

**MODULO M08:**  
**Programación multimedia y dispositivos  
móviles**  
**Profesor: Jorge Juan Delgado**

# UNIDAD FORMATIVA 1

---

## DESARROLLO DE APLICACIONES PARA DISPOSITIVOS MÓVILES

- Android es un Sistema Operativo de última generación basado en Linux y creado por Google para poder gobernar dispositivos móviles.
- Su interfaz de usuario está basada en la tecnología **DMI** (Direct Manipulation Interface)
  - Interacción hombre-ordenador que representa objetos gráficos de interés de forma rápida, reversible y de acciones incrementales y que proporcionan un resultado que el usuario puede interpretar de forma sencilla y rápida

android 

- ¿Qué acciones podemos realizar con un dispositivo móvil?
  - Deslizar el dedo por la pantalla
  - Presionar ligeramente un punto
  - Hacer doble pulsación en un punto
  - Pulsar con diferentes grados de presión
  - Detectar proximidad respecto de la mano o de la cara
  - Ubicación GPS

Android puede controlar todas estas opciones



## > Qué es Android

ILERNA  
Online

- Diseñado para
  - Pantallas táctiles - Smartphones y tablets
  - Smart TV
  - Cámaras digitales
  - Otros dispositivos electrónicos



android 

# Hardware

Memoria limitada

CPU

Conexión de datos

Batería

Pantalla

Al desarrollar una aplicación hemos de tener en cuenta las limitaciones de nuestros dispositivos *target*.





¿Cuántos dispositivos Android diferentes existen?



Antes de desarrollar una APP

## ESTUDIO DE VIABILIDAD

- ¿A qué target de público va destinada?
- ¿Qué tipo de dispositivo tiene este target de media?
- ¿Qué características hardware necesitan?
- ¿Cuál queremos que sea la versión de Android mínima?





## > Limitaciones en dispositivos móviles

- Las plataformas móviles evolucionan rápidamente y no tienen ningún interés en ser compatibles con los frameworks híbridos ya existentes.
- A) Si eres un programador nativo, podrás adaptarte rápidamente a las nuevas interfaces y funcionalidades, tendrás documentación y ejemplos
- B) Si utilizas un framework, tendrás que esperar a que los creadores del framework saquen una actualización compatible con la nueva versión del sistema operativo





- Los dispositivos móviles Android generalmente tienen una pantalla de tamaño pequeño o mediano y con recursos limitado, aunque cada vez menos
- Interfaz de usuario adaptada a tamaño de pantallas pequeñas
- El dispositivo debería tener conexión de datos
- Seguridad contra virus y malware
- Limitaciones de memoria RAM y almacenamiento interno
- Consumo de batería
- **VERSIÓN MÍNIMA Y RECOMENDADA**

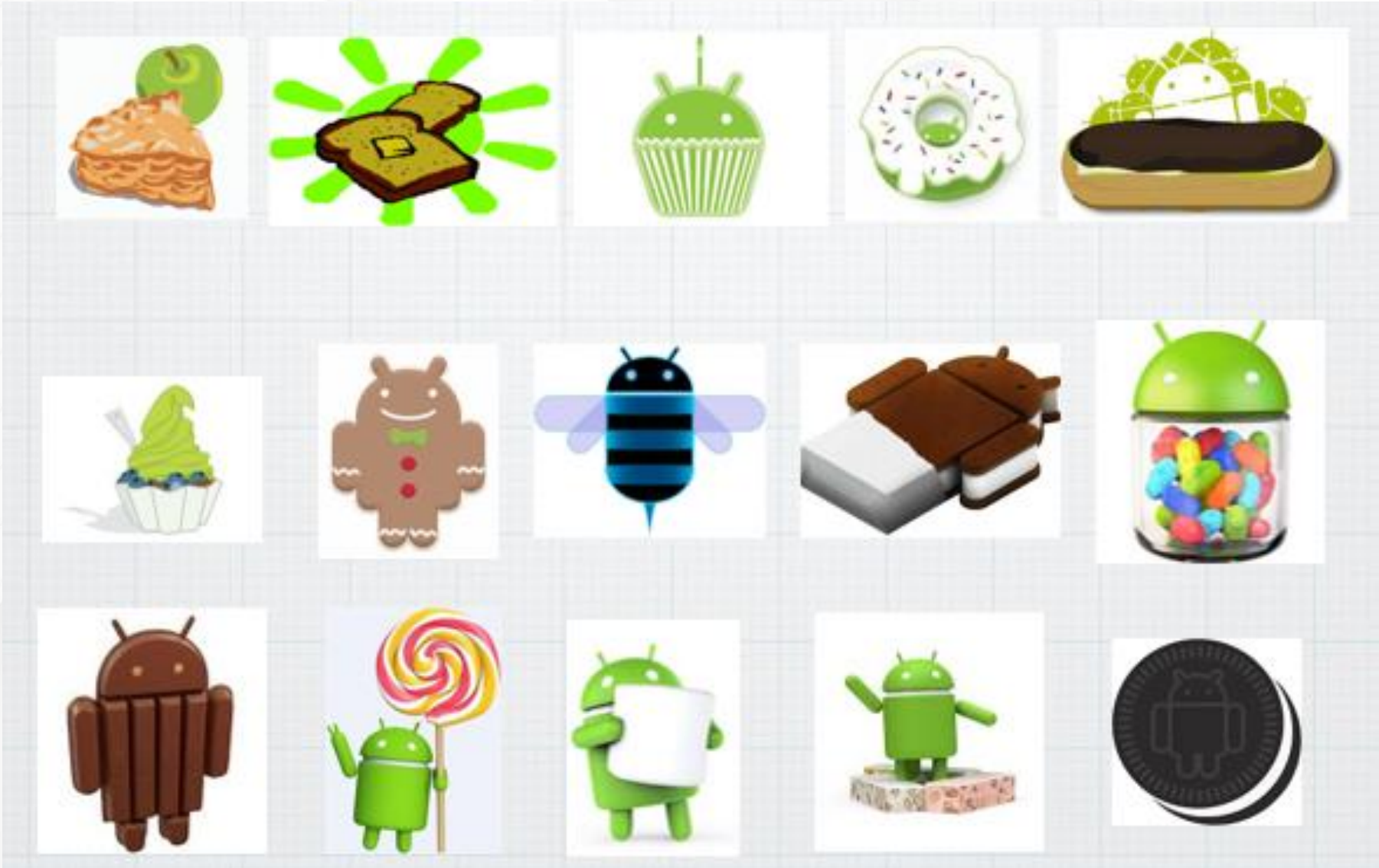
NOMBRE	NIVEL DE API	NOMBRE	NOVEDADES
1.0	1	Apple Pie	Primera versión comercial
1.1	2	Banana Bread	Resuelve problemas de la versión 1.0
1.5	3	Cupcake	Núcleo de Linux 2.6.27
1.6	4	Donut	Incluye la Galería y motor Text-to-speech
2.0 / 2.1	5 / 7	Eclair	GUI renovada, calendario, Google Maps renovado
2.2.x	8	Froyo	Soporte para pantallas HD y optimización de memoria y rendimiento

NOMBRE	NIVEL DE API	NOMBRE	NOVEDADES
2.3.x	9 / 10	Gingerbread	Mejoras interfaz, audios y gráficos, Google Talk, mejoras cámara y batería
3.x	11 / 13	Honeycomb	ActionBar, teclado rediseñado, almacenamiento SD, multitarea
4.0.x	14 / 15	Ice Cream Sandwich	Optimizaciones y corrección de errores
4.1	16	Jelly Bean	Interfaz de usuario mejorada, barra notificaciones y gestos
4.2	17	Gummy Bear	Multiusuario, acceso rápido a barra de notificaciones
4.3	18	Jelly Bean	Soporte Bluetooth de baja energía, Open GL 3.0, mejoras de seguridad

NOMBRE	NIVEL DE API	NOMBRE	NOVEDADES
4.4	19	Kit Kat	Corrección de errores, app de Contactos, optimización RAM y procesador
5.0	21	Lollipop	Nueva interfaz usuario, nuevos controles de notificaciones, ahorro batería, soporte 64 bits
6.0	23	Marshmallow	Nuevo modelo de permisos, USB-TypeC, enlaces verificados
7.0	24 / 25	Nougat	Multiventana, mejora de notificaciones, encender la pantalla al levantar el teléfono
8.0	26 / 27	Oreo	Optimización de batería en segundo plano, mejora de notificaciones, nuevo diseño



> Versiones de Android





- Android domina el mercado de las APPs para dispositivos móviles, sin menospreciar a otros como iOS
- Mercado de desarrollo de APPs rentable
- Mercado de desarrollo de videojuegos rentable
- En la actualidad, más de 800.000 aplicaciones subidas en Google Play, sin contar con las que existen en el Black Market
- Lenguajes de programación Android: Java y Kotlin
- Usa XML para el diseño de interfaces

- Los elementos principales son:
  - Software Development Kit (SDK), que incluye librerías y herramientas propias
  - Entornos de desarrollo, en nuestro caso usaremos Android Studio

SDK

IDE

Android Studio es el entorno de desarrollo integrado oficial para la plataforma Android. Fue anunciado el 16 de mayo de 2013 en la conferencia Google I/O, y reemplazó a Eclipse como el IDE oficial para el desarrollo de aplicaciones para Android.

Es el IDE más recomendado y está basado en el IDE de Java *Idea IntelliJ*, *permite una instalación muy rápida de diferentes plugins. Por último pero no menos importante permite la compilación y exportación de APK.*



Requisitos  
mínimos

SO: Windows 7 (o superior) / Linux / Mac OSX

RAM: 2GB

HDD: 2GB de espacio libre

Java: JDK v 1.8

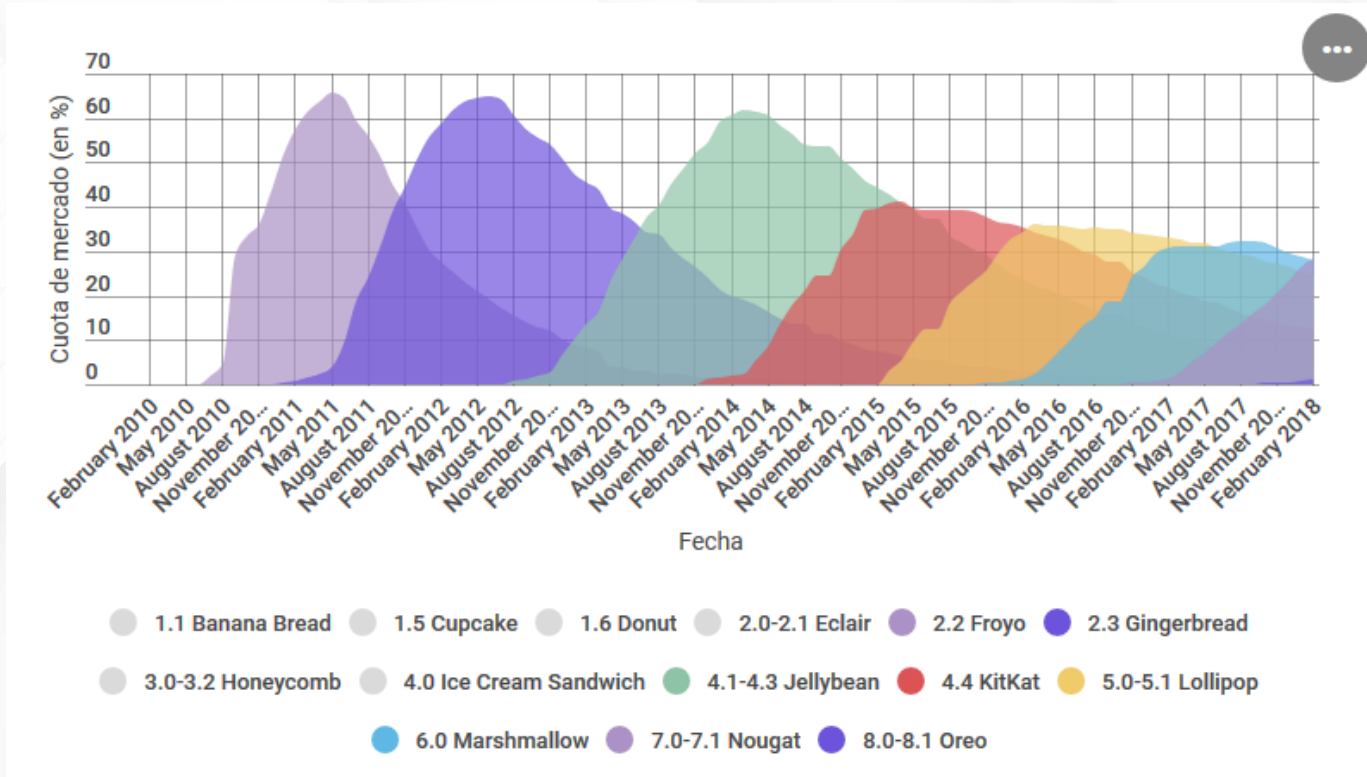
Requisitos  
recomendados

SO: Windows 7 (o superior) / Linux / Mac OSX





RAM: 8GB

HDD: 4GB de espacio libre

Java: JDK v 1.8



<https://infogram.com/despliegue-versiones-android-historico-feb-08-1h9j6qx8x0l52gz>

Type	Name	Play Store	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	Nexus 5X API 29 x86		1080 × 1920: 420dpi	29	Android 10.0 (Google APIs)	x86	3,9 GB	  



The image displays an Android emulator interface. On the left is a virtual smartphone screen showing the date 'Monday, Sep 23' and a dock with icons for Phone, Messages, and Chrome. On the right is the 'Extended controls' window for a Nexus 5X device. The 'Location' section is active, showing 'GPS data point' with a 'Coordinate system' set to 'Decimal'. The 'Currently reported location' is displayed as follows:

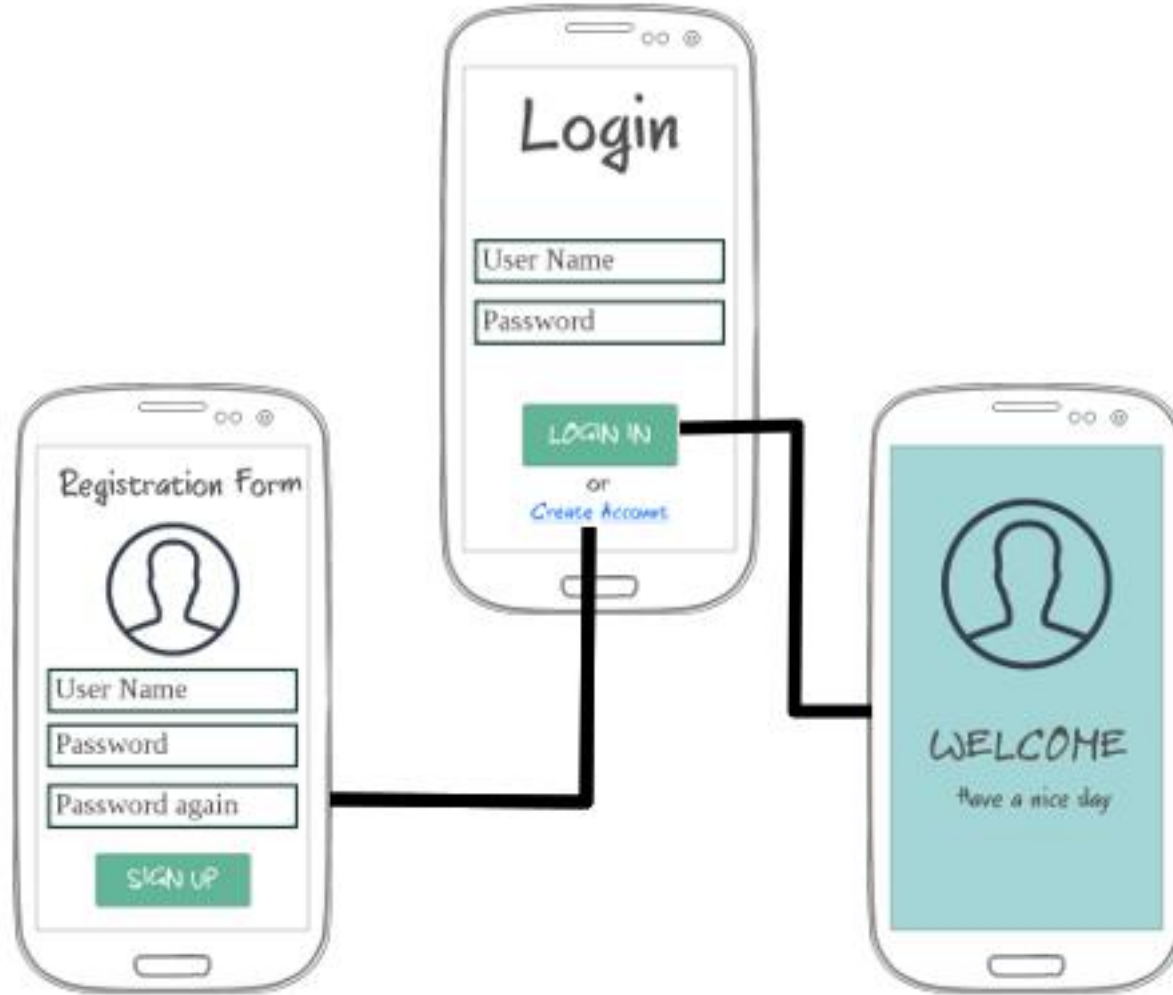
Latitude: 37.4220	Longitude: -122.0840	Altitude: 5.0	Speed: 0.0
-------------------	----------------------	---------------	------------

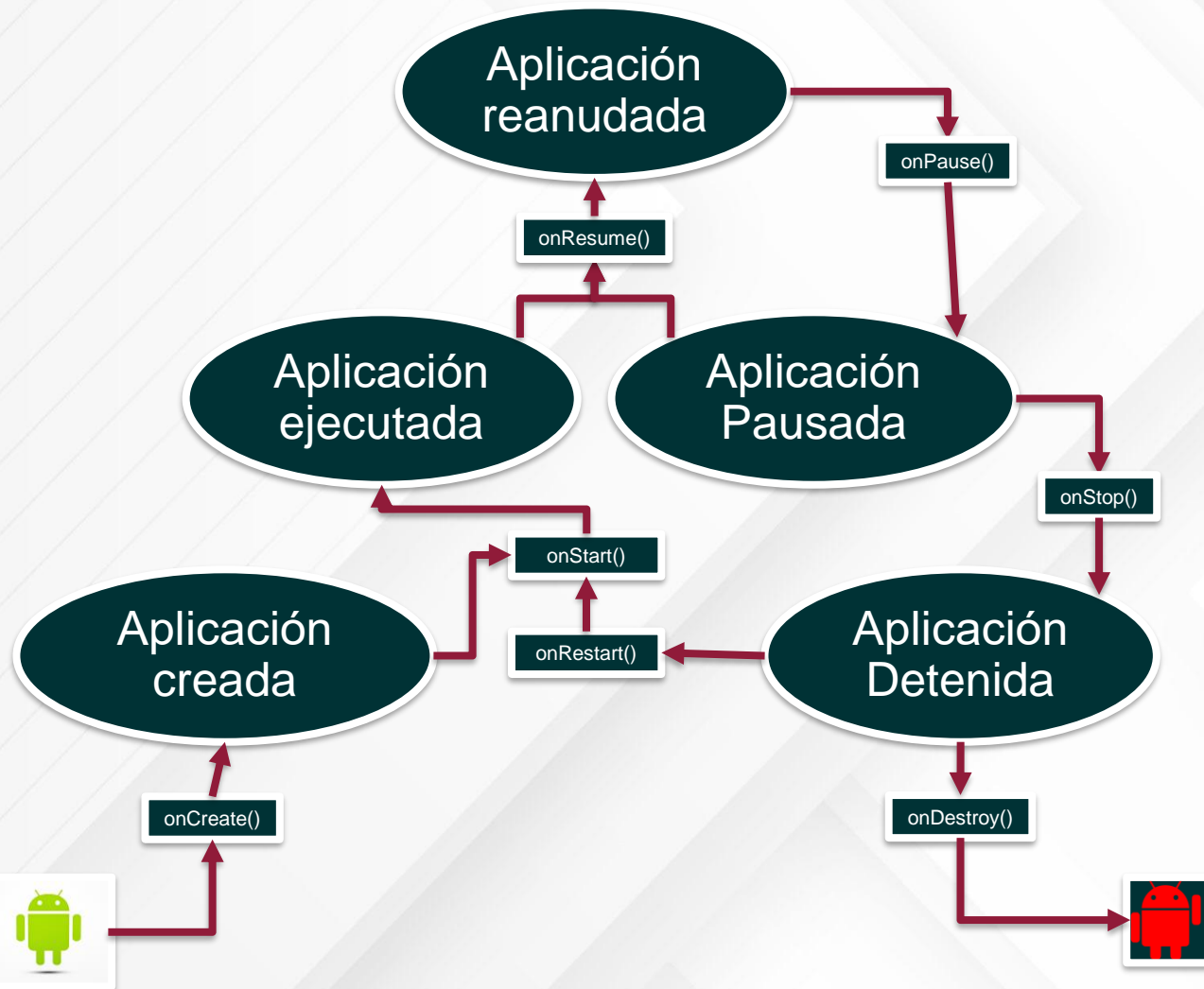
Below this, the 'GPS data playback' section features a table with the following headers: Delay (sec), Latitude, Longitude, Elevation, Name, and Description. The table is currently empty. At the bottom of the controls window, there is a play button, a 'Speed 1X' dropdown menu, and a 'LOAD GPX/KML' button.

Las aplicaciones Android están compuestas por dos partes



**Una aplicación se compone de una o más actividades. Una actividad es el componente de la aplicación que permite la interacción con el usuario, por lo tanto, una actividad es cada una de las pantallas que componen la aplicación.**





**onCreate():** Método invocado al crear una actividad.

**onStart():** Método invocado justo antes que la actividad se haga visible.

**onResume():** Método invocado cuando el usuario va a interactuar con la Actividad.

**onPause():** Método invocado cuando la actividad se está poniendo en background.

**onRestart():** Método invocado al recuperar la actividad.

**onStop():** Método invocado cuando la actividad ya no está visible, debido a que otra actividad la está eclipsando.

**onDestroy():** Método invocado cuando la actividad antes de que se destruya.

---

## Manifest

Describe información esencial de tu aplicación para las herramientas de creación de Android, el sistema operativo Android y Google Play

---

## Java

Código del aplicativo (Java + Kotlin)

---

## Res

Recursos de la aplicación

Drawable

---

Layout

---

Menu

---

Mipmap

---

Values

Colors.xml

Dimensions.xml

Strings.xml

Styles.xml

---

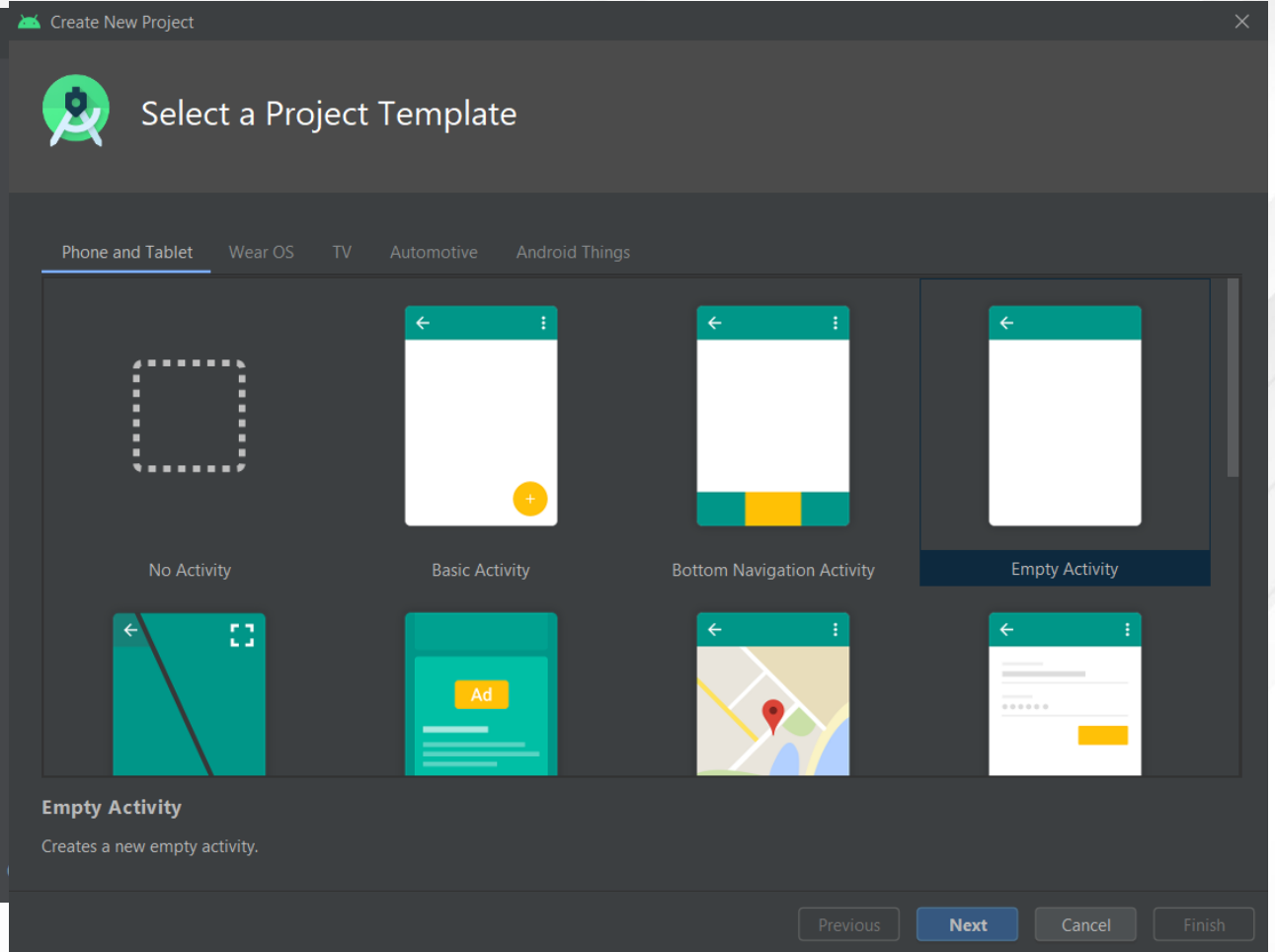
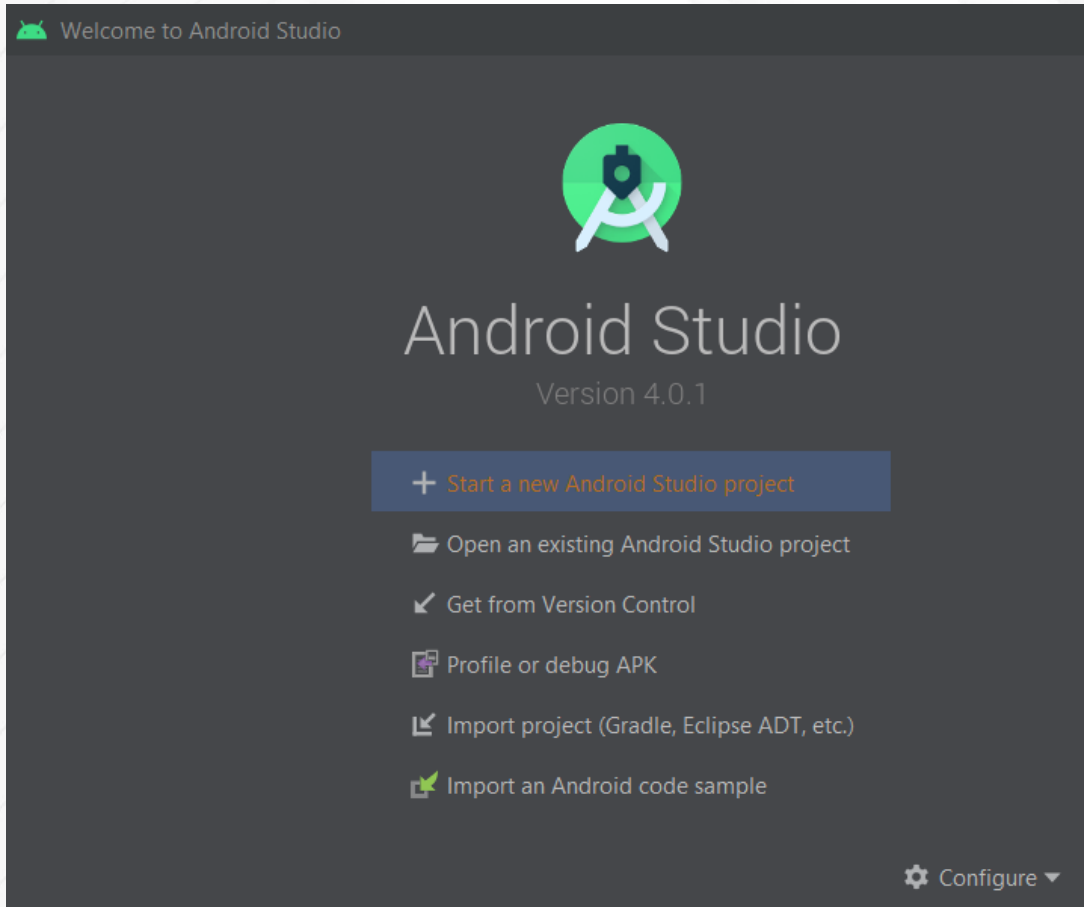


Nuestro primer proyecto de Android

---


UF1

## > Primer Proyecto



Create New Project

# Configure Your Project



Empty Activity

Creates a new empty activity.

Name  
PrimerProyecto

Package name  
com.example.primerproyecto

Save location  
C:\Users\jjdelgado\AndroidStudioProjects\PrimerProyecto

Language  
Kotlin

Minimum SDK  
API 16: Android 4.1 (Jelly Bean)

**i** Your app will run on approximately **99,8%** of devices.  
[Help me choose](#)

Use legacy android.support libraries [?](#)

Previous Next Cancel Finish

## > Activity

```
package com.example.primerproyecto

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}
```

Archivos .kt - Programación

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Archivos de layout .xml



Virtual Device Configuration

## Select Hardware

Choose a device definition

Category	Name	Play Store	Size	Resolution	Density
TV	Pixel 2		5,0"	1080x1920	420dpi
	Pixel		5,0"	1080x1920	420dpi
Phone	Nexus S		4,0"	480x800	hdpi
	Nexus One		3,7"	480x800	hdpi
Wear OS	Nexus 6P		5,7"	1440x2560	560dpi
	Nexus 6		5,96"	1440x2560	560dpi
Tablet	Nexus 5X		5,2"	1080x1920	420dpi
	Nexus 5		4,95"	1080x1920	xxhdpi

1080px

5,2"

1920px

Nexus 5X

Size: large  
Ratio: long  
Density: 420dpi

New Hardware Profile Import Hardware Profiles Clone Device...

Previous Next Cancel Finish

# Configurando un Emulador

Virtual Device Configuration


## System Image

Select a system image

Recommended x86 Images Other Images

Release Name	API Level	ABI	Target
<b>R</b>	30	x86	Android 10.0+ (Google Play)
<a href="#">Q Download</a>	29	x86	Android 10.0 (Google Play)
<a href="#">Pie Download</a>	28	x86	Android 9.0 (Google Play)
<a href="#">Oreo Download</a>	27	x86	Android 8.1 (Google Play)
<a href="#">Oreo Download</a>	26	x86	Android 8.0 (Google Play)
<a href="#">Nougat Download</a>	25	x86	Android 7.1.1 (Google Play)
<a href="#">Nougat Download</a>	24	x86	Android 7.0 (Google Play)

**R**



API Level  
**30**

Android  
**10.0+**

Google Inc.

System Image  
**x86**

We recommend these Google Play images because this device is compatible with Google Play.

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



Previous **Next** Cancel Finish

Virtual Device Configuration



## Android Virtual Device (AVD)

### Verify Configuration

AVD Name:

 Nexus 5X	5.2 1080x1920 420dpi	<input type="button" value="Change..."/>
 R	Android 10.0+ x86	<input type="button" value="Change..."/>

Startup orientation

 Portrait   Landscape

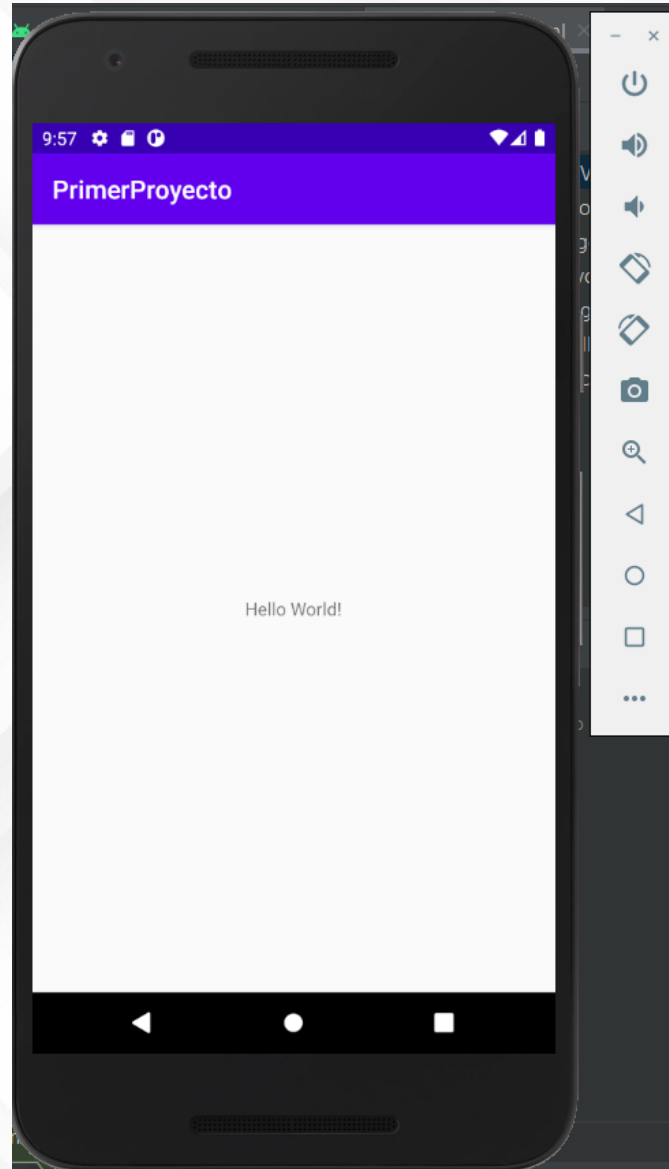
Emulated Performance Graphics:

### AVD Name

The name of this AVD.

? Previous Next Cancel **Finish**

## Configurando un Emulador







- Las unidades de medida más comunes son:
  - Match\_parent → Igual que el padre.
  - Wrap\_content → Ajustar al contenido
- Tamaños fijos:
  - dp → Pixels independientes de la densidad, coge como base una pantalla de 160dpi (puntos por pulgada).
  - sp → Similar a dp pero coge como base tamaño de fuente seleccionada.
  - pt → 1/72 de una pulgada
  - px → Se corresponde con la resolución actual de la pantalla.
  - mm,in → Basado en el tamaño físico de la pantalla.

Lienzo: actividad en la pantalla de nuestra aplicación

- Componentes de una Aplicación

- Vista (View) Clase padre de todos los elementos. Todo lo que podemos pintar en nuestro lienzo seán objetos de tipo vista.
- Layout Molde en el que vamos a insertar nuestros elementos.
- Actividad (Activity)
- Servicio (Service)
- Intención (Intent) Peticiones de comunicaciones para avanzar en la actividad.
- Receptor de Anuncios (Broadcast receiver) Para recibir notificaciones.
- Proveedor de contenido (Content provider) Para almacenar información dentro de la aplicación.

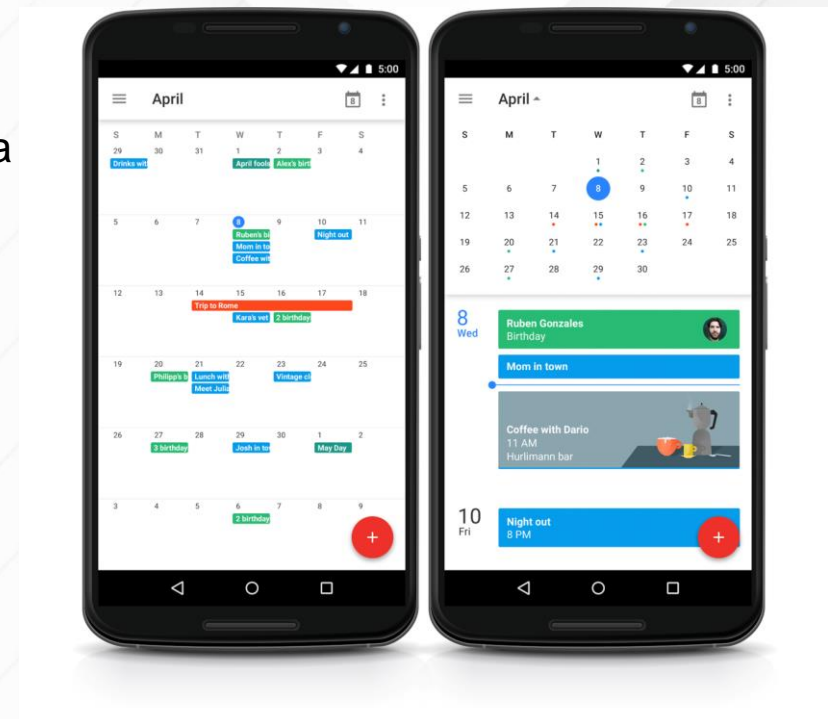
- Componentes de una Aplicación

- **VISTA (VIEW)**

- La interfaz de usuario se construye a partir de objetos Vistas (Views) y Grupo de Vistas (ViewGroup)

- Se usan para dibujar contenido en la pantalla del dispositivo Android.

- Aunque puedes instanciar una View en el código Kotlin o Java, la forma más sencilla de usarlo, es a través de un archivo de diseño XML



- Componentes de una Aplicación

- VISTA (VIEW)
- El archivo layout sería **activity\_main.xml** y el código que contiene podría tener esta forma
- La vista que estamos usando es de tipo texto(**TextView**)
- Los atributos `layout_width` y `layout_height` indican el espacio que queremos usar de la pantalla
- El atributo `text` contiene el texto a representar
- El atributo `id` sirve para referenciar al objeto `TextView`

```
1 <TextView
2     android:id="@+id/hello_world"
3     android:layout_width="wrap_content"
4     android:layout_height="wrap_content"
5     android:text="Hello World!" />
```

- Componentes de una Aplicación

- VISTA (VIEW)
- Las cadenas de texto que vamos a usar en la aplicación se pueden definir en el fichero res/values/strings.xml

```
values
strings.xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="hello">Hello World, HolaMundo!</string>
  <string name="app_name">Hola, Mundo</string>
</resources>
```

- De esta forma, estamos haciendo una separación entre código y contenido
- Facilitamos la traducción a otras lenguas de nuestra aplicación [y la corrección](#)



- Componentes de una Aplicación

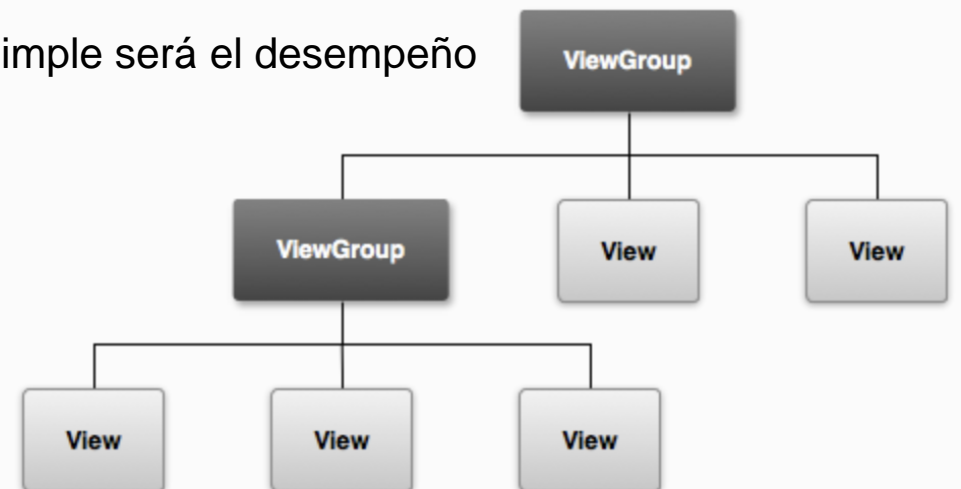
- VISTA (VIEW)

- Grupo de Vistas (View Group)

- Es un objeto invisible que se usa para contener otros objetos View y ViewGroup y poder así organizar y controlar el layout de una pantalla

- Los objetos ViewGroup se usan para establecer una jerarquía de objetos View de forma que podamos crear layouts más complejos

- Eso sí, cuánto más simple pueda mantenerse un layout, más simple será el desempeño



- Componentes de una Aplicación

- VISTA (VIEW)

- Grupo de Vistas (View Group)

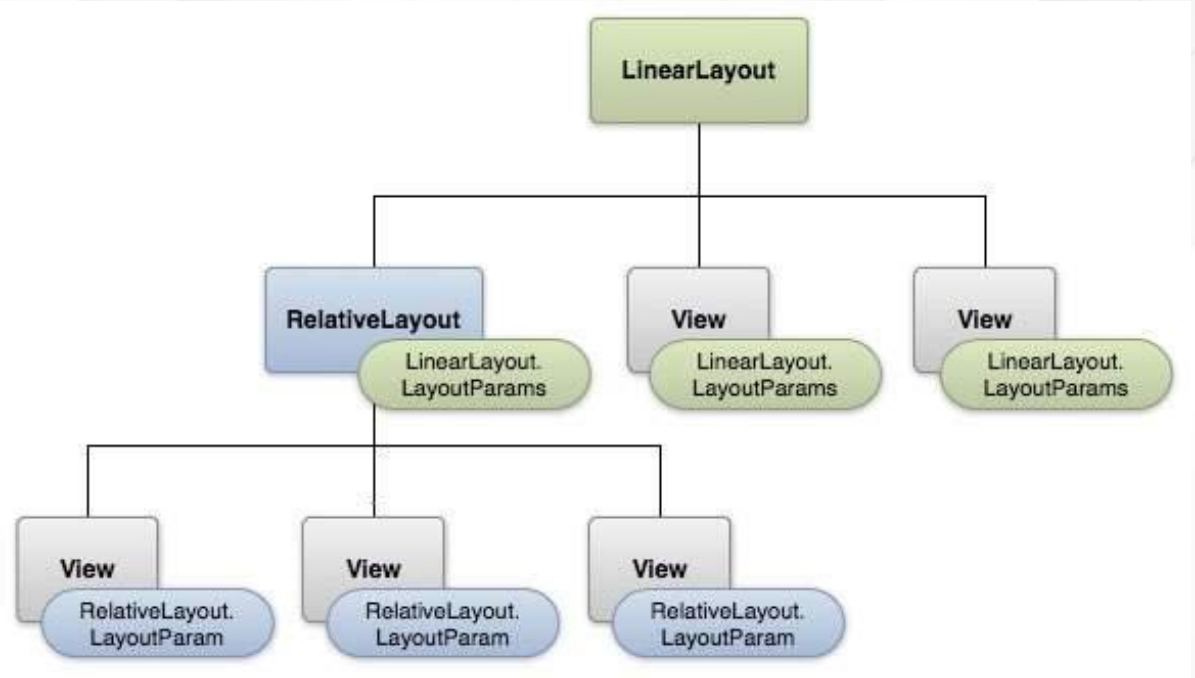
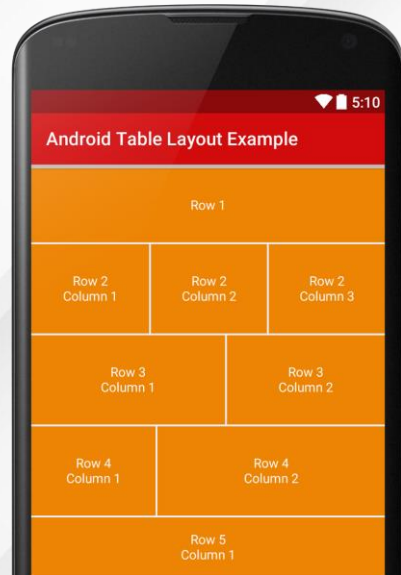
- Los objetos ViewGroup pueden ser instanciados de la misma forma que los objetos View, tanto en XML como en Java

```
01 <?xml version="1.0" encoding="utf-8"?>
02 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/andrc
03     android:layout_width="match_parent"
04     android:layout_height="match_parent"
05     android:orientation="vertical">
06     <TextView
07         android:id="@+id/hello_world"
08         android:layout_width="wrap_content"
09         android:layout_height="wrap_content"
10         android:text="Hello World!" />
11     <TextView
12         android:id="@+id/hello_world_2"
13         android:layout_width="wrap_content"
14         android:layout_height="wrap_content"
15         android:text="Hello World 2!" />
16 </LinearLayout>
```

- Componentes de una Aplicación

- LAYOUTS**

- Se define así a un contenedor de vistas
  - También es una subclase de View
  - Para combinar varios elementos de tipo vista, tendremos que usar un objeto de tipo Layout



- Componentes de una Aplicación

- LAYOUTS

- Un Layout es un elemento que representa el diseño de la interfaz de usuario de componentes gráficos como una actividad, fragmento o widget
- Gracias a al Layout podemos controlar el comportamiento y la posición de las vistas

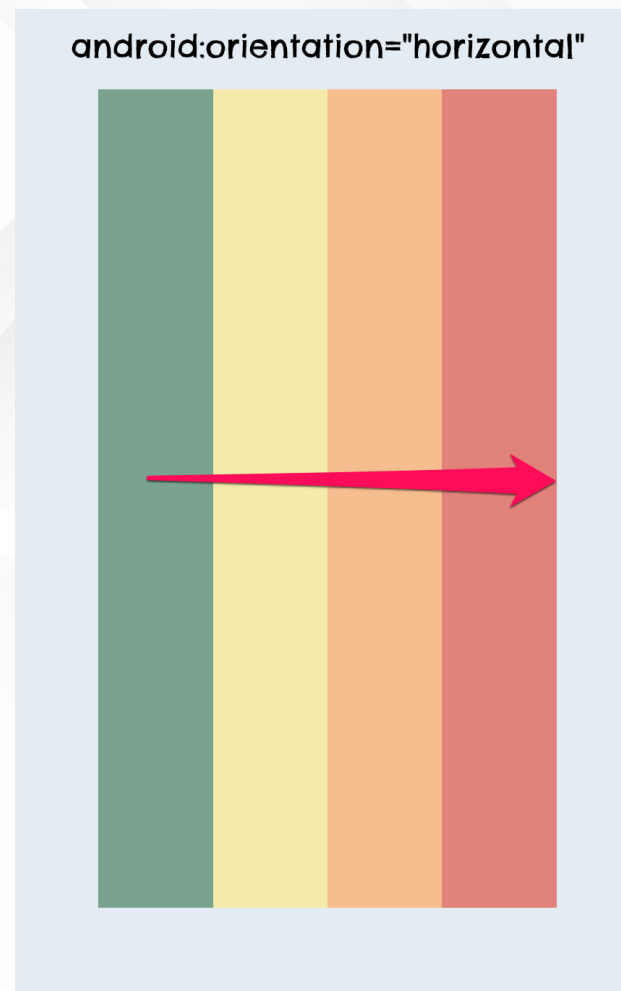
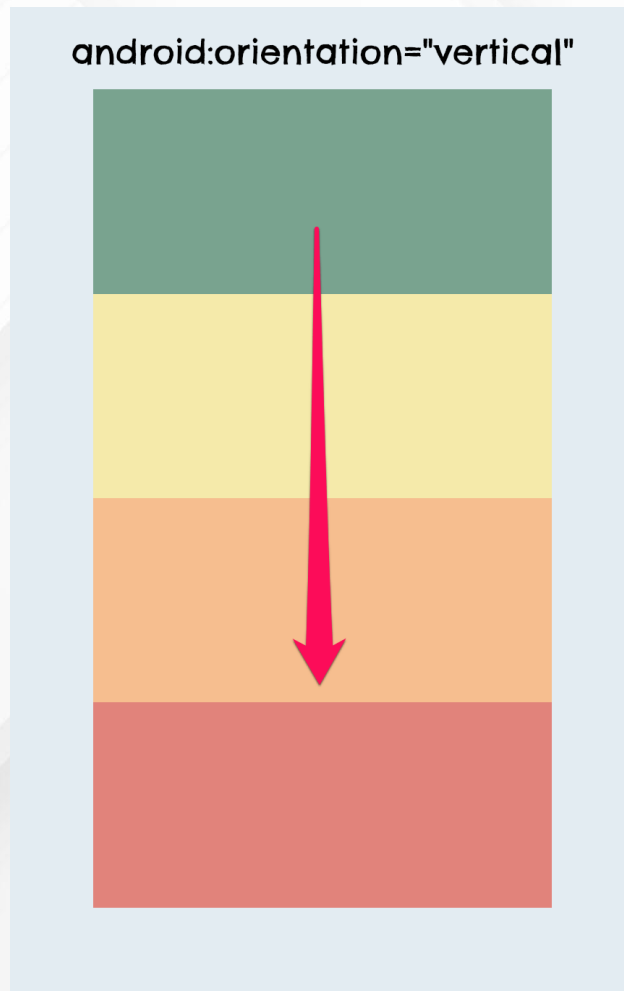
como por ejemplo que pasa si giramos el dispositivo. Por ejemplo, que Instagram siempre esté en vertical aunque giremos el smartphone.

