

Diseño y Programación de Dispositivos Móviles

EIF 204

Profesor: Ms.C Gregorio Villalobos Camacho

Correo: gregorio.villalobos.camacho@una.ac.cr

Introducción a Tecnologías Móviles

- La comunicación ha avanzado cada vez más
- Hoy en día tenemos aparatos inteligentes
- Sus formas y tamaños varían
 - Teléfonos
 - Laptops
 - Tables
 - Fablets

Introducción a Tecnologías Móviles

Primeros Dispositivos	Últimas tecnologías
Solo se podía llamar y enviar SMS	Se incluye internet
Eran monocromáticos	Hay una amplia gama de apps
Los primeros tenían baterías externas	Reemplazan multiples dispositivos
Gran tamaño vs pocas funciones	Reproducen Video y audio
Sonido limitado (Midi)	Su tamaño depende de la necesidad

Generaciones: 1G

- Alrededor de 1979
- Solo transmitía voz
- Era tecnología analógica
- La conexión era de baja calidad
- No existía seguridad alguna



Generaciones: 2G

- Alrededor de 1990
- Incluye SMS y MMS (multimedia)
- Inicial tecnologías como
 - GSM
 - GPRS
 - EDGE
- Mejora la calidad



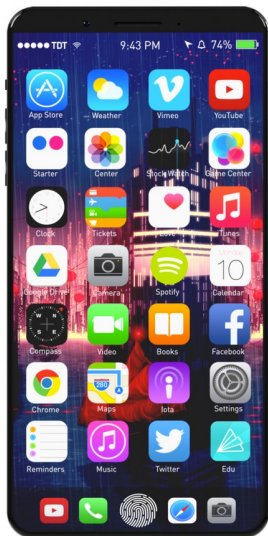
Generaciones: 3G

- Alrededor del 2001
- Se puede hacer video llamadas
- Tiene 2 mb/s
- Inicia el acceso inalámbrico a internet



Generaciones: 4G

- Inicia en el 2010
- 50 veces más veloz que 3G
- Aparatos con más capacidades (Specs)
- Servicio más estable y ubicuo

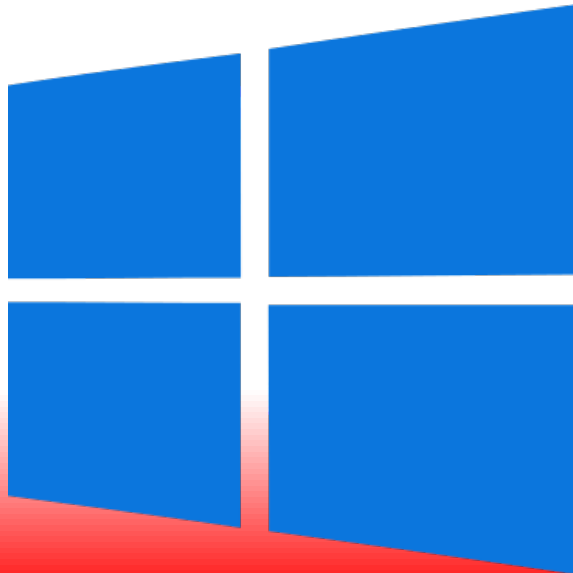


Generaciones: 5G

- Se espera entre el 2020 y el 2030
- Ya se ven prototipos y se discute su red
- Se espera mejor cobertura
- Más dispositivos conectados



Sistemas Operativos Más Comunes



Sistemas Operativos

- Nos sirven para multiples tareas
- Sobre los cuales se instalan aplicaciones
- Nos facilitan el trabajo diario
- Se encargan de mediar hardware y software

Tarea de Investigación

- Realizar la investigación en grupo
- Estudiar un ambiente de desarrollo móvil
- Desarrollar un app pequeño
 - Página de inicio y al menos 2 ventanas de nav.
- Ideas de apps
 - Tic Tac Toe
 - Meme Generator: 4 posiciones, salvar/compartir
 - Image Enhancer: 4 modificaciones, salvar/compartir
 - Capturador de imágenes (Usando la cámara)
 - Cuenta pollos

Tarea de Investigación

- Debe entregar
 - El código fuente
 - Un manual técnico del proyecto
 - La presentación a utilizar
- Se debe presentar el ambiente max 20 minutos
 - Se muestra la herramienta de desarrollo
 - Se hace un demo o proyecto corto
 - Se nos debe enseñar a hacer un primer programa
 - Se detalla y muestra cómo funciona el ambiente
 - Se explica el proceso
 - Facilidades, limitaciones, usos

Tarea de Investigación

- Temas - IDEs

1. Visual Studio + Xamarin
2. XCode + Swift
3. App Inventor
4. Appcelerator Titanium
5. HTML5 (PWA)
6. LiveCode
7. Basic 4 Android
8. Phone Gap
9. Appery.io

- Temas - IDEs

10. AIDE
11. Corona Solar 2D
12. Droid Script
13. Unity 3D
14. Android Web Developer
15. IntelliJ Idea
16. iBuild App
17. Deuter IDE
18. Appcelerator Titanium

Lenguajes Para Apps: iOS

- Swift
 - Todo en IOS se desarrolla en Objective-C
 - En el 2014 Mac sacó Swift, lenguaje simplificado
 - Es propietario
 - Simplifica el proceso de programación
 - Trabaja tanto para iOS como para Mac OS

Lenguajes Para Apps: Windows Mobile

- Visual Studio
 - Se supone es multiplataforma
 - Trabaja en Android, iOS
 - Requiere Xamarin
 - El desarrollo es en C Sharp (C#)

Lenguajes Para Apps: Android

- Lenguajes alternativos
 - App Inventor
 - Desarrollado por Google Labs para no-programadores
 - Basic 4 Android
 - Gran competencia de Java
 - LiveCode
 - Es un ambiente multiplataforma
 - Appcelerator Titanium
 - Con muchos usuarios (eBay, PayPal)
- Existen otros ambientes adicionales

Lenguajes Para Apps: Android

- Java
 - Todo para Android está desarrollado en Java
 - No es Java puro, pero es simple el cambio
 - Algunos otros lenguajes trabajan para Android
 - C++
 - Kotlin
 - Lenguaje adicional
 - Cambia la sintaxis de programación
 - Android Studio traduce lenguaje en C++ y Java a Kotlin

Android Studio: Requerimientos

Install Android Studio on ChromeOS

Android Studio has been officially supported on ChromeOS s follow the Android Studio [ChromeOS install instructions](#) .

