

INSTITUTO TECNOLÓGICO DE CIUDAD MADERO

INGENIERÍA EN SISTEMAS COMPUTACIONALES

INTEGRANTES

PEREZ ANASTASIO KARLA ZAFIRO 20070574

GARAY HERNANDEZ MIGUEL ENRIQUE 20070600

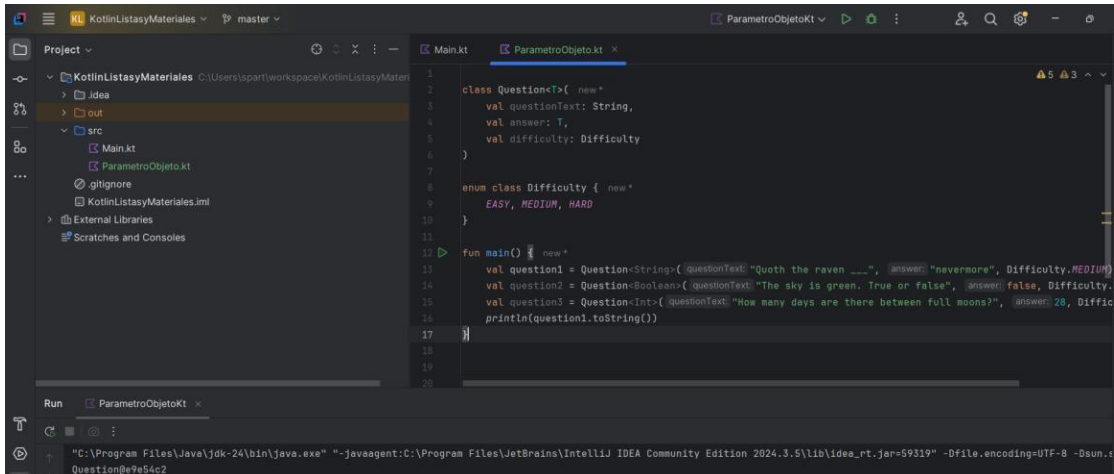
MATERIA

PROGRAMACIÓN NATIVA PARA MÓVILES

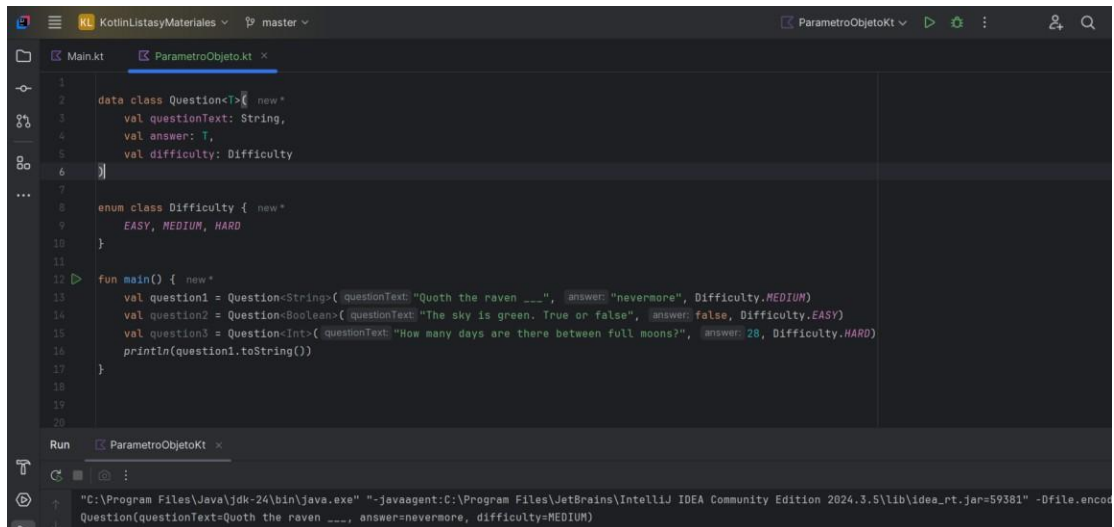
TAREA

Tarea No 9 Como mostrar listas y usar Material Desing

Mas aspectos básicos de kotlin



```
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 data class Question<T> { new *
2     val difficulty: Difficulty
3 }
4
5
6
7
8 enum class Difficulty { new *
9     EASY, MEDIUM, HARD
10 }
11
12 object StudentProgress { new *
13     var total: Int = 10
14     var answered: Int = 3
15 }
16
17 fun main() { new *
18     val question1 = Question<String>("Quoth the raven ____", answer="nevermore", Difficulty.MEDIUM)
19     val question2 = Question<Boolean>("The sky is green. True or false?", answer=false, Difficulty.EASY)
20     val question3 = Question<Int>("How many days are there between full moons?", answer=28, Difficulty.HARD)
21     println(question1.toString())
22     println("${StudentProgress.answered} of ${StudentProgress.total} answered.")
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=59518" -Dfile.encoding=UTF-8 -Dsun...
Question(questionText=Quoth the raven ____, answer=nevermore, difficulty=MEDIUM)
3 of 10 answered.

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

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=59642" -Dfile.encoding=UTF-8 -Dsun...
Question(questionText=Quoth the raven ____, answer=nevermore, difficulty=MEDIUM)
3 of 10 answered.

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

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=59844" -Dfile.encoding=UTF-8 -Dsun...
Question(questionText=Quoth the raven ____, answer=nevermore, difficulty=MEDIUM)
3 of 10 answered



```
class Quiz {
    var answered: Int = 3
}

fun Quiz.StudentProgress.printProgressBar() {
    repeat(Quiz.answered) { print("█") }
    repeat(10 - Quiz.answered) { print("░") }
    println()
    println(Quiz.progressText)
}

val Quiz.StudentProgress.progressText: String
    get() = "${answered} of ${total} answered"

object StudentProgress {
    var total: Int = 10
    var answered: Int = 3
}
```

Run

Question(questionText=Quoth the raven ____, answer=nevermore, difficulty=MEDIUM)
3 of 10 answered
██████████

```
data class Question<T> {
    val questionText: String,
    val answer: T,
    val difficulty: Difficulty
}

enum class Difficulty {
    EASY, MEDIUM, HARD
}

interface ProgressPrintable {
    val progressText: String
    fun printProgressBar()
}

class Quiz : ProgressPrintable {
    val question1 = Question("Quoth the raven ____, answer=nevermore", Difficulty.MEDIUM)
    val question2 = Question("The sky is green. True or false?", answer=false, Difficulty.EASY)
}
```

Run

Question(questionText=Quoth the raven ____, answer=nevermore, difficulty=MEDIUM)
3 of 10 answered

```
class Quiz : ProgressPrintable { new *
    companion object StudentProgress { new *
        var total: Int = 10
        var answered: Int = 3
    }

    override val progressText: String new *
        get() = "$answered of $total answered"

    override fun printProgressBar() { new *
        repeat(answered) { print("█") }
        repeat(times: total - answered) { print("░") }
        println()
        println(progressText)
    }
}

fun main() { new *
```

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 -Dsun

Quoth the raven ---

nevermore

MEDIUM

```
override fun printProgressBar() { new *
    repeat(answered) { print("█") }
    repeat(times: total - answered) { print("░") }
    println()
    println(progressText)
}

fun main() { new *
    Quiz().apply {
        printQuiz()
    }
}
```

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 -Dsun

Quoth the raven ---

nevermore

MEDIUM

```
fun main() { new *
    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 ColeccionesKt

"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=61779" -Dfile.encoding=UTF-8 -Dsun