- Componentes de una Aplicación
  - LAYOUTS
  - Tipos de Layouts o ViewGroup más usados
  - **LinearLayout**: Dispone los elementos en una fila o en una columna.
  - **TableLayout**: Distribuye los elementos de forma tabular.
  - **RelativeLayout**: Dispone los elementos en relación a otro o al padre.
  - **AbsoluteLayout**: Posiciona los elementos de forma absoluta.
  - **FrameLayout**: Permite el cambio dinámico de los elementos que contiene.
  - **ConstraintLayout**: Versión mejorada de RelativeLayout, que permite una edición visual desde el editor y trabajar con porcentajes.

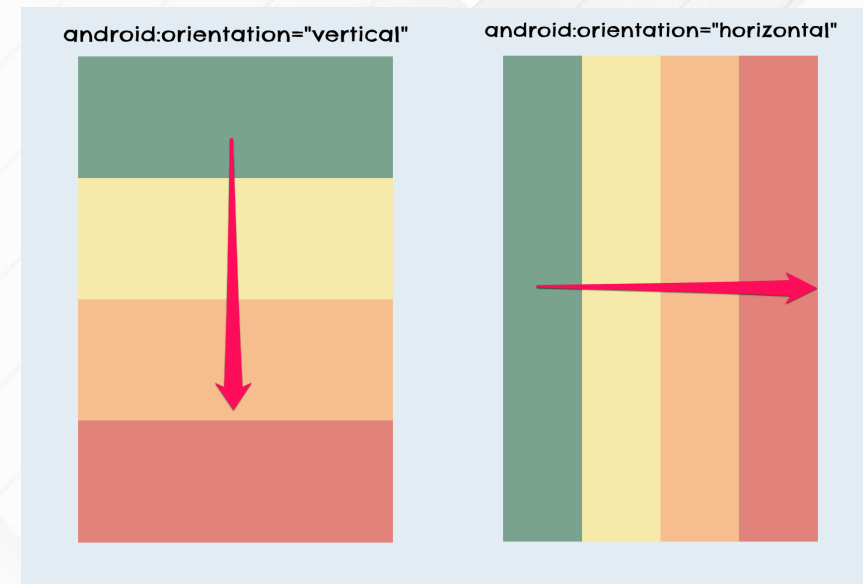


- Componentes de una Aplicación

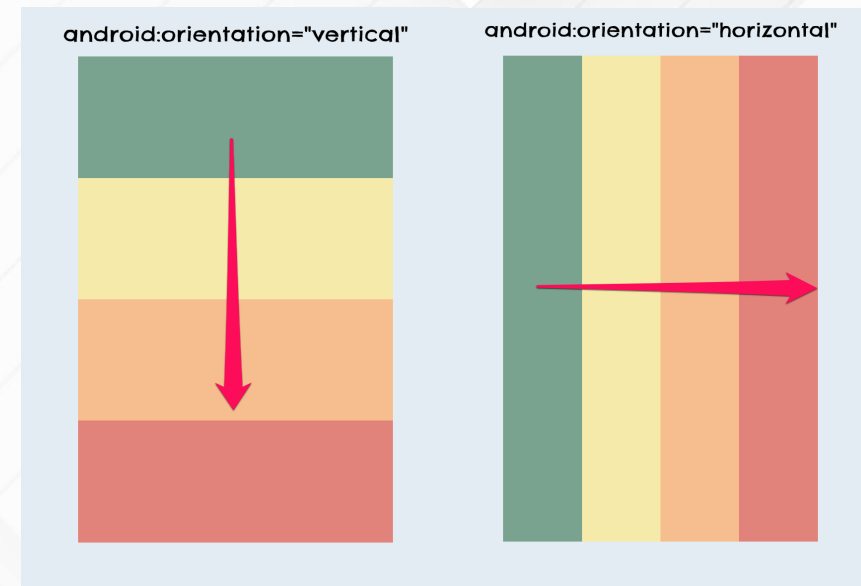
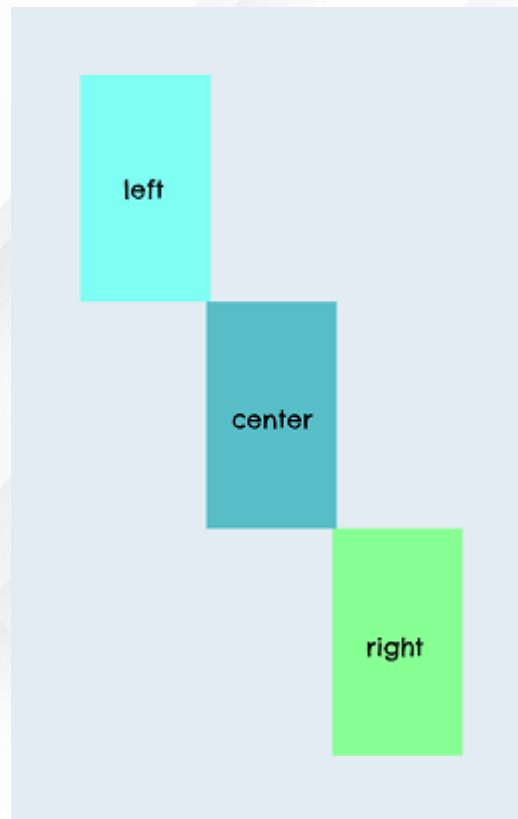
- LAYOUTS – LINEARLAYOUT



- Componentes de una Aplicación
  - LAYOUTS – LINEARLAYOUT
  - ViewGroup que distribuye sus hijos en una sola dimensión establecida
  - Todos organizados en una sola columna (vertical) o en una sola fila (horizontal).
  - La orientación puedes elegirla a través del atributo android:orientation



- Componentes de una Aplicación
  - LAYOUTS – LINEARLAYOUT
  - Este layout permite asignar una **gravedad** a cada componente según el espacio que ocupa



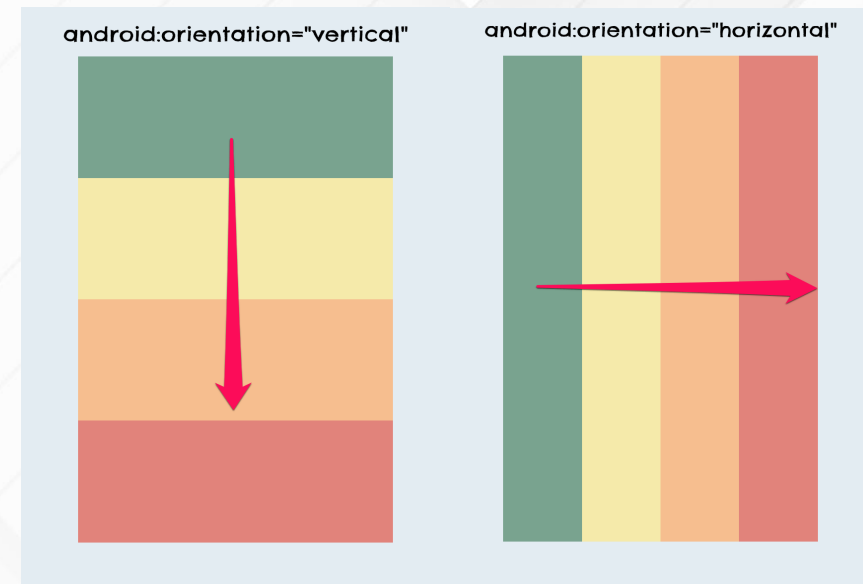
- Componentes de una Aplicación

- LAYOUTS – LINEARLAYOUT

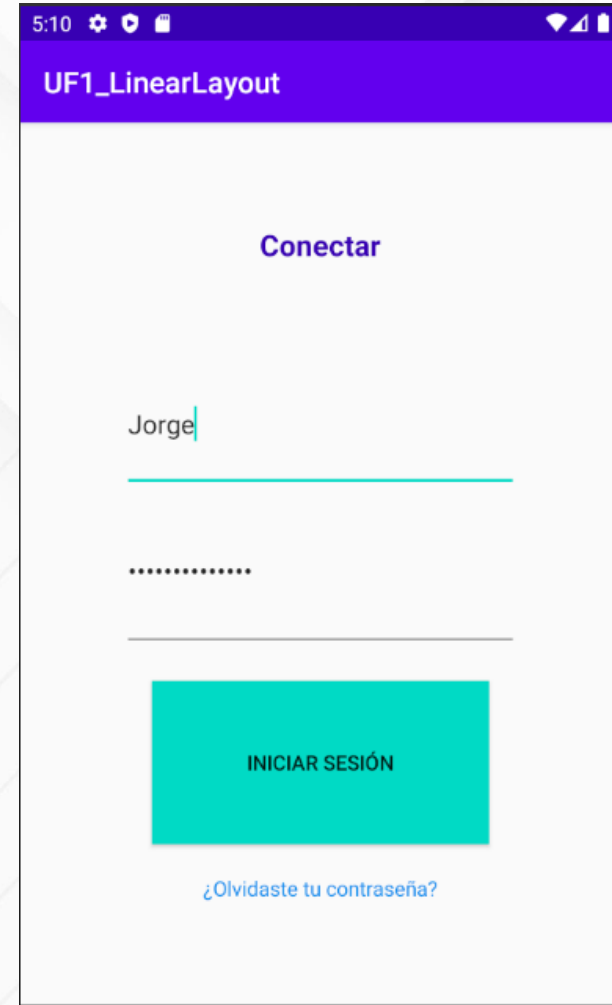
- Además existe un parámetro llamado `android_layout_weight` que define la importancia que tiene un view dentro del `LinearLayout`

- A mayor importancia, mayor espacio ocupa

`android:weightSum="6"`



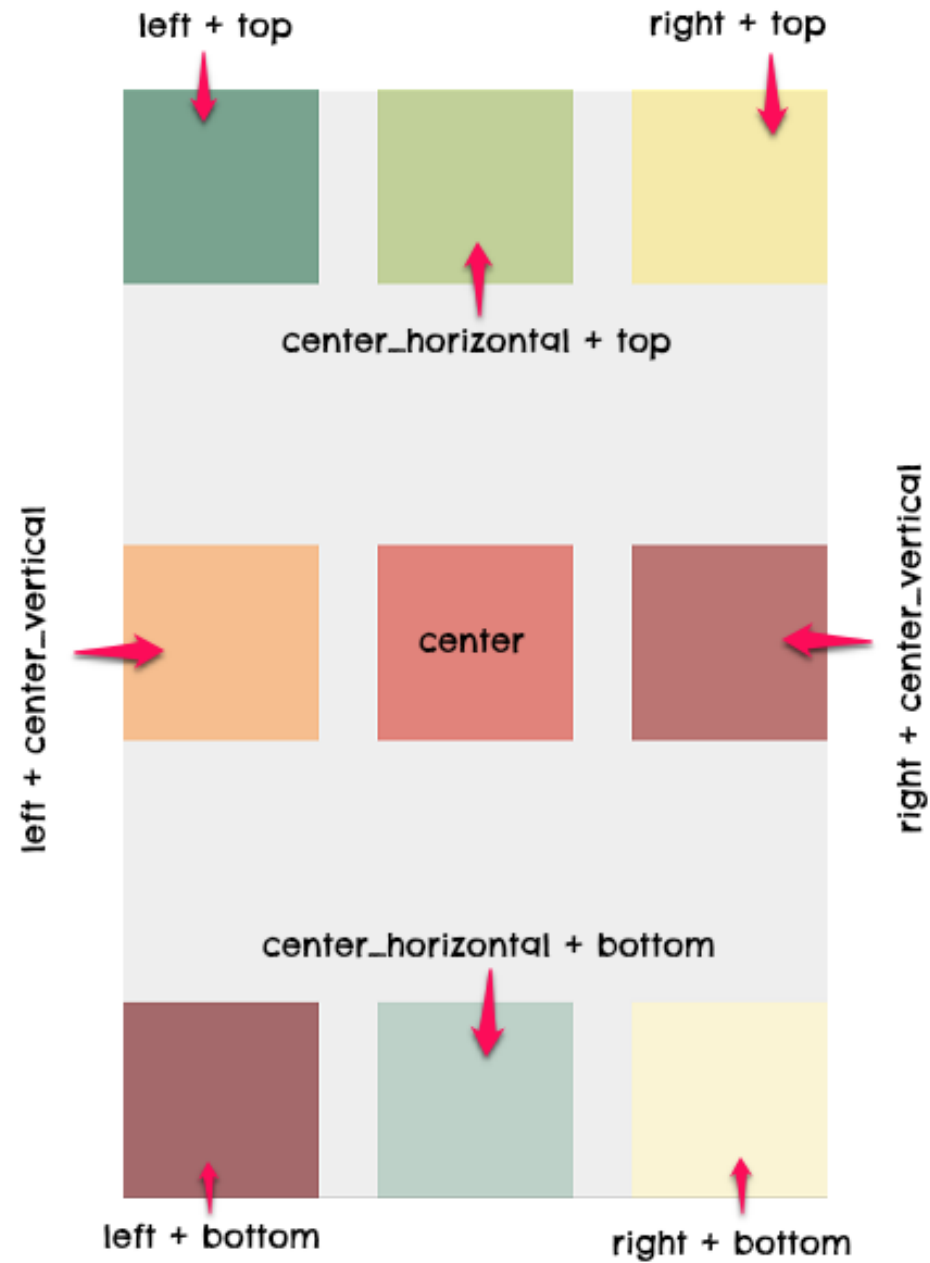
- PARA PRACTICAR
- Abre Android Studio y crea un proyecto para crear una pantalla como la que se muestra en la imagen
- Pantalla de Login con un texto, dos campos para introducir datos, y botón y un texto para solicitar la contraseña olvidada en un LinearLayout



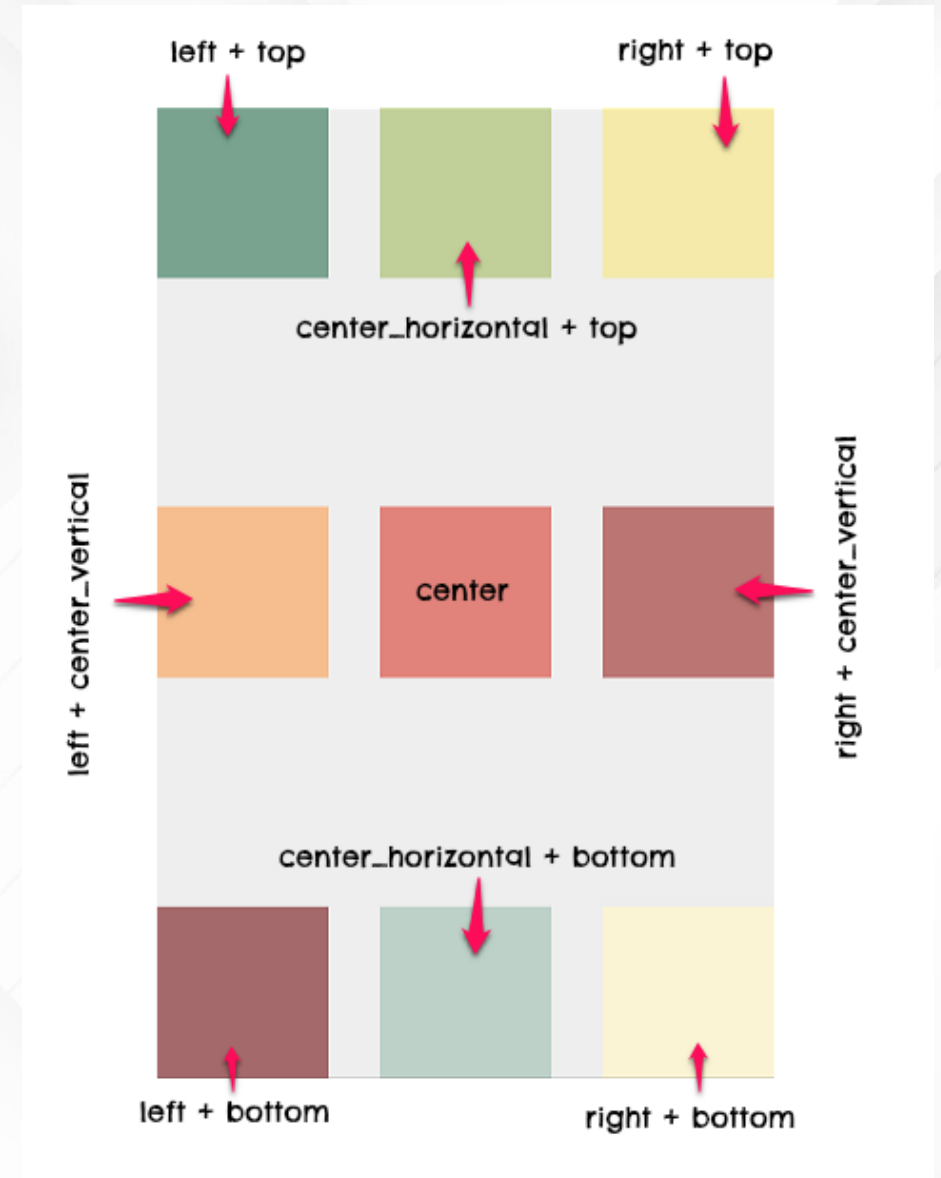


## > Conceptos básicos

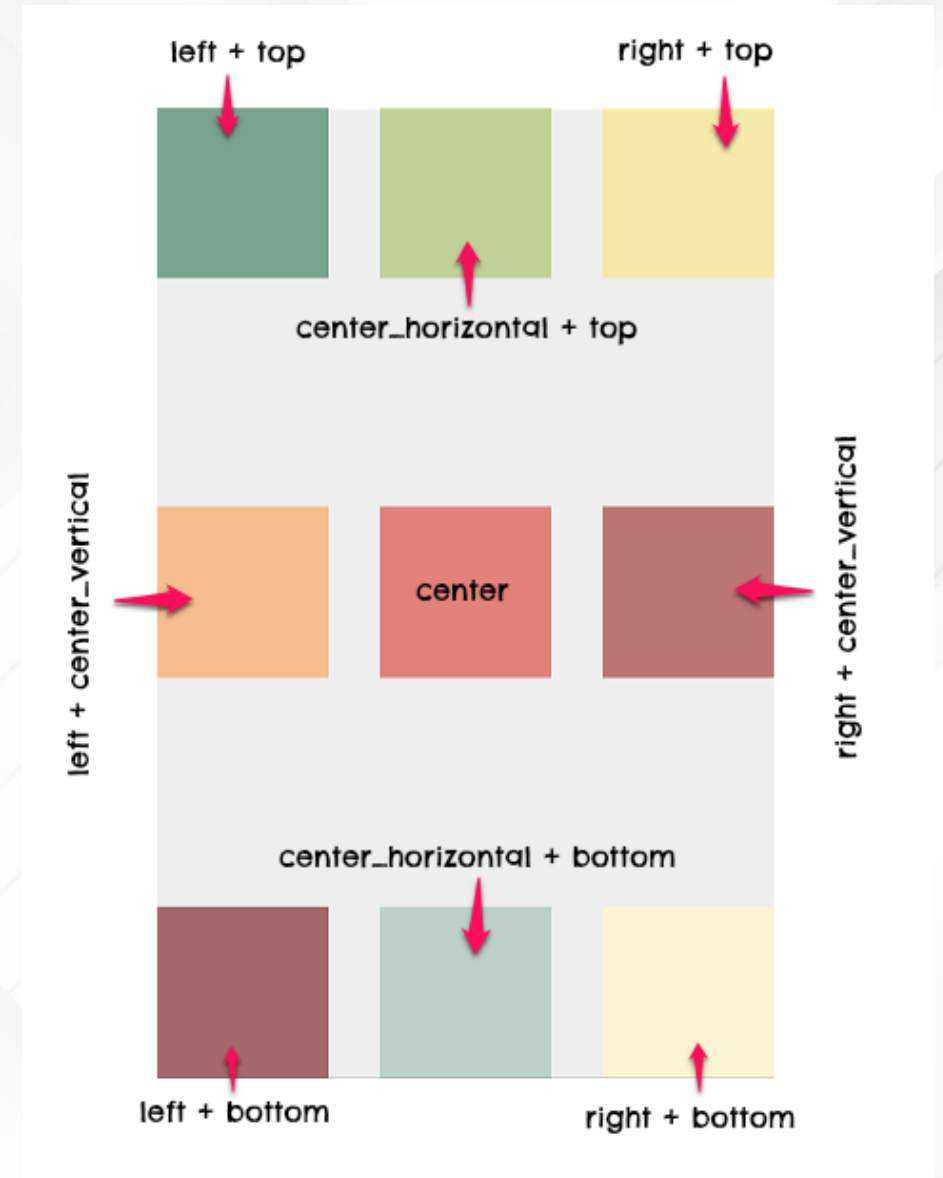
- Componentes de una Aplicación
  - LAYOUTS – FRAMELAYOUT



- Componentes de una Aplicación
  - LAYOUTS – FRAMELAYOUT
  - Es un view group creado para mostrar un solo elemento en pantalla
  - Sin embargo puedes añadir varios hijos con el fin de superponerlos, donde el ultimo hijo agregado, es el que se muestra en la parte superior y el resto se pone por debajo en forma de pila
  - Alinearemos cada elemento del FrameLayout con el parámetro android:layout\_gravity



- Componentes de una Aplicación
  - LAYOUTS – FRAMELAYOUT
  - El parámetro gravity se basa en las posiciones comunes de un view dentro del layout.
  - Se describe con constantes de orientación:
  - top: parte superior del layout
  - left: parte izquierda del layout
  - right: parte derecha del layout
  - bottom: límite inferior del layout



- Componentes de una Aplicación

- LAYOUTS – FRAMELAYOUT

- El parámetro gravity se basa en las posiciones comunes de un view dentro del layout. Se describe con constantes de orientación:

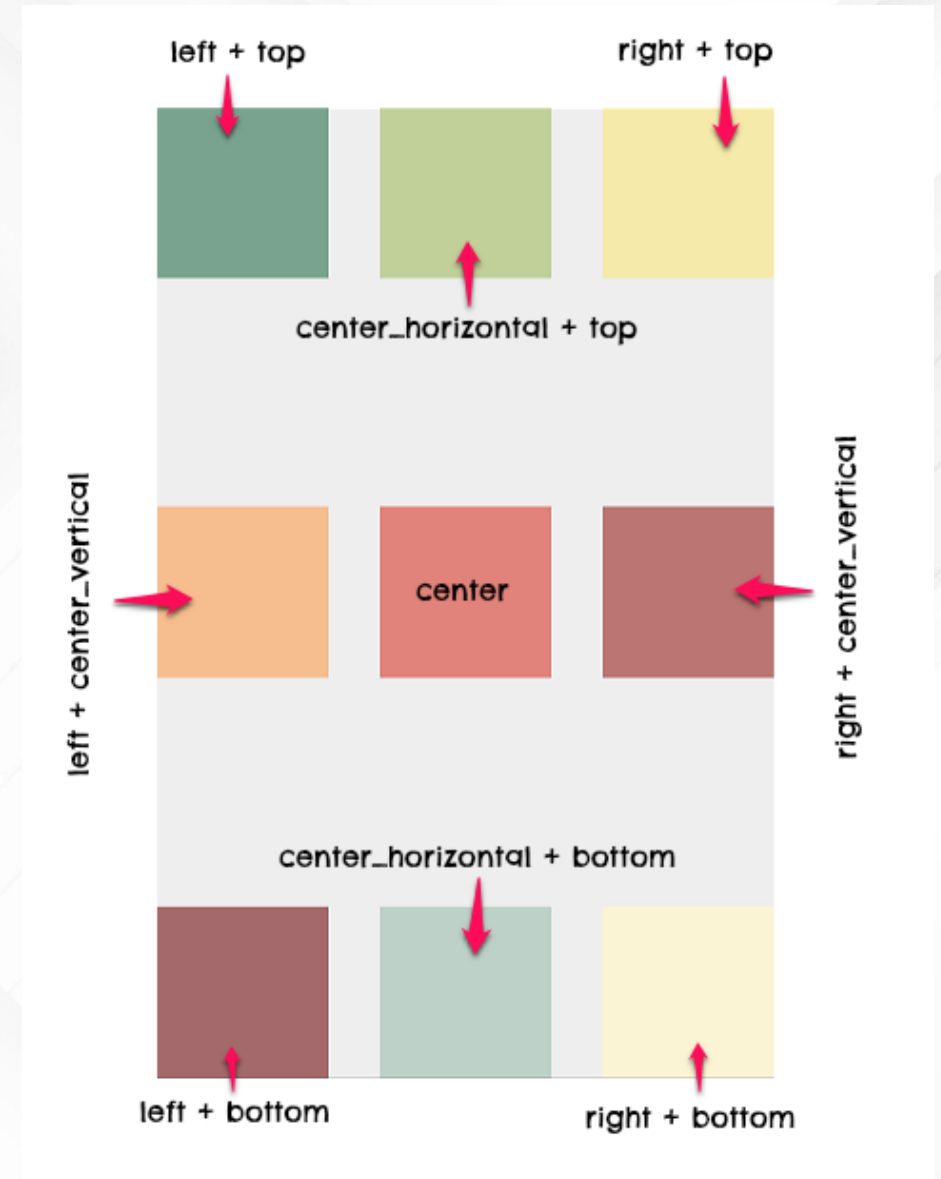
- center\_horizontal: centro horizontal

- center\_vertical: centro vertical

- center: combinación de centro vertical y horizontal

- Es posible crear variaciones combinadas, como right + bottom que se definiría así:

- android:layout\_gravity="right|bottom"



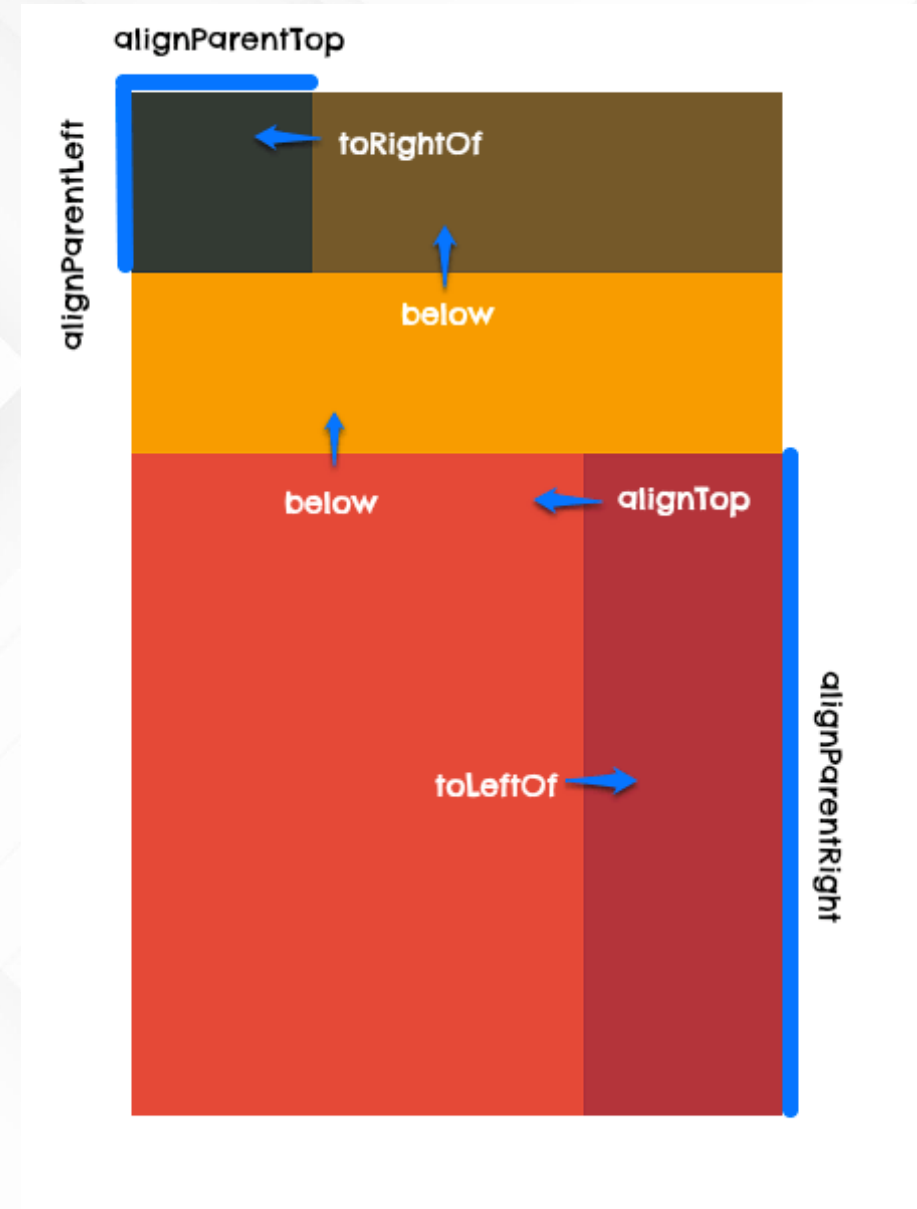
## > Conceptos básicos

- PARA PRACTICAR
- Abre Android Studio y crea un proyecto para crear una pantalla como la que se muestra en la imagen
- La pantalla contendrá una imagen de fondo, una imagen en una capa superior con fondo transparente y un botón en la parte inferior



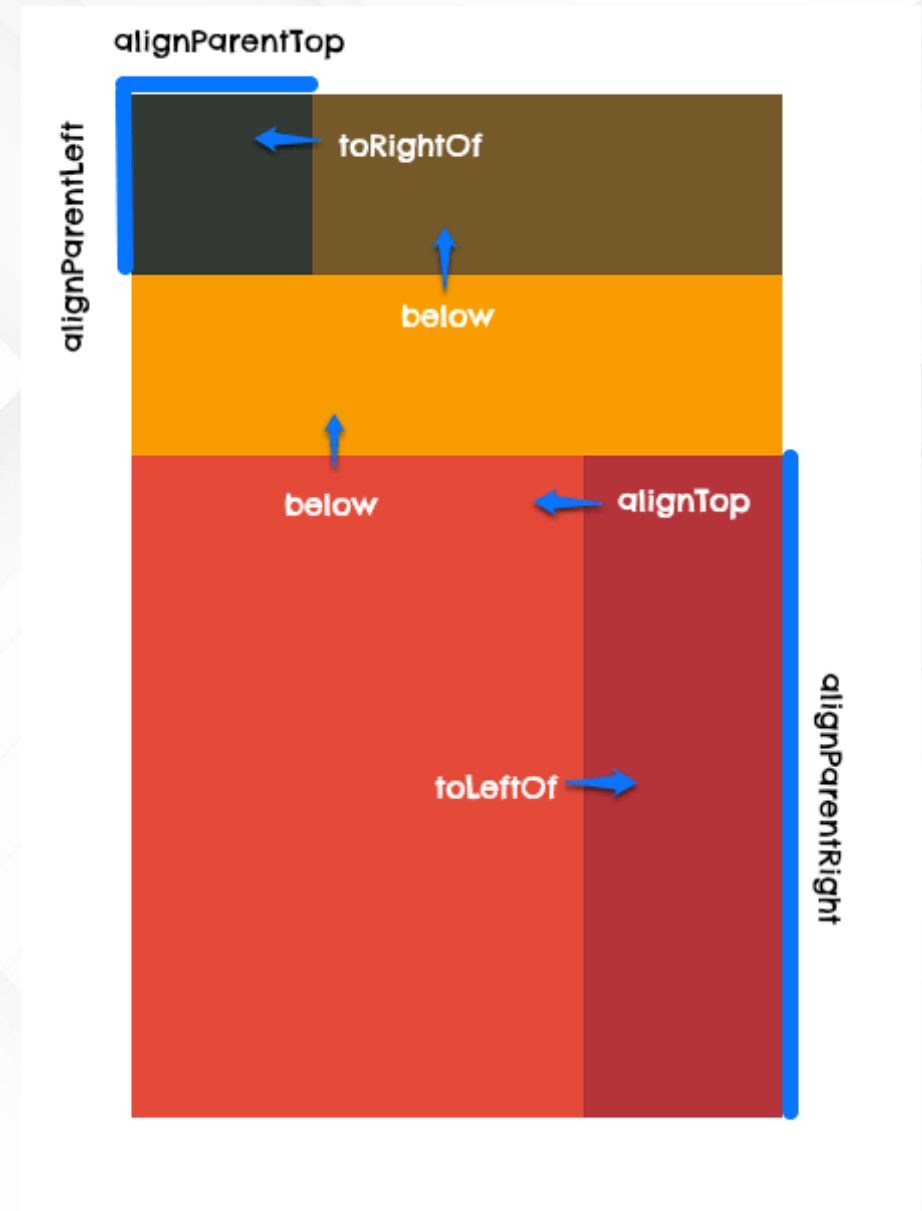


- Componentes de una Aplicación
  - LAYOUTS – RELATIVE LAYOUT
  - Es el elemento más flexible y elaborado de todos los view groups
  - Permite alinear cada hijo con referencias subjetivas de cada hermano
  - Alinear los bordes de cada view con otros
  - En la imagen vemos un ejemplo irregular
  - Esto es posible gracias a unos parámetros que determinan como se juntan los bordes de cada uno y en que alineación
  - Cada referencia es indicada usando el identificador de cada view

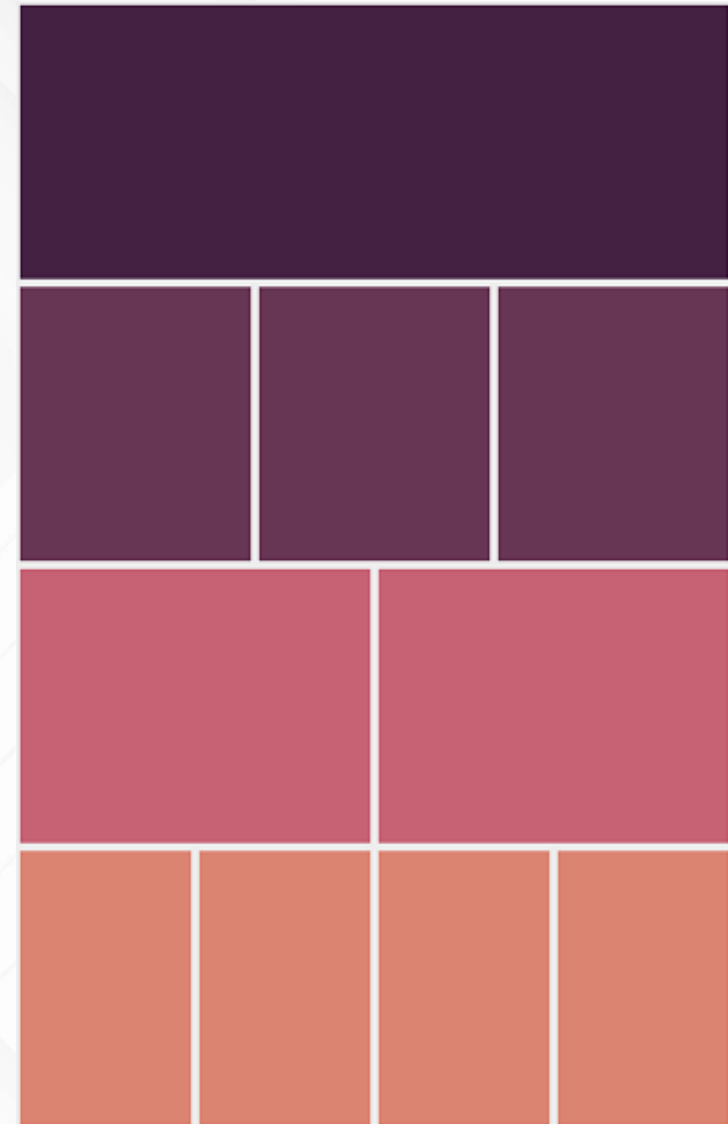


- Componentes de una Aplicación

- LAYOUTS – RELATIVE LAYOUT
- Parámetros para definir posiciones
- `android:layout_above`: posiciona el borde inferior del elemento actual con el borde superior del view referenciado por id
- `android:layout_centerHorizontal`: usa true para indicar que el view será centrado horizontalmente con respecto al padre
- `android:layout_alignParentBottom`: Usa true para alinear el borde inferior de este view con el borde inferior del padre
- `android:layout_alignStart`: alinea el borde inicial de este elemento con el borde inicial del view referido por id

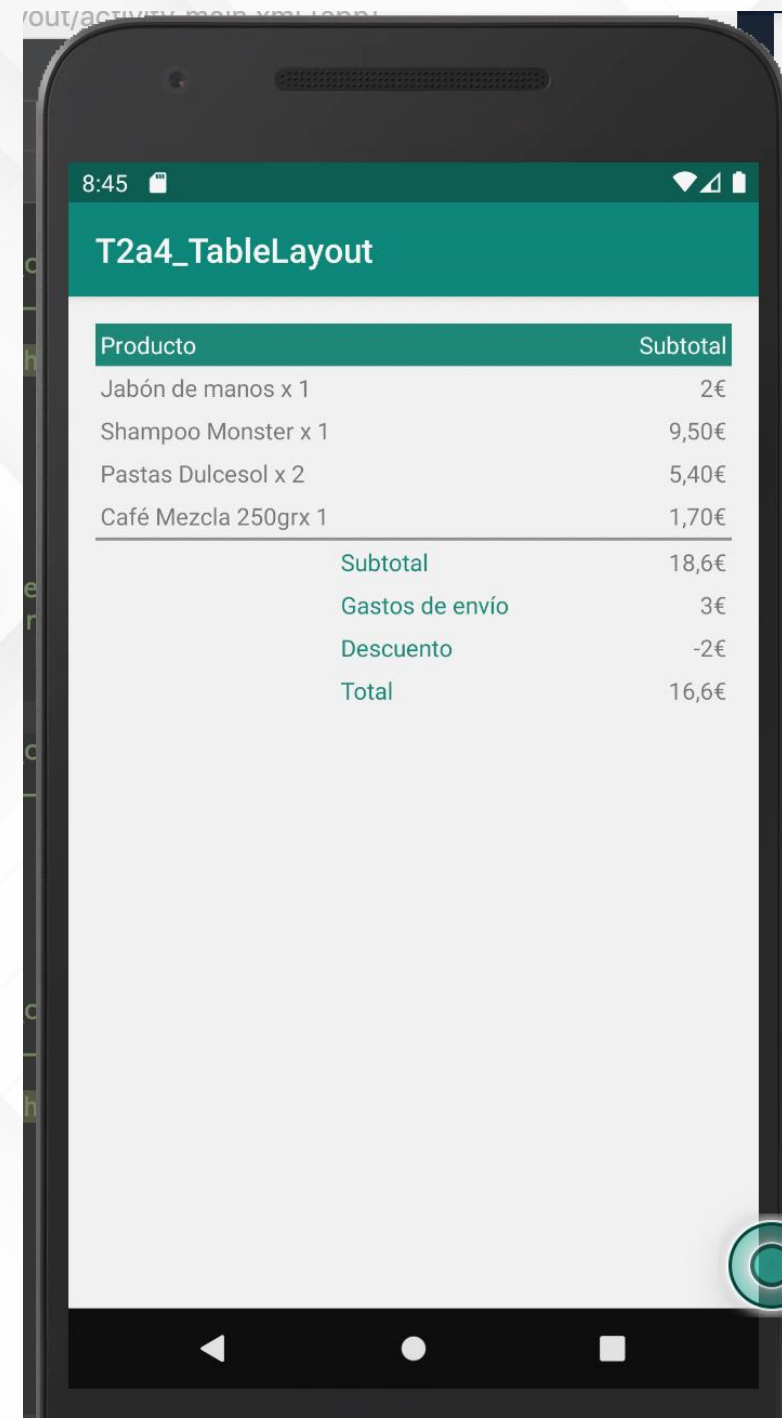


- Componentes de una Aplicación
  - LAYOUTS – TABLELAYOUT
  - TableRow tiene un parámetro llamado `android:layout_column` para asignar la columna a la que pertenece cada celda en su interior
  - Además, podemos usar el parámetro `weight` para declarar el peso de las celdas
  - El ancho de cada columna es definido tomando como referencia la celda más ancha
  - Aunque también podemos definir el comportamiento del ancho de las celdas con los siguientes atributos:
    - `android_shrinkColumns`, reduce ancho de columna seleccionada hasta ajustar fila al ancho del padre
    - `android:stretchColumns`, rellena el espacio vacío que queda en el `TableLayout` expandiendo la columna seleccionada

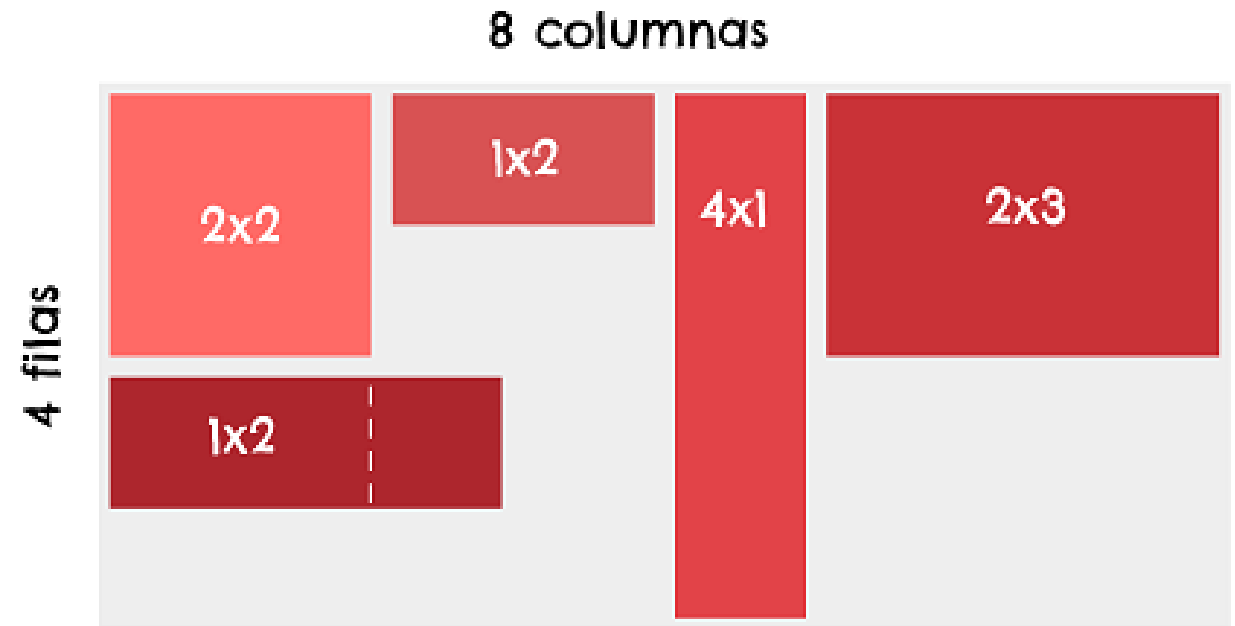


## > Conceptos básicos

- PARA PRACTICAR
- Abre Android Studio y crea un proyecto para crear una pantalla como la que se muestra en la imagen
- Deberás usar un `TableLayout` para representar la información
- Puedes modificar el diseño a tu gusto para practicar

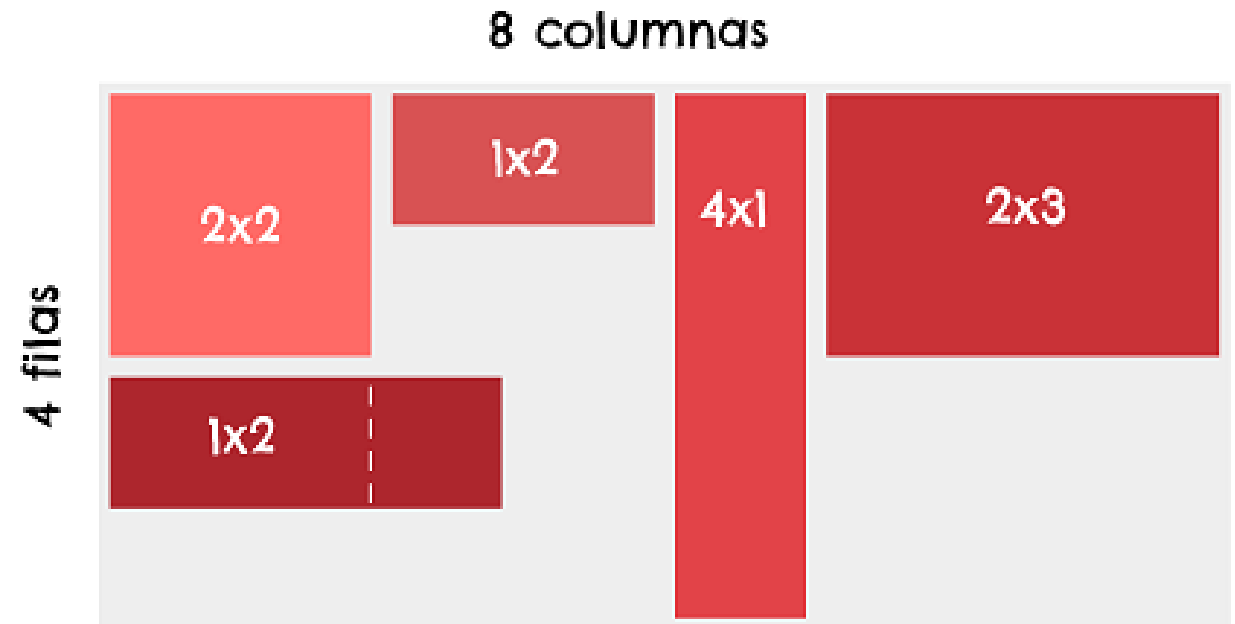


- Componentes de una Aplicación
  - LAYOUTS – GRIDLAYOUT
  - Con este layout se alinean los elementos hijos en una cuadrícula (grilla ó grid)
  - Nace con el fin de evitar anidar linear layouts para crear diseños complejos
  - Su funcionamiento se basa en un sistema de índices con inicio en cero
  - La primera columna (o fila) tiene asignado el índice 0, la segunda el 1, la tercera el 2, etc.





