



**TECNOLÓGICO  
NACIONAL DE MÉXICO**



**INSTITUTO TECNOLÓGICO  
DE CIUDAD MADERO**  
www.itcm.edu.mx

# **Instituto Tecnológico de Ciudad Madero**

## **INGENIERÍA EN SISTEMAS COMPUTACIONALES**

**EQUIPO**

**Flores Ramírez Manuel Abraham**

**Ian samuel García Pérez**

**No. De control**

**20070511**

**20070611**

**MATERIA**

**Programación nativa para móviles**

**TAREA**

**Tarea No 9 Unidad 3 Como mostrar listas y usar Material  
Desing**

**Semestre**

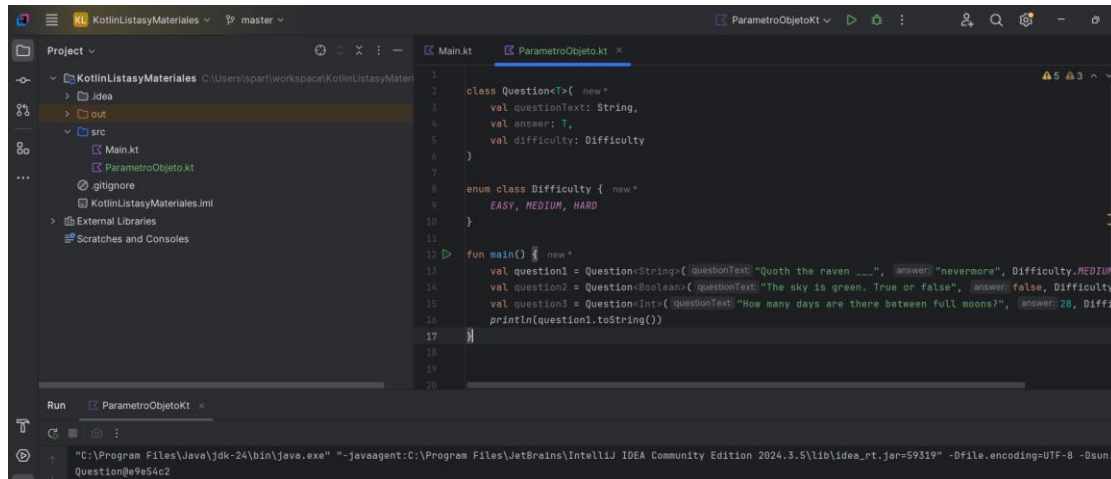
**Decimo**

**Maestro**

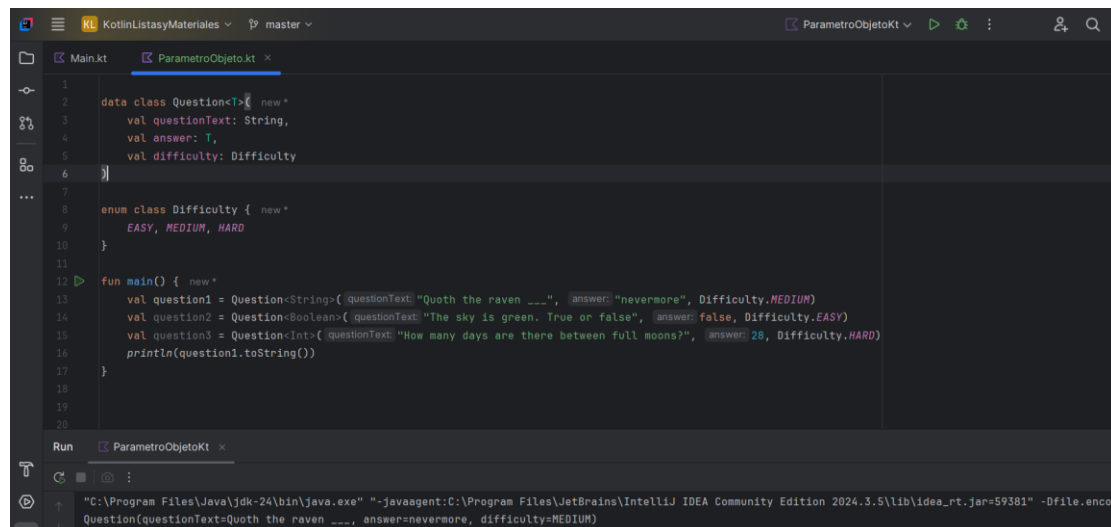
**Jorge Peralta Escobar**

# Mas aspectos básicos de kotlin

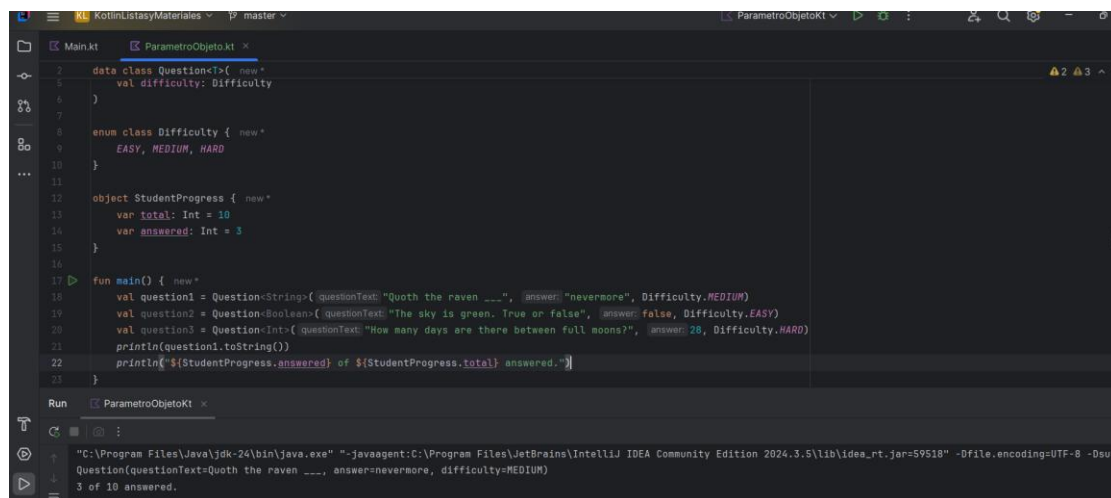
<https://github.com/Abraham20070511/KotlinListasyMateriales>



```
1 class Question<T> { new *
2     val questionText: String,
3     val answer: T,
4     val difficulty: Difficulty
5 }
6
7 enum class Difficulty { new *
8     EASY, MEDIUM, HARD
9 }
10
11
12 fun main() { new *
13     val question1 = Question<String>("Quoth the raven ____", "answer: nevermore", Difficulty.MEDIUM)
14     val question2 = Question<Boolean>("The sky is green. True or false?", "answer: false", Difficulty.EASY)
15     val question3 = Question<Int>("How many days are there between full moons?", "answer: 28", Difficulty.HARD)
16     println(question1.toString())
17 }
18
19
20
```



```
1
2 data class Question<T> { new *
3     val questionText: String,
4     val answer: T,
5     val difficulty: Difficulty
6 }
7
8 enum class Difficulty { new *
9     EASY, MEDIUM, HARD
10 }
11
12 fun main() { new *
13     val question1 = Question<String>("Quoth the raven ____", "answer: nevermore", Difficulty.MEDIUM)
14     val question2 = Question<Boolean>("The sky is green. True or false?", "answer: false", Difficulty.EASY)
15     val question3 = Question<Int>("How many days are there between full moons?", "answer: 28", Difficulty.HARD)
16     println(question1.toString())
17 }
18
19
20
```



```
1
2 data class Question<T> { new *
3     val difficulty: Difficulty
4 }
5
6
7 enum class Difficulty { new *
8     EASY, MEDIUM, HARD
9 }
10
11 object StudentProgress { new *
12     var total: Int = 10
13     var answered: Int = 3
14 }
15
16 fun main() { new *
17     val question1 = Question<String>("Quoth the raven ____", "answer: nevermore", Difficulty.MEDIUM)
18     val question2 = Question<Boolean>("The sky is green. True or false?", "answer: false", Difficulty.EASY)
19     val question3 = Question<Int>("How many days are there between full moons?", "answer: 28", Difficulty.HARD)
20     println(question1.toString())
21     println("${StudentProgress.answered} of ${StudentProgress.total} answered.")
22 }
23
```

The screenshot shows the IntelliJ IDEA code editor with a Kotlin file named `ParametroObjetoKt.kt`. The code defines a `Quiz` class with a companion object `StudentProgress` and a `main` function. The `main` function creates three questions and prints the results.

```

12 class Quiz { new *
13
14     companion object StudentProgress { new *
15         var total: Int = 10
16         var answered: Int = 3
17     }
18
19     fun main() { new *
20         val question1 = Question<String>(questionText: "Quoth the raven ___", answer: "nevermore", Difficulty.MEDIUM)
21         val question2 = Question<Boolean>(questionText: "The sky is green. True or false?", answer: false, Difficulty.EASY)
22         val question3 = Question<Int>(questionText: "How many days are there between full moons?", answer: 28, Difficulty.HARD)
23         println(question1.toString())
24         println("${Quiz.answered} of ${Quiz.total} answered.")
25     }
26 }

```

The Run console shows the output of 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=59642" -Dfile.encoding=UTF-8 -Dsun
Question(questionText=Quoth the raven ___, answer=nevermore, difficulty=MEDIUM)
3 of 10 answered.

```

```
11 class Quiz { new *
12 }
13 }
14
15 val Quiz.StudentProgress.progressText: String new *
16 get() = "${answered} of $total answered"
17
18 object StudentProgress { new *
19     var total: Int = 10
20     var answered: Int = 3
21 }
22
23 fun main() { new *
24     val question1 = Question<String>( questionText: "Quoth the raven __", answer: "nevermore", Difficulty.MEDIUM)
25     val question2 = Question<Boolean>( questionText: "The sky is green. True or false?", answer: false, Difficulty.EASY)
26     val question3 = Question<Int>( questionText: "How many days are there between full moons?", answer: 28, Difficulty.HARD)
27     println(question1.toString())
28     println(quiz.progressText)
29 }
30 }
```

The screenshot shows an IDE with two panes. The top pane displays the source code for a Kotlin application named 'Quiz'. The code defines a 'Quiz' class with a 'total' property (10) and an 'answered' property (3). It includes a 'StudentProgress' object with a 'progressText' property. The 'main' function prints the progress bar and the progress text.

