



TALLER DE APLICACIONES MÓVILES

SEMANA 4: HELLO WORLD EN TU MÓVIL.





ESCUELA DE INGENIERÍA Y CONSTRUCCION

Director: Marcelo Lucero

ELABORACIÓN

Experto disciplinar: Javier Ignacio Miles Avello

Diseño instruccional: Felipe Molina

VALIDACIÓN Experto disciplinar: Helmut

Aubel

Jefa de Diseño Instruccional: Alejandra San Juan

EQUIPO DE DESARROLLO

AIEP

AÑO

2021



Tabla de contenidos

Aprendizaje esperado de la semana.....	4
Ejecutan programa básico mediante máquina virtual, considerando lenguajes de programación y compiladores para dispositivos móviles.	4
Introducción.....	4
1. Características de un entorno de desarrollo, herramientas IDE Android Studio, Eclipse y Visual Code.....	5
2. Android SDK e Ionic	6
3. Instalación de Android Studio	7
4. Lenguaje de programación KOTLIN	9
5. Lenguaje de programación JAVA	14
7. Configuración de un dispositivo móvil con Android Debug Bridge ADB.....	27
8. Configuración de un dispositivo físico para depuración de aplicaciones.....	28
Conclusiones.....	35
Referencias	36



Aprendizaje esperado de la semana

Ejecutan programa básico mediante máquina virtual, considerando lenguajes de programación y compiladores para dispositivos móviles.

Introducción

- ¿Qué lenguaje de programación debería utilizar?
- ¿Qué necesito para mi ambiente de desarrollo?
- ¿Qué es un archivo compilado?

En la semana anterior, ya conoces acerca de los diferentes lenguajes de programación, aplicable a dispositivos móviles, . Ahora falta tu imaginación e investigación, para conocer cómo se instala un sistema operativo en un dispositivo Android, lo cual no forma parte como objetivo del curso. Diversos son los lenguajes de programación, y debemos seleccionar el lenguaje que de solución a la problemática del cliente. En esta semana, podrás profundizar aún más en relación con los lenguajes de programación, y cómo funcionan.

En este documento, podrás reconocer qué lenguaje seleccionar, para el desarrollo de aplicaciones.



1. Características de un entorno de desarrollo, herramientas IDE Android Studio, Eclipse y Visual Code.

Durante la lección anterior, aprendimos acerca de los lenguajes de programación, y pudimos deducir que basta con un editor de texto, para poder programar, y posteriormente un compilador. También pudimos concluir, que programar con herramientas básicas, debemos invertir más tiempo en el proceso de desarrollo. Es ahí en donde entran en juego, las herramientas IDE (Integrated Development Environment), lo cual en español significa Entorno de Desarrollo Integrado, los cuales poseen herramientas automatizadas, que realizan las operaciones de compilación vistas en la semana anterior.

Uno de los editores más populares, es eclipse (el mismo que seguramente has utilizado en java), pero de acuerdo con (Eclipse Foundation), puedes descargar la herramienta, y adicionar plugins para distintos lenguajes. Entre ellos Android, aunque hay que considerar la continuidad de mantenimiento de estos packages, y es probable que no continúen adaptándose a las nuevas características.

Otro de las herramientas que hoy en día soporta muchos lenguajes de programación, y se puede realizar personalizaciones, e incluso desarrollar tus propios plugins es VSCode (Microsoft corp.), el cual está bajo licencia MIT. Existen plugins para poder desarrollar una aplicación Android, que permite la ejecución del mismo.

Por último, la herramienta Android Studio (Google Inc.), posee funcionalidades visuales, y características que facilita el desarrollo de “aplicaciones nativas”. Continuamente lanzan actualizaciones y nuevas funcionalidades, que automatizan ciertas características, para ahorrar tiempo en la codificación.



2. Android SDK e Ionic

Dado que en la lección anterior indicamos la naturaleza de Android (en cuanto a sistemas operativos y componentes), podemos concluir que podemos desarrollar en varios lenguajes, de manera nativa. Además, existen componentes incluidos (como Aplicaciones Web Progresivas), en donde se han desarrollado diversos framework, para ello, como lo es (IONIC Framework), el cual consiste en elementos Javascript, CSS y HTML, el cual es posible ejecutarlo en dispositivos con Android, IOS y la web en general. Vale decir, ¡SOLAMENTE PROGRAMAS UNA VEZ!, y puedes ejecutarlo en cualquier dispositivo que la soporte. Otros frameworks que realizan lo mismo, son: React Native (Facebook Inc), Vue Native (Vue Native), entre otros.

En lecciones anteriores, has podido concluir acerca del SDK y NDK para Android, en nuestro caso, utilizaremos el SDK.



3. Instalación de Android Studio

El proceso de instalación es bastante simple. Sin embargo, debes considerar algunas características mínimas, para que tu equipamiento funcione correctamente. De acuerdo con (Google Inc.), los requerimientos mínimos para la versión 2022.3.1, la cual, hasta la emisión de este documento, puedes encontrarlo en: <https://developer.android.com/studio>, en donde de acuerdo con la siguiente ilustración, debes poseer:

System requirements			
Windows	Mac	Linux	Chrome OS
<ul style="list-style-type: none">• 64-bit Microsoft® Windows® 8/10• x86_64 CPU architecture; 2nd generation Intel Core or newer, or AMD CPU with support for a Windows Hypervisor• 8 GB RAM or more• 8 GB of available disk space minimum (IDE + Android SDK + Android Emulator)• 1280 x 800 minimum screen resolution	<ul style="list-style-type: none">• MacOS® 10.14 (Mojave) or higher• ARM-based chips, or 2nd generation Intel Core or newer with support for Hypervisor.Framework• 8 GB RAM or more• 8 GB of available disk space minimum (IDE + Android SDK + Android Emulator)• 1280 x 800 minimum screen resolution	<ul style="list-style-type: none">• Any 64-bit Linux distribution that supports Gnome, KDE, or Unity DE; GNU C Library (glibc) 2.31 or later.• x86_64 CPU architecture; 2nd generation Intel Core or newer, or AMD processor with support for AMD Virtualization (AMD-V) and SSE3• 8 GB RAM or more• 8 GB of available disk space minimum (IDE + Android SDK + Android Emulator)• 1280 x 800 minimum screen resolution	<p>For information on recommended devices and specifications, as well as Android Emulator support, visit chromeos.dev.</p>

Ilustración 1: requerimientos mínimos de instalación

Fuente: (GOOGLE INC.)

En caso de que no cuentes con el equipamiento necesario, entonces deberás buscar alternativas de equipamiento, como por ejemplo máquinas virtuales en amazon (www.awseducate.com), para que puedas realizar las actividades. Para ello, te puedes inscribir como estudiante, con tu cuenta educativa, y te asignarán una cantidad de dólares para que puedas instalar una máquina virtual con



Windows. **IMPORTANTE:** Procura dejar la máquina apagada cuando no la estés utilizando.

Puedes revisar desde la página del desarrollador (Google Inc., 2021), en donde hay un tutorial de instalación. En general, debes descargar, ejecutar, y seguir las recomendaciones que vengan por defecto.

4. Lenguaje de programación KOTLIN

Para utilizar una herramienta IDE, primero debemos conocer cómo funciona el compilador de kotlin, llamado kotlinc (Kotlin Compiler), el cual puede ser descargado desde (Kotling Foundation, 2021).

De acuerdo con las instrucciones, simplemente descargas el archivo zip desde la [página oficial](#) indicada en la bibliografía, y luego lo descomprimes en una carpeta, como indica la siguiente imagen:

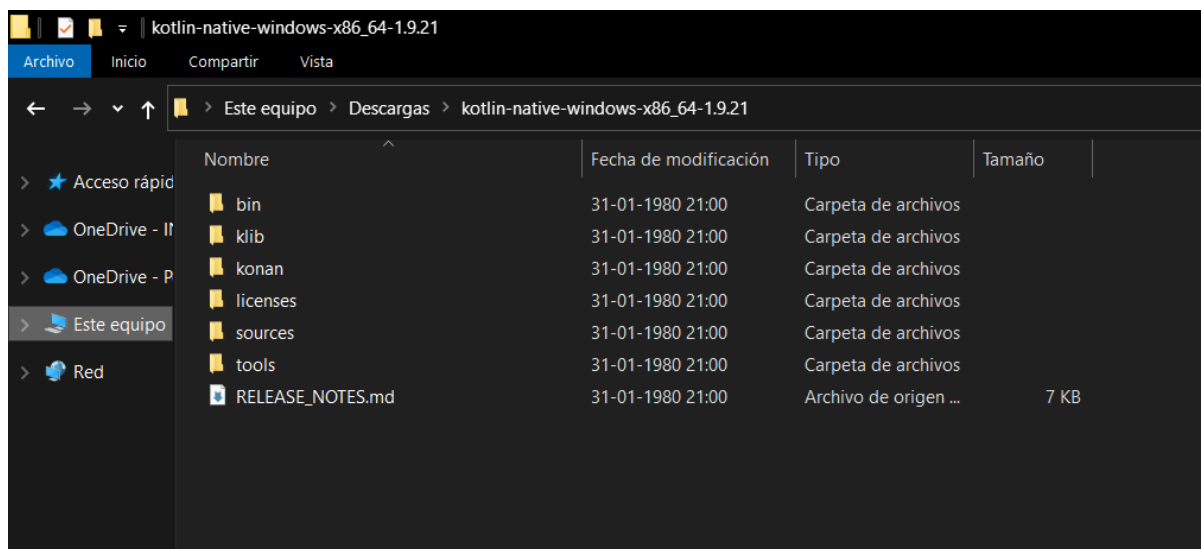
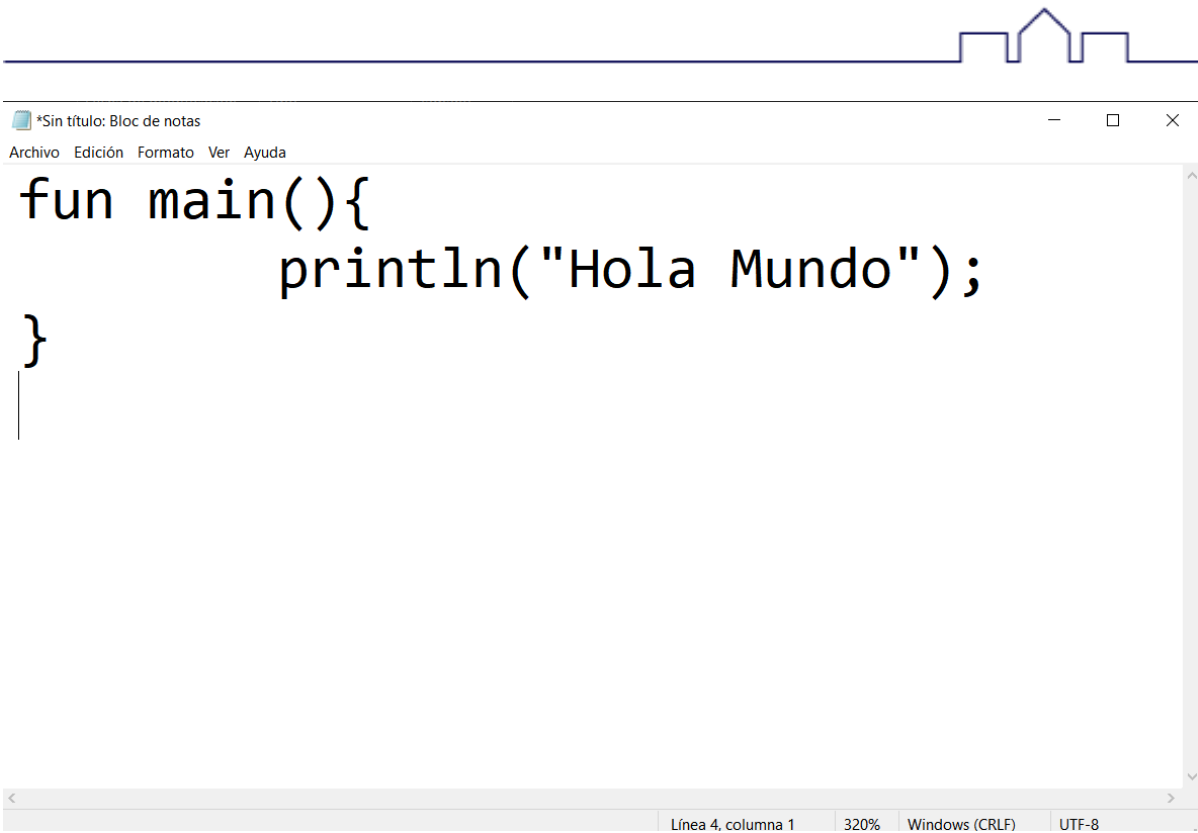


