



Curso de Kotlin Desde Cero

Giuseppe Vetri

The background is a deep purple with various abstract elements. There are large, dark purple organic shapes. Several thin, light purple lines are scattered across the frame. Three solid purple circles of different sizes are visible. On the right side, there are two circular patterns with horizontal stripes, one partially visible at the top and one at the bottom right.

Bienvenido al curso

¿Qué es Kotlin?



**Kotlin es desarrollado
por JetBrains**

JETBRAINS

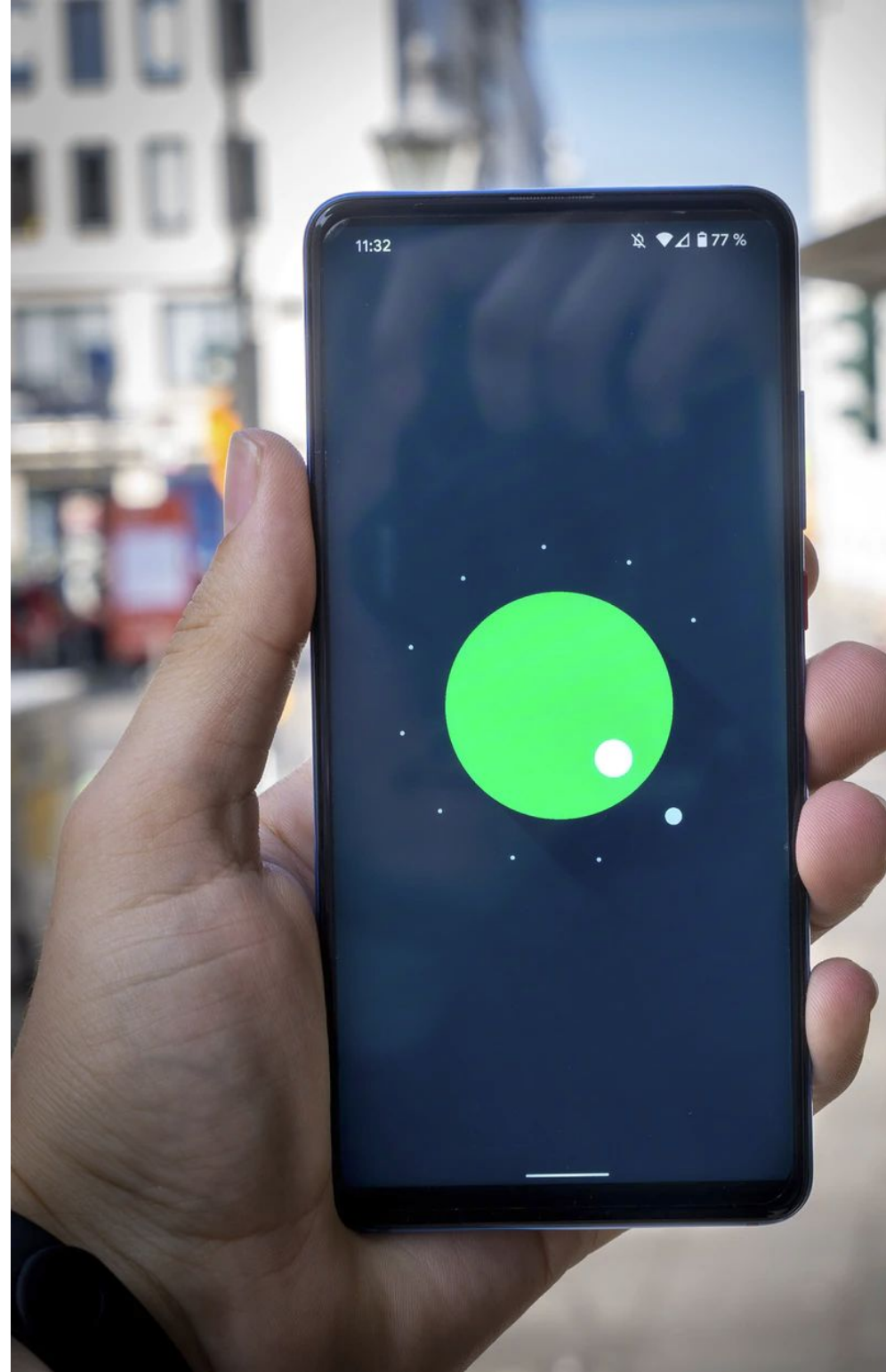
The drive to develop

**JET
BRAINS**



Nació en el 2010

Sin embargo,
es bastante maduro
y la base del desarrollo
Android hoy en día.