Mercury

Venus

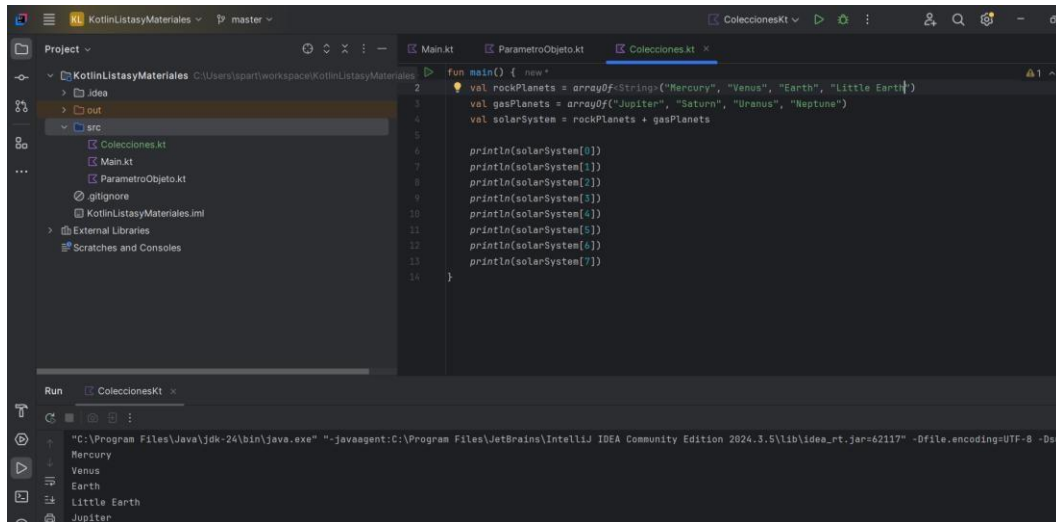
Earth

Mars

Jupiter

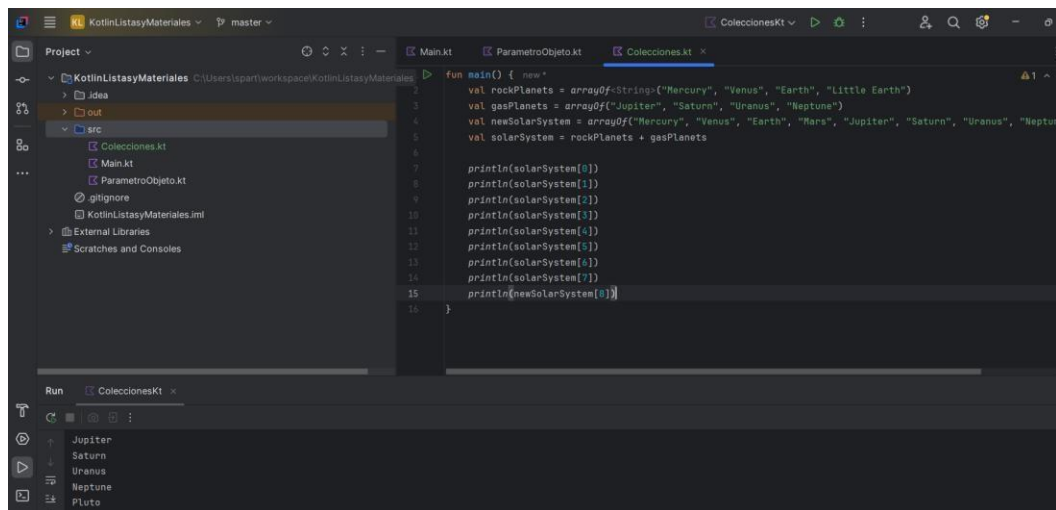
Saturn

Uranus



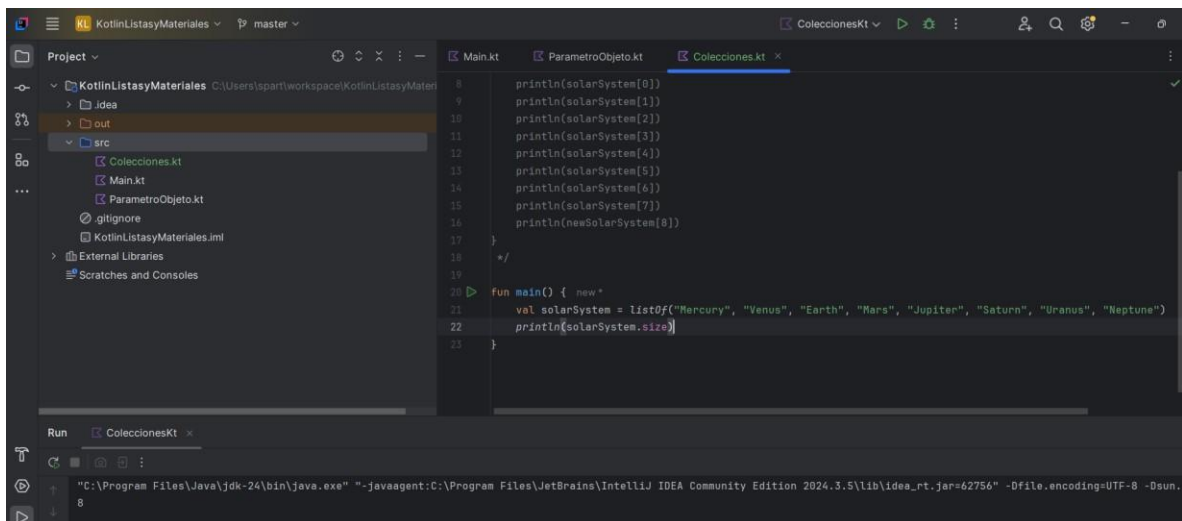
The screenshot shows the IntelliJ IDEA interface with a project named 'KotlinListasyMateriales'. The 'src' directory contains three files: 'Colecciones.kt', 'Main.kt', and 'ParametroObjeto.kt'. The 'Main.kt' file is open, showing a Kotlin function 'main()' that defines two arrays: 'rockPlanets' (Mercury, Venus, Earth, Little Earth) and 'gasPlanets' (Jupiter, Saturn, Uranus, Neptune). These are combined into 'solarSystem'. The code then prints each element of the 'solarSystem' array using 'println(solarSystem[i])' for i from 0 to 7. The 'Run' window at the bottom shows the output: Mercury, Venus, Earth, Little Earth, Jupiter, Saturn, Uranus, Neptune.

```
fun main() { new *  
    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])  
}
```



The screenshot shows the same IntelliJ IDEA interface. The 'Main.kt' file is open, and the code is similar to the previous one, but it includes an additional line: 'val newSolarSystem = arrayOf("Mercury", "Venus", "Earth", "Mars", "Jupiter", "Saturn", "Uranus", "Neptune")'. This new array is added to the 'solarSystem' array. The 'Run' window at the bottom shows the output: Jupiter, Saturn, Uranus, Neptune, Pluto. Note that the output in the image seems to be a mix of the previous run and the new one, with 'Pluto' appearing at the end.

```
fun main() { new *  
    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[8])  
}
```



The screenshot shows the IntelliJ IDEA interface. The 'Main.kt' file is open, and the code is using a 'listOf' function to create the 'solarSystem' array. The code is as follows: 'val solarSystem = listOf("Mercury", "Venus", "Earth", "Mars", "Jupiter", "Saturn", "Uranus", "Neptune")'. The 'Run' window at the bottom shows the output: 8. This suggests that the code is printing the size of the list instead of the elements.

```
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[8])  
}  
*/  
fun main() { new *  
    val solarSystem = listOf("Mercury", "Venus", "Earth", "Mars", "Jupiter", "Saturn", "Uranus", "Neptune")  
    println(solarSystem.size)  
}
```




The screenshot shows the IntelliJ IDEA IDE with a project named 'KotlinListasyMateriales'. The 'src' directory contains three files: 'Colecciones.kt', 'Main.kt', and 'ParametroObjeto.kt'. The 'Colecciones.kt' file is open, showing a Kotlin program that defines a list of planets and prints them. The code is as follows:

```
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 }
```

The Run console shows the output of the program:

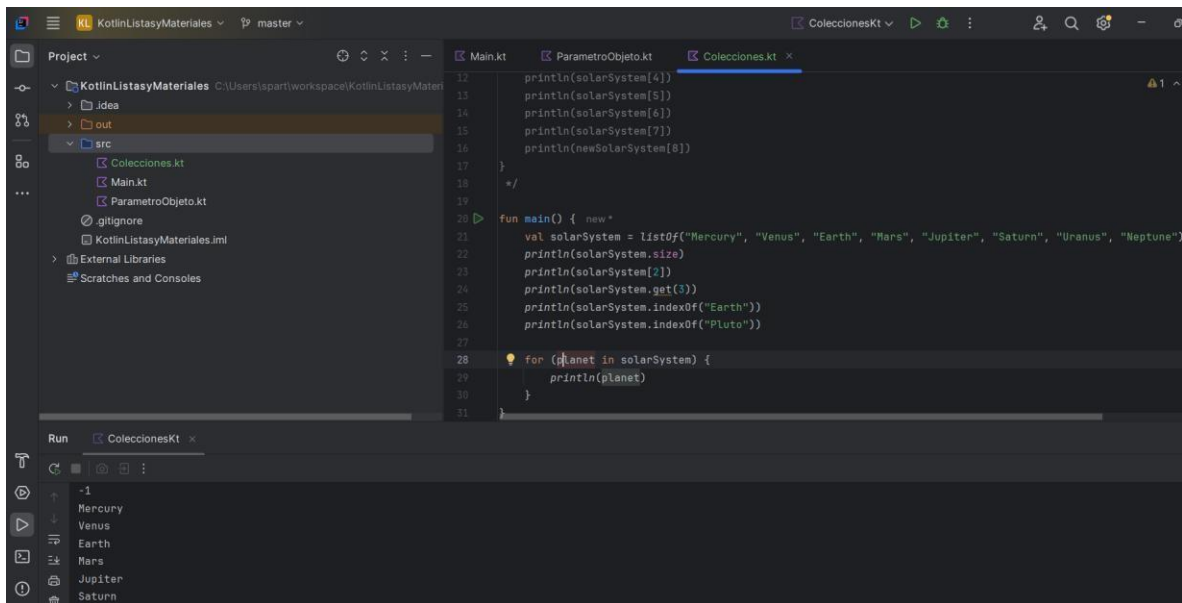
```
"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=63029" -Dfile.encoding=UTF-8 -Dsun.
8
Earth
Mars
```

The screenshot shows the IntelliJ IDEA IDE with the same project. The 'Colecciones.kt' file is open, showing a Kotlin program that defines a list of planets and prints them with their indices. The code is as follows:

```
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 }
```

