

PRÁCTICA 1.2

INTRODUCCIÓN A KOTLIN

PROGRAMACIÓN MULTIMEDIA Y DISPOSITIVOS MÓVILES 24/25
CFGS DAM

Autor: Mara Vañó

m.vanoalonso@edu.gva.es

Fecha: 2024/2025

Licencia Creative Commons

versión 4.0



Reconocimiento – NoComercial – CompartirIgual (by-nc-sa): No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

ÍNDICE

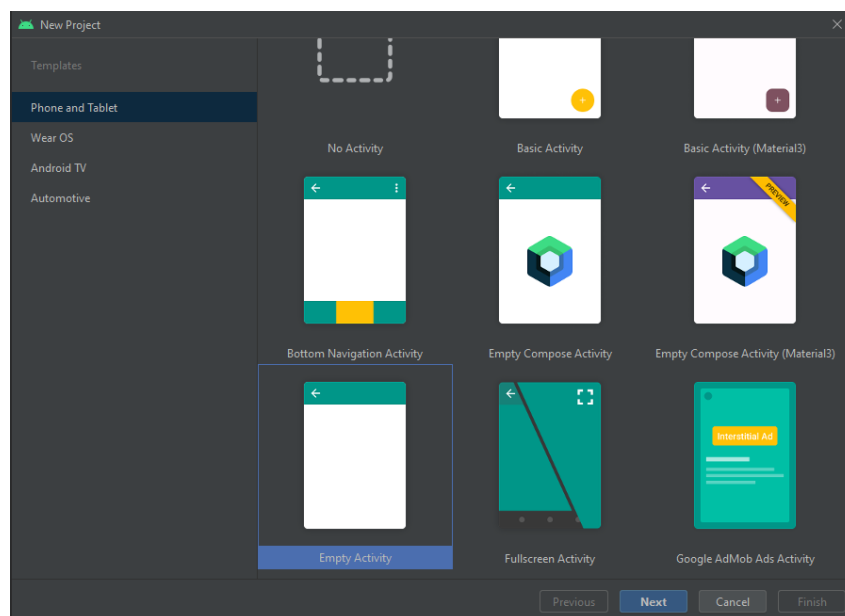
Objetivos de la práctica	3
Ejercicio 0. Crear mi primer proyecto	3
Ejercicio 1. Definir una variable o una constate.....	5
Ejercicio 2. Imprimir por pantalla.	6
Ejercicio 3 (TEÓRICO): Operadores lógicos y operadores condicionales.....	7
Ejercicio 4. Sentencia IF ELSE	8
Ejercicio 5: Sentencia when	8

1. OBJETIVOS DE LA PRÁCTICA

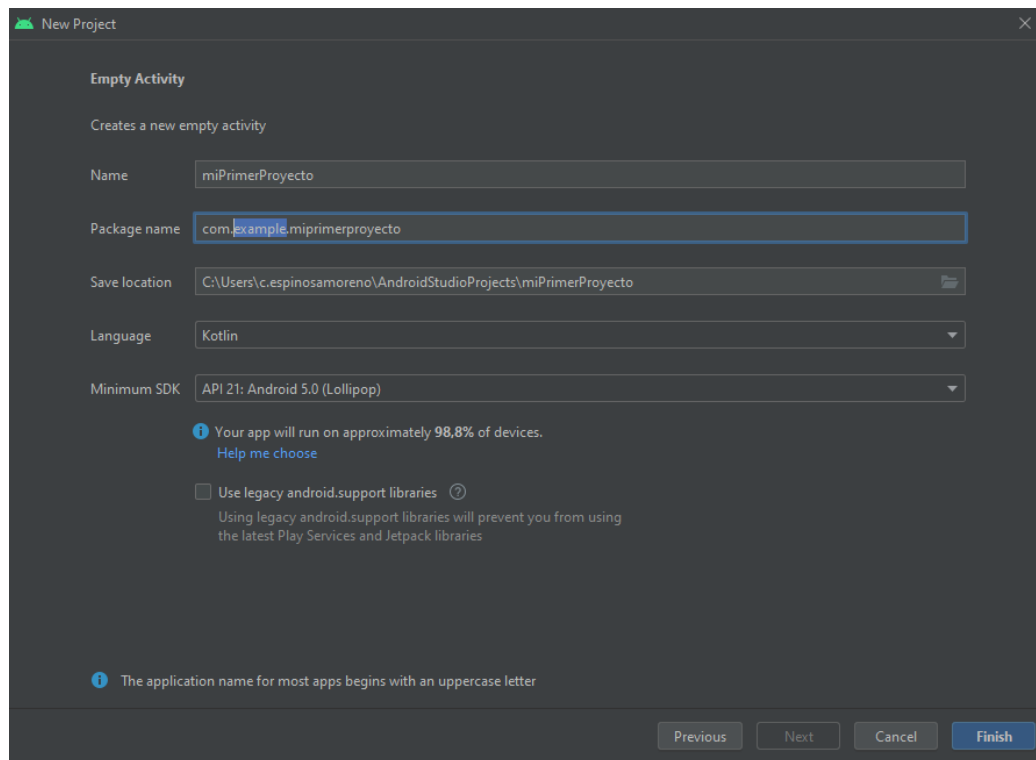
- Aprender a **crear** nuestro primer **proyecto** en Android Studio
- **Familiarizar** con el nuevo **entorno** de **desarrollo**
- Repaso de las **instrucciones básicas**
- Realizar **ejercicios** de **introducción** para repasar el funcionamiento de **Kotlin**

