avanza al paso 4
    }
  )
}
4 -> {
  // Paso 4: reiniciar el proceso
  LemonTextAndImage(
    textLabelResourceId = R.string.lemon_empty_glass,
    drawableResourceId = R.drawable.lemon_restart,
    contentDescriptionResourceId =
R.string.empty_glass_content_description,
    onClick = {
```



```
        currentStep = 1 // Vuelve al paso 1
    }
}
)
}
}
}
}
}

// Función que muestra el texto y la imagen (con botón) en cada paso
@Composable
fun LemonTextAndImage(
    textLabelResourceId: Int, // ID del texto a mostrar
    drawableResourceId: Int, // ID de la imagen a mostrar
    contentDescriptionResourceId: Int, // ID de la descripción para accesibilidad
    onImageClick: () -> Unit, // Acción a ejecutar al pulsar la imagen
    modifier: Modifier = Modifier
) {
    Box(
        modifier = modifier
    ) {
        Column(
            horizontalAlignment = Alignment.CenterHorizontally,
            verticalArrangement = Arrangement.Center,
            modifier = Modifier.fillMaxSize()
        ) {
            // Botón con imagen dentro
            Button(
                onClick = onImageClick,
                shape =
                    RoundedCornerShape(dimensionResource(R.dimen.button_corner_radius)),
                colors = ButtonDefaults.buttonColors(containerColor =
                    MaterialTheme.colorScheme.tertiaryContainer)
            ) {
                Image(
                    painter = painterResource(drawableResourceId),
                    contentDescription = stringResource(contentDescriptionResourceId),
                    modifier = Modifier
                        .width(dimensionResource(R.dimen.button_image_width))
                        .height(dimensionResource(R.dimen.button_image_height))
                )
            }
        }
    }
}
```




```
        .padding(dimensionResource(R.dimen.button_interior_padding))
    )
}

// Espaciado entre imagen y texto
Spacer(modifier =
Modifier.height(dimensionResource(R.dimen.padding_vertical)))

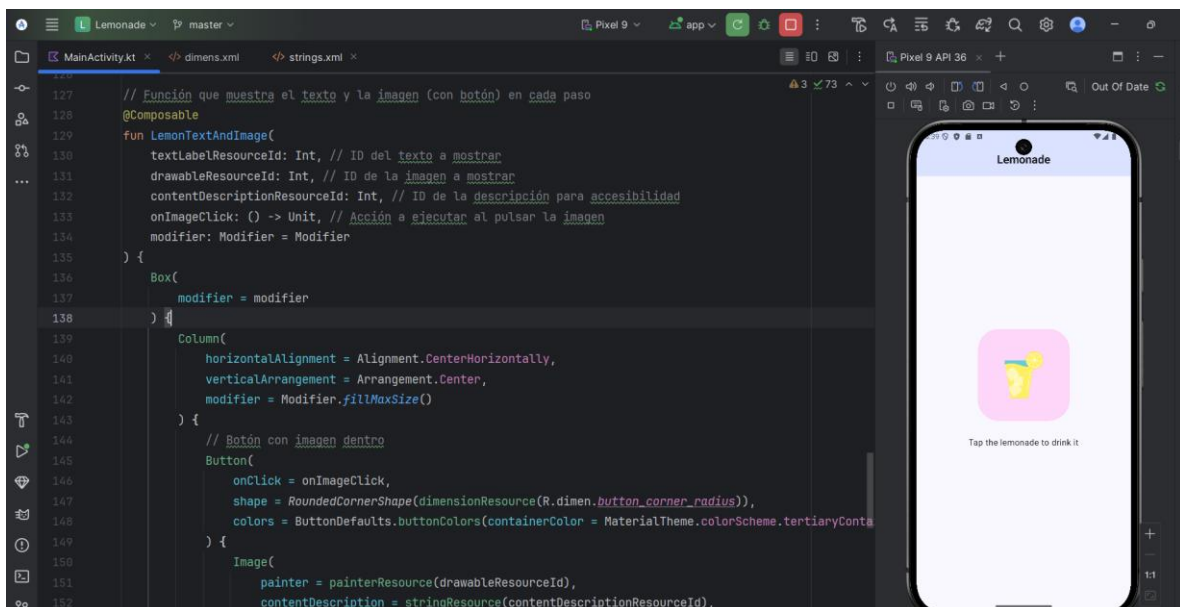
// Texto descriptivo debajo de la imagen
Text(
    text = stringResource(textLabelResourceId),
    style = MaterialTheme.typography.bodyLarge
)
}
}
}
```

// Vista previa de la app en el editor de Android Studio

@Preview

@Composable

