**INSTITUTO TECNOLÓGICO DEL VALLE DE OAXACA**

CARRERA: ING. INFORMATICA

SEMESTRE: 8° GRUPO: B

ASIGNATURA: DESARROLLO DE APLICACIONES PARA DISPOSITIVOS MOVILES

NOMBRE DEL ALUMNO: RODOLFO ALBERTO VÁSQUEZ ALAVEZ

NOMBRE DEL FACILITADOR: BENEDICTO RAMÍREZ SANTIAGO

TEMA: INVESTIGACIÓN DE LOS ELEMENTOS BÁSICOS Y SENTENCIAS DE CONTROL DE UN LENGUAJE DE PROGRAMACIÓN (KOTLIN) PARA APLICACIONES MÓVILES.

22-MARZO-2022

# ÍNDICE

# INTRODUCCIÓN

En esta investigación se hablara sobre Kotlin, sus elementos básicos y sentencias de control. Kotlin es un lenguaje de programación de tapado estático que puede correr sobre JVM, JavaScript y desde hace poco sin necesidad de ninguna de estas ya que paralelamente se está desarrollando en nativo con LLVM. Gracias a esto es totalmente interoperable con código Java lo que nos pe permite migrar de una forma.

Los principales conceptos y relaciones que encontrarás en el desarrollo de aplicaciones Android con Kotlin son:

- ViewModel: almacena y gestiona la lógica de vista, de manera optimizada. Permite que los datos sobrevivan a cambios de configuración que el sistema realiza sobre las vistas. Se comunica con la vista mediante propiedades de tipo LiveData de forma que la refrescan en los cambios de estado.
- Fragment: es la unidad mínima y recomendada de vista, aunque siempre debe estar alojado en una Activity. Google promueve cada vez más aplicaciones SingleActivity. Debe ser statless, para almacenar el estado y toda la lógica tenemos el ViewModel. Se encarga de escuchar a las propiedades de LiveData que expone el ViewModel y reacciona en consecuencia según cambien. Su ciclo de vida está afectado por el de su “padre”, pero es independiente.
- LiveData: es un contenedor de datos observable, optimizado para los ciclos de vida de los componentes de vista.
- Lanza eventos solo en los cambios de estado a sus observadores. Solo actualiza observadores de componentes con ciclo de vida activo. Es la tecnología recomendada para la interconexión del Fragment con su ViewModel. Aumento del número de operadores al estilo RX.

# DESARROLLO (ELEMENTOS BÁSICOS Y SENTENCIAS DE CONTROL DE UN LENGUAJE KOTLIN)

## ¿Qué es Kotlin?

La historia de Kotlin data de 2016, cuando JetBrains, la famosa empresa creadora de varios de los IDE's más usados como IntelliJ, WebStorm entre otros, publica la primera versión de este lenguaje de programación.

Se trata de un lenguaje de programación de tapado estático que puede correr sobre JVM, JavaScript y desde hace poco sin necesidad de ninguna de estas ya que paralelamente se está desarrollando en nativo con LLVM. Gracias a esto es totalmente interoperable con código Java lo que nos permite migrar de una forma gradual nuestros proyectos.

## ¿Para qué sirve Kotlin?

Si has oído hablar de Kotlin seguramente sea en el desarrollo de Android, pues desde el año 2019 Google se ha declarado *Kotlin First* es decir, los nuevos desarrollos se harán para Kotlin y una vez terminados se implementarán en Java. Esto hizo que el sector móvil se diese cuenta de la importancia de dicho lenguaje y se centrara en él. Es por ello que actualmente más del 80% de las 1000 apps más populares para Android usan Kotlin.

Pero diferencia de lo que comúnmente se cree, Kotlin no solo sirve para el desarrollo de aplicaciones Android, sino que a lo largo de sus años de desarrollo ha ido abriéndose puertas en otros sectores.

Cada vez es más común ver el backend escrito en Kotlin ya que permite crear un código muy legible y expresivo mientras mantiene actualizado ya que corre a través de JVM. Son varios los frameworks disponibles que nos simplifican el trabajo, entre los que destacan:

- Spring: Posiblemente uno de los frameworks más populares en el desarrollo del *server-side* y desde la versión 5.0 es compatible con Kotlin.

- Ktor: Creado también por JetBrains lo que significa que su integración será completa. Se definen como un sistema que no añade restricciones sobre la tecnología que uses en tu proyecto, asíncrono (aprovechando las famosas corrutinas de Kotlin) y *testable* ya que nos proveerá de varias herramientas que nos simplificará el desarrollo de los test.

- JavaLin: A diferencia de los anteriores destaca por su ligereza y simpleza, pero que no te engañen, más de una empresa grande lo utiliza. Entre ellas Microsoft, Uber y Revolut.