The Run console shows the output of the program:

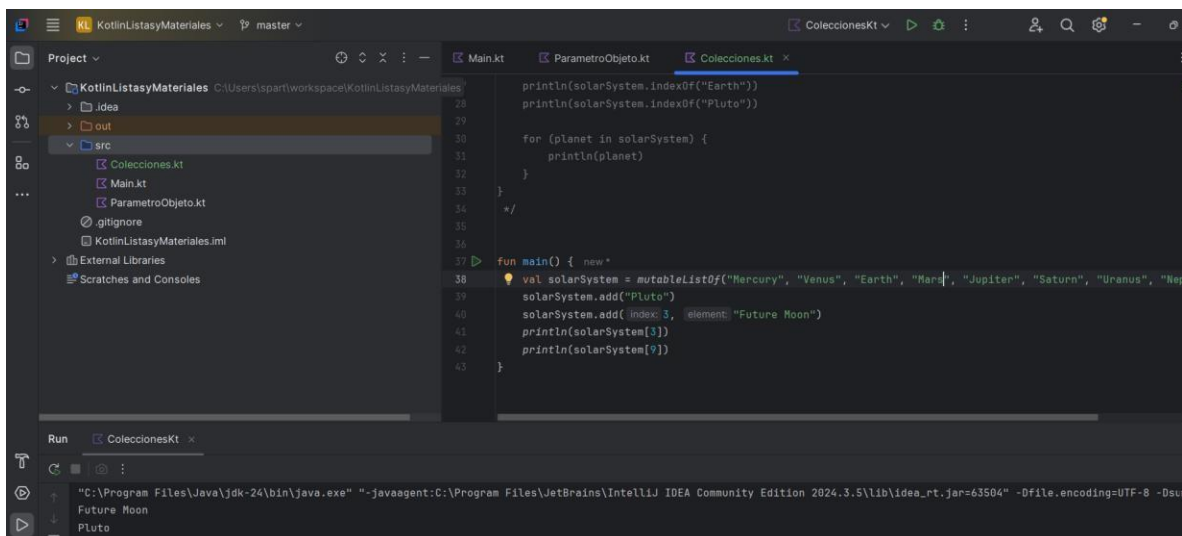
```
8
Earth
Mars
2
-1
```



The screenshot shows the IntelliJ IDEA interface with a project named 'KotlinListasyMateriales'. The 'src' directory contains three files: 'Colecciones.kt', 'Main.kt', and 'ParametroObjeto.kt'. The 'Colecciones.kt' file is open, showing the following code:

```
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
22 val solarSystem = listOf("Mercury", "Venus", "Earth", "Mars", "Jupiter", "Saturn", "Uranus", "Neptune")
23 println(solarSystem.size)
24 println(solarSystem[2])
25 println(solarSystem.get(3))
26 println(solarSystem.indexOf("Earth"))
27 println(solarSystem.indexOf("Pluto"))
28
29 for (planet in solarSystem) {
30     println(planet)
31 }
```

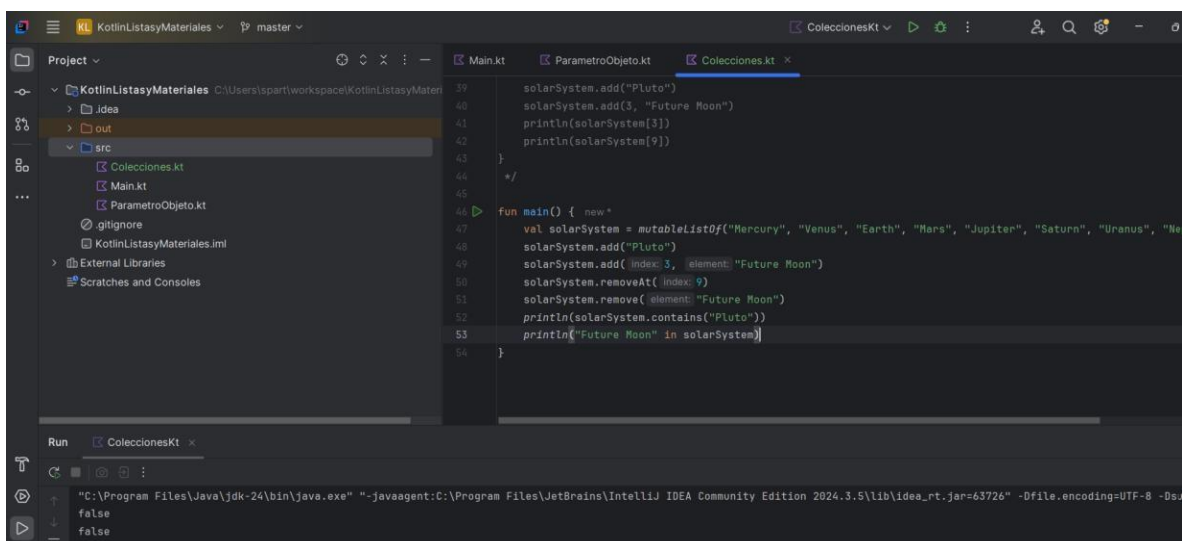
The 'Run' window at the bottom shows the output of the program, which is a list of planets: Mercury, Venus, Earth, Mars, Jupiter, Saturn, and Neptune.



The screenshot shows the IntelliJ IDEA interface with the 'Colecciones.kt' file open. The code has been updated to include Pluto and Future Moon:

```
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
39 val solarSystem = mutableListOf("Mercury", "Venus", "Earth", "Mars", "Jupiter", "Saturn", "Uranus", "Neptune")
40 solarSystem.add("Pluto")
41 solarSystem.add(index=3, element="Future Moon")
42 println(solarSystem[3])
43 println(solarSystem[9])
44 }
```

The 'Run' window at the bottom shows the output of the program, which now includes Future Moon: Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus, Neptune, Future Moon, and Pluto.



The screenshot shows the IntelliJ IDEA interface with the 'Colecciones.kt' file open. The code has been updated to remove Future Moon:

```
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
48 val solarSystem = mutableListOf("Mercury", "Venus", "Earth", "Mars", "Jupiter", "Saturn", "Uranus", "Neptune")
49 solarSystem.add("Pluto")
50 solarSystem.add(index=3, element="Future Moon")
51 solarSystem.removeAt(index=9)
52 solarSystem.remove(element="Future Moon")
53 println(solarSystem.contains("Pluto"))
54 println("Future Moon" in solarSystem)
55 }
```

The 'Run' window at the bottom shows the output of the program, which now only includes Pluto: Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus, Neptune, Future Moon, and Pluto.

The screenshot shows the IntelliJ IDEA interface with a project named 'KotlinListasyMateriales'. The 'src' directory contains three files: 'Colecciones.kt', 'Main.kt', and 'ParametroObjeto.kt'. The 'Colecciones.kt' file is open, showing the following Kotlin code:

```
1 solarSystem.remove("Future Moon")
2 println(solarSystem.contains("Pluto"))
3 println("Future Moon" in solarSystem)
4 }
5
6 */
7
8 fun main() { new *
9     val solarSystem = mutableSetOf("Mercury", "Venus", "Earth", "Mars", "Jupiter", "Saturn", "Uranus", "Neptune")
10
11     println(solarSystem.size)
12     solarSystem.add("Pluto")
13     println(solarSystem.size)
14     println(solarSystem.contains("Pluto"))
15     solarSystem.add("Pluto")
16     println(solarSystem.size)
17 }
```

The 'Run' tab at the bottom shows the execution output:

```
8
9 true
10 false
```

The screenshot shows the IntelliJ IDEA interface with the same project. The 'Colecciones.kt' file is open, showing the following Kotlin code:

```
1 solarSystem.remove("Future Moon")
2 println(solarSystem.contains("Pluto"))
3 println("Future Moon" in solarSystem)
4 }
5
6 */
7
8 fun main() { new *
9     val solarSystem = mutableSetOf("Mercury", "Venus", "Earth", "Mars", "Jupiter", "Saturn", "Uranus", "Neptune")
10
11     println(solarSystem.size)
12     solarSystem.add("Pluto")
13     println(solarSystem.size)
14     println(solarSystem.contains("Pluto"))
15     solarSystem.add("Pluto")
16     println(solarSystem.size)
17     solarSystem.remove(element="Pluto")
18     println(solarSystem.size)
19     println(solarSystem.contains("Pluto"))
20 }
```

The 'Run' tab at the bottom shows the execution output:

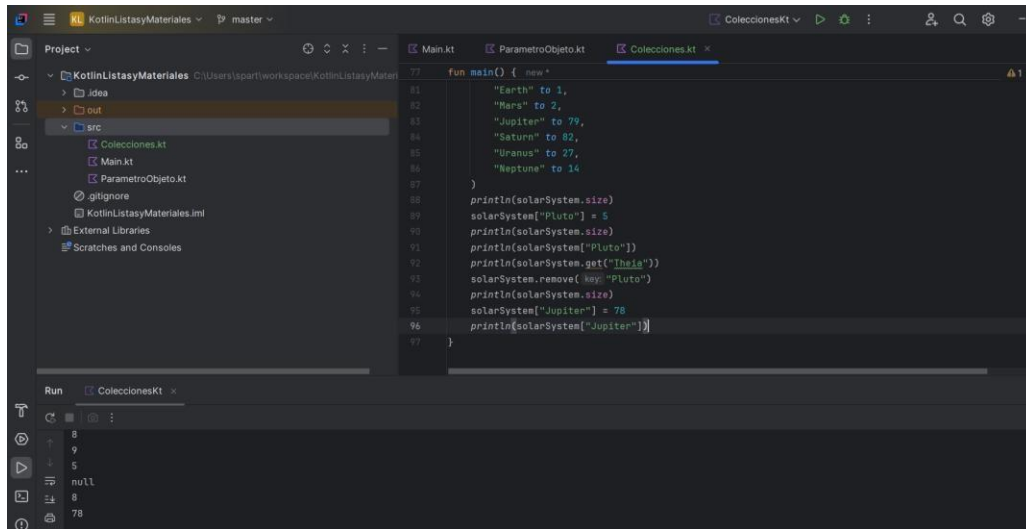
```
9 true
10 false
11 false
```

The screenshot shows the IntelliJ IDEA interface with the same project. The 'Colecciones.kt' file is open, showing the following Kotlin code:

```
1 */
2
3 fun main() { new *
4     val solarSystem = mutableMapOf(
5         "Mercury" to 0,
6         "Venus" to 0,
7         "Earth" to 1,
8         "Mars" to 2,
9         "Jupiter" to 79,
10        "Saturn" to 82,
11        "Uranus" to 27,
12        "Neptune" to 14
13    )
14    println(solarSystem.size)
15    solarSystem["Pluto"] = 5
16    println(solarSystem.size)
17    println(solarSystem["Pluto"])
18    println(solarSystem.get("Neia"))
19 }
```

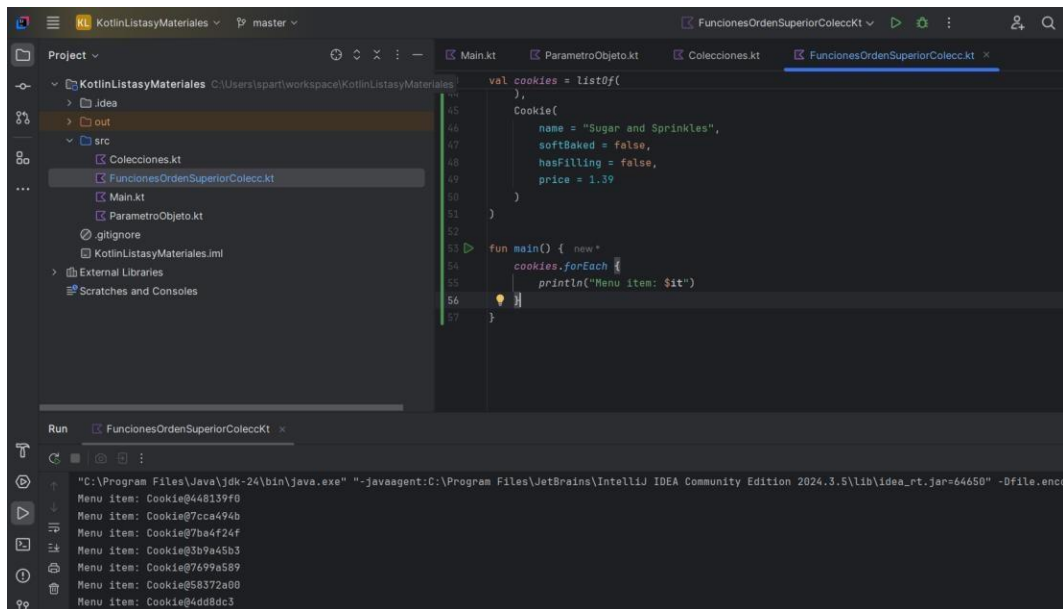
The 'Run' tab at the bottom shows the execution output:

```
8
9 5
10 null
```



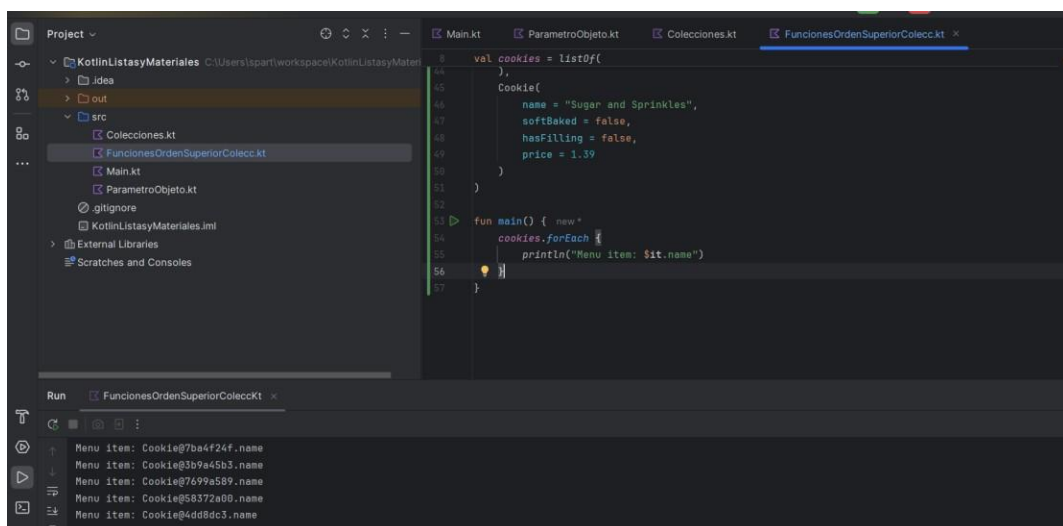
The screenshot shows the IntelliJ IDEA IDE with a Kotlin project named "KotlinListasyMateriales". The project structure includes a "src" directory with files "Colecciones.kt", "Main.kt", and "ParametroObjeto.kt". The "Colecciones.kt" file is open, showing a `fun main()` function that creates a `solarSystem` map, prints its size, adds "Pluto" (size 5), prints the size again, removes "Pluto", prints the size again, adds "Jupiter" (size 78), and prints the size one final time. The Run console shows the output: 8, 9, 5, null, 8, 78.

```
fun main() { new *
    "Earth" to 1,
    "Mars" to 2,
    "Jupiter" to 79,
    "Saturn" to 82,
    "Uranus" to 27,
    "Neptune" to 14
}
println(solarSystem.size)
solarSystem["Pluto"] = 5
println(solarSystem.size)
println(solarSystem["Pluto"])
println(solarSystem.get("Theia"))
solarSystem.remove(key="Pluto")
println(solarSystem.size)
solarSystem["Jupiter"] = 78
println(solarSystem["Jupiter"])
```



The screenshot shows the IntelliJ IDEA IDE with the same Kotlin project. The "FuncionesOrdenSuperiorColecc.kt" file is open, showing a `val cookies = listOf()` declaration and a `Cookie` class with properties `name`, `softBaked`, `hasFilling`, and `price`. The `fun main()` function uses `cookies.forEach` to print "Menu item: \$it". The Run console shows the output: Menu item: Cookie@4d8139f0, Menu item: Cookie@7cca494b, Menu item: Cookie@7ba4f24f, Menu item: Cookie@3b9a45b3, Menu item: Cookie@7699a589, Menu item: Cookie@58372a00, Menu item: Cookie@4dd8dc3.

```
val cookies = listOf()
Cookie {
    name = "Sugar and Sprinkles",
    softBaked = false,
    hasFilling = false,
    price = 1.39
}
fun main() { new *
    cookies.forEach {
        println("Menu item: $it")
    }
}
```



The screenshot shows the IntelliJ IDEA IDE with the same Kotlin project. The "FuncionesOrdenSuperiorColecc.kt" file is open, showing the same `val cookies = listOf()` declaration and `Cookie` class. The `fun main()` function uses `cookies.forEach` to print "Menu item: \$it.name". The Run console shows the output: 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.

```
val cookies = listOf()
Cookie {
    name = "Sugar and Sprinkles",
    softBaked = false,
    hasFilling = false,
    price = 1.39
}
fun main() { new *
    cookies.forEach {
        println("Menu item: $it.name")
    }
}
```



The screenshot shows the IntelliJ IDEA IDE with a Kotlin project named 'KotlinListasyMateriales'. The project structure on the left includes 'src' with files 'Colecciones.kt', 'FuncionesOrdenSuperiorColecc.kt', 'Main.kt', and 'ParametroObjeto.kt'. The 'FuncionesOrdenSuperiorColecc.kt' file is open in the editor, showing a list of cookies and a main function that prints each item's name. The Run window at the bottom displays the output: 'Menu item: Vanilla Creme', 'Menu item: Chocolate Peanut Butter', 'Menu item: Snickerdoodle', 'Menu item: Blueberry Tart', and 'Menu item: Sugar and Sprinkles'.

```
8 val cookies = listOf(
44 ),
45 Cookie(
46     name = "Sugar and Sprinkles",
47     softBaked = false,
48     hasFilling = false,
49     price = 1.39
50 )
51 )
52 }
53 fun main() { new *
54     cookies.forEach {
55         println("Menu item: ${it.name}")
56     }
57 }
```