```

12 class Quiz { new *
13     var answered: Int = 3
14 }
15
16 fun Quiz.StudentProgress.printProgressBar() { new *
17     repeat(Quiz.answered) { print("█") }
18     repeat(10 - Quiz.answered) { print("░") }
19     println()
20     println(Quiz.progressText)
21 }
22
23 val Quiz.StudentProgress.progressText: String new *
24     get() = "${answered} of $total answered"
25
26 object StudentProgress { new *
27     var total: Int = 10
28     var answered: Int = 3
29 }
30
31 fun main() {
32     Quiz().StudentProgress.printProgressBar()
33 }

```

The bottom pane shows the 'Run' output. It displays the command used to run the application and the resulting output. The output shows the progress bar and the progress text.

```

Run
"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=60755" -Dfile.encoding=UTF-8 -Dsun.
Question(questionText:Quoth the raven ____, answer=nevermore, difficulty=MEDIUM)
3 of 10 answered
██████████
3 of 10 answered

```

```
1 data class Question<T>() { new *
2     val questionText: String,
3     val answer: T,
4     val difficulty: Difficulty
5 }
6
7 enum class Difficulty { new *
8     EASY, MEDIUM, HARD
9 }
10
11 interface ProgressPrintable { new *
12     val progressText: String
13     fun printProgressBar() new *
14 }
15
16 class Quiz : ProgressPrintable { new *
17     val question1 = Question<String>(questionText = "Quoth the raven ___", answer = "nevermore", Difficulty.MEDIUM)
18     val question2 = Question<Boolean>(questionText = "The sky is green. True or false?", answer = false, Difficulty.EASY)
19 }
20
21 fun main() {
22     Quiz().printQuiz()
23 }
```

Run ParametroObjetoKt

"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=61042" -Dfile.encoding=UTF-8 -DmainClass=ParametroObjetoKt

Quoth the raven \_\_\_  
nevermore  
3 of 10 answered

```
17 class Quiz : ProgressPrintable { new *
18
19     companion object StudentProgress { new *
20         var total: Int = 10
21         var answered: Int = 3
22     }
23
24     override val progressText: String new *
25     get() = "$answered of $total answered"
26
27     override fun printProgressBar() { new *
28         repeat(answered) { print("█") }
29         repeat(times: total - answered) { print("░") }
30         println()
31         println(progressText)
32     }
33 }
34
35 fun main() { new *
36     Quiz().printQuiz()
37 }
```

Run ParametroObjetoKt

"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=61164" -Dfile.encoding=UTF-8 -DmainClass=ParametroObjetoKt

Quoth the raven \_\_\_  
██░░░░░░  
3 of 10 answered

```
17 class Quiz : ProgressPrintable { new *
18
19     companion object StudentProgress { new *
20         var total: Int = 10
21         var answered: Int = 3
22     }
23
24     override val progressText: String new *
25     get() = "$answered of $total answered"
26
27     override fun printProgressBar() { new *
28         repeat(answered) { print("█") }
29         repeat(times: total - answered) { print("░") }
30         println()
31         println(progressText)
32     }
33 }
34
35 fun main() { new *
36     Quiz().printQuiz()
37 }
```

Run ParametroObjetoKt

"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=61528" -Dfile.encoding=UTF-8 -DmainClass=ParametroObjetoKt

Quoth the raven \_\_\_  
██░░░░░░  
3 of 10 answered

```
fun main() {  
    val rockPlanets = arrayOf<String>("Mercury", "Venus", "Earth", "Mars")  
    val gasPlanets = arrayOf("Jupiter", "Saturn", "Uranus", "Neptune")  
    val solarSystem = rockPlanets + gasPlanets  
  
    println(solarSystem[0])  
    println(solarSystem[1])  
    println(solarSystem[2])  
    println(solarSystem[3])  
    println(solarSystem[4])  
    println(solarSystem[5])  
    println(solarSystem[6])  
    println(solarSystem[7])  
}
```

Run Console Output:

```
Mercury  
Venus  
Earth  
Mars  
Jupiter  
Saturn  
Uranus
```

```
fun main() {  
    val rockPlanets = arrayOf<String>("Mercury", "Venus", "Earth", "Little Earth")  
    val gasPlanets = arrayOf("Jupiter", "Saturn", "Uranus", "Neptune")  
    val solarSystem = rockPlanets + gasPlanets  
  
    println(solarSystem[0])  
    println(solarSystem[1])  
    println(solarSystem[2])  
    println(solarSystem[3])  
    println(solarSystem[4])  
    println(solarSystem[5])  
    println(solarSystem[6])  
    println(solarSystem[7])  
}
```

Run Console Output:

```
Mercury  
Venus  
Earth  
Little Earth  
Jupiter
```

```
fun main() {  
    val rockPlanets = arrayOf<String>("Mercury", "Venus", "Earth", "Little Earth")  
    val gasPlanets = arrayOf("Jupiter", "Saturn", "Uranus", "Neptune")  
    val newSolarSystem = arrayOf("Mercury", "Venus", "Earth", "Mars", "Jupiter", "Saturn", "Uranus", "Neptune")  
    val solarSystem = rockPlanets + gasPlanets  
  
    println(solarSystem[0])  
    println(solarSystem[1])  
    println(solarSystem[2])  
    println(solarSystem[3])  
    println(solarSystem[4])  
    println(solarSystem[5])  
    println(solarSystem[6])  
    println(solarSystem[7])  
    println(newSolarSystem[0])  
}
```

Run Console Output:

```
Jupiter  
Saturn  
Uranus  
Neptune  
Pluto
```

```
Project > KotlinListasyMateriales C:\Users\spart\workspace\KotlinListasyMateriales
  > src
    > Colecciones.kt
    > Main.kt
    > ParametroObjeto.kt
    > KotlinListasyMateriales.iml
    > External Libraries
    > Scratches and Consoles

Run > ColeccionesKt 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=62756" -Dfile.encoding=UTF-8 -Dsun.java2d.c
8
```

```
8      println(solarSystem[0])
9      println(solarSystem[1])
10     println(solarSystem[2])
11     println(solarSystem[3])
12     println(solarSystem[4])
13     println(solarSystem[5])
14     println(solarSystem[6])
15     println(solarSystem[7])
16     println(newSolarSystem[8])
17 }
18 */
19
20 fun main() { new *
21     val solarSystem = listOf("Mercury", "Venus", "Earth", "Mars", "Jupiter", "Saturn", "Uranus", "Neptune")
22     println(solarSystem.size)
23 }
```

```
Project > KotlinListasyMateriales C:\Users\spart\workspace\KotlinListasyMateriales
  > src
    > Colecciones.kt
    > Main.kt
    > ParametroObjeto.kt
    > KotlinListasyMateriales.iml
    > External Libraries
    > Scratches and Consoles

Run > ColeccionesKt 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=63029" -Dfile.encoding=UTF-8 -Dsun.
8
Earth
Mars
```

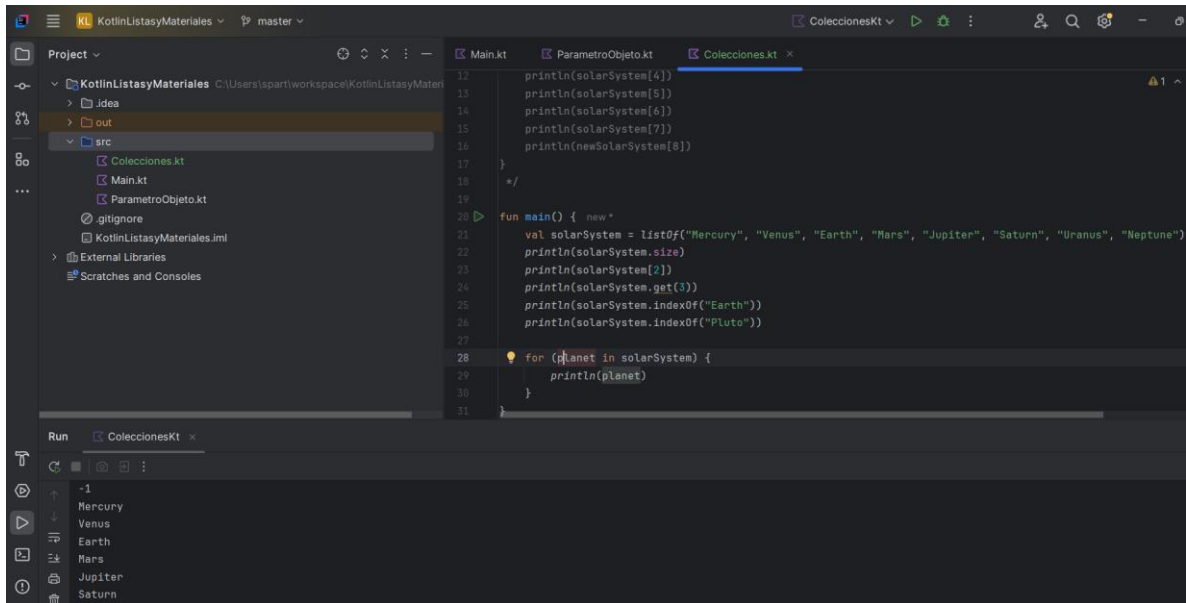
```
8      println(solarSystem[0])
9      println(solarSystem[1])
10     println(solarSystem[2])
11     println(solarSystem[3])
12     println(solarSystem[4])
13     println(solarSystem[5])
14     println(solarSystem[6])
15     println(solarSystem[7])
16     println(newSolarSystem[8])
17 }
18 */
19
20 fun main() { new *
21     val solarSystem = listOf("Mercury", "Venus", "Earth", "Mars", "Jupiter", "Saturn", "Uranus", "Neptune")
22     println(solarSystem.size)
23     println(solarSystem[2])
24     println(solarSystem.get(3))
25 }
```

```
Project > KotlinListasyMateriales C:\Users\spart\workspace\KotlinListasyMateriales
  > src
    > Colecciones.kt
    > Main.kt
    > ParametroObjeto.kt
    > KotlinListasyMateriales.iml
    > External Libraries
    > Scratches and Consoles

Run > ColeccionesKt x

8
Earth
Mars
2
-1
```

```
11     println(solarSystem[3])
12     println(solarSystem[4])
13     println(solarSystem[5])
14     println(solarSystem[6])
15     println(solarSystem[7])
16     println(newSolarSystem[8])
17 }
18 */
19
20 fun main() { new *
21     val solarSystem = listOf("Mercury", "Venus", "Earth", "Mars", "Jupiter", "Saturn", "Uranus", "Neptune")
22     println(solarSystem.size)
23     println(solarSystem[2])
24     println(solarSystem.get(3))
25     println(solarSystem.indexOf("Earth"))
26     println(solarSystem.indexOf("Pluto"))
27 }
```



