INSTITUTO POLITÉCNICO NACIONAL


# Escuela Superior de Cómputo (ESCOM)

**PROFESOR**: Alejandro Sigfrido Cifuentes Alvarez.

**UNIDAD DE APRENDIZAJE**: Application development for mobile devices.

REPORTE: Kotlin.

ALUMNO: Monroy Martos Elioth.

GRUPO: 3CM8

# Índice

## Objetivo

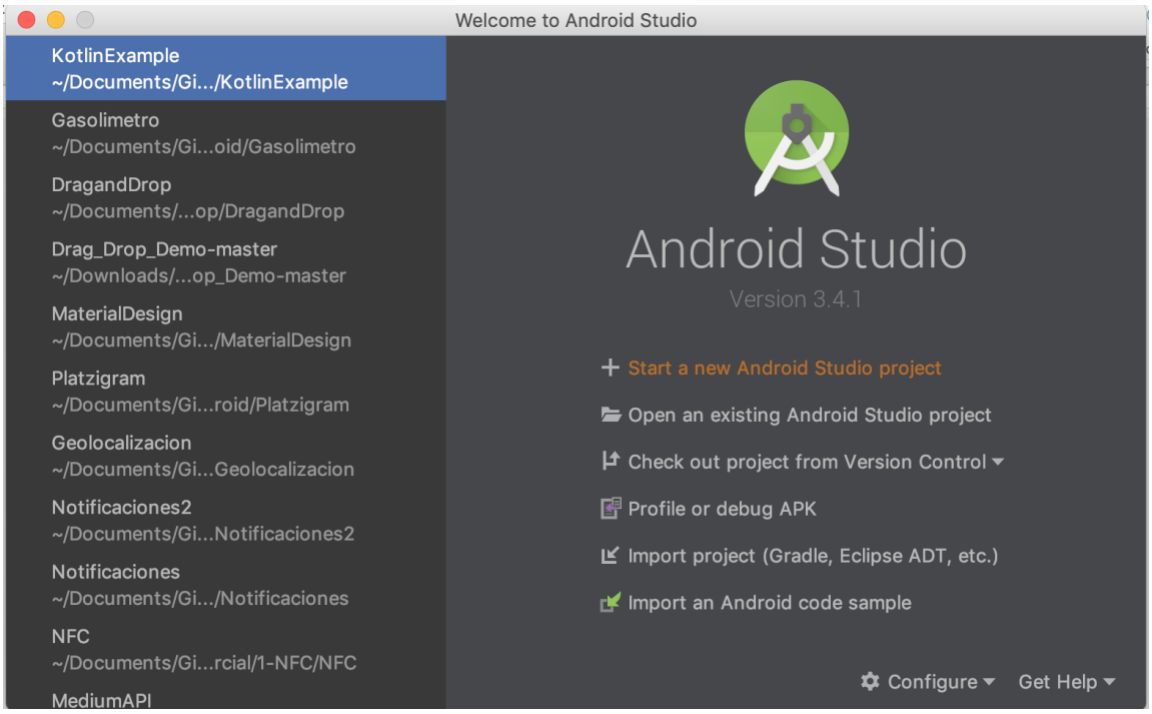
Comprender los conceptos básicos de Kotlin y desarrollar una aplicación móvil en la cual se apliquen. Tales como la ausencia de variables nulas por defecto, el uso de operadores tales labda u Elvis, además de la definición de objetos singleton inline.

## Conceptos

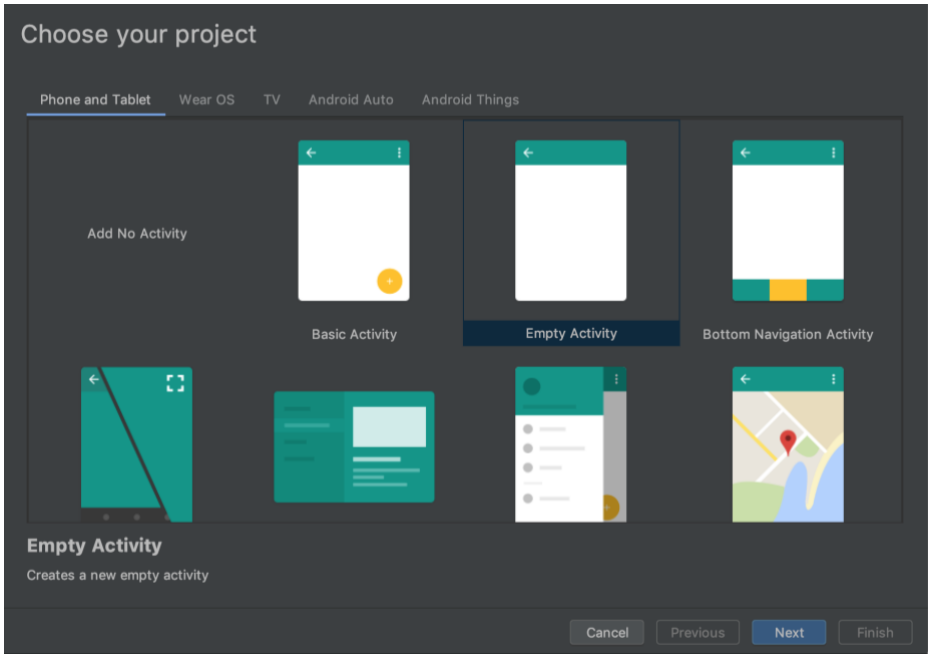
- Kotlin: Lenguaje de programación oficial para el desarrollo de aplicaciones Android, creado por la compañía Jetbrains. El cual se caracterísa por ser compatible completamente con Java, variables de tipo finales y públicas por defecto, no nulas por defecto, el uso del operador lambda, y además, permite extender la funcionalidad de las clases definidas en el sdk de Android, agregando funciones según necesitemos.[1]