- Componentes de una Aplicación
  - LAYOUTS – GRIDLAYOUT
  - Los atributos más importantes del GridLayout son:
    - `columnCount`: número de columnas de la grilla
    - `rowCount`: número de filas de la cuadrícula
    - `useDefaultMargins`: para establecer márgenes predeterminados entre los ítems
  - Los parámetros que más vamos a usar son:
    - `android:layout_columnSpan`: para especificar el número de columnas que ocupará una celda
    - `android:layout_rowSpan`: para especificar el número de filas que ocupará una celda

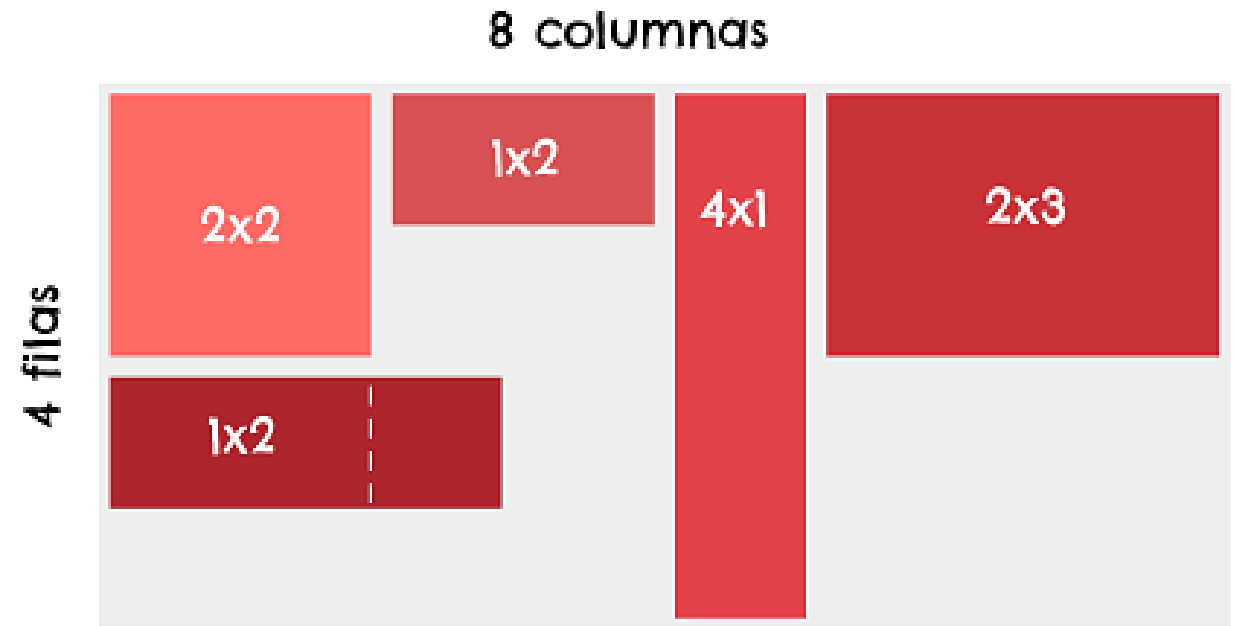


**Esto nos permite crear diseños irregulares que no podríamos hacer con `TableLayout`**

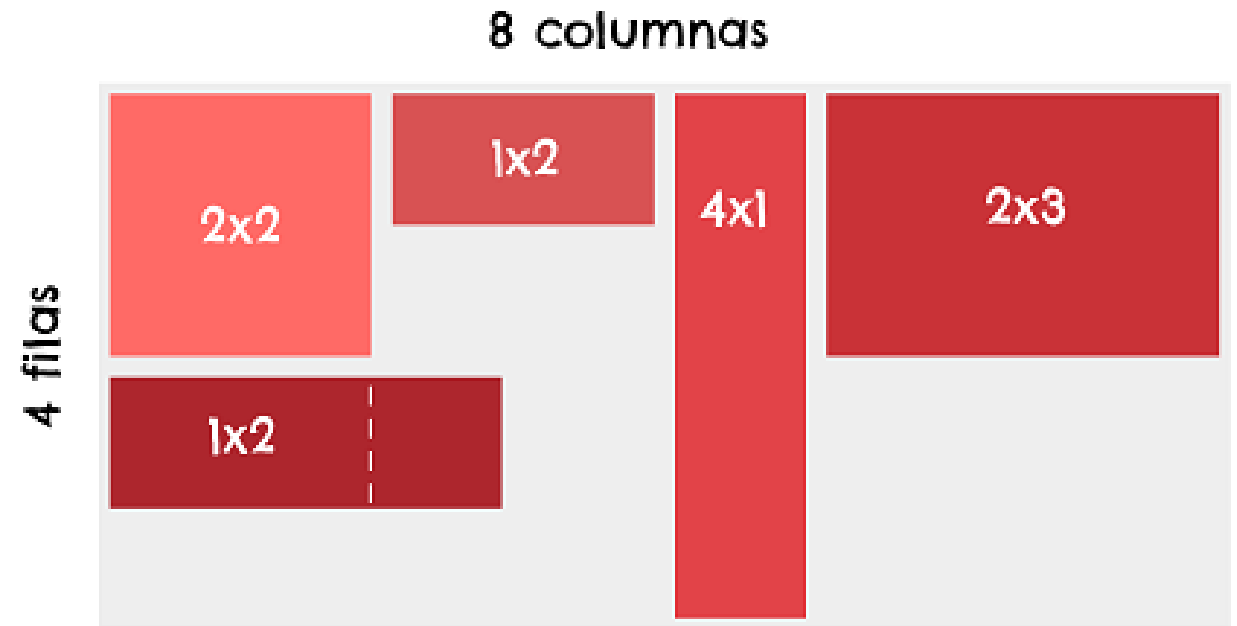
## > Conceptos básicos

- Componentes de una Aplicación
  - LAYOUTS – GRIDLAYOUT
  - Supongamos que un TextView ocupe dos columnas y dos filas.
  - ¿Cómo lo podríamos escribir?

```
<TextView
    android:id="@+id/celda_1"
    android:layout_columnSpan="2"
    android:layout_rowSpan="3"
    android:text="Celda 1" />
```



- Componentes de una Aplicación
  - LAYOUTS – GRIDLAYOUT
  - El item que se encuentra en la segunda fila con las especificaciones 1x2 se cruza en la columna 3
  - Esto se debe a que su ancho es de 3 unidades, pero su atributo `columnSpan` es igual a 2, lo que facilita al framework crear el cruce si existe espacio en blanco



- Componentes de una Aplicación
  - LAYOUTS
  - Cargar un layout XML en Android
  - Al tener definido un recurso, ya es posible inflar su contenido en la actividad
  - Para ello usaremos el método `setContentView()` dentro del controlador `onCreate()`

```
activity_main.xml x MainActivity.kt x
1 package com.example.primerproyecto
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    }
11 }
```

- Componentes de una Aplicación

- LAYOUTS

- Identificador de un View

- Existe un atributo que heredan todos los elementos llamado id
- Representa un identificador único para cada elemento.
- Esto permite obtener una referencia de cada objeto de forma específica
- La sintaxis para el id sería la siguiente

```
android:id="@+id/nombre_identificador"
```



- Es un layout con restricciones (constraint), por la que fijaremos su posición respecto al dispositivo o al resto de elementos.
- Se deben fijar los elementos en horizontal y en vertical.
- Nos permite crear una interfaz responsive.



- Nos permite crear nuevos elementos que podremos usar en nuestros diseños
- Se realiza mediante XML
- Selector → Lista de estados de una imagen
- Item → Cada elemento que deseamos dibujar
  - Cada <item> usa varios atributos para describir el estado en que debería usarse como gráfico del elemento de diseño
- Shape → Forma
- android:shape = "rectangle" | "oval" | "line" | "ring"

```
<?xml version="1.0" encoding="UTF-8"?>
<shape xmlns:android=
"http://schemas.android.com/apk/res/android">
  <gradient
    android:angle="135"
    android:endColor="@color/blue"
    android:startColor="@color/light_blue" />
</shape>
```

## > Conceptos básicos

- PARA PRACTICAR
- Abre Android Studio y crea un proyecto para crear una pantalla como la que se muestra en imagen
- Pantalla para sumar dos números introducidos por el usuario





- ¿Qué Kotlin?
- Lenguaje de programación desarrollado por el equipo de JetBrains (IntelliJ IDEA, Android Studio, etc)
- Necesitaban un lenguaje más conciso que Java que compilara en JVM
- En el desarrollo de Android, compila a bytecode JVM y también se puede compilar con JavaScript
- Compatible con Java





- Google lo ha proclamado lenguaje oficial en Android al mismo nivel que Java.
- Podemos convertir código de Kotlin a Java y viceversa con un plugin de JetBrains
- En Android, se integra por Gradle
- Gradle es un paquete de herramientas de compilación avanzadas, para automatizar y administrar el proceso de compilación, y al mismo tiempo definir configuraciones de compilación personalizadas y flexibles
- Cada configuración de compilación puede definir su propio conjunto de código y recursos, y reutilizar las partes comunes a todas las versiones de tu app

- CARACTERÍSTICAS DE KOTLIN

- **1. KOTLIN ES CONCISO**

- Uso de Lambdas
- Un lambda es una forma de representación de una función
  - `boton.setOnClickListener (escuchadores. Procesos que están escuchando en segundo plano y saltan cuando detectan una pulsación en la pantalla) { // instrucciones }`
- Uso de clases
  - `class Persona { var nombre: String "Juan" }`
- No se necesita la referencia `findViewById`
  - `textView.text = " Bienvendo a Kotlin "`

- CARACTERÍSTICAS DE KOTLIN
- **2. KOTLIN ES LEGIBLE Y EXPRESIVO**
  - Código más legible
  - Desaparecen los getters y setters
  - Bucles for y when más expresivos
  - Se hereda la funcionalidad de una clase sin necesidad de usar la herencia tal y como la conocemos

- CARACTERÍSTICAS DE KOTLIN
- **3. KOTLIN ES SEGURO**
  - Impide generar NullPointerException
  - Variables Inmutables como constantes
    - Variable: var nombre = “Juan”
    - Variable inmutable: val (como una constante) iva = 0.21

- CARACTERÍSTICAS DE KOTLIN
- **3. KOTLIN ES COMPATIBLE CON JAVA**
  - Podemos usar las clases y librerías de Java

```
package com.example.mi_app;  
import android.util.Log;  
import android.view.View;  
import android.widget.EditText;  
...
```

```
package com.example.mi_app;  
import android.util.Log;  
import android.view.View;  
import android.widget.EditText;  
...
```

- En un mismo proyecto, podemos combinar bloques en Kotlin y bloques en Java



- DECLARACIÓN DE VARIABLES EN KOTLIN

JAVA	KOTLIN	KOTLIN
<pre>int a = 12; String s = "Juan"; double d = 3.1415;</pre>	<pre>var a: Int = 12 var s: String = "Juan" var d: Double = 3.1415</pre>	<pre>var a = 12 var s = "Juan" var d = 3.1415</pre>

- Tipos de variables

- var: variables mutables, que pueden cambiar su valor
- val: variables inmutables, que permanecen constantes

JAVA	KOTLIN	KOTLIN
<pre>final int a = b + 3; final String s = "Juan"</pre>	<pre>val a = b +3; val s = "Juan"</pre>	<pre>val i = 1</pre>

- Valores conocidos en tiempo de compilación

KOTLIN	KOTLIN
<pre>const val PI = 3.1416</pre>	<pre>const val PAIS = "España"</pre>

- TIPOS DE DATOS PRIMITIVOS EN KOTLIN
- Los tipos de datos primitivos en Java (int, double, char, etc) se convertirán en clases (Int, Double, Char, etc)
- Internamente no son clases, externamente son objetos, pero internamente son datos primitivos

tipo	valor	tamaño	rango	
Byte	5.toByte()	8 bits	-128	127
Short	5.toShort()	16 bits	-32.768	32.767
Int	5	32 bits	-2.147.483.648	2.147.483.647
Long	5L	64 bits	$-9.223.372 \cdot 10^{12}$	$9.223.372.036.854.775.807$
Float	1.45F	32 bits	$-3.4 \cdot 10^{38}$	$3.4 \cdot 10^{38}$ (mínimo $1.4 \cdot 10^{-45}$ )
Double	1.45	64 bits	$-1.8 \cdot 10^{308}$	$1.8 \cdot 10^{308}$ (mínimo $4.9 \cdot 10^{-324}$ )
Boolean	<b>true</b>		false	true
Char	'H'	16 bits	Unicode 0	Unicode $2^{16}-1$
Unit	Unit	0 bits	-	-

- CONVERSIONES DE TIPOS DE DATOS
- Se usan las funciones toXXX()

- `val c: Char = 'c'`

- `val i : Int = c.toInt()`

- TIPO STRING
- Unir dos cadenas con operador +
  - `println("Hola " + nombre + ", buenos " + "días")`
- Insertar variables o expresiones en cadena con \$
  - `println("La cadena $s tiene ${s.length} caracteres")`
- Añadir un carácter especial con \$
  - `println("El precio total es 123 ${'€'})`
- Usar comillas triples para añadir saltos de línea y tabuladores, en lugar de `\n`, `\t` y `\`

- ESTRUCTURAS DE CONTROL
- IF
  - if (edad > 18)
  - Podemos usar if en una expresión:
    - max = if(a>b){ print(a); a} else{print (b); b}

- WHEN, sustituto de Switch
  - when(x){
    - 1 -> print("Uno")
    - 2,3 -> print("2 ó 3")
    - else -> {
      - println("error")

También se puede usar en expresiones

```
numero = when(x){ x ==1 -> 1  
                s.contains("able") -> 2  
                !x in 2..10 -> 3  
                y is String -> 5  
                else -> 0  
            }
```

- ESTRUCTURAS DE CONTROL
- FOR, más parecido a FOREACH
  - for ( i in 1..100)
  - Recorremos los elementos de una colección con Iterator (next() (desplazarlo) y hasNext() (permite saber si existe un siguiente. Si false, hemos llegado al tope.))
  - Definir colecciones en Kotlin
    - for (i in 4 downTo 1) //4, 3, 2, 1
    - for (i in 1.. 9 step 2) //1, 3, 6, 8
    - for (i in 1 until 4) //1, 2, 3
    - for (c in "Saludos") //'S', 'a', 'l', 'u', 'd', 'o', 's'
  - Podemos usar for con arrays, string y colecciones

```
for (i in array.índices){  
    print(array[i])  
}
```

```
for((índice, valor) in array.withIndex()){  
    println("Posición $índice: valor $valor")  
}
```



- ESTRUCTURAS DE CONTROL

- WHILE

- Igual que en Java

- `while (x > 0){`

```
    x--
```

```
}
```

- DO WHILE

- Igual que en Java

- `do {`

```
    val x = leerNumero()
```

```
} while( x!= null)
```

- ESTRUCTURAS DE SALTO
  - return: sale de la función donde es ejecutado
  - break: termina el bucle en el que es ejecutado
  - continue: salta una iteración en el bucle en el que es ejecutado

- FUNCIONES EN KOTLIN
- Se usa la palabra **fun** para declarar una función
  - `fun areaTriangulo (b: Double, h: Double) : Double{`

```
    return (b*h)/2
```

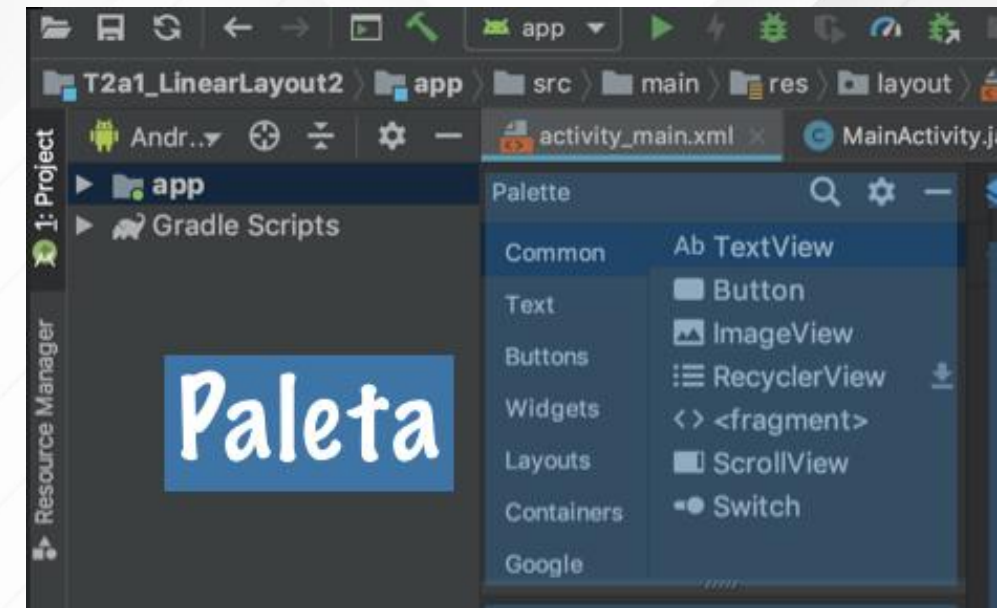
```
}
```

- Se sigue la notación de parámetros y tipo de vuelto que se usa en Pascal
- Se puede omitir el tipo que devuelve
- No podemos agrupar varias variables del mismo tipo
  - `fun función(a: Int, b: Int) { }`



# TRABAJANDO CON ANDROID STUDIO

- PALETA
- Contiene todos los elementos gráficos posibles que podemos implementar dentro de un layout.
- Desde este lugar puedes seleccionar cualquier elemento y arrastrarlo al lienzo
- Dentro de esta existen varias categorías, como Layouts, donde se encuentran los layouts que hemos visto hasta ahora

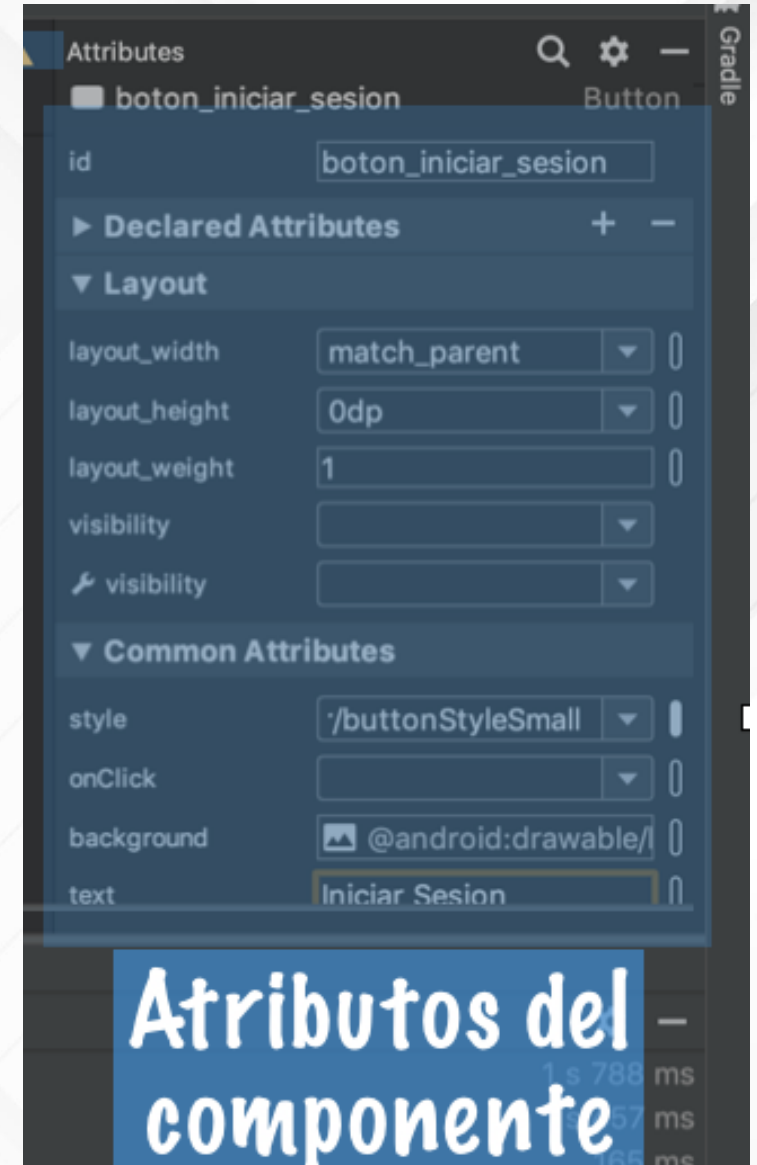


- LIENZO
- Es la representación visual en tiempo real de la interfaz de la aplicación
- Cuando arrastras un elemento de la paleta hacia el lienzo, este proyecta guías visuales para indicarte como será acomodado el view si lo sueltas en esa posición

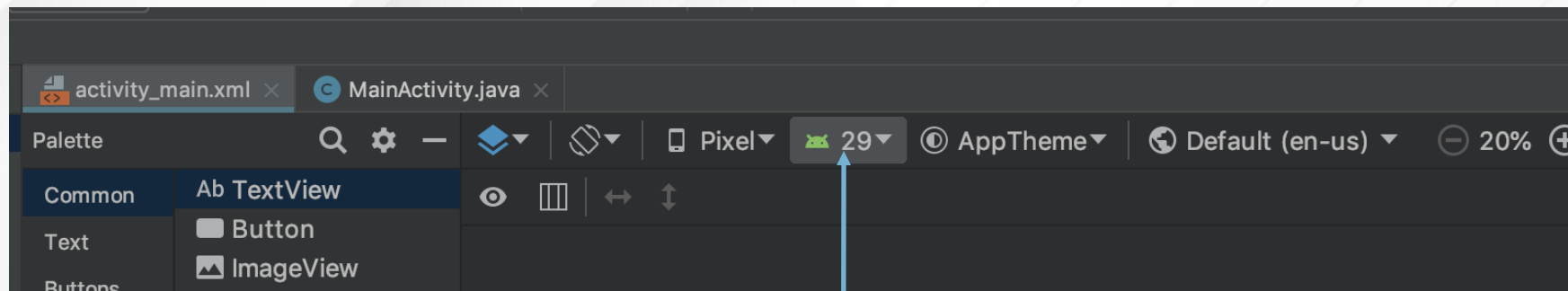




- PANEL DE COMPONENTES
- Muestra los atributos del view seleccionado
- En él veremos una lista con pares clave-valor para escoger las opciones que deseamos sin ver el código XML

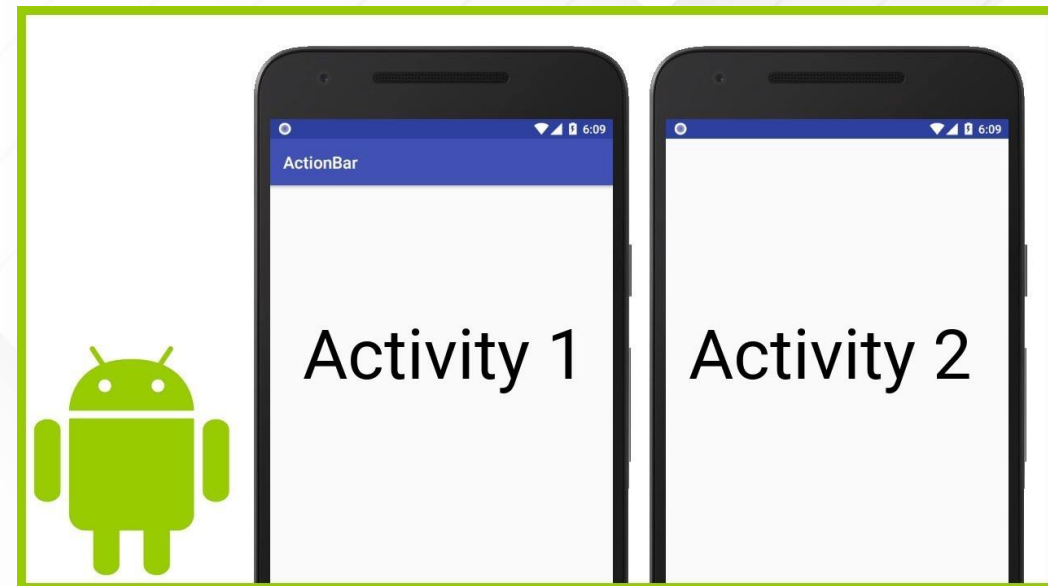


- CAMBIO DE VERSIÓN DE ANDROID
- Cambiar la versión de Android
- Para cambiar la versión haz clic en el icono que se indica en la parte superior derecha y despliega la lista de versiones disponibles
- Selecciona la que desees y analiza el cambio



Cambio de Versiones

- ACTIVITY
- Sirven para representar una pantalla de la aplicación
- Su función es la creación del interfaz de usuario
- Una aplicación estará formada por un conjunto de actividades independientes, aunque todas trabajando para un objetivo común
- Toda actividad ha de ser una subclase de Activity

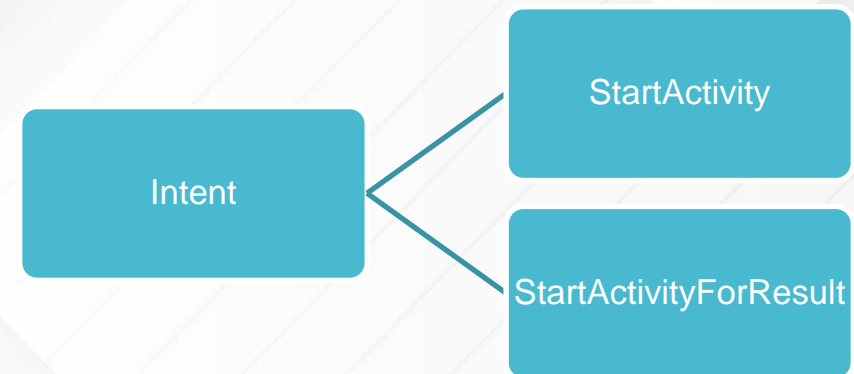
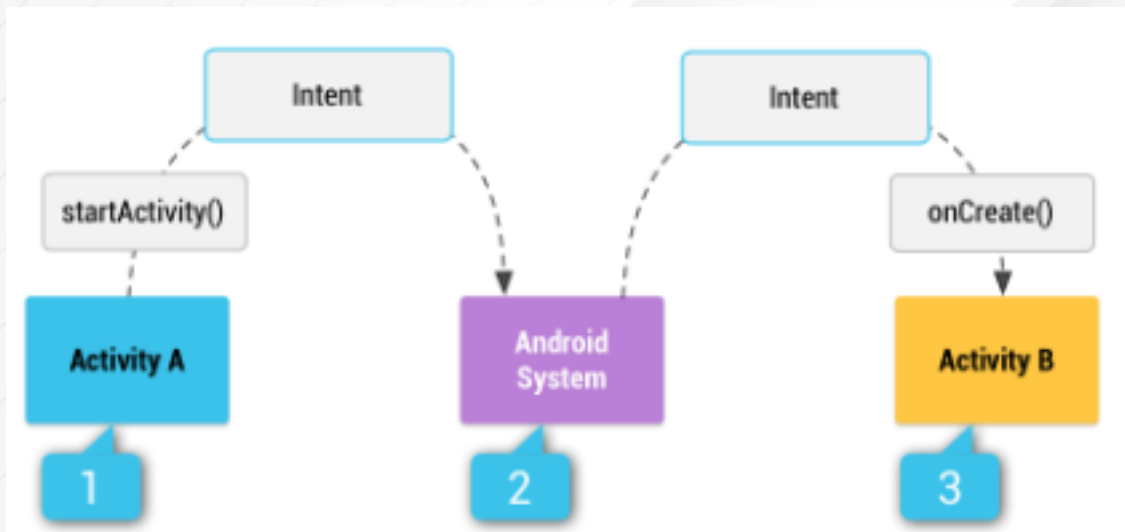




## > CREAR UN INTENT

Una Activity representa una única pantalla en una aplicación.

Puedes iniciar una nueva instancia de una Activity se debe usar un Intent, quien describe la actividad que se debe iniciar y contiene los datos necesarios para ello



```
val intent = Intent (this, Main2Activity::class.java);  
startActivity(intent);
```

Bundle: Elemento utilizada para pasar información entre Activities.

Se podría definir como un `Map<K,V>`, elemento que puede guardar una clave y su valor asociado.

Los Bundle van dentro de un Intent por tanto podremos hacer uso de el siempre que deseemos pasar información al lanzar las funciones `StartActivity()` o `StartActivityForResult()`;



1. ¿Qué es el PutExtras?

```
@NonNull public @NonNull Intent putExtra(String name, @Nullable String value) {  
    if (mExtras == null) {  
        mExtras = new Bundle();  
    }  
    mExtras.putString(name, value);  
    return this;  
}
```

2. Añadir información al intent, mediante putExtras:

```
fun onClick(view: android.view.View) {  
    val intent = Intent()  
    val datos = "Estos son datos generados en ${SecondActivity::class.simpleName}"  
    intent.putExtra("datos", datos)  
    setResult(RESULT_OK, intent)  
    finish()  
}
```

Una vez se ha pasado la información en el intent se tiene que recuperar:

1. Declarar un elemento del tipo Bundle.
2. Recuperar el Bundle del Intent, mediante getIntent.getExtras();

```
@Nullable public @Nullable Bundle getExtras() {  
    return (mExtras != null)  
        ? new Bundle(mExtras)  
        : null;  
}
```

3. Recuperar el valor del Bundle.

```
val bundle: Bundle? = intent.extras  
val nombre:String = intent.getStringExtra("datos")
```

Podemos lanzar el Intent usando dos métodos

StartActivity → Actividad lanzada sin ánimo de que devuelva un resultado.

```
class MainActivity : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
        btnCambiarSaludo.setOnClickListener {  
            val intent = Intent(this, MainActivity2::class.java)  
            startActivity(intent)  
        }  
    }  
}
```

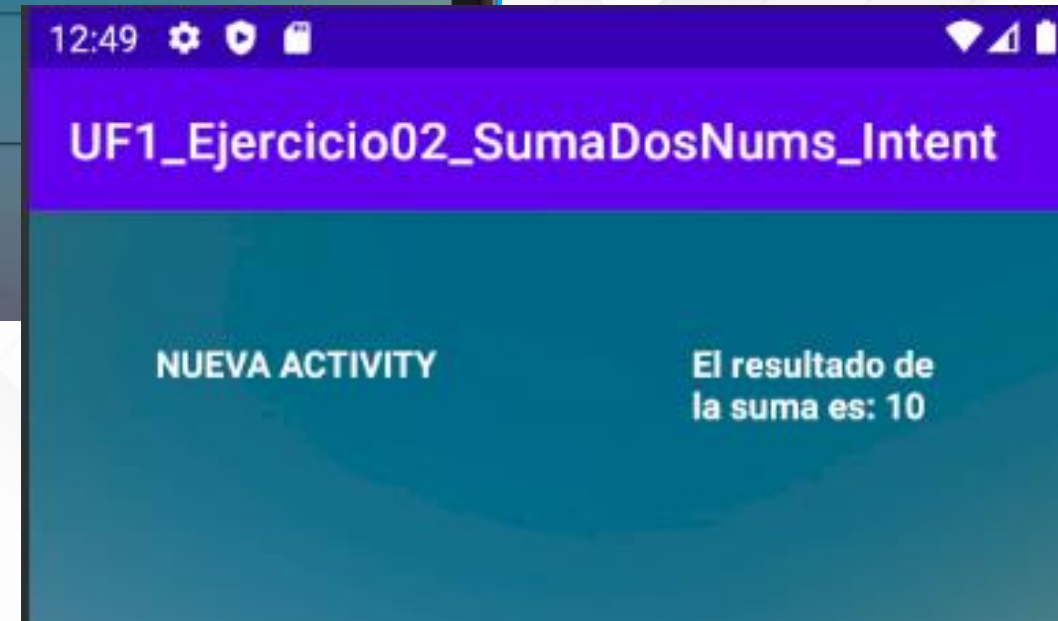
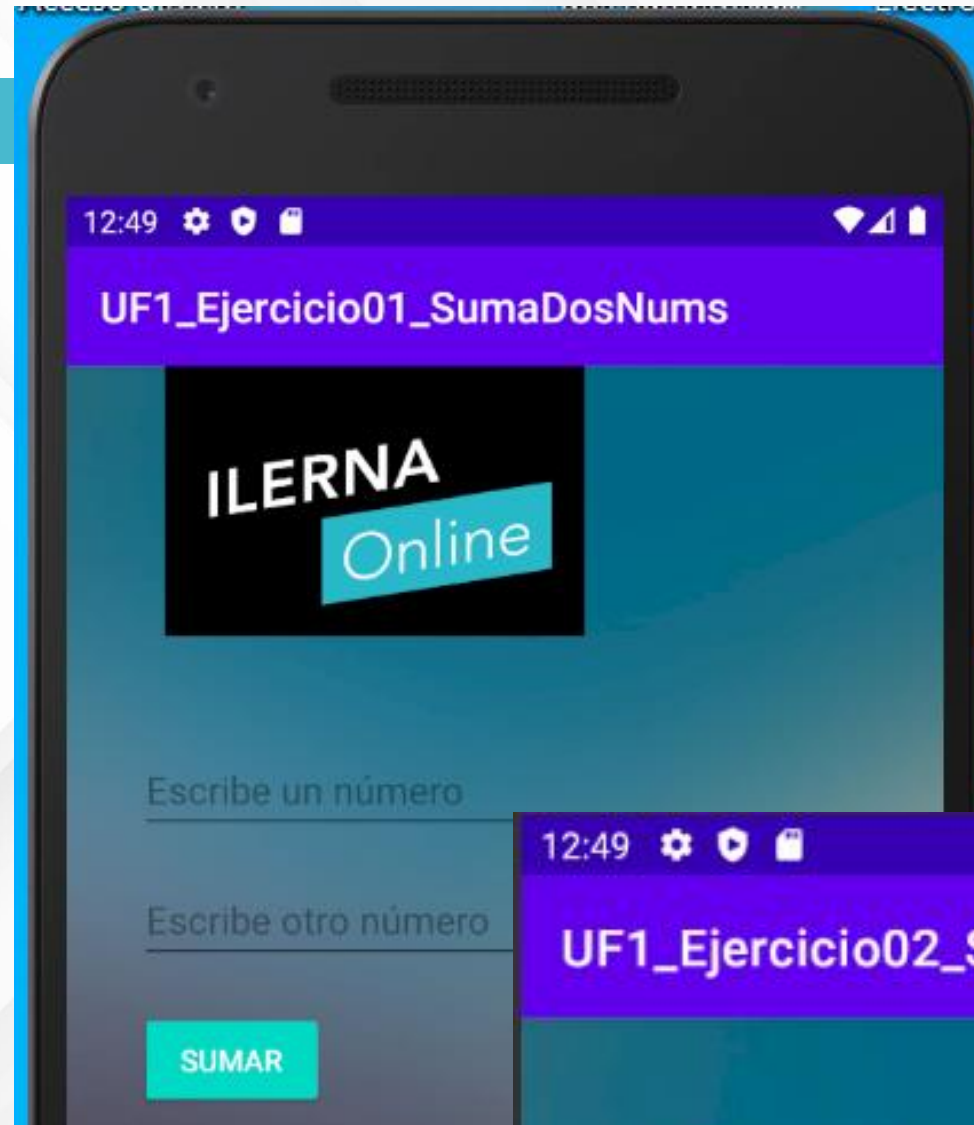
StartActivityForResult() → Se espera un resultado de la actividad para luego ser tratado.

```
btnLanzarSegundaActividad.setOnClickListener {  
    val intent = Intent(this, SecondActivity::class.java)  
    startActivityForResult(intent, REQUEST_SEGUNDA_ACTIVIDAD)  
}
```

Las Activities  
previas se  
guardan en una  
pila de Activities

## > Conceptos básicos

- PARA PRACTICAR
- Abre Android Studio y crea un proyecto para crear una pantalla como la que se muestra en la imagen
- El resultado de la suma se mostrará en una nueva pantalla





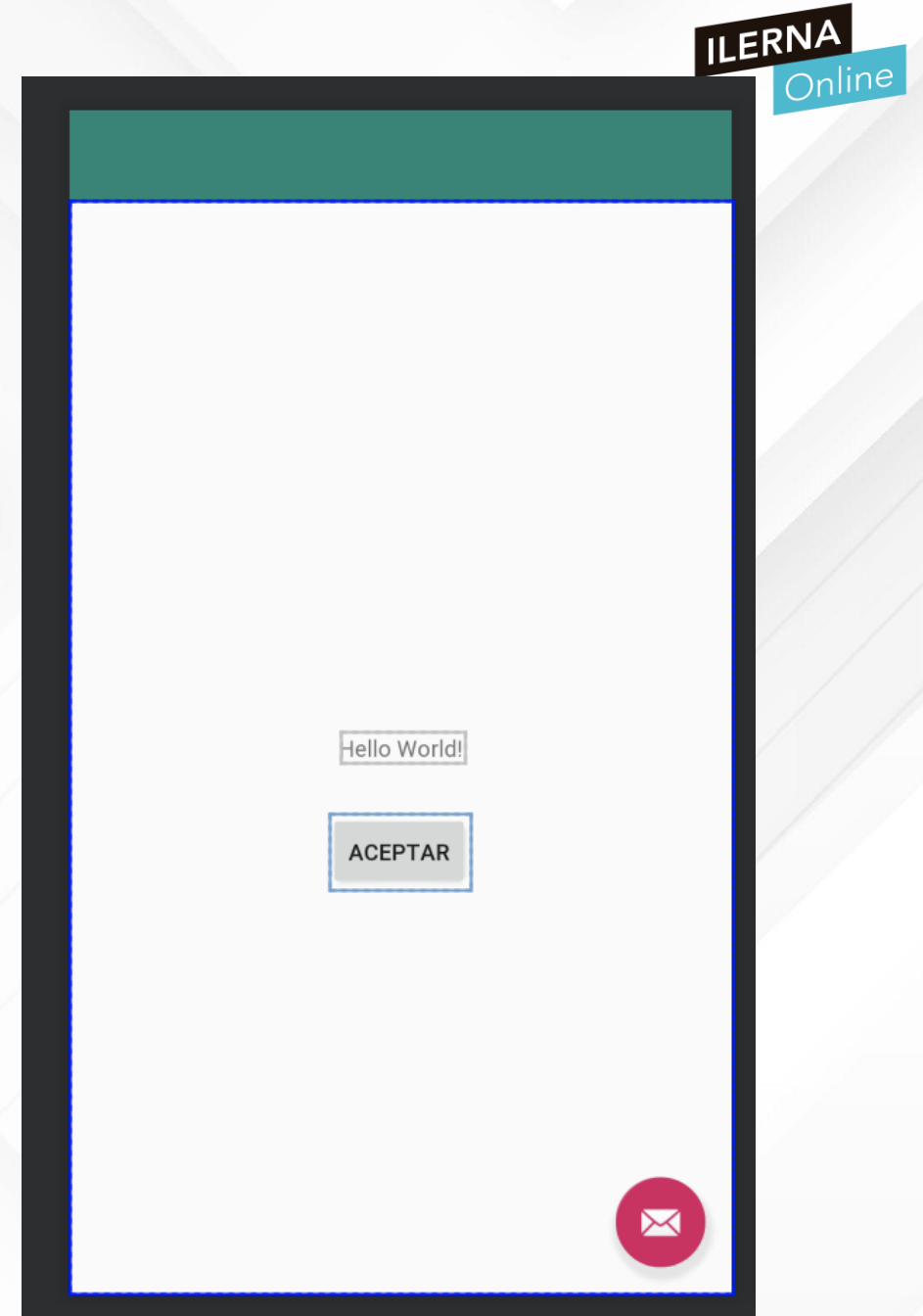
## > CONTROLES DE ENTRADA

Los Controles de Entrada son los componentes visuales mediante los cuales un usuario puede aportar información a la aplicación (texto, números, casillas o eventos de pulsación en pantalla)


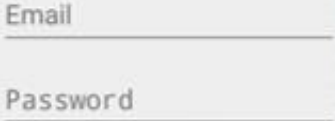


En Android reciben el nombre de Widget


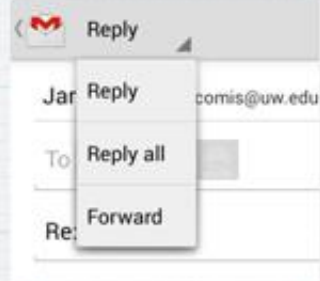
En el ejemplo, podemos encontrar dos objetos de tipo widget:

- Tipo Texto
- Tipo Botón





Tipo de Control	Descripción	Clase	Imagen
Botón	Elemento que puede ser pulsado por el usuario para realizar una acción	Button	
Texto	Texto no editable (TextView) Texto editable (EditText), permite filtrar tipo, longitud del valor que se introduce	TextView, EditText, AutoCompleteText, View	
Casilla de Verificación	Campo que puede ser encendido o apagado por el usuario. Se muestra como opciones no mutuamente excluyentes entre sí	CheckBox	
Botón de Opción	Igual que CheckBox pero redondos y las opciones sí son mutuamente excluyentes entre sí	RadioButton, RadioGroup	

Tipo de Control	Descripción	Clase	Imagen
Switches	Opción de tipo bool, con valores ON/OFF, Encendido / Apagado	ToggleButton	
Spinners	También llamados "Combo Boxes" o listas desplegables. Permiten seleccionar un valor entre varios	Spinner	
Selectores	Diálogos de usuario que permiten seleccionar un valor (fecha/hora) mediante flechas o gestos con la mano	DatePicker. TimePicker	
Imágenes	Comportamiento similar al botón. Permite ejecutar una acción al pulsar la imagen	ImageView, ImageButt on	



- Tipos de layouts: Layout Vertical y Layout Horizontal
- Hasta ahora estamos trabajando con versiones de layouts que disponen sus elementos en sentido vertical, pero ¿qué ocurre cuando giramos nuestro dispositivo móvil?
  - En función de los layouts que hayamos definido en nuestra Activity, los elementos se ajustarán al nuevo espacio en el que están representados
- ¿Puedo configurar este comportamiento?



## > TIPOS DE LAYOUTS

- Para empezar, diseña en un nuevo proyecto una Activity similar a la que se propone en esta imagen
- ¿Qué elementos forman esta Activity?
- ¿Cómo se han dispuesto en ella?

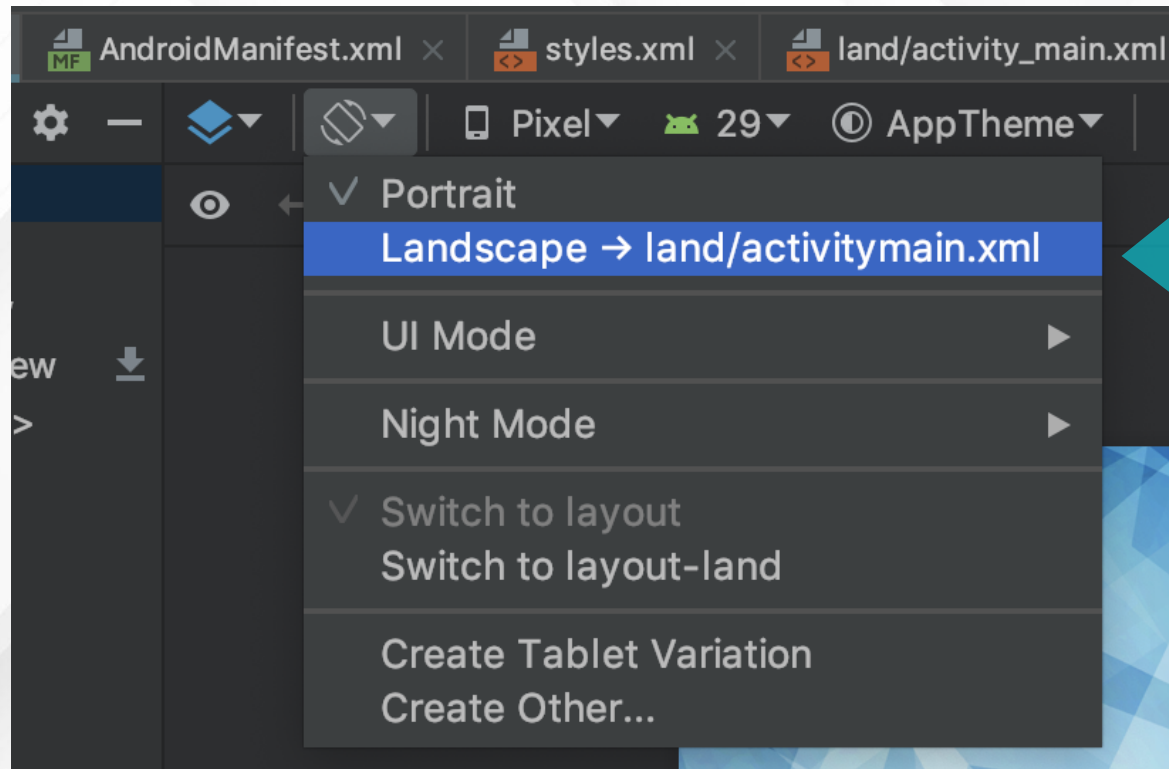


## > TIPOS DE LAYOUTS

The screenshot displays the Android Studio IDE. The main canvas shows a vertical layout titled "Vista Vertical" with a blue geometric background and four buttons labeled "BOTÓN A", "BOTÓN B", "BOTÓN C", and "BOTÓN D" stacked vertically. A large teal arrow points to the eye icon in the top toolbar, which is the trigger for the context menu. The context menu is open, showing options for switching the layout orientation. The "Landscape → land/activitymain.xml" option is highlighted in blue. Other options include "Portrait", "UI Mode", "Night Mode", "Switch to layout", "Switch to layout-land", "Create Tablet Variation", and "Create Other...". The Component Tree on the left shows a hierarchy starting with FrameLayout, containing an ImageView, a TextView, and a TableLayout with four TableRow elements, each containing a button.



- En el menú que se despliega, pulsa la opción Landscape (horizontal)  
Otro layout para la vista apaisada.





# INTERNACIONALIZACIÓN

- En la actualidad, el número de apps que hay en la tienda Google Play Store, es realmente amplio. En concreto, y según datos de AppBrain, actualmente existen más de 2.879.000 aplicaciones.
- El mercado de los videojuegos lidera este número
- Versiones de una aplicación en varios idiomas, con un solo desarrollo
  - Un solo proyecto
  - Ahorro de costes
  - Fácilmente actualizable

- En la actualidad, el número de apps que hay en la tienda Google Play Store, es realmente amplio. En concreto, y según datos de AppBrain, actualmente existen más de 2.879.000 aplicaciones.
- Podemos diferenciar entre los juegos y resto de aplicaciones, las que no son puro entretenimiento.
  - En torno a 383.000 juegos en Google
  - El 15,34 % de las apps de Google Play son juegos
- ¿Es rentable la industria del videojuego en Android?

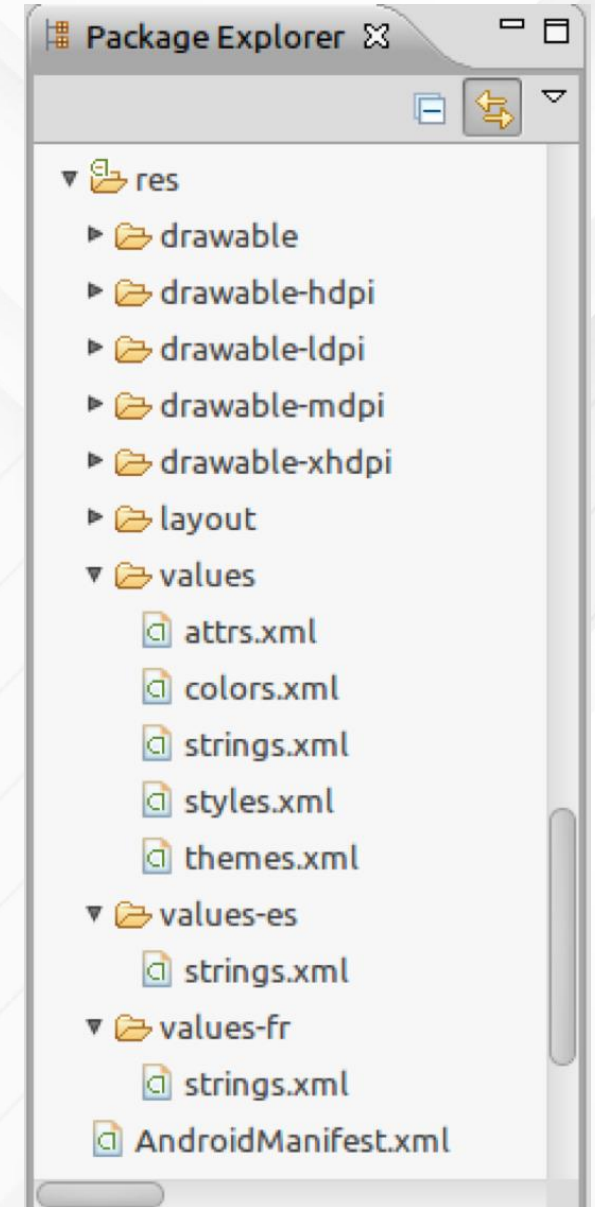


- Crear aplicaciones o juegos en varios idiomas, nuevas oportunidades de negocio
- ¿Distintas aplicaciones para distintos idiomas?
- Ventajas de Internacionalización
  - Desarrollo un único proyecto
  - Separar los textos de la programación de código
  - Ahorro de costes
  - Fácil mantenimiento y actualización
  - Casi sin incremento de tamaño del proyecto

## > INTERNACIONALIZACIÓN

- Para tener textos estén disponibles en varios idiomas, tendremos que añadir una carpeta diferente para cada copia del fichero strings.xml
- Para que nuestra aplicación se muestre en inglés y francés, además de la versión en castellano deberemos crear dos directorios más en la ruta res/
  - res/values => (valores por defecto). **El lenguaje que tenga configurado el dispositivo por móvil por defecto.**
  - res/values-en
  - res/values-fr
- ¿Por qué esos valores?