## Características de Kotlin

Kotlin destaca por varias características que no solo simplifican la lectura del código sino el propio desarrollo de este. Es por ello que las ventajas de Kotlin respecto a Java a la hora de desarrollar aplicaciones móviles ha hecho que este lenguaje se haga cada vez más popular.

## Curva de aprendizaje

La simpleza de la sintaxis permite una curva de aprendizaje muy sencilla, ideal para aprender tu primer lenguaje de programación. Por ponerte un ejemplo muy habitual al trabajar con listas en Java, para acceder a la primera posición tendrás que buscar en la posición 0, en Kotlin simplemente llamarías a la función *first()*.

## Orientado a objetos y Funcional

Aunque lo habitual en el desarrollo de aplicaciones móviles era un paradigma de programación orientada a objetos (o POO), Kotlin ha venido a romper los esquemas y demostrarnos que la POO puede trabajar de la mano de la programación funcional. La posibilidad de trabajar con lambdas en este entorno simplifica las tareas más comunes y tediosas en el desarrollo.

Estas son solo algunas de las características de Kotlin que lo hacen especialmente poderoso para trabajar en nuestras apps, aun así, Kotlin vs Java es un debate que se mantendrá al rojo vivo durante varios años más, pero si aún te estás preguntado si deberías seguir con Java o empezar en Kotlin te recomiendo que simplemente eches un ojo a LinkedIn para que veas por ti mismo que se demanda más a día hoy.

## Corrutinas

Posiblemente una de las mayores ventajas que este lenguaje nos aposta. Las corrutinas optimizan la programación asíncrona simplificando el tedioso trabajo de las llamas de red y accesos a nuestras bases de datos entre otras. Con ello podrás olvidarte de librerías como rxJava.

## Comunidad

La comunidad de Kotlin está muy enfocada en el *open source* y gracias a eso tenemos una ingente cantidad de documentación y código libre que nos ayudará a entender a fondo este lenguaje desde el comienzo, como por ejemplo este curso de Kotlin

### Kotlin Multiplatform mobile

Posiblemente no hayas escuchado hablar de esto ya que todavía está en alpha, pero Kotlin Multiplatform nace con la idea de compartir la misma lógica de negocio en iOS y Android con Kotlin. Este tema es muy interesante ya que podríamos tener

en el futuro casi la misma app en ambas plataformas 3ecompartiendo todo el *core* del desarrollo. Y aunque pienses que todavía está muy verde ya que es alpha, grandes empresas como Netflix y Philips ya lo están usando.

## Fundamentos Básicos

- Variables – te enseña a declarar y usar variables de solo lectura y mutables.
- Tipos primitivos – te introduce a los tipos elementales de Kotlin como enteros, flotantes, booleanos y caracteres.
- Strings – tutorial descriptivo para el uso de cadenas de texto en tus aplicaciones Kotlin.
- Arreglos – aprende a crear arreglos en Kotlin y a leer, modificar y recorrer sus elementos.
- Conversión de tipos – te muestra cómo usar la conversión explícita entre tipos de variables.
- Casting – en este tutorial te mostraremos cómo funciona el casting en Kotlin y varias de sus características auxiliares como Smart cast y operadores de casting seguro e inseguro
- Operadores – te muestra los operadores existentes para crear expresiones entre variables.
- Flujo de entrada y salida – te enseña a usar funciones para la escritura en pantalla y lectura de datos desde teclado.
- Anulabilidad – el uso del literal null es diferente en Kotlin, ya que existen tipos anulables y no anulables. Aquí verás esta diferencia y diferentes formas de comprobar si un valor es null.
- Comentarios – te mostraremos las formas de escribir comentarios en Kotlin para agregar información a tu código que te permita expresar aspectos relacionados con documentación, intenciones, claridad, consecuencias, legalidad, etc.
- Paquetes e importaciones – organiza estructuralmente las declaraciones de tus programas e importalas en otros otros archivos fuente de Kotlin.

## Control De Flujo

- Expresión if – te muestra a como bifurcar el flujo de tus programas con la expresión if.
- Expresión when – te enseña el uso de la expresión when como alternativa al switch de Java.
- Bucle for – verás el uso de la sentencia for para iterar sobre colecciones de elementos.
- Bucle while – usa las sentencia while y do..while para ejecutar acciones basadas en el cumplimiento de una condición.
- Expresión break – aprende a terminar un bucle prematuramente con break.
- Expresión continue – te muestra como saltar iteraciones de bucles con continue.

- Excepciones – aprenderás sobre el lanzamiento y manejo de excepciones en Kotlin con el objetivo de conducir el flujo de tus programas a un estado consistente.

## Sintaxis de Kotlin

En Kotlin podemos decidir si nuestra variable será como su propio nombre indica una variable o si por el contrario será constante, esto nos da el don de la inmutabilidad que nos librará de errores y además ahorrará memoria.