```
fun LemonPreview() {
    LemonadeTheme(){
        LemonadeApp()
    }
}
```





Lemonade, desarrollada con **Jetpack Compose**. La aplicación simula de forma interactiva el proceso de preparar una limonada a través de una secuencia de cuatro etapas.

Al iniciar la aplicación, se presenta una imagen de un árbol de limones. Al tocarla, el usuario "selecciona" un limón, lo que da paso al segundo estado. En esta fase, se debe "exprimir" el limón realizando entre 2 y 4 toques consecutivos, cantidad determinada aleatoriamente. Una vez completado este número de interacciones, la aplicación transita al tercer estado, en el que se muestra un vaso lleno de limonada. Al tocar el vaso, se avanza al cuarto y último estado, que muestra un vaso vacío, indicando que la limonada ha sido consumida. Al tocar nuevamente el vaso vacío, el ciclo se reinicia, regresando al árbol de limones.

La interfaz está construida utilizando componentes de Jetpack Compose como Scaffold, Column, Image y Button. La gestión del flujo de la aplicación se realiza mediante el uso de variables de estado declaradas con remember y mutableStateOf, lo que permite recomposiciones dinámicas en función de las interacciones del usuario. Además, la UI se adapta al diseño de **Material 3**, e incluye una barra superior con el título "Lemonade", así como recursos visuales y de texto definidos en los archivos de recursos del proyecto.

Interactúa con la IU y el estado

<https://github.com/Zafirows/Programacion-nativa>

/*

* Copyright (C) 2023 The Android Open Source Project
package com.example.tiptime

```
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.enableEdgeToEdge
import androidx.compose.foundation.layout.Arrangement
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.layout.safeDrawingPadding
import androidx.compose.foundation.layout.statusBarsPadding
```



```
import androidx.compose.foundation.rememberScrollState
import androidx.compose.foundation.text.KeyboardOptions
import androidx.compose.foundation.verticalScroll
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.material3.TextField
import androidx.compose.runtime.Composable
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.runtime.setValue
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.res.stringResource
import androidx.compose.ui.text.input.KeyboardType
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import com.example.tiptime.ui.theme.TipTimeTheme
import java.text.NumberFormat

// Clase principal que representa una actividad de la app
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        // Habilita el diseño de borde a borde (sin márgenes del sistema)
        enableEdgeToEdge()
        super.onCreate(savedInstanceState)

        // Establece el contenido de la UI usando Jetpack Compose
        setContent {
            // Aplica el tema de la app
            TipTimeTheme {
                // Superficie que ocupa toda la pantalla
                Surface(
                    modifier = Modifier.fillMaxSize(),
                ) {
                    // Llama a la función que define el diseño de la pantalla principal
                    TipTimeLayout()
                }
            }
        }
    }
}
```



```
}  
}  
}
```

@Composable

```
fun TipTimeLayout() {
```

```
    // Variable de estado para guardar la entrada del usuario como texto
```

```
    var amountInput by remember { mutableStateOf("") }  
  
    // Convierte la entrada a Double si es posible, si no, usa 0.0 como valor por defecto
```