```
12 println(solarSystem[4])
13 println(solarSystem[5])
14 println(solarSystem[6])
15 println(solarSystem[7])
16 println(solarSystem[8])
17 }
18 */
19
20 fun main() { new *
21     val solarSystem = listOf("Mercury", "Venus", "Earth", "Mars", "Jupiter", "Saturn", "Uranus", "Neptune")
22     println(solarSystem.size)
23     println(solarSystem[2])
24     println(solarSystem.get(3))
25     println(solarSystem.indexOf("Earth"))
26     println(solarSystem.indexOf("Pluto"))
27
28     for (planet in solarSystem) {
29         println(planet)
30     }
31 }
```

Run ColeccionesKt x

-1

Mercury

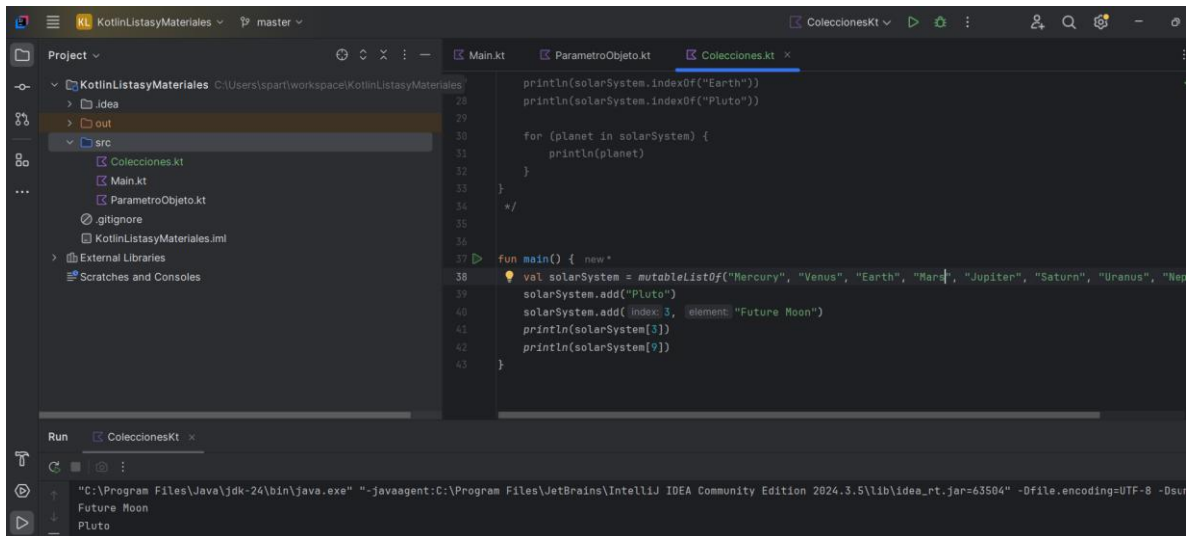
Venus

Earth

Mars

Jupiter

Saturn



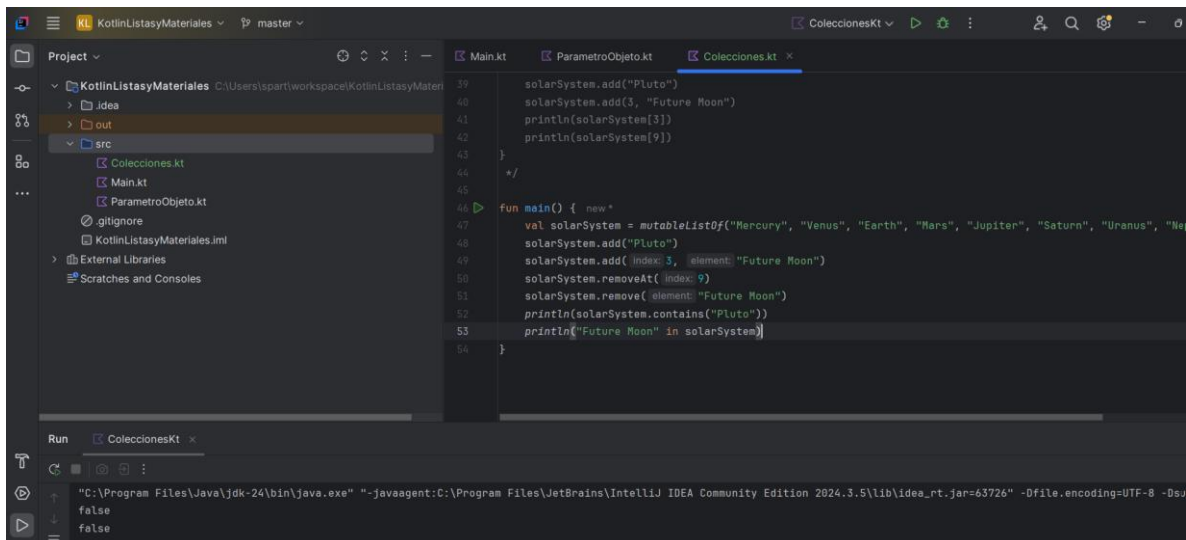
```
28 println(solarSystem.indexOf("Earth"))
29 println(solarSystem.indexOf("Pluto"))
30
31 for (planet in solarSystem) {
32     println(planet)
33 }
34 */
35
36
37 fun main() { new *
38     val solarSystem = mutableListOf("Mercury", "Venus", "Earth", "Mars", "Jupiter", "Saturn", "Uranus", "Neptune")
39     solarSystem.add("Pluto")
40     solarSystem.add(index=3, element="Future Moon")
41     println(solarSystem[3])
42     println(solarSystem[9])
43 }
```

Run ColeccionesKt 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=63504" -Dfile.encoding=UTF-8 -Dsun.java2d.c

Future Moon

Pluto



```
39 solarSystem.add("Pluto")
40 solarSystem.add(3, "Future Moon")
41 println(solarSystem[3])
42 println(solarSystem[9])
43 }
44 */
45
46 fun main() { new *
47     val solarSystem = mutableListOf("Mercury", "Venus", "Earth", "Mars", "Jupiter", "Saturn", "Uranus", "Neptune")
48     solarSystem.add("Pluto")
49     solarSystem.add(index=3, element="Future Moon")
50     solarSystem.removeAt(index=9)
51     solarSystem.remove(element="Future Moon")
52     println(solarSystem.contains("Pluto"))
53     println("Future Moon" in solarSystem)
54 }
```

Run ColeccionesKt 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=63726" -Dfile.encoding=UTF-8 -Dsun.java2d.c

false

false







```
77 fun main() { new *
81     "Earth" to 1,
82     "Mars" to 2,
83     "Jupiter" to 79,
84     "Saturn" to 89,
85     "Uranus" to 22,
86     "Neptune" to 34
87 }
88 println(solarSystem.size)
89 solarSystem["Pluto"] = 5
90 println(solarSystem.size)
91 println(solarSystem["Pluto"])
92 println(solarSystem.get("Jupiter"))
93 solarSystem.remove(key="Pluto")
94 println(solarSystem.size)
95 solarSystem["Jupiter"] = 78
96 println(solarSystem["Jupiter"])
97 }
```

```
45 val cookies = listOf(
46     Cookie(
47         name = "Sugar and Sprinkles",
48         softBaked = false,
49         hasFilling = false,
50         price = 1.39
51     )
52 )
53 fun main() { new *
54     cookies.forEach {
55         println("Menu item: $it")
56     }
57 }
```

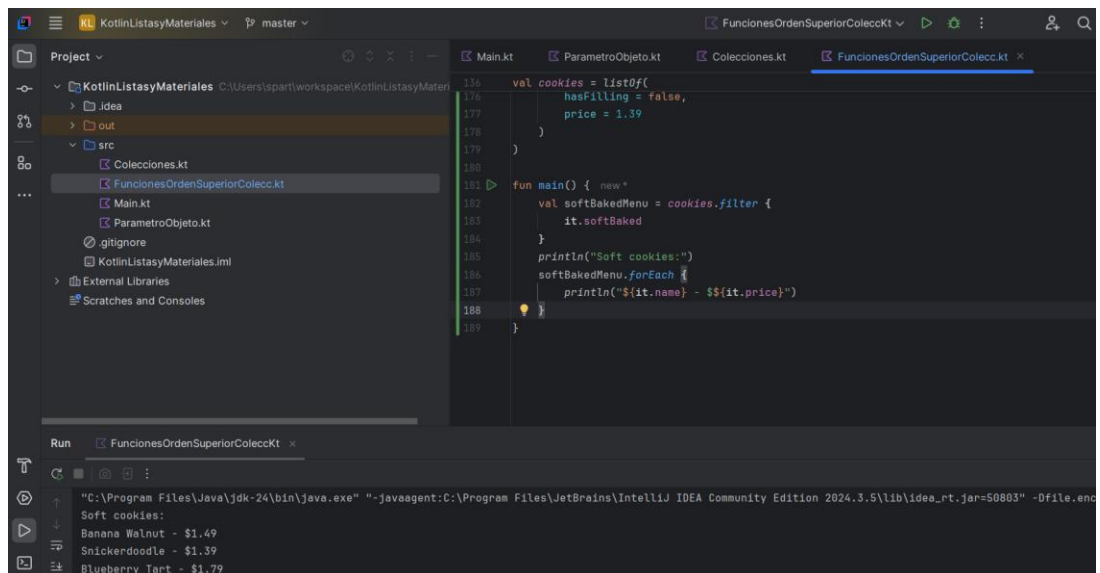
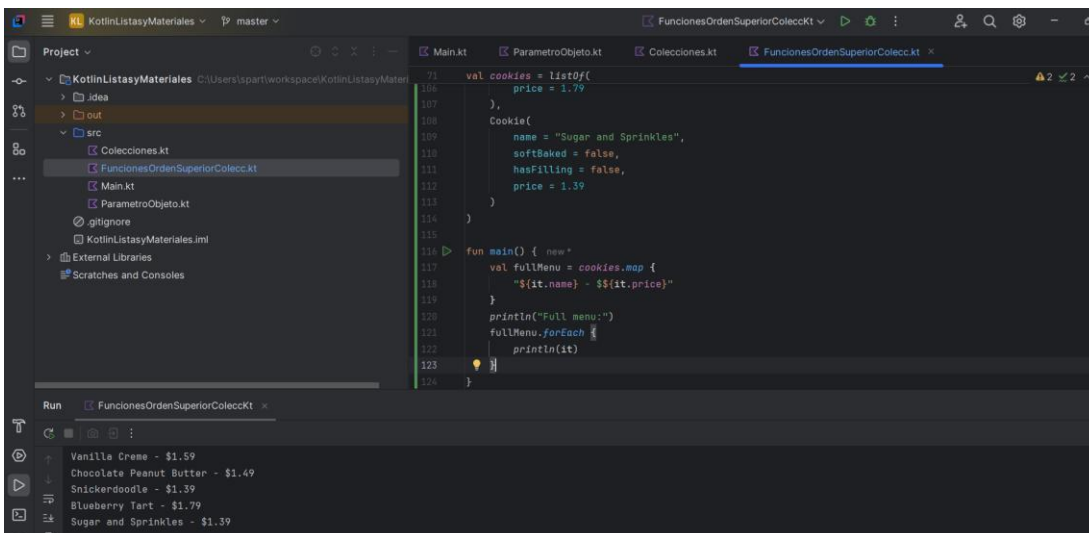
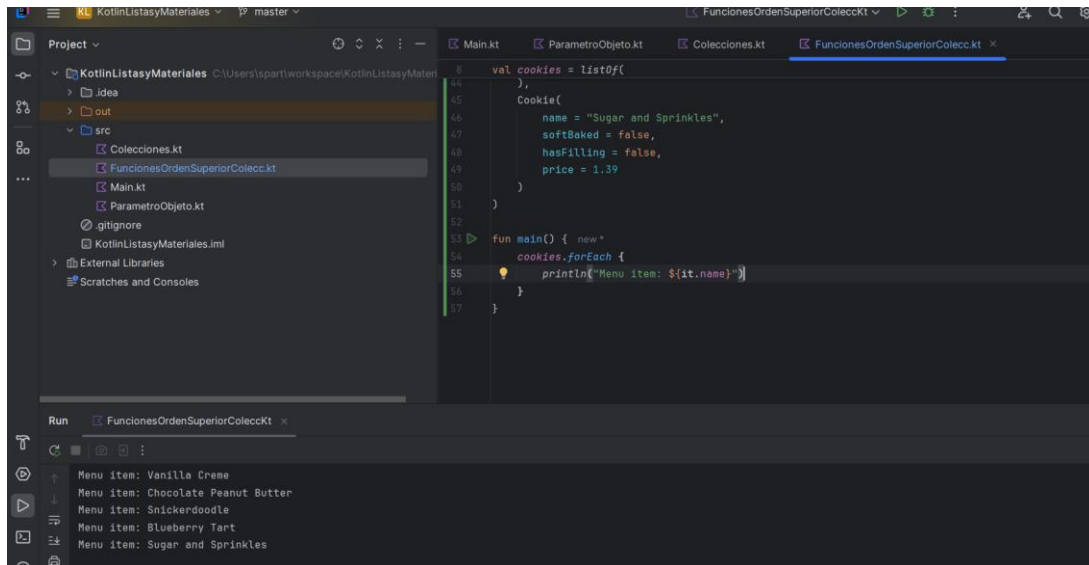
Run FuncionesOrdenSuperiorColeccKt