[https://es.wikipedia.org/wiki/ISO\\_639-1](https://es.wikipedia.org/wiki/ISO_639-1)



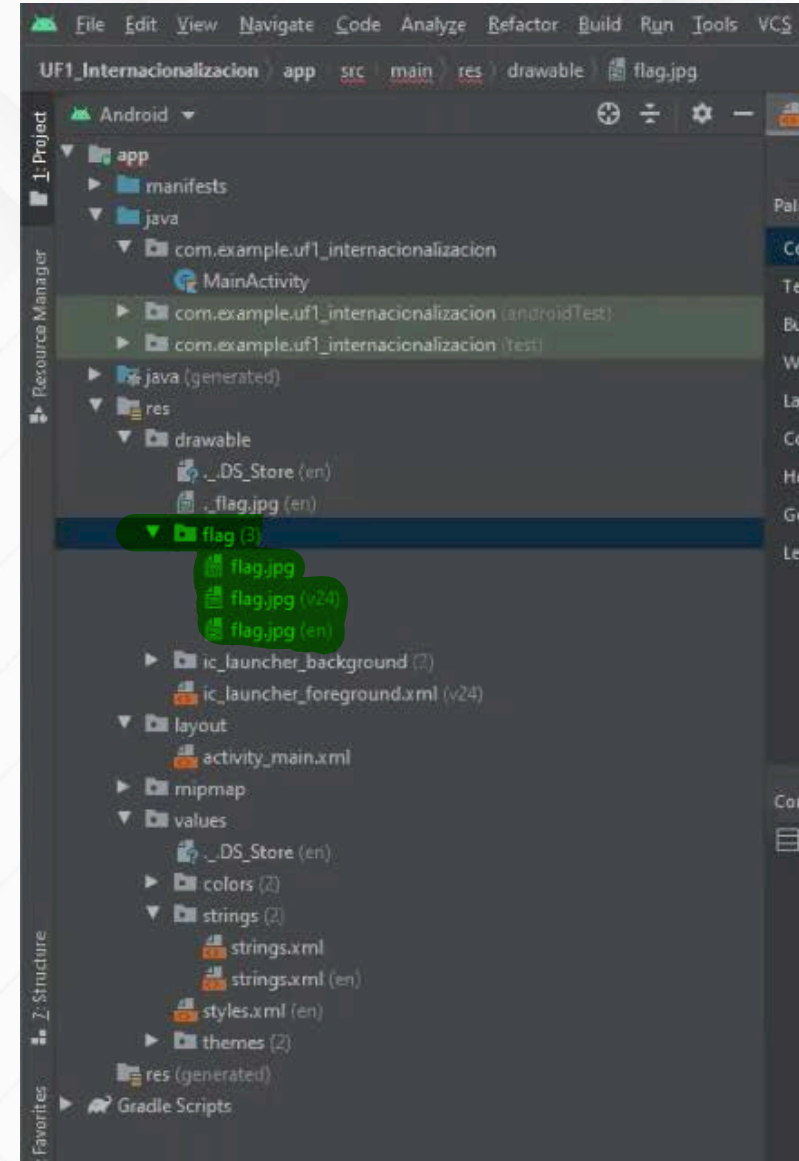


- Podemos indicar imágenes que estén disponibles sólo para un idioma y de diferente calidad según la resolución de la pantalla o la potencia del dispositivo.
- Para ello usaremos los prefijos de idiomas en los directorios habilitados para los recursos de imágenes
- Esto se consigue creando una carpeta res/drawable-en-xhdpi, o cualquier otra densidad de pantalla que necesitemos

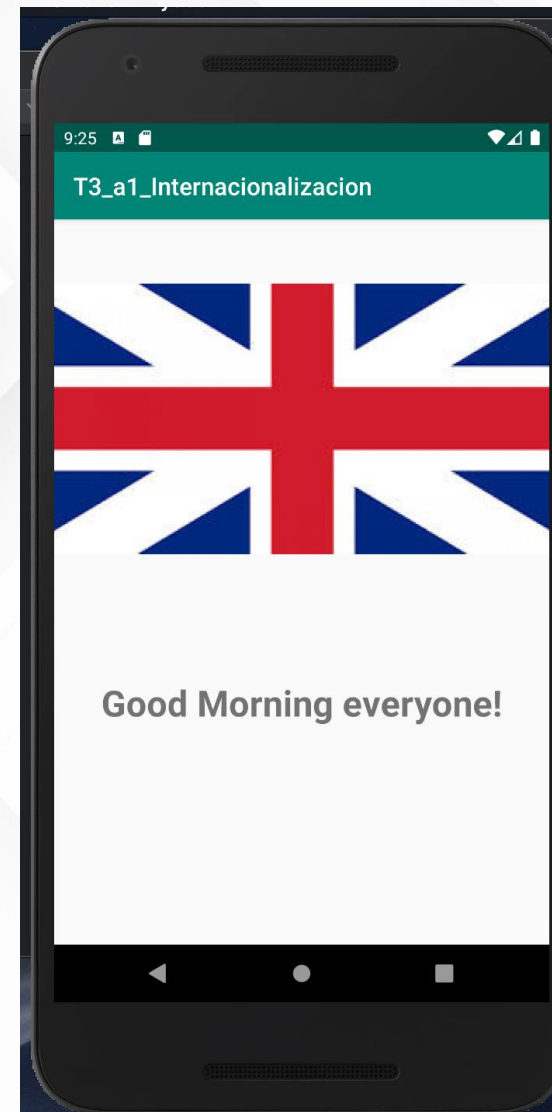
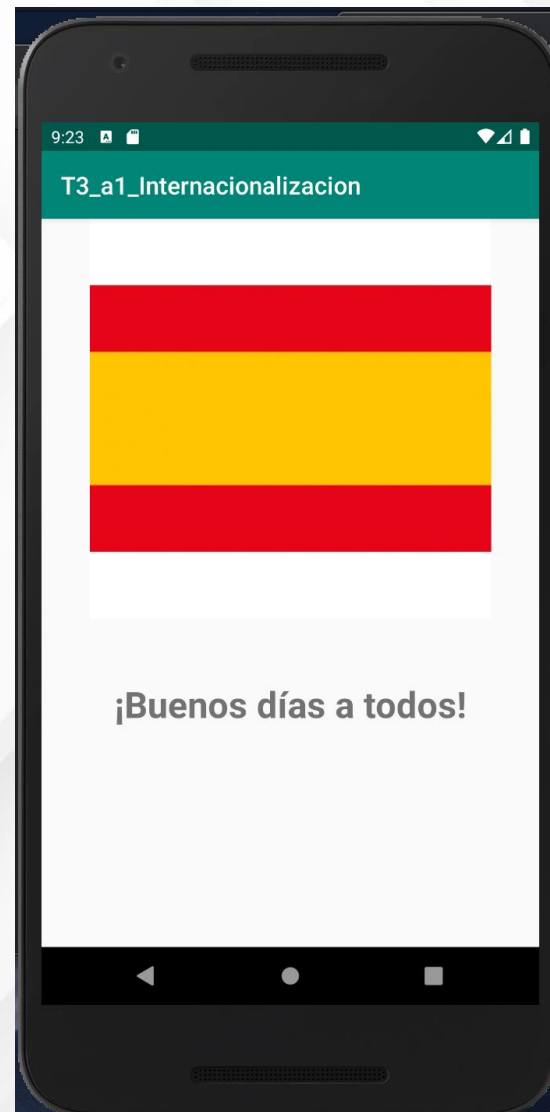
- ✓ drawable-ar-hdpi
- ✓ drawable-ar-mdpi
- ✓ drawable-ar-xhdpi
- ✓ drawable-ar-xxhdpi
- ✓ drawable-hdpi
- ✓ drawable-mdpi
- ✓ drawable-xhdpi
- ✓ drawable-xxhdpi
- ✓ values
- ✓ values-ar

## > INTERNACIONALIZACIÓN

- RECUERDA
- Solamente se cambia el valor de las variables, los nombre de las variables deben de seguir sin modificación
- Debes cambiar el idioma del teléfono o dispositivo donde ejecutes la aplicación al idioma que quieras visualizar para ver los cambios



- EJEMPLO INTERNACIONALIZACIÓN





# LIBRERÍAS MULTIMEDIA INTEGRADAS

- LIBRERÍAS MULTIMEDIAS

- Las aplicaciones multimedia son unas de las más utilizadas y descargadas de los stores
- Utilizamos sistemas operativos móviles, ya sean tablets o smart TV, para consumir películas y series desde alguna plataforma de streaming.
- Desarrollar aplicaciones de audio y vídeo, apps de redes sociales o de mensajería, que proporcionan videollamadas, broadcasts de vídeo, reproducción de mensajes de audio, etc

- LIBRERÍAS MULTIMEDIAS

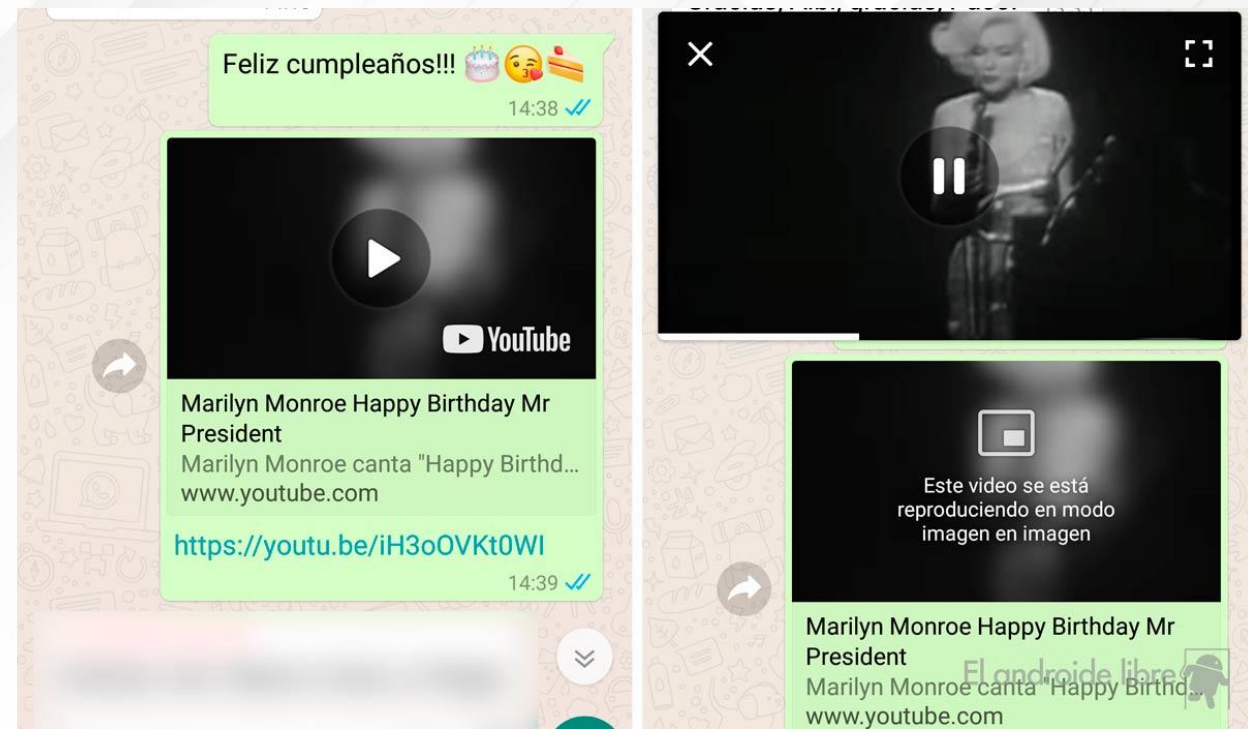
- En el desarrollo de apps multimedia para dispositivos móviles debemos tener en cuenta varios factores, :

- Resoluciones y tamaños de pantalla
- Tipos de procesadores
- Capacidades de memoria
- Códecs soportados, etcétera.
- Uso de periféricos, como teclados, joysticks, auriculares Bluetooth o con cable, micros y cámaras



- LIBRERÍAS MULTIMEDIAS
  - Los sistemas operativos móviles tratan de estandarizar, mediante las librerías multimedia, sus clases e interfaces, todas las diferentes funcionalidades del hardware, ocultando además las posibles diferencias entre terminales.
    - El programador podrá usarlas, en lugar de codificar mecanismos para trabajar con contenido multimedia
    - El avance tecnológico hace que estas clases o librerías sean actualizadas con frecuencia o incluso queden deprecadas

- ARQUITECTURA DEL API
  - Diferenciar aplicaciones que reproducen audio y las que reproducen vídeo
  - Mientras usuario escucha música desde una app, puede estar trabajando con un editor de textos o chateando por una app de mensajería
  - Se tiene el control de la reproducción y se puede poner en primer plano cuando lo necesitemos



- ARQUITECTURA APP AUDIO
  - Clase MediaPlayer
  - Clase que se usa para controlar archivos de vídeo y/o audio en nuestra aplicación
  - Permite el acceso al servicio de media player (reproductor multimedia)
  - Se inicia desde el método onCreate() y llamando a la función create() de dicha clase

## FUNCIONES

## DESCRIPCIÓN

`public void setDataSource(String path)`

Define la ruta del recurso que se usará

`public void prepare()`

Prepara el reproductor de forma sincronizada

`public void start()`

Comienza o continúa la reproducción

`public void stop()`

Detiene la reproducción

`public void pause()`

Pausa la reproducción

`public boolean isPlaying()`

Comprueba si se está reproduciendo

`public void seekTo(int millis)`

Avanza una determinada posición

`public void setLooping(boolean looping)`

Activa o desactiva la reproducción continua

`public boolean isLooping()`

Comprueba si está activada la reprod. Continua

`public void selectTrack(int index)`

Elige una determinada pista

`public int getCurrentPosition()`

Devuelve la posición actual de reproducción

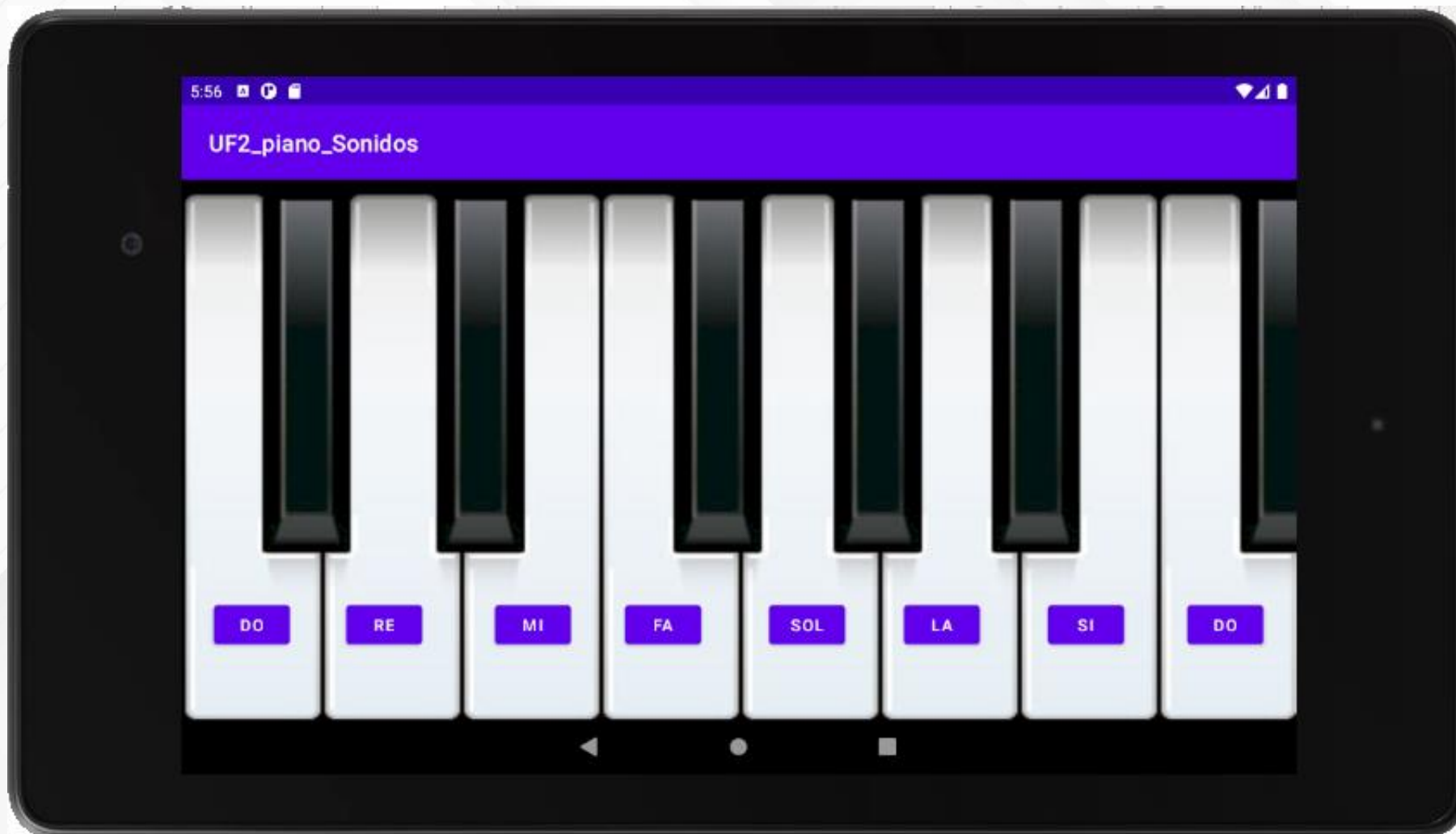
`public int getDuration()`

Devuelve la duración del archivo

`public void setVolume(float leftVolume, float rightVolume)`

Ajusta el volumen de reproducción

- EJEMPLO DE REPRODUCCIÓN MEDIAPLAYER

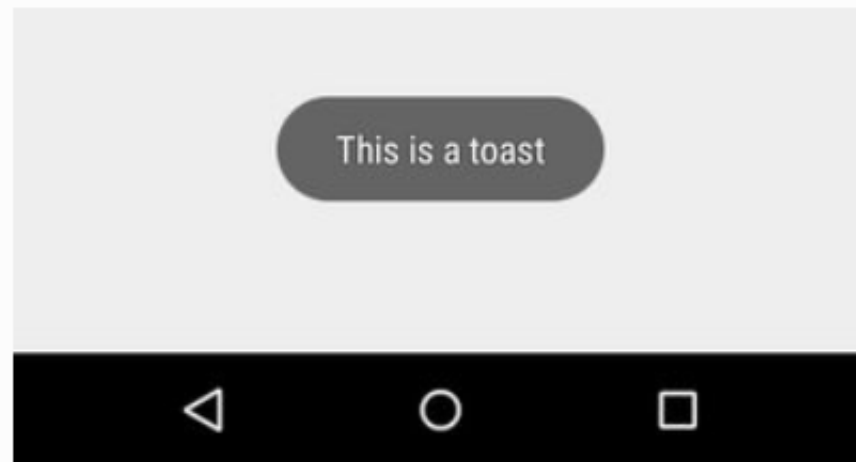




# CUADROS DE DIÁLOGO ALERT DIALOG



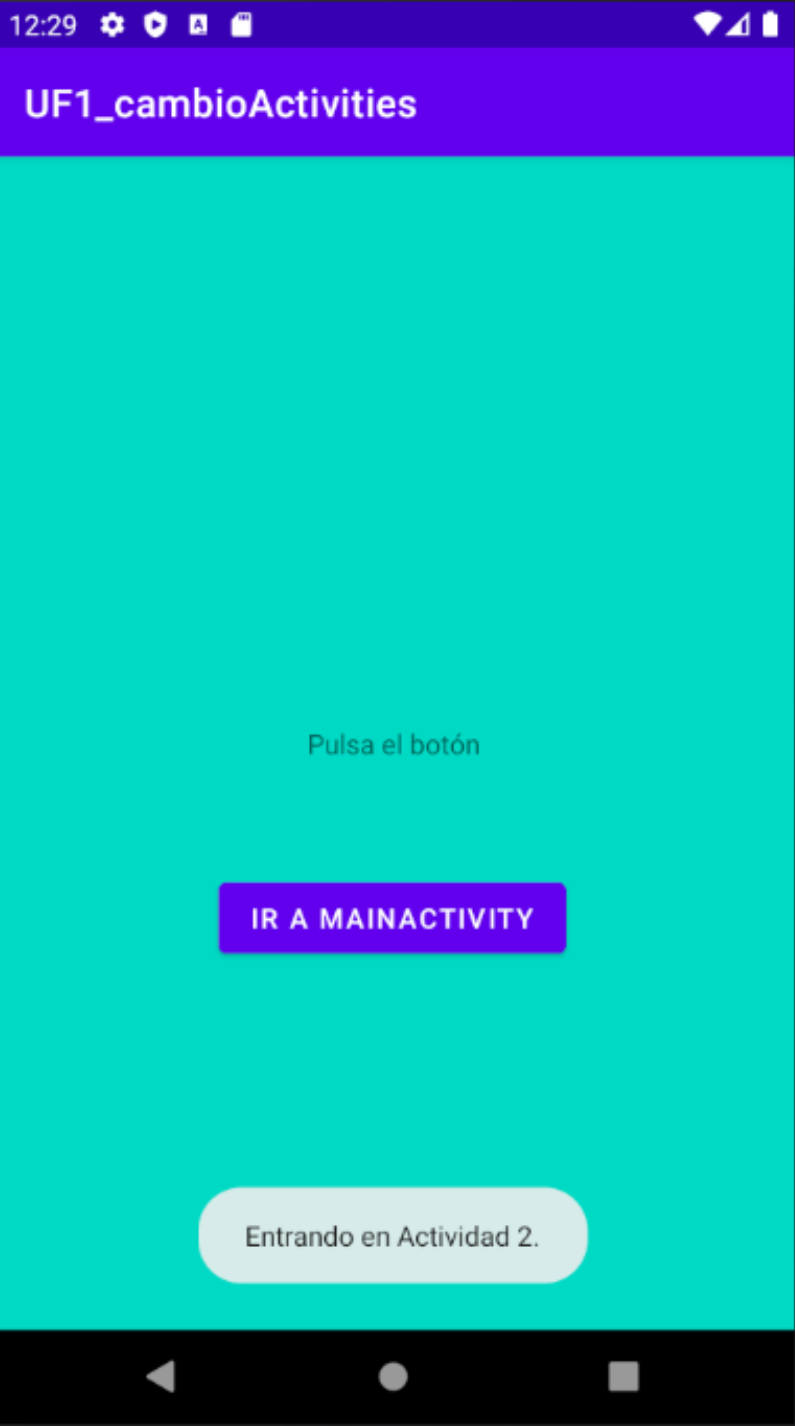
- ANDROID TOAST
- Widget de interfaz de usuario que aparece en pantalla durante algún tiempo en forma de notificación. Por lo general, se usan cuando desea informar al usuario de algo procedimiento no crítico, solo con base de notificación de un suceso.



- ANDROID TOAST
- Los argumentos que requiere la función Toast
  - Context: contexto
  - Message: mensaje
  - Duration: bandera de duración

```
1Toast.makeText(this, "Notificación corta", Toast.LENGTH_SHORT).show()
```

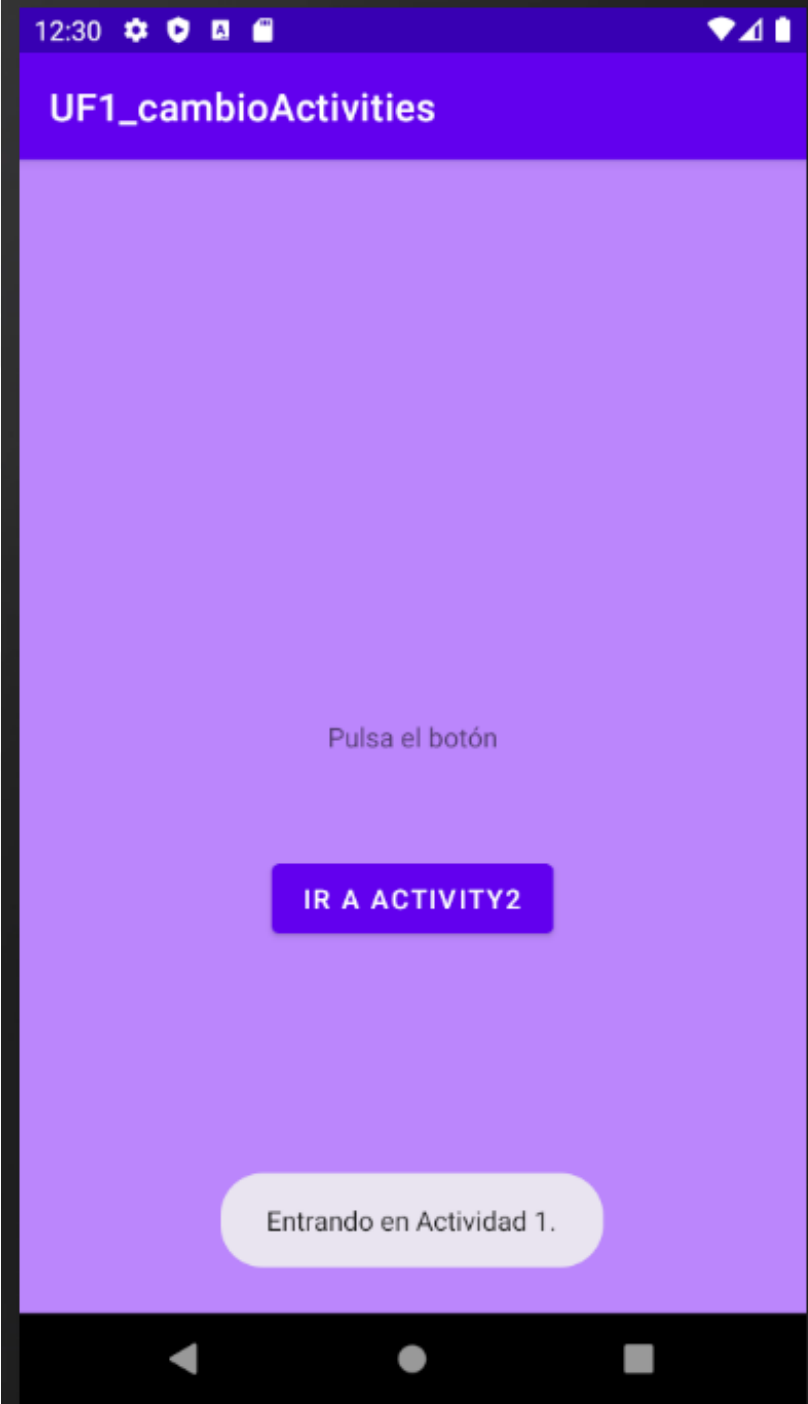
```
2Toast.makeText(this, "Notificación larga", Toast.LENGTH_LONG).show()
```



- ANI
- Los

toast

otificaci  
show()  
otificaci  
now()



- ANDROID TOAST
- FUNCIONES DE EXTENSIÓN DE ANDROID TOAST EN KOTLIN
  - Usando las funciones de Extensión podemos acortar nuestra invocación de Toast estableciendo algunos valores predeterminados en la extensión

```
fun Context?.toast(text: CharSequence, duration: Int = Toast.LENGTH_LONG) =  
this?.let { Toast.makeText(it, text, duration).show() }
```

```
fun Context?.toast(@StringRes textId: Int, duration: Int = Toast.LENGTH_LONG) = this?.let {  
Toast.makeText(it, textId, duration).show() }
```

- ANDROID TOAST
- FUNCIONES DE EXTENSIÓN DE ANDROID TOAST EN KOTLIN

- Podemos usarlo dentro de un activity

```
toast("Esto es un mensaje")  
toast(R.string.recurso_string)
```

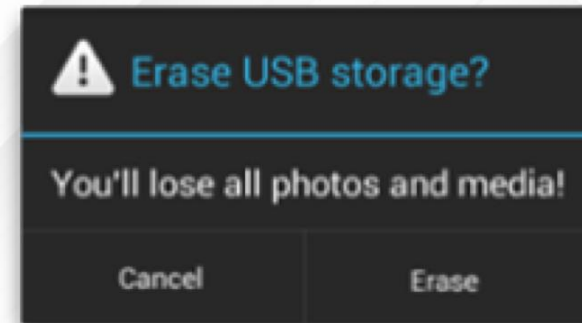
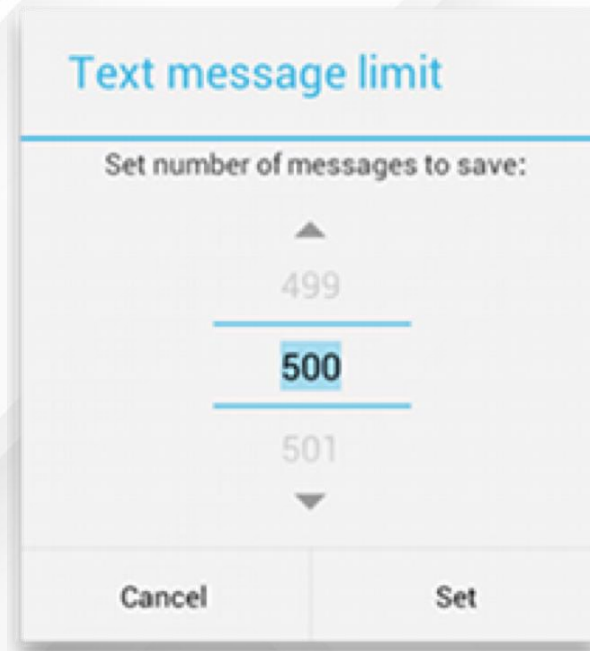
- Dentro de un fragment

```
context?.toast("Esto es un mensaje")  
context?.toast(R.string.recurso_string)
```

- Cambiar la duración de tiempo en que se muestra por pantalla

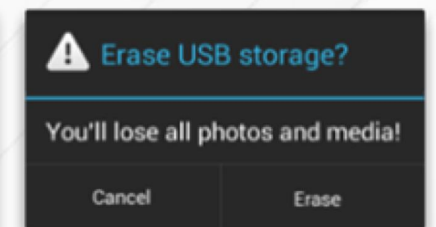
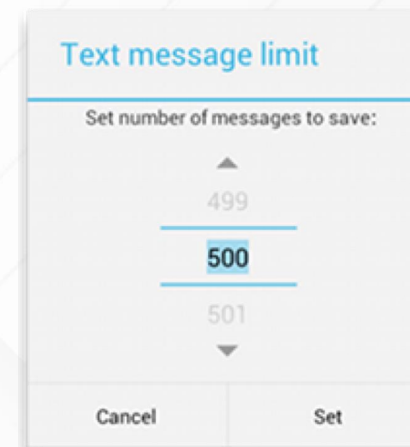
```
toast("Esto es un mensaje",  
Toast.LENGTH_SHORT)
```

- Ventana pequeña que le pide al usuario que haga una determinada acción o que tome una decisión

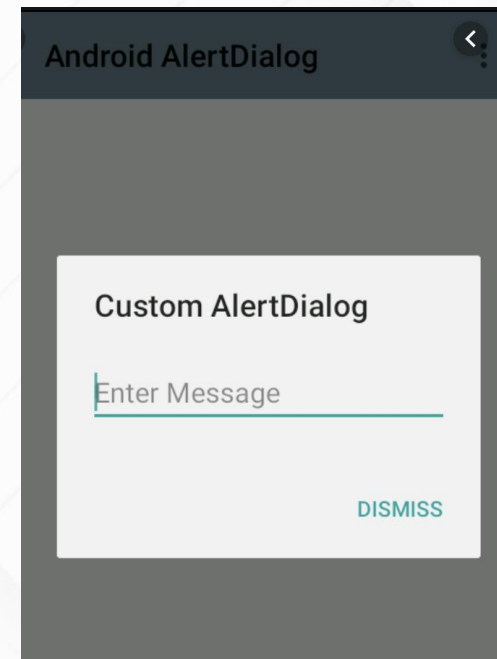
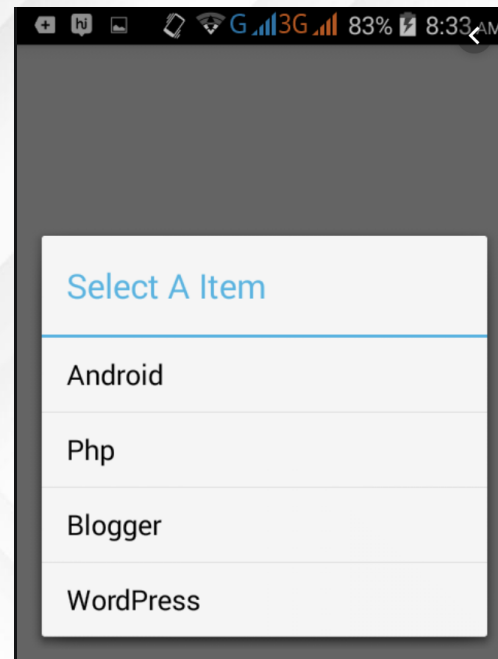
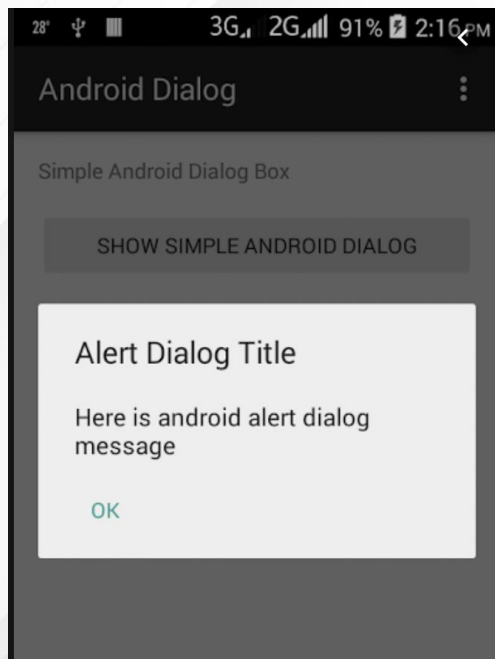




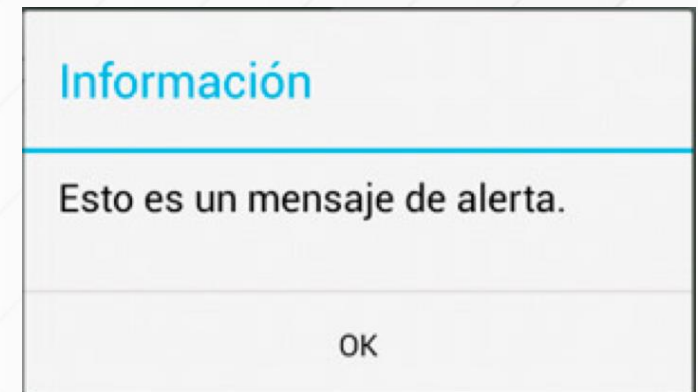
- CLASE DIALOG
  - Clase de base que podemos usar para los diálogos
  - **IMPORTANTE:** Debes evitar crear instancias de Dialog de forma directa
  - En su lugar, se recomienda usar una de las siguientes subclases:
    - AlertDialog
    - DatePickerDialog o TimePickerDialog



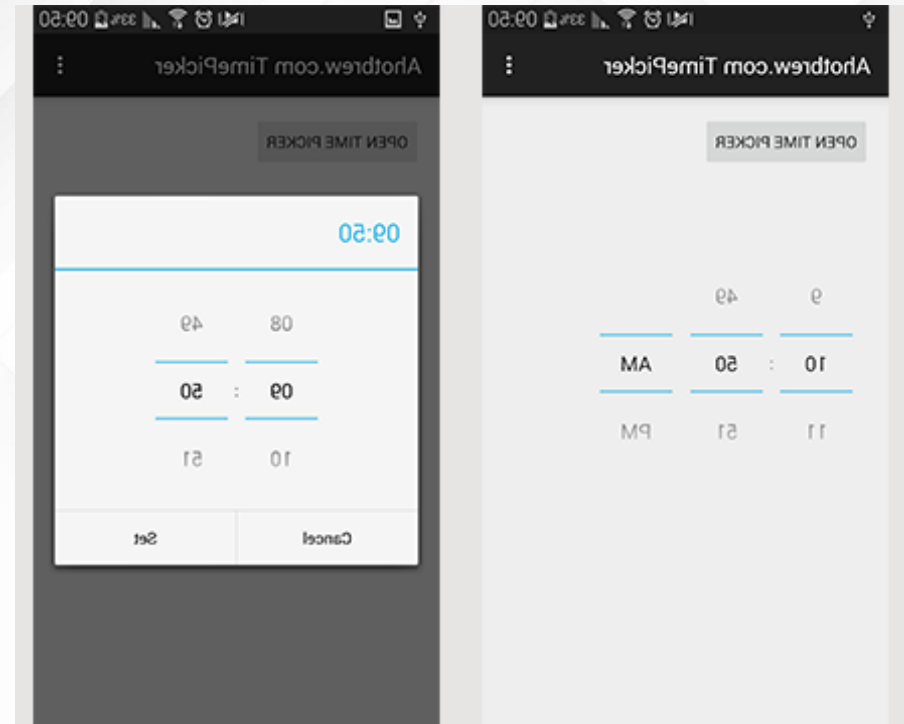
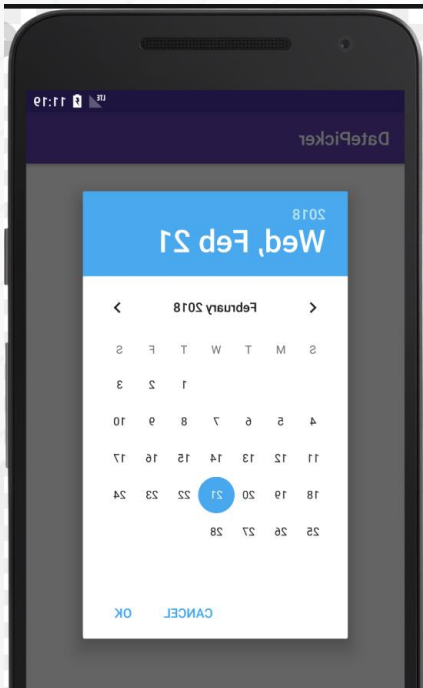
- Clase Dialog
  - Subclase AlertDialog
  - Diálogo que puede mostrar un título, hasta tres botones, una lista de elementos seleccionables o un diseño personalizado



- Clase Dialog
  - Subclase AlertDialog
  - Este tipo de diálogo se limita a mostrar un mensaje sencillo al usuario, y un único botón de OK para confirmar su lectura.
- Lo construiremos mediante la clase AlertDialog, y más concretamente su subclase AlertDialog.Builder
  - título [setTitle()]
  - mensaje [setMessage()]
  - Texto y botón [setPositiveButton()]



- Clase Dialog
  - Subclase DatePickerDialog o TimePickerDialog (I)
    - Diálogo con una IU predefinida que le permite al usuario seleccionar una fecha o una hora



- Clase Dialog
  - Otros Fragmentos de Diálogo
    - Diálogo de Selección: Cuando podemos seleccionar valores de una lista de opciones y además, podemos elegir más de una de ellas (array tradicional)
- Diálogos Personalizados: definiendo un layout XML con los elementos que queremos incluir

Selección	
Español	
Inglés	
Francés	

Selección	
Español	<input checked="" type="checkbox"/>
Inglés	<input checked="" type="checkbox"/>
Francés	<input type="checkbox"/>

## > EJEMPLO ALERTDIALOG

```
val builder = AlertDialog.Builder(context: this)
builder.setTitle("Ejemplo AlertDialog")
builder.setMessage("Esto es un ejemplo de uso de AlertDialog en Android")
//builder.setPositiveButton("OK", DialogInterface.OnClickListener { function = x })

builder.setPositiveButton(android.R.string.ok) { dialog, which ->
    Toast.makeText(applicationContext,
        android.R.string.ok, Toast.LENGTH_SHORT).show()
    //Aquí se define la acción al pulsar SÍ
}

builder.setNegativeButton(android.R.string.cancel) { dialog, which ->
    Toast.makeText(applicationContext,
        android.R.string.cancel, Toast.LENGTH_SHORT).show()

    //Aquí se define la acción al pulsar NO
}

builder.setNeutralButton(text: "Omitir") { dialog, which ->
    Toast.makeText(applicationContext,
        text: "Has pulsado Omitir", Toast.LENGTH_SHORT).show()

    //Aquí se define la acción al pulsar OMITIR
}
builder.show()
```

UF2\_AlertDialog

M08 DAM PMDM

USO ALERTDIALOG

ABRIR ALERTDIALOG

Ejemplo AlertDialog

Esto es un ejemplo de uso de AlertDialog en Android

OMITIR

CANCELAR ACEPTAR





# SharedPreferences

---

Con las SharedPreferences es posible almacenar datos en pares de clave y valor, es decir, podemos definir una clave univoca a la cual asociamos el dato que queremos almacenar.

Estos pares de datos (clave y valor) son almacenados en ficheros que luego pueden permanecer privados y únicamente los podrá usar la aplicación que los ha creado.

`Context.MODE_PRIVATE` → Únicamente puede hacer uso la app.

`Context.MODE_WORLD_READABLE` → Pueden leer el resto de aplicaciones.

`Context.MODE_WORLD_WRITEABLE` → Pueden escribir el resto de aplicaciones.

## > Obtener Preferencias

- Obtenemos una referencia de un objeto de la clase SharedPreferences a través del método `getSharedPreferences` heredado de la clase `AppCompatActivity`.
- El primer parámetro es el nombre del archivo de preferencias y el segundo la forma de creación del archivo (`MODE_PRIVATE` indica que solo esta aplicación puede consultar el archivo XML que se crea)

```
val preferencias = getSharedPreferences("datos", Context.MODE_PRIVATE)
```

## > Obtener Preferencias

- Para extraer los datos del archivo de preferencias debemos indicar el nombre a extraer y un valor de retorno si dicho nombre no existe en el archivo de preferencias.
- Si no existe, se crea:

```
et1.setText(preferencias.getString("mail", ""))
```

- Al pulsar el botón, tendremos que guardar los datos en el archivo de preferencias usando la variable mail

```
boton1.setOnClickListener {  
    val editor = preferencias.edit()  
    editor.putString("mail", et1.text.toString())  
    editor.commit()  
    finish()  
}
```

- Usaremos un objeto de la clase Editor y la referencia del objeto de la clase SharedPreferences que acabamos de crear. Con el método putString guardamos en mail el valor del String que se ha insertado en el EditText.
- Luego debemos llamar al método commit de la clase Editor para que el dato quede guardado en el archivo de preferencias.
- Al iniciar la aplicación se cargará el último valor que hayamos registrado en el fichero.

> Obtener Preferencias

UF2\_SharedPreferences



**SUSCRÍBETE**  
a nuestra Newsletter



Recibirás todas nuestras ofertas y las últimas noticias en moda. ¡NO TE LO PIERDAS!

[hola@hola.com](mailto:hola@hola.com)

**CONFIRMAR**





## > Obtener Preferencias

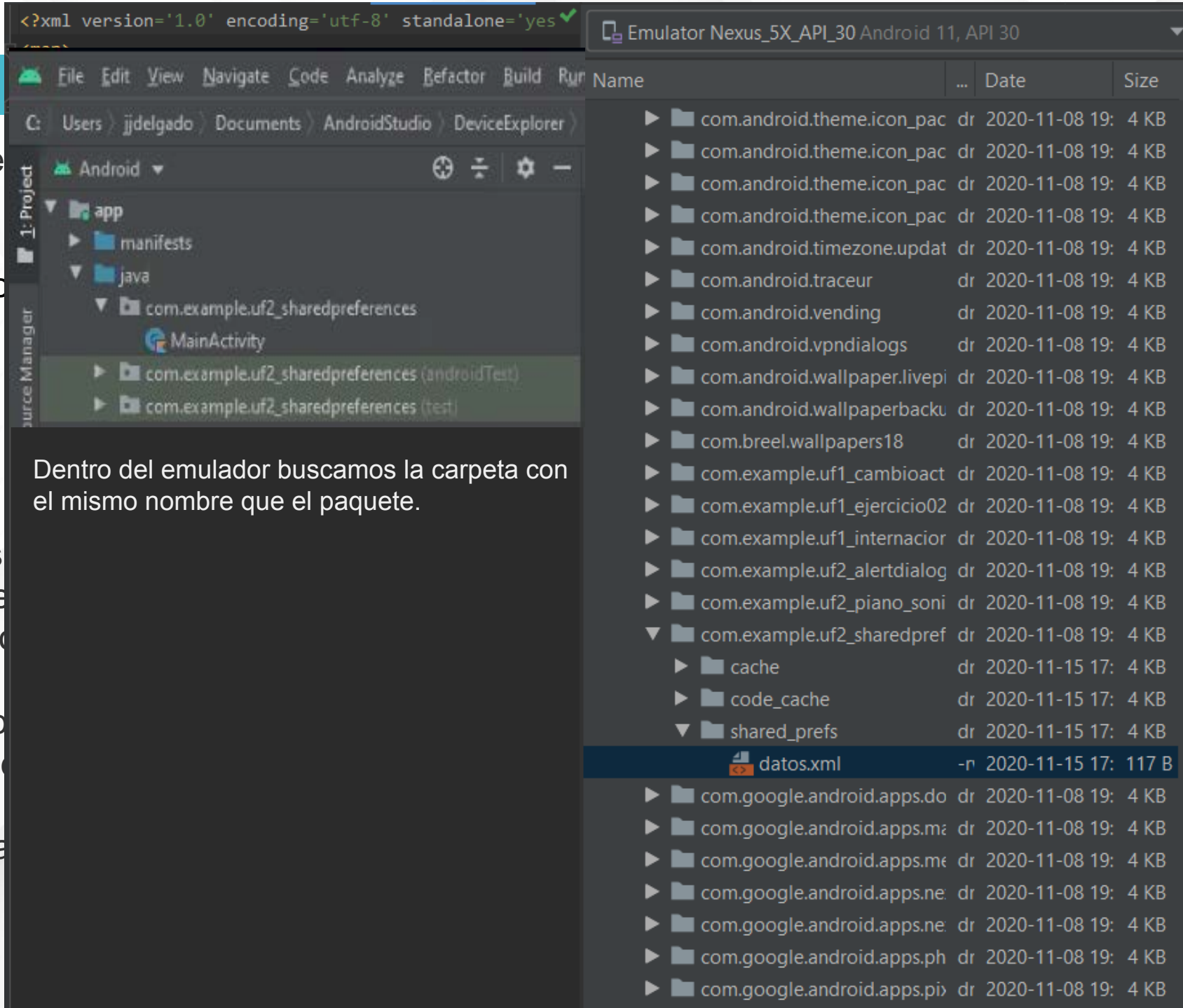
- Al pulsar e usando la

bo

- Usaremos SharedPre mail el valo

- Luego deb guardado

- Al iniciar la fichero.



The screenshot shows the Android Studio interface. The top bar indicates the emulator is running on a Nexus 5X with API level 30. The file explorer on the left shows the project structure with the following folders and files:

- 1: Project
  - app
    - manifests
    - java
      - com.example.uf2\_sharedpreferences
        - MainActivity
      - com.exmple.uf2\_sharedpreferences (androidTest)
      - com.example.uf2\_sharedpreferences (test)

Dentro del emulador buscamos la carpeta con el mismo nombre que el paquete.

Name	...	Date	Size
▶ com.android.theme.icon_pac	dr	2020-11-08 19:	4 KB
▶ com.android.theme.icon_pac	dr	2020-11-08 19:	4 KB
▶ com.android.theme.icon_pac	dr	2020-11-08 19:	4 KB
▶ com.android.theme.icon_pac	dr	2020-11-08 19:	4 KB
▶ com.android.timezone.updat	dr	2020-11-08 19:	4 KB
▶ com.android.traceur	dr	2020-11-08 19:	4 KB
▶ com.android.vending	dr	2020-11-08 19:	4 KB
▶ com.android.vpndialogs	dr	2020-11-08 19:	4 KB
▶ com.android.wallpaper.livepi	dr	2020-11-08 19:	4 KB
▶ com.android.wallpaperbacku	dr	2020-11-08 19:	4 KB
▶ com.breel.wallpapers18	dr	2020-11-08 19:	4 KB
▶ com.example.uf1_cambioact	dr	2020-11-08 19:	4 KB
▶ com.example.uf1_ejercicio02	dr	2020-11-08 19:	4 KB
▶ com.example.uf1_internacior	dr	2020-11-08 19:	4 KB
▶ com.example.uf2_alerdialog	dr	2020-11-08 19:	4 KB
▶ com.example.uf2_piano_soni	dr	2020-11-08 19:	4 KB
▼ com.example.uf2_sharedpref	dr	2020-11-08 19:	4 KB
▶ cache	dr	2020-11-15 17:	4 KB
▶ code_cache	dr	2020-11-15 17:	4 KB
▼ shared_prefs	dr	2020-11-15 17:	4 KB
📄 datos.xml	-r	2020-11-15 17:	117 B
▶ com.google.android.apps.do	dr	2020-11-08 19:	4 KB
▶ com.google.android.apps.m	dr	2020-11-08 19:	4 KB
▶ com.google.android.apps.m	dr	2020-11-08 19:	4 KB
▶ com.google.android.apps.ne	dr	2020-11-08 19:	4 KB
▶ com.google.android.apps.ne	dr	2020-11-08 19:	4 KB
▶ com.google.android.apps.ph	dr	2020-11-08 19:	4 KB
▶ com.google.android.apps.pi	dr	2020-11-08 19:	4 KB

# **TRABAJANDO CON CLASES MULTIMEDIA**

---

## > REPRODUCTOR MULTIMEDIA

- Crea un `Proyecto en Android Studio que incluya un reproductor multimedia, capaz de reproducir un archivo MP3 y que incluya botones para:
  - Arrancar la reproducción
  - Pausar la reproducción
  - Reanudar la reproducción
  - Detener la reproducción
  - Activar la reproducción en bucle



## > REPRODUCTOR MULTIMEDIA

- Crea un `Proyecto en Android Studio que incluya un reproductor multimedia, capaz de reproducir un archivo MP3 y que incluya botones para:
  - Usaremos ImageButton
  - Combinaremos Layouts
  - Añadir recursos a drawable y raw



**SQLite**

---



- SQLite es un motor de base de datos SQL transaccional de código abierto, ligero, autónomo, de configuración simple y sin servidor, que se caracteriza por almacenar información persistente de forma sencilla.
- Es gratuito tanto para fines privados como para comerciales, se puede descargar de forma libre desde su sitio oficial.
- SQLite cuenta con varios enlaces a lenguajes de programación entre los que podemos destacar: Java, C, C ++, JavaScript, C #, Python, VB Script, entre otros.





- VENTAJAS
- FÁCIL DE CONFIGURAR: usa una clase, reduciendo de forma significativa todos aquellos esfuerzos sobre la administración
- NO NECESITA SOPORTE DE SERVIDOR: almacenamiento local en la aplicación y usa librerías que se encargan de la gestión y por ende no ejecuta procesos para administrar la información
- SOFTWARE LIBRE
- SE ALMACENA EN ARCHIVO: bueno para seguridad y migración
- ALMACENAMIENTO PERSISTENTE DE DATOS



- Android incorpora de serie todas las herramientas necesarias para la creación y gestión de bases de datos SQLite, y entre ellas una completa API para llevar a cabo de manera sencilla todas estas tareas
- La forma más habitual para crear, actualizar, y conectar con una base de datos SQLite será a través de una clase auxiliar llamada SQLiteOpenHelper
- Definiremos una clase propia que derive de ella y tendremos que personalizarla para adaptarnos a las necesidades concretas de nuestra aplicación

- La clase SQLiteOpenHelper incluye:
- Un único constructor, que normalmente no necesitaremos sobrescribir
- Dos métodos abstractos, onCreate() y onUpgrade(), que deberemos personalizar con el código necesario para crear nuestra base de datos y para actualizar su estructura respectivamente

- EJEMPLO: Proyecto Gestión de Stock
- Aplicación que almacena en una BD SQLite la información de los productos almacenados en la tienda
- Uso de EditText
- Uso de Botones para:
- Crear producto, buscar usando el código, buscar por el nombre, eliminar un producto a partir del código y modificar un producto



# Services

---

- Un servicio es simplemente un componente que puede ejecutarse en segundo plano, incluso cuando el usuario no está interactuando con tu aplicación. Por eso, solo debes crear un servicio si es lo que necesitas.
- En cambio, es necesario crear un subproceso si debes trabajar fuera de tu subproceso principal solamente mientras el usuario interactúa con tu aplicación.
- También considera la posibilidad de utilizar AsyncTask o HandlerThread en lugar de la clase Thread tradicional



Es importante que crees un subproceso nuevo, en el que el servicio pueda completar todo el trabajo. El servicio utiliza el subproceso principal de tu aplicación de forma predeterminada, lo que puede ralentizar el rendimiento de cualquier actividad que tu aplicación esté ejecutando.