```
    val amount = amountInput.toDoubleOrNull() ?: 0.0  
  
    // Calcula la propina usando la función calculateTip
```

```
    val tip = calculateTip(amount)  
  
    // Contenedor en columna para organizar los elementos verticalmente
```

```
    Column(  
        modifier = Modifier  
            .statusBarsPadding() // Añade espacio para la barra de estado  
            .padding(horizontal = 40.dp) // Padding horizontal de 40dp  
            .verticalScroll(rememberScrollState()) // Habilita scroll si el contenido es largo  
        ) {  
        .safeDrawingPadding(), // Padding seguro para áreas del sistema  
        horizontalAlignment = Alignment.CenterHorizontally, // Centrado horizontal  
        verticalArrangement = Arrangement.Center // Centrado vertical  
    } {  
        // Texto de encabezado: "Calculate Tip"
```

```
        Text(  
            text = stringResource(R.string.calculate_tip),  
            modifier = Modifier  
                .padding(bottom = 16.dp, top = 40.dp) // Espaciado superior e inferior  
                .align(alignment = Alignment.Start) // Alineación a la izquierda  
        )  
  
        // Campo de texto para introducir el monto de la cuenta
```

```
        EditNumberField(  
            value = amountInput, // Valor actual del texto  
            onChange = { amountInput = it }, // Actualiza la variable de estado  
            modifier = Modifier
```

```
        )  
    }  
}
```

```
    ) {  
        // Texto de encabezado: "Calculate Tip"
```

```
        Text(  
            text = stringResource(R.string.calculate_tip),  
            modifier = Modifier  
                .padding(bottom = 16.dp, top = 40.dp) // Espaciado superior e inferior  
                .align(alignment = Alignment.Start) // Alineación a la izquierda  
        )  
  
        // Campo de texto para introducir el monto de la cuenta
```

```
        EditNumberField(  
            value = amountInput, // Valor actual del texto  
            onChange = { amountInput = it }, // Actualiza la variable de estado  
            modifier = Modifier
```

```
        )  
    }  
}
```

```
    ) {  
        // Texto de encabezado: "Calculate Tip"
```

```
        Text(  
            text = stringResource(R.string.calculate_tip),  
            modifier = Modifier  
                .padding(bottom = 16.dp, top = 40.dp) // Espaciado superior e inferior  
                .align(alignment = Alignment.Start) // Alineación a la izquierda  
        )  
  
        // Campo de texto para introducir el monto de la cuenta
```

```
        EditNumberField(  
            value = amountInput, // Valor actual del texto  
            onChange = { amountInput = it }, // Actualiza la variable de estado  
            modifier = Modifier
```

```
        )  
    }  
}
```

```
        // Texto de encabezado: "Calculate Tip"
```

```
        Text(  
            text = stringResource(R.string.calculate_tip),  
            modifier = Modifier  
                .padding(bottom = 16.dp, top = 40.dp) // Espaciado superior e inferior  
                .align(alignment = Alignment.Start) // Alineación a la izquierda  
        )  
  
        // Campo de texto para introducir el monto de la cuenta
```

```
        EditNumberField(  
            value = amountInput, // Valor actual del texto  
            onChange = { amountInput = it }, // Actualiza la variable de estado  
            modifier = Modifier
```

```
        )  
    }  
}
```

```
        .padding(bottom = 16.dp, top = 40.dp) // Espaciado superior e inferior  
        .align(alignment = Alignment.Start) // Alineación a la izquierda  
    )  
  
    // Campo de texto para introducir el monto de la cuenta
```

```
    EditNumberField(  
        value = amountInput, // Valor actual del texto  
        onChange = { amountInput = it }, // Actualiza la variable de estado  
        modifier = Modifier
```

```
    )  
}
```

```
    // Campo de texto para introducir el monto de la cuenta
```

```
    EditNumberField(  
        value = amountInput, // Valor actual del texto  
        onChange = { amountInput = it }, // Actualiza la variable de estado  
        modifier = Modifier
```