```
"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=64650" -Dfile.encoding=UTF-8
Menu item: Cookie@448139f0
Menu item: Cookie@7cca494b
Menu item: Cookie@7ba4f24f
Menu item: Cookie@3b9a45b3
Menu item: Cookie@7699a589
Menu item: Cookie@58372a00
Menu item: Cookie@4dd8dc3
```

```
45 val cookies = listOf(
46     Cookie(
47         name = "Sugar and Sprinkles",
48         softBaked = false,
49         hasFilling = false,
50         price = 1.39
51     )
52 )
53 fun main() { new *
54     cookies.forEach {
55         println("Menu item: $it.name")
56     }
57 }
```

Run FuncionesOrdenSuperiorColeccKt

```
Menu item: Cookie@7ba4f24f.name
Menu item: Cookie@3b9a45b3.name
Menu item: Cookie@7699a589.name
Menu item: Cookie@58372a00.name
Menu item: Cookie@4dd8dc3.name
```



This screenshot shows the IntelliJ IDEA IDE with a Kotlin project named 'KotlinListasyMateriales'. The 'Project' view on the left shows the file structure with 'FuncionesOrdenSuperiorColecc.kt' selected. The editor displays the following Kotlin code:

```
247 fun main() { new *
250 val crunchyMenu = groupedMenu[false] ?: listOf()
251 val totalPrice = cookies.fold(initial: 0.0) {total, cookie ->
252     total + cookie.price
253 }
254 println("Soft cookies:")
255 softBakedMenu.forEach {
256     println("${it.name} - ${it.price}")
257 }
258 println("Crunchy cookies:")
259 crunchyMenu.forEach {
260     println("${it.name} - ${it.price}")
261 }
262 println("Total price: ${totalPrice}")
263 }
```

The 'Run' view at the bottom shows the output of the program:

```
Chocolate Chip - $1.69
Vanilla Creme - $1.59
Chocolate Peanut Butter - $1.49
Sugar and Sprinkles - $1.39
Total price: $10.83
```

This screenshot shows the IntelliJ IDEA IDE with the same Kotlin project. The 'Project' view shows 'FuncionesOrdenSuperiorColecc.kt' selected. The editor displays the following Kotlin code:

```
247 fun main() { new *
251 println("Soft cookies:")
252 softBakedMenu.forEach {
253     println("${it.name} - ${it.price}")
254 }
255 println("Crunchy cookies:")
256 crunchyMenu.forEach {
257     println("${it.name} - ${it.price}")
258 }
259 println("Total price: ${totalPrice}")
260
261 println("Alphabetical menu:")
262 alphabeticalMenu.forEach {
263     println(it.name)
264 }
265 }
```

The 'Run' view at the bottom shows the output of the program:

```
Chocolate Chip
Chocolate Peanut Butter
Snickerdoodle
Sugar and Sprinkles
Vanilla Creme
```

This screenshot shows the IntelliJ IDEA IDE with the same Kotlin project. The 'Project' view shows 'Practicas.kt' selected. The editor displays the following Kotlin code:

```
1 // Definición de la clase de datos Event
2 data class Event(
3     val title: String,
4     val description: String?,
5     val daypart: String,
6     val durationInMinutes: Int
7 )
8
9 fun main() {
10     // Creación de una instancia de Event con los datos indicados
11     val studyEvent = Event(
12         title = "Study Kotlin",
13         description = "Commit to studying Kotlin at least 15 minutes per day.",
14         daypart = "Evening",
15         durationInMinutes = 15
16     )
17
18     // Impresión del objeto
19     println(studyEvent)
20 }
```

The 'Run' view at the bottom shows the output of 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=51347" -Dfile.encoding=UTF-8
Event(title=Study Kotlin, description=Commit to studying Kotlin at least 15 minutes per day., daypart=Evening, durationInMinutes=15)
```

```
enum class Daypart {  
    EVENING  
}  
  
// Clase de datos Event usando el enum Daypart  
data class Event(  
    val title: String,  
    val description: String?,  
    val daypart: Daypart,  
    val durationInMinutes: Int  
)  
  
fun main() {  
    // Crear una instancia del evento usando el enum  
    val studyEvent = Event(  
        title = "Study Kotlin",  
        description = "Commit to studying Kotlin at least 15 minutes per day.",  
        daypart = Daypart.EVENING,  
        durationInMinutes = 15  
    )  
}
```

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=51537" -Dfile.encoding=UTF-8  
Event(title=Study Kotlin, description=Commit to studying Kotlin at least 15 minutes per day., daypart=EVENING, durationInMinutes=15)

```
data class Event(  
    val title: String,  
    val description: String? = null,  
    val daypart: Daypart,  
    val durationInMinutes: Int  
)  
  
fun main() {  
    // Crear eventos individuales  
    val event1 = Event(title = "Wake up", description = "Time to get up", daypart = Daypart.MORNING,  
    val event2 = Event(title = "Eat breakfast", daypart = Daypart.MORNING, durationInMinutes = 15)  
    val event3 = Event(title = "Learn about Kotlin", daypart = Daypart.AFTERNOON, durationInMinutes = 30)  
    val event4 = Event(title = "Practice Compose", daypart = Daypart.AFTERNOON, durationInMinutes = 60)  
    val event5 = Event(title = "Watch latest DevBytes video", daypart = Daypart.AFTERNOON, durationInMinutes = 10)  
    val event6 = Event(title = "Check out latest Android Jetpack library", daypart = Daypart.EVENING, durationInMinutes = 45)  
  
    // Agrupar todos los eventos en una lista mutable  
    val events = mutableListOf<Event>(event1, event2, event3, event4, event5, event6)  
  
    // Mostrar el total de eventos  
}
```

Run PracticasKt

Event(title=Eat breakfast, description=null, daypart=MORNING, durationInMinutes=15)  
Event(title=Learn about Kotlin, description=null, daypart=AFTERNOON, durationInMinutes=30)  
Event(title=Practice Compose, description=null, daypart=AFTERNOON, durationInMinutes=60)  
Event(title=Watch latest DevBytes video, description=null, daypart=AFTERNOON, durationInMinutes=10)  
Event(title=Check out latest Android Jetpack library, description=null, daypart=EVENING, durationInMinutes=45)

```
data class Event(  
    val title: String,  
    val description: String? = null,  
    val daypart: Daypart,  
    val durationInMinutes: Int  
)  
  
fun main() {  
    // Crear eventos  
    val event1 = Event(title = "Wake up", description = "Time to get up", daypart = Daypart.MORNING,  
    val event2 = Event(title = "Eat breakfast", daypart = Daypart.MORNING, durationInMinutes = 15)  
    val event3 = Event(title = "Learn about Kotlin", daypart = Daypart.AFTERNOON, durationInMinutes = 30)  
    val event4 = Event(title = "Practice Compose", daypart = Daypart.AFTERNOON, durationInMinutes = 60)  
    val event5 = Event(title = "Watch latest DevBytes video", daypart = Daypart.AFTERNOON, durationInMinutes = 10)  
    val event6 = Event(title = "Check out latest Android Jetpack library", daypart = Daypart.EVENING, durationInMinutes = 45)  
  
    // Lista de eventos  
    val events = mutableListOf<Event>(event1, event2, event3, event4, event5, event6)  
  
    // Mostrar el total de eventos  
}
```