No solo se usa en Android

- **Backend development**

Con frameworks como Spring, Micronaut o Ktor.

- **De forma nativa**

Puede correr en Windows y otras plataformas de forma nativa.



Y aún hay más

- **Scripts**

Permite crear scripts que pueden ser ejecutados en cualquier entorno.

- **Multiplatform**

Permite crear lógica de negocio que se comparte entre apps de Android e iOS.



Surge de la necesidad de Java

Era necesario una alternativa a Java.
Con sintaxis moderna, nuevas
funcionalidades y capaz de ser
interoperable con otros lenguajes.



La adopción en Android

Es el lenguaje recomendado para desarrollar apps de Android y esto se debe a las limitantes que tiene Java.

Qué es la Java Virtual Machine

Instalación de IntelliJ

Cómo instalar el SDK de Java

Kotlin con Visual Studio Code

The background is a solid dark purple color. It features several abstract geometric elements: a large, dark purple, irregular blob shape in the upper center; a smaller, lighter purple circle in the top right; a circle with horizontal grey stripes in the bottom right; and several thin, light purple diagonal lines in the lower left.

Hola mundo con Kotlin

Variables en Kotlin

Kotlin y sus tipos de variables

Modificadores y tipos de datos en Kotlin

¿Qué es un tipo de dato?

Los enteros, las cadenas de texto, los booleanos. El tipo de valor que van a tener nuestras variables.



¿Qué es un tipo de dato primitivo?

Son los tipos de datos originales de un lenguaje de programación.



¿Qué es un objeto?

Un objeto es una combinación de variables, funciones y otros objetos.



En Kotlin todo es un objeto

Kotlin trae por defecto objetos, que parecen primitivos pero no lo son.

String, Int, Boolean.



Ventajas de tratar como objetos

Posibilidad de crear funciones
específicas para ese tipo de objeto.



Ventajas de tratar como objetos

Poder sobrescribir operadores
como la suma para sumar dos objetos
del mismo tipo.



Ventajas de tratar como objetos

Extender del lenguaje para crear nuevas funciones que permitan a tu equipo evitar repetir código y mantener una base de código saludable.



Retrocompatibilidad con Java

Debido a que Kotlin tiene que compilar el código que nosotros escribimos y hacerlo interoperable con Java. Debe seguir algunas reglas.



Retrocompatibilidad con Java

Un entero que puede ser nulo,
no se convertirá a primitivo. Pero
un entero que no puede ser nulo,
se convertirá a primitivo.



Kotlin y la programación funcional



Paradigma de programación

Existen varias formas de escribir código y estas formas son llamadas:

Paradigmas



Paradigma imperativo

Este paradigma se basa en modificar el estado de tu programa modificando estados dentro del mismo. Se centra en describir cómo funciona un programa.



Programación funcional

- Es un paradigma de programación declarativo.
- Expresa la lógica de un programa sin describir lo que hace.
- Se enfoca en lo que el programa debe hacer no en cómo lo hace.

Kotlin

Programación funcional

No es un lenguaje funcional 100% como Haskell o Scala. Pero tiene varios conceptos que nos ayudarán a sacarle mayor provecho a Kotlin.

Nunca mutable siempre inmutable

Un elemento es mutable cuando puede cambiar, inmutable cuando no.

Es recomendable usar variables de solo lectura y estructuras de datos no mutables.



Las funciones son objetos

Las funciones pueden almacenarse en variables, pasarse como parámetros y tratarse como cualquier otro objeto.



Usa funciones puras

Recibe siempre los mismos parámetros y devuelve siempre el mismo resultado.

No puede verse afectada por elementos fuera de su entorno.

Estructuras de control: if

Estructuras de Control: when

Bucles: While y Do While

Ciclos

The background is a deep purple with several large, overlapping, wavy organic shapes in a slightly darker shade. There are three solid purple circles: one in the top right, one on the left edge, and one in the bottom right. A circular area in the bottom right contains horizontal grey stripes. Several thin, light purple lines are scattered across the background, some parallel and some intersecting.