Run: FuncionesOrdenSuperiorColeccKt

Menu item: Vanilla Creme
Menu item: Chocolate Peanut Butter
Menu item: Snickerdoodle
Menu item: Blueberry Tart
Menu item: Sugar and Sprinkles

The screenshot shows the same IntelliJ IDEA IDE with the 'FuncionesOrdenSuperiorColecc.kt' file open. The code has been updated to use a map function to create a full menu string and then print it. The Run window now shows the full menu items with their prices: 'Vanilla Creme - \$1.59', 'Chocolate Peanut Butter - \$1.49', 'Snickerdoodle - \$1.39', 'Blueberry Tart - \$1.79', and 'Sugar and Sprinkles - \$1.39'.

```
71 val cookies = listOf(
186     price = 1.79
187 ),
188 Cookie(
189     name = "Sugar and Sprinkles",
190     softBaked = false,
191     hasFilling = false,
192     price = 1.39
193 )
194 )
195 }
196 fun main() { new *
197     val fullMenu = cookies.map {
198         "${it.name} - ${it.price}"
199     }
200     println("Full menu:")
201     fullMenu.forEach {
202         println(it)
203     }
204 }
```

Run: FuncionesOrdenSuperiorColeccKt

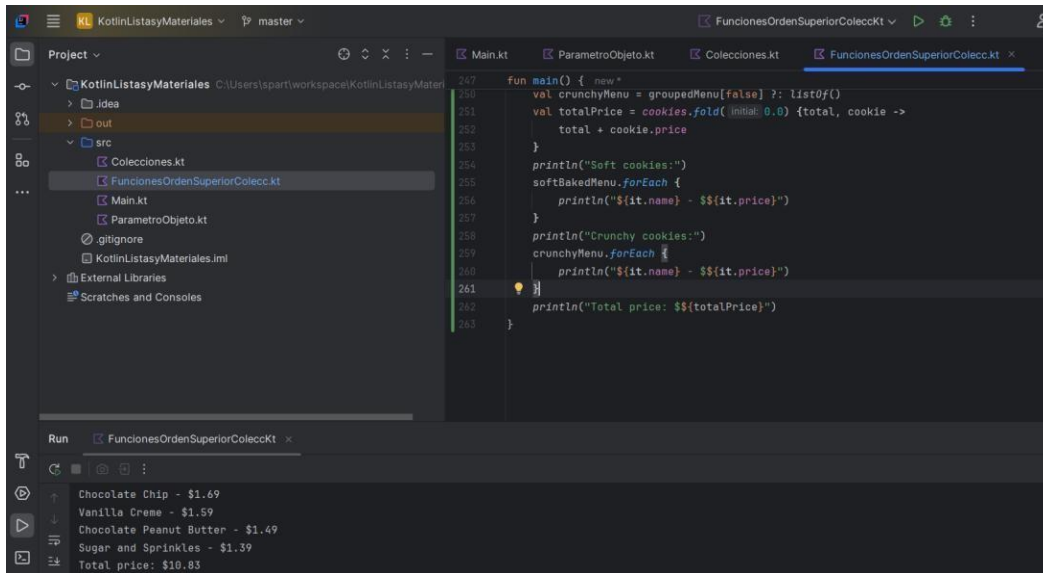
Vanilla Creme - \$1.59
Chocolate Peanut Butter - \$1.49
Snickerdoodle - \$1.39
Blueberry Tart - \$1.79
Sugar and Sprinkles - \$1.39

The screenshot shows the same IntelliJ IDEA IDE with the 'FuncionesOrdenSuperiorColecc.kt' file open. The code has been updated to use a filter function to create a soft baked menu and then print it. The Run window now shows the soft baked menu items with their prices: 'Soft cookies:', 'Banana Walnut - \$1.49', 'Snickerdoodle - \$1.39', and 'Blueberry Tart - \$1.79'.

```
156 val cookies = listOf(
176     hasFilling = false,
177     price = 1.39
178 )
179 )
180 }
181 fun main() { new *
182     val softBakedMenu = cookies.filter {
183         it.softBaked
184     }
185     println("Soft cookies:")
186     softBakedMenu.forEach {
187         println("${it.name} - ${it.price}")
188     }
189 }
```

Run: FuncionesOrdenSuperiorColeccKt

Soft cookies:
Banana Walnut - \$1.49
Snickerdoodle - \$1.39
Blueberry Tart - \$1.79



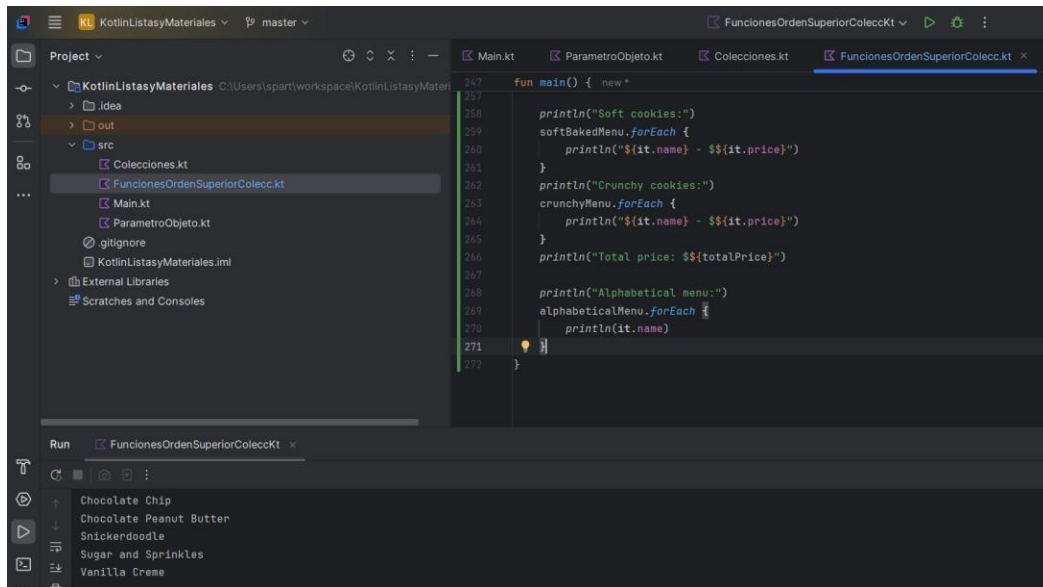
```

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

```

Run FuncionesOrdenSuperiorColeccKt

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



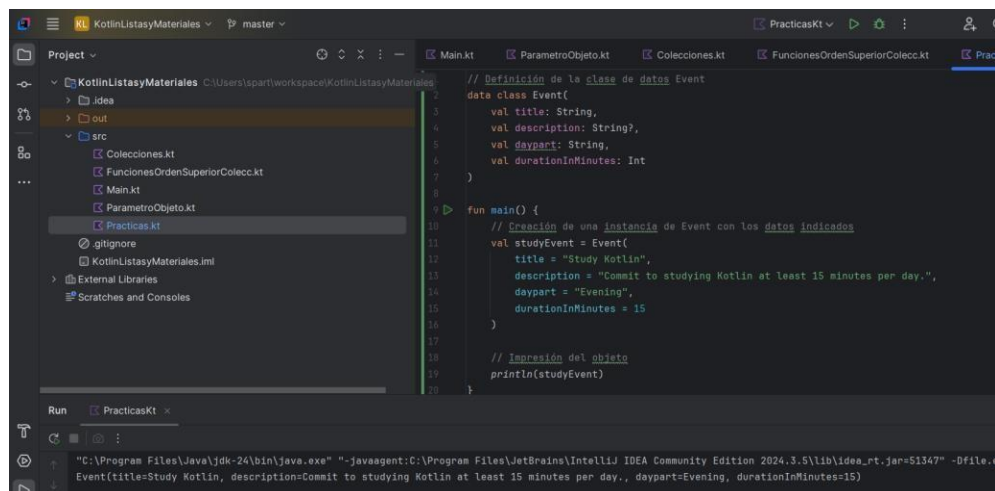
```

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

```

Run FuncionesOrdenSuperiorColeccKt

Chocolate Chip
Chocolate Peanut Butter
Snickerdoodle
Sugar and Sprinkles
Vanilla Creme