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=51537" -Dfile.encoding=UTF-8  
You have 5 short events.

```
145 fun main() {
155
156 // lista de eventos
157 val events = mutableListOf(event1, event2, event3, event4, event5)
158
159 // agrupar eventos por franja horaria
160 val groupedEvents = events.groupBy { it.daypart }
161
162 // imprimir resumen
163 groupedEvents.forEach { (daypart, eventsInPart) ->
164     println("${daypart.name.capitalize()}: ${eventsInPart.size}")
165 }
166 }
167
```

Run PracticasKt 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" 167168169170171172173174175176177178179180181182183184185

MORNING: 3 events  
AFTERNOON: 3 events  
EVENING: 2 events

```
167 }
168
169 */
170
171 data class Event(val title: String, val time: String)
172
173 fun main() {
174     val events = listOf(
175         Event(title="Breakfast Meeting", time="08:00 AM"),
176         Event(title="Team Standup", time="10:00 AM"),
177         Event(title="Lunch with Client", time="12:30 PM"),
178         Event(title="Project Presentation", time="03:00 PM"),
179         Event(title="Wrap-up and Review", time="05:00 PM")
180     )
181
182     println("Last event of the day: ${events.last().title}")
183 }
184
185
```

Run PracticasKt 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" 167168169170171172173174175176177178179180181182183184185

Last event of the day: Wrap-up and Review

```
192 val Event.durationOfEvent: String
193     get() = if (this.durationInMinutes < 60) {
194         "Long"
195     }
196
197 fun main() {
198     val events = listOf(
199         Event(title="Breakfast Meeting", time="08:00 AM", durationInMinutes=120),
200         Event(title="Team Standup", time="10:00 AM", durationInMinutes=15),
201         Event(title="Lunch with Client", time="12:30 PM", durationInMinutes=45),
202         Event(title="Project Presentation", time="03:00 PM", durationInMinutes=30),
203         Event(title="Wrap-up and Review", time="05:00 PM", durationInMinutes=10)
204     )
205
206     println("Duration of first event of the day: ${events[0].durationOfEvent}")
207 }
208
209
210
```

Run PracticasKt 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" 192193194195196197198199200201202203204205206207208209210

Duration of first event of the day: short

## Crea una lista desplazable

<https://github.com/Abraham20070511/TrainingAffirmations>

```
// Declaración del paquete
package com.example.affirmations

// Imports necesarios para Android, Jetpack Compose y recursos
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.material3.*
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.platform.LocalLayoutDirection
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.res.stringResource
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import com.example.affirmations.R
import com.example.affirmations.data.Datasource
import com.example.affirmations.model.Affirmation
import com.example.affirmations.ui.theme.AffirmationsTheme

// Clase principal de la aplicación (punto de entrada)
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        // Establece el contenido de la UI usando Jetpack Compose
        setContent {
            // Aplica el tema definido en ui/theme
            AffirmationsTheme {
                // Crea un fondo que cubre toda la pantalla
            }
        }
    }
}
```

```

        Surface(
            modifier = Modifier.fillMaxSize(),
            color = MaterialTheme.colorScheme.background
        ){
            // Llama a la función principal de la UI
            AffirmationsApp()
        }
    }
}
}
}

```

// Función principal de la UI

@Composable

fun AffirmationsApp() {

val layoutDirection = *LocalLayoutDirection*.current

// Crea un contenedor principal con relleno para las barras de estado

Surface(

modifier = Modifier

.*fillMaxSize*()

.*statusBarsPadding*()

.*padding*(

start = WindowInsets.*safeDrawing*.asPaddingValues()

.*calculateStartPadding*(layoutDirection),

end = WindowInsets.*safeDrawing*.asPaddingValues()

.*calculateEndPadding*(layoutDirection),

),

){

// Muestra la lista de afirmaciones cargadas desde el Datasource

AffirmationList(

affirmationList = Datasource().loadAffirmations()

)

}

}

// Muestra una lista vertical (LazyColumn) de tarjetas de afirmación

@Composable

fun AffirmationList(affirmationList: List<Affirmation>, modifier: Modifier = Modifier) {



```

LazyColumn(modifier = modifier) {
    // Recorre la lista de afirmaciones y crea una tarjeta para cada una
    items(affirmationList) { affirmation ->
        AffirmationCard(
            affirmation = affirmation,
            modifier = Modifier.padding(8.dp) // Espaciado entre tarjetas
        )
    }
}
}

// Muestra una tarjeta con imagen y texto de una afirmación
@Composable
fun AffirmationCard(affirmation: Affirmation, modifier: Modifier = Modifier) {
    Card(modifier = modifier) {
        Column {
            // Muestra la imagen asociada a la afirmación
            Image(
                painter = painterResource(affirmation.imageResourceId),
                contentDescription = stringResource(affirmation.stringResourceId),
                modifier = Modifier
                    .fillMaxWidth()
                    .height(194.dp), // Altura fija para la imagen
                contentScale = ContentScale.Crop // Recorta para llenar el espacio
            )
            // Muestra el texto de la afirmación
            Text(
                text = LocalContext.current.getString(affirmation.stringResourceId),
                modifier = Modifier.padding(16.dp),
                style = MaterialTheme.typography.headlineSmall
            )
        }
    }
}
}

```

```

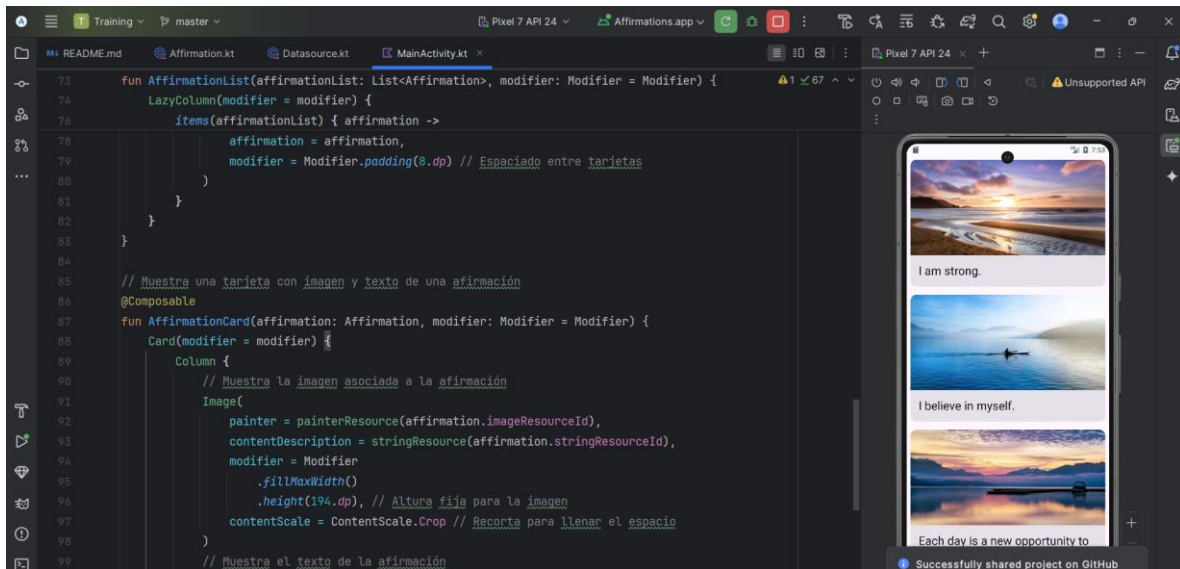
// Vista previa en el editor de Android Studio
@Preview(showBackground = true)
@Composable

```

```

private fun AffirmationCardPreview() {
    AffirmationsTheme {
        // Muestra una tarjeta de ejemplo con un recurso simulado
        AffirmationCard(
            affirmation = Affirmation(R.string.affirmation1, R.drawable.image1)
        )
    }
}

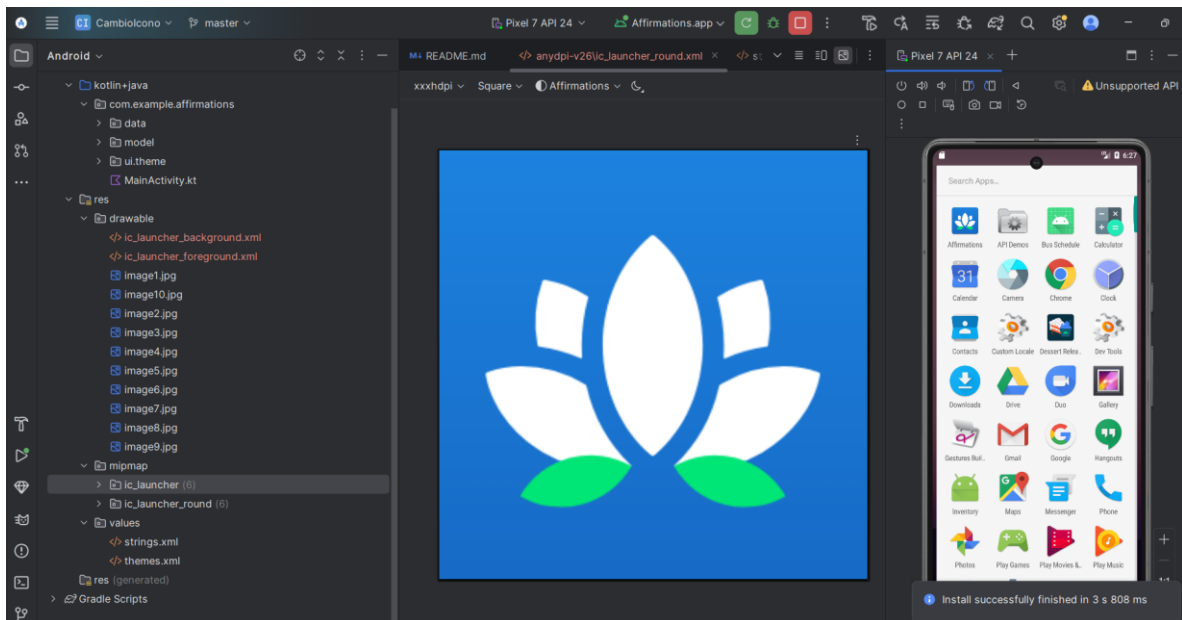
```



Una aplicación Android desarrollada con Jetpack Compose que muestra una lista de afirmaciones motivacionales, cada una acompañada por una imagen. Al iniciar, la app aplica un tema visual definido y carga una lista de objetos `Affirmation` desde una clase `Datasource`. Cada afirmación se presenta en una tarjeta (`Card`) que contiene una imagen y un texto, ambos obtenidos desde recursos del proyecto. La interfaz utiliza una lista perezosa (`LazyColumn`) para renderizar eficientemente los elementos. Además, se ajusta automáticamente al espacio seguro de la pantalla (status bar) y permite previsualizar componentes individuales durante el desarrollo con la anotación `@Preview`.