Null Safety en Kotlin

Tipos

Hemos venido hablando de
int, string, boolean.





Tipos

Si tenemos un boolean, tenemos dos posibles valores. **true o false.**
Pero qué ocurre si te digo que puede existir un **tercer valor.**



La nada o el null

Existe un elemento llamado **null** que indica que el contenido de nuestra variable no existe, que está apuntando a una referencia de memoria que no existe.

El origen del boolean de tres valores

Un booleano puede tener solo dos valores `true` o `false`. Pero un boolean nullable aquel que puede ser nulo, tiene tres, **`true`**, **`false`** o **`null`**.

El origen del boolean de tres valores

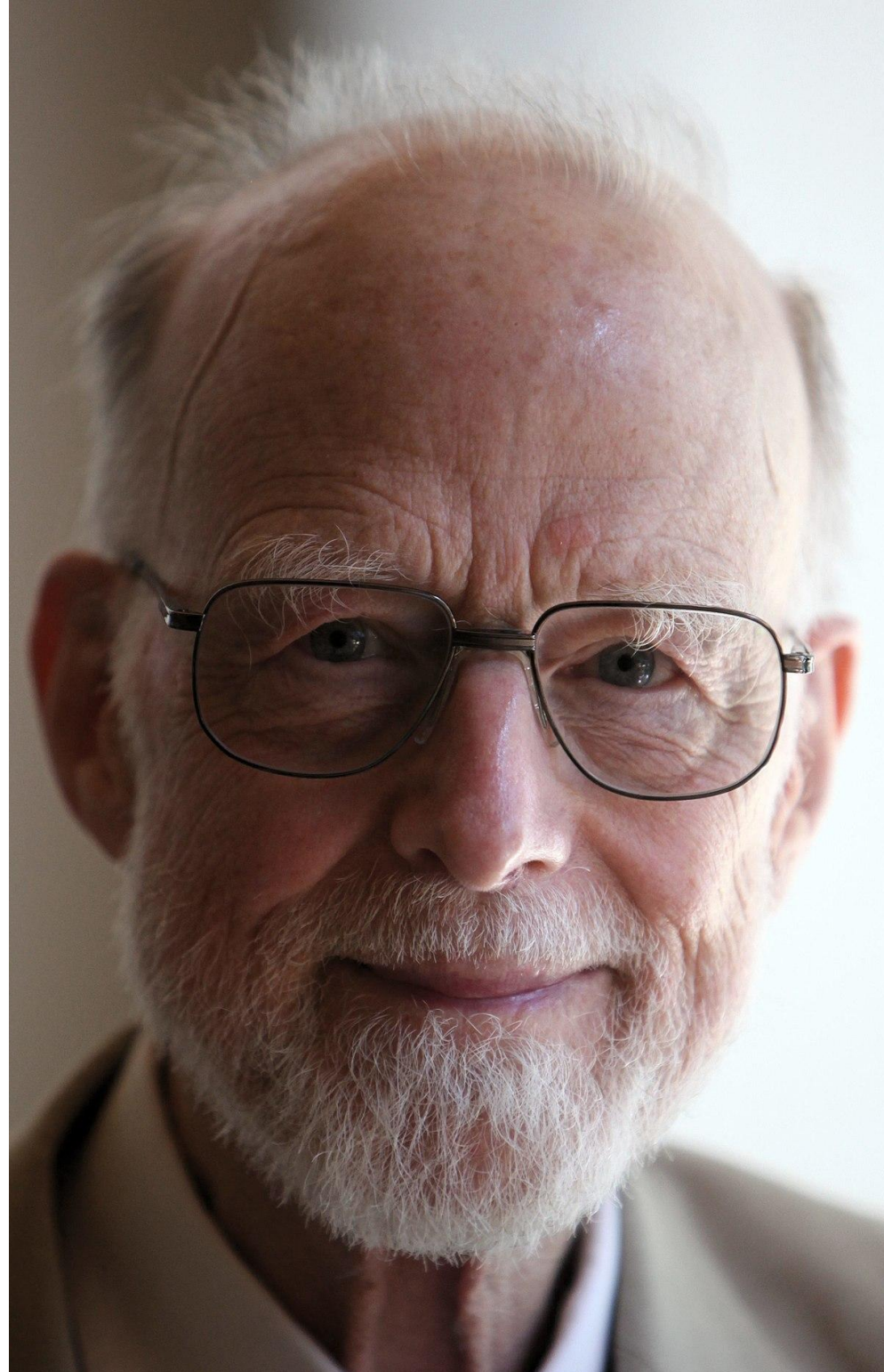
Este animal para nada mitológico suele vivir en código escrito en Java, en la respuesta de servidores y en código con malas prácticas.