Ilustración 2: Carpeta Kotlin descomprimida

Fuente: Elaboración Propia



```
fun main(){
    println("Hola Mundo");
}
```

Ilustración 3: Código fuente kotlin hola mundo

Fuente: Elaboración propia

Procura dejar el archivo fuente dentro de la carpeta que descomprimiste, para no tener que utilizar la ruta completa

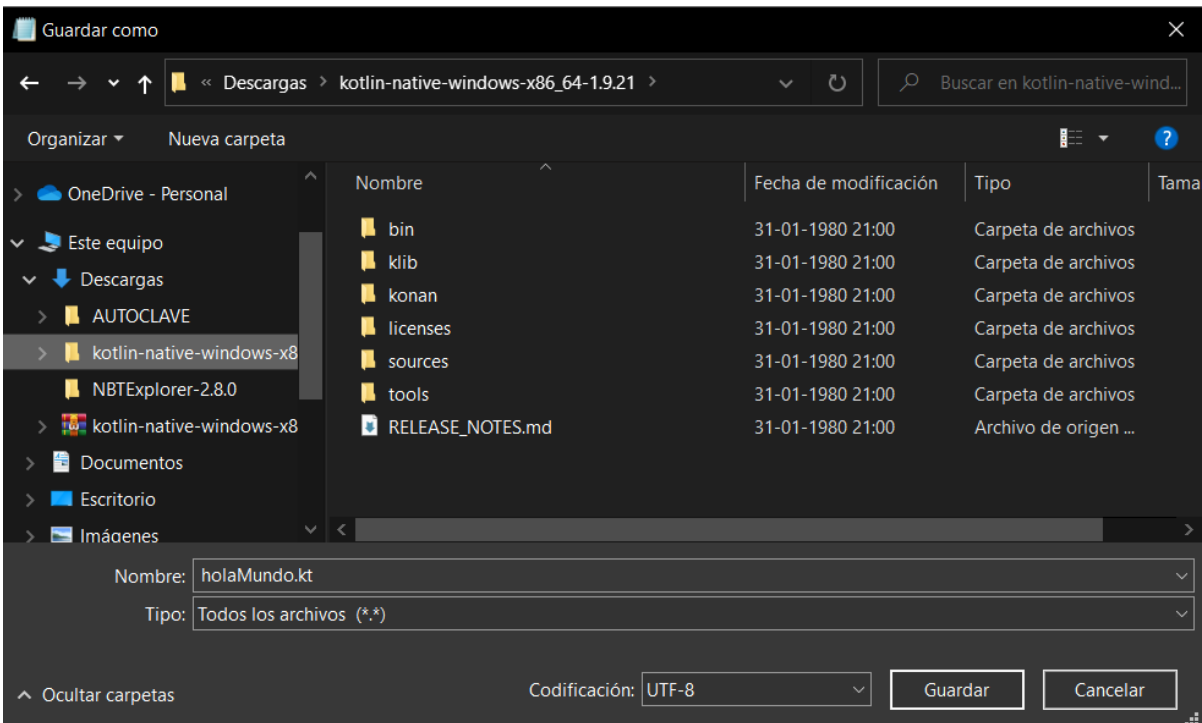


Ilustración 4: Guardar código fuente desde bloc de notas

Fuente: Elaboración propia.

Para compilar el archivo, debes ejecutar el compilador que está dentro de la carpeta bin. Para ello, desde una consola, ingresa a la carpeta en donde se encuentra el archivo KT creado previamente, y compila por medio del comando: " bin\kotlinc holamundo.kt "

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.19045.3693]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\sebas\Downloads\kotlin-native-windows-x86_64-1.9.21>bin\kotlinc-native.bat holaMundo.kt
```

Ilustración 5: compilación de código fuente kotlin

Fuente: Elaboración propia.

Una vez que se haya ejecutado, kotlin descargará automáticamente las dependencias para poder compilar el proyecto (Procura tener 1G disponible para la instalación).

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.19045.3693]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\sebas\Downloads\kotlin-native-windows-x86_64-1.9.21\bin\kotlinc-native.bat holaMundo.kt
Downloading native dependencies (LLVM, sysroot etc). This is a one-time action performed only on the first run of the compiler.

(KonanProperties) Downloading dependency: https://download.jetbrains.com/kotlin/native/lldb-2-windows.zip (0/54796930).
(KonanProperties) Downloading dependency: https://download.jetbrains.com/kotlin/native/lldb-2-windows.zip (24911329/54796930).
(KonanProperties) Downloading dependency: https://download.jetbrains.com/kotlin/native/lldb-2-windows.zip (53020777/54796930).
(KonanProperties) Downloading dependency: https://download.jetbrains.com/kotlin/native/lldb-2-windows.zip (54796930/54796930). Done.
Extracting dependency: C:\Users\sebas\.konan\dependencies\cache\lldb-2-windows.zip into C:\Users\sebas\.konan\dependencies

(KonanProperties) Downloading dependency: https://download.jetbrains.com/kotlin/native/lld-12.0.1-windows-x64.zip (0/22630140).
(KonanProperties) Downloading dependency: https://download.jetbrains.com/kotlin/native/lld-12.0.1-windows-x64.zip (22630140/22630140). Done.
Extracting dependency: C:\Users\sebas\.konan\dependencies\cache\lld-12.0.1-windows-x64.zip into C:\Users\sebas\.konan\dependencies

(KonanProperties) Downloading dependency: https://download.jetbrains.com/kotlin/native/msys2-mingw-w64-x86_64-2.zip (0/135111082).
(KonanProperties) Downloading dependency: https://download.jetbrains.com/kotlin/native/msys2-mingw-w64-x86_64-2.zip (25690112/135111082).
(KonanProperties) Downloading dependency: https://download.jetbrains.com/kotlin/native/msys2-mingw-w64-x86_64-2.zip (51476952/135111082).
(KonanProperties) Downloading dependency: https://download.jetbrains.com/kotlin/native/msys2-mingw-w64-x86_64-2.zip (71513008/135111082).
(KonanProperties) Downloading dependency: https://download.jetbrains.com/kotlin/native/msys2-mingw-w64-x86_64-2.zip (97517568/135111082).
(KonanProperties) Downloading dependency: https://download.jetbrains.com/kotlin/native/msys2-mingw-w64-x86_64-2.zip (117979608/135111082).
(KonanProperties) Downloading dependency: https://download.jetbrains.com/kotlin/native/msys2-mingw-w64-x86_64-2.zip (135111082/135111082). Done.
Extracting dependency: C:\Users\sebas\.konan\dependencies\cache\msys2-mingw-w64-x86_64-2.zip into C:\Users\sebas\.konan\dependencies

(KonanProperties) Downloading dependency: https://download.jetbrains.com/kotlin/native/llvm-11.1.0-windows-x64-essentials.zip (0/156381420).
(KonanProperties) Downloading dependency: https://download.jetbrains.com/kotlin/native/llvm-11.1.0-windows-x64-essentials.zip (25747973/156381420).
(KonanProperties) Downloading dependency: https://download.jetbrains.com/kotlin/native/llvm-11.1.0-windows-x64-essentials.zip (54376939/156381420).
(KonanProperties) Downloading dependency: https://download.jetbrains.com/kotlin/native/llvm-11.1.0-windows-x64-essentials.zip (83829253/156381420).
(KonanProperties) Downloading dependency: https://download.jetbrains.com/kotlin/native/llvm-11.1.0-windows-x64-essentials.zip (113433565/156381420).
(KonanProperties) Downloading dependency: https://download.jetbrains.com/kotlin/native/llvm-11.1.0-windows-x64-essentials.zip (142303749/156381420).
(KonanProperties) Downloading dependency: https://download.jetbrains.com/kotlin/native/llvm-11.1.0-windows-x64-essentials.zip (156381420/156381420). Done.
Extracting dependency: C:\Users\sebas\.konan\dependencies\cache\llvm-11.1.0-windows-x64-essentials.zip into C:\Users\sebas\.konan\dependencies

(KonanProperties) Downloading dependency: https://download.jetbrains.com/kotlin/native/libffi-3.3-windows-x64-1.zip (0/111136).
(KonanProperties) Downloading dependency: https://download.jetbrains.com/kotlin/native/libffi-3.3-windows-x64-1.zip (111136/111136). Done.
Extracting dependency: C:\Users\sebas\.konan\dependencies\cache\libffi-3.3-windows-x64-1.zip into C:\Users\sebas\.konan\dependencies

C:\Users\sebas\Downloads\kotlin-native-windows-x86_64-1.9.21\program.exe
Hola Mundo
```

Ilustración 6: resultados de compilación y ejecución

Fuente: Elaboración propia.