## Cambio de iconos

<https://github.com/Abraham20070511/Cambiolcono>



Este proyecto es una aplicación de Android donde implementé el cambio dinámico del ícono de la app utilizando Jetpack Compose. La idea principal fue aprender y demostrar cómo modificar el ícono principal de la aplicación desde el código, mostrando distintos íconos según se seleccione dentro de la app.

## Cuadrícula

<https://github.com/Abraham20070511/Cuadrícula>

/\*

\* Copyright (C) 2023 The Android Open Source Project

\* Licencia Apache 2.0

\*/

package com.example.courses

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.\*

import androidx.compose.foundation.lazy.grid.GridCells

```
import androidx.compose.foundation.lazy.grid.LazyVerticalGrid
import androidx.compose.foundation.lazy.grid.items
import androidx.compose.material3.*
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.dimensionResource
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.res.stringResource
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import com.example.courses.data.DataSource
import com.example.courses.model.Topic
import com.example.courses.ui.theme.CoursesTheme
```

// Actividad principal de la aplicación

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        enableEdgeToEdge() // Habilita diseño de borde a borde (pantalla completa)
        super.onCreate(savedInstanceState)
        setContent {
            CoursesTheme {
                // Contenedor principal con color de fondo del tema
                Surface(
                    modifier = Modifier
                        .fillMaxSize()
                        .statusBarsPadding(),
                    color = MaterialTheme.colorScheme.background
                ) {
                    // Muestra una cuadrícula de temas
                    TopicGrid(
                        modifier = Modifier.padding(
                            start = dimensionResource(R.dimen.padding_small),
                            top = dimensionResource(R.dimen.padding_small),
                            end = dimensionResource(R.dimen.padding_small),
                        )
                    )
                }
            }
        }
    }
}
```

```

        }
    }
}
}

// Composable que muestra los temas en una cuadrícula de 2 columnas
@Composable
fun TopicGrid(modifier: Modifier = Modifier) {
    LazyVerticalGrid(
        columns = GridCells.Fixed(2), // Fija dos columnas
        verticalArrangement =
            Arrangement.spacedBy(dimensionResource(R.dimen.padding_small)),
        horizontalArrangement =
            Arrangement.spacedBy(dimensionResource(R.dimen.padding_small)),
        modifier = modifier
    ){
        // Recorre la lista de temas y los muestra como tarjetas
        items(dataSource.topics) { topic ->
            TopicCard(topic)
        }
    }
}

```

```

// Composable que representa una tarjeta de tema individual
@Composable
fun TopicCard(topic: Topic, modifier: Modifier = Modifier) {
    Card {
        Row {
            Box {
                // Imagen representativa del tema
                Image(
                    painter = painterResource(id = topic.imageRes),
                    contentDescription = null,
                    modifier = modifier
                        .size(width = 68.dp, height = 68.dp)
                        .aspectRatio(1f),
                    contentScale = ContentScale.Crop
                )
            }
        }
    }
}

```

```

    }

    // Columna con el nombre del tema y número de cursos
    Column {
        // Nombre del tema
        Text(
            text = stringResource(id = topic.name),
            style = MaterialTheme.typography.bodyMedium,
            modifier = Modifier.padding(
                start = dimensionResource(R.dimen.padding_medium),
                top = dimensionResource(R.dimen.padding_medium),
                end = dimensionResource(R.dimen.padding_medium),
                bottom = dimensionResource(R.dimen.padding_small)
            )
        )
    }
    // Fila con ícono y número de cursos disponibles
    Row(verticalAlignment = Alignment.CenterVertically) {
        Icon(
            painter = painterResource(R.drawable.ic_grain),
            contentDescription = null,
            modifier = Modifier
                .padding(start = dimensionResource(R.dimen.padding_medium))
        )
        Text(
            text = topic.availableCourses.toString(),
            style = MaterialTheme.typography.labelMedium,
            modifier = Modifier.padding(start =
dimensionResource(R.dimen.padding_small))
        )
    }
}
}
}
}
}
}

```

```

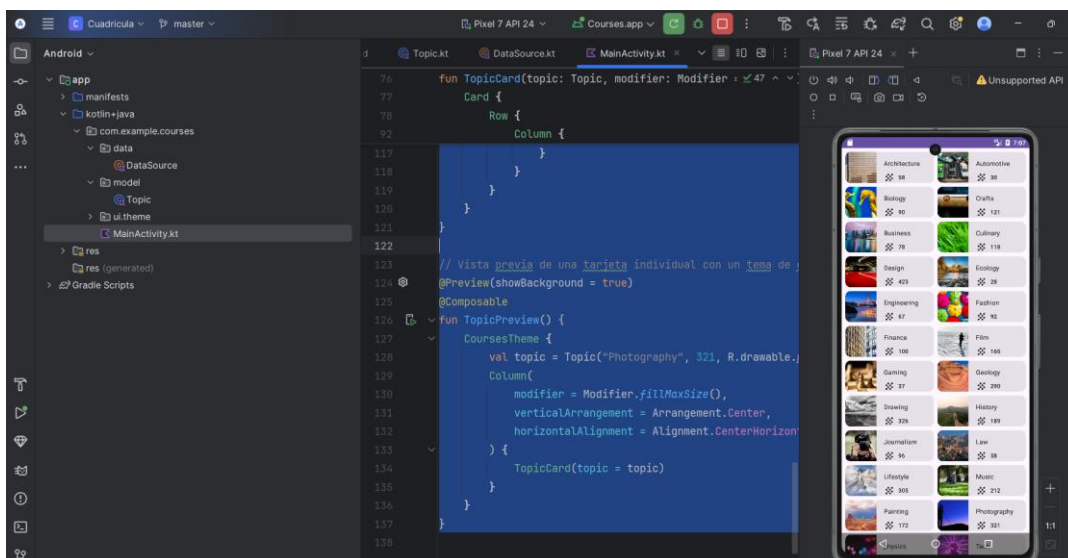
// Vista previa de una tarjeta individual con un tema de ejemplo
@Preview(showBackground = true)
@Composable

```

```

fun TopicPreview() {
    CoursesTheme {
        val topic = Topic(R.string.photography, 321, R.drawable.photography)
        Column(
            modifier = Modifier.fillMaxSize(),
            verticalArrangement = Arrangement.Center,
            horizontalAlignment = Alignment.CenterHorizontally
        ){
            TopicCard(topic = topic)
        }
    }
}

```



TopicGrid es una función composible que organiza las tarjetas en un LazyVerticalGrid, una grilla de dos columnas con separación uniforme entre los elementos. Cada elemento de esta grilla representa un tema, generado dinámicamente a partir de los datos definidos en el objeto DataSource.

El composable TopicCard define cómo luce cada tarjeta: en una fila horizontal, se muestra una imagen del tema en un Box y, junto a ella, una columna con el nombre del tema y el número de cursos. Esta estructura se adapta bien a dispositivos móviles, manteniendo el contenido legible y bien distribuido.

Finalmente, la función TopicPreview permite previsualizar cómo se ve una tarjeta individual durante el desarrollo. Esto es útil para verificar el diseño visual de los elementos sin tener que ejecutar toda la aplicación.



# Compila apps fabulosas

## Woof

<https://github.com/Abraham20070511/Woof>

/\*

\* Copyright (C) 2023 The Android Open Source Project

\*/

package com.example.woof

import android.os.Bundle

import androidx.activity.ComponentActivity

import androidx.activity.compose.setContent

import androidx.annotation.DrawableRes

import androidx.compose.ui.draw.clip

import androidx.annotation.StringRes

import androidx.compose.foundation.Image

import androidx.compose.foundation.layout.Column

import androidx.compose.foundation.layout.Row

import androidx.compose.foundation.layout.fillMaxSize

import androidx.compose.foundation.layout.fillMaxWidth

import androidx.compose.material3.CenterAlignedTopAppBar

import androidx.compose.ui.Alignment

import androidx.compose.foundation.layout.padding

import androidx.compose.foundation.layout.size

import androidx.compose.foundation.lazy.LazyColumn

import androidx.compose.foundation.lazy.items

import androidx.compose.material3.ExperimentalMaterial3Api

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.ui.Modifier

import androidx.compose.ui.layout.ContentScale

import androidx.compose.ui.res.dimensionResource

import androidx.compose.ui.res.painterResource

```

import androidx.compose.ui.res.stringResource
import androidx.compose.ui.tooling.preview.Preview
import com.example.woof.data.Dog
import com.example.woof.data.dogs
import com.example.woof.ui.theme.WoofTheme

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            WoofTheme {
                // A surface container using the 'background' color from the theme
                Surface(
                    modifier = Modifier.fillMaxSize()
                ){
                    WoofApp()
                }
            }
        }
    }
}

/**
 * Composable that displays an app bar and a list of dogs.
 */
@Composable
fun WoofApp() {
    Scaffold(
        topBar = {
            WoofTopAppBar()
        }
    ){ it ->
        LazyColumn(contentPadding = it) {
            items(dogs) {
                DogItem(
                    dog = it,
                    modifier = Modifier.padding(dimensionResource(R.dimen.padding_small))
                )
            }
        }
    }
}

```

```

    }
  }
}
}

```

```

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun WoofTopAppBar(modifier: Modifier = Modifier) {
    CenterAlignedTopAppBar(
        title = {
            Row(
                verticalAlignment = Alignment.CenterVertically
            ){
                Image(
                    modifier = Modifier
                        .size(dimensionResource(id = R.dimen.image_size))
                        .padding(dimensionResource(id = R.dimen.padding_small)),
                    painter = painterResource(R.drawable.ic_woof_logo),

                    contentDescription = null
                )
                Text(
                    text = stringResource(R.string.app_name),
                    style = MaterialTheme.typography.displayLarge
                )
            }
        },
        modifier = modifier
    )
}

```

```

/**
 * Composable that displays a list item containing a dog icon and their information.
 *
 * @param dog contains the data that populates the list item
 * @param modifier modifiers to set to this composable
 */

```

```

@Composable

```

```

fun DogItem(
    dog: Dog,
    modifier: Modifier = Modifier
){
    androidx.compose.material3.Card(modifier = modifier) {
        Row(
            modifier = Modifier
                .fillMaxWidth()
                .padding(dimensionResource(R.dimen.padding_small))
        ){
            DogIcon(dog.imageResourceId)
            DogInformation(dog.name, dog.age)
        }
    }
}

```