```
        value = amountInput, // Valor actual del texto
```

```
        onChange = { amountInput = it }, // Actualiza la variable de estado
```

```
        modifier = Modifier
```



```
.padding(bottom = 32.dp) // Espaciado inferior
.fillMaxWidth() // Ocupa todo el ancho disponible
)

// Muestra el resultado de la propina calculada
Text(
    text = stringResource(R.string.tip_amount, tip),
    style = MaterialTheme.typography.displaySmall // Aplica estilo de texto del
tema
)

// Espaciador para dar separación inferior
Spacer(modifier = Modifier.height(150.dp))
}
}

@Composable
fun EditNumberField(
    value: String, // Valor actual del campo de texto
    onValueChange: (String) -> Unit, // Función que se llama cuando cambia el
texto
    modifier: Modifier = Modifier // Modificador opcional
) {
    // Campo de texto para ingresar números
    TextField(
        value = value, // Muestra el texto actual
        onValueChange = onValueChange, // Actualiza el texto cuando cambia
        singleLine = true, // Limita a una sola línea
        label = { Text(stringResource(R.string.bill_amount)) }, // Etiqueta del campo:
"Bill Amount"
        keyboardOptions = KeyboardOptions(keyboardType =
KeyboardType.Number), // Teclado numérico
        modifier = modifier // Aplica los modificadores recibidos
    )
}

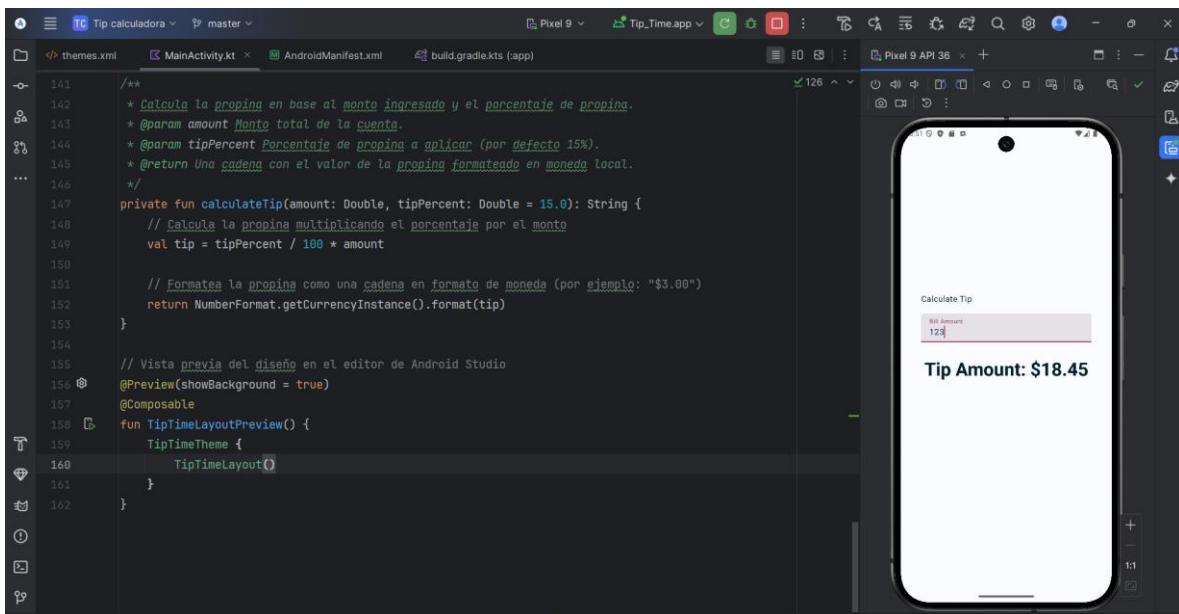
/**
 * Calcula la propina en base al monto ingresado y el porcentaje de propina.
 * @param amount Monto total de la cuenta.
 * @param tipPercent Porcentaje de propina a aplicar (por defecto 15%).
```