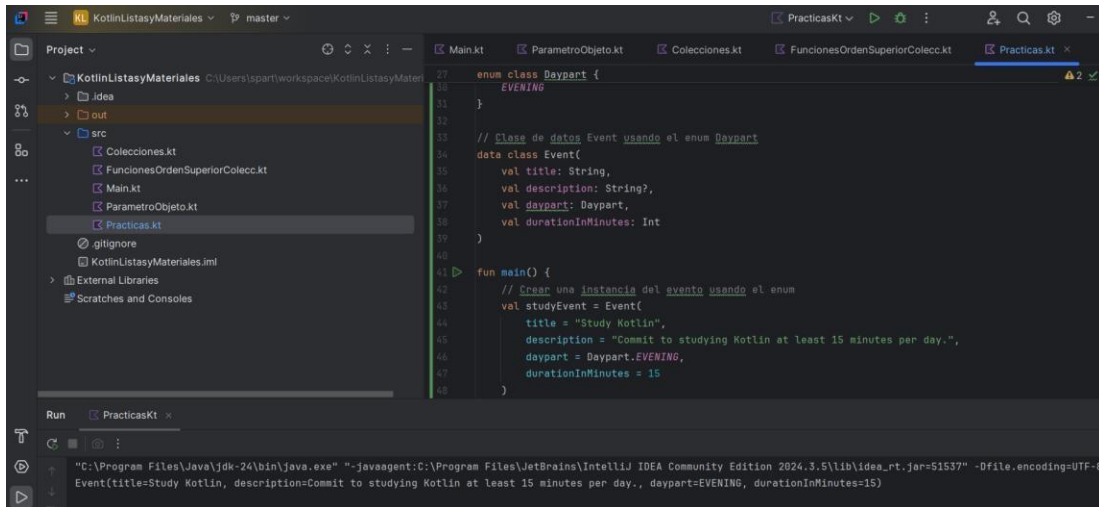
```

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 }

```

Run PracticasKt

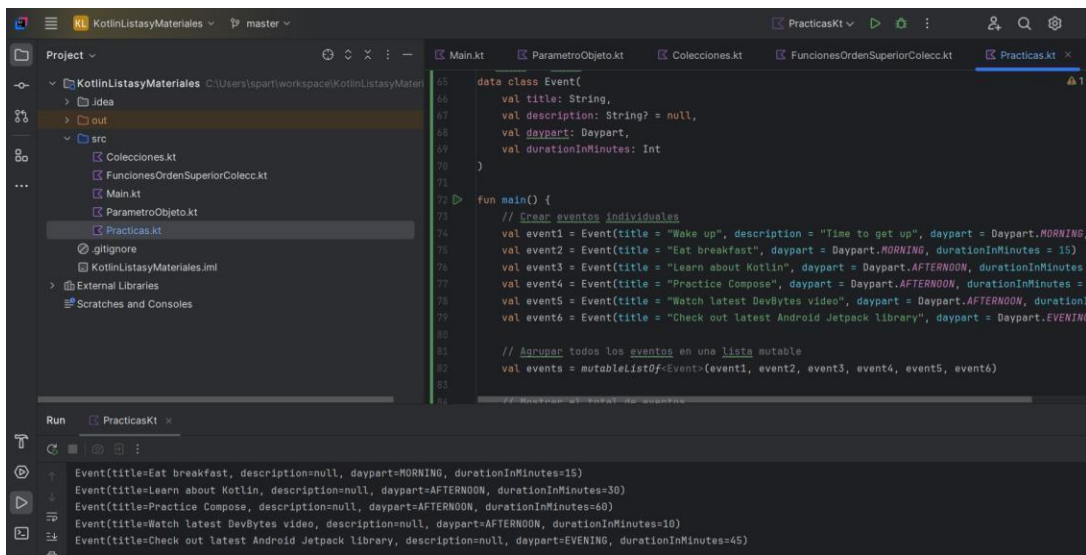
Event(title=Study Kotlin, description=Commit to studying Kotlin at least 15 minutes per day., daypart=Evening, durationInMinutes=15)



```
enum class Daypart {  
    MORNING,  
    AFTERNOON,  
    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

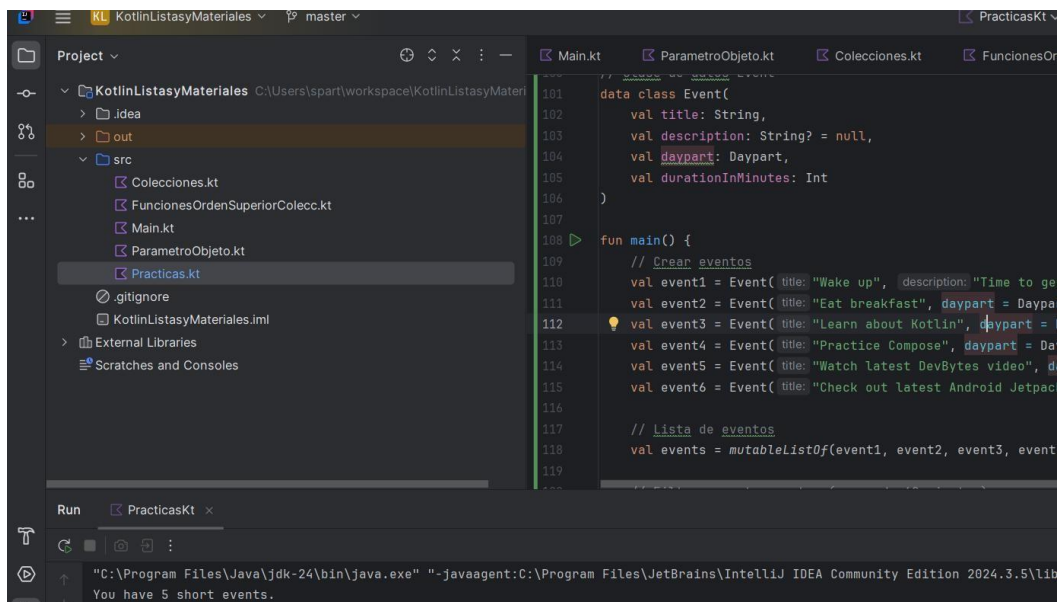
"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\idea_rt.jar=51537" -Dfile.encoding=UTF-8
Event(title=Study Kotlin, description=Commit to studying Kotlin at least 15 minutes per day., daypart=EVENING, durationInMinutes=15)



```
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, durationInMinutes = 15)  
    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, durationInMinutes = 15)  
    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(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, e
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.si
165 }
166 }
167 }
```

Run PracticasKt

"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.5\lib\ide

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

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

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

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", durationInM
200         Event(title="Team Standup", time="10:00 AM", durationInMinutes
201         Event(title="Lunch with Client", time="12:30 PM", durationInM
202         Event(title="Project Presentation", time="03:00 PM", duratio
203         Event(title="Wrap-up and Review", time="05:00 PM", durationIn
204     )
205
206     println("Duration of first event of the day: ${events[0].durat
207 }
208
209 }
```

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.

Duration of first event of the day: short

Crea una lista desplazable

```
// 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,
            )
        }
    }
}

```

```

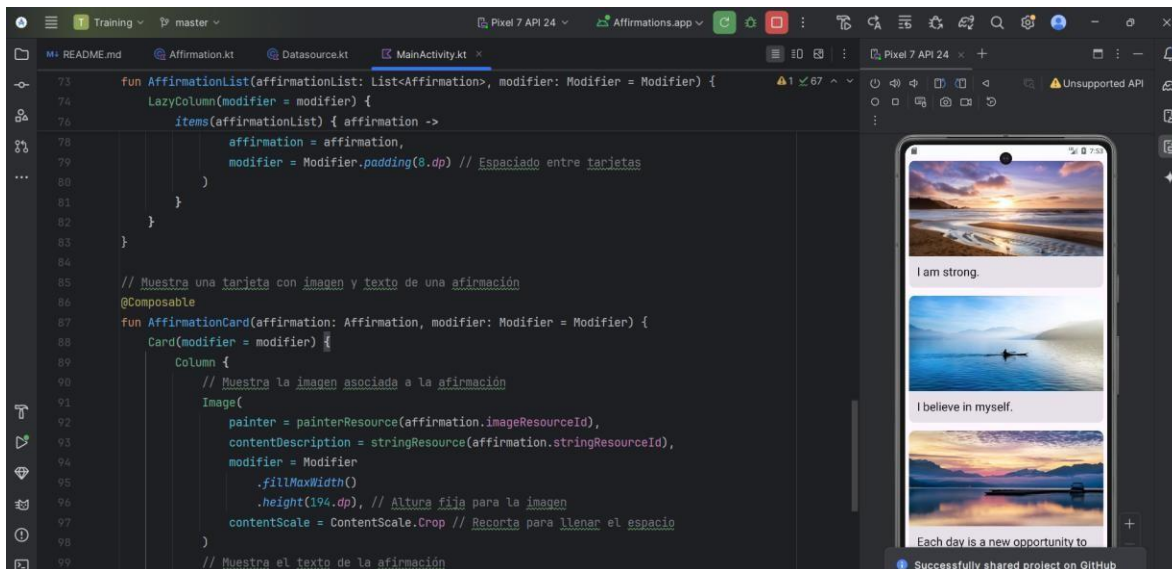
        modifier = Modifier.padding(8.dp) // Espaciado entre tarjetas
    )
}
}
}

// 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)
        )
    }
}

```

```
}
}
```



Esta aplicación Android, desarrollada con **Jetpack Compose**, presenta una **galería de afirmaciones motivacionales**, cada una acompañada por una imagen representativa. Desde el inicio, la app aplica un **tema visual personalizado** que define la estética general de la interfaz, incluyendo colores, tipografía y estilos.