0 vs NULL

El origen de la nada

La referencia nula
o null pointer.
Fue creada por
Sir Tony Hoare en 1965.



El error de los mil millones de dólares

En 2009 contó en una charla que fue un error haber creado el **null** y que fue una solución rápida que no evolucionó muy bien.



¿Cómo nos ayuda Kotlin?

Kotlin da herramientas para detectar cuando una variable es nula, agregando el tipo nullable. Con esto el compilador es capaz de advertirnos que parte puede fallar debido a una referencia nula.

**Valores nulos,
double bang y
cómo solucionarlos.**

Los null no son malos, son incomprendidos

Como toda herramienta puede
ser mal utilizada o bien utilizada.



Como declaramos un tipo que puede ser nulo

Para esto debemos utilizar el símbolo de interrogación después del tipo de dato que queremos utilizar.



Utilizando el null:

```
var segundoNombre : String? = "Antonio"
```



El compilador es tu amigo no tu enemigo

El compilador es capaz de interpretar estos tipos. Y advertirnos de lo que puede ocurrir al correr nuestro programa.



Crea código para otros

Crea código como si tuvieras que trabajar en ello dentro de un año.

Esto te ayudará a pensar en una solución a futuro y no solo en complacer el compilador.



La regla de los Boy Scout

Deja siempre el código mejor
que lo encontraste.



Safe calls

Las safe calls son una herramienta del lenguaje que nos ayudan a ejecutar un código cuando una variable del tipo nullable no es nula.

Utilizando los safe-calls.

```
println(segundoNombre?.length())
```



Double bang

Este operador se indica con dos símbolos de exclamación !!

Con esto le dices al compilador que sabes que en este punto la variable no puede ser nula.

Gran poder, gran responsabilidad

No está bien usarlo siempre
para que tu código compile y ya.
Úsalo solo cuando sea necesario
y sepas que esa variable no será nula
en ese punto.



Interoperabilidad con Java

Al ejecutar código escrito por otras personas en Java desde Kotlin.

Puede que te encuentres con estos tipo de variable: **Integer!**



Interoperabilidad con Java

Los tipos **Integer**! Son la forma que tiene Kotlin de avisarte que no puede asegurarse de que ese código **no** devuelve null. Como recomendación siempre trátalos como nullables.

Try Catch

Elvis operator

The background is a solid dark purple. It features several abstract geometric elements: a large, dark purple, irregular blob shape in the upper center; a smaller, lighter purple circle in the top right; a circle with horizontal grey stripes in the bottom right; and several thin, light purple lines of varying lengths and orientations scattered across the left and bottom-left areas.

The background is a deep purple with several large, overlapping, wavy organic shapes in a slightly darker shade. There are also several thin, light purple diagonal lines scattered across the background. In the top right corner, there is a small, solid purple circle. In the bottom right corner, there is a larger, solid purple circle. To the right of the bottom right circle, there is a circular area with horizontal grey stripes. In the top left corner, there are some faint, light purple geometric shapes, possibly rectangles or trapezoids.

Listas

Cómo ordenar listas con las funciones que tiene Kotlin

Maps

The background is a deep purple color. It features several abstract geometric elements: a large, dark purple, irregular shape in the upper left; a smaller, lighter purple circle in the upper right; a series of horizontal grey lines on the right side; and several thin, light purple lines in the lower left. A large, dark purple circle is partially visible at the bottom right.

Sets

The background is a deep purple with several large, overlapping, wavy organic shapes in a slightly darker shade. There are three solid purple circles: one in the top right, one on the left edge, and one in the bottom right. A circular area in the bottom right contains horizontal grey stripes. Several thin, light purple lines are scattered across the background, some parallel and some intersecting.

The background is a deep purple color with various abstract geometric elements. There are several overlapping circles and semi-circles in different shades of purple. Some circles have horizontal stripes. There are also several thin, light purple lines scattered across the background. The text is centered in the middle of the image.

¿Qué son las
funciones?