Una vez finalizada la compilación, verás que has creado tu primer archivo ejecutable para Windows (archivo.exe), simplemente ejecuta el archivo program.exe como la ilustración anterior, y estará listo tu primer “HOLA MUNDO”, programado en kotlin, utilizando el compilador.

Ahora, vamos a realizar un ejercicio un poco más complejo que nos permitirá conocer el uso de la programación orientada a objetos desde Kotlin para fabricar programas más complejos.



Para esta actividad, se requiere de un block de notas, en donde escribirás el código inicial de una aplicación en Kotlin, especificando el nombre de la clase, como:

```
class MiClase {  
  
}
```

Para leer datos desde teclado, se utiliza la clase scanner, el cual debe ser "importado", para poder utilizarlo

```
class MiClase {  
    print("Ingresa un número: ")  
    val teclado = readLine()  
    var numero = teclado.toInt()  
    println("El número ingresado es: " + numero)  
}
```

Puedes generar nuevas clases, como, por ejemplo, el de animal, el cual su código, sería:

```
class Animal(var nombre: String, var edad: Int){  
    fun getNombre(): String{ return this.nombre }  
    fun getEdad(): Int{ return this.edad }  
    fun setNombre(nombre: String){ this.nombre = nombre }  
    fun setEdad(edad: Int){ this.edad = edad }  
}
```

Para utilizar esta clase, podríamos crear un objeto desde la función main en el archivo. kt para obtener el nombre y la edad por la misma terminal.

```
fun main(){  
    val mascota = Animal("Perro",12)  
    print(mascota.getNombre)  
    print(mascota.getEdad)  
}
```

5. Lenguaje de programación JAVA

Ahora realizaremos la misma operación, con lenguaje Java.

Para esta actividad, se requiere de un block de notas, en donde escribirás el código inicial de una aplicación java, especificando el nombre de la clase, como:

```
class MiClase {  
}
```

posteriormente, deberás agregar el método main, que reciba como argumento, un arreglo de tipo string. Este método, debe ser público, y no retorna valor.

```
public static void main(){  
  
}
```

El resultado, será como el que indica la siguiente imagen:

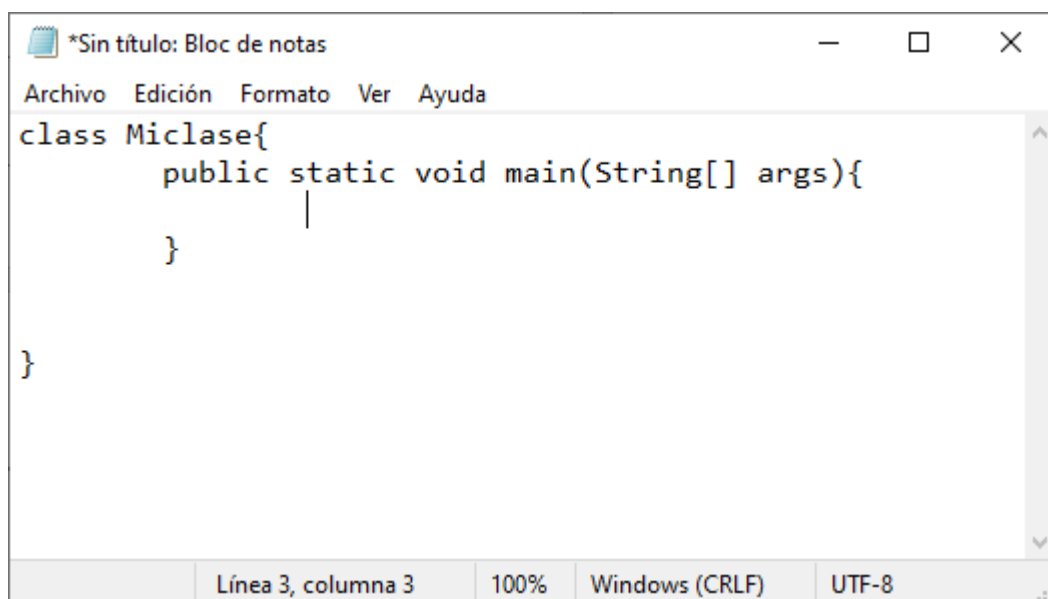
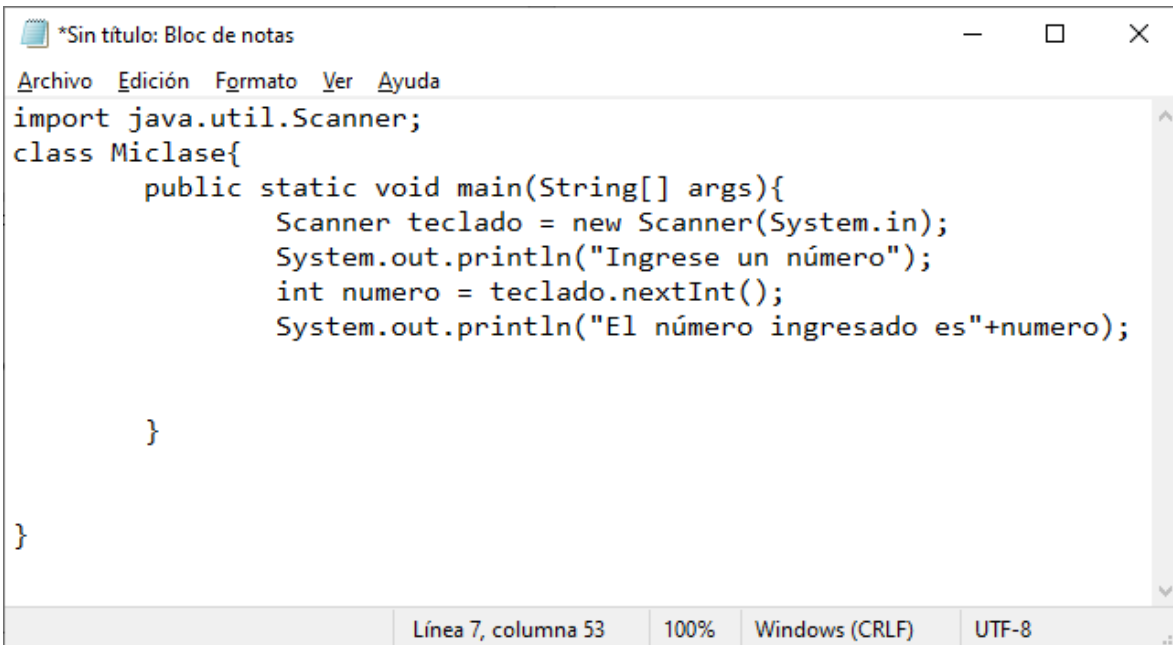


ILUSTRACIÓN 7: RESULTADOS DE COMPILACIÓN Y EJECUCIÓN

FUENTE: ELABORACIÓN PROPIA.

Para leer datos desde teclado, se utiliza la clase scanner, el cual debe ser "importado", para poder utilizarlo



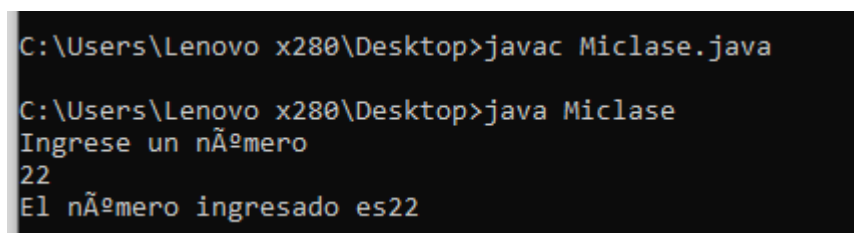
```
*Sin título: Bloc de notas
Archivo Edición Formato Ver Ayuda
import java.util.Scanner;
class Miclase{
    public static void main(String[] args){
        Scanner teclado = new Scanner(System.in);
        System.out.println("Ingresa un número");
        int numero = teclado.nextInt();
        System.out.println("El número ingresado es"+numero);
    }
}
```

Línea 7, columna 53 100% Windows (CRLF) UTF-8

Ilustración 8: Ejemplo interacción con el usuario

Fuente: Elaboración propia.