Si quisiéramos un valor constante utilizaríamos la palabra reservada `val`

```
val name = "Aris"
```

Si por el contrario queremos que sea variable la reemplazaríamos por `var`

```
var name = "Aris"
```

Si te fijas bien en los ejemplos anteriores verás que no he definido el tipo de variable ya que Kotlin es capaz de adivinar el tipo por el valor asignado. Aunque también podremos añadírselo nosotros mismos para forzar.

```
var name:String = "Aris"
```

Funciones

Podemos distinguir tres tipos distintos, básicas, con parámetros de salida o con parámetros de entrada.

La más básica sería utilizando la palabra reservada `fun` seguido del nombre de esta.

```
fun greetings(){ print("Hello!") }
```

Si quisiéramos que nos saludara con un nombre concreto crearíamos una función con variable de entrada.

```
fun greetings(name:String){ print("Hello! $name") }
```

Nuestra función puede recibir entre cero y N parámetros y lo concatenamos a nuestro saludo. A diferencia de Java podemos crear parámetros opcionales y gracias a ello tenemos la posibilidad de pasar el parámetro o no.

```
fun greetings(name:String = "Guest"){ print("Hello! $name") }
```

Si recibe el parámetro `name` contendrá dicho valor, sino usará el de por defecto.

Para terminar, te mostraré como devolver el saludo, es tan sencillo que solo debemos definir el tipo de valor que vamos a devolver y retornarlo.

```
fun greetings(name:String = "Guest"): String{ return "Hello! $name" }
```

Sencillo ¿Verdad? Pues lo podemos aún simplificar más ya que Kotlin nos permite dejar el código impoluto con cualidades como esta.

```
fun greetings(name:String = "Guest") = "Hello! $name"
```

Hemos pasado de tener una función de tres líneas a conseguir la optimización por excelencia y dejar un código limpio y sencillo.

## Funciones de extensión

Esta es otra de las novedades de Kotlin, este tipo tan singular de funciones nos permite extender de cualquier tipo de código para crear funciones exclusivas. Si ya desarrollas con Java sería como la función vitamina de las funciones estáticas.

Esto es realmente útil para las funciones repetitivas que debemos realizar por todo nuestro proyecto. Pongamos el ejemplo de Glide, posiblemente la librería más popular a día de hoy para cargar urls en *imageViews*. Normalmente tendríamos que ir en cada una de las clases donde lo queramos usar añadiendo

```
Glide.with(context) .load(myUrl) .into(myImageView)
```

Ahora solo tendríamos que definir una función que extienda de un *imageView*

```
fun ImageView.loadUrl(url:String){ Glide.with(this) .load(myUrl) .into(this) }
```

Además, podemos usar el propio componente para acceder a su contexto.

Ahora en cualquier sitio de nuestra app que queramos usar Glide solo tendremos que acceder al *imagieView* y tendrá la nueva función implementada.

## Función

```
fun max(a: Int, b: Int): Int {
    return if (a > b) a else b
}
>>> println(max(1, 2))
2
```

La declaración de una función comienza con la palabra clave fun, seguida del nombre de la función. El nombre de la función en este ejemplo es max, seguido de la lista de parámetros entre paréntesis. La lista de parámetros va seguida del tipo de valor de retorno, separado por dos puntos.

## Cuerpo de la función de expresión

La función anterior se puede simplificar. Debido a que el cuerpo de su función está compuesto por una sola expresión, puede usar esta expresión como un cuerpo de función completo y quitar las llaves y la declaración return:

```
fun max(a: Int, b: Int): Int = if (a > b) a else b
```

Si el cuerpo de la función está escrito entre llaves, entonces la función tiene un cuerpo de bloque de código, si devuelve directamente una expresión, tiene un cuerpo de expresión.

Puede simplificar aún más la función max y omitir el tipo de valor de retorno:

```
fun max(a: Int, b: Int) = if (a > b) a else b
```

Tenga en cuenta que solo se puede omitir el tipo de valor de retorno de la función del cuerpo de la expresión. Para una función de bloque de código con un valor de retorno, el tipo de valor de retorno y la declaración de retorno deben escribirse explícitamente.

## Variables

La declaración de una variable en Kotlin comienza con una palabra clave, luego el nombre de la variable y finalmente el tipo.

```
val str = "Hola"
val age = 25
```

Si es necesario, también puede especificar el tipo de variable explícitamente:

```
val age: Int = 25
```

Si la variable no tiene un inicializador, debe especificar su tipo explícitamente:

```
val age: Int
age = 25
```

## Variable e invariable

Hay dos palabras clave para declarar variables:

- val (valor) ------- referencia inmutable. Las variables declaradas con val no se pueden volver a asignar después de la inicialización, que corresponde a las variables finales en Java.
- var (variable) --- referencia de variable. El valor de esta variable se puede cambiar.