System requirements for Android Studio

- 8 GB RAM or more recommended
- 20 GB of available disk space minimum
- 1280 x 800 minimum screen resolution
- Intel i5 or higher (U series or higher) recommended

Android Studio

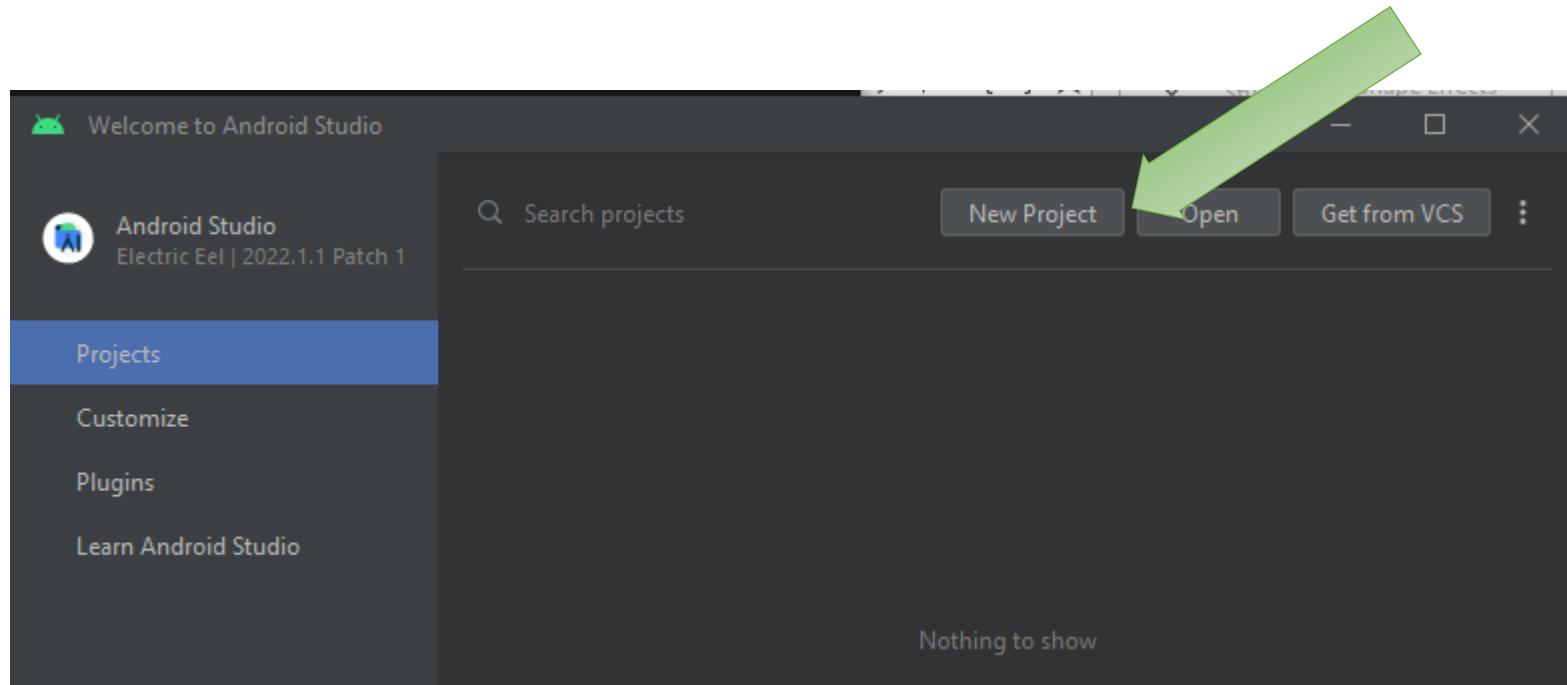
- Se debe tener instalado JDK (Actualizado)
 - Se instala primero
- Se debe descargar el Android Studio
 - En el link <https://developer.android.com/studio/install>
 - Está el detalle/manual de como ejecutarlo
 - El manual incluye un video tutorial