```

/**

```

```

 * Composable that displays a photo of a dog.

```

```

 *

```

```

 * @param dogIcon is the resource ID for the image of the dog

```

```

 * @param modifier modifiers to set to this composable

```

```

 */

```

```

@Composable

```

```

fun DogIcon(

```

```

    @DrawableRes dogIcon: Int,

```

```

    modifier: Modifier = Modifier

```

```

){

```

```

    Image(

```

```

        modifier = modifier

```

```

        .size(dimensionResource(id = R.dimen.image_size))

```

```

        .padding(dimensionResource(id = R.dimen.padding_small))

```

```

        .clip(MaterialTheme.shapes.small),

```

```

        contentScale = ContentScale.Crop,

```

```

        painter = painterResource(dogIcon),

```

```

        // Content Description is not needed here - image is decorative, and setting a null
        content
    )
}

```

```

        // description allows accessibility services to skip this element during navigation.

        contentDescription = null
    )
}

/**
 * Composable that displays a dog's name and age.
 *
 * @param dogName is the resource ID for the string of the dog's name
 * @param dogAge is the Int that represents the dog's age
 * @param modifier modifiers to set to this composable
 */
@Composable
fun DogInformation(
    @StringRes dogName: Int,
    dogAge: Int,
    modifier: Modifier = Modifier
){
    Column(modifier = modifier) {
        Text(
            text = stringResource(dogName),
            modifier = Modifier.padding(top = dimensionResource(R.dimen.padding_small))
        )
        Text(
            text = stringResource(R.string.years_old, dogAge),
            style = MaterialTheme.typography.bodyLarge
        )
    }
}

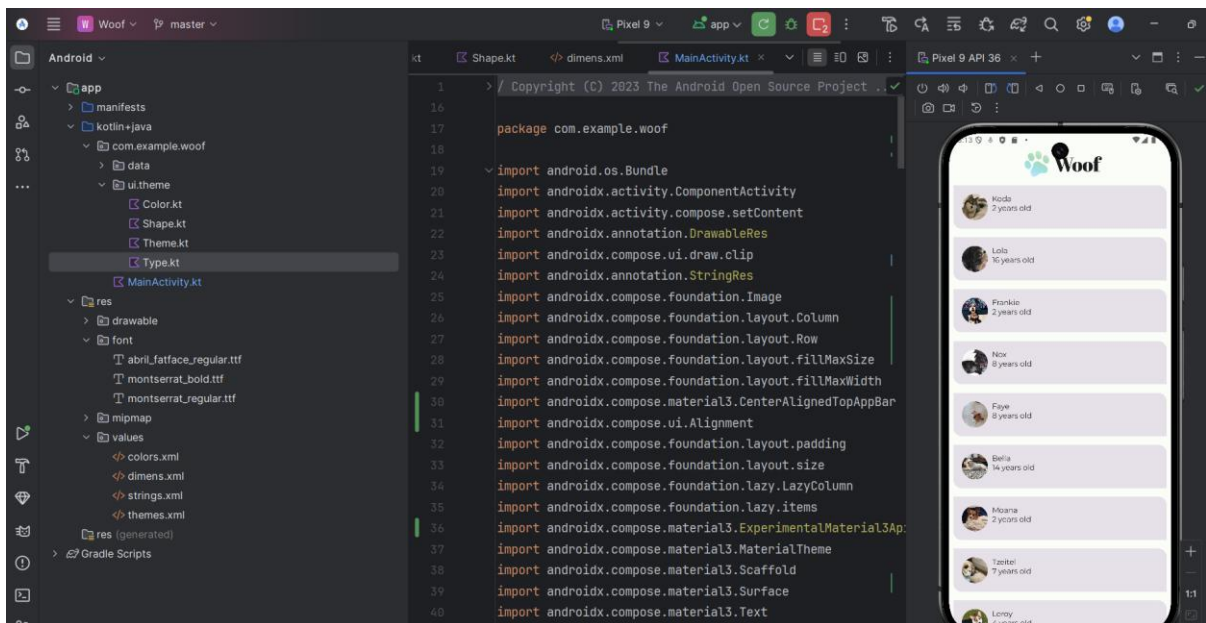
@Preview
@Composable
fun WoofPreview() {
    WoofTheme(darkTheme = false) {
        WoofApp()
    }
}

```

```

/**
 * Composable that displays what the UI of the app looks like in light theme in the
 * design tab.
 */
@Preview
@Composable
fun WoofDarkThemePreview() {
    WoofTheme(darkTheme = true) {
        WoofApp()
    }
}

```



El código define la interfaz de una app que muestra una lista de perros usando Jetpack Compose. La estructura principal se organiza con un Scaffold que incluye una barra superior personalizada con el logo y el nombre de la app. Cada perro se muestra dentro de una tarjeta con su imagen y datos. La imagen se recorta de forma decorativa y los textos se extraen de los recursos. También se incluyen vistas previas en tema claro y oscuro para facilitar el diseño.

## WoofAnimation

<https://github.com/Abraham20070511/WoofAnimation>

// Copyright de Google para el código base del proyecto Woof

```
package com.example.woof
```

```
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
```

```
// Anotaciones para recursos
import androidx.annotation.DrawableRes
import androidx.annotation.StringRes
```

```
// Elementos de diseño de Compose
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.Image
import androidx.compose.material.icons.filled.ExpandMore
import androidx.compose.material.icons.Icons
import androidx.compose.material3.Icon
```

```
// Layouts y estructuras
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.runtime.setValue
```

```
// Animaciones
import androidx.compose.animation.animateContentSize
import androidx.compose.animation.core.Spring
import androidx.compose.animation.core.spring
```

```
// Más layouts
```



```
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.size
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
```

```
// Iconos y componentes Material
```

```
import androidx.compose.material.icons.filled.ExpandLess
import androidx.compose.material3.Card
import androidx.compose.material3.CenterAlignedTopAppBar
import androidx.compose.material3.IconButton
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Scaffold
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
```

```
// Compose runtime y herramientas
```

```
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.dimensionResource
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.res.stringResource
import androidx.compose.ui.tooling.preview.Preview
```

```
// Datos del modelo y tema
```

```
import com.example.woof.data.Dog
import com.example.woof.data.dogs
import com.example.woof.ui.theme.WoofTheme
```

```
// Clase principal de la actividad que lanza la app
```

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        // Se establece el contenido principal de la app usando Jetpack Compose
        setContent {
            WoofTheme {
```

```

        // Superficie que ocupa toda la pantalla con fondo según el tema
        Surface(modifier = Modifier.fillMaxSize()) {
            WoofApp()
        }
    }
}
}
}

```

// Composable principal que organiza la interfaz de la app

@Composable

fun WoofApp() {

    Scaffold(

        topBar = {

            WoofTopAppBar() // Barra superior personalizada

        }

    ){ it ->

        // Lista desplazable de perros

        LazyColumn(contentPadding = it) {

            items(dogs) { dog ->

                DogItem(

                    dog = dog,

                    modifier = Modifier.padding(dimensionResource(R.dimen.padding\_small))

                )

            }

        }

    }

}

// Composable que representa cada tarjeta de perro

@Composable

fun DogItem(

    dog: Dog,

    modifier: Modifier = Modifier

){

    // Estado para saber si el elemento está expandido

    var expanded by remember { mutableStateOf(false) }

```

// Tarjeta para contener la información del perro
Card(modifier = modifier) {
    // Columna con animación al cambiar de tamaño
    Column(
        modifier = Modifier.animateContentSize(
            animationSpec = spring(
                dampingRatio = Spring.DampingRatioNoBounciness,
                stiffness = Spring.StiffnessMedium
            )
        )
    ){
        // Fila con imagen, datos e icono para expandir
        Row(
            modifier = Modifier
                .fillMaxWidth()
                .padding(dimensionResource(R.dimen.padding_small))
        ){
            DogIcon(dog.imageResourceId)
            DogInformation(dog.name, dog.age)
            Spacer(modifier = Modifier.weight(1f)) // Espacio para empujar el botón al final
            DogIconButton(
                expanded = expanded,
                onClick = { expanded = !expanded } // Cambia el estado expandido
            )
        }
        // Si está expandido, muestra los pasatiempos del perro
        if (expanded) {
            DogHobby(
                dog.hobbies,
                modifier = Modifier.padding(
                    start = dimensionResource(R.dimen.padding_medium),
                    top = dimensionResource(R.dimen.padding_small),
                    end = dimensionResource(R.dimen.padding_medium),
                    bottom = dimensionResource(R.dimen.padding_medium)
                )
            )
        }
    }
}

```

```
}  
}
```

```
// Composable que muestra los pasatiempos del perro
```

```
@Composable
```

```
fun DogHobby(
```

```
    @StringRes dogHobby: Int,
```

```
    modifier: Modifier = Modifier
```

```
){
```

```
    Column(modifier = modifier) {
```

```
        Text(
```

```
            text = stringResource(R.string.about),
```

```
            style = MaterialTheme.typography.labelSmall
```

```
        )
```

```
        Text(
```

```
            text = stringResource(dogHobby),
```

```
            style = MaterialTheme.typography.bodyLarge
```

```
        )
```

```
    }
```

```
}
```

```
// Botón para expandir o contraer el contenido
```

```
@Composable
```

```
private fun DogItemButton(
```

```
    expanded: Boolean,
```

```
    onClick: () -> Unit,
```

```
    modifier: Modifier = Modifier
```

```
){
```

```
    IconButton(onClick = onClick) {
```

```
        Icon(
```

```
            imageVector = if (expanded) Icons.Filled.ExpandLess else
```

```
Icons.Filled.ExpandMore,
```

```
            contentDescription =
```

```
stringResource(R.string.expand_button_content_description),
```

```
            tint = MaterialTheme.colorScheme.secondary
```

```
        )
```

```
    }
```

```
}
```

```

// Barra superior con el nombre y logo de la app
@Composable
fun WoofTopAppBar(modifier: Modifier = Modifier) {
    CenterAlignedTopAppBar(
        title = {
            Row(verticalAlignment = Alignment.CenterVertically) {
                Image(
                    modifier = Modifier
                        .size(dimensionResource(R.dimen.image_size))
                        .padding(dimensionResource(R.dimen.padding_small)),
                    painter = painterResource(R.drawable.ic_woof_logo),
                    contentDescription = null // decorativa
                )
                Text(
                    text = stringResource(R.string.app_name),
                    style = MaterialTheme.typography.displayLarge
                )
            }
        },
        modifier = modifier
    )
}