En caso de un crear un subproceso, si la carga es muy grande el hilo puede quedar bloqueado.

Por ejemplo:

```
Log.d(TAG, "Servicio Iniciado");  
try {  
    Thread.sleep(15000);  
} catch (InterruptedException e) {  
    e.printStackTrace();  
}
```

¿Como crear un servicio?

Crear una nueva clase que extienda de Service (android.app.Service)  
Sobre escribir método onBind, para inicilizarlo.  
Poner funcionalidad en el método

```
override fun onStartCommand(intent: Intent?, flags: Int, startId: Int):  
Int {  
    Toast.makeText(this, "service starting", Toast.LENGTH_SHORT).show()  
    return super.onStartCommand(intent, flags, startId)  
}
```

Sobrescribir el método onDestroy para finalizar correctamente el servicio  
Declarar el service en el AndroidManifest.xml

Para invocar a un Service debe utilizarse la siguiente estructura:

```
Intent(this, HelloService::class.java).also { intent ->  
    startService(intent)  
}
```

## ¿Como crear un servicio?

```
override fun onStartCommand(intent: Intent?, flags: Int, startId: Int):  
Int {  
    Toast.makeText(this, "service starting", Toast.LENGTH_SHORT).show()  
    return super.onStartCommand(intent, flags, startId)  
}
```

De acuerdo con al definición anterior, `onStartCommand()` devuelve un número entero, que describe la forma en la que debe continuar el servicio en caso de que el sistema lo finalice

Valores posibles (constantes):

- `START_NOT_STICKY`: después de haber sido terminado, no se vuelve a crear, salvo que queden cosas por terminar. Así no se duplicará
- `START_STICKY`: adecuado para reproductores multimedia que no ejecutan comandos y se ejecutan de forma indefinida y esperan un trabajo
- `START_REDELIVERY_INTENT`: adecuado para servicios que ejecutan activamente un trabajo y deben reanudarse inmediatamente, como la descarga de archivos.

## Otros métodos

`onBind()`: se llama cuando otro componente quiere enlazarse con el servicio. Si no se desea permitir estos enlaces, hay que implementarlo devolviendo NULL

```
override fun onBind(intent: Intent): IBinder? {  
    // We don't provide binding, so return null  
    return null  
}
```

`onDestroy()`: se llama cuando queremos dejar de usar el servicio. Deberíamos borrar los recursos, subprocesos y todo lo que hayamos usado. Es la última llamada que recibe el servicio

```
override fun onDestroy() {  
    Toast.makeText(this, "service done", Toast.LENGTH_SHORT).show()  
}
```

```
class MyServices : Service() {
    private MediaPlayer player;

    override fun onBind(intent: Intent): IBinder? {
        return null;
    }

    override fun onStartCommand(intent: Intent?, flags: Int, startId: Int) {
        Toast.makeText(this, "service starting", Toast.LENGTH_SHORT).show()
        if (player != null && player.isPlaying())
            player.stop();
        player = MediaPlayer.create(this, R.raw.train);
        player.setLooping(true);
        player.start();

        return super.onStartCommand(intent, flags, startId);
    }

    override fun void onDestroy() {
        Toast.makeText(this, "Servicio finalizado", Toast.LENGTH_SHORT).show()
        if (player != null)
            player.release();
        stopSelf();
    }
}
```

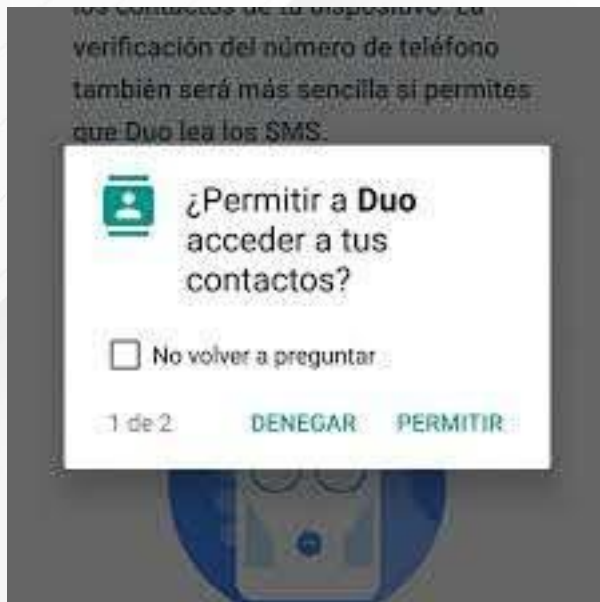
# Permisos en Android

---



- En versiones anteriores a la de Android 6.0
  - En el momento de la instalación, se solicitaba al usuario la aceptación de una lista de permisos como el acceso a memoria externo, a la cámara, a internet, a los contactos, etc
- A partir de Android 6.0 (nivel de API 23), los usuarios conceden permisos a las apps mientras se ejecutan, no cuando instalan la app.
  - Este enfoque simplifica el proceso de instalación de la app, ya que el usuario no necesita conceder permisos cuando instala o actualiza la app.

- Se le otorga al usuario mayor control sobre la funcionalidad de la app; por ejemplo, un usuario podría optar por proporcionar a una app de cámara acceso a esta, pero no a la ubicación del dispositivo.
- Una ventaja es que el usuario puede revocar los permisos en cualquier momento desde la pantalla de configuración de la app.



- Los permisos del sistema se dividen en dos categorías, normal y de riesgo:
  - Los **permisos normales** no ponen en riesgo la privacidad del usuario de forma directa. Si tu app tiene un permiso normal en su Android Manifest, el sistema concede el permiso automáticamente.
  - Los **permisos de riesgo** pueden permitir que la app acceda a información confidencial del usuario. Si tienes un permiso peligroso, el usuario debe autorizar explícitamente a tu app.

- En todas las versiones de Android, las aplicaciones deben declarar los permisos normales y de riesgo que necesitan para ser usadas en el fichero Android Manifest

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.snazzyapp">

    <uses-permission android:name="android.permission.SEND_SMS"/>

    <application ...>
        ...
    </application>

</manifest>
```

- No obstante, el efecto de esa declaración es diferente según la versión del sistema y el nivel de SDK de destino de tu app:
- En dispositivos Android 5.1 o una versión anterior, o el nivel de SDK de destino de la app es el 22 o inferior
  - Si existe algún permiso peligroso en el Android Manifest, el usuario debe concederlo en el momento de la instalación. En caso de no hacerlo, se cancelará la instalación
- En dispositivos Android 6.0 o una versión posterior, o con el nivel de SDK 23 o posterior
  - Los permisos deben estar indicados en el manifiesto de la app, y esta debe solicitar cada permiso riesgoso que necesite mientras la app esté en ejecución.
  - El usuario puede conceder o negar cada permiso y la app puede continuar ejecutándose con capacidades limitadas aun cuando el usuario rechace una solicitud de permiso.



Grupo de Permisos	Permisos	
CALENDAR	READ_CALENDAR WRITE_CALENDAR	
CAMERA	CAMERA	
CONTACTS	READ_CONTACTS WRITE_CONTACTS GET_ACCOUNTS	
LOCATION	ACCESS_FINE_LOCATION ACCESS_COARSE_LOCATION	
MICROPHONE	RECORD_AUDIO	
PHONE	READ_PHONE_STATE CALL_PHONE READ_CALL_LOG WRITE_CALL_LOG	ADD_VOICE_MAIL USE_SIP PROCESS_OUTGOING_CALLS
SENSOR	BODY_SENSORS	
SMS	SEND_SMS RECEIVE_SMS READ_SMS	RECEIVE_WAP_PUSH RECEIVE_MMS
STORAGE	READ_EXTERNAL_STORAGE WRITE_EXTERNAL_STORAGE	



- PERMISOS NORMALES

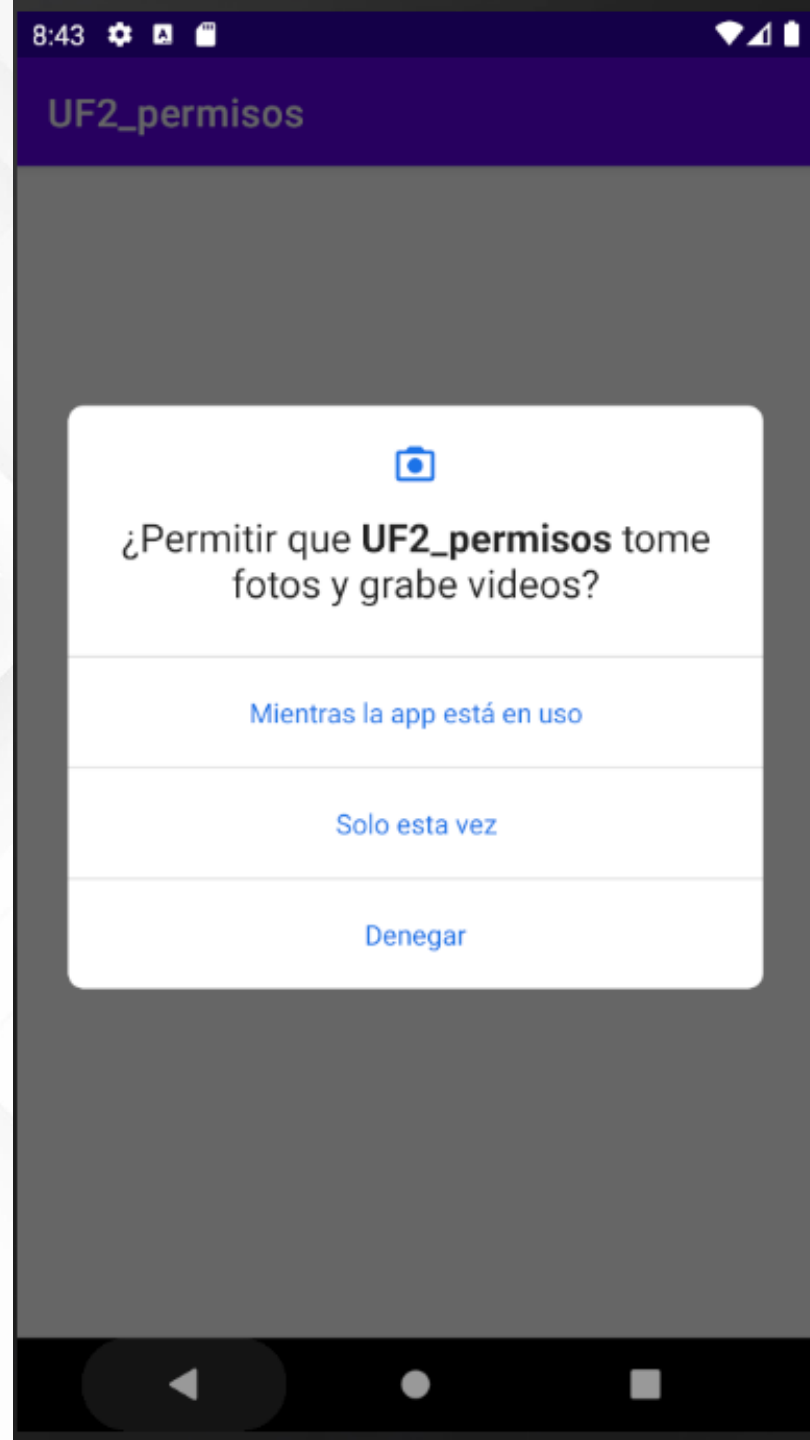
- Añadir el permiso en el fichero AndroidManifest.xml
- `<uses-permission android:name="android.permission.INTERNET"/>`

- PERMISOS DE RIESGO:

- Los dispositivos con API 23 o superior tendrán el permiso desactivado por defecto y si intentamos realizar una acción que los necesite, la aplicación se detendrá
- Tendremos que activarlo

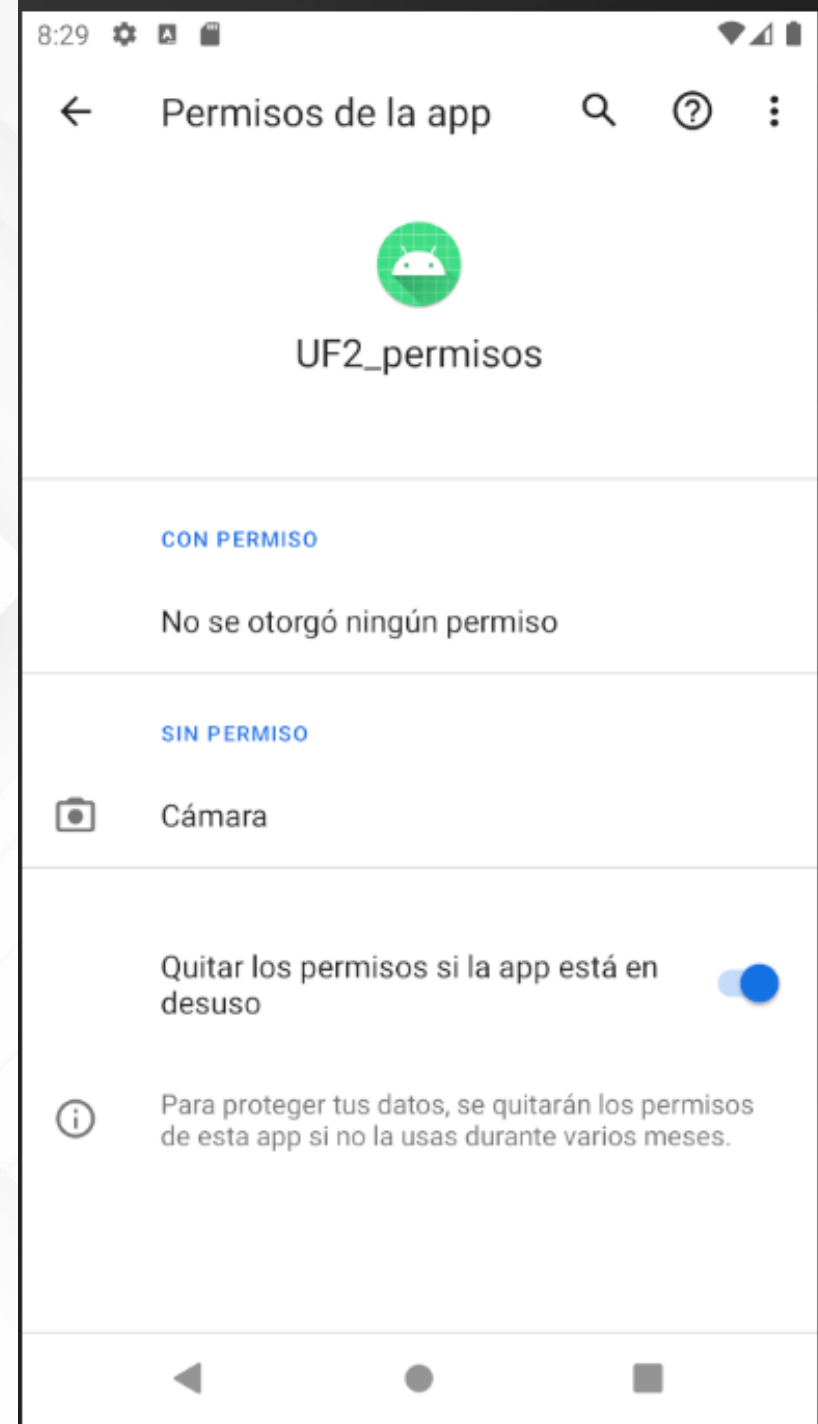
## > PROGRAMANDO UN PERMISO

- En versiones posteriores a la 23, el permiso se solicita al usar el objeto que requiere del permiso
- Pueden ofrecerse varias opciones respecto al permiso



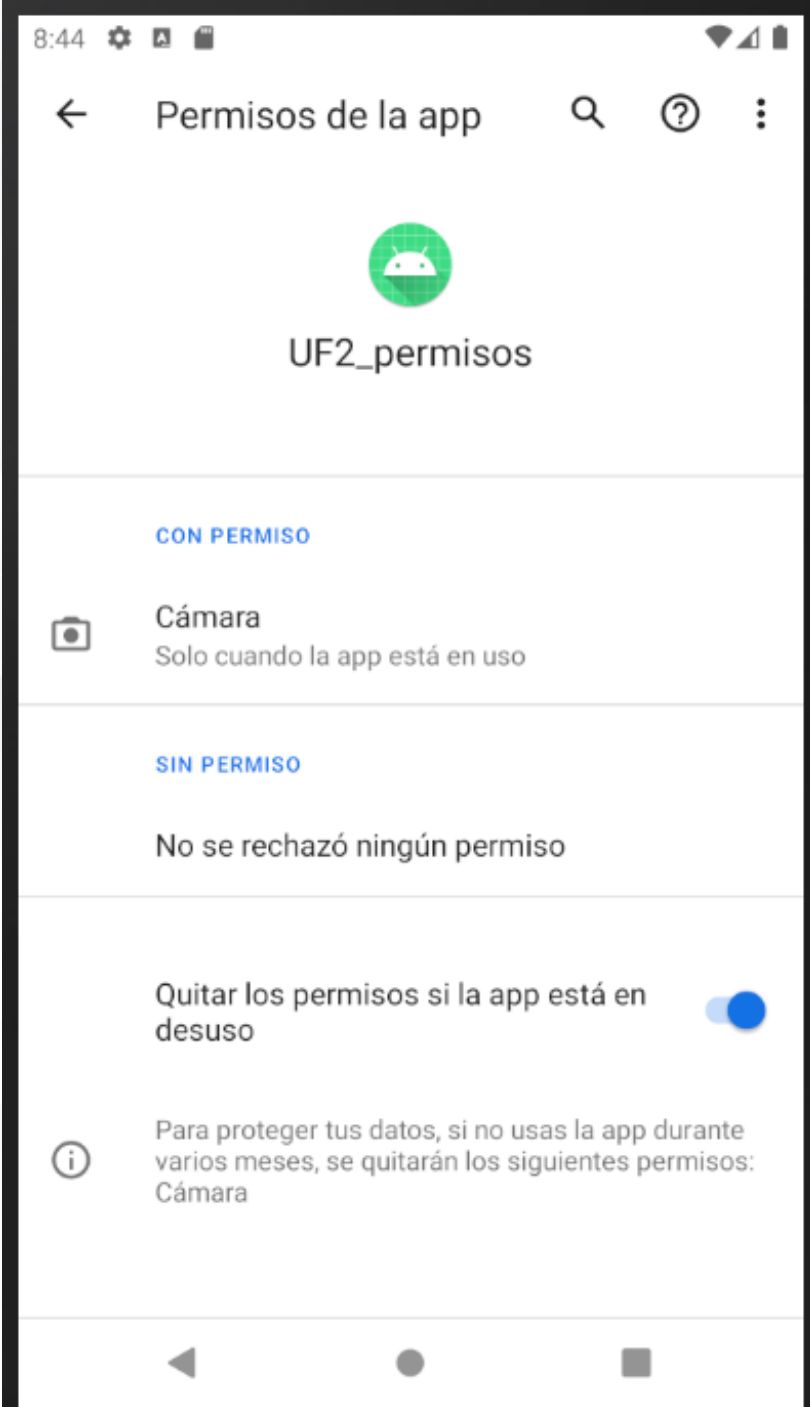
## > PROGRAMANDO UN PERMISO

- En las propiedades de la Aplicación podemos aceptar o deshabilitar los permisos que se necesitan para ejecutarla (parcial o totalmente)



## > PROGRAMANDO UN PERMISO

- Una vez hayamos configurado un elemento del layout que active el recurso del permiso, tendremos que referenciarlo en el Activity correspondiente y programar las funciones para:
  - Comprobar si se ha otorgado el permiso
  - Solicitar el permiso
  - Qué hacer si se ha concedido el permiso



- COMPROBAR PERMISO
- Usaremos la función `checkCameraPermission()` para comprobar si se han otorgado los permisos.

En caso de no haberlo hecho, lanzará otra función para hacerlo

```
private fun checkCameraPermission() {  
    if (ContextCompat.checkSelfPermission( context: this,  
        Manifest.permission.CAMERA)  
        != PackageManager.PERMISSION_GRANTED) {  
        //El permiso no está aceptado.  
        requestCameraPermission()  
    } else {  
        //El permiso está aceptado.  
    }  
}
```

- COMPROBAR PERMISO
- Usaremos la función `requestCameraPermission` para mostrar la ventana que solicita el permiso

```
private fun requestCameraPermission() {
    if (ActivityCompat.shouldShowRequestPermissionRationale( activity: this,
        Manifest.permission.CAMERA)) {
        //El permiso ya ha rechazado el permiso anteriormente
        //Tendremos que indicar al usuario que active el permiso desde Ajustes
        //...
    } else {
        //Nunca se ha preguntado si se concede el permiso
        //Mostrar diálogo para aceptar el permiso
        ActivityCompat.requestPermissions( activity: this,
            arrayOf(Manifest.permission.CAMERA),
            CAMERA_REQUEST_CODE)
    }
}
```



- COMPROBAR PERMISO
- Usaremos la función `onRequestPermissionsResult()` para configurar qué hará la aplicación en función de qué ocurrió en la ventana anterior, esto es, si se aceptó o no