Android Studio

- Ubicación de proyectos

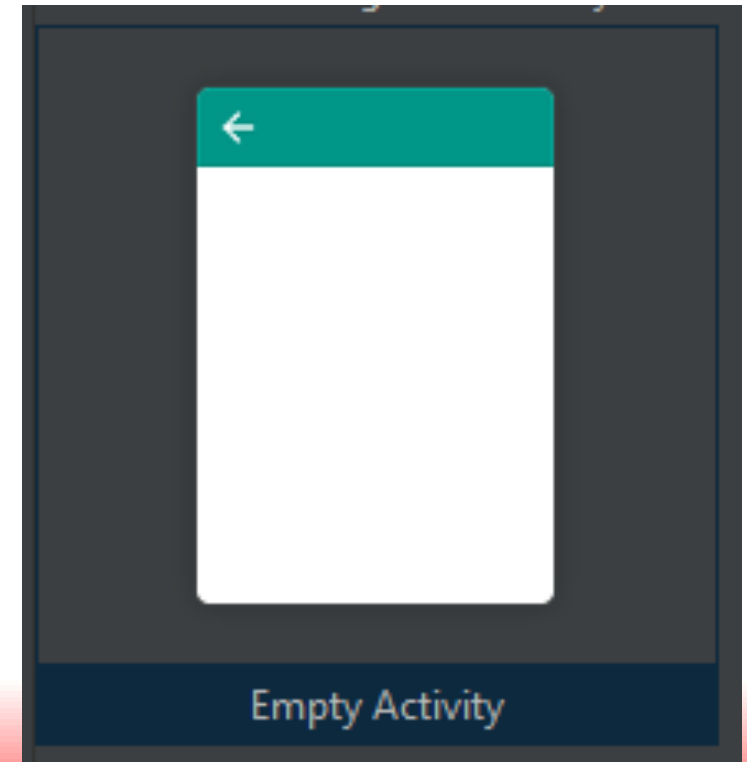
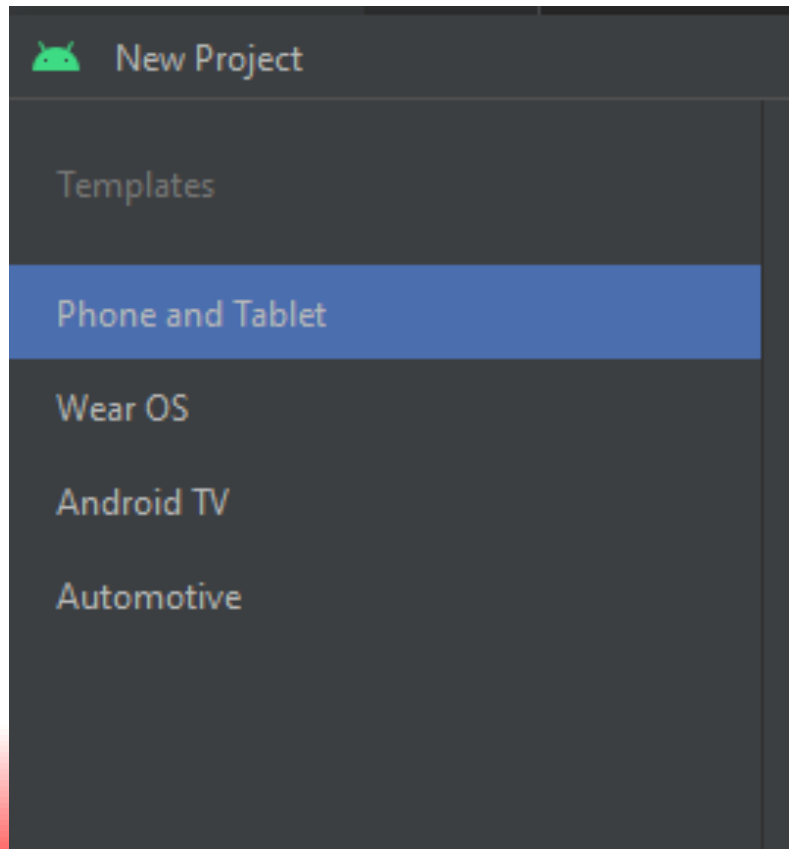
users/``nombreUsr``/AndroidStudioProjects

Android Studio: Proyecto1



Android Studio: Proyecto1

- Podemos crear proyectos para



Android Studio: Proyecto1

- Que es un Activity
 - Es la unidad más básica al desarrollar un app
 - Se puede decir que cada pantalla es un activity
 - Si hay 5 pantallas i.e. hay 5 actividades
 - Está separado en dos partes
 - Lógica
 - Archivos .java que contiene la clase donde está el código
 - Con esta se manipula la actividad
 - Gráfica
 - XML que tiene todas las etiquetas de los elementos en pantalla

Android Studio:

- Debemos configurar

Empty Activity

Creates a new empty activity

Name

Package name

Save location

Language

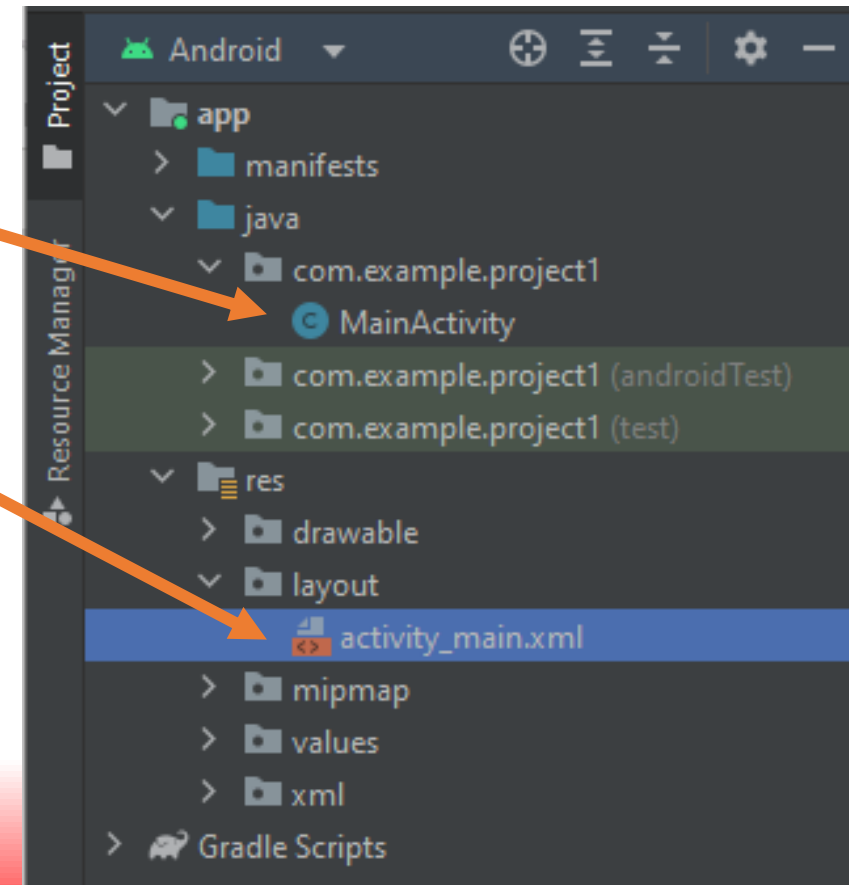
Minimum SDK

i Your app will run on approximately **94.4%** of devices.
[Help me choose](#)