2. EJERCICIO 0. CREAR MI PRIMER PROYECTO.

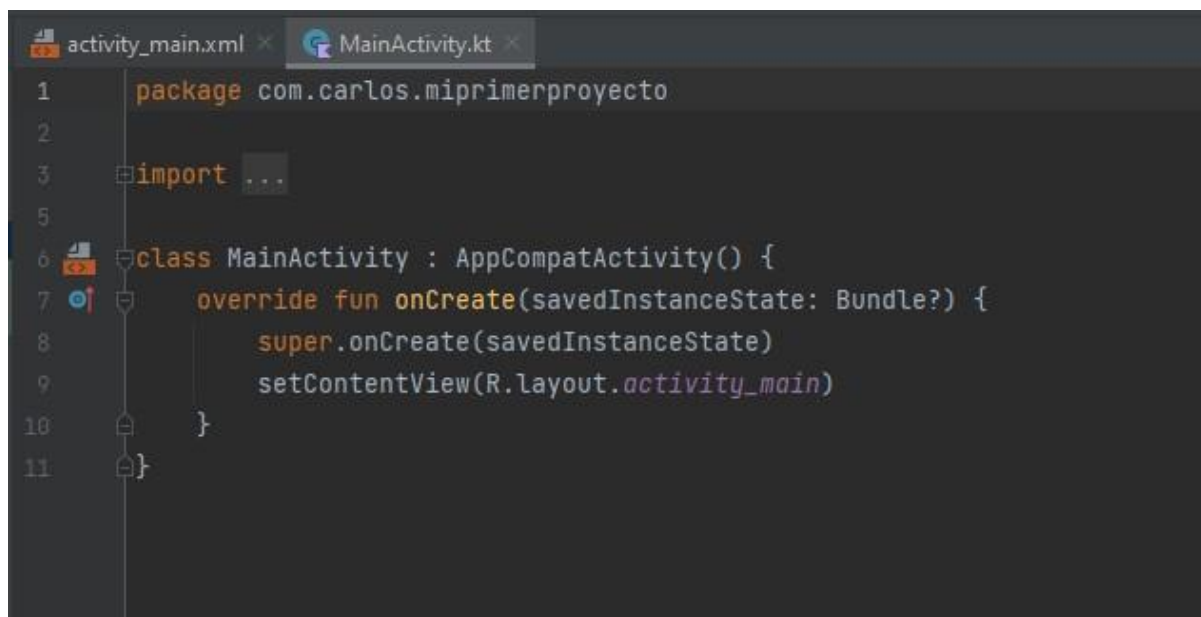
Para crear un proyecto en Android Studio iremos a File → New... → New Project y elegiremos Empty Activity. Tenemos diferentes plantillas predefinidas dependiendo del uso que queramos dar a nuestra aplicación, pero en nuestro caso como estamos aprendiendo desde cero creamos un proyecto vacío.



- En la configuración básica del proyecto sugiero nombrar el proyecto con el nombre de la práctica utilizando la nomenclatura **Camel Case** donde pondremos en minúscula la primera palabra y posteriormente en mayúscula la primera de cada palabra que concatenemos. Ej.: **miPrimerProyecto**. Tened en cuenta que Kotlin es un lenguaje de programación que diferencia entre mayúsculas y minúsculas.
- En el apartado **Package name**, pondremos nuestro **nombre** o el de nuestra organización, será el nombre del directorio donde se incluirán todos los elementos de la aplicación.
- Seleccionamos **Kotlin** como lenguaje preferido para programar.
- Finalmente, en el apartado **minimum SDK** seleccionaremos la versión **Android 5 Lollipop** ya que es soportada por el **98,8%** de los usuarios en la actualidad. A lo largo del módulo, si necesitásemos alguna función muy específica para nuestra aplicación, podremos seleccionar una versión más actual.

