Primeros pasos con Kotlin: Tu primer «Hola mundo», variables y plantillas de texto

1. Introducción
2. El clásico «Hola mundo»
3. Variables: Guardando información en memoria
    3.1. val vs var: ¿Cuál debería usar?
4. Plantillas de cadenas (String Templates): Texto inteligente
    4.1. La magia oculta: La inferencia de tipos
5. Ejercicios
    5.1. Presentando a Mary
    5.2. Batería del móvil
    5.3. Calculadora de la compra inteligente
    5.4. ¿print o println?
6. Soluciones a los ejercicios
    6.1. Presentando a Mary
    6.2. Batería del móvil
    6.3. Calculadora de la compra inteligente
    6.4. ¿print o println?

1. Introducción

Si estás empezando en el mundo del desarrollo (ya sea para Android o backend), Kotlin es uno de los lenguajes más modernos, limpios y seguros que puedes aprender hoy en día.

En esta unidad, vamos a dar los primeros pasos. Entenderemos cómo funciona la estructura básica de un programa, cómo almacenar información usando variables y cómo imprimir mensajes por pantalla de forma elegante. Al final, tendrás **ejercicios prácticos** para asentar lo aprendido.

2. El clásico «Hola mundo»

Por tradición, el primer programa que todo desarrollador escribe al aprender un lenguaje nuevo es el que imprime las palabras «¡Hola, mundo!» en la pantalla.

En Kotlin, se hace así de fácil:

```
1.  fun main() {
2.      println("¡Hola, mundo!")
3.      // ¡Hola, mundo!
4.  }
```

Analizando el código línea a línea:

- `fun` : Es la palabra reservada (abreviatura de *function*) que usamos en Kotlin para declarar una función. Una función no es más que un bloque de código que realiza una tarea específica.
- `main()` : No es una función cualquiera. Es el **punto de entrada** (entry point) de tu programa. Siempre que ejecutes una aplicación en Kotlin, el ordenador buscará esta función y empezará a ejecutar las instrucciones que haya dentro de ella.
- `{ }` (Las llaves): Todo lo que esté dentro de las llaves es el «cuerpo» de la función, es decir, las instrucciones que se van a ejecutar.
- `println()` : Es una función nativa de Kotlin que coge lo que le pongas entre paréntesis (los «argumentos») y lo imprime por la salida estándar (la consola de tu pantalla). Al terminar de imprimir, añade un salto de línea (por eso termina en *ln*, de *line*). Si usas `print()` (sin el *ln*), el texto se imprimirá, pero el siguiente mensaje aparecerá pegado justo a la derecha, en la misma línea.

> *El dato: ¿Te has dado cuenta de algo? ¡No hay punto y coma (;) al final de la línea! En lenguajes más antiguos como Java o C++, olvidar el punto y coma era un dolor de cabeza. Kotlin es un lenguaje moderno y no los necesita.*

3. Variables: Guardando información en memoria

Cualquier programa informático necesita almacenar datos (nombres de usuario, puntuaciones de un juego, precios...). Para eso utilizamos las **variables**.

En Kotlin, tenemos dos formas principales de crear variables, y esto es muy importante:

1. **Variables de solo lectura (inmutables) con** `val` (de *value*).
2. **Variables mutables con** `var` (de *variable*).

Para asignarle un valor a una variable, simplemente usamos el operador de asignación `=`. Fíjate en este ejemplo:

```
1.  fun main() {
2.      val palomitas = 5   // Tenemos 5 cajas de palomitas
3.      val perritos = 7    // Tenemos 7 perritos calientes
4.      var clientes = 10   // Hay 10 clientes en la cola
5.
6.      // De repente, un par de clientes se cansan de esperar y se van
    de la cola
7.      clientes = 8
8.      println(clientes)
9.      // Resultado en pantalla: 8
10. }
```

3.1. `val` vs `var`: ¿Cuál debería usar?

Como `clientes` se declaró con `var`, pudimos reasignarle un nuevo valor (`8`) más adelante en el programa. Sin embargo, si intentáramos hacer `palomitas = 6`, el compilador de Kotlin nos daría un error, porque `palomitas` es un `val` y **no puede cambiar una vez se le ha dado un valor inicial**.

> *La regla de oro en Kotlin: Utiliza siempre `val` por defecto. Usa `var` única y exclusivamente cuando sepas seguro que ese valor va a tener que cambiar en el futuro (como un contador, o la vida de un personaje en un juego). Esto hace que tu código sea mucho más seguro, predecible y libre de errores accidentales (los temidos* bugs*).*

4. Plantillas de cadenas (String Templates): Texto inteligente

En programación, es súper común querer imprimir un texto que contenga el valor de nuestras variables. En otros lenguajes tendrías que «sumar» o concatenar trozos de texto con el símbolo `+`, lo cual queda feo y es fácil equivocarse.