Guarda el archivo, con extensión .java, compila y ejecuta. El resultado, sería como el de la siguiente imagen:



```
C:\Users\Lenovo x280\Desktop>javac Miclase.java
C:\Users\Lenovo x280\Desktop>java Miclase
Ingresa un número
22
El número ingresado es22
```

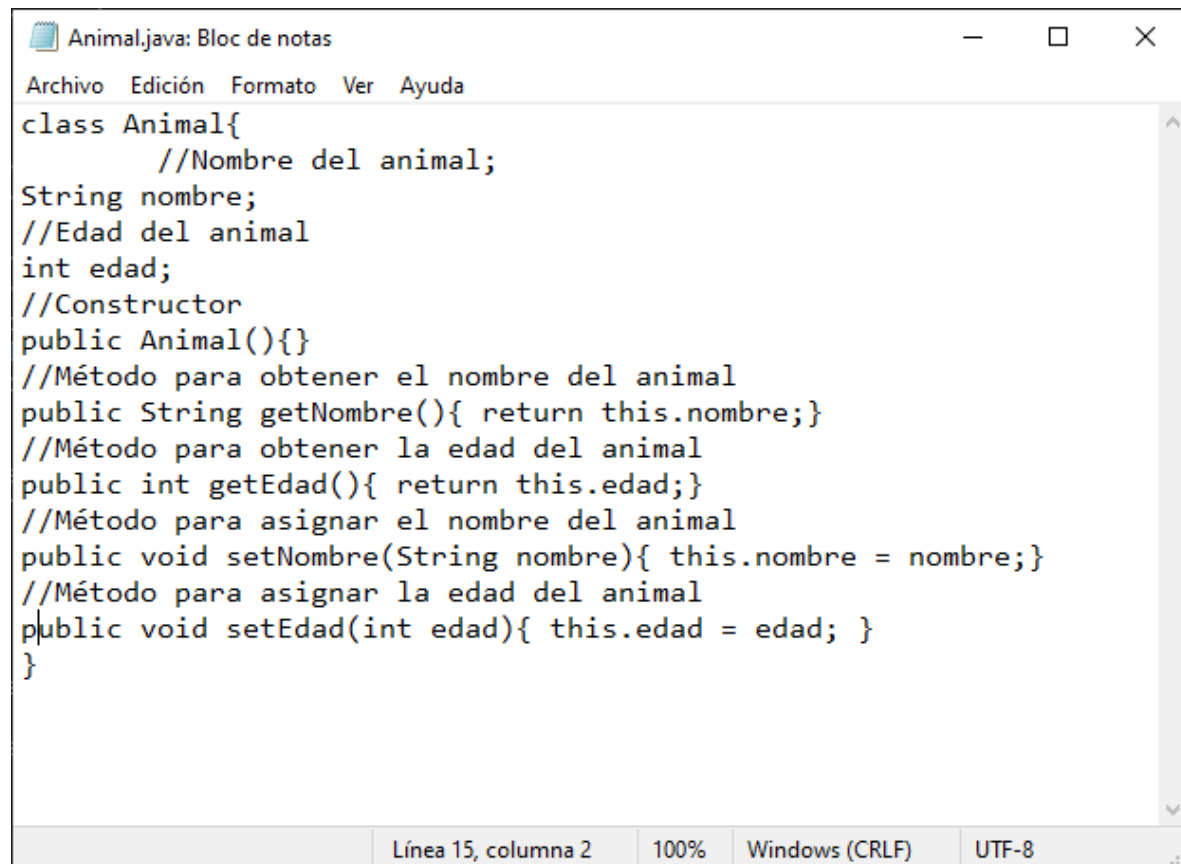
ILUSTRACIÓN 9: COMPILACIÓN Y EJECUCIÓN JAVA

Puedes utilizar los ciclos (while, for, do while), y decisiones if/else, switch, por ejemplo:

Del código anterior:

```
If(numero<10){  
System.out.println("El número es inferior a 10");  
}
```

Puedes generar nuevas clases, como por ejemplo, el de animal, el cual su código, sería:

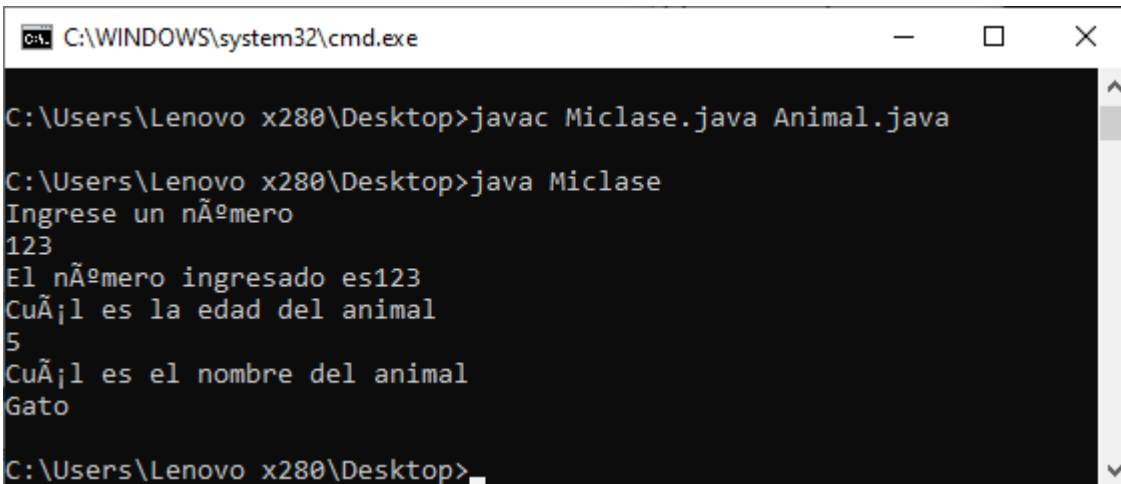


```
Animal.java: Bloc de notas  
Archivo Edición Formato Ver Ayuda  
class Animal{  
    //Nombre del animal;  
String nombre;  
    //Edad del animal  
int edad;  
    //Constructor  
public Animal(){}  
    //Método para obtener el nombre del animal  
public String getNombre(){ return this.nombre;}  
    //Método para obtener la edad del animal  
public int getEdad(){ return this.edad;}  
    //Método para asignar el nombre del animal  
public void setNombre(String nombre){ this.nombre = nombre;}  
    //Método para asignar la edad del animal  
public void setEdad(int edad){ this.edad = edad; }  
}
```

Línea 15, columna 2 100% Windows (CRLF) UTF-8

ILUSTRACIÓN 10: EJEMPLO DE CLASE

Para utilizar esta clase, podríamos crear un objeto desde el método main (por ejemplo), quedando de la siguiente manera:



```
C:\WINDOWS\system32\cmd.exe

C:\Users\Lenovo x280\Desktop>javac Miclase.java Animal.java

C:\Users\Lenovo x280\Desktop>java Miclase
Ingresa un número
123
El número ingresado es 123
Cual es la edad del animal
5
Cual es el nombre del animal
Gato

C:\Users\Lenovo x280\Desktop>
```

ILUSTRACIÓN 11: EJEMPLO COMPILACIÓN DOS ARCHIVOS

Como resultado, puedes crear clases en archivos distintos, para poder utilizarlos por medio de objetos, desde otras clases.

6. Codificación de una aplicación básica con el Android SDK

Como hemos visto en puntos anteriores, puedes compilar una aplicación sin tener una herramienta IDE. Con Android SDK, también lo puedes realizar, por medio del comando GRADLEW. Si deseas obtener más información de cómo compilar desde la línea de comandos, puedes revisar la página oficial de Android: <https://developer.android.com/studio/build/building-cmdline>

Pues ahora, ha llegado la hora de crear nuestra primera aplicación. Para ello, debes:

- a. Abrir Android Studio, y luego "Create New Project"

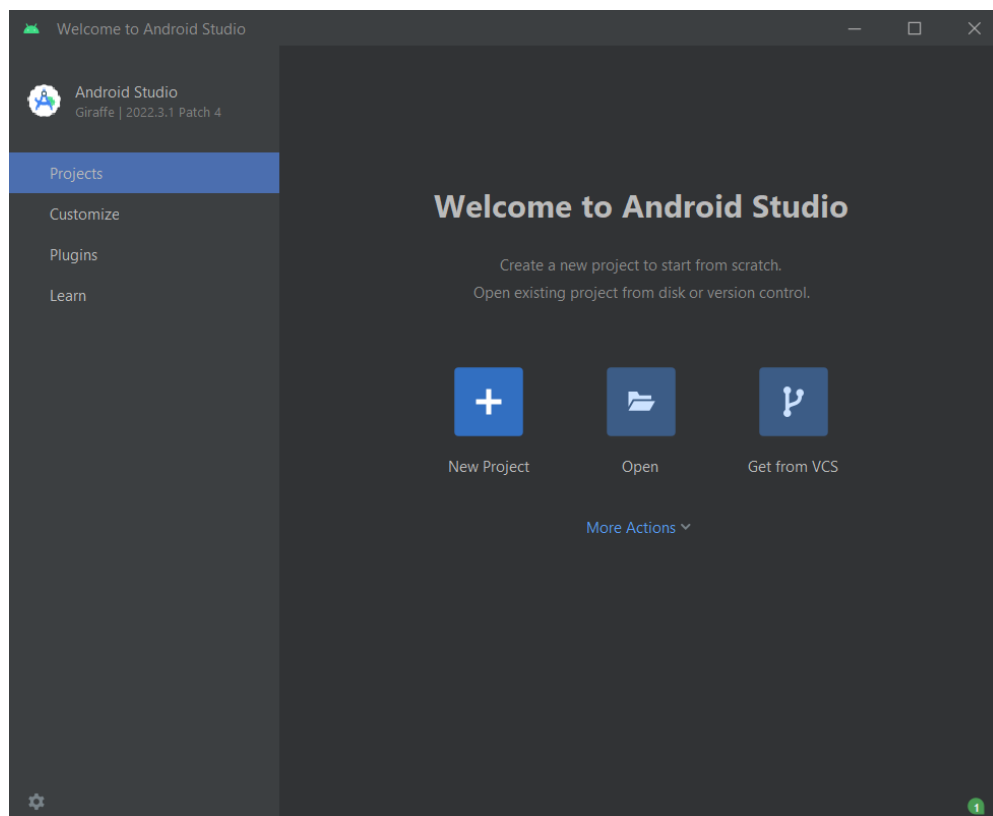


ILUSTRACIÓN 12: INICIO ANDROID STUDIO

-
- b. A continuación, crearemos un “Empty Activity”, para que nos entregue una pantalla y código con el mínimo de opciones, como demuestra la siguiente imagen:

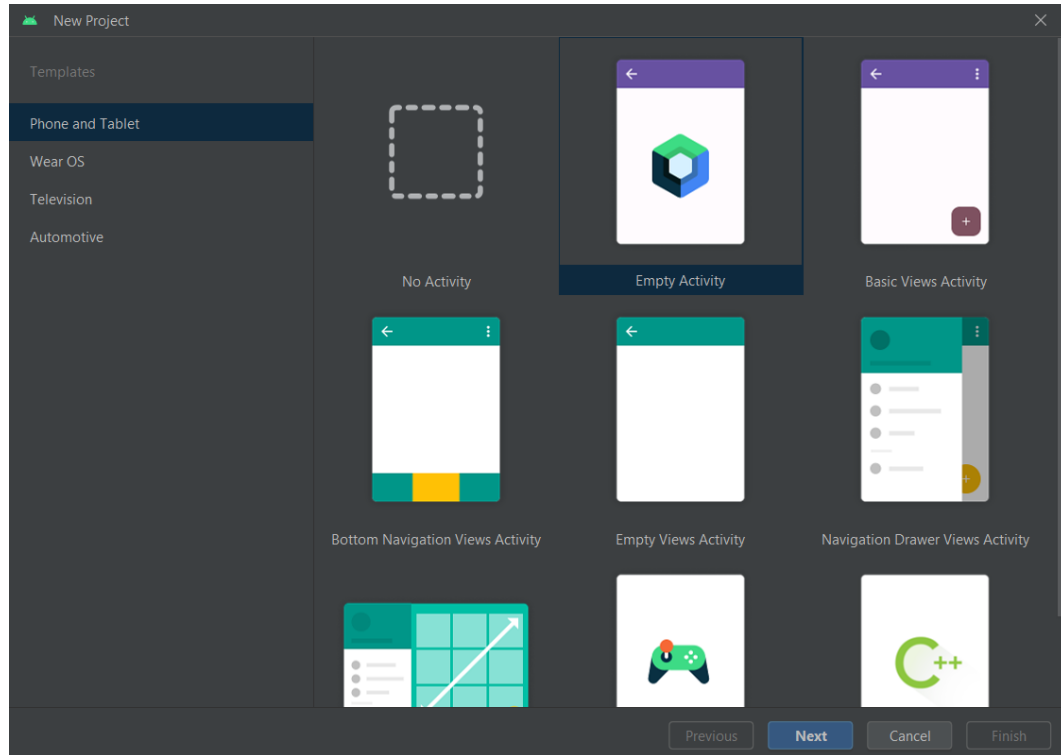


ILUSTRACIÓN 13: SELECCIÓN DE ACTIVITY VACÍO

- c. Asigna un nombre a tu proyecto, especificando luego el nombre del package utilizando la nomenclatura (dominio.tuempresa.tuproyecto) por ejemplo: `cl.miles.miprimeraapp`, como demuestra la siguiente ilustración. Procura seleccionar como SDK mínimo, la versión en donde vayas a instalar tu aplicación. Si seleccionas la versión 7 de Android y tu dispositivo tiene la versión 6, entonces no podrás instalar la aplicación en dicho dispositivo.

IMPORTANTE: La ruta en donde se guardará el proyecto, no debe contener espacios, de lo contrario, no dejará finalizar.

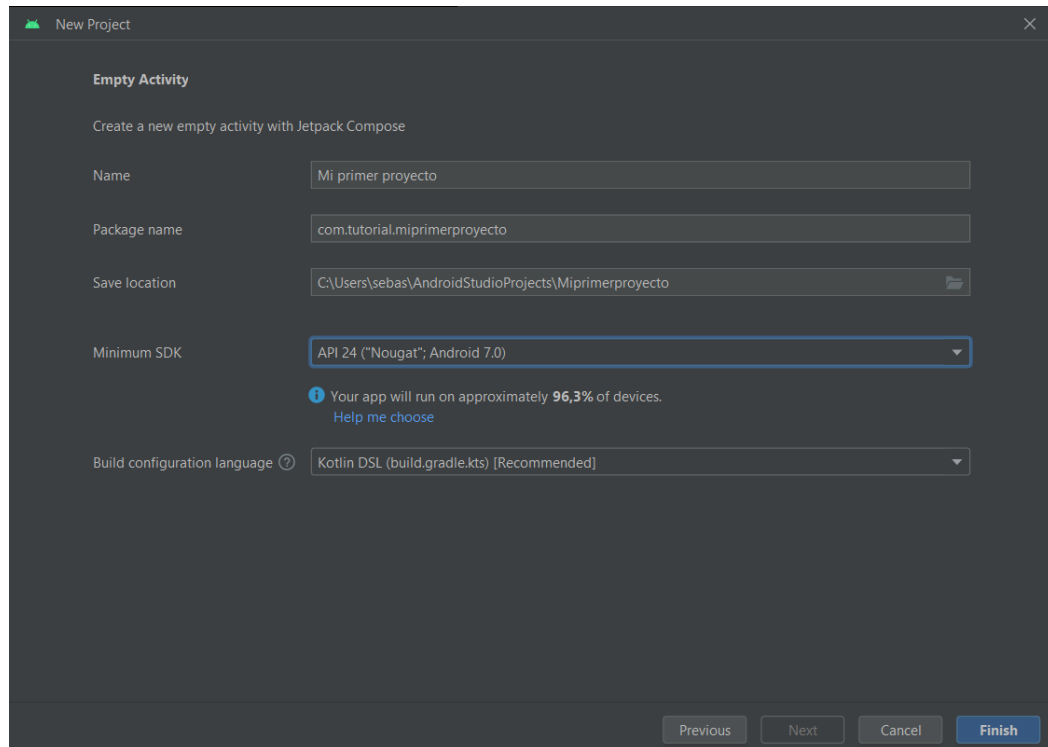


ILUSTRACIÓN 14. CONFIGURACIÓN DE PARÁMETROS DEL PROYECTO

- d. Si deseas analizar qué versión debes utilizar, puedes presionar el hipervínculo “Help me choose”, y te mostrará la distribución de versiones que utilizan los usuarios en el mundo, como indica la siguiente ilustración.

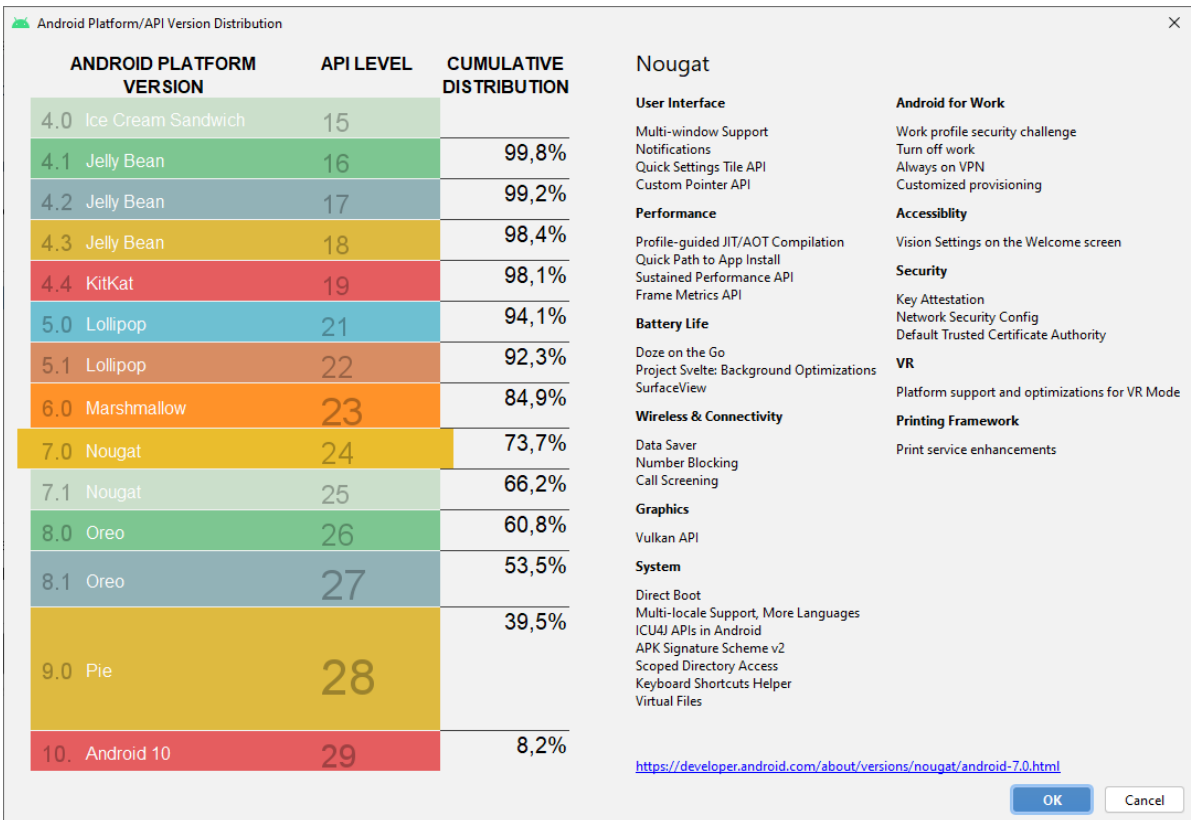


ILUSTRACIÓN 15: DISTRIBUCIÓN DE PLATAFORMAS DE ANDROID EN PORCENTAJE

- e. Finaliza la creación del proyecto
- f. Realiza el punto 8 de este documento, habilitando las opciones de desarrollador, y habilitando la depuración USB del dispositivo.
- g. Una vez realizado lo anterior, conecta tu dispositivo Android a un puerto USB del computador, y Android studio lo reconocerá para poder instalar tu primera aplicación.
- h. Si todo está correctamente instalado y configurado, entonces al lado del botón "Play", aparecerá el nombre de tu dispositivo celular. Si no aparece tu dispositivo, se puede deber a:
 - El cable USB está algún cable (como el de datos), en mal estado. Solución, prueba otro cable de mejor calidad.

- No has habilitado las opciones de desarrollador.
- No has habilitado la depuración USB.
- No tienes correctamente instalado los drivers USB de la placa.