¿Qué es una función?

Una función es un código que se ejecuta cada vez que lo llamamos. Las hemos venido utilizando anteriormente pero ahora vamos a profundizar.

Sintaxis de una función

Las funciones más básicas se componen de 4 partes. Empecemos con el **nombre**.

Toda función empieza con la palabra reservada **fun** y luego el nombre de la función.

Sintaxis de una función

Luego tenemos los **parámetros**, que son las variables que le damos a la función para que las use en el código que tiene dentro.



Sintaxis de una función

Sigue el tipo de retorno.

Es decir el tipo que va a ejecutar una vez ejecutada la función.



Sintaxis de una función

Para terminar tenemos el código que vamos a ejecutar cuando llamemos la función.

Sintaxis de una función

Nombre

Parámetros

Tipo de retorno

```
fun suma(primerValor: Int, segundoValor: Int) : Int {  
    return primerValor + segundoValor  
}
```

Código que
ejecuta

Todas las funciones en Kotlin devuelven un valor

La palabra reservada **return** indica el valor que vamos a retornar.

¿Pero qué ocurre si no queremos devolver nada?

Todas las funciones en Kotlin devuelven un valor

En ese caso la función regresaría Unit.

En este caso Kotlin te recomendará eliminar el tipo de retorno de la función.

Función que devuelve Unit

```
fun imprimirNombre(nombre : String, apellido: String){  
    print("Mi nombre es $nombre y mi apellido es $apellido")  
}
```

Funciones y funciones de extensión

Tipos de parámetros en las funciones

Lambdas

The background is a deep purple color. It features several abstract geometric elements: a large, dark purple, irregular shape in the upper center; a smaller, lighter purple circle in the top right; a circle with horizontal grey stripes in the bottom right; and several thin, light purple lines of varying lengths and orientations scattered across the left and bottom-left areas.

High Order Functions

The background is a solid dark purple. It features several abstract geometric elements: a large, dark purple, irregular shape in the upper left; a smaller, lighter purple circle in the upper right; a series of horizontal white lines forming a semi-circle on the right side; and several thin, light purple lines of varying lengths and orientations scattered across the lower left and center.



Let

The background is a deep purple with several large, dark purple, organic, wavy shapes. There are three solid purple circles: one in the top right, one on the left edge, and one in the bottom right. The bottom right circle is partially overlapping a circular area with horizontal grey stripes. Several thin, light purple lines are scattered across the background, some parallel and some intersecting.

With

The background is a deep purple with several large, dark purple, wavy organic shapes. There are also some thin, light purple lines and a few small, solid purple circles. In the bottom right corner, there is a circular area with horizontal grey stripes. The word "Run" is centered in a white, bold, sans-serif font.

Run


The background is a deep purple with several large, overlapping, wavy organic shapes in a slightly darker shade. There are also several thin, light purple diagonal lines scattered across the left side. On the right side, there are three circular elements: a solid purple circle at the top, a circle with horizontal grey stripes in the middle, and a solid purple circle at the bottom right.

Apply

The background is a deep purple with various abstract elements. There are large, dark purple organic shapes. Several thin, light purple lines are scattered across the frame. Three solid purple circles are visible: one in the top right, one on the left edge, and one in the bottom right. A circular area in the bottom right contains horizontal grey stripes. The word "Also" is centered in a white, bold, sans-serif font.

Also

Declarando las constantes de nuestra bola mágica

The background is a deep purple with various abstract elements. There are several overlapping circles and semi-circles in different shades of purple. Some circles have horizontal stripes. There are also several thin, light purple lines scattered across the background. The text is centered in a large, white, sans-serif font.

**Creando el menú de
nuestra bola mágica**

The background is a deep purple with various abstract elements. There are large, dark purple organic shapes. Several thin, light purple lines are scattered across the frame. Three solid purple circles of different sizes are visible. On the right side, there are two circular patterns with horizontal stripes, one partially visible at the top and one at the bottom right.

**Contestando
aleatoriamente**

The background is a deep purple color. It features several abstract geometric elements: a large, dark purple, irregular shape in the center; a smaller, lighter purple circle in the top right; a circle with horizontal stripes in the bottom right; and several thin, light purple lines of varying lengths and orientations scattered across the left and bottom areas.

**¿Cómo continuar tu
camino en Kotlin?**