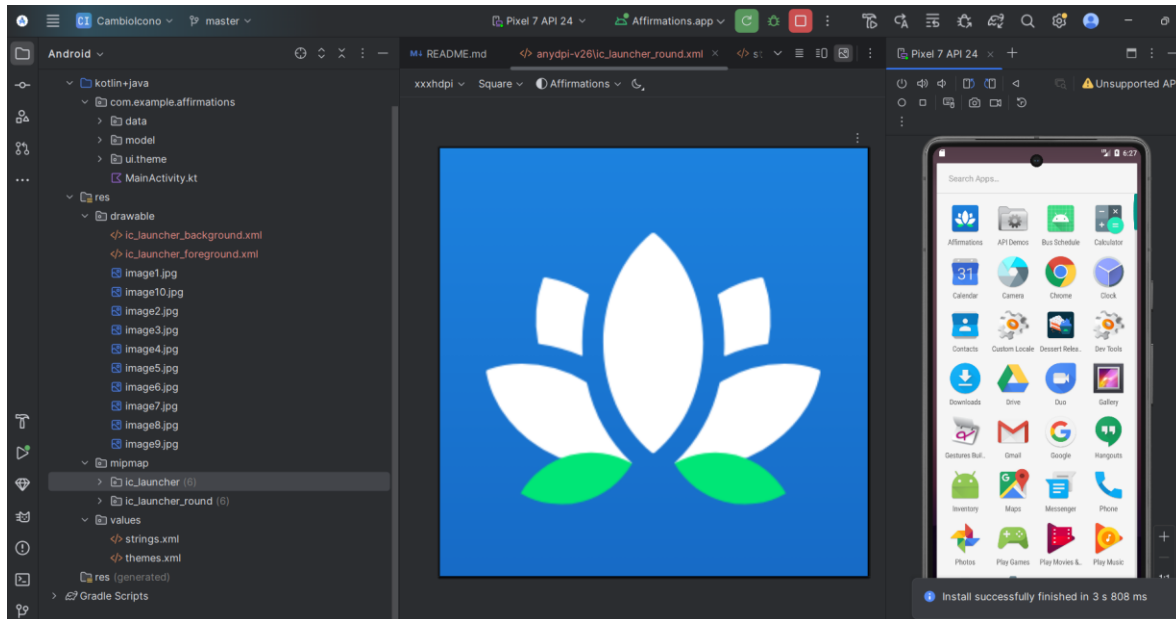
El contenido se genera dinámicamente a partir de una lista de objetos **Affirmation**, suministrados por una clase **Datasource**, lo que permite una separación clara entre los datos y la lógica de presentación. Cada afirmación se renderiza dentro de un componente **Card**, el cual encapsula tanto una imagen como el texto motivacional asociado. Ambos elementos se obtienen directamente de los recursos del proyecto (res/drawable e res/strings), asegurando una gestión centralizada y eficiente del contenido.

La interfaz utiliza una **LazyColumn**, un componente optimizado para listas extensas, que permite renderizar solo los elementos visibles en pantalla, mejorando el rendimiento y reduciendo el consumo de memoria. Además, la aplicación respeta los **márgenes seguros del sistema** (como la barra de estado y la barra de navegación) mediante el uso de **Modifier.padding(WindowInsets)**, garantizando una visualización adecuada en todo tipo de dispositivos.

Durante el desarrollo, los desarrolladores pueden beneficiarse de la anotación **@Preview**, que permite visualizar composables individualmente en tiempo real desde Android Studio, sin necesidad de compilar o ejecutar la aplicación, acelerando así el ciclo de diseño e iteración.

Cambio de iconos

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



Este proyecto consiste en una aplicación Android desarrollada con **Jetpack Compose**, en la cual se implementa la funcionalidad de **cambio dinámico del ícono de la aplicación en tiempo de ejecución**. El objetivo principal fue explorar y demostrar cómo gestionar programáticamente la modificación del ícono principal de la app, permitiendo al usuario seleccionar entre distintas opciones disponibles desde la propia interfaz.

La implementación hace uso de componentes modernos de la arquitectura de Android, y se apoya en técnicas avanzadas para manipular los alias de actividad definidos en el manifiesto (AndroidManifest.xml), habilitando o deshabilitando dinámicamente componentes a través del PackageManager. Esta funcionalidad se integra de manera fluida en una interfaz desarrollada con Jetpack Compose, lo que permite ofrecer una experiencia interactiva, eficiente y alineada con las mejores prácticas de diseño moderno.

Este proyecto ejemplifica una aplicación práctica de conceptos avanzados en la personalización del comportamiento del sistema Android, dentro del contexto de una arquitectura declarativa y basada en estado.

Cuadricula

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

```
/*
 * 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
                )
            }

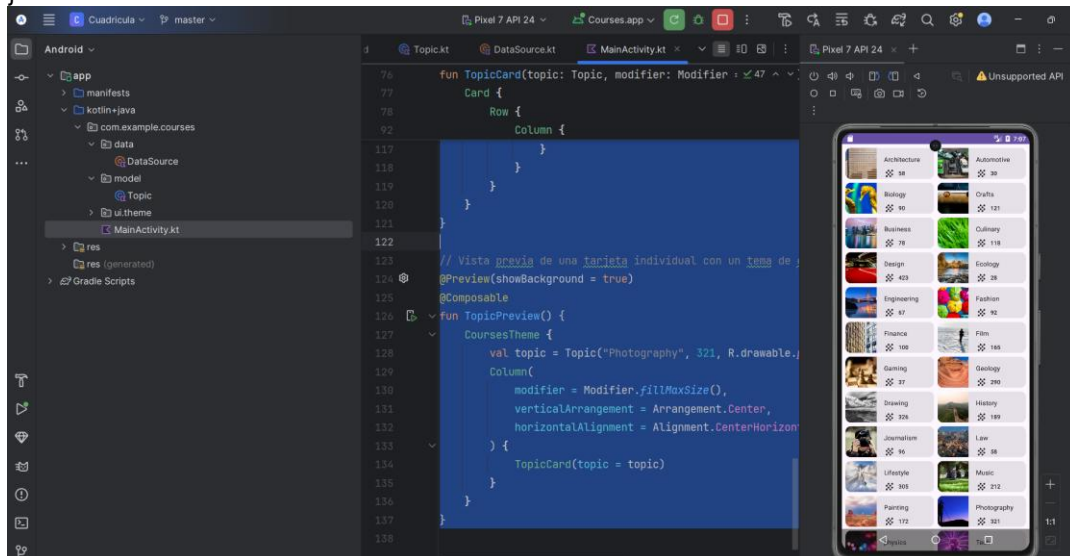
            // Columna con el nombre del tema y número de cursos
            Column {
                // Nombre del tema
                Text(
                    text = stringResource(id = topic.name),
                    style = MaterialTheme.typography.bodyMedium,
                    modifier = Modifier.padding(
                        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 *composable* que organiza una colección de tarjetas temáticas en una disposición de tipo LazyVerticalGrid, configurada con dos columnas y espaciado uniforme entre los elementos. Cada tarjeta dentro de la grilla representa un tema específico, generado dinámicamente a partir de los datos definidos en el objeto DataSource.

El componente TopicCard define la estructura visual de cada tarjeta. Utiliza una disposición en fila (Row) en la que se presenta, dentro de un Box, una imagen representativa del tema, seguida de una columna que muestra el nombre del tema y la cantidad de cursos asociados. Este diseño está optimizado para dispositivos móviles, asegurando una distribución clara, legible y estéticamente equilibrada del contenido.

Además, la función TopicPreview permite visualizar individualmente una tarjeta durante el proceso de desarrollo, facilitando la verificación del diseño y la consistencia visual sin necesidad de ejecutar la aplicación completa. Esta capacidad de previsualización contribuye a un flujo de trabajo más eficiente en la construcción de interfaces con Jetpack Compose.

Compila apps fabulosas

Woof

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

/*

* 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.*

*/

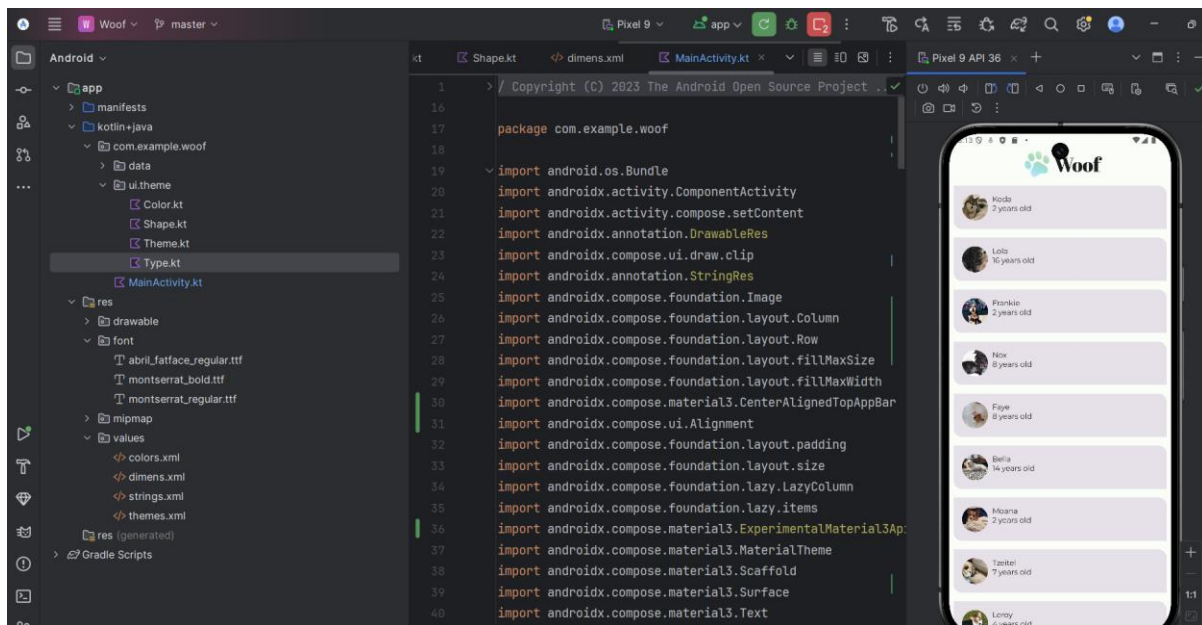
@Preview

@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/Zafirows/Programacion-nativa>

// 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.DampingRatioNoBouncy,  
                    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 DogIconButton(
    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))
    )
}
```