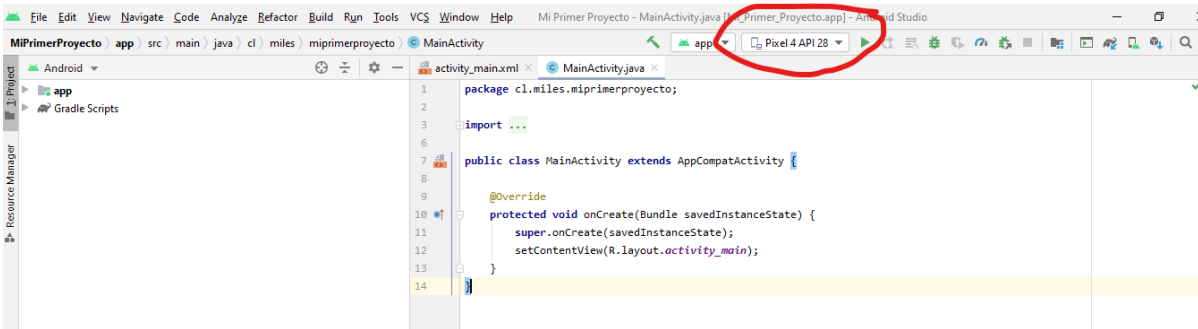


ILUSTRACIÓN 16: IDENTIFICACIÓN DE DISPOSITIVO MÓVIL INSTALADO

Probablemente no tengas algún dispositivo Android para seguir el curso, ya que utilizas IOS, por ejemplo. Puedes utilizar otras alternativas, como Bluestack, o bien algún otro virtualizador. Android studio, también provee una máquina virtual para poder testear las aplicaciones.

Para crear un dispositivo virtual, debes hacer click en **TOOLS > AVD MANAGER**

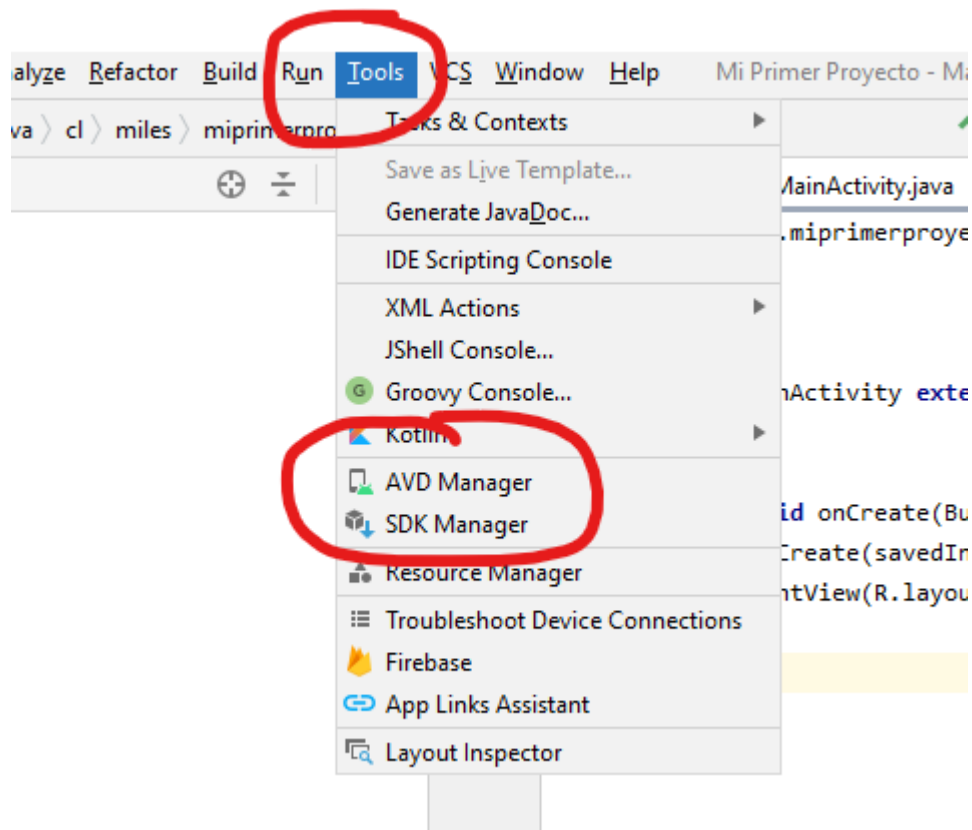


ILUSTRACIÓN 17: MENU AVD MANAGER

Haz click sobre el símbolo “+ Create Virtual Device”

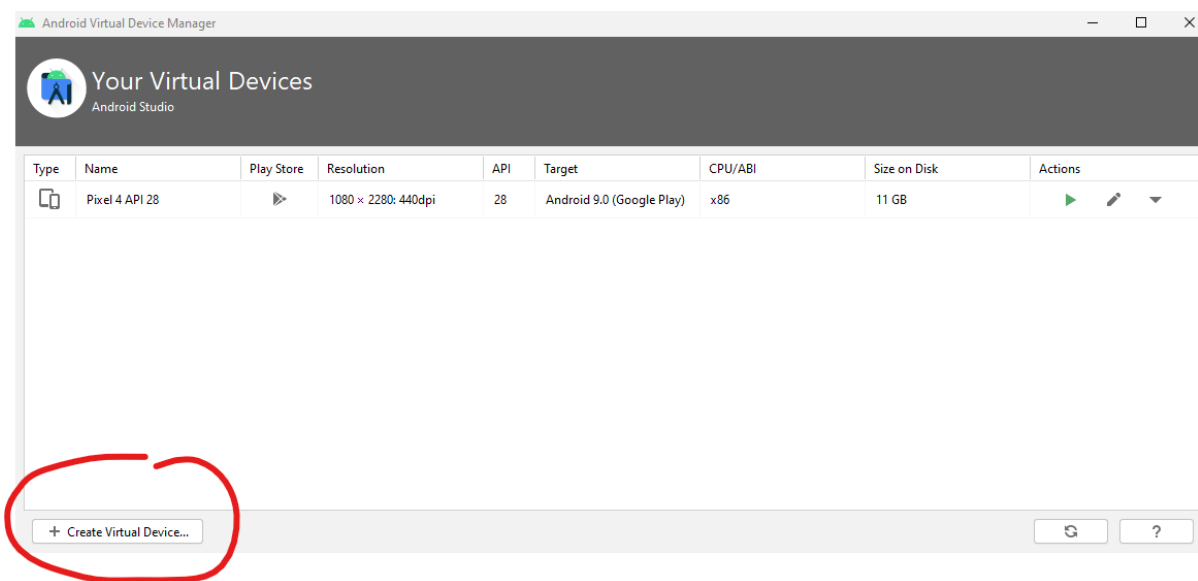


ILUSTRACIÓN 18: CREAR NUEVO VIRTUAL DEVICE

A continuación, selecciona el tipo de dispositivo que quieres crear. Debes considerar el tamaño (de la pantalla por ejemplo), y si deseas, que venga con playstore incluido (debes acceder con una cuenta gmail).

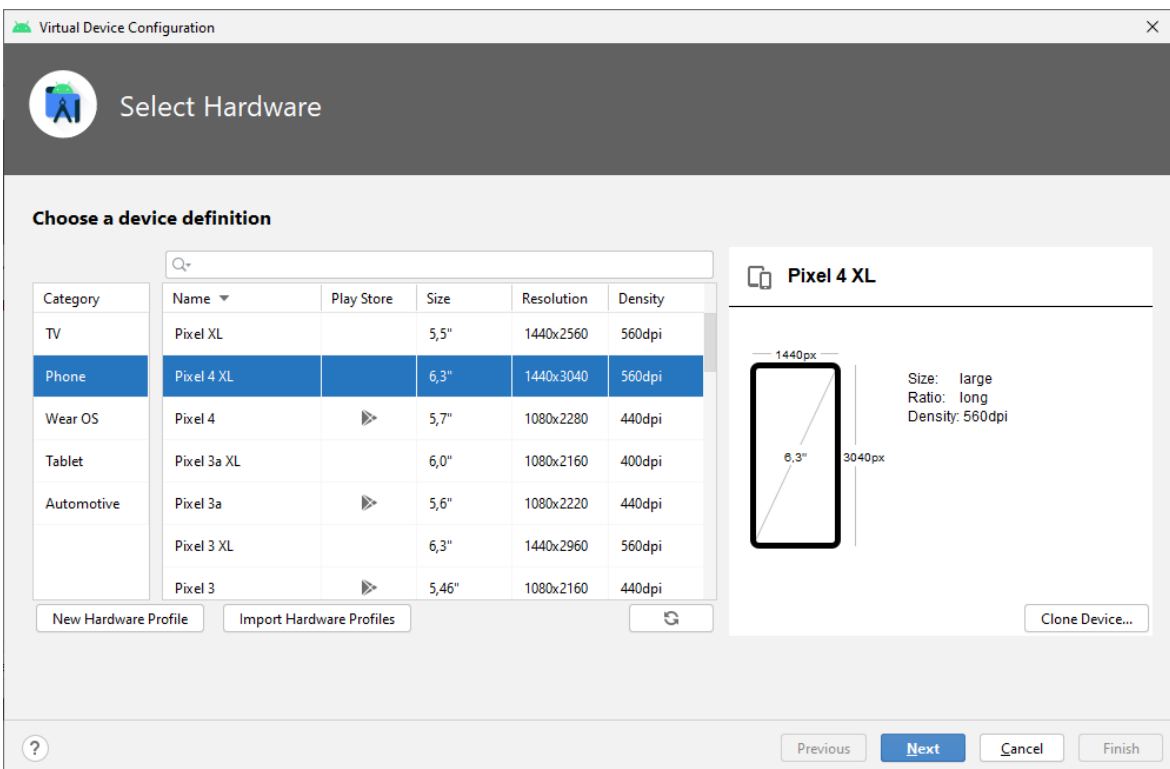


ILUSTRACIÓN 19. SELECCIÓN DE DISPOSITIVO VIRTUAL

Posteriormente, selecciona la versión de Android que deseas instalar en el dispositivo. Te recomiendo que solamente descargues la versión sobre la cual deseas trabajar, ya que utilizará recursos de almacenamiento de tu computador.