```
* @return Una cadena con el valor de la propina formateado en moneda local.
*/
private fun calculateTip(amount: Double, tipPercent: Double = 15.0): String {
    // Calcula la propina multiplicando el porcentaje por el monto
    val tip = tipPercent / 100 * amount

    // Formatea la propina como una cadena en formato de moneda (por ejemplo:
    "$3.00")
    return NumberFormat.getCurrencyInstance().format(tip)
}

// Vista previa del diseño en el editor de Android Studio
@Preview(showBackground = true)
@Composable
fun TipTimeLayoutPreview() {
    TipTimeTheme {
        TipTimeLayout()
    }
}
```



Este código implementa una aplicación Android llamada **TipTime**, desarrollada con Jetpack Compose, que permite a los usuarios calcular propinas fácilmente. Al iniciar la aplicación, se muestra una pantalla sencilla con un campo de texto para introducir el monto de una cuenta y, automáticamente, se calcula y muestra la propina correspondiente, que por



defecto es del 15%. La aplicación convierte la entrada del usuario en un número (Double) y utiliza una función auxiliar llamada `calculateTip` para obtener el valor de la propina, que luego se muestra en formato de moneda local. El diseño de la interfaz está organizado en una columna centrada vertical y horizontalmente, y se adapta a las zonas seguras de la pantalla como la barra de estado. También se implementa desplazamiento vertical por si el contenido excede el tamaño de la pantalla. La interfaz está estilizada con Material 3, utilizando temas, tipografías y componentes como `TextField`, `Text`, `Spacer` y `Surface`. En resumen, el código define una app funcional y minimalista para calcular propinas, demostrando el uso práctico de estados, entrada de texto y diseño adaptable con Jetpack Compose.

Tip calculator personalizado

<https://github.com/Zafirows/Programacion-nativa>

/*

```
* Copyright (C) 2023 The Android Open Source Project
package com.example.tiptime
import android.os.Bundle
import androidx.compose.material3.Switch
import androidx.compose.foundation.layout.wrapContentSize
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.size
import androidx.compose.ui.text.input.ImeAction
import androidx.annotation.StringRes
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.enableEdgeToEdge
import androidx.annotation.DrawableRes
import androidx.compose.material3.Icon
import androidx.compose.foundation.layout.Arrangement
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.layout.safeDrawingPadding
import androidx.compose.foundation.layout.statusBarsPadding
import androidx.compose.foundation.rememberScrollState
import androidx.compose.foundation.text.KeyboardOptions
import androidx.compose.foundation.verticalScroll
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.material3.TextField
import androidx.compose.runtime.Composable
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.runtime.setValue
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.res.stringResource
import androidx.compose.ui.text.input.KeyboardType
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import com.example.tiptime.ui.theme.TipTimeTheme
import java.text.NumberFormat

// Clase principal de la app, extiende de ComponentActivity para usar Jetpack
Compose
class MainActivity : ComponentActivity() {
    // Método que se ejecuta al crear la actividad
    override fun onCreate(savedInstanceState: Bundle?) {
        enableEdgeToEdge() // Habilita diseño de pantalla completa (de borde a
borde)
        super.onCreate(savedInstanceState)
        setContent {
            TipTimeTheme { // Aplica el tema de la app
                Surface(
                    modifier = Modifier.fillMaxSize(), // Hace que ocupe todo el tamaño
disponible
                ) {
                    TipTimeLayout() // Muestra el contenido principal de la app
                }
            }
        }
    }
}
```