☐ Use legacy android.support libraries [?](#)
Using legacy android.support libraries will prevent you from using the latest Play Services and Jetpack libraries

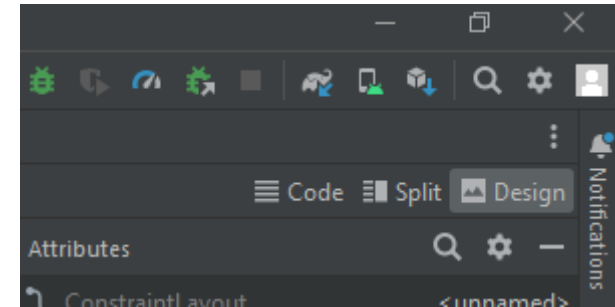
Android Studio: Proyecto1

- Donde está en el proyecto los archivos
 - Archivo .java
 - Archivo .xml



Android Studio: Proyecto1

- Note que hay dos archivos
 - activity_main.java
 - MainActivity.xml
 - Este tiene dos tabs **Design** y **Code**
 - En **Code** puede modificar el XML directamente
 - Modifique el Hello World!, en el tab **Code**
 - Note como se modifica en caliente y se visualiza
 - En el tab **Design**
 - Agregue un botón, note el error resultante
 - Puede usted identificar el error y cómo resolverlo?
 - Vea el blueprint





Android Studio: Proyecto1

- Explore donde en la herramienta
 - Puede cambiar el tamaño del texto
 - Puede modificar el texto sin acceder al XML
 - Se puede visualizar por aparte
 - Design
 - Blue Print
 - Design + Blue Print

Emulador para Android Studio


- Es un emulador de dispositivos móviles
- La herramienta levanta aparatos virtuales
- Con esto probamos el app
 - En distintos equipos
 - Desde el IDE
 - Verificamos su comportamiento
- Se llama Virtual Device Manager
 - Está en [Tools/Device Manager](#)
 - Note que puede seleccionar
 - Teléfono, Tablet, reloj o TV


 Virtual Device Configuration

 Select Hardware

Choose a device definition

Category	Name ▾	Play Store	Size	Resolution	Density
Phone	Resizable (Experimen...		6.0"	1080x2340	420dpi
Tablet	Pixel XL		5.5"	1440x2560	560dpi
Wear OS	Pixel 6 Pro		6.7"	1440x3120	560dpi
Desktop	Pixel 6		6.4"	1080x2400	420dpi
TV	Pixel 5		6.0"	1080x2340	440dpi
Automotive	Pixel 4a		5.8"	1080x2340	440dpi
	Pixel 4 XL		6.3"	1440x3040	560dpi

 Pixel 6



1080px

6.4"

2400px

Size: large
Ratio: long
Density: 420dpi

New Hardware Profile

Import Hardware Profiles

↺

Clone Device...

?

Previous


Next


Cancel

Finish

Android Studio Emulador

- En la creación del teléfono virtual
 - Cree un Pixel 6
 - Le puede cargar Oreo 8.0
 - En esta sección puede seleccionar versiones para x86-64
 - También está para AMD
 - Note que se recomienda usar la versión de API
- En la ventana de confirmación "Verify Config"
 - Habilite las opciones avanzadas
 - Note como podemos dar acceso al webcam del PC

 Virtual Device Configuration



System Image

Select a system image

Recommended
 x86 Images
 Other Images

Release Name	API Level ▼	ABI	Target
S	31	x86_64	Android 12.0
R	30	x86	Android 11.0 (Google AP
R	30	x86_64	Android 11.0 (Google AP
R	30	x86_64	Android 11.0
R	30	x86	Android 11.0 (AOSP ATD
Q	29	x86_64	Android 10.0 (Google AP
Q	29	x86_64	Android 10.0
Q	29	x86	Android 10.0
Pie	28	x86_64	Android 9.0 (Google AP

A system image must be selected to continue.

?


Previous

Next

Cancel

Finish

Oreo



API Level
26

Android
8.0

Google Inc.

System Image
x86_64

Questions on API level?
[See the API level distribution chart](#)

AVD Name Pixel 6 API 26

AVD Id Pixel_6_API_26



Pixel 6

6.4 1080x2400 420dpi

Change...



Oreo

Android 8.0 x86_64

Change...

Startup orientation



Portrait



Landscape

Camera

Front:

Emulated

Back:

VirtualScene

Network

Speed:

Full

Latency:

None

Emulated
Performance

Graphics:

Automatic

Boot option:



Cold boot



Quick boot

Hide Advanced Settings

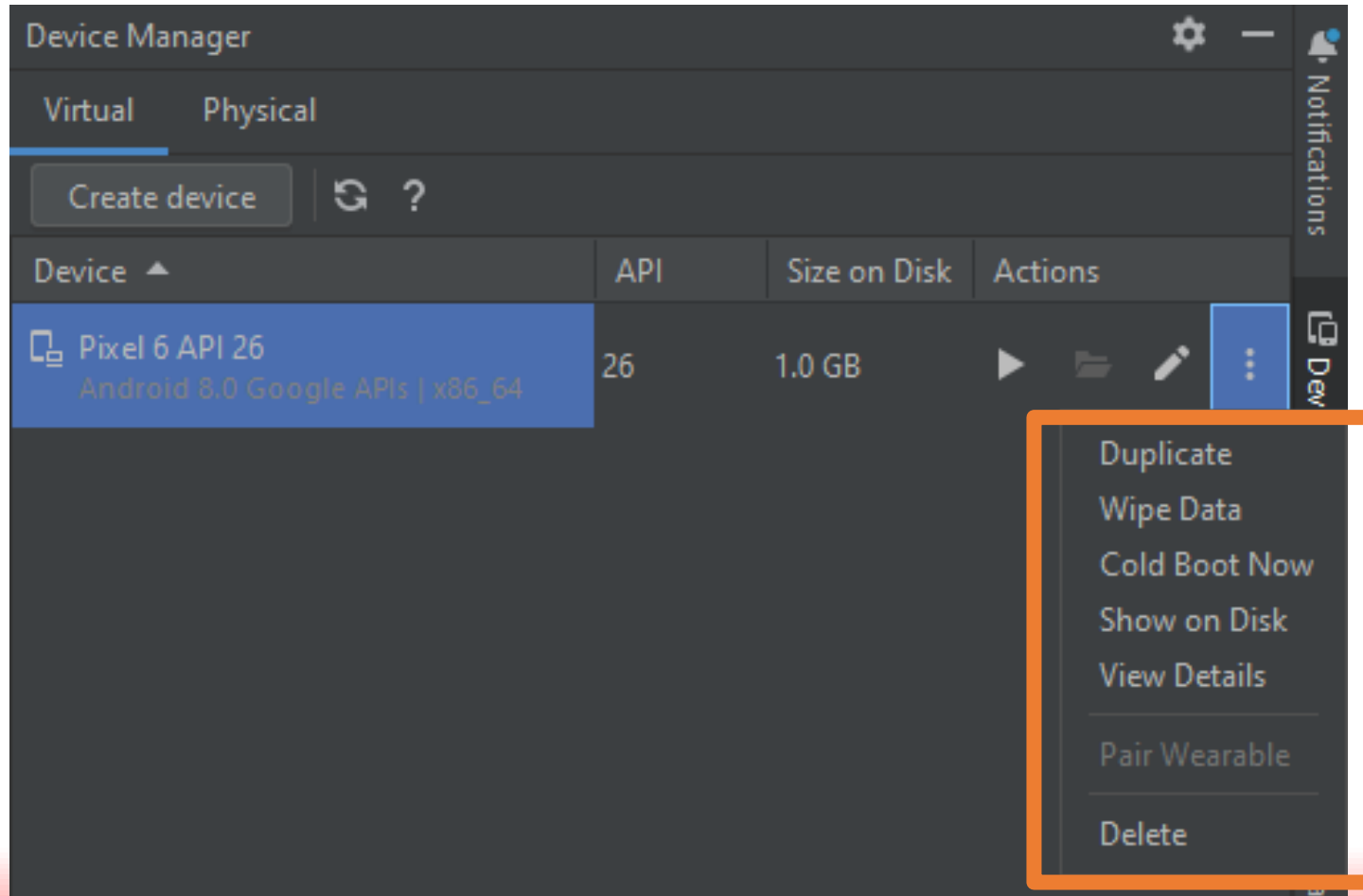
Front Camera

None - no camera installed for AVD

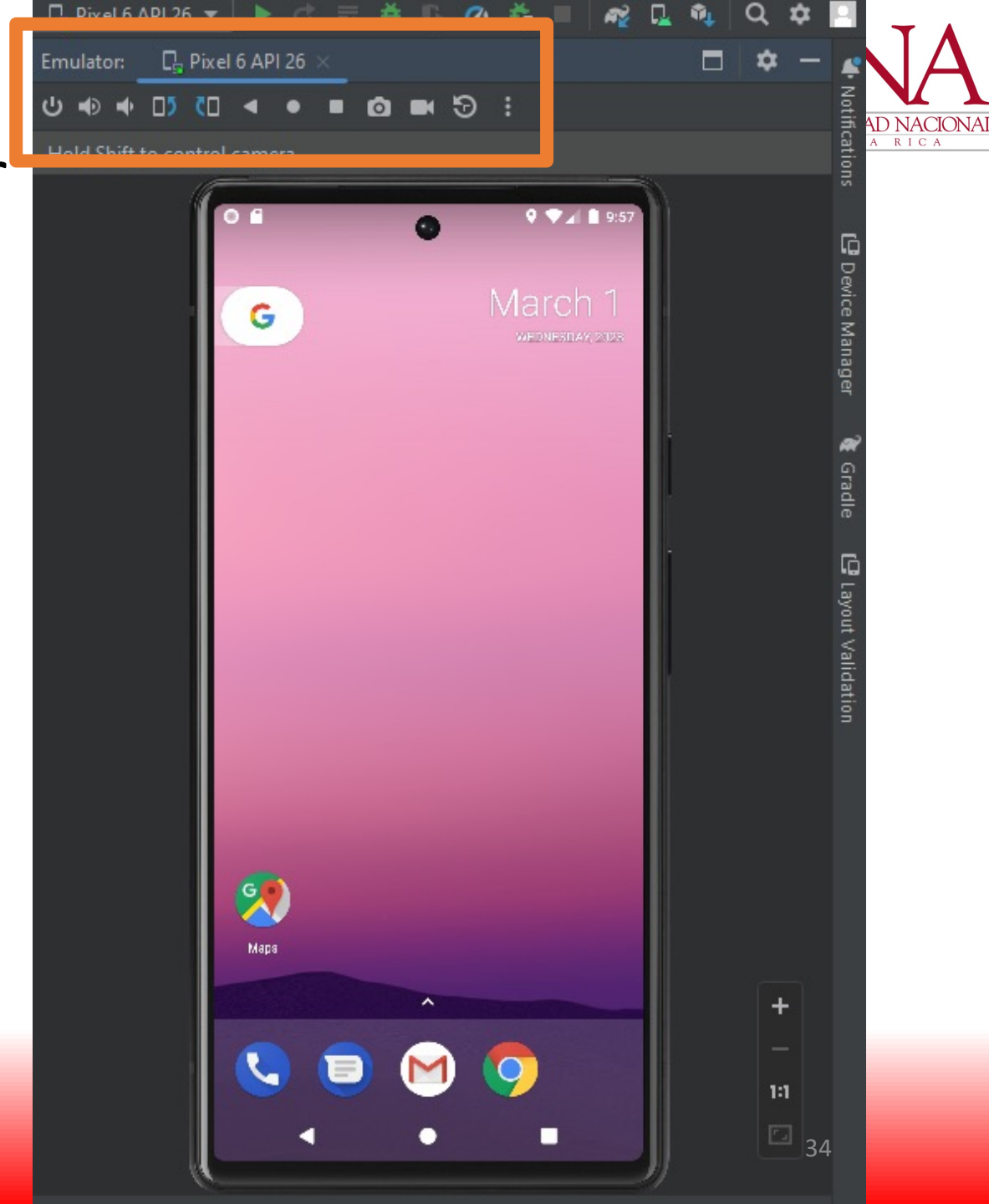
Emulated - use a simulated camera

Device - use host computer webcam or built-in camera

Android Studio Emulador



Android Studio Emulador



Pixel 6 API 26 - Extended Controls

×

Location

Cellular

Battery

Camera

Phone

Directional pad

Microphone

Fingerprint

Virtual sensors

Bug report

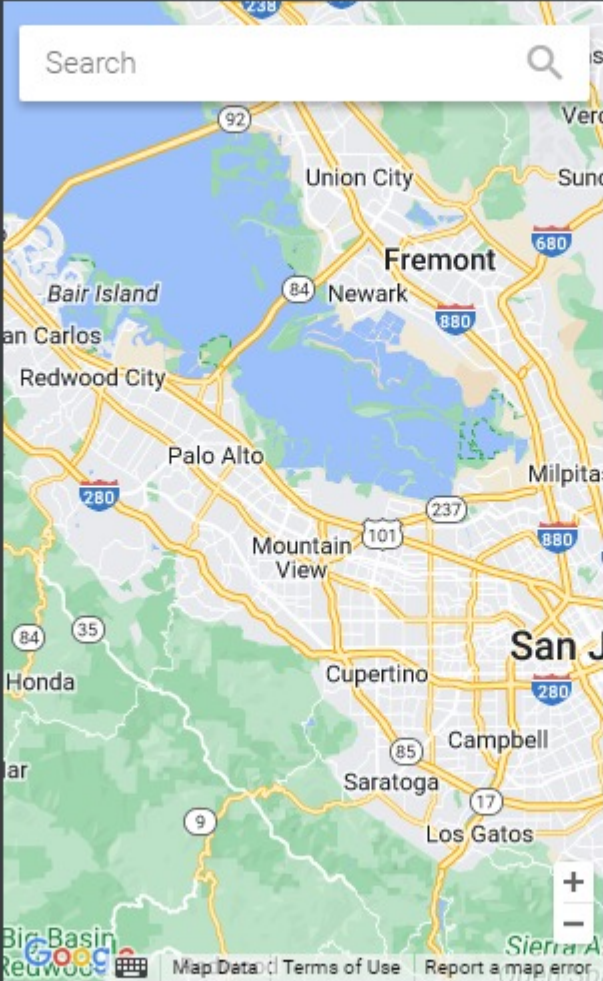
Record and Playback

Settings

Help

Single points Routes

Search



Enable GPS signal

☐

Saved points

Points that you save shall appear here