## Desarrollo

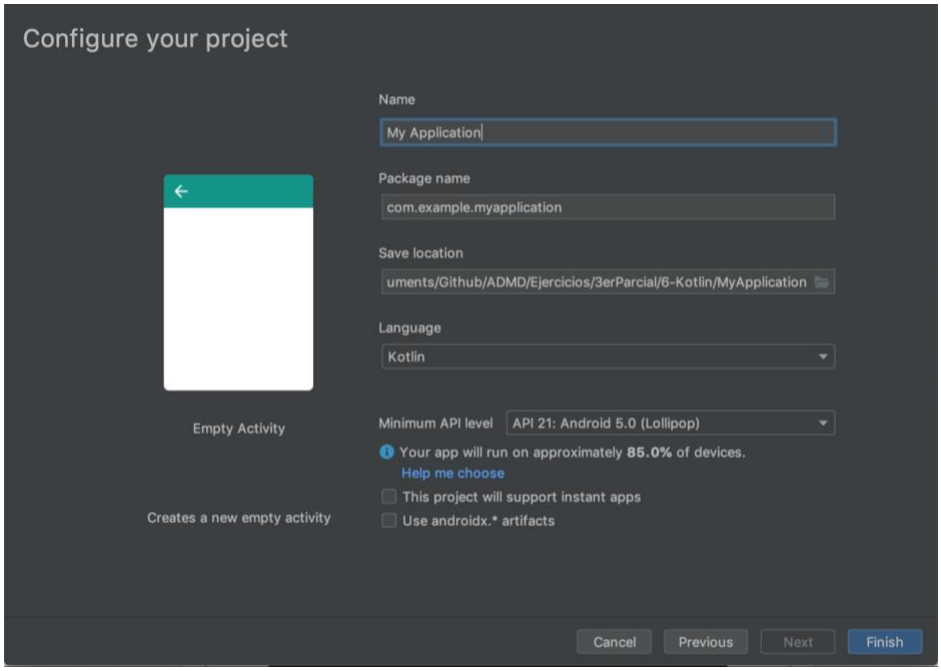
Para trabajar con Kotlin, es necesario usar el IDE oficial para el desarrollo de aplicaciones Android, Android Studio, el cual es desarrollado por la empresa Jetbrains, quienes también dan mantenimiento al proyecto de Kotlin. Al instalar la versión más reciente, nos aparece una pantalla como la siguiente, donde debemos seleccionar “Iniciar un nuevo proyecto de Android Studio”.



Posteriormente, podemos seleccionar alguna de la plantillas que se nos muestran, la cual será cargada en nuestro main activity. En este caso, seleccionamos “Empty Activity”.



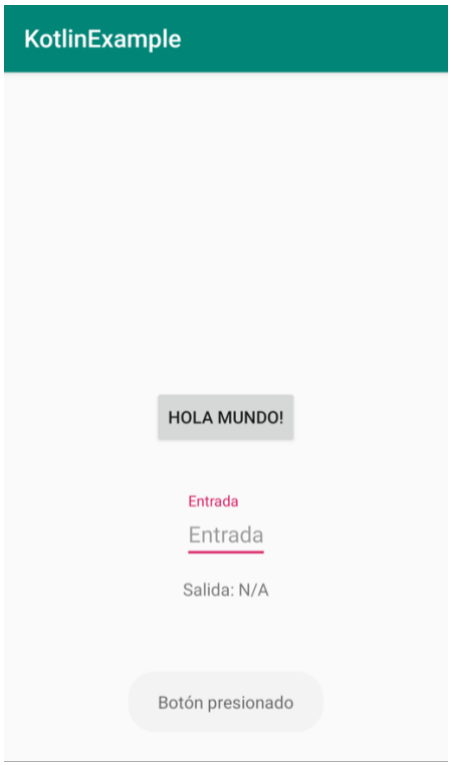

Al presionar el botón de “Siguiente” en la pantalla anterior, ahora aparece la siguietne pantalla, la cual nos pide configurar nuestro proyecto, desde el nombre, versión de Android mínima en la que daremos soporte, y también el lenguaje que deseamos usar. Para trabajar con Kotlin, debemos cambiar en esa sección de “Java” a “Kotlin”.