```
    }  
  }  
}
```

// Función composable que define el layout de la calculadora de propinas

@Composable

fun TipTimeLayout() {

var amountInput by remember { mutableStateOf("") } // Estado para el monto
 ingresado

var tipInput by remember { mutableStateOf("") } // Estado para el porcentaje de
 propina

var roundUp by remember { mutableStateOf(false) } // Estado para saber si
 redondear la propina

// Convierte los textos ingresados a números (Double), usa 0.0 si el valor es
 inválido

val amount = amountInput.toDoubleOrNull() ?: 0.0

val tipPercent = tipInput.toDoubleOrNull() ?: 0.0

// Calcula la propina

val tip = calculateTip(amount, tipPercent, roundUp)

// Columna principal con padding y scroll

Column(

modifier = Modifier

.padding(40.dp)

.verticalScroll(rememberScrollState()),

horizontalAlignment = Alignment.CenterHorizontally,

verticalArrangement = Arrangement.Center

) {

// Título "Calculate Tip"

Text(

text = stringResource(R.string.calculate_tip),

modifier = Modifier

.padding(bottom = 16.dp)

.align(alignment = Alignment.Start)

)

// Campo para ingresar el monto de la cuenta

EditNumberField(

label = R.string.bill_amount, // Etiqueta



```
leadingIcon = R.drawable.money, // Ícono de dinero
keyboardOptions = KeyboardOptions.Default.copy(
    keyboardType = KeyboardType.Number, // Solo números
    imeAction = ImeAction.Next // Acción "Siguiente" en el teclado
),
value = amountInput, // Valor actual del input
onValueChanged = { amountInput = it }, // Actualiza el estado al cambiar
modifier = Modifier
    .padding(bottom = 32.dp)
    .fillMaxWidth()
)

// Campo para ingresar el porcentaje de propina
EditNumberField(
    label = R.string.how_was_the_service, // Etiqueta
    leadingIcon = R.drawable.percent, // Ícono de porcentaje
    keyboardOptions = KeyboardOptions.Default.copy(
        keyboardType = KeyboardType.Number,
        imeAction = ImeAction.Done // Acción "Hecho" en el teclado
    ),
    value = tipInput,
    onValueChanged = { tipInput = it },
    modifier = Modifier
        .padding(bottom = 32.dp)
        .fillMaxWidth()
)

// Fila con switch para redondear la propina
RoundTheTipRow(
    roundUp = roundUp, // Estado actual
    onRoundUpChanged = { roundUp = it }, // Se actualiza al cambiar el switch
    modifier = Modifier.padding(bottom = 32.dp)
)

// Texto que muestra la propina calculada
Text(
    text = stringResource(R.string.tip_amount, tip),
    style = MaterialTheme.typography.displaySmall
)
```



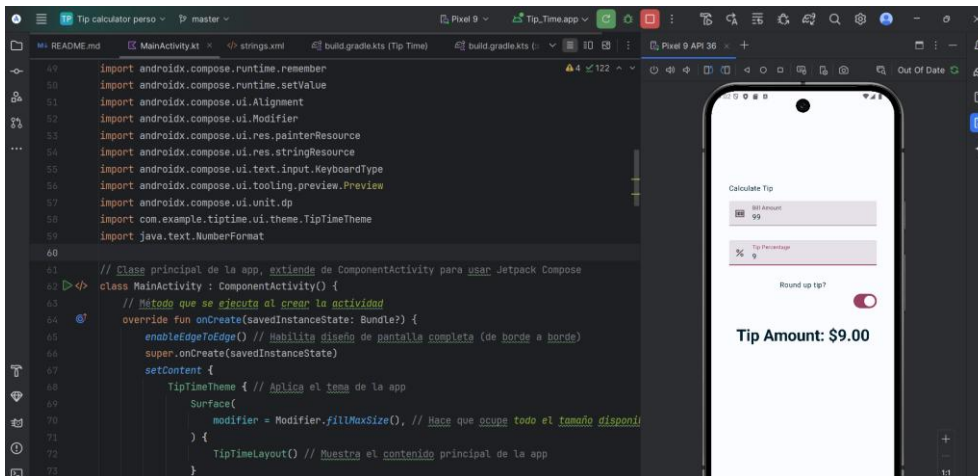
```
// Espaciado extra al final
Spacer(modifier = Modifier.height(150.dp))
}
}