```
override fun onRequestPermissionsResult(  
    requestCode: Int,  
    permissions: Array<String>, grantResults: IntArray  
) {  
    when (requestCode) {  
        CAMERA_REQUEST_CODE -> {  
            if ((grantResults.isNotEmpty() && grantResults[0] == PackageManager.PERMISSION_GRANTED)) {  
                //El usuario ha aceptado el permiso  
                // Podemos lanzar la funcionalidad desde aquí.  
            } else {  
                //El usuario ha rechazado el permiso,  
                // Podemos desactivar la funcionalidad o mostrar una vista/diálogo.  
            }  
            return  
        }  
        else -> {  
            // Ignorar el resto de peticiones  
        }  
    }  
}
```

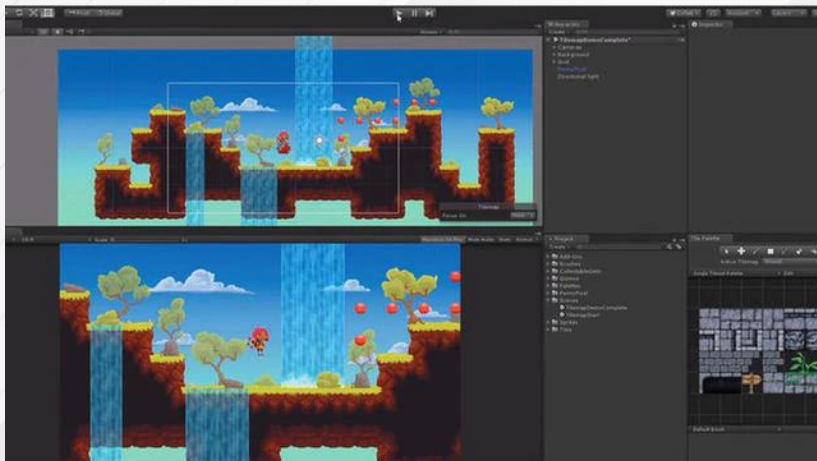
UF3:

---

JUEGOS 2D / 3D



- Qué es UNITY
- Es uno de los motores de creación de juegos más usados en la actualidad
- Motor de videojuego: software que tiene una serie de rutinas de programación que permiten el diseño, la creación y el funcionamiento de un entorno interactivo (videojuego)





- FUNCIONALIDADES DE UN MOTOR DE VIDEOJUEGOS
- Motor gráfico para renderizar gráficos 2D y 3D
- Motor físico que simule las leyes de la física
- Animaciones
- Sonidos
- Inteligencia Artificial
- Programación o scripting



- CLASIFICACIÓN DE MOTORES
- En función de las herramientas disponibles:
  - Librerías Gráficas: DirectX, OpenGL, SDL, XNA
  - Motores: desarrollo visual completo. OGRE, Unreal, idTech
  - Herramientas de creación especializadas: GameMaker, ShiVa, Unity
- En función de la licencia:
  - Motores privados
  - Motores Opensource

- DESCARGAR UNITY
- Web: <https://unity3d.com/es/get-unity/download>
- Elegir Unity Hub



# Descargar Unity

¡Bienvenido! Está aquí porque desea descargar Unity, la plataforma de desarrollo más popular del mundo para crear juegos multiplataforma y experiencias interactivas 2D y 3D.

Antes de descargar, elija la versión de Unity que sea adecuada para usted.

Elige tu Unity + descargar

Descarga Unity Hub

[Descubrir más acerca del nuevo Unity Hub aquí.](#)





- INSTALAR UNITY
- Requiere licencia
- Manual Activation

### Manual Activation

- 1 Save License Request
- 2 Load License

Click here to save your license request key.

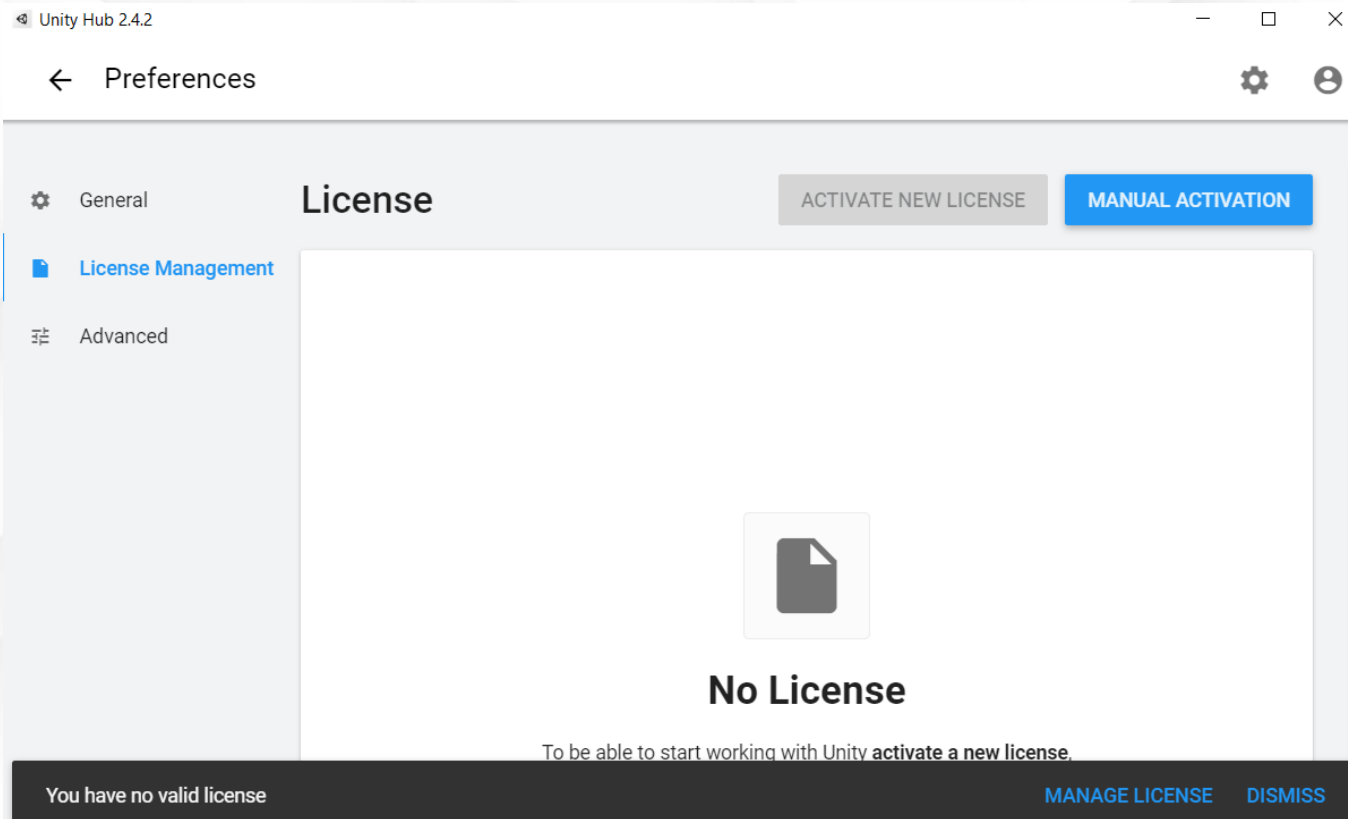
SAVE LICENSE REQUEST

Now go to manual activation page [license.unity3d.com/manual](https://license.unity3d.com/manual)

CANCEL

BACK

NEXT





- INSTALAR UNITY
- Requiere licencia
- Manual Activation
- Crear cuenta



Sign into your Unity ID

If you don't have a Unity ID, please [create one](#).

Email


Password


[Forgot your password?](#)  
[Can't find your confirmation email?](#)

Remember me

Sign in

Or

 Sign in with google

 Sign in with facebook

 Single Sign on



- INSTALAR UNITY
- Requiere licencia
- Manual Activation
- Crear cuenta
- Después de activar la cuenta,  
sube el fichero de licencia del  
paso “Manual Activation”



Sign in

License

Start

## Manual activation

Welcome to the second step in the Unity manual activation process (first you must create a license request file inside Unity). Upload the license request file to proceed.

Choose the license request file

✓ Unity\_lic.alf

Browse

It should be named Unity\_vX.alf or .ilf

Next



- INSTALAR UNITY
- Requiere licencia
- Manual Activation
- Crear cuenta
- Después de activar la cuenta,  
sube el fichero de licencia del  
paso “Manual Activation”
- Elige modo personal



## Activate your license

Thank you for downloading Unity. Please select one of the following license options.

- Unity Plus or Pro
- Unity Personal Edition

### Please select one of the options below

- The company or organization I represent earned **more than** \$100,000 in gross revenue in the previous fiscal year.
- The company or organization I represent earned **less than** \$100,000 in gross revenue in the previous fiscal year.
- I don't use Unity in a professional capacity.

Next

[Why does Unity need to know this?](#)

- INSTALAR UNITY
- Requiere licencia
- Manual Activation
- Crear cuenta
- Después de activar la cuenta,  
sube el fichero de licencia del  
paso “Manual Activation”
- Elige modo personal
- Descarga la licencia



## Download license file

1. Download your Unity license file by clicking "Download license file" below.
2. Return to your Unity installation with the downloaded file.
3. Open Manage License -> Manual Activation and click "Load License".

[Download license file](#)



- INSTALAR UNITY
- Requiere licencia
- Manual Activation
- Crear cuenta
- Después de activar la cuenta, sube el fichero de licencia del paso “Manual Activation”
- Elige modo personal
- Descarga la licencia final
- Sube la licencia final

### Manual Activation

✓ Save License Request 2 Load License

Load license file to this machine.

License File \* ...

CANCEL BACK CONFIRM





- INSTALAR UNITY
- Inicia sesión
- Licencia activada para uso personal

Unity Hub 2.4.2

← Preferences

General

License Management

Advanced

### License

ACTIVATE NEW LICENSE

MANUAL ACTIVATION

#### Personal

Unity Personal is a great place for beginners and hobbyists to get started. It includes access to all core game engine features, continuous updates, beta releases, and all publishing platforms.

Activation: 2020/11/26

[Buy Professional Edition](#) — [Help](#) — [FAQ](#)

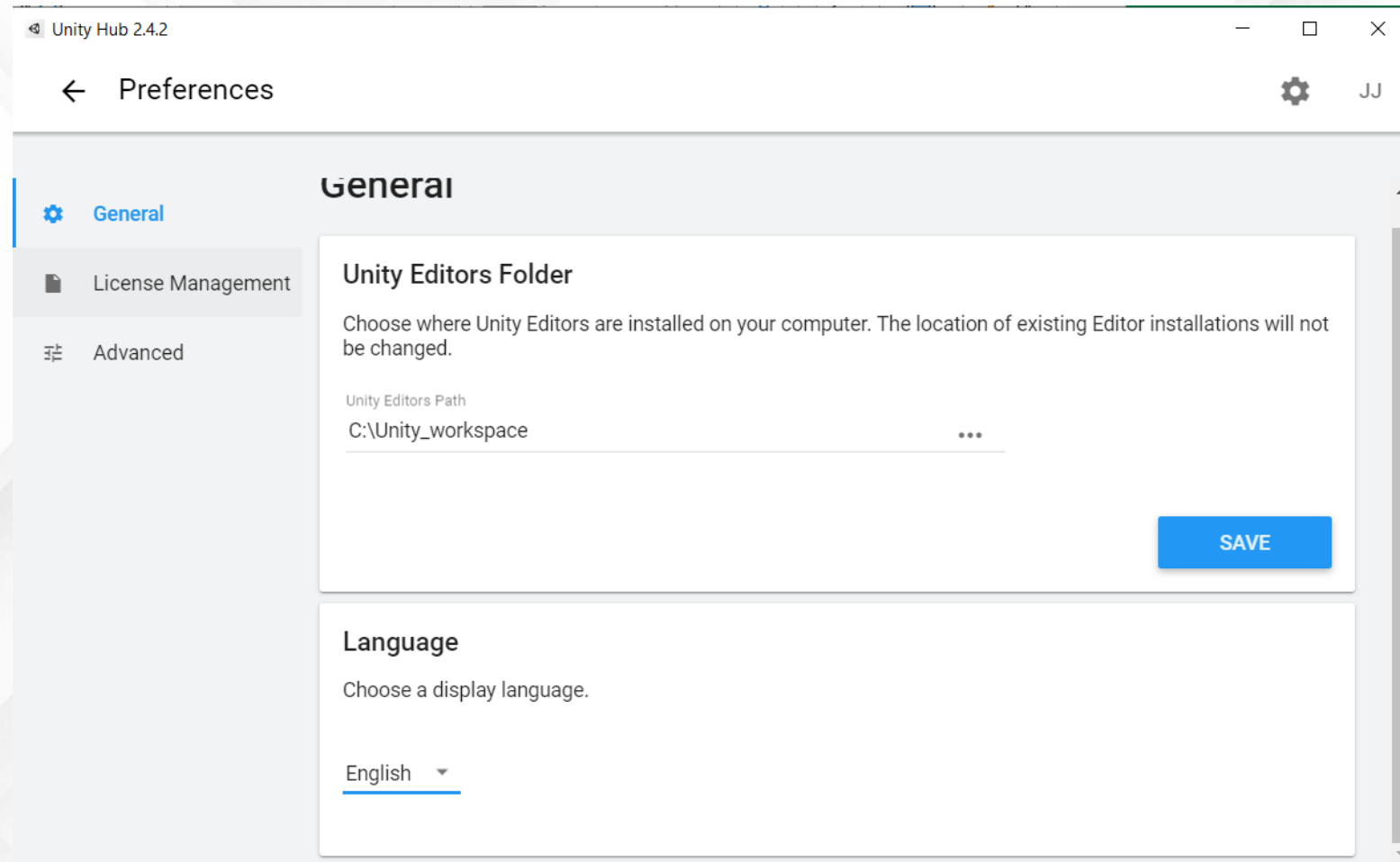
RETURN LICENSE

CHECK FOR UPDATES

You need to be logged in to manage your license. [LOGIN](#) [DISMISS](#)

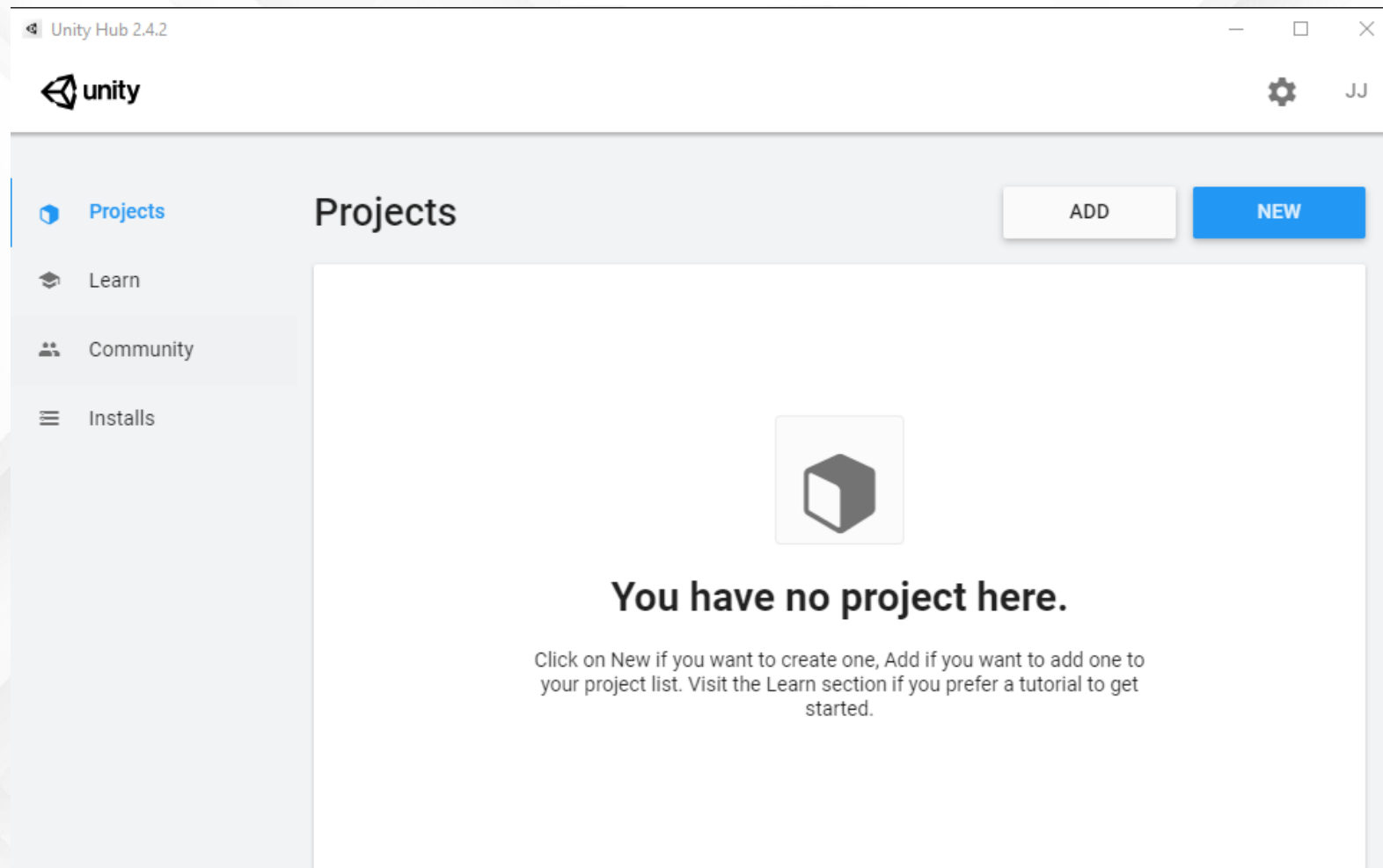


- INSTALAR UNITY
- Inicia sesión
- Licencia activada para uso personal
- Configurar el directorio de trabajo y el idioma

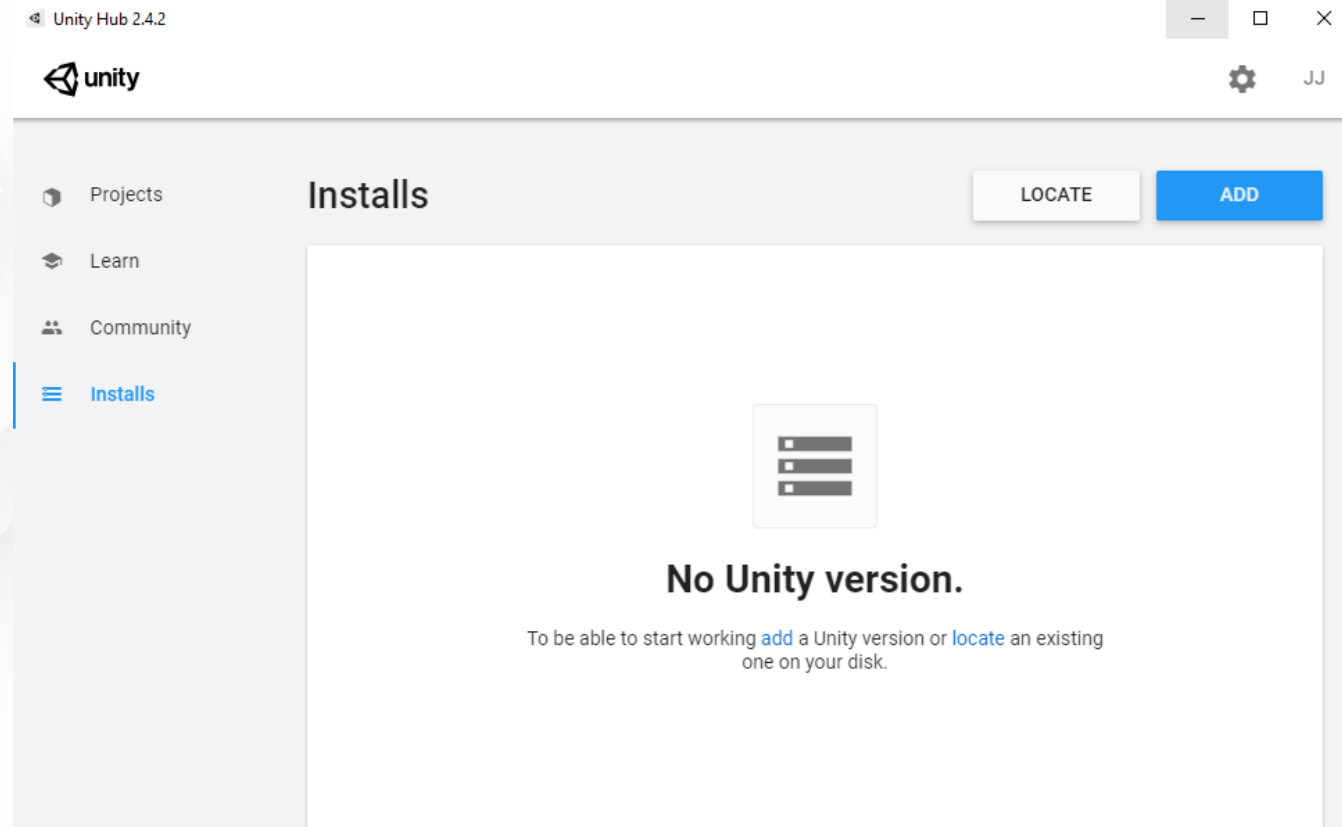




- INSTALAR UNITY
- Espacio de trabajo
- Crear nuevo proyecto



- INSTALAR UNITY
- Espacio de trabajo
- Crear nuevo proyecto
- Instalaciones en Unity





- INSTALAR UNITY
- Espacio de trabajo
- Crear nuevo proyecto
- Instalaciones en Unity
- Añadiremos la última versión estable de Unity

## Add Unity Version

- 1 Select a version of Unity ————— 2 Add modules to your install

Can't find the version you're looking for? Visit our [download archive](#) for access to [long-term support](#) and [patch releases](#), or join our [Open Beta program](#) releases.

### Recommended Release

Unity 2019.4.15f1 (LTS)

### Official Releases

Unity 2020.1.15f1

Unity 2018.4.29f1 (LTS)

### Pre-Releases

Unity 2021.1.0a7 (Alpha)

CANCEL

BACK

NEXT



- INSTALAR UNITY
- Espacio de trabajo
- Crear nuevo proyecto
- Instalaciones en Unity
- Añadiremos la última versión estable de Unity
- Elegimos el módulo para nuestra instalación

**Add Unity Version** ✕

1  Select a version of Unity 2  Add modules to your install

Add modules to Unity 2019.4.15f1 : total space available 304.9 GB - total space required 10.0 GB

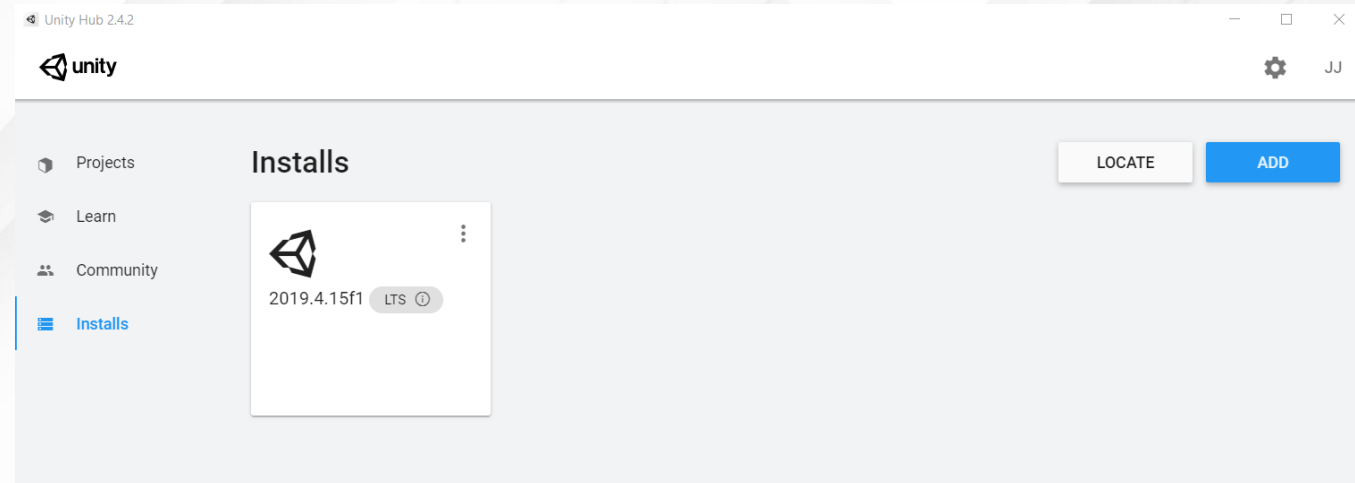
Dev tools	Download Size	Install Size
<input checked="" type="checkbox"/> Microsoft Visual Studio Community 2019	1.4 GB	1.3 GB
Platforms		
> <input type="checkbox"/> Android Build Support	239.3 MB	1.1 GB
<input type="checkbox"/> iOS Build Support	668.4 MB	2.8 GB
<input type="checkbox"/> tvOS Build Support	342.8 MB	1.5 GB
<input type="checkbox"/> Linux Build Support (IL2CPP)	56.8 MB	252.1 MB
<input type="checkbox"/> Linux Build Support (Mono)	56.8 MB	260.6 MB

CANCEL BACK NEXT



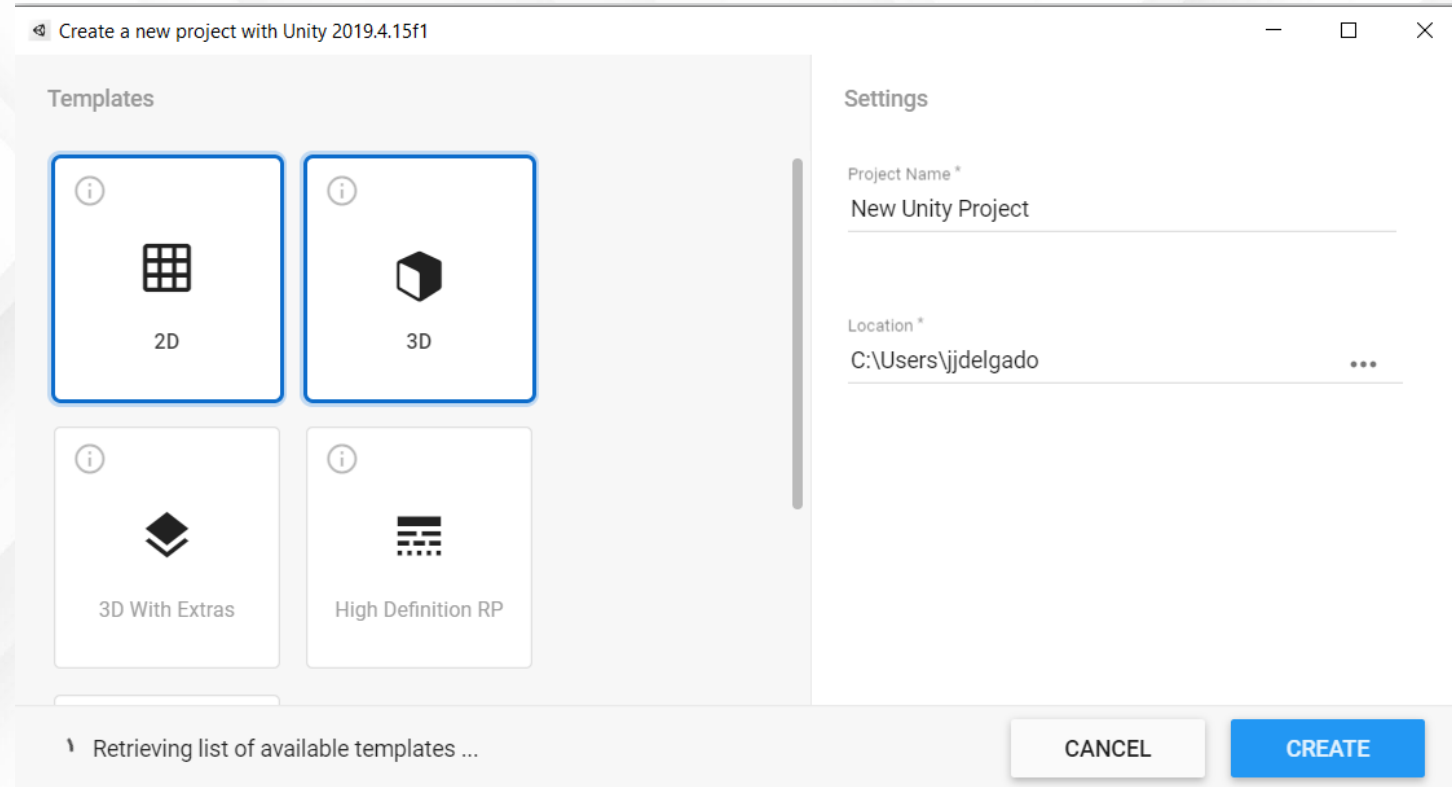


- INSTALAR UNITY
- Espacio de trabajo
- Crear nuevo proyecto
- Instalaciones en Unity
- Añadiremos la última versión estable de Unity