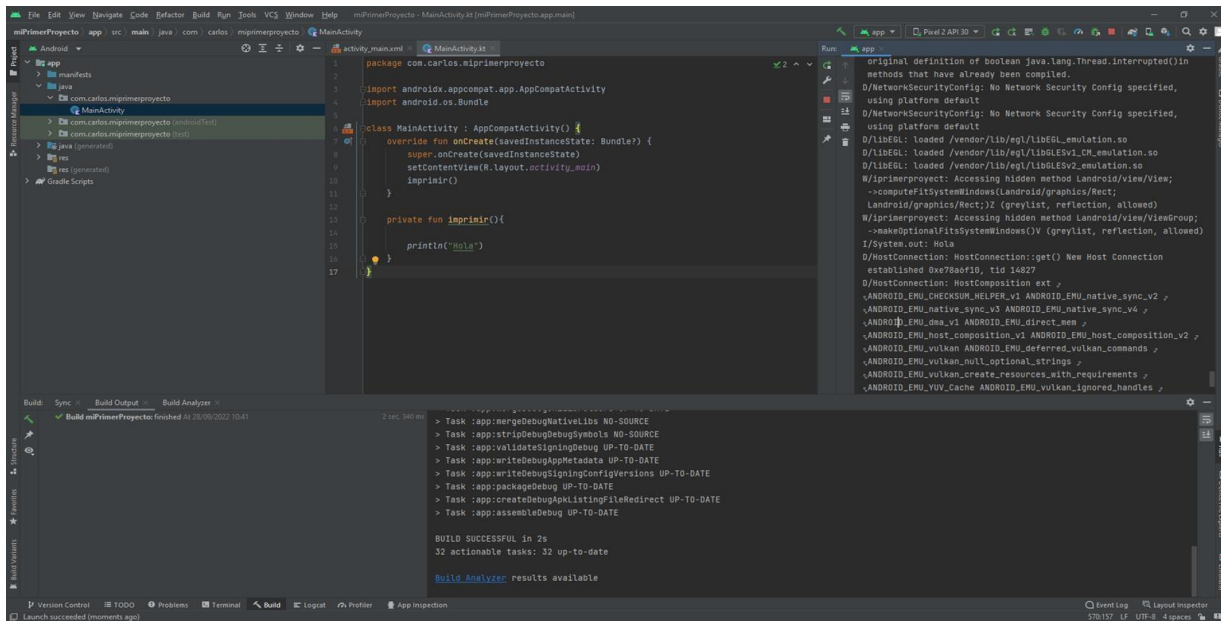


En la parte central de Android Studio encontraremos **Main Activity**; esta es la parte principal de nuestro programa y donde empezaremos a desarrollar nuestra aplicación.



En la pestaña **Run** nos aparecerá toda la salida del proceso de compilación de nuestro código. Si no nos aparece la pestaña de Run podemos abrirla en **View → Tool Window → Run**

Mi consejo es que tengáis la pantalla de Main Activity y la de Run una al lado de la otra para poder ver por pantalla el resultado de nuestras instrucciones, tal y como aparece en la siguiente imagen:



3. EJERCICIO 1. DEFINIR UNA VARIABLE O UNA CONSTANTE.

- Define como **constante** el número **pi = 3,141516** y la **velocidad de la luz = 299792458 m/s**
- Crea dos **variables** uno con tu **nombre** y el otro con tu **edad**

En Kotlin podemos definir diferentes **tipos de datos**:

Númerico:

- Double:** El tipo de dato double es un dato en coma flotante IEEE 754 de 64 bits y precisión doble.
- Float:** El tipo de dato float es un dato en coma flotante IEEE 754 de 32 bits y precisión simple.
- Long:** El tipo de dato long es un entero de 64 bits complemento a dos. Su valor mínimo es 9,223,372,036,854,775,808 y el máximo 9,223,372,036,854,775,807
- Int:** El tipo de dato int es un entero de 32 bits complemento a dos. Su valor mínimo es 2,147,483,648 y el máximo 2,147,483,647.
- Short:** El tipo de dato short es un entero de 16 bits complemento a dos. Su valor mínimo es 32,768 y el máximo 32,767.
- Byte:** El tipo de dato byte es un entero de 8 bits complemento a dos. Su valor mínimo es 128 y el máximo 127.