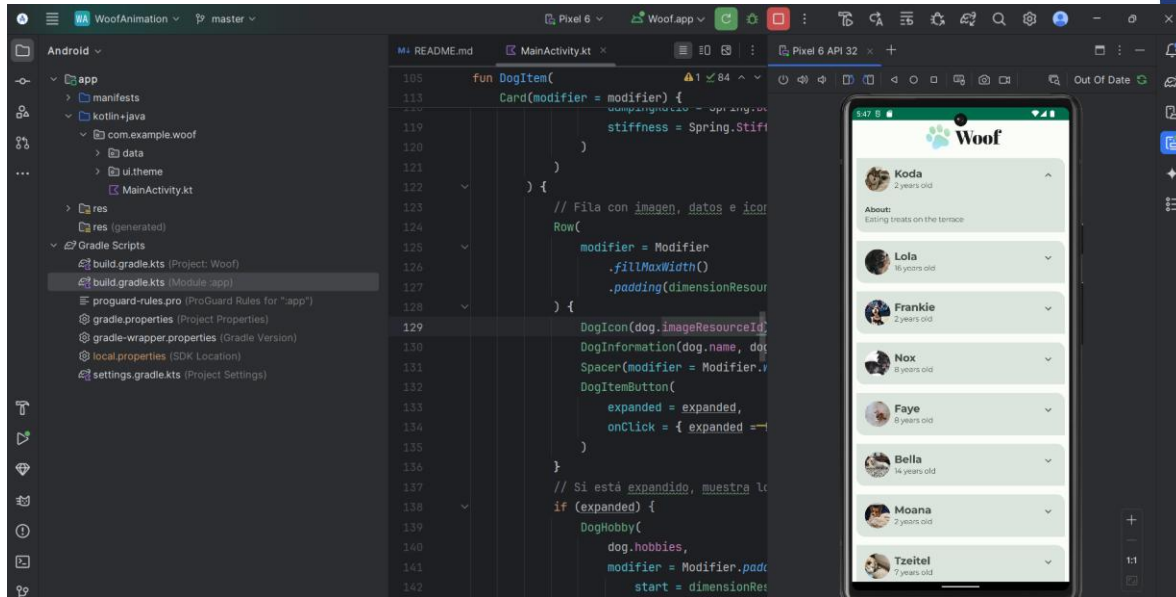


```
.clip(MaterialTheme.shapes.small), // forma redondeada
contentScale = ContentScale.Crop,
painter = painterResource(dogIcon),
contentDescription = null // decorativa
)
}

// 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 implementa una aplicación Android utilizando **Jetpack Compose**, cuyo propósito es presentar una lista de perros con información relevante como su nombre, edad y pasatiempos. La interfaz está estructurada mediante el componente Scaffold, e incorpora una **barra superior personalizada** que enmarca visualmente la experiencia de usuario.

La lista de perros se presenta mediante un componente desplazable verticalmente, en el que cada elemento se encapsula dentro de una **tarjeta expandible**. Estas tarjetas permiten mostrar u ocultar información adicional de forma interactiva, mejorando la organización del contenido sin comprometer la claridad visual. Cada tarjeta carga dinámicamente la imagen, el nombre y la edad del perro a partir de los recursos definidos en el proyecto.

Además, se incluyen funciones de vista previa adaptadas tanto al **modo claro como al modo oscuro**, lo que facilita el diseño visual responsivo directamente desde el entorno de desarrollo Android Studio. En conjunto, esta aplicación demuestra el uso efectivo de componentes modernos de Compose, manejo dinámico de datos y principios de diseño adaptativo.

SuperHeroes

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

```
/*
 * 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.

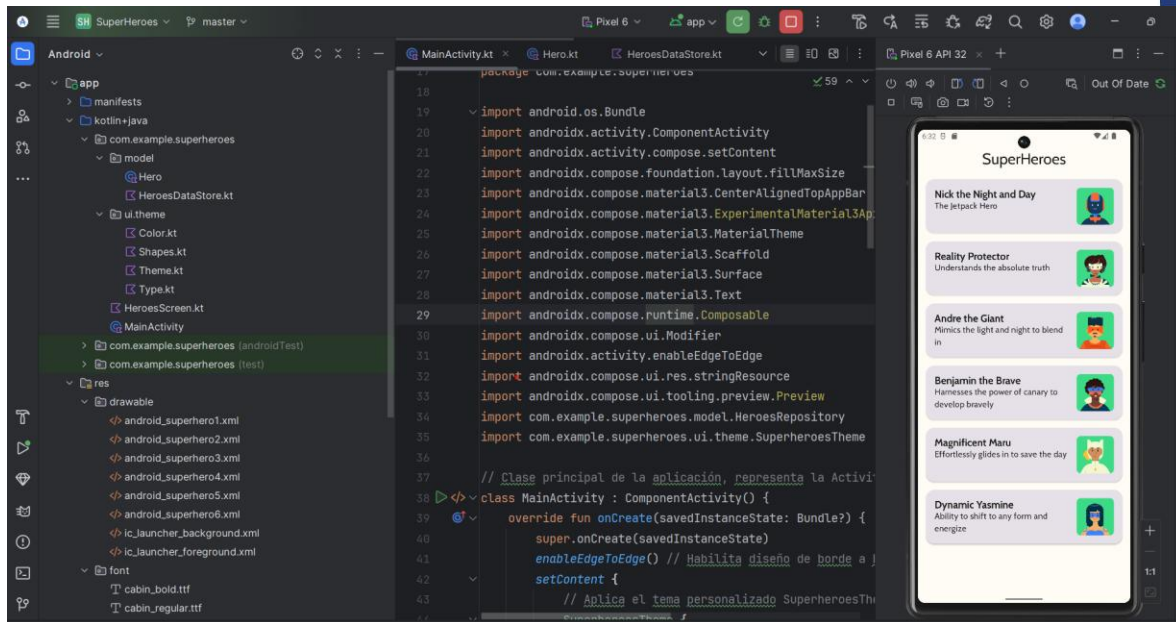
```
/**
 * Composable principal que muestra la estructura general de la app: una TopAppBar y una
lista de héroes.
 */
@Composable
```



```
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 denominada **SuperHeroes**, desarrollada con **Jetpack Compose**. La clase **MainActivity**, que extiende de **ComponentActivity**, configura la interfaz de usuario en el método **onCreate()**, donde se establece el tema **SuperheroesTheme** y se habilita el modo **edge-to-edge** mediante la función **enableEdgeToEdge()**, permitiendo que el contenido utilice toda la superficie de la pantalla.

La estructura visual se construye sobre un contenedor **Surface** que actúa como fondo de la interfaz, dentro del cual se invoca la función composible principal **SuperheroesApp()**. Esta, a su vez, implementa un diseño basado en **Scaffold**, que incluye una **barra superior personalizada** (**TopAppBar**) y una **lista de héroes** (**HeroesList**) obtenida directamente desde un repositorio de datos llamado **HeroesRepository**.

La barra superior muestra el nombre de la aplicación centrado, utilizando la tipografía definida por el tema actual. Para facilitar el proceso de diseño y revisión, se incluye una función de vista previa (**SuperHeroesPreview()**), que permite renderizar la interfaz directamente en Android Studio sin necesidad de ejecutar la aplicación completa.

Cabe destacar dos observaciones importantes en cuanto a la arquitectura del código:

1. **Organización de funciones composables:** Actualmente, las funciones composables están definidas dentro de la clase **MainActivity**, lo cual no se considera una buena práctica en Jetpack Compose. Se recomienda moverlas fuera de la clase para mantener una estructura de código modular y mantenible.
2. **Acceso al repositorio desde la UI:** La obtención directa de datos desde **HeroesRepository** en la capa de interfaz no sigue los principios recomendados de separación de responsabilidades. A futuro, se sugiere utilizar un **ViewModel** como intermediario para gestionar el estado y el acceso a los datos, siguiendo la arquitectura recomendada por Android para aplicaciones modernas.