- Elegimos el módulo para nuestra instalación
- Resultado final

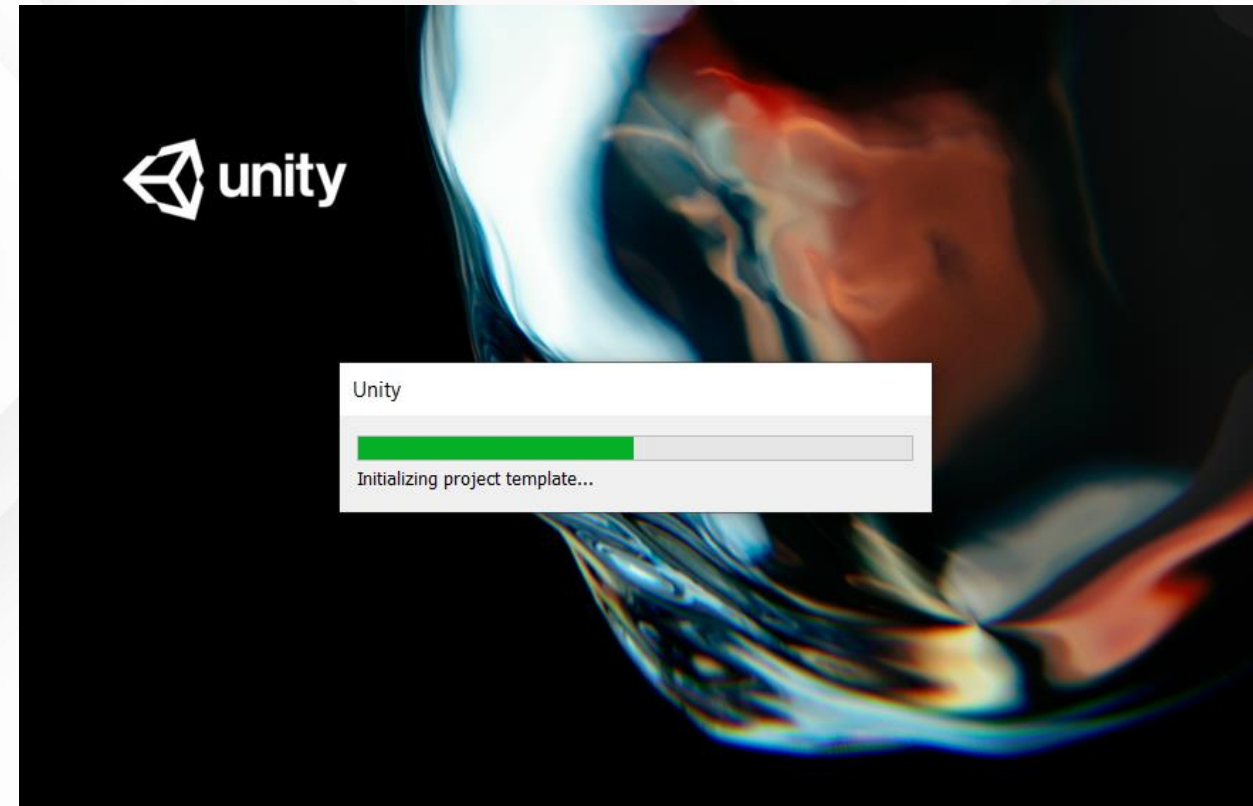




- CREAR UN PROYECTO
- Pulsar “NEW”
- Elegimos el tipo de Proyecto
- En el ejemplo que vamos a hacer:
- Template: 2D
- Project Name: VT\_ANDROID
- Location: donde queramos almacenarlo

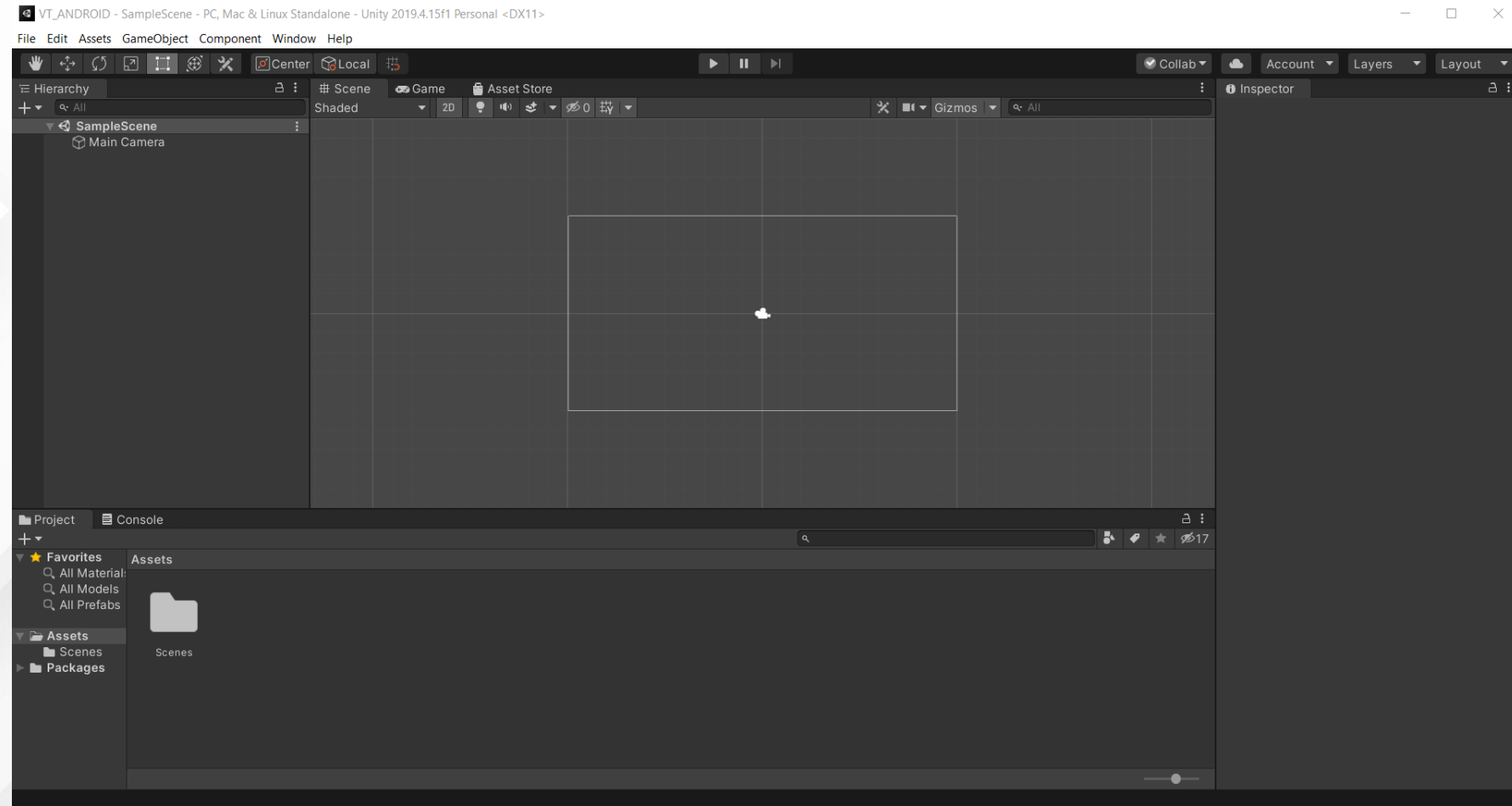


- CREAR UN PROYECTO
- Se crea el proyecto



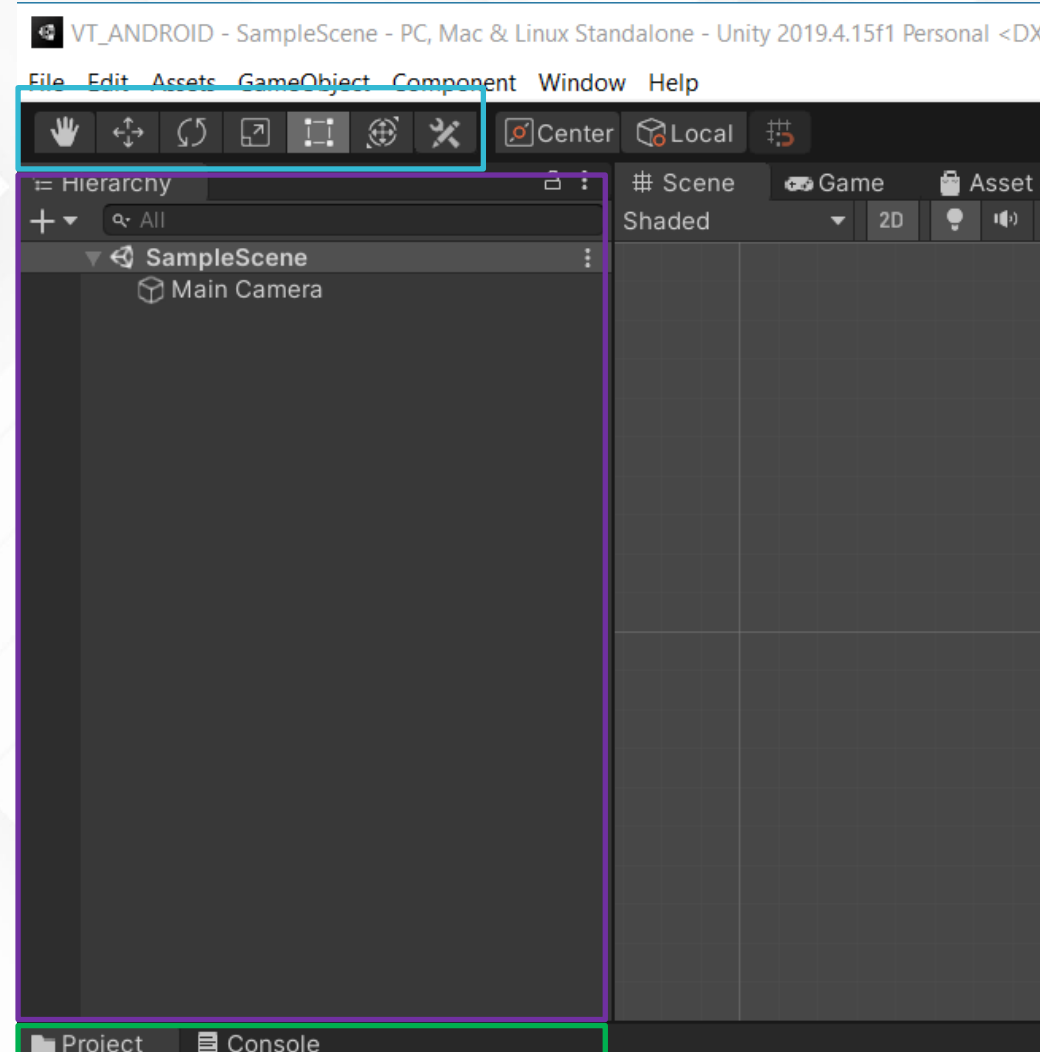


- CREAR UN PROYECTO
- Se crea el proyecto
- Escritorio de trabajo



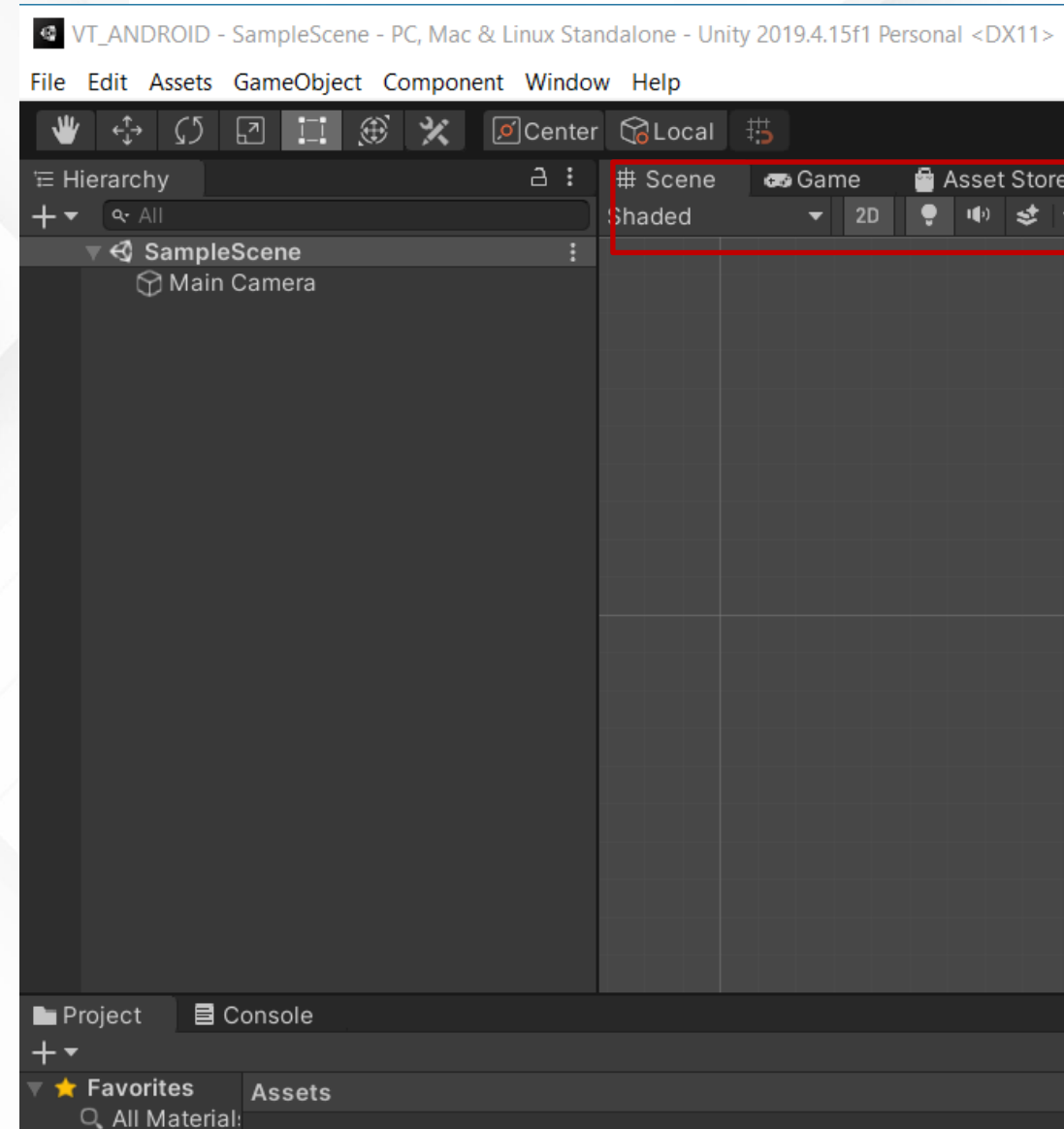


- ESCRITORIO DE UNITY
- Barra de Herramientas
  - Para modificar los elementos del juego y moverlos por el espacio
- Jerarquía
  - Para organizar los objetos que vamos a incluir en el proyecto
- Pestaña Project
  - Todos los elementos que forman parte del paquete (Assets)
- Pestaña Console
  - Mensajes de compilación y ejecución



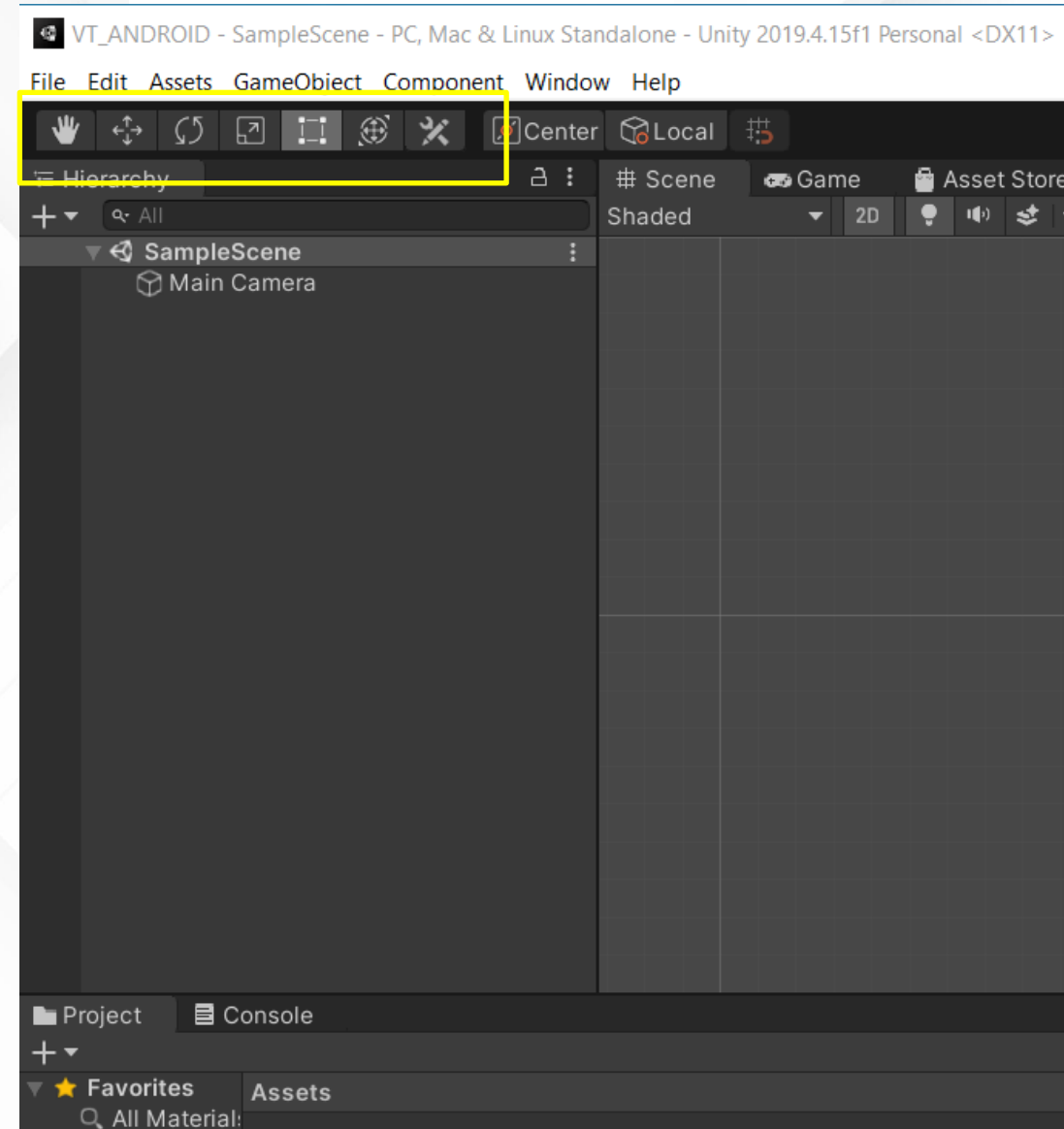


- ESCRITORIO DE UNITY
- **SECENE**
  - Modo de diseño que tenemos, cómo se colocan los objetos en el mundo
- **GAME**
  - Funcionamiento del juego en tiempo de ejecución
- **ASSET STORE**
  - Tienda de Asset, para comprar o descargar elementos gratuitos o de pago

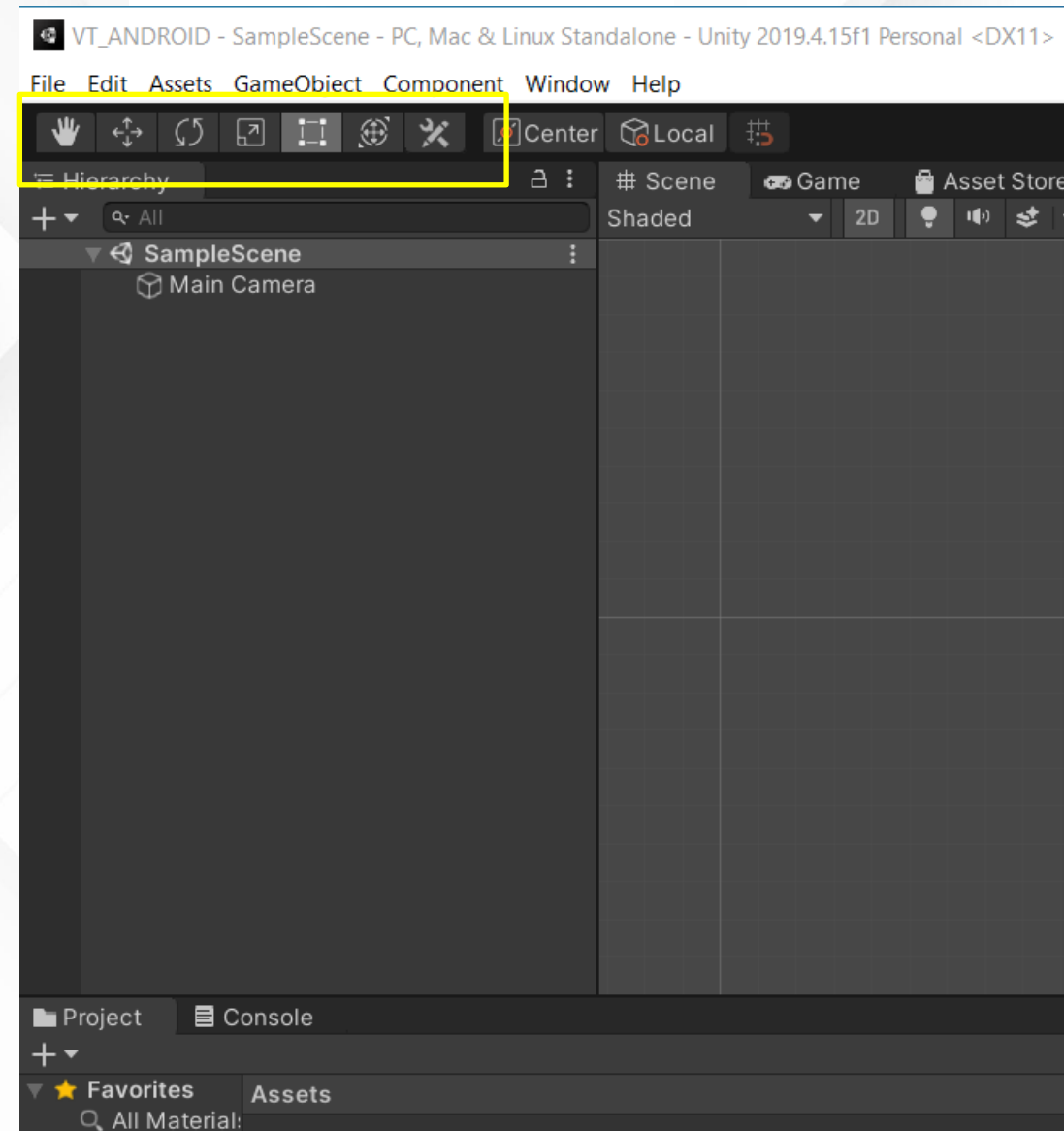




- ESCRITORIO DE UNITY
- MANO (tecla Q)
  - Para movernos como usuarios en la pantalla, desplazando todo el lienzo
- MOVE TOOL
  - Desplazar los objetos en ejes X e Y
- ROTATE TOOL
  - Rotar el objeto un determinado ángulo en eje X, Y y Z

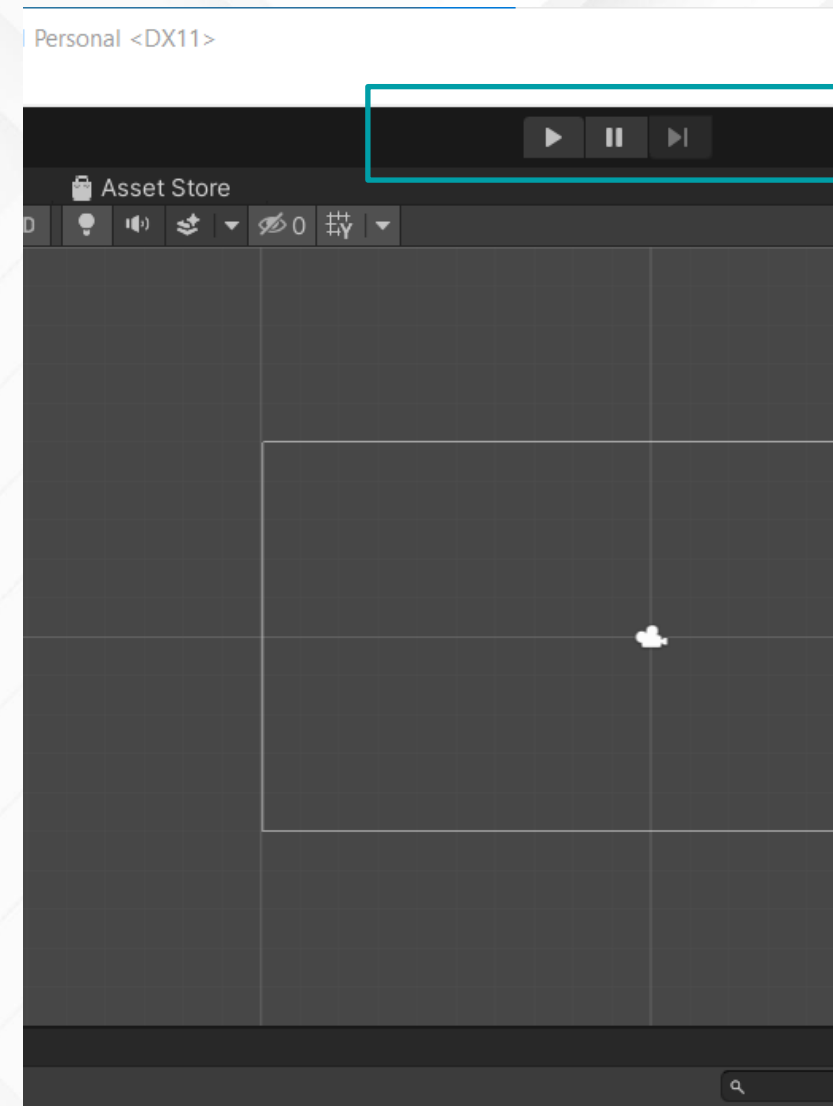


- ESCRITORIO DE UNITY
- SCALE TOOL
  - Reescalar el objeto un determinado ángulo en eje X, Y o en ambos al mismo tiempo, en el cuadrado
- RECT TOOL
  - Mover el elemento a donde queramos y ampliar su tamaño (combinación de Move y Scale)
- MOVE ROTATE AND SCALE TOOL
  - Combinación de todas las herramientas anteriores
  - Se recomienda usar las anteriores por separado



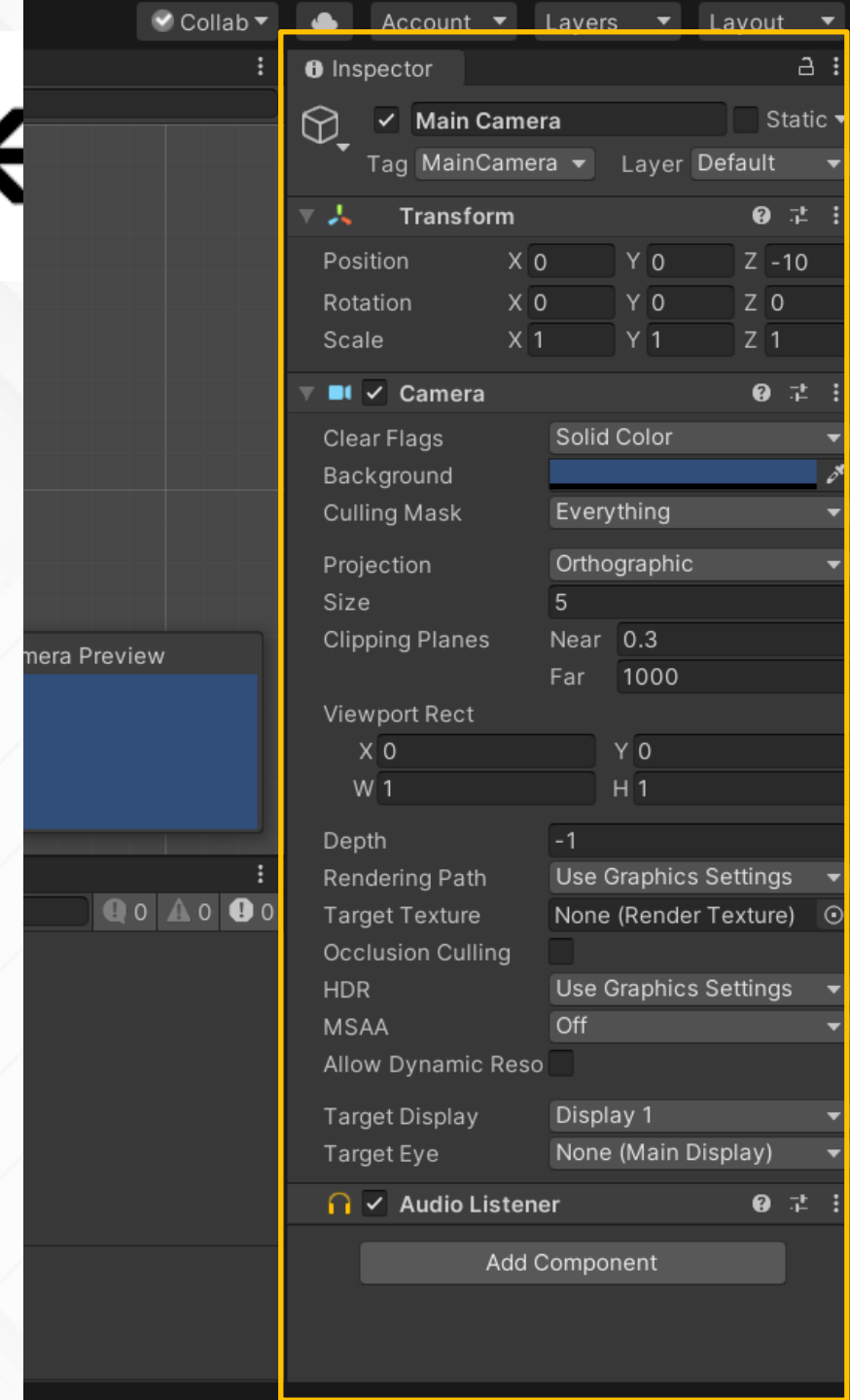


- ESCRITORIO DE UNITY
- REPRODUCCIÓN
  - PLAY: Inicia la reproducción del juego
  - PAUSE: Detiene la reproducción del juego
  - AVANCE: Reproduce el juego fotograma a fotograma
- IMPORTANTE: Todo las acciones que realicemos mientras esté el PLAY activado, se borrará



## > UNITY

- ESCRITORIO DE UNITY
- INSPECTOR
  - Propiedades del objeto o elemento seleccionado
  - Componente Transform: indica su posición, rotación y escala
  - Componente Camera: para pintar el mundo que estamos pintando en la escena
  - Componente Audio Listener: por si queremos añadir un audio

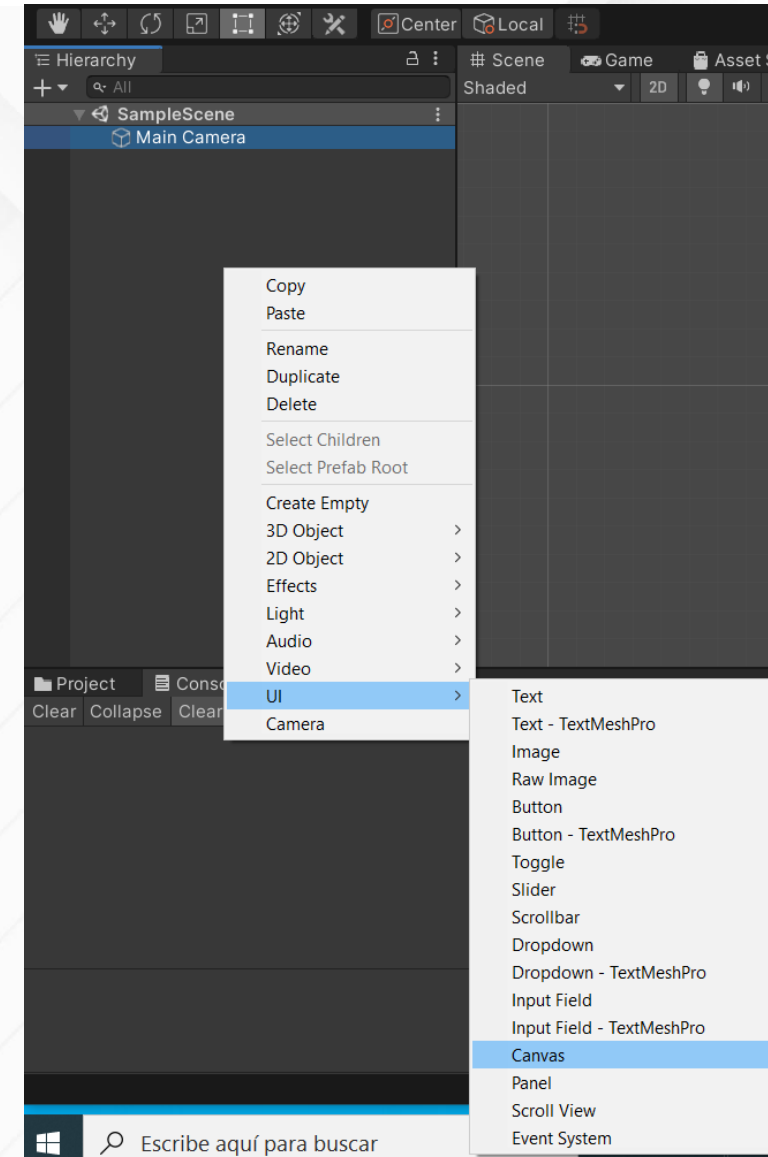




- PRIMER PROYECTO UNITY

- INTERFAZ GRÁFICA

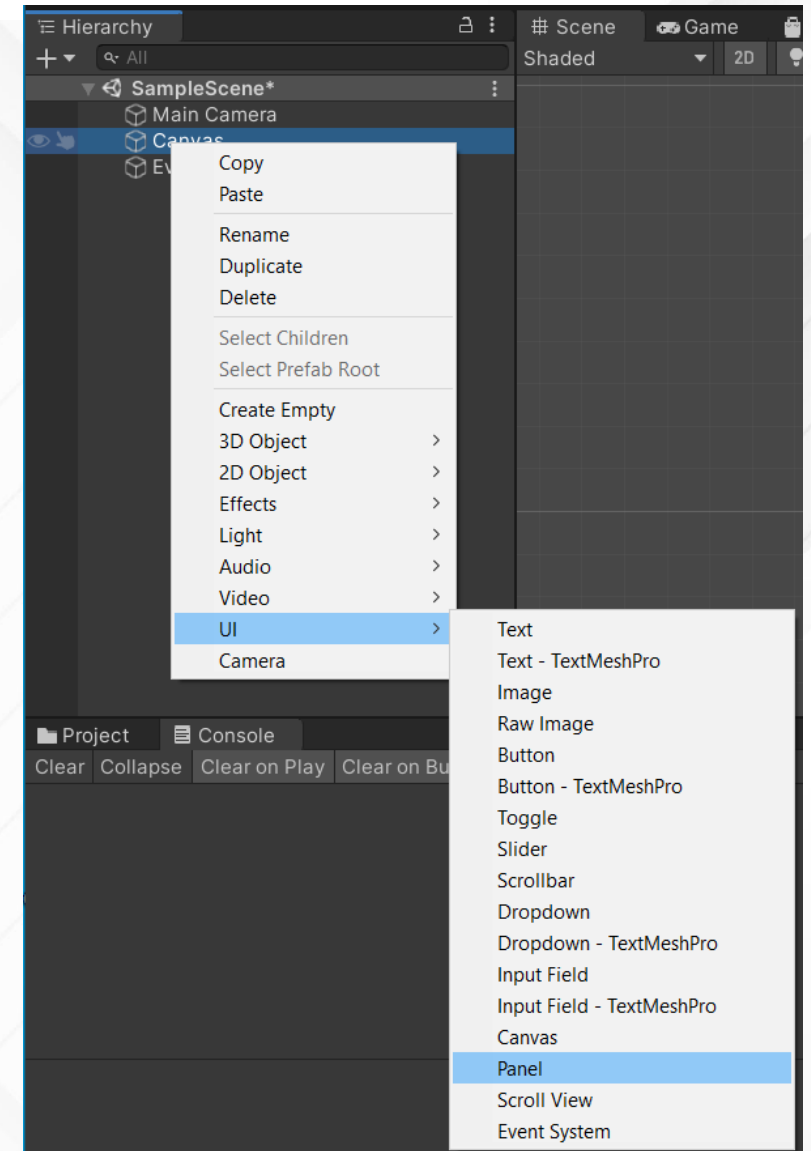
- Vamos a insertar elementos desde la parte visual, no desde código
- Hay muchos componentes ya creados que podemos usar
- Desde Hierarchy, clic derecho y seleccionamos UI / Canvas



- PRIMER PROYECTO UNITY

- INTERFAZ GRÁFICA

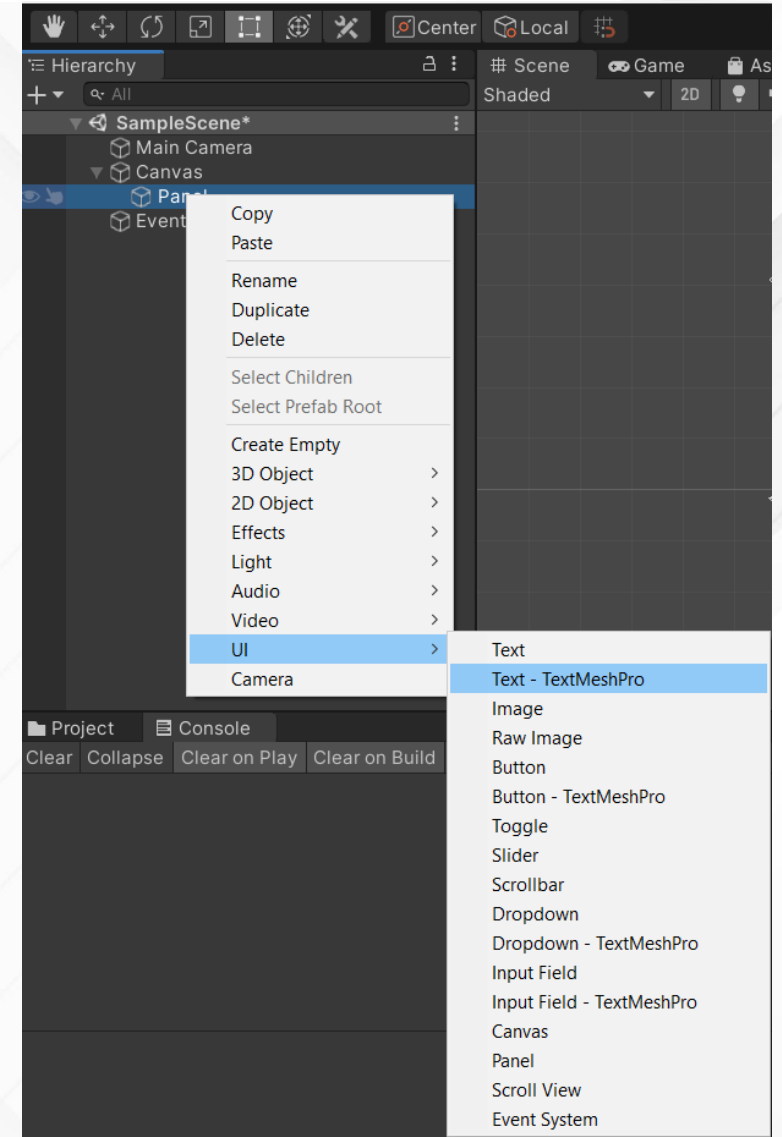
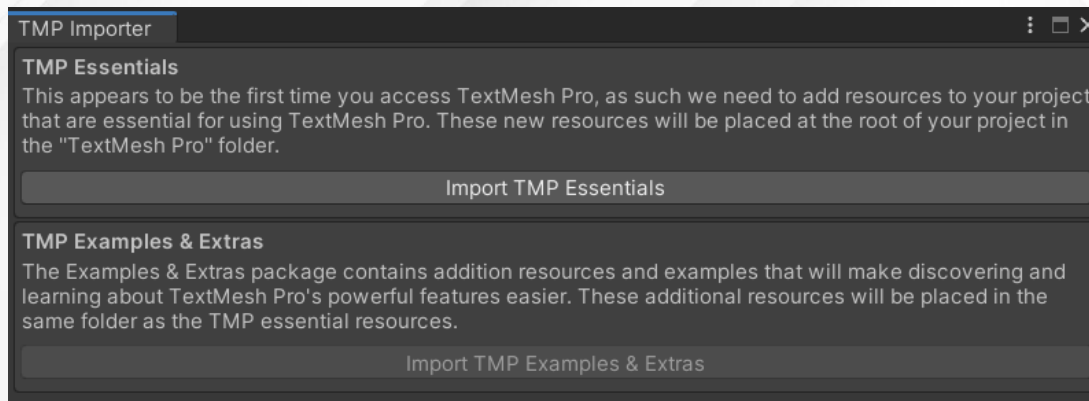
- Para visualizarlo dentro del canvas, vamos a añadir UI panel: clic derecho en canvas y seleccionamos UI / Panel





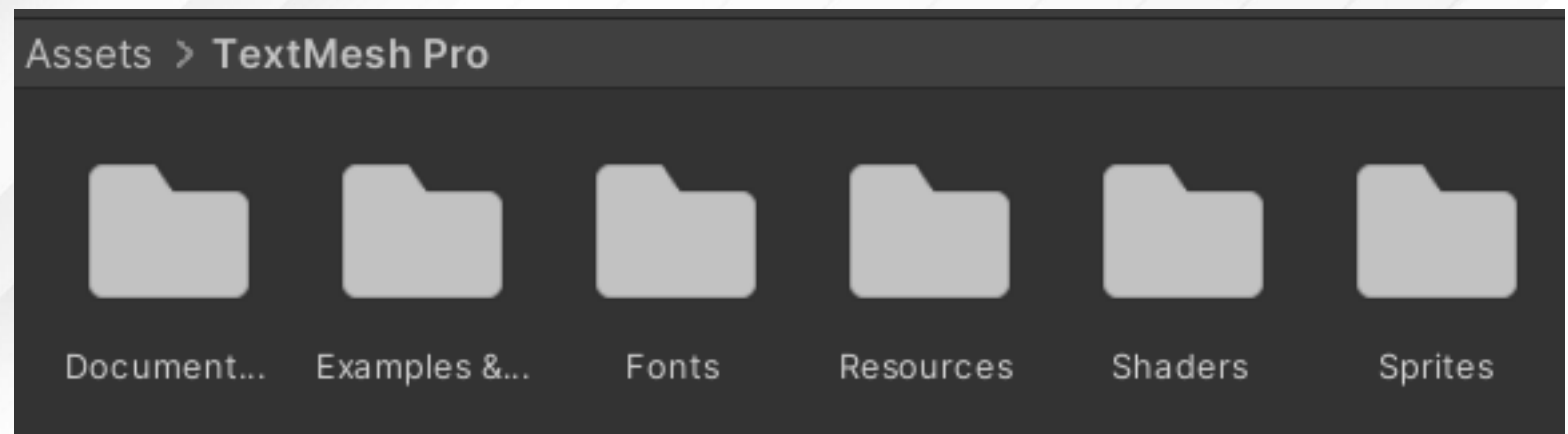
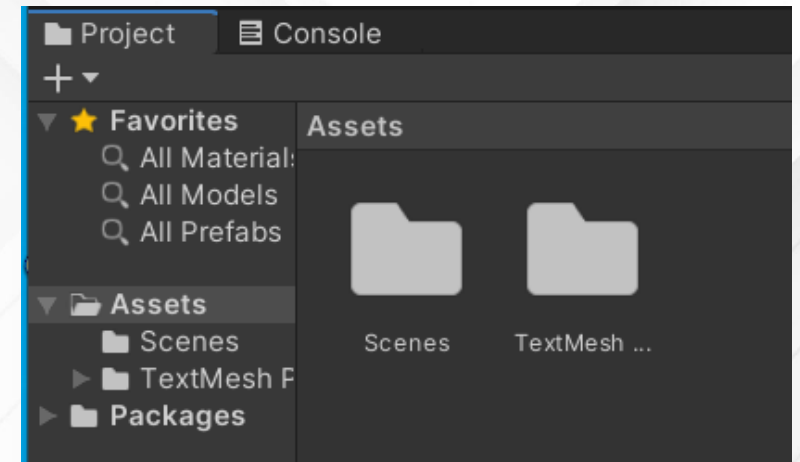


- PRIMER PROYECTO UNITY
- INTERFAZ GRÁFICA
  - Vamos a insertar un texto
  - Dos opciones
    - Text
    - Text MeshPro (igual que la anterior pero con más funciones)
    - La primera vez que lo importáis os pedirá instalar una libería



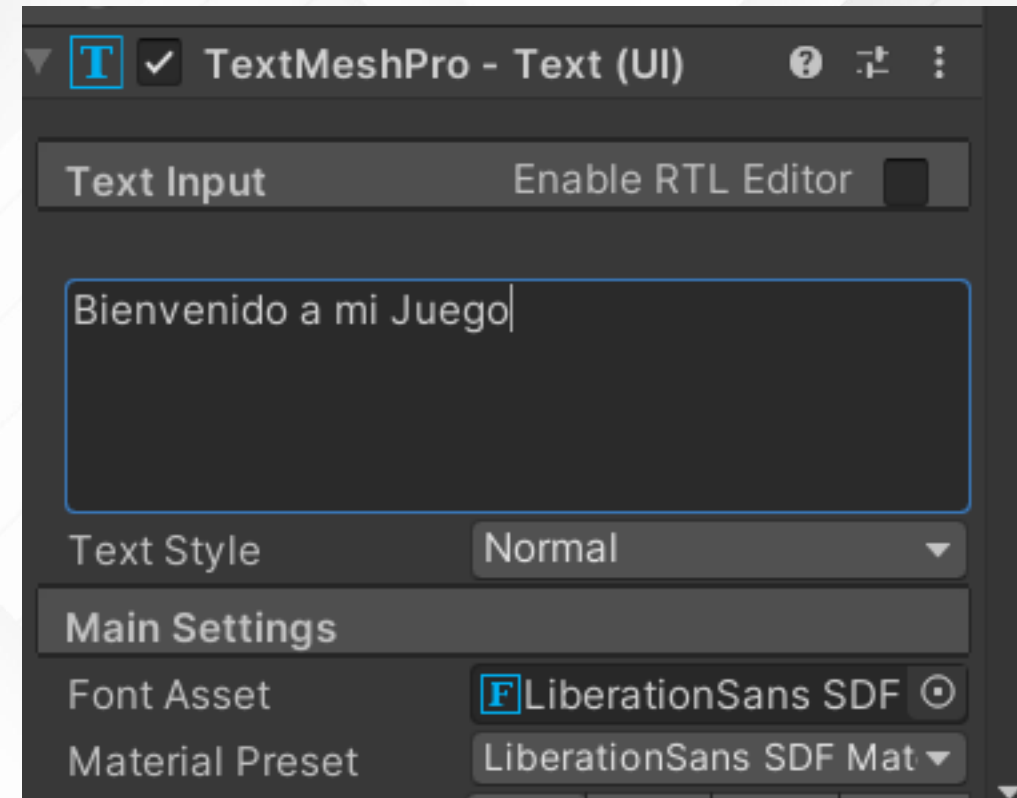
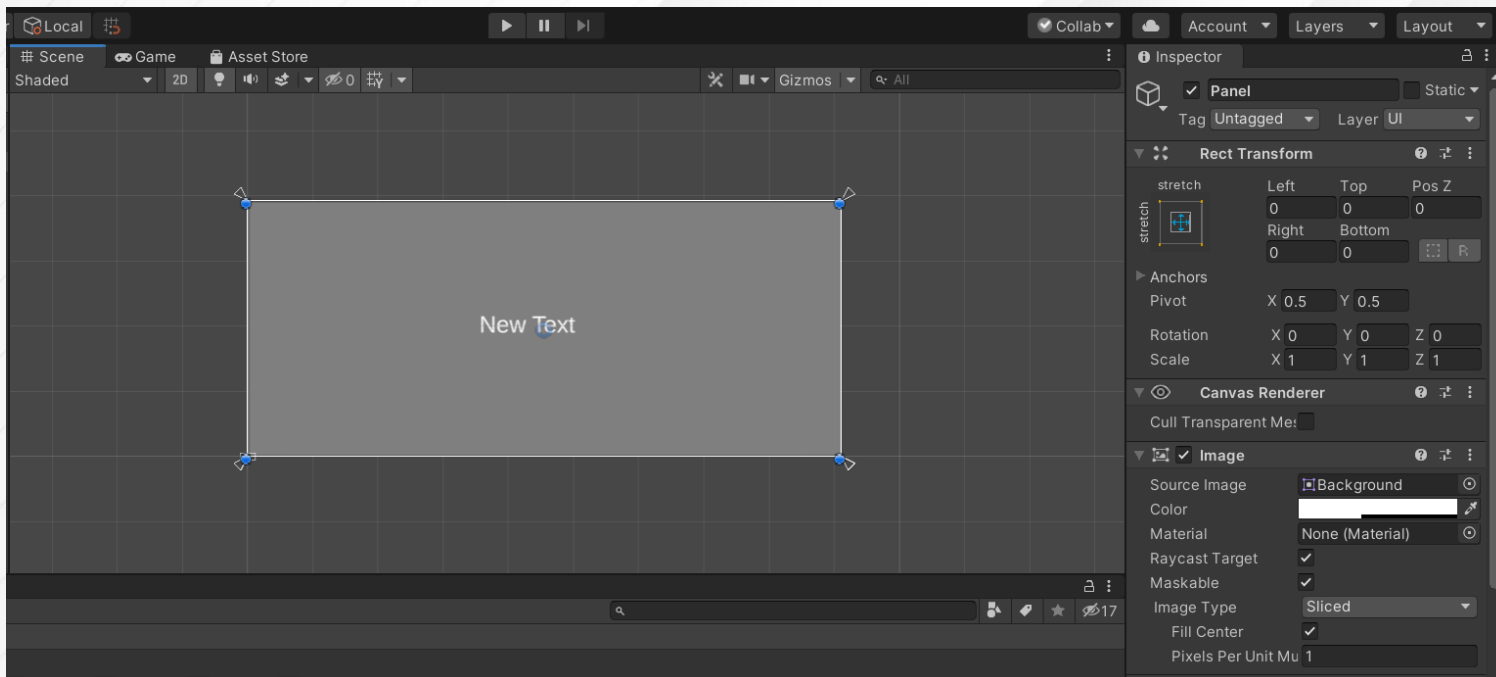


- PRIMER PROYECTO UNITY
- INTERFAZ GRÁFICA
  - En la carpeta de Assets nos deberá aparecer la carpeta de TextMesh Pro que hemos instalado y añadido al proyecto



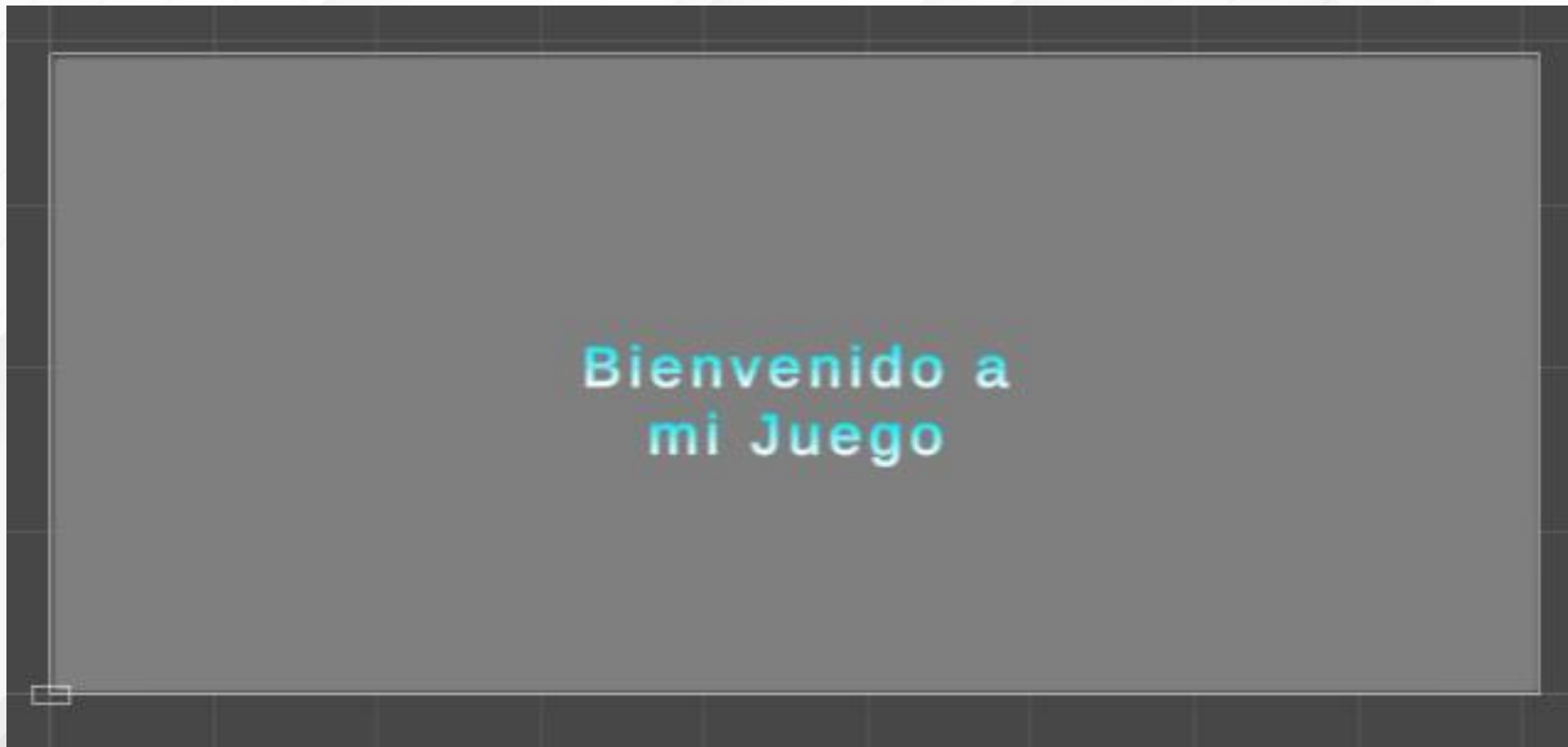


- PRIMER PROYECTO UNITY
- INTERFAZ GRÁFICA
  - Podmos definir las propiedades del texto (posición, el punto de anclaje de referencia, el texto que se muestra,...) a partir de las opciones del panel





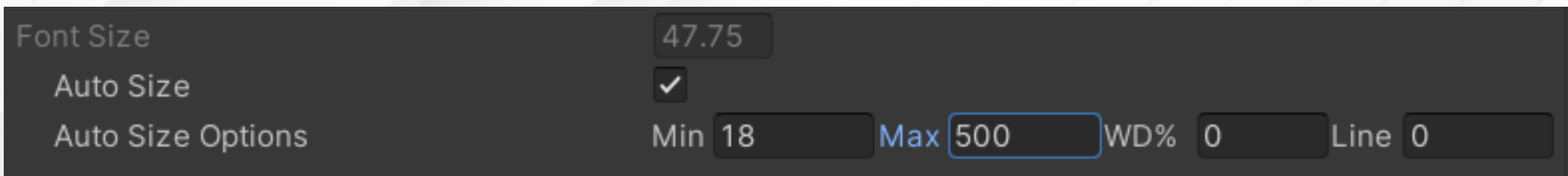
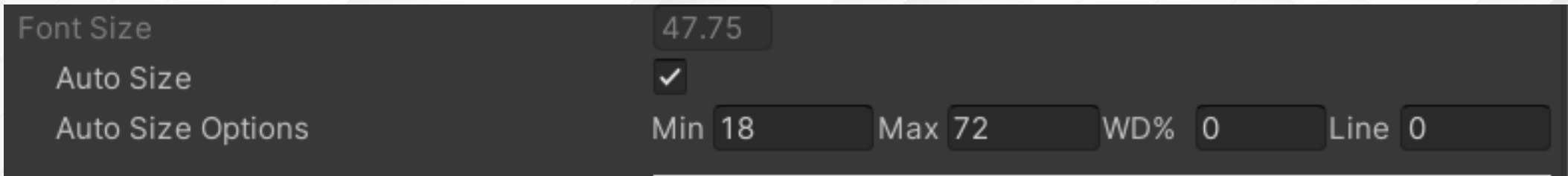
- PRIMER PROYECTO UNITY
- INTERFAZ GRÁFICA
  - Configura el texto para que aparezca como en la imagen





- PRIMER PROYECTO UNITY
- INTERFAZ GRÁFICA

- La opción Autosize permite que el tamaño del texto se adapte al cambio de tamaño del objeto
- Autosize Min y Max para definir estos valores





- PRIMER PROYECTO UNITY
- INTERFAZ GRÁFICA
  - Pulsamos PLAY
  - Si vemos que el texto no se adapta al tamaño de pantalla

Bienvenido a  
mi Juego



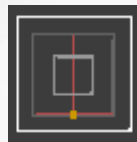


- PRIMER PROYECTO UNITY
- INTERFAZ GRÁFICA

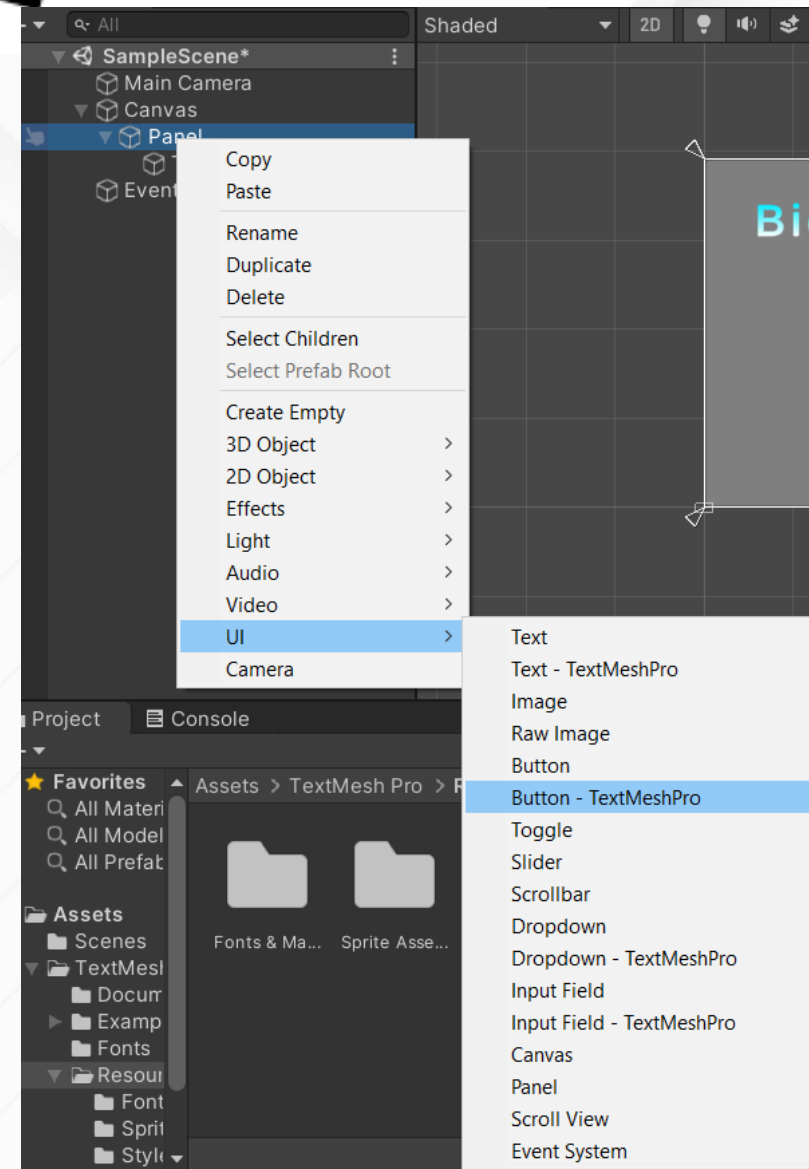
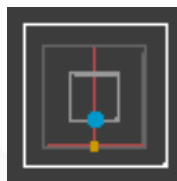
- Vamos a insertar un botón
- Dos opciones
  - Button
  - Button MeshPro (igual que la anterior pero con más funciones)
  - Lo centramos alineado en la parte inferior

- RectTransform

- Icono



- Presionando Shift + Alt





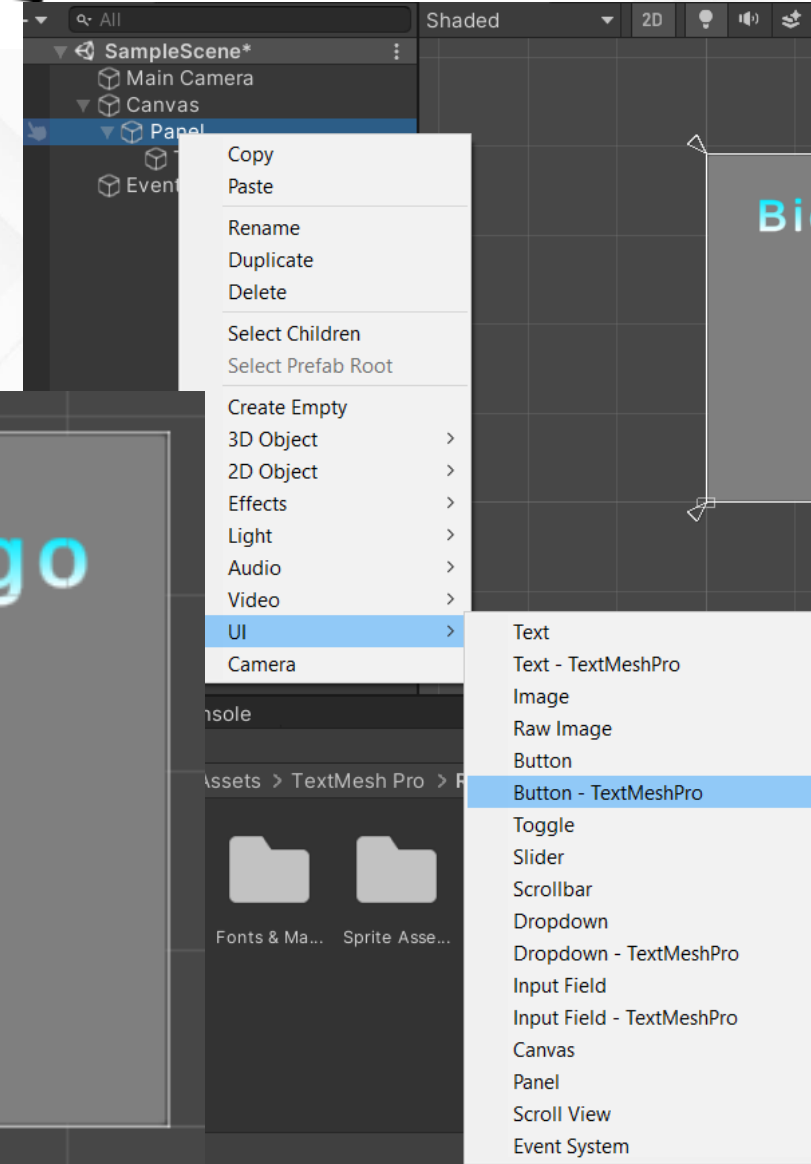
- PRIMER PROYECTO UNITY
- INTERFAZ GRÁFICA

- Vamos a insertar un botón
- Dos opciones

- But
- But
- Lo c

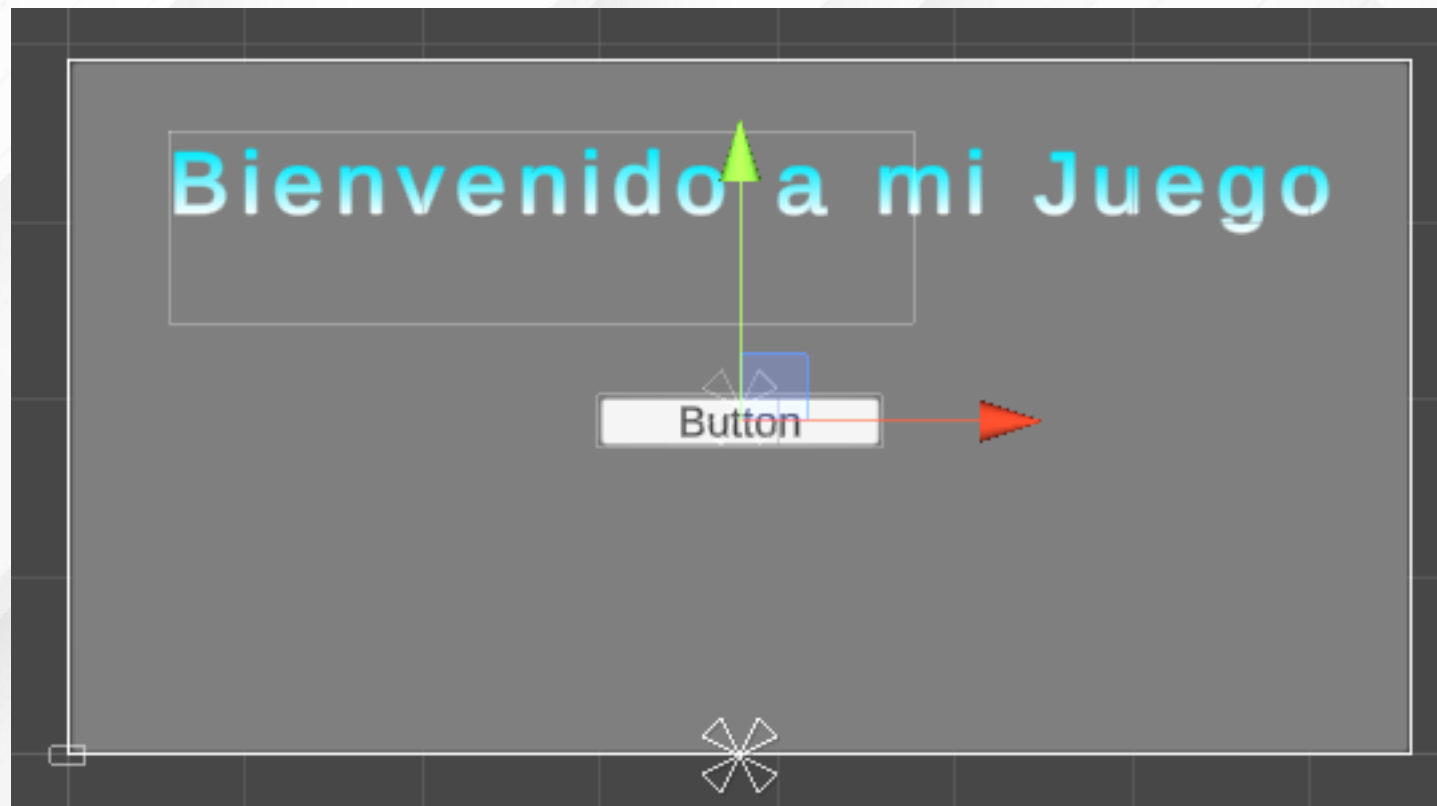
Bienvenido a mi Juego

Button





- PRIMER PROYECTO UNITY
- INTERFAZ GRÁFICA
  - Desplazamos el botón hacia arriba
  - Anchor fijado en el eje Y



- PRIMER PROYECTO UNITY
- INTERFAZ GRÁFICA

➤ Cambiamos el diseño del botón:

- Texto: Comenzar a jugar
- Fondo:
  - Normal color:normal
  - Highlighted Color:al pasar el ratón por encima
  - Pressed Color: presionado
  - Selected Color:seleccionado
  - Disabled Color:no disponible



Bienvenido a mi Juego

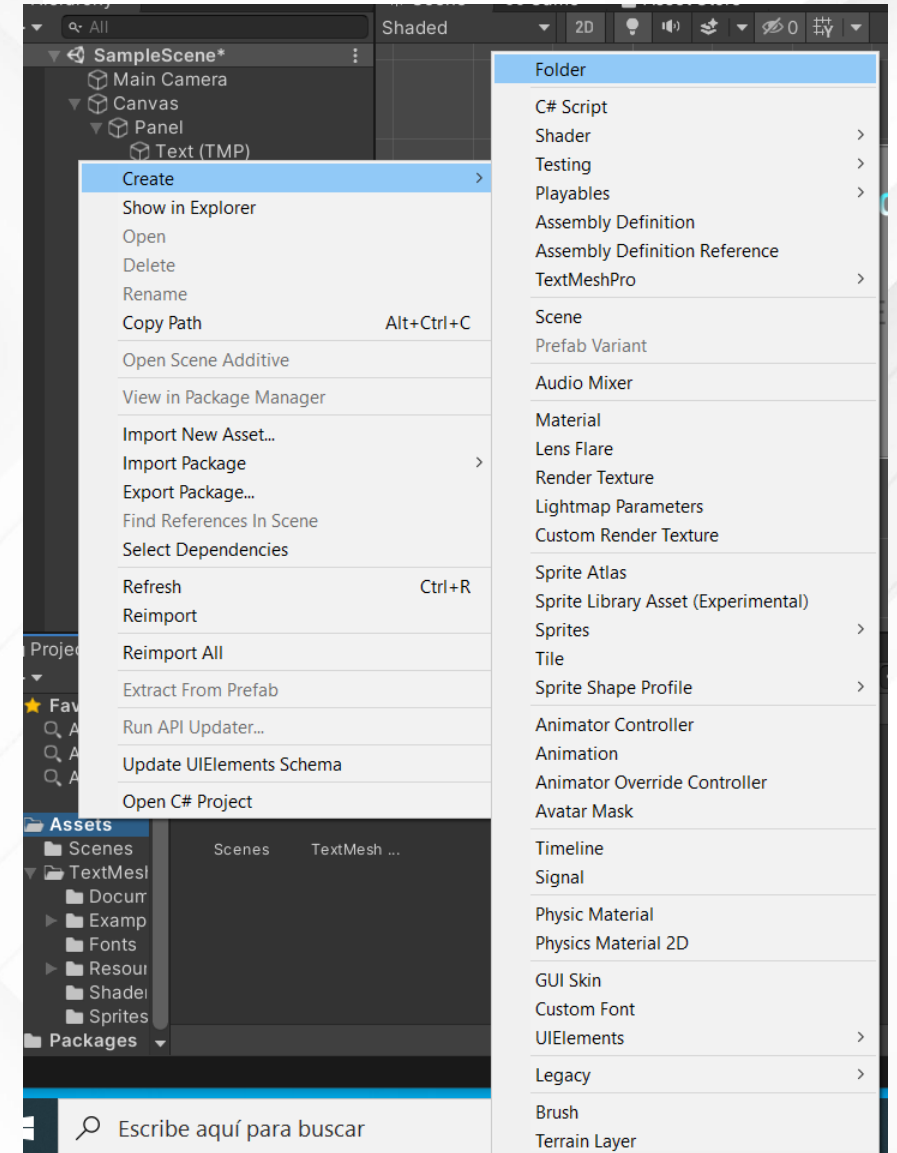
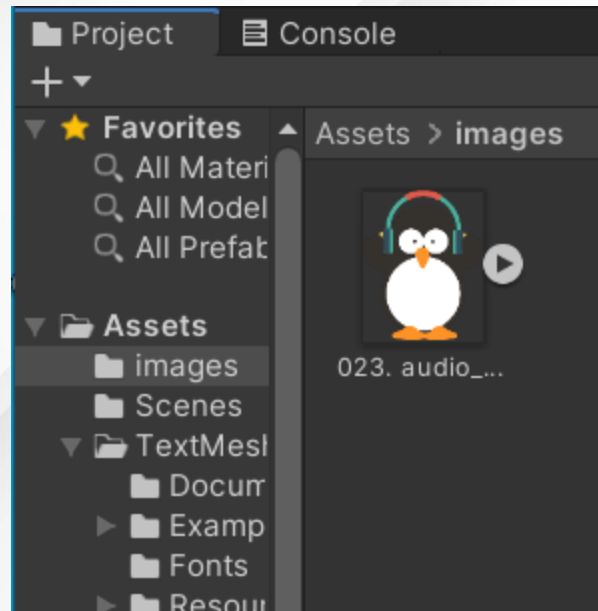
ENTRAR

Bienvenido a mi Juego

ENTRAR

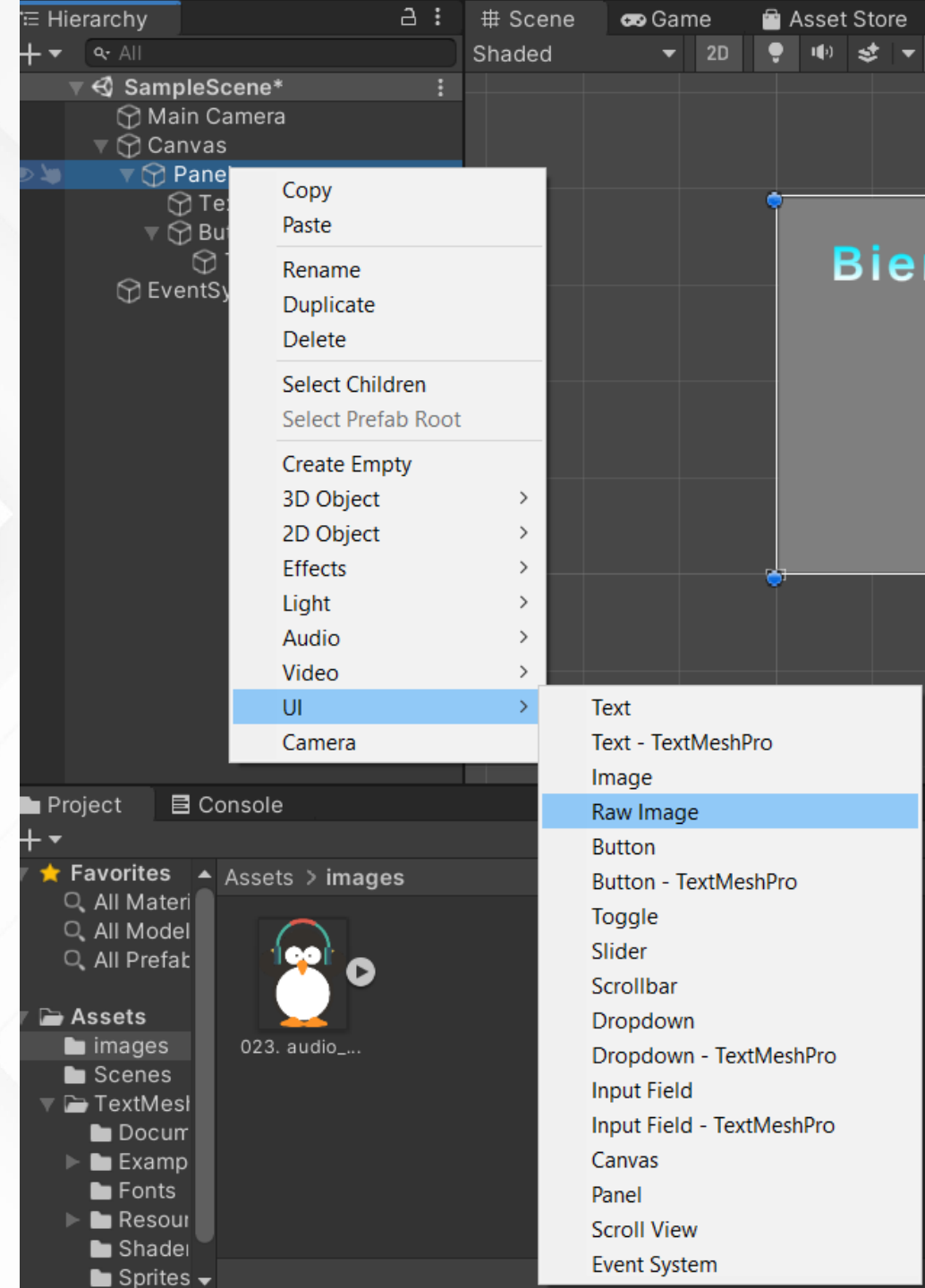
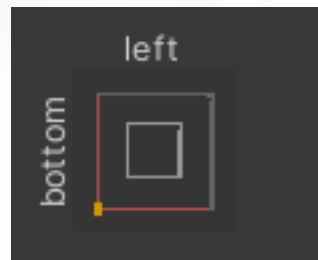
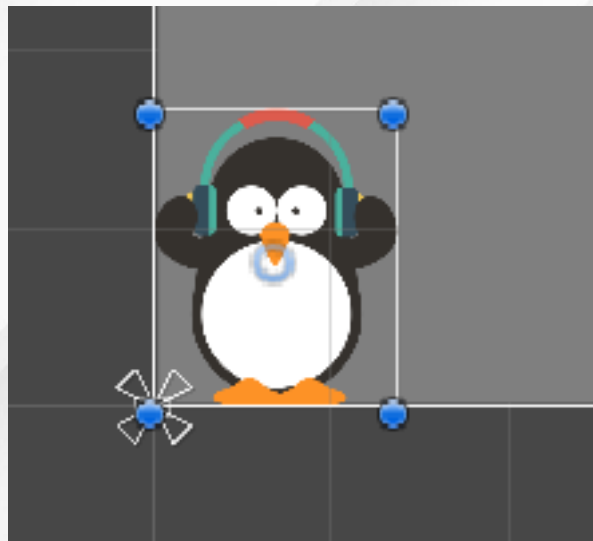
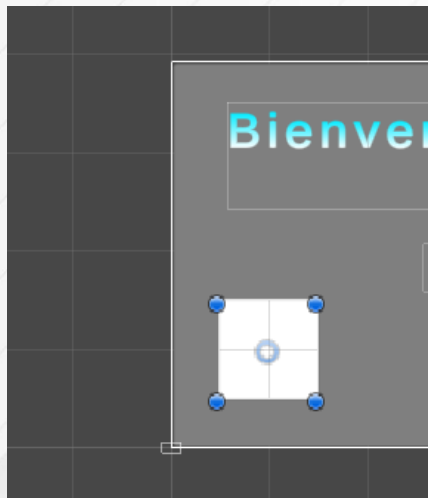
- PRIMER PROYECTO UNITY
- INTERFAZ GRÁFICA

- Insertar una imagen
- Arrastramos la imagen al directorio images que hemos creado en Assets



## > UNITY

- PRIMER PROYECTO UNITY
- INTERFAZ GRÁFICA
  - Insertamos en el panel una imagen de tipo Raw Image
  - Arrastramos la imagen al marco que acabamos de añadir en la barra de propiedades – Texture
  - Lo anclamos a la esquina inferior izquierda







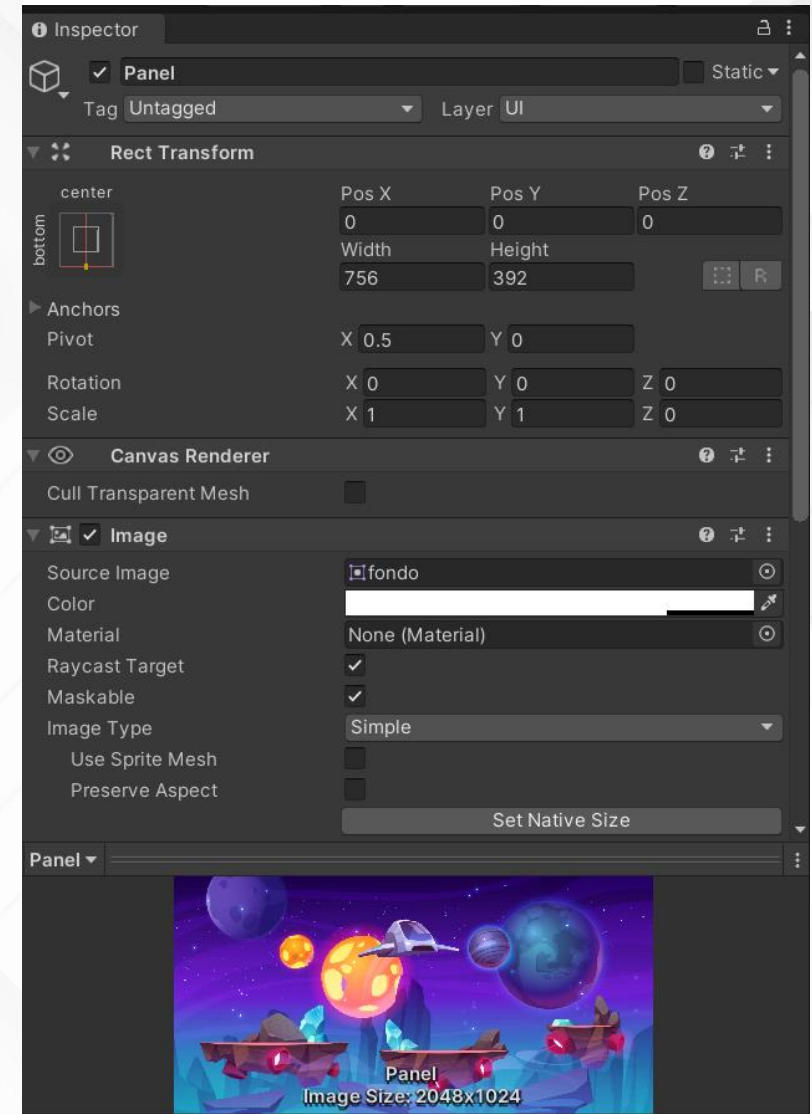
- PRIMER PROYECTO UNITY
- INTERFAZ GRÁFICA
  - Insertar imagen de fondo
  - Bancos de imágenes gratuitos
  - Pixabay.com
  - Pexels.com
  - Stokpic.com
  - Lifeofpix.com
  - Freepik.es

## > UNITY

- PRIMER PROYECTO UNITY
- INTERFAZ GRÁFICA
  - Insertar imagen de fondo
  - Guardamos imagen de fondo en images
  - Arrastramos imagen a propiedad Background

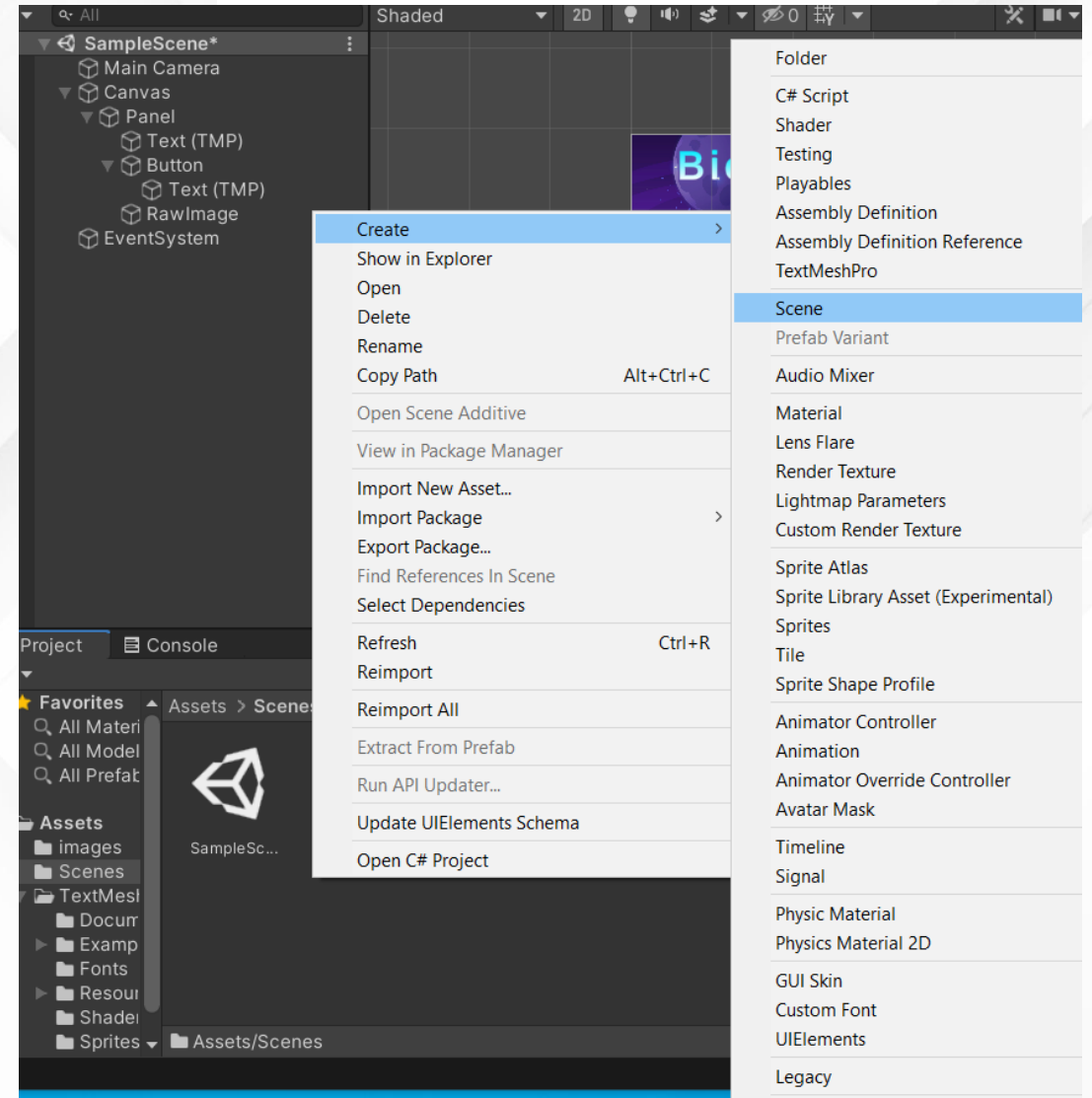
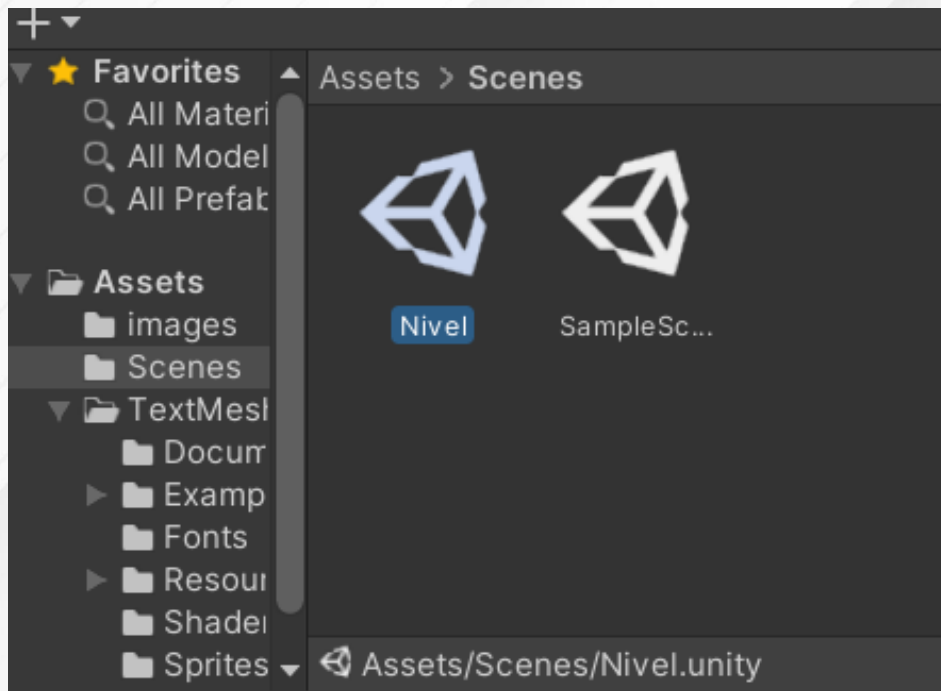


ILERNA  
Online

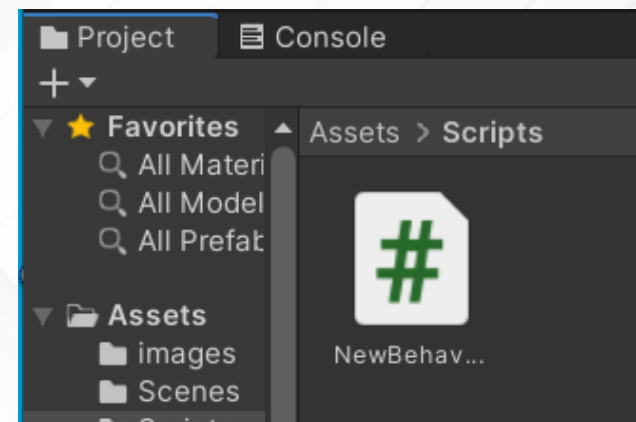
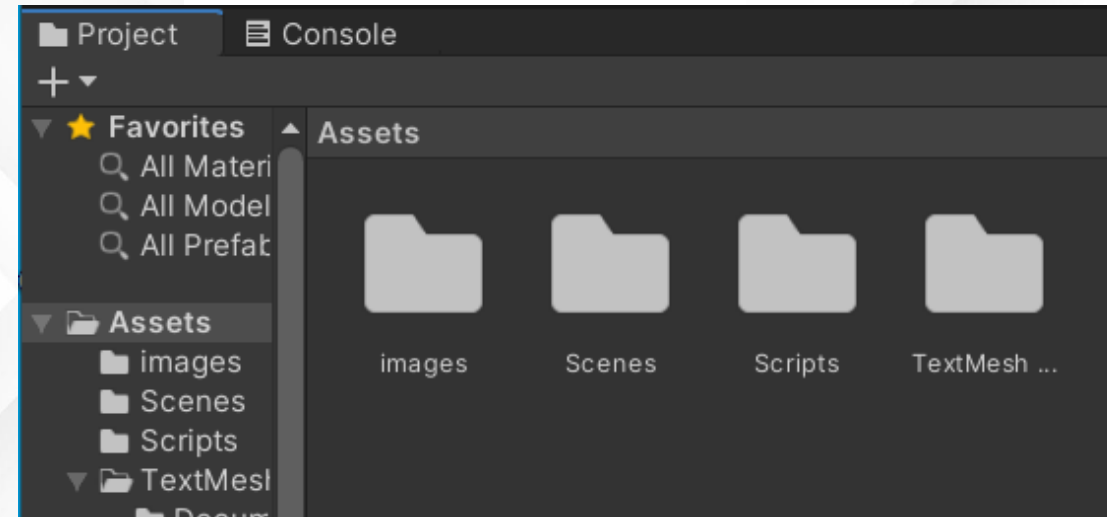




- PRIMER PROYECTO UNITY
- CONFIGURAR BOTÓN
  - Añadimos una nueva Scene en el Proyecto



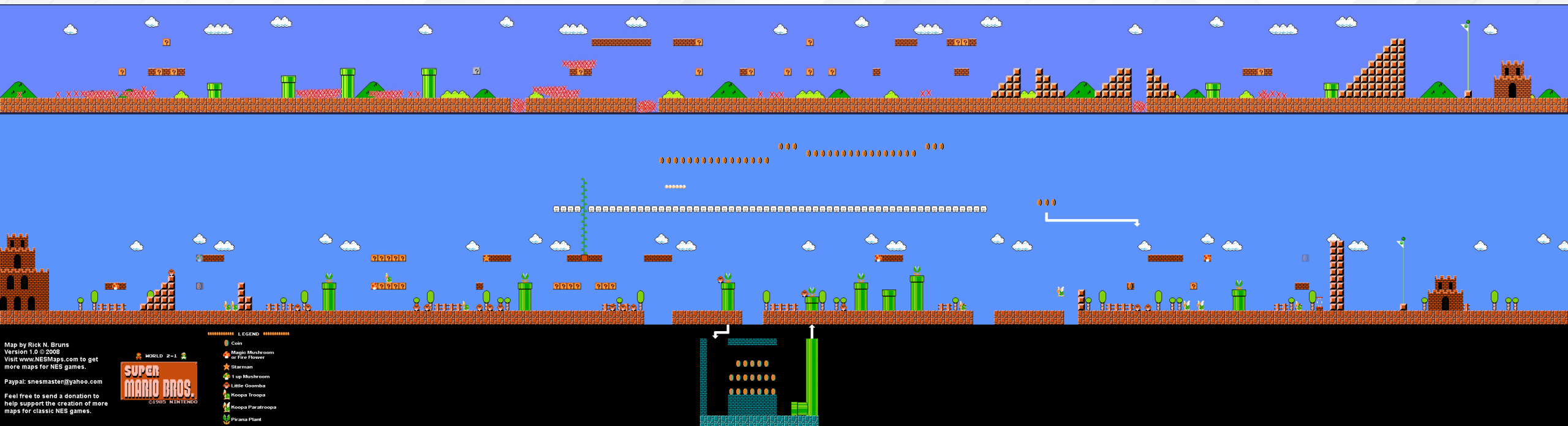
- PRIMER PROYECTO UNITY
- CONFIGURAR BOTÓN
  - Añadimos una nueva Scene en el Proyecto
  - Directorio Scripts en Assets
  - Dentro, nuevo Fichero C#, con nombre de la clase que vamos a crear





- PRIMER PROYECTO UNITY
- CONFIGURAR BOTÓN

- La escena puede contener un layout tan grande como queramos
- Desplazaremos la cámara para mostrar la parte que queramos







- PRIMER PROYECTO UNITY
- CONFIGURAR BOTÓN
  - Doble clic en el fichero y se abrirá
    - Visual Studio, MonoDB, etc
  - Cargaremos la nueva escena en la función startGame()
  - Usaremos las librerías que ya están definidas para Unity
  - SceneManagement