Import GPX/KML

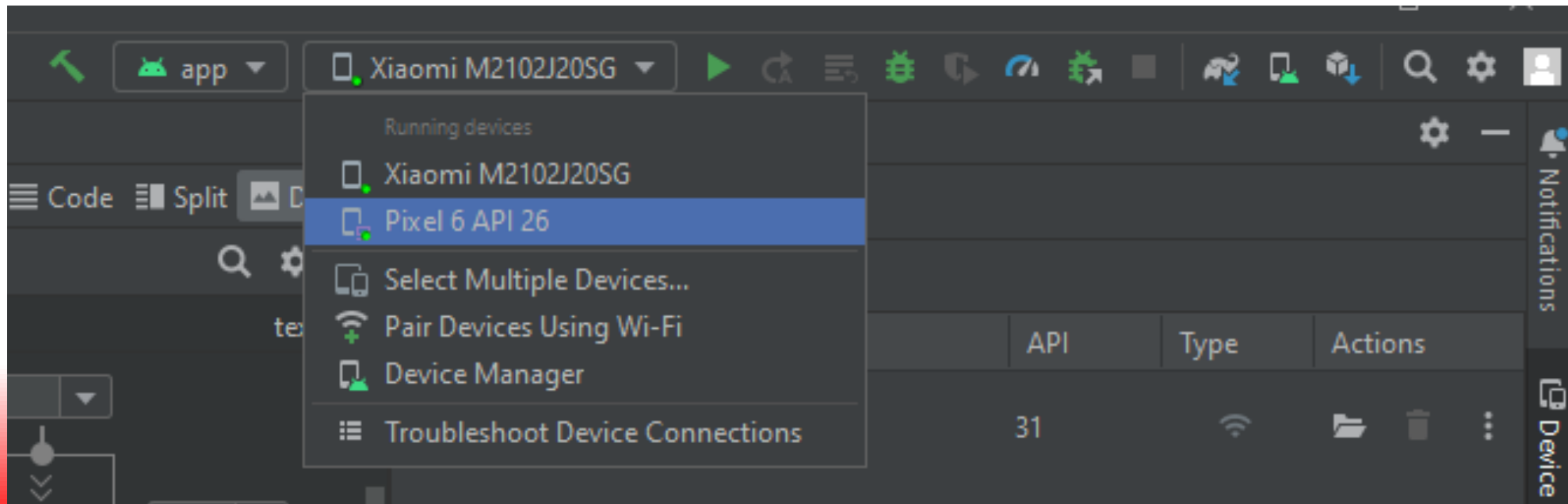
Set Location

Ver mis Apps usando mi Teléfono

- Aquí puede ver cómo habilitar el teléfono como dispositivo físico
 - [How to connect your Android device](#)
 - Se debe activar el **modo Developer**
 - Para conexión con USB se debe activar el **modo USB Debugging**
 - Para conexión WIFI se debe activar el modo **Wireless Debugging**
 - En ambos debe activar **Install via USB**
- Esto permite visualizar mis apps en mi propio teléfono
- Evitamos el consumo de recursos en mi PC
- Sacamos provecho de nuestro teléfono
- Se puede conectar tanto por USB como vía WIFI