Nota: Aunque la referencia val en sí es inmutable, el objeto al que apunta puede ser mutable. P.ej:

```
val languages = arrayListOf("Java")
languages.add("Kotlin")
```

Aunque la palabra clave var permite que una variable cambie su valor, su tipo no se puede cambiar. P.ej:

```
var age = 25
 age = "En ese momento joven"
```

No se compilará.

## Formato de cadena más simple: plantilla de cadena

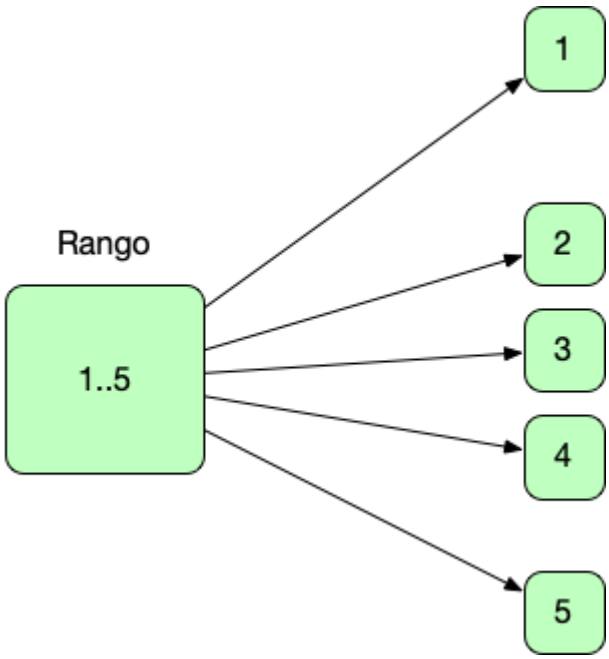
```
val name = "En ese momento joven"
println("Hello, $name")

 Hola era joven
```

Kotlin le permite hacer referencia a variables locales en cadenas literales, simplemente agregue el carácter $ delante del nombre de la variable. Esto es equivalente a la concatenación de cadenas en Java`"Hello ," + name`, Misma eficiencia pero más compacta.

## Kotlin Ranges y estructura de control

Las estructras de control en kotlin se pueden generar de una forma más compacta a través del uso de rangos. Un rango no es ni más ni menos que un conjunto de valores soportados en este caso números enteros.

**val numeros:Int:Range=1..5**




Este concepto de Rango se puede encajar perfectamente dentro de nuestra estructura if/else pero usando Kotlin como lenguaje.

```
var nota = 3

var mensaje=if (nota in 1..2) {
    "muy mal"
} else if (nota in 3..4) {
    "mal"
} else if (nota in 5..7) {
    "ok"
} else {
    "genial"
}
print(mensaje)
```

Acabamos de construir una estructura que nos permite generar de una forma más compacta nuestro código y sobre todo más sencilla de entender ya que usamos

if/else combinado con rangos. Esta sintaxis la podemos mejorar y compactar todavía más usando la clausula when de Kotlin.

```
var nota = 9

var mensaje=when (nota) {
  in 1..2->"muy mal"
  in 3..4->"mal"
  in 5..7->"ok"
 else->"genial"
}
print(mensaje)
```

## CONCLUSIÓN

Conclui que Kotlin es un lenguaje de programación de código abierto, de tipo estático y de uso general. Un estudio reciente de Google indica que los desarrolladores que adoptan Kotlin, se ejecuta en Java Virtual Machine (JVM) y también se puede utilizar para desarrollar aplicaciones de Android, aplicaciones del lado del servidor y mucho más. Kotlin fue desarrollado por el equipo de JetBrains. Permite concentrarse en expresar sus ideas y escribir menos código repetitivo. Este lenguaje se ha incluido como un enfoque alternativo al compilador estándar de Java. Una encuesta reciente muestra que el 60% de los desarrolladores de Android Pro utilizan Kotlin en su código base.

## BIBLIOGRAFÍA


Orozco, A. G. (5 de Abril de 2021). *OpenWebinars*. Recuperado el 22 de Marzo de 2022, de Qué es Kotlin y características: https://openwebinars.net/blog/que-es-kotlin/

Programmerclick. (s.f.). *programador clic*. Recuperado el 22 de Marzo de 2022, de Elementos básicos de Kotlin 2.1: funciones y variables: https://programmerclick.com/article/87981563847/

VALENCIA, U. P. (s.f.). *Estructuras de control en Kotlin*. Recuperado el 22 de Marzo de 2022, de http://www.androidcurso.com/index.php/99-kotlin/912-estructuras-de-control-en-kotlin