```
Imported Object
# ControladorMenu
Assembly Information
Filename Assembly-CSharp.dll
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class NewBehaviourScript : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {
    }

    // Update is called once per frame
    void Update()
    {
    }
}
```





- PRIMER PROYECTO UNITY
- CONFIGURAR BOTÓN
  - Doble clic en el fichero y se abrirá
    - Visual Studio VS Code, MonoDB, etc
  - Cargaremos la nueva escena en la función startGame()
  - Usaremos las librerías que ya están definidas para Unity
  - SceneManagement
  - Métodos Start() y Update() se verán más adelante
  - Lo sustituiremos por la función

```
File Edit Selection View Go Run Terminal Help ControladorMenu.cs - VT_ANDROID - Visual Studio C
EXPLORER
> OPEN EDITORS
VT_ANDROID
  Assets
  Scenes
  Scripts
    ControladorMenu.cs
    ControladorMenu.cs.meta
  TextMesh Pro
  images.meta
  Scenes.meta
  Scripts.meta
  TextMesh Pro.meta
  Library
  Logs
  Packages
  ProjectSettings
  Temp
Assets > Scripts > ControladorMenu.cs
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class NewBehaviourScript : MonoBehaviour
6 {
7     // Start is called before the first frame updat
8     void Start()
9     {
10
11     }
12
13     // Update is called once per frame
14     void Update()
15     {
16
17     }
18 }
19
```

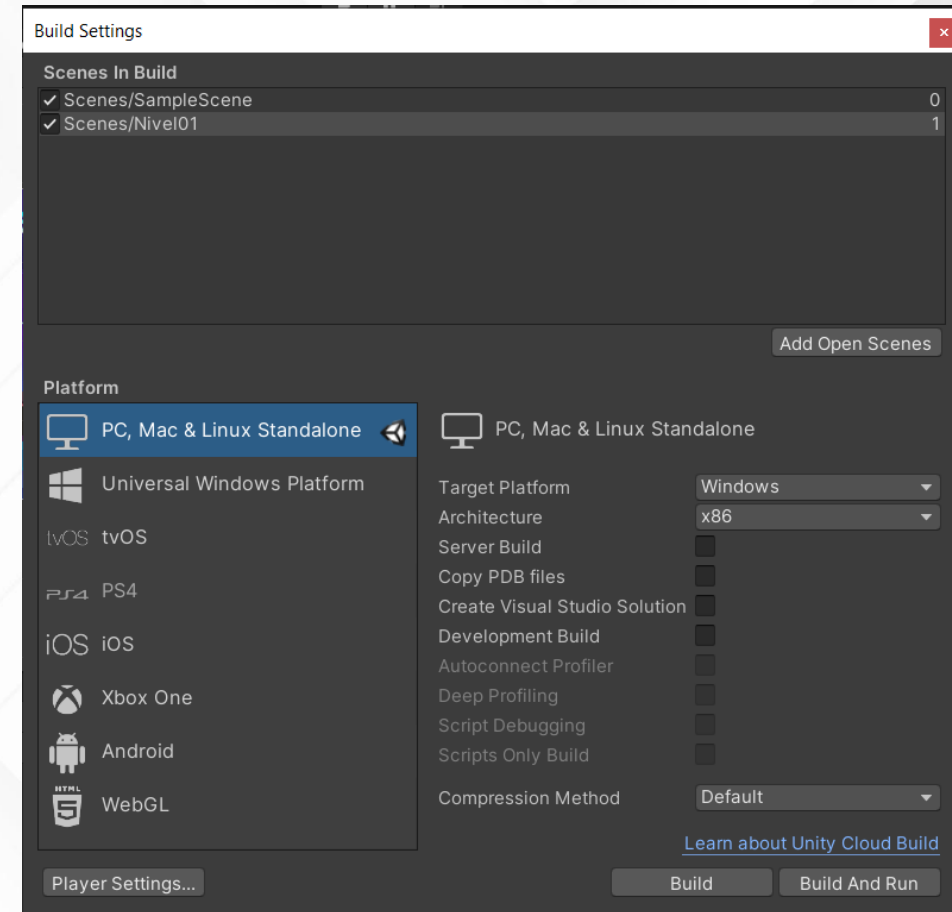
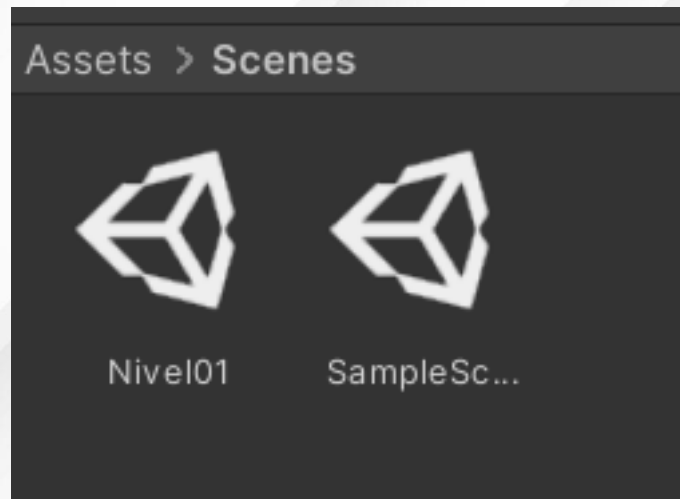
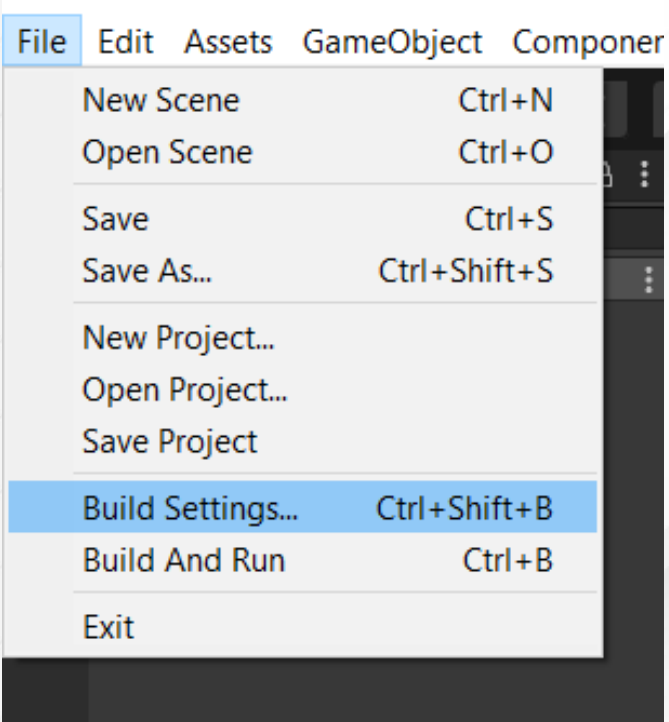


- PRIMER PROYECTO UNITY
- CONFIGURAR BOTÓN
  - Doble clic en el fichero y se abrirá
    - Visual Studio VS Code, MonoDB, etc
  - Cargaremos la nueva escena en la función startGame()
  - Usaremos las librerías que ya están definidas para Unity
  - SceneManager
  - Métodos Start() y Update() se verán más adelante
  - Lo sustituiremos por la función startGame()

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.SceneManagement;
5
6  public class ControladorMenu : MonoBehaviour
7  {
8      public void startGame()
9      {
10         //Cargaremos la escena del inicio del juego
11         SceneManager.LoadScene("Nivel01");
12     }
13 }
14
15
```

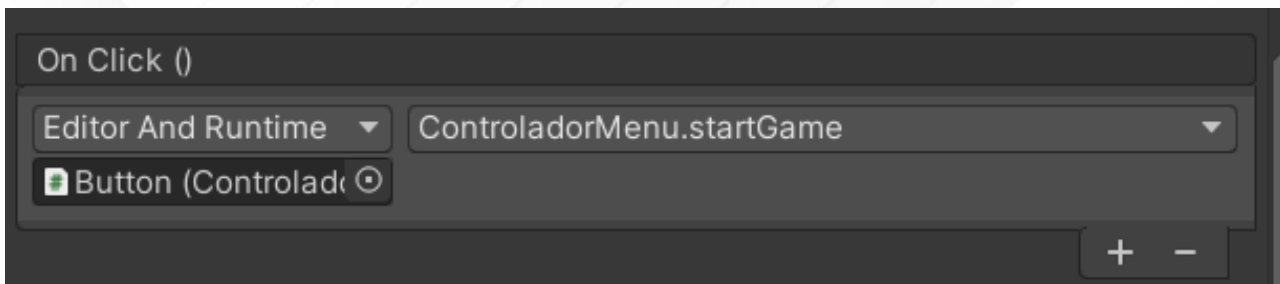
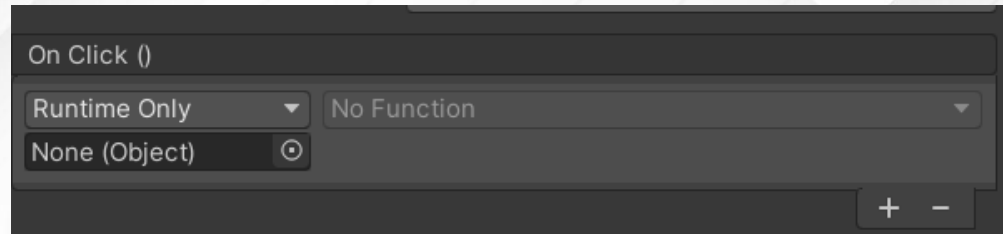
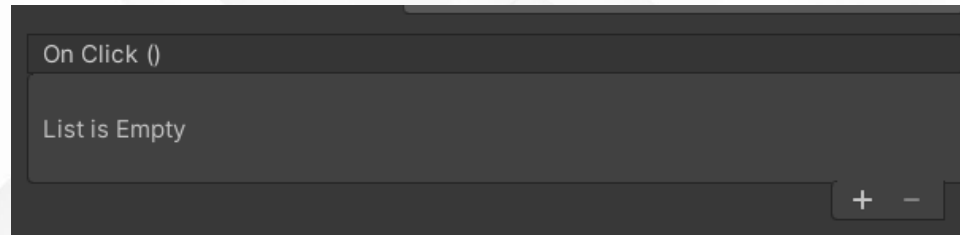
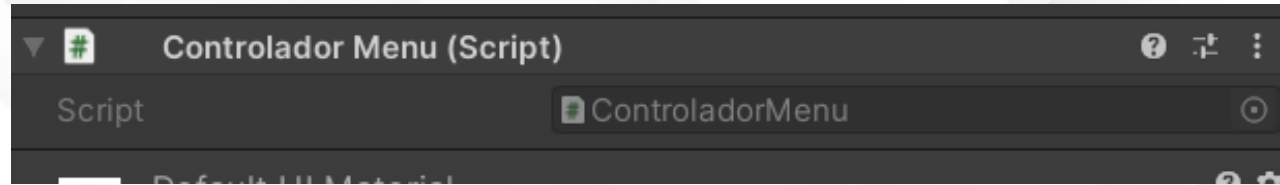
## > UNITY

- PRIMER PROYECTO UNITY
- CONFIGURAR BOTÓN
  - Añadir la nueva escena al proyecto





- PRIMER PROYECTO UNITY
- CONFIGURAR BOTÓN
  - Vincular el Script a la escena; arrastramos el script al botón en Hierchary y se añadirá el bloque en el panel de propiedades
  - En el apartado on Click del botón configuramos el comportamiento en +
  - Runtime: Editor & Runtime
  - Object: Arrastramos el botón
  - Function: ControladorGame / startGame





- PRIMER PROYECTO UNITY
- DESCARGAS DE ASSETSTORE

The screenshot shows the Unity Asset Store website interface. At the top, there is a navigation bar with the Unity Asset Store logo and menu items: Assets, Tools, Services, By Unity, Industries, and Sale. A search bar is located below the navigation bar with the placeholder text "Search for assets". Below the search bar, there are three statistics: "Over 11,000 5 star assets", "Rated by 85,000+ customers", and "Supported by over 100,000 forum members". The main content area features a large banner for "CYBER WEEK SAVINGS" with the text "Best of Super Sale ends soon." and "Save 50% on 700+ best assets." A button labeled "See what's on sale" is positioned at the bottom left of the banner. The banner background is a colorful geometric pattern of triangles in shades of blue, purple, pink, and yellow.







- PRIMER PROYECTO UNITY

You purchased this item on Nov 29, 2020.



### Sunny Land

Ansimuz

★★★★★ 5 | 198 Reviews

FREE

Open in Unity



Securely checkout with:

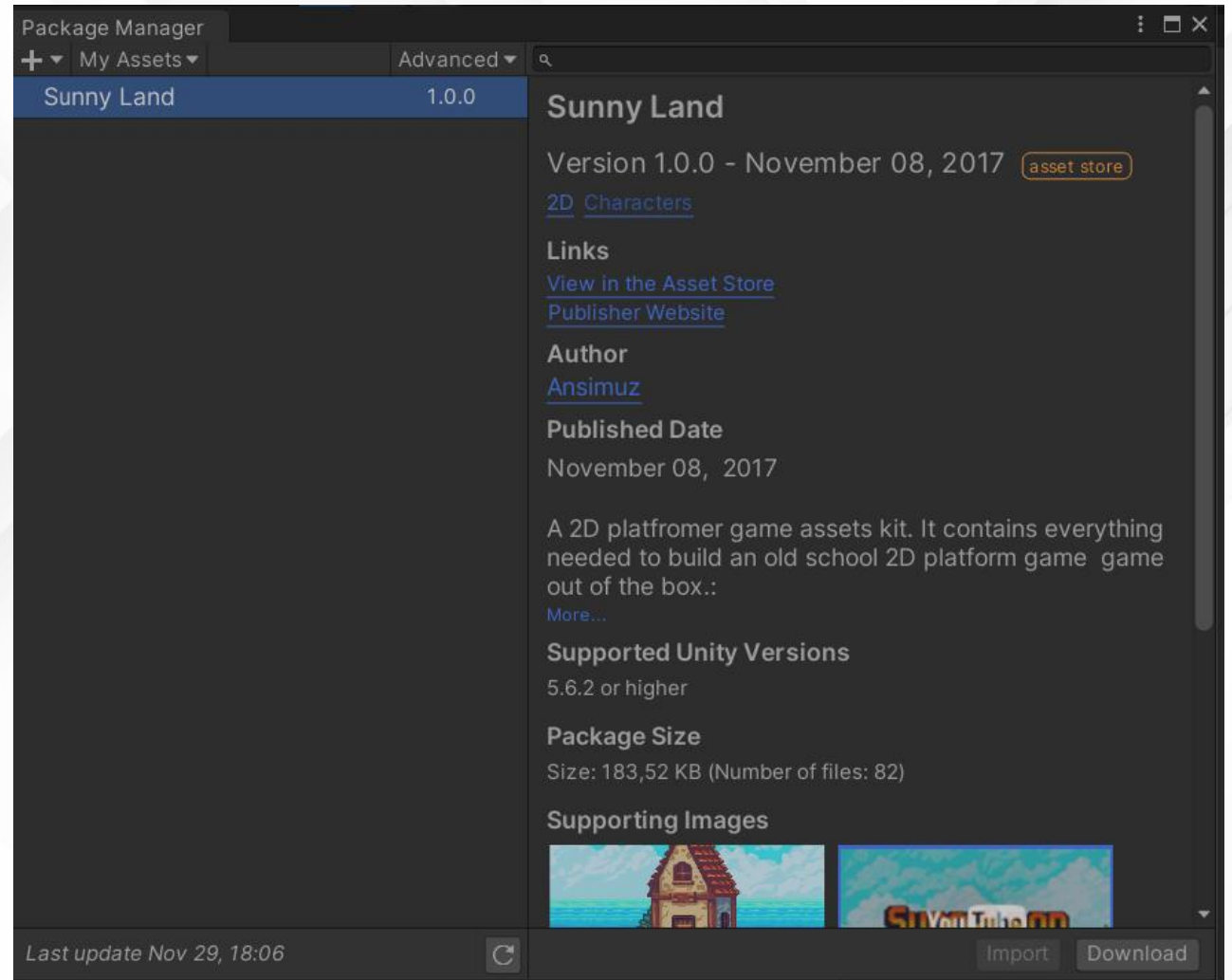
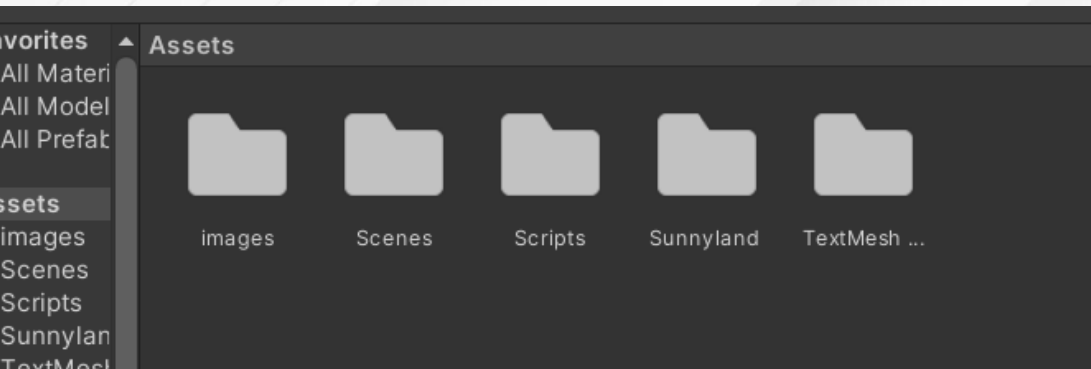


License	Extension Asset
File size	183.5 KB
Latest version	1.0
Latest release date	Nov 8, 2017
Supported Unity versions	5.6.2 or higher

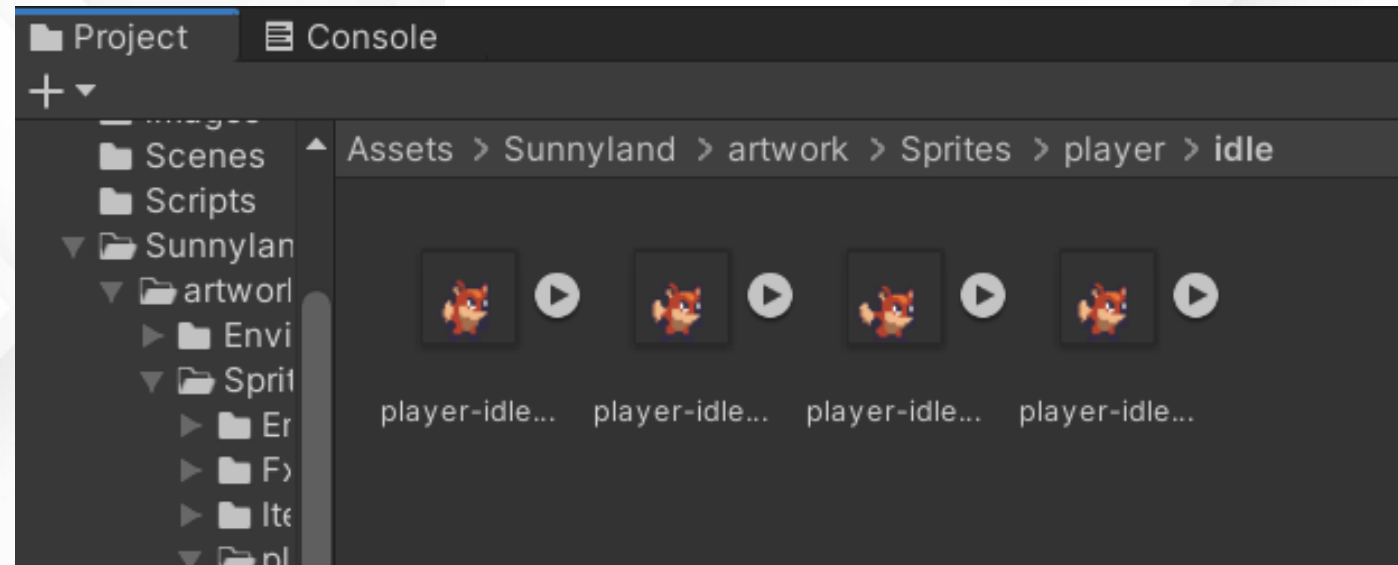




- PRIMER PROYECTO UNITY
- **DESCARGAS DE ASSETSTORE**
  - En la ventana emergente, confirmamos la descarga en nuestro proyecto de Unity
  - Luego lo importamos en Import (recuerda no tener activado el play)
  - Se creará una nueva carpeta en Assets

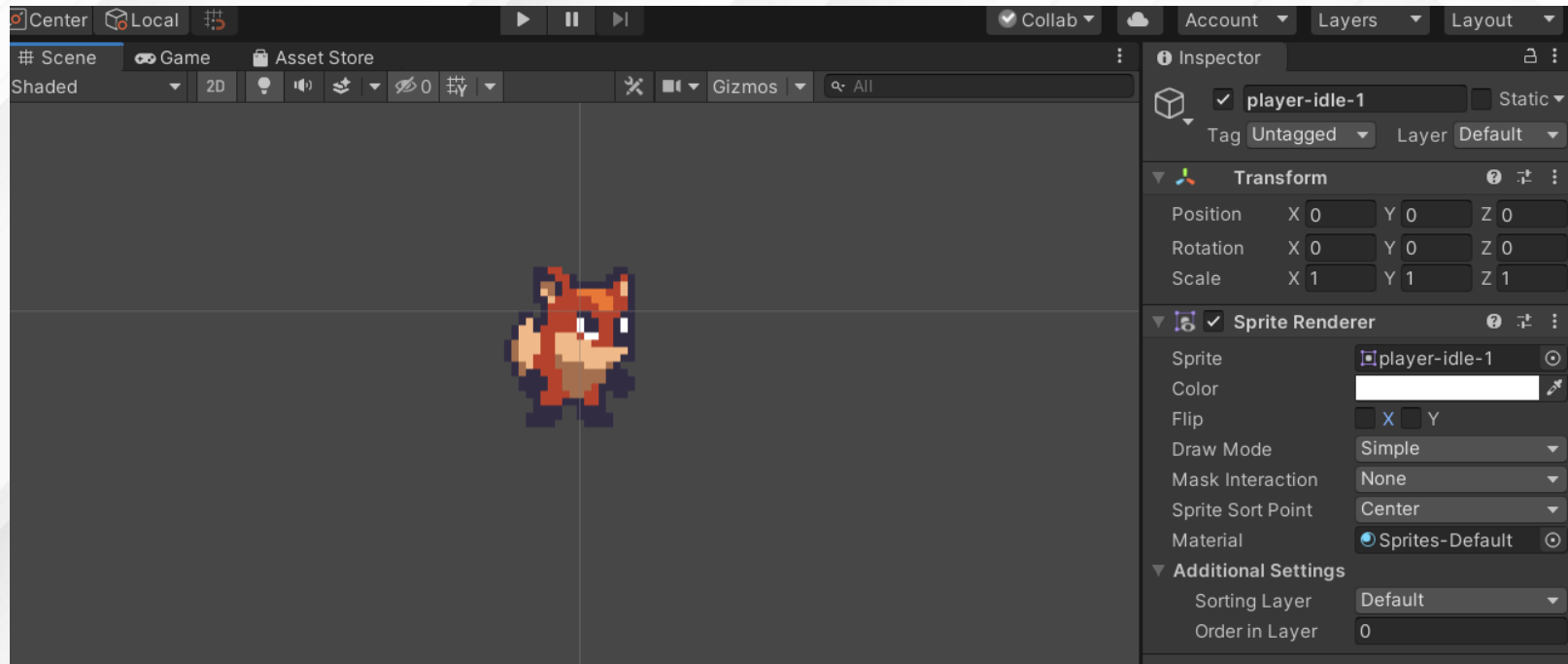


- PRIMER PROYECTO UNITY
- ELECCIÓN DEL PERSONAJE
  - Carpeta Assets / Sunnyland / artwork / Sprites/player
  - En esta carpeta tenemos las diferentes animaciones que se incluyen para el personaje
  - IDLE: personaje quieto
  - Seleccionamos la primera imagen y la arrastramos a la pantalla
  - Se crea un GameObject player-idle-1
  - Propiedades:
  - Transform y Sprite Render



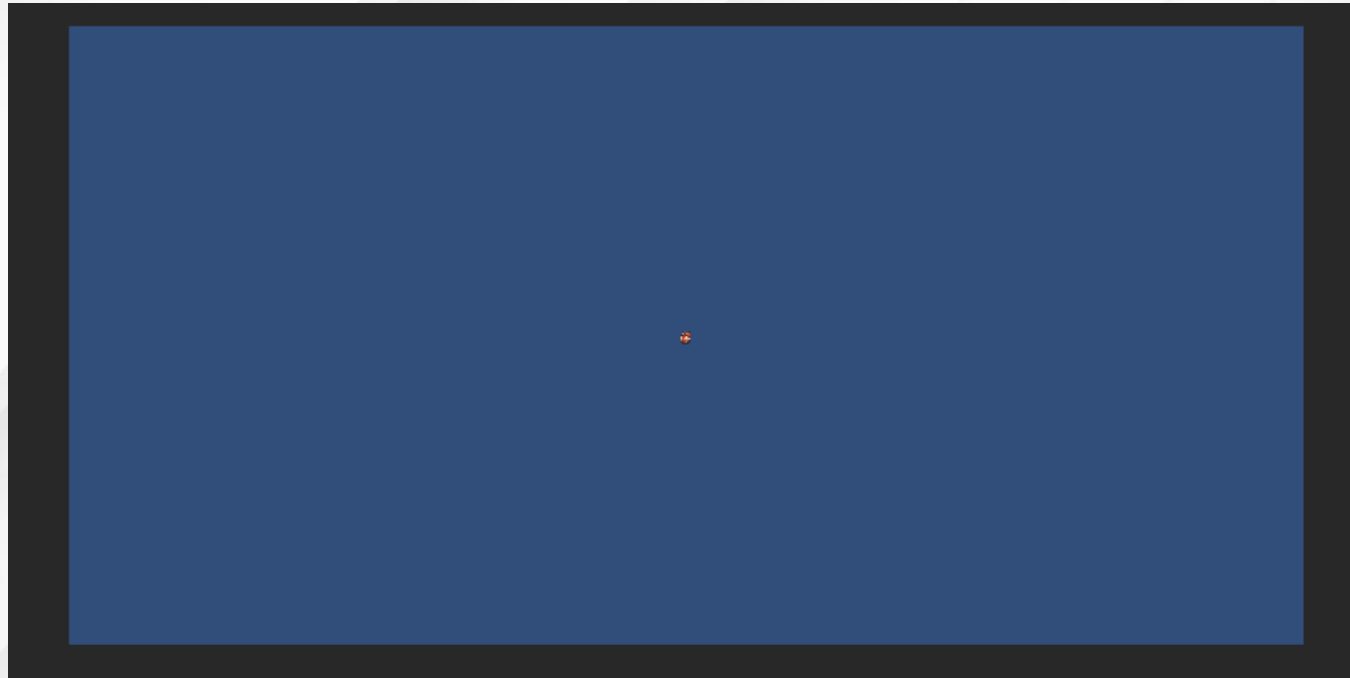


- PRIMER PROYECTO UNITY
- ELECCIÓN DEL PERSONAJE
  - Transform: para darle posición, rotación y escala. Inicialmente ponerlo en posición 0, 0, 0 (en el centro de la pantalla)
  - Sprite Renderer: pinta el Sprite en el juego



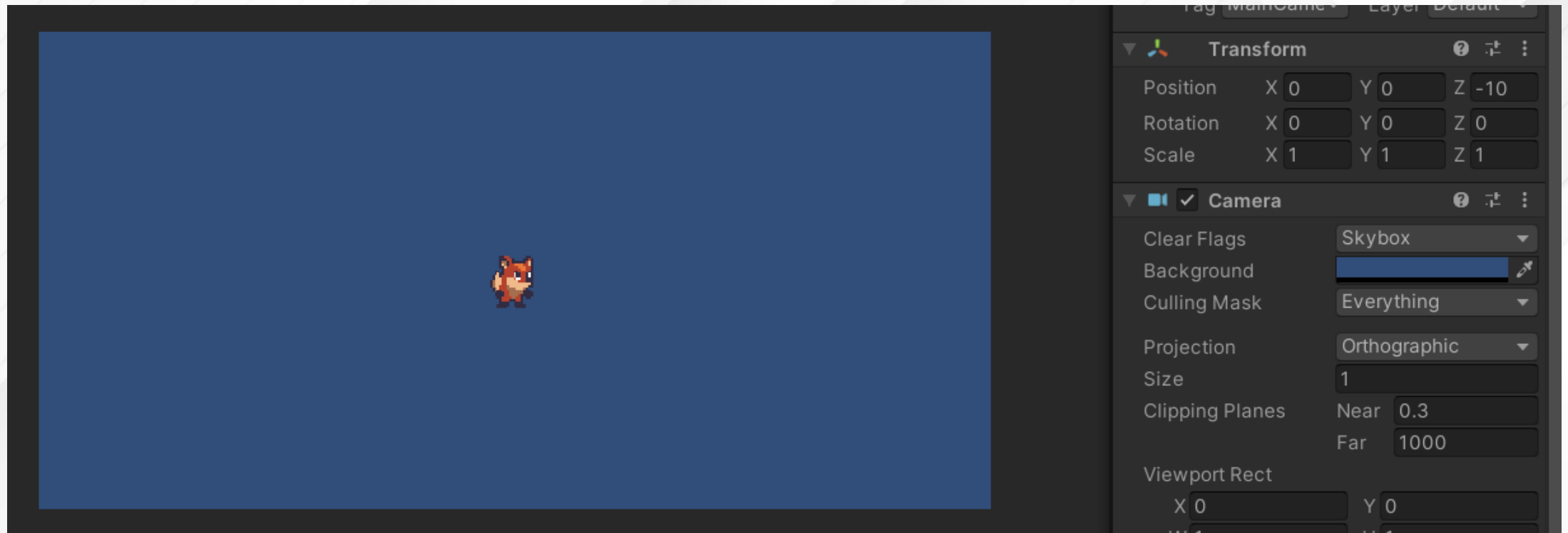


- PRIMER PROYECTO UNITY
- ELECCIÓN DEL PERSONAJE
  - Si pulsamos en PLAY, nuestro personaje se verá muy pequeño
  - Cambiamos la configuración de la cámara y fijamos un tamaño (5) de 1



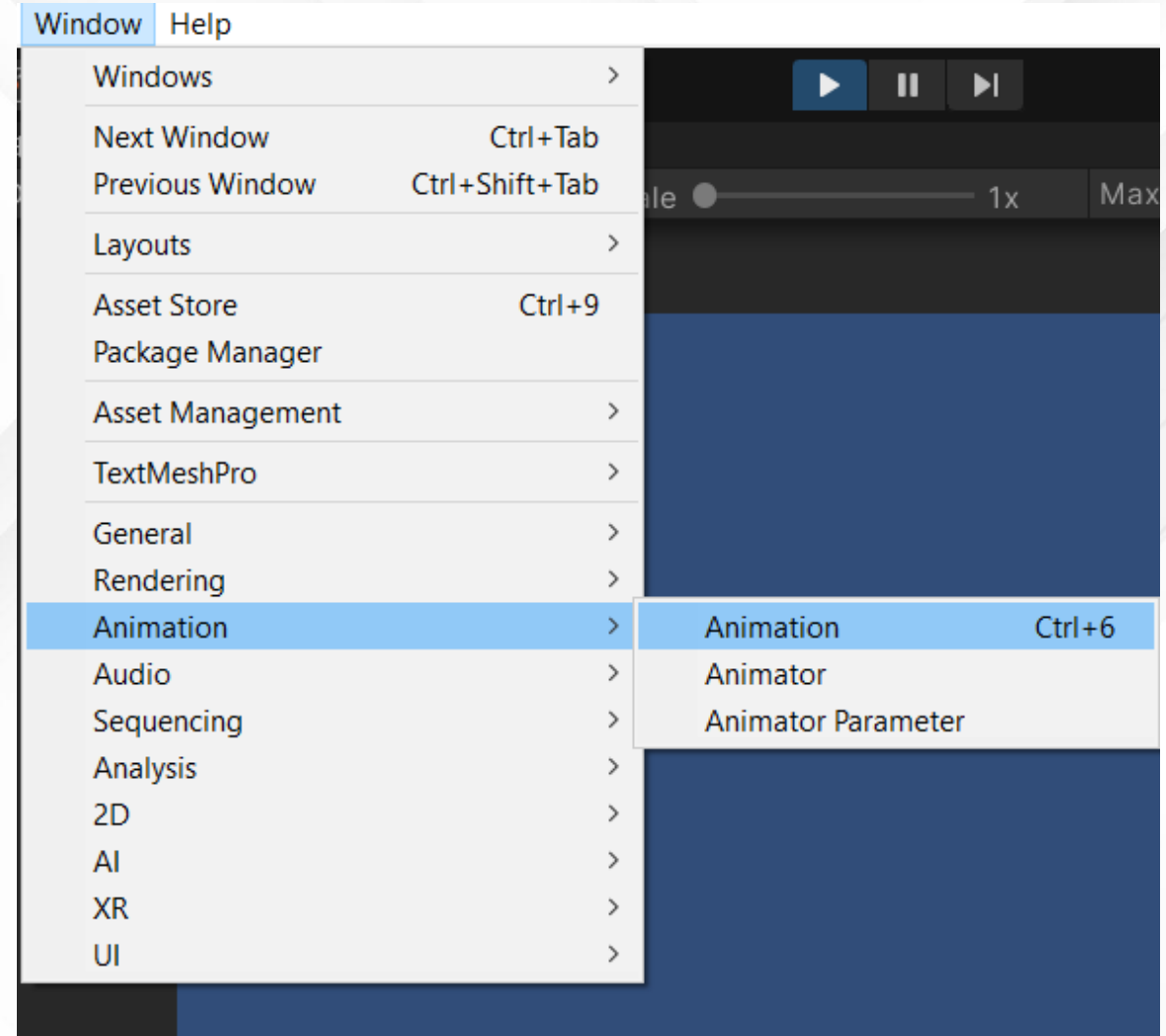


- PRIMER PROYECTO UNITY
- ELECCIÓN DEL PERSONAJE
  - Si pulsamos en PLAY, nuestro personaje se verá muy pequeño
  - Cambiamos la configuración de la cámara y fijamos un tamaño (5) de 1





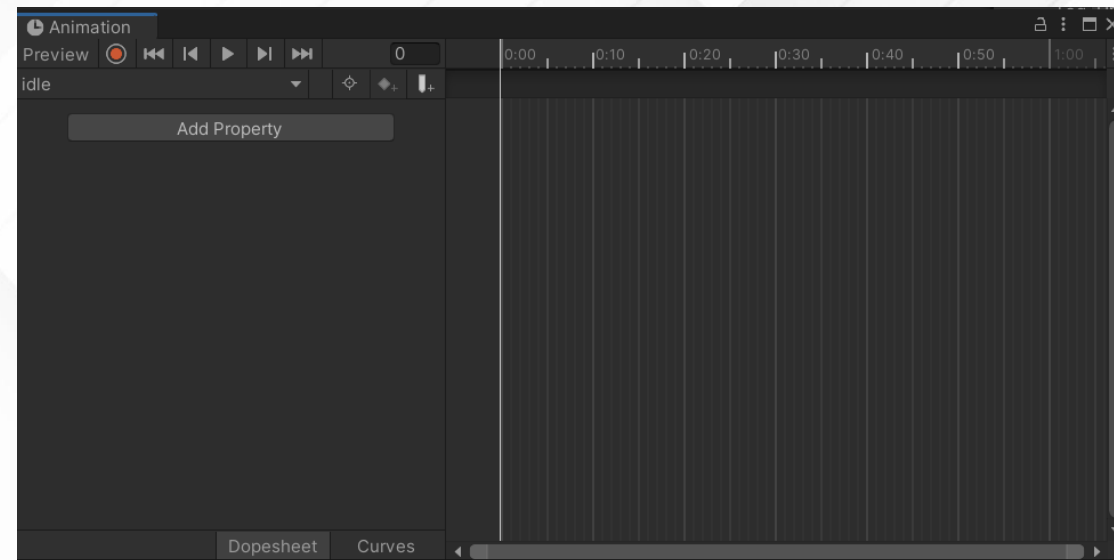
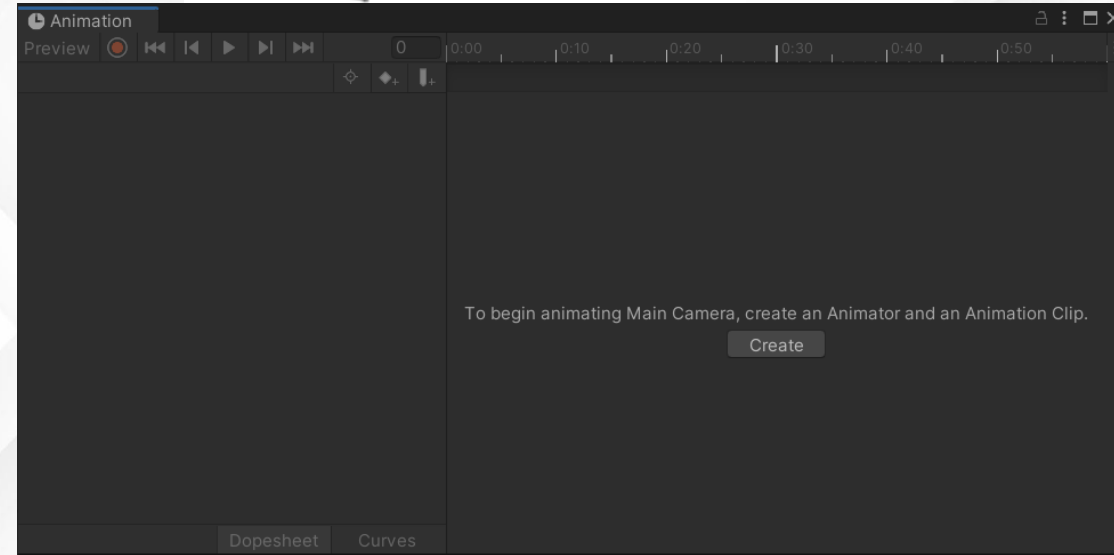
- PRIMER PROYECTO UNITY
- ANIMACIÓN DEL PERSONAJE
  - Cada x ms vamos a cambiar el Sprite del personaje para simular la animación
  - Unity proporciona un motor de animaciones
  - Window / Animation
    - Animation
    - Animator
    - Animator parameter







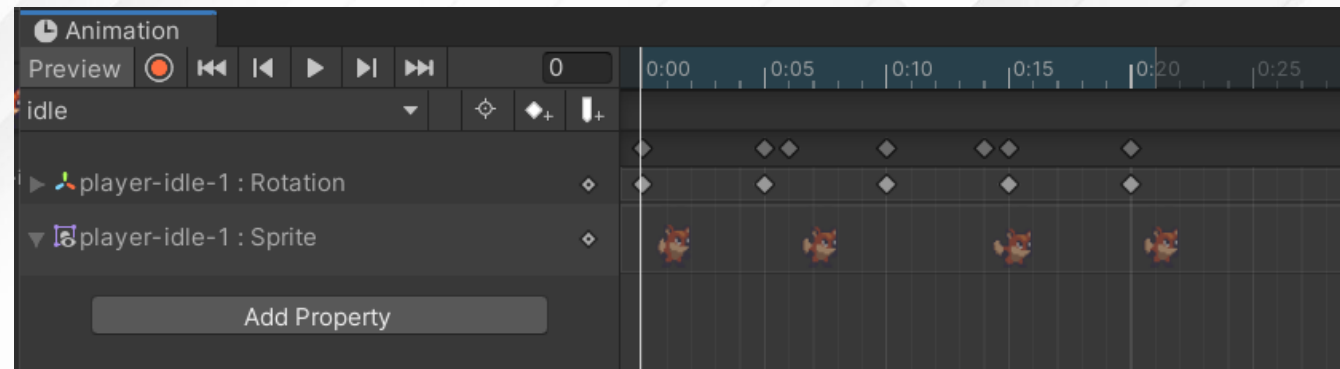
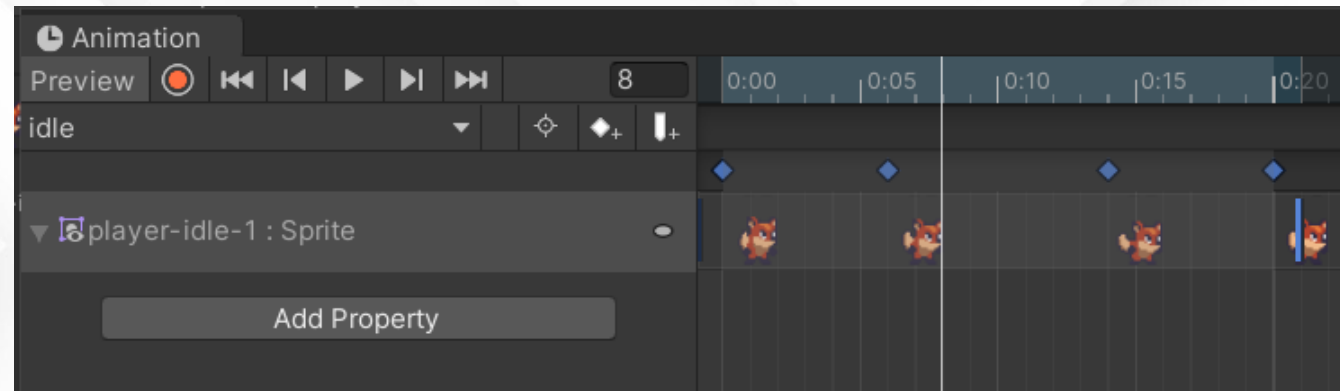
- PRIMER PROYECTO UNITY
- ANIMACIÓN DEL PERSONAJE
  - 1. ANIMATION
  - Creamos una carpeta en Assets llamada Animations y dentro de ella creamos otra nueva llamada Player
  - Dentro de esta carpeta guardamos la animación con el nombre de idle
  - Arrastramos los Sprites de la carpeta idle que teníamos en Sunnysland a esta línea de tiempo



- PRIMER PROYECTO UNITY
- ANIMACIÓN DEL PERSONAJE

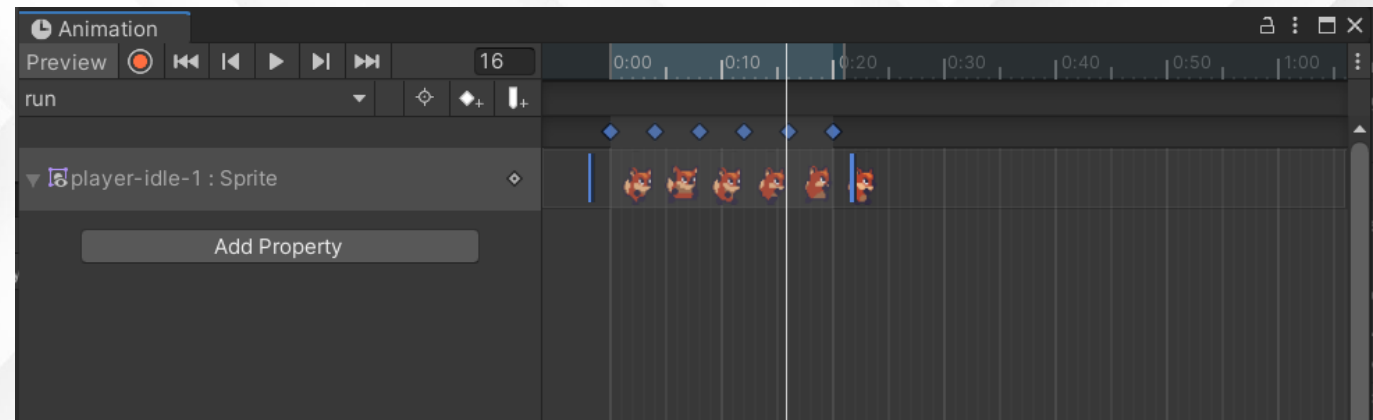
➤ 1. ANIMATION

- Para que la animación sea más lenta, las seleccionamos y arrastramos la última para más segundos hasta conseguir el efecto deseado.
- Además podemos hacer animaciones a nivel de transformación del Sprite con el botón de grabar
- Cambiaremos la rotación en el eje z en los fotogramas 5, 10, 15 y 20
- Añade animación de Rotation



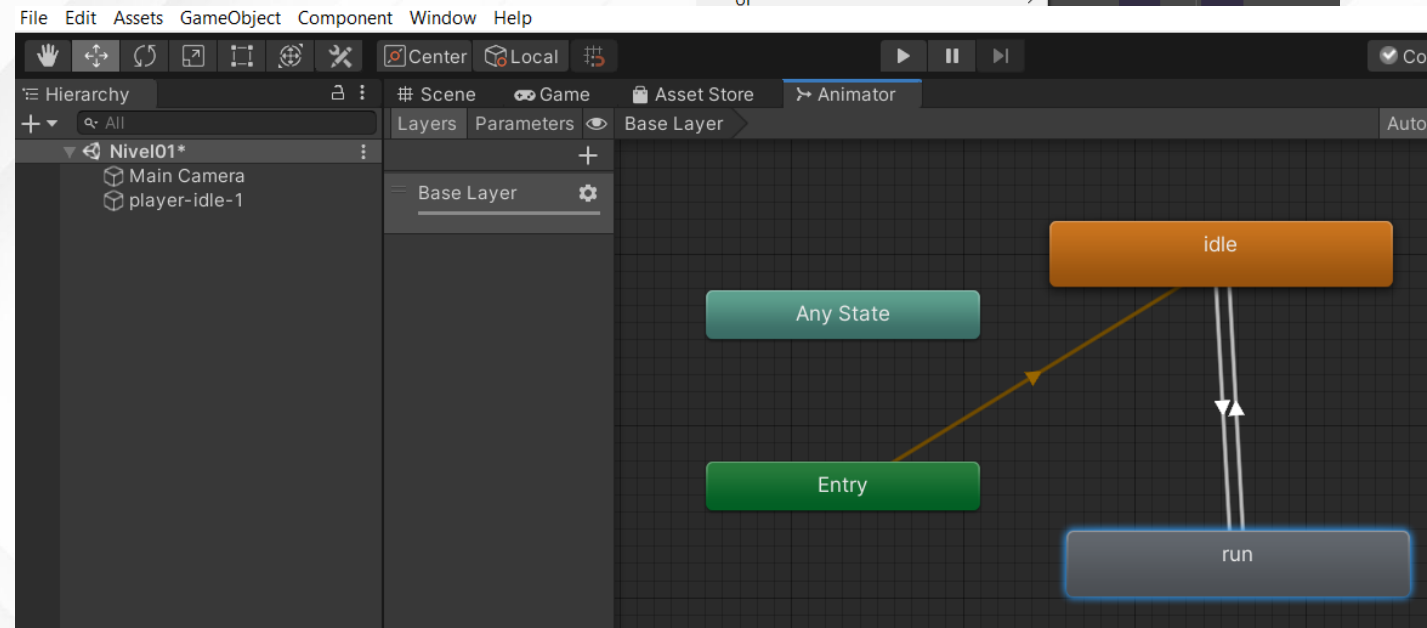
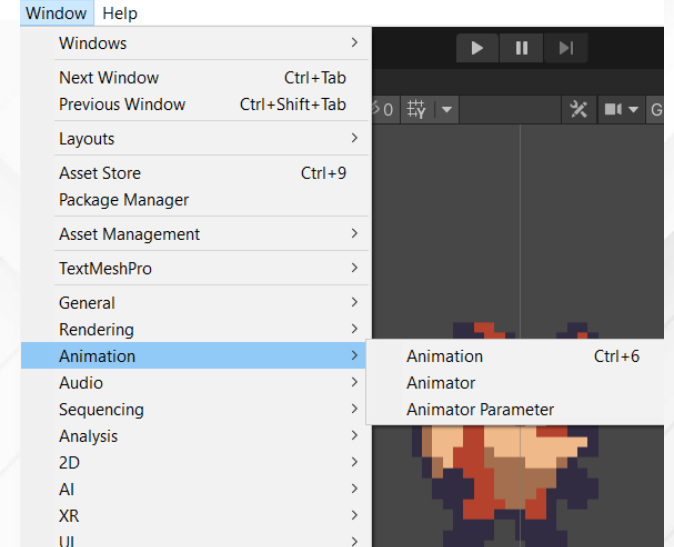


- PRIMER PROYECTO UNITY
- ANIMACIÓN DEL PERSONAJE
  - 1. ANIMATION
  - Creamos una segunda animación llamada run (correr)
  - Window/Animations/Animation/New clip/run
  - Añadimos los clips de Assets... run
  - Ajustamos el tiempo para estos fotogramas



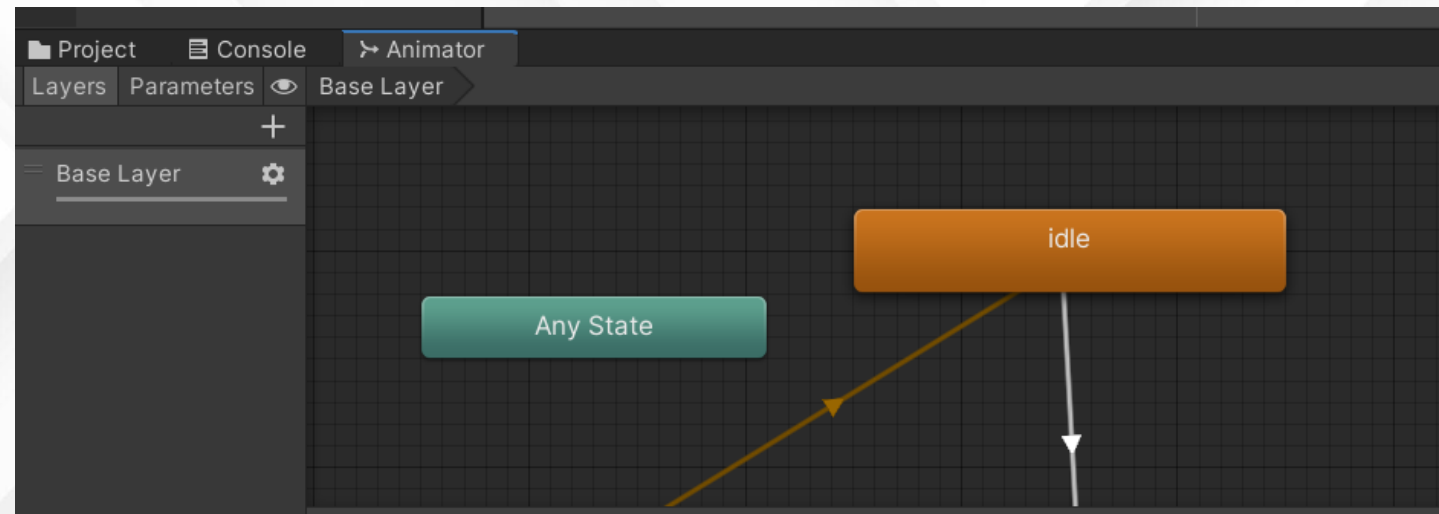


- PRIMER PROYECTO UNITY
- ANIMACIÓN DEL PERSONAJE
  - 2. ANIMATOR
  - Cargamos esta ventana y vamos a definir cómo serán las transiciones posibles.
  - Al entrar queremos que el personaje esté parado, que pueda ir a correr pero también que pueda volver a estar parado
  - Lo definimos añadiendo las flechas entre las cajas

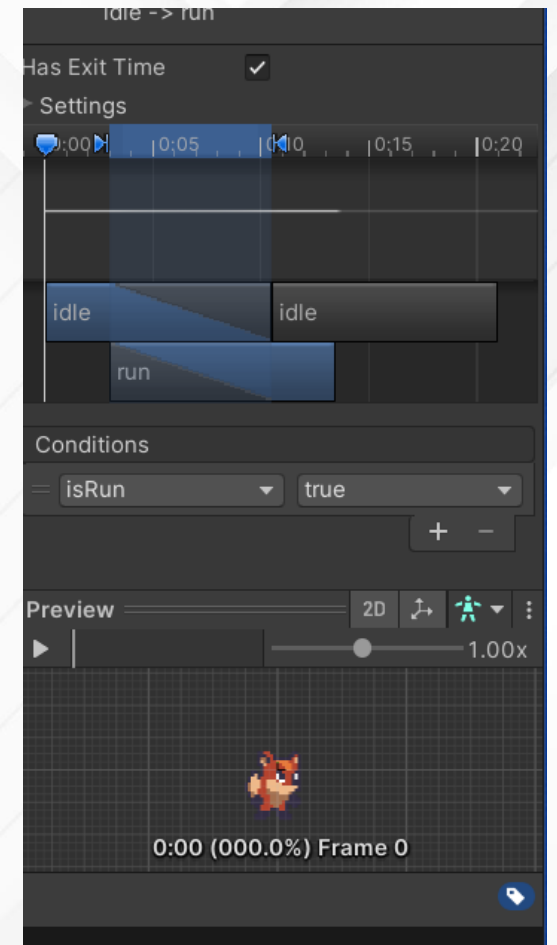