Booleanos: El tipo de dato boolean solamente tiene dos valores posibles: true (verdadero) y false (falso).

- true:** se cumple la sentencia
- false:** no se cumple la sentencia

Cadenas de texto

- String:** es una clase que permite la definición y manejo de cadenas de caracteres

Arrays, Null...

EJEMPLO EJERCICIO 1:

```
1 package com.carlos.miprimerproyecto
2
3 import androidx.appcompat.app.AppCompatActivity
4 import android.os.Bundle
5
6 class MainActivity : AppCompatActivity() {
7     override fun onCreate(savedInstanceState: Bundle?) {
8         super.onCreate(savedInstanceState)
9         setContentView(R.layout.activity_main)
10        variables()
11    }
12
13    private fun variables(){
14
15        //constates
16        val numeroPi = 3.141516
17
18        //variables
19        var x= 2
20        var y: Int= 5
21        var miNombre: String = "Carlos"
22
23    }
24 }
25
```

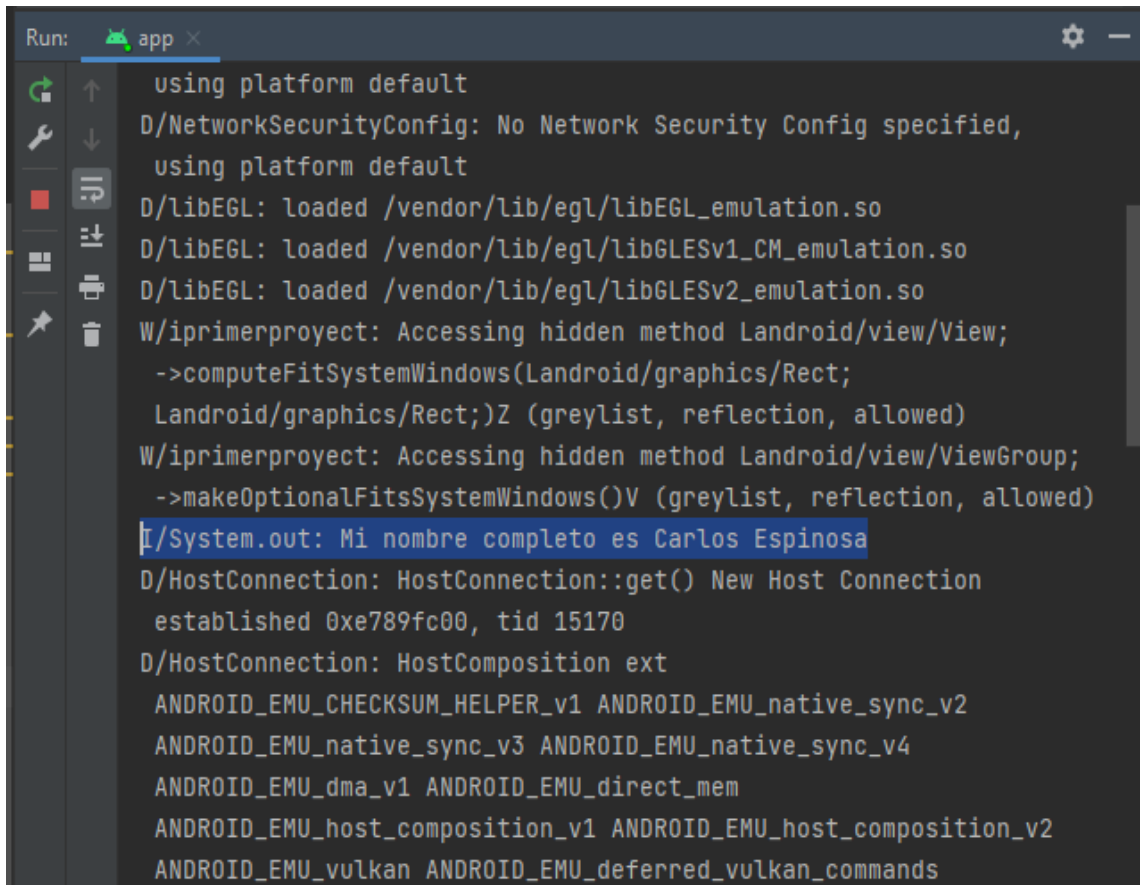
4. EJERCICIO 2. IMPRIMIR POR PANTALLA.