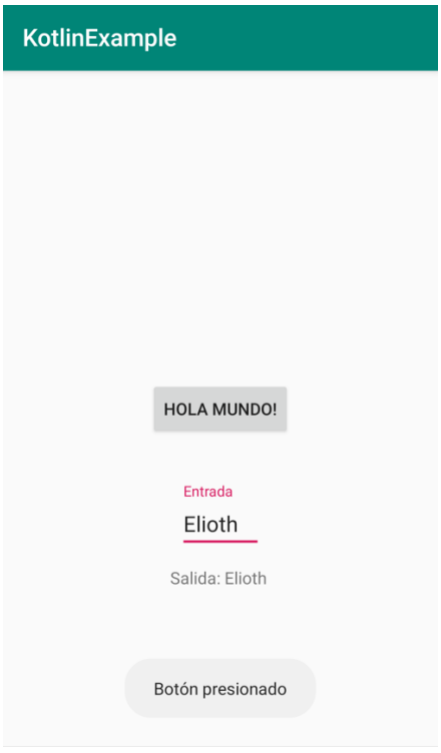
## Pruebas

A continuación, se muestra una aplicación Android elaborada en Kotlin, en la cual se aplicaron los conceptos básicos del lenguaje. Como se puede observar, la pantalla solo se compone de un toolbar por defecto, un botón, y una entrada de texto. Al iniciar el activity, se crea un objeto singleton y se llama a una función interna del mismo la cual imprime en un toast el mensaje “Objeto singleton”. Además, mediante el uso de lambda se agrega el actionListener al botón, el cual al ser presionado, muestra en la parte inferior del campo de texto, el texto ingresado, en caso de ser este nulo, muestra el valor por defecto “N/A”.

En la pantalla siguiente, se puede observar el mensaje que se muestra al ser nula la entrada. Esto se logra, al usar el operador elvis (?:) de Kotlin.




Finalmente, en la última imagén, se muestra el resultado de ingresar algún texto en el TextField.



## Código fuente

A continuación, se muestra el código xml con el que se definió la interfaz anterior.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".MainActivity">
    <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/boton"
            android:text="Hello World!"
            android:layout_gravity="center"
            />
    <android.support.design.widget.TextInputLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:gravity="center"
            android:layout_marginTop="180dp"
            android:hint="Entrada">
        <android.support.design.widget.TextInputEditText
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:id="@+id/entrada"
                android:hint="Entrada"/>
    </android.support.design.widget.TextInputLayout>
    <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:gravity="center"
            android:layout_marginTop="280dp"
            android:id="@+id/salida">
    </TextView>
</android.support.design.widget.CoordinatorLayout>
```

Finalmente, se muestra el código Kotlin con el cual se maneja la lógica de la interfaz de usuario mostrada anteriormente.

ActivityMain.kt

```
package com.example.kotlinexample
import android.content.Context
import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import android.widget.Toast
import kotlinx.android.synthetic.main.activity_main.*
class MainActivity : AppCompatActivity() {
    //Variable que no aceptará nulos
    var username:String=""
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        boton.text = "Hola mundo!"
        //Variable nula
        var ejemplo : String? = null
        //Verificación nulo
        Log.d("tag:",""+ejemplo?.length)
        //Lambda
        boton.setOnClickListener{
            toast("Botón presionado")
            //Operador elvis
            if(entrada.text!!.toString().equals("")){
                setName(null)
            }else{
                setName(entrada.text.toString())
            }
        }
        //Invocando al singleton
        KotlinSingleton.myFunction(this,"Objeto
singleton",Toast.LENGTH_LONG)
    }
    fun setName(name:String?){
        this.username =name?:"N/A"
        salida.text="Salida: "+this.username
    }
    fun AppCompatActivity.toast(msg:String){
        Toast.makeText(this,msg,Toast.LENGTH_LONG).show()
    }
}
//Objetos singleton
object KotlinSingleton{
    fun myFunction(context:Context, msg:String, int: Int){
        Toast.makeText(context,msg,int).show() }}
```

## Conclusiones

El lenguaje de programación Kotlin, ofrece una alternativa al estándar de desarrollo para aplicaciones móviles para el teléfono que antes era Java, debido a que permite hacer desarrollo nativo para Android, completamente independiente o inclusive combinado con Java. Lo cual brinda aún mayor flexibilidad para los desarrolladores, para que así puedan obtar por alguno de los dos lenguajes según sean las necesidades del proyecto y de la aplicación. Kotlin trae consigo algunas cosas modernas e interesantes que Java en su momento tardo de adoptar, o que simplemente por como estaba diseñado el lenguaje no era posible hacer. Como es el caso de evitar los NullPointerException, situación que por default Kotlin prevee. Sin ser un lenguaje mejor que el otro, Kotlin trae un nuevo mundo de posibilidades al desarrollo de aplicaciones Android.

## Bibliografía


- [1] M. Lozano Ortega and B. Bonev, "Introducción a Kotlin", *Curso de Android y Java para Dispositivos Móviles*, 2017. [Online]. Disponible en: http://www.jtech.ua.es/apuntes/ajdm2017/sesiones/sesion08-apuntes.html. [Consultado: 23-Mayo- 2019].