Kotlin lo soluciona de forma magistral con las **String Templates** (plantillas de cadenas).

Una cadena de texto (String) se escribe siempre entre comillas dobles `" "`. Para inyectar el valor de una variable dentro de ese texto, **solo tienes que poner el símbolo del dólar `$` seguido del nombre de la variable**.

Y si lo que quieres es hacer una operación matemática u otro código más complejo dentro del texto, lo metes entre llaves `${ }`. Mira por ejemplo el siguiente código:

```
1.  fun main() {
2.      val clientes = 10
3.
4.      // Imprimiendo una variable directamente
5.      println("Actualmente hay $clientes clientes esperando.")
6.      // Salida: Actualmente hay 10 clientes esperando.
7.
8.      // Haciendo una operación matemática dentro del texto
9.      println("Si llega uno más, habrá ${clientes + 1} clientes en
    total.")
10.     // Salida: Si llega uno más, habrá 11 clientes en total.
11. }
```

4.1. La magia oculta: La inferencia de tipos

Si prestas atención, en ningún momento le hemos dicho a Kotlin que `clientes` es un número. Kotlin es muy listo: al ver que le asignamos un `10`, él automáticamente infiere (deduce) que el tipo de dato es un número entero (`Int`).

5. Ejercicios

La programación solo se aprende tecleando. Aquí tienes varios ejercicios de menos a más dificultad. Intenta resolverlos por tu cuenta en un editor o en el Kotlin Playground antes de mirar las soluciones más abajo.

5.1. Presentando a Mary

Completa el siguiente código para que el programa imprima por consola el mensaje exacto: `"Mary tiene 20 años"`. Debes usar obligatoriamente las variables dadas y las plantillas de cadenas (`$`).

```
fun main() {
    val nombre = "Mary"
    val edad = 20
    // Escribe tu código debajo de esta línea:

}
```

5.2. Batería del móvil

Crea un pequeño programa que simule la batería de tu móvil.

1. Declara una variable llamada `bateria` que empiece al 100%. (Piensa si debe ser `val` o `var`).
2. Imprime el texto: `"La batería inicial es del 100%"`.
3. Simula que juegas a un videojuego y la batería baja a 75. Actualiza el valor de la variable.
4. Vuelve a imprimir el texto: `"Tras jugar, la batería es del 75%"`.

5.3. Calculadora de la compra inteligente

Declara dos variables **inmutables**: `precioZapatillas` con un valor de 50, y `cantidadCajas` con un valor de 2.

Utilizando un único `println()` y las llaves de expresión `${}`, imprime el siguiente mensaje calculando el total sobre la marcha:

```
"Has comprado 2 pares de zapatillas. El precio total es de 100 euros."
```

5.4. ¿`print` o `println`?

Imagina que quieres que la consola muestre **exactamente** esto en dos líneas:

```
Hola Mundo
¡Amo Kotlin!
```

Escribe el código usando tres instrucciones de impresión para conseguir esa salida exacta. Haz que la palabra «Hola» y «Mundo» se impriman en instrucciones separadas pero se queden en la misma línea.

6. Soluciones a los ejercicios

¡No hagas trampas! Solo mira esto si ya has intentado resolverlos.

6.1. Presentando a Mary

```
fun main() {
    val nombre = "Mary"
    val edad = 20
    println("$nombre tiene $edad años")
}
```

6.2. Batería del móvil

```
fun main() {
    // Usamos 'var' porque la batería va a cambiar a lo largo del tiempo
    var bateria = 100
    println("La batería inicial es del $bateria%")

    bateria = 75
```

```
    println("Tras jugar, la batería es del $bateria%")
}
```

6.3. Calculadora de la compra inteligente

```
fun main() {
    val precioZapatillas = 50
    val cantidadCajas = 2

    // Usamos ${} para multiplicar variables dentro del mismo texto
    println("Has comprado $cantidadCajas pares de zapatillas. El precio total es de ${precioZapatillas * cantidadCajas} euros.")
}
```

6.4. ¿`print` o `println`?

```
fun main() {
    print("Hola ")       // Usa print, por lo que la siguiente palabra se pega a esta. (Fíjate en el espacio al final)
    println("Mundo")     // Se pega al 'Hola ', y luego hace un salto de línea por el 'ln'
    println("¡Amo Kotlin!") // Se imprime en la nueva línea
}
```