Ejecutemos Hola Mundo

- En la parte superior del Studio
 - Note cómo podemos seleccionar el teléfono virtual o su propio teléfono
 - Ejecute la aplicación con el botón de “Play”



Ejecutemos Hola Mundo

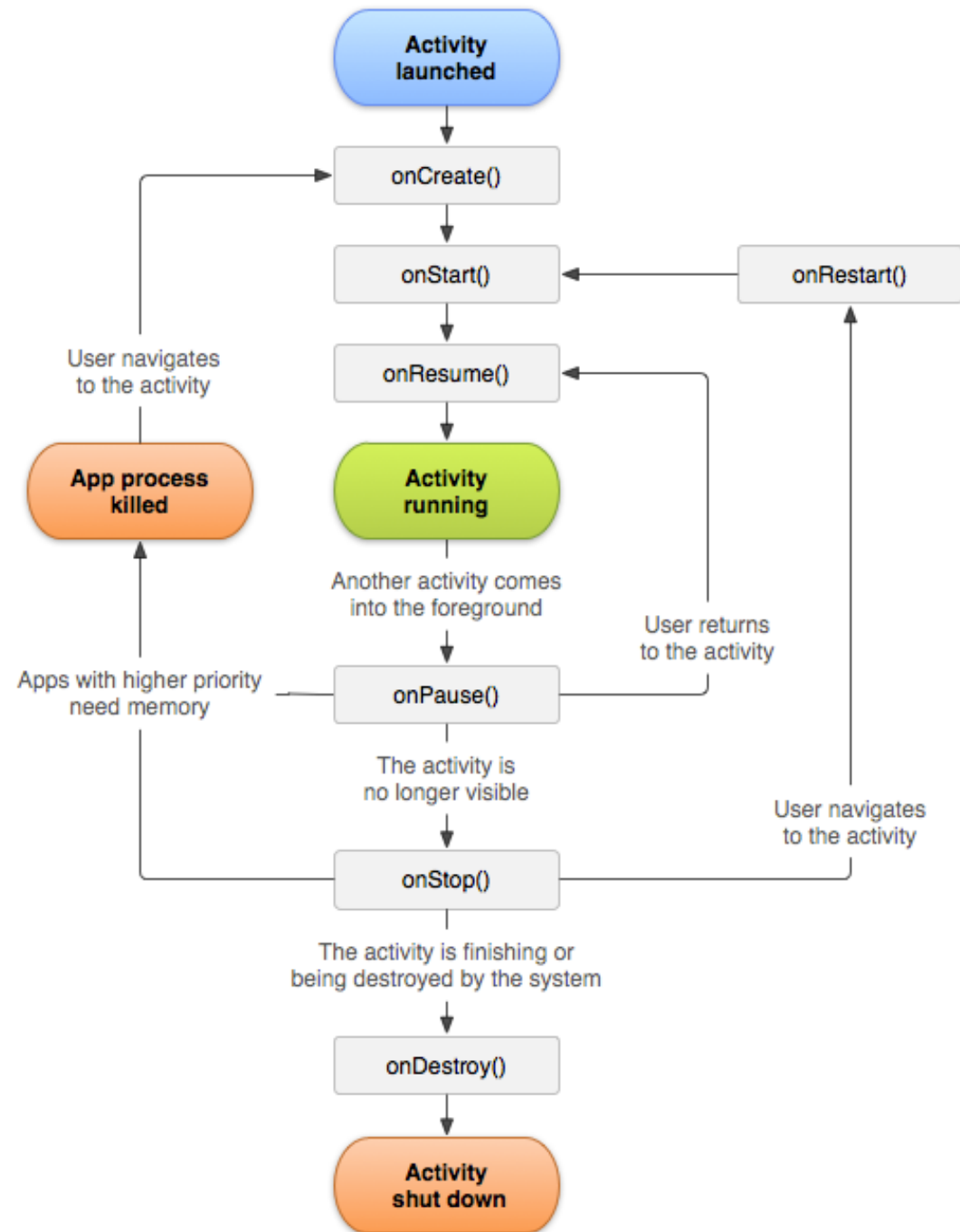
- En teléfono físico
 - Algunas configuraciones cambian según el dispositivo/idioma
 - Si está debidamente conectado al PC
 - Al seleccionar dispositivo
 - Se lista el teléfono físico y el o los virtuales
 - Puede suceder dos cosas
 - Que la aplicación se instale de inmediato
 - Que solicite en el teléfono autorización de instalación
 - Al finalizar la prueba **BORRE** el app del teléfono

Activity: Ciclo de Vida

- Cuando el usuario abre un APP se ejecuta
 - onCreate() que crea el app
 - onStart() que inicia el app
 - onResume() que muestra el app al usuario
 - A partir de aquí el Activity está ejecutándose
 - Se tiene completa interacción
 - Con el APP
 - Con el sistema del teléfono

Activity: Ciclo de Vida

- Si otra App entra en pantalla (foreground)
 - Se ejecuta onPause()
- Si el usuario minimiza el App
 - Se ejecuta onPause()
- El método onStop()
 - Se llama automáticamente por onPause()
- Si el usuario regresa al app
 - Desde onPause() llama a onResume()
 - Desde onStop() llama a onStart()
- Si el usuario cierra completamente el App
 - El sistema ejecuta onDestroy()
 - Esto cierra completamente la actividad



Modifique el App Hola Mundo

- Vamos a agregar un widget “Toast” al app
 - En el archivo .java
 - Debajo de onCreate()
 - Debemos agregar el siguiente código
 - Nota: Toast debe importar una librería
 - Al colocar Toast da error por falta de la librería
 - De click sobre la palabra y presione Alt+Enter
- `import android.widget.toast`