- a) Imprime por pantalla tu nombre y tu edad sin utilizar variables
- b) Imprime por pantalla tu nombre y edad utilizando las variables definidas en el ejercicio 1.

Para sacar por pantalla nuestras variables o texto debemos utilizar la función println

```
private fun imprimir(){
    val nombre = "Carlos"
    val apellido = "Espinosa"
    println("Mi nombre completo es $nombre $apellido")
}
}
```

En la pestaña **Run** aparecerá como **System.out**. Revisad bien la salida porque Android Studio imprime por pantalla todos los procesos.



```
Run: app ×
using platform default
D/NetworkSecurityConfig: No Network Security Config specified,
using platform default
D/libEGL: loaded /vendor/lib/egl/libEGL_emulation.so
D/libEGL: loaded /vendor/lib/egl/libGLESv1_CM_emulation.so
D/libEGL: loaded /vendor/lib/egl/libGLESv2_emulation.so
W/iprimerproyect: Accessing hidden method Landroid/view/View;
->computeFitSystemWindows(Landroid/graphics/Rect;
Landroid/graphics/Rect;)Z (greylist, reflection, allowed)
W/iprimerproyect: Accessing hidden method Landroid/view/ViewGroup;
->makeOptionalFitsSystemWindows()V (greylist, reflection, allowed)
I/System.out: Mi nombre completo es Carlos Espinosa
D/HostConnection: HostConnection::get() New Host Connection
established 0xe789fc00, tid 15170
D/HostConnection: HostComposition ext
ANDROID_EMU_CHECKSUM_HELPER_v1 ANDROID_EMU_native_sync_v2
ANDROID_EMU_native_sync_v3 ANDROID_EMU_native_sync_v4
ANDROID_EMU_dma_v1 ANDROID_EMU_direct_mem
ANDROID_EMU_host_composition_v1 ANDROID_EMU_host_composition_v2
ANDROID_EMU_vulkan ANDROID_EMU_deferred_vulkan_commands
```

5. EJERCICIO 3 (TEÓRICO): OPERADORES LÓGICOS Y OPERADORES CONDICIONALES.

De manera común a muchos lenguajes de programación en Kotlin contamos con operadores lógicos y con operadores condicionales que nos permiten realizar comparaciones y operaciones lógicas. Os dejo aquí un resumen con las más utilizadas:

Operadores condicionales

- > mayor que
- < menor que
- >= mayor o igual
- <= menor o igual
- == igualdad
- != desigualdad

Operadores lógicos

- && operador AND
- || operador OR
- ! operador not

6. EJERCICIO 4. SENTENCIA IF ELSE.

a) Realiza una suma entre dos variables.

Si está en un rango determinado imprímelo por pantalla si no di que tu número esta fuera de ese rango

Cuando estamos desarrollando nuestra aplicación nos puede surgir la necesidad de escribir diferentes condiciones que evaluarán diferentes procesos dentro de la misma. Para ello contamos con la sentencia If, else e Ifelse. Dentro de dicha instrucción utilizaremos operadores lógicos.

EJEMPLO EJERCICIO 4:

```
53 private fun sentenciaIf(){
54
55     val myNumber = 50
56     println("$myNumber")
57
58     if (myNumber <= 10 && myNumber >5){
59         //sentencia
60         println("$myNumber es menor o igual a 20 y mayor que 5")
61     } else if(myNumber == 60){
62         //sentencia if
63         println("$myNumber es igual a 50")
64     }
65     else {
66         // sentencia
67         println("$myNumber es mayor que 10 o menor o igual a 5 ")
68     }
69 }
```

7. EJERCICIO 5: SENTENCIA WHEN.

a) Realiza un programa utilizando la sentencia when que sea capaz de decir a qué continente pertenece un país, utiliza al menos 15 países.

La sentencia when funciona de manera similar a un switch de Java o C++.

Dentro de la instrucción when escribimos el elemento que queremos comparar de una lista de casos. En el momento que coincida con uno de ellos se ejecutara el código de dicho elemento.


```
fun sentenciaWhen() {  
  
    val country = "España"  
  
    when(country) {  
        "España", "Perú", "Argentina" -> {  
            println("El idioma es Español")  
        } "USA" -> {  
            println("El idioma es Inglés")  
        } "Francia" -> {  
            println("El idioma es Francés")  
        } else -> {  
            println("No conocemos el idioma")  
        }  
    }  
}
```