```

```

// Imagen circular del perro
@Composable
fun DogIcon(
    @DrawableRes dogIcon: Int,
    modifier: Modifier = Modifier
){
    Image(
        modifier = modifier
            .size(dimensionResource(R.dimen.image_size))
            .padding(dimensionResource(R.dimen.padding_small))
            .clip(MaterialTheme.shapes.small), // forma redondeada
        contentScale = ContentScale.Crop,
        painter = painterResource(dogIcon),
        contentDescription = null // decorativa
    )
}

```

```

    )
}

// Muestra el nombre y la edad del perro
@Composable
fun DogInformation(
    @StringRes dogName: Int,
    dogAge: Int,
    modifier: Modifier = Modifier
){
    Column(modifier = modifier) {
        Text(
            text = stringResource(dogName),
            style = MaterialTheme.typography.displayMedium,
            modifier = Modifier.padding(top = dimensionResource(R.dimen.padding_small))
        )
        Text(
            text = stringResource(R.string.years_old, dogAge),
            style = MaterialTheme.typography.bodyLarge
        )
    }
}

// Vista previa en Android Studio con tema claro
@Preview
@Composable
fun WoofPreview() {
    WoofTheme(darkTheme = false) {
        WoofApp()
    }
}

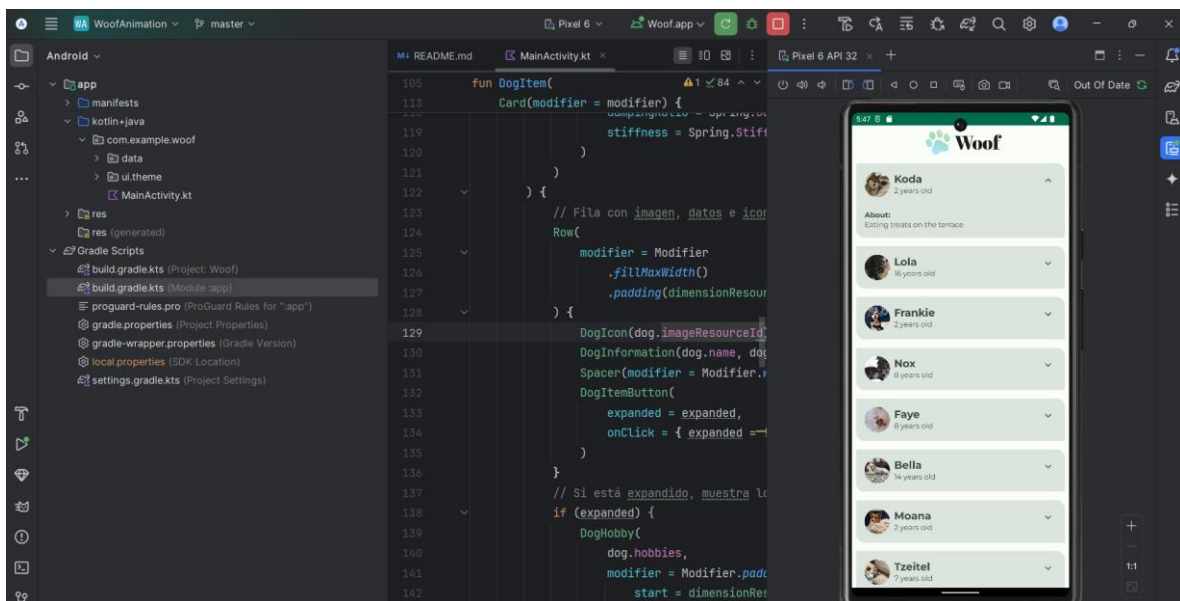
```

// Vista previa en Android Studio con tema oscuro

@Preview

@Composable

```
fun WoofDarkThemePreview() {  
    WoofTheme(darkTheme = true) {  
        WoofApp()  
    }  
}
```



Este código define una app en Jetpack Compose que muestra una lista de perros con su nombre, edad y pasatiempos. Usa un diseño con Scaffold, incluyendo una barra superior personalizada y una lista desplazable. Cada perro se muestra en una tarjeta que puede expandirse para revelar más información. La imagen, nombre y edad se cargan dinámicamente desde recursos. También incluye vistas previas para modo claro y oscuro, facilitando el diseño desde Android Studio.



## SuperHeroes

<https://github.com/Abraham20070511/SuperHeroes>

```
/*
 * Copyright (C) 2023 The Android Open Source Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * https://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
```

```
package com.example.superheroes
```

```
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material3.CenterAlignedTopAppBar
import androidx.compose.material3.ExperimentalMaterial3Api
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.ui.Modifier
import androidx.activity.enableEdgeToEdge
import androidx.compose.ui.res.stringResource
import androidx.compose.ui.tooling.preview.Preview
import com.example.superheroes.model.HeroesRepository
import com.example.superheroes.ui.theme.SuperheroesTheme
```

```
// Clase principal de la aplicación, representa la Activity de entrada.
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge() // Habilita diseño de borde a borde (sin padding por defecto
en status/nav bar)
        setContent {
            // Aplica el tema personalizado SuperheroesTheme a toda la interfaz
            SuperheroesTheme {
                // Surface proporciona un fondo utilizando el color del tema
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
                ){
                    SuperheroesApp() // Llama al composable principal de la app
                }
            }
        }
    }
}
```

// Esta función composable está definida dentro de la Activity, lo cual NO es recomendable.

// Debería definirse fuera de la clase MainActivity para seguir buenas prácticas de Jetpack Compose.

```
/**
 * Composable principal que muestra la estructura general de la app: una
TopAppBar y una lista de héroes.
 */
@Composable
fun SuperheroesApp() {
    Scaffold(
        modifier = Modifier.fillMaxSize(),
        topBar = {
            TopAppBar() // Barra superior centrada con el nombre de la app
        }
    ){

```

```

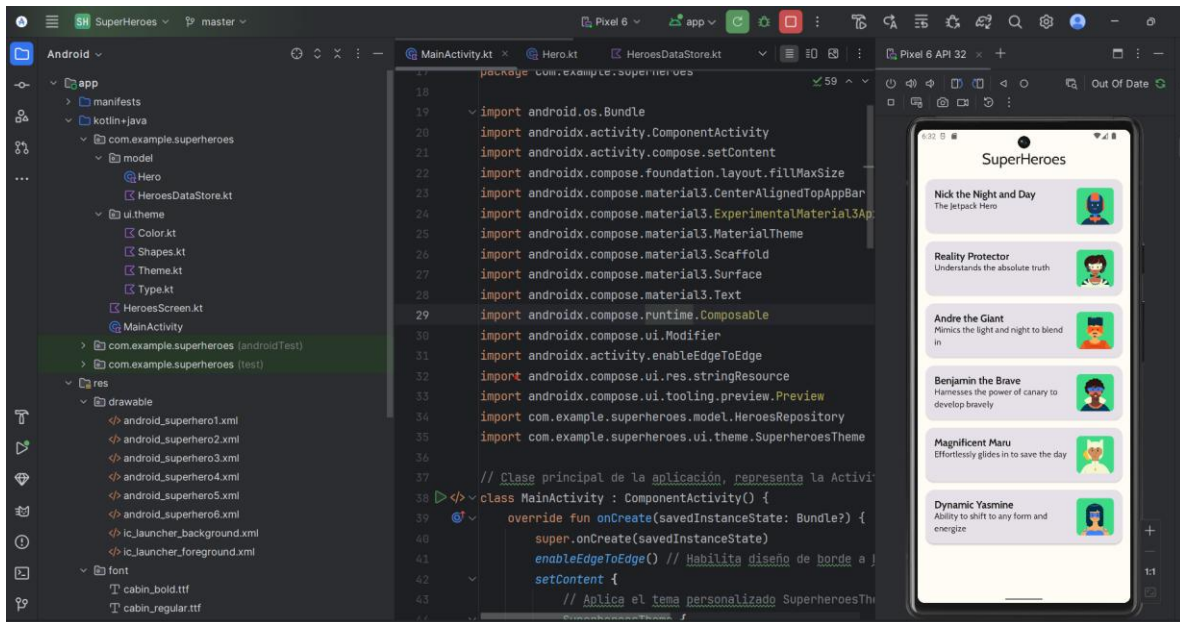
        // Acceder directamente al repositorio desde la UI no es una buena práctica.
        // En el futuro se usará un ViewModel que exponga los datos.
        val heroes = HeroesRepository.heroes
        HeroesList(heroes = heroes, contentPadding = it) // Muestra la lista de héroes
    }
}

/**
 * Composable que muestra una barra superior con el nombre de la app.
 *
 * @param modifier permite modificar el comportamiento visual del componente
 */
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun TopAppBar(modifier: Modifier = Modifier) {
    CenterAlignedTopAppBar(
        title = {
            Text(
                text = stringResource(R.string.app_name), // Obtiene el texto desde
strings.xml
                style = MaterialTheme.typography.displayLarge,
            )
        },
        modifier = modifier
    )
}

/**
 * Vista previa para Android Studio: permite ver cómo se verá la interfaz sin ejecutar
la app.
 */
@Preview(showBackground = true)
@Composable
fun SuperHeroesPreview() {
    SuperheroesTheme {
        SuperheroesApp()
    }
}

```

```
}  
}
```



Este código define la pantalla principal de una aplicación Android llamada *SuperHeroes*, utilizando Jetpack Compose. La clase *MainActivity* hereda de *ComponentActivity* y dentro de su método *onCreate()* configura el contenido de la interfaz con el tema *SuperheroesTheme*. También activa el diseño de borde a borde (*enableEdgeToEdge()*), que permite que la interfaz use todo el espacio de pantalla.

Dentro de la interfaz, se utiliza un componente *Surface* que actúa como fondo, y dentro de él se llama a *SuperheroesApp()*, que es la función composible principal. Esta función crea la estructura visual de la app usando un *Scaffold*, con una barra superior (*TopAppBar*) y una lista de héroes (*HeroesList*), obtenidos directamente desde un repositorio de datos llamado *HeroesRepository*.

La barra superior, definida en *TopAppBar()*, muestra el nombre de la aplicación centrado, usando el estilo de texto del tema. Además, hay una función de vista previa (*SuperHeroesPreview()*) que permite visualizar la interfaz directamente desde Android Studio sin ejecutar la app.

Una observación importante es que las funciones composables están definidas dentro de la clase *MainActivity*, lo cual no es recomendable. Se sugiere moverlas fuera para seguir buenas prácticas de organización del código en Compose. También se advierte que en el futuro se debería usar un *ViewModel* en lugar de acceder al repositorio directamente desde la UI.