Modifique el App Hola Mundo

```
override fun onStart() {
    super.onStart();
    Toast.makeText(this, "OnStart", Toast.LENGTH_SHORT).show();
}
```

```
override fun onResume() {
    super.onResume();
    Toast.makeText(this, "OnResume", Toast.LENGTH_SHORT).show();
}
```

Modifique el App Hola Mundo

```
override fun onPause() {
    super.onPause();
    Toast.makeText(this, "OnPause", Toast.LENGTH_SHORT).show();
}
```

```
override fun onStop() {
    super.onStop();
    Toast.makeText(this, "OnStop", Toast.LENGTH_SHORT).show();
}
```

Modifique el App Hola Mundo

```
override fun onDestroy() {
    super.onDestroy();
    Toast.makeText(this, "OnDestroy", Toast.LENGTH_SHORT).show();
}
```

- Agregue la siguiente linea en onCreate() al final

```
Toast.makeText(this, "OnCreate", Toast.LENGTH_SHORT).show();
```

- Ejecute el APP

Depurando un APP

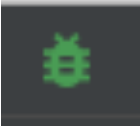
- Como todo IDE podemos poner break points
- Comente los métodos agregados
- Comente la línea del widget en onCreate()
- Agregue el siguiente código

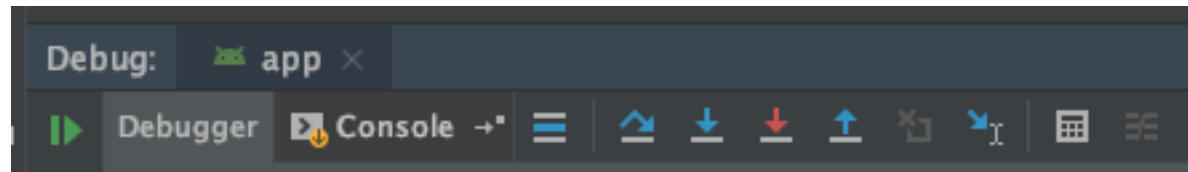
```
val materia1: Int = 7;
val materia2: Int = 7;
val materia3: Int = 7;
```

```
val promedio: Int = (materia1 + materia2 + materia3) / 3;
```

```
if(promedio >= 6){
    Toast.makeText(this,"Aprobado", Toast.LENGTH_LONG).show();
}else if(promedio <= 6){
    Toast.makeText(this,"Reprobado", Toast.LENGTH_LONG).show();
}
```

Depurando un APP

- En la línea de materia1 coloque un break point
- Puede Ejecutar con el bug 
- En la parte inferior están los botones de debug



Depurando un APP

- Ejecute paso a paso y analice los cambios
- El ambiente indica los valores de las variables
 - Tanto en el código fuente
 - Como en la ventana de variables

La clase Toast

- Es una notificación Emergente
- Nos permite mostrar mensajes al usuario
- Se ejecuta desde nuestra aplicación
- Se muestra en pantalla
- No bloquea las funciones del app
 - El app o activity en ejecución está visible y activa
- Toast no acepta ningún tipo de interacción
- Toast puede mostrar
 - Texto
 - Imágenes
 - Texto+Imagen

La clase Toast

- Es necesario importar
 - `import android.widget.toast;`
- La estructura de un toast es

`Toast.makeText(contexto, texto, duración).show();`

Ejercicio1: Construcción

- Cree una nueva aplicación
 - Es una calculadora coloque un nombre acorde
- En el XML
 - Elimine el Hello World! (en el preview)
- En la opción de Design (visualiza el diseño)
 - Coloque dos objetos de tipo Number en pantalla
 - Cambie su ID y su HINT
 - Ejecute la aplicación
 - Funciona bien?

Ejercicio1: Parte Gráfica

- Para corregir vaya al BluePrint
- Conecte el primer Number
 - Con el margen izquierdo
- Conecte el segundo Number
 - Con el Number primero
 - Dele espacio suficiente
- Como se puede centra?

Operando 1

Operando 2

Resultado

Sumar

Ejercicio1: Programación

- En el .kt dentro de la clase MainActivity
 - Agregue dos variables de tipo EditText
 - Note que Android Studio agrega la librería de forma automática
 - Coloque un nombre descriptivo
 - Son necesarias para capturar la info que da el usuario
 - Agregue una variable TextView
- Note que estamos creando variables
 - Que vamos a conectar con la estructura visual
 - La clase R nos ayuda a realizar dicha conexión

Ejercicio1: Programación

- Al inicio de MainActivity

```
lateinit var operando1: EditText  
lateinit var operando2: EditText  
lateinit var tvResultado: TextView  
lateinit var boton: Button
```

Ejercicio1: Programación

- Dentro de onCreate() al final del método conectamos con la vista

```
operando1 = findViewById(R.id.nOperando1)
operando2 = findViewById(R.id.nOperando2)
tvResultado = findViewById(R.id.tv_Resultado)
```

- Creamos un método nuevo para sumar (importante la conexión View)

```
fun sumar(vista : View){
    var entero1: Int? = operando1.text.toString().toInt()
    var entero2: Int? = operando2.text.toString().toInt()
    var resultado: Int? = entero1!! + entero2!!
    var resString: String? = resultado.toString()
    val tv1: TextView = findViewById(R.id.tv_Resultado)
    tvResultado.text = resultado.toString()
}
```

Ejercicio1: Conectamos el Botón

- Regrese al diseño
- Seleccione el botón Sumar
 - En su atributo onClick
 - Seleccione el método que agregamos
- Ejecute y pruebe su aplicación

Ejercicio 2

- Evaluar si el promedio de las 3 materias
 - Aprueba ≥ 7
 - Reprueba < 7

Estatus del Alumno

Español

Matemática

Química

EVALUAR