// Composable para un campo de texto con ícono y etiqueta
@Composable
fun EditNumberField(
    @StringRes label: Int, // Etiqueta del campo (recurso de string)
    @DrawableRes leadingIcon: Int, // Ícono al inicio del campo
    keyboardOptions: KeyboardOptions, // Configuración del teclado
    value: String, // Valor actual del campo
    onValueChanged: (String) -> Unit, // Callback cuando cambia el valor
    modifier: Modifier = Modifier
) {
    TextField(
        value = value,
        leadingIcon = { Icon(painter = painterResource(id = leadingIcon), null) },
        singleLine = true,
        modifier = modifier,
        onValueChange = onValueChanged,
        label = { Text(stringResource(label)) },
        keyboardOptions = keyboardOptions
    )
}

// Composable para mostrar la opción de redondear la propina con un switch
@Composable
fun RoundTheTipRow(
    roundUp: Boolean, // Estado actual del switch
    onRoundUpChanged: (Boolean) -> Unit, // Callback cuando se cambia el switch
    modifier: Modifier = Modifier
) {
    Text(text = stringResource(R.string.round_up_tip)) // Texto descriptivo
    Switch(
        modifier = modifier
            .fillMaxWidth()
            .wrapContentWidth(Alignment.End), // Alinea a la derecha
        checked = roundUp, // Valor actual del switch
        onCheckedChange = onRoundUpChanged, // Se actualiza al cambiar
    )
}
```



```
}  
// Función que calcula y formatea la propina según el monto y porcentaje  
ingresado  
private fun calculateTip(amount: Double, tipPercent: Double = 15.0, roundUp:  
Boolean): String {  
    var tip = tipPercent / 100 * amount // Calcula la propina base  
    if (roundUp) {  
        tip = kotlin.math.ceil(tip) // Redondea hacia arriba si es necesario  
    }  
    return NumberFormat.getCurrencyInstance().format(tip) // Devuelve la propina  
formateada  
}  
// Vista previa de TipTimeLayout en modo diseño  
@Preview(showBackground = true)  
@Composable  
fun TipTimeLayoutPreview() {  
    TipTimeTheme {  
        TipTimeLayout() // Muestra la vista previa con el tema aplicado  
    }  
}
```



Este código implementa una aplicación Android denominada **TipTime**, desarrollada utilizando **Jetpack Compose**, cuyo objetivo es permitir a los usuarios calcular propinas de manera flexible y precisa. La aplicación permite introducir tanto el monto total de una cuenta como un porcentaje personalizado de propina, además de ofrecer una opción para redondear el valor calculado hacia arriba.

Al iniciar la aplicación, se presenta una interfaz intuitiva compuesta por dos campos de entrada: uno para el importe de la cuenta y otro para el porcentaje de



propina, cada uno acompañado de íconos representativos que mejoran la experiencia de usuario. Adicionalmente, se incluye un componente Switch que permite activar o desactivar la opción de redondeo. El cálculo de la propina se realiza de forma automática mediante la función `calculateTip`, la cual también se encarga de formatear el resultado como una cantidad monetaria en la moneda local del dispositivo.

La interfaz está diseñada con un enfoque minimalista y responsivo, estructurada en un diseño vertical desplazable que garantiza su adaptabilidad en distintas resoluciones de pantalla. Se utilizan componentes de **Material 3**, tales como `TextField`, `Text`, `Switch` y `Surface`, junto con el sistema de tematización de Jetpack Compose, lo que asegura una estética moderna, coherente y en línea con las directrices actuales de diseño de interfaces en Android.