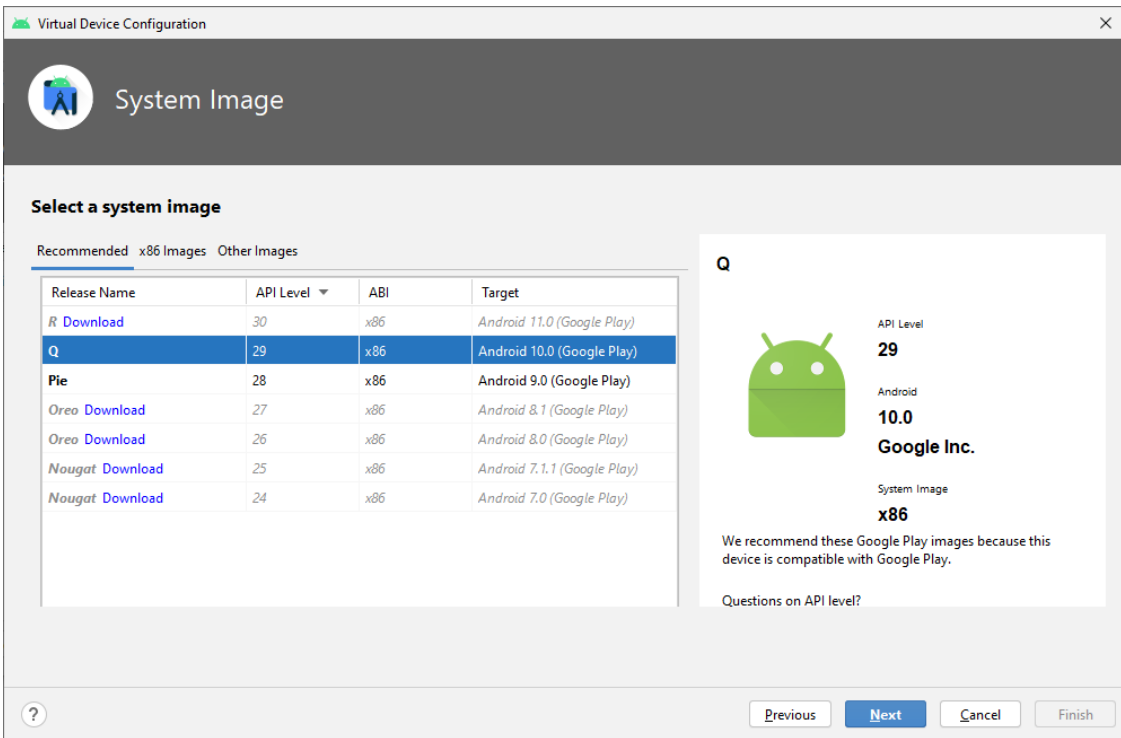


ILUSTRACIÓN 20: SELECCIÓN DE VERSIÓN DE ANDROID PARA AVD

Asigna un nombre al dispositivo virtual, de manera que sea reconocible dentro del programa Android studio, y ya tienes creado tu primer dispositivo Android virtual instalado, para que puedas testear tus aplicaciones. Realiza el punto “h” en este mismo apartado, presionando “play”, y listo, has creado tu primer “hola mundo” en Android.

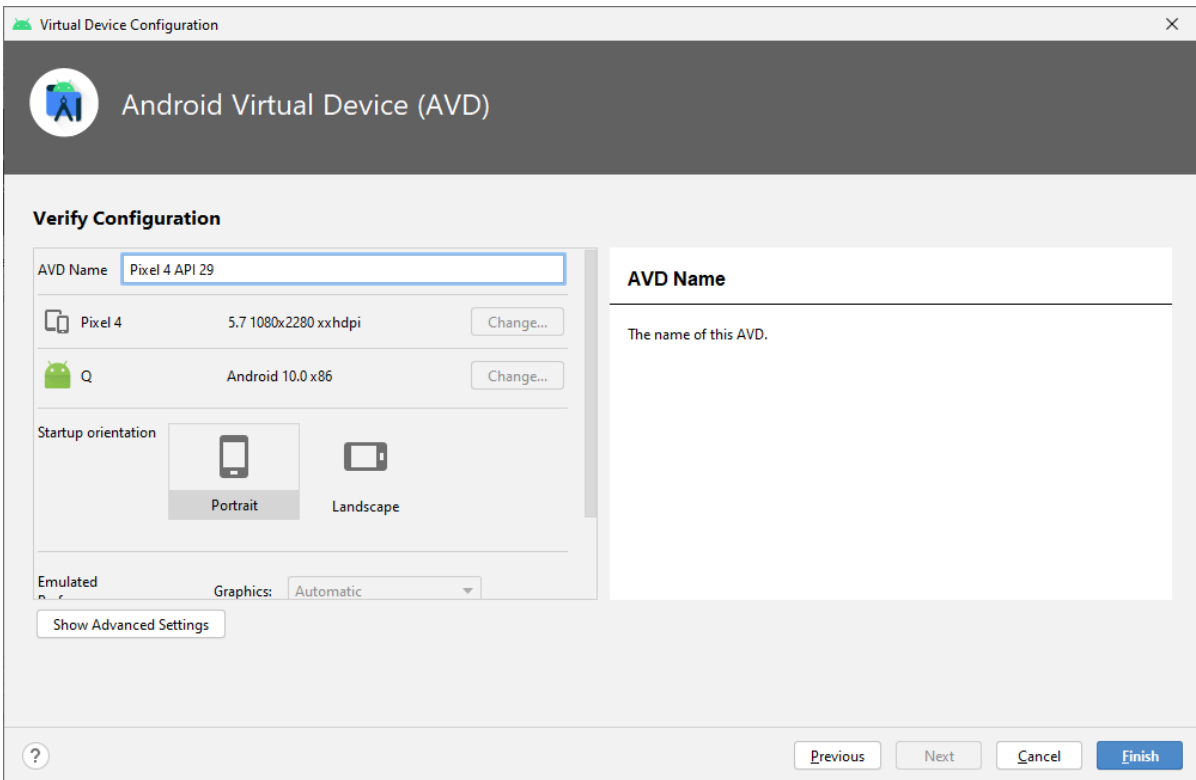


ILUSTRACIÓN 21: CREACION DE AVD. CONFIGURACIÓN DEL DISPOSITIVO

Presta atención al archivo XML en donde aparece el hola mundo. Intenta cambiar el texto de inicio.

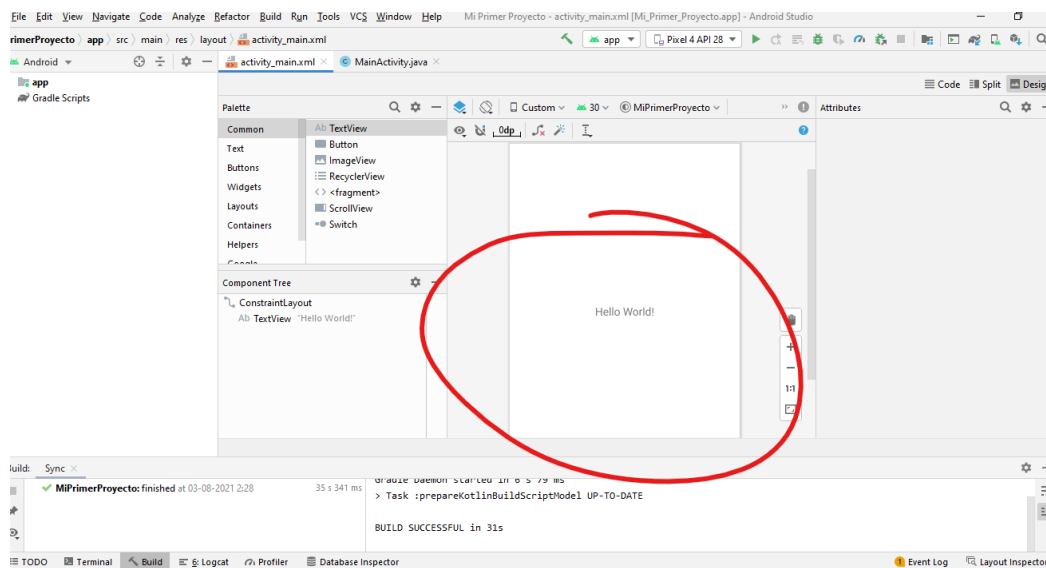


ILUSTRACIÓN 22: HOLA MUNDO EN ANDROID


Una vez que hayas instalado el dispositivo virtual, presiona el botón play, para iniciar la primera aplicación que viene por defecto.



Intenta ejecutar esta aplicación en tu dispositivo móvil (físico), habilitando las opciones de desarrollador, y dentro de las opciones, habilita la depuración por USB.

7. Configuración de un dispositivo móvil con Android Debug Bridge ADB

“Android Debug Bridge (adb) es una herramienta de línea de comandos que permite comunicarte con un dispositivo. El comando adb permite realizar una variedad de acciones en el dispositivo, como instalar y depurar apps, y proporciona acceso a un shell de Unix que puedes usar para ejecutar distintos comandos en un dispositivo. Es un programa cliente-servidor que incluye tres componentes:

-
- 
- Un cliente, que envía comandos. El cliente se ejecuta en tu máquina de desarrollo. Puedes invocar un cliente desde un terminal de línea de comandos emitiendo un comando adb.
 - Un daemon (adbd), que ejecuta comandos en un dispositivo. El daemon se ejecuta como un proceso en segundo plano en cada dispositivo.
 - Un servidor, que administra la comunicación entre el cliente y el daemon. El servidor se ejecuta en tu máquina de desarrollo como un proceso en segundo plano.

Adb está incluido en el paquete de herramientas de la plataforma de Android SDK. Puedes descargar este paquete con SDK Manager, que lo instala en `android_sdk/platform-tools/`. O, si quieres el paquete independiente de herramientas de la plataforma del SDK de Android.”

Fuente: (Google Inc.)

8. Configuración de un dispositivo físico para depuración de aplicaciones

Cada versión de Android posee una secuencia para poder habilitar las opciones de desarrollador.

Para conocer más acerca de cómo habilitar las opciones de desarrollador, puedes visitar la página oficial de Android

“En Android 4.1 y versiones anteriores, la pantalla Opciones para desarrolladores está disponible de forma predeterminada. En Android 4.2 y versiones posteriores, debes habilitarla. Si quieres habilitar las Opciones para desarrolladores, presiona la opción Número de compilación 7 veces. Puedes



encontrar esta opción en una de las siguientes ubicaciones, según tu versión de Android:

- Android 9 (nivel de API 28) y versiones posteriores: Configuración > Acerca del dispositivo > Número de compilación
- Android 8.0.0 (API nivel 26) y Android 8.1.0 (API nivel 26): Configuración > Sistema > Acerca del dispositivo > Número de compilación
- Android 7.1 (nivel de API 25) y versiones anteriores: Configuración > Acerca del dispositivo > Número de compilación"

Fuente: (Google Inc, 2021)

Una vez que hayas habilitado las opciones de desarrollador, entonces debes habilitar la opción de Depuración por USB, para que se pueda conectar el ADB y poder instalar los archivos APK.

"Para poder usar el depurador y otras herramientas, debes habilitar la depuración por USB, que permite que Android Studio y otras herramientas del SDK reconozcan tu dispositivo cuando se conecta mediante USB. A fin de habilitar la depuración por USB, activa o desactiva la opción depuración por USB en el menú Opciones para desarrolladores. Puedes encontrar esta opción en una de las siguientes ubicaciones, según tu versión de Android:

Android 9 (nivel de API 28) y versiones posteriores: Configuración > Sistema > Avanzado > Opciones para desarrolladores > Depuración por USB



Android 8.0.0 (API nivel 26) y Android 8.1.0 (API nivel 26): Configuración > Sistema > Opciones para desarrolladores > Depuración por USB

Android 7.1 (API nivel 25) y versiones anteriores: Configuración > Opciones para desarrolladores > Depuración por USB"

Obtenido desde (Google Inc, 2021)

9. Emuladores Android en línea

Existen muchas aplicaciones para poder desarrollar y testear nuestras aplicaciones. De acuerdo con (Nieto, 2019), puedes acceder a APKONLINE, para poder ejecutar aplicaciones Android. Para ello, debes acceder a <https://www.apkonline.net/>, acceder a “Free Android Emulator”, y posteriormente comenzar a utilizar, como indica la siguiente ilustración:

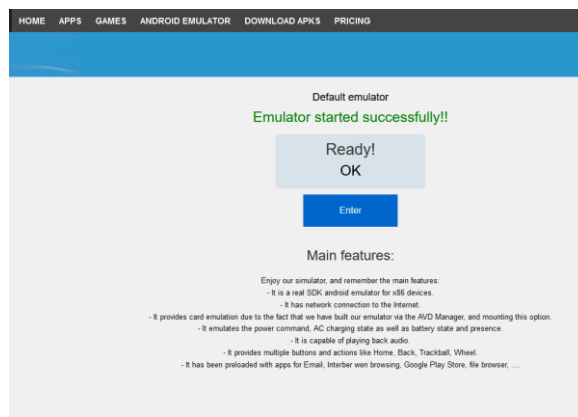


Ilustración 23: Inicialización de apkonline

Fuente: Elaboración propia.

Una vez que presionas enter, ahora tienes un dispositivo Android OnLine, como demuestra la siguiente figura:

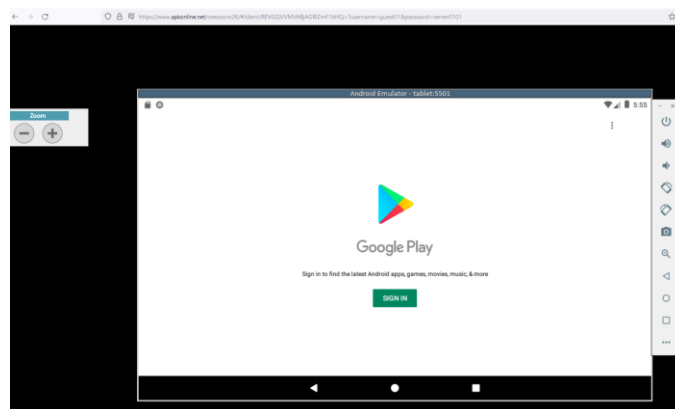


Ilustración 24: EMULADOR ONLINE ANDROID

FUENTE: ELABORACIÓN PROPIA.

10. Depuración de una aplicación Android

La opción debug de Android studio, permite realizar las siguientes acciones, para poder identificar problemas de codificación:

- Seleccionar un dispositivo en el cual depurarás tu app
- Establecer interrupciones en tu código Java, Kotlin y C/C++
- Examinar variables y evaluar expresiones en el tiempo de ejecución

El uso de estas funcionalidades de las herramientas, te permite ir haciendo la "traza" del código, pudiendo ir línea a línea, viendo el contenido de las variables, en tiempo de ejecución del programa.

En relación a la documentación, en donde indica que:

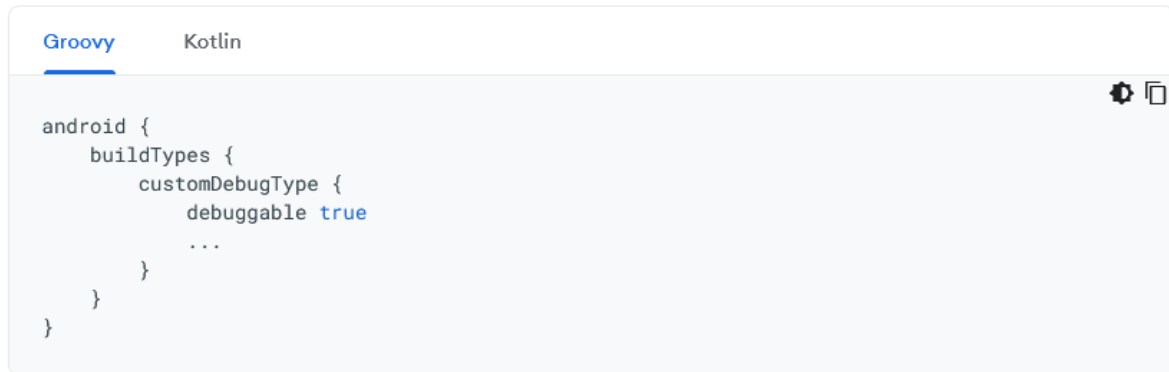
Habilita la depuración en tu dispositivo:

Si estás usando el emulador, esta opción estará activada de forma predeterminada. Sin embargo, en el caso de un dispositivo conectado, deberás habilitar la depuración en las opciones para desarrolladores del dispositivo.

Ejecuta una variante de compilación depurable:

Debes usar una variante de compilación que incluya `debuggable true` en la configuración de compilación. Por lo general, simplemente puedes seleccionar la variante "debug" predeterminada que se incluye en cada proyecto de Android Studio (aunque no esté visible en el archivo

build.gradle). Sin embargo, si defines nuevos tipos de compilación que deberían ser depurables, debes agregar "debuggable true" al tipo de compilación de la siguiente manera:

A screenshot of an IDE window showing a Groovy file. The window has tabs for 'Groovy' and 'Kotlin'. The code is for an Android build configuration. It shows a nested structure: 'android' block containing a 'buildTypes' block, which contains a 'customDebugType' block with 'debuggable true'. There are also ellipses indicating more code. The IDE has a light blue theme and a settings icon in the top right corner.

```
android {  
    buildTypes {  
        customDebugType {  
            debuggable true  
            ...  
        }  
    }  
}
```

Ilustración 25: Habilidad de depuración

Fuente: (Google Inc., 2021)

Puedes encontrar mayor información, en la página oficial de Android developers:

<https://developer.android.com/studio/debug>

11. Utilización de la documentación desde Android Developer.

La documentación que proporciona (Google Inc.), es bastante simple de leer. Esta documentación proporciona ejemplos de código, y la completa descripción de métodos y clases que podemos utilizar. Mucha documentación, también se encuentra en los tutoriales de la herramienta (ejemplo: firebase), el cual utilizaremos en futuras lecciones.





Conclusiones

Para el desarrollo de aplicaciones móviles, es indispensable que selecciones el lenguaje que más te acomode, sin embargo, debes considerar la cantidad de información disponible en la red, para cada lenguaje, además que sea actualizada.

Es muy importante consultar las referencias del lenguaje, directamente desde el sitio de Android Studio, en donde entrega una completa descripción acerca del uso, con algunos ejemplos.



Referencias

- Eclipse Foundation. (s.f.). *Eclipse for Android Developers*. Obtenido de <https://www.eclipse.org/downloads/packages/>
- Facebook Inc. (s.f.). *React Native*. Obtenido de <https://reactnative.dev/>
- GNU. (s.f.). *GNU Project*. Obtenido de Emacs: <https://www.gnu.org/software/emacs/>
- Google Inc. (01 de 08 de 2021). *Habilitar Opciones de desarrollador*. Obtenido de Guía de usuario: <https://developer.android.com/studio/debug/dev-options?hl=es-419>
- Google Inc. (02 de 08 de 2021). *Android Studio*. Obtenido de Guía del Usuario: <https://developer.android.com/studio/install>
- Google Inc. (s.f.). *Android Developers*. Obtenido de <http://developer.android.com>
- IONIC Framework. (s.f.). *Ionic*. Obtenido de <https://ionicframework.com/>
- Kochan, S. G. (2013). *Programming in Objective-C* (6 ed.). Addison-Wesley Professional.
- Kotling Foundation. (03 de 08 de 2021). *Kotlin Lang Command Line*. Obtenido de <https://kotlinlang.org/docs/command-line.html>
- Meijer, E., & Drayton, P. (02 de 01 de 2004). Static Typing Where Possible, Dynamic Typing When Needed: The End of the Cold War Between Programming Languages. Obtenido de <https://web.archive.org/web/20070216025556/http://pico.vub.ac.be/~wdmeuter/RDL04/papers/Meijer.pdf>
- Microsoft corp. (s.f.). *VSCode*. Obtenido de <https://code.visualstudio.com/>
- Miles Avello, J. I. (Productor). (s.f.). *Aprende Java Classes en una hora (y cinco minutos)* [Película]. Obtenido de <https://youtu.be/1Fm9Ys9uio0>
- Nieto, J. G. (22 de 10 de 2019). *xatakandroid*. Obtenido de APKOnline, un emulador de Android online gratuito para probar apps y juegos desde el navegador : <https://www.xatakandroid.com/aplicaciones-android/apkonline-emulador-android-online-gratuito-para-probar-apps-juegos-navegador>
- Oracle Corp. (s.f.). *JAVA*. Obtenido de Qué es Java: https://www.java.com/es/download/help/whatis_java.html
- Oracle Inc. (2021 de 08 de 02). *Java SE*. Obtenido de <https://www.oracle.com/cl/java/technologies/javase/javase-jdk8-downloads.html>
- Vue Native. (s.f.). *Vue Native*. Obtenido de <https://vue-native.io/>
- WindowsCentral. (06 de 07 de 2011). *WindowsCentral*. Obtenido de <https://www.windowscentral.com/dont-wait-until-windows-11-heres-how-run-android-apps-your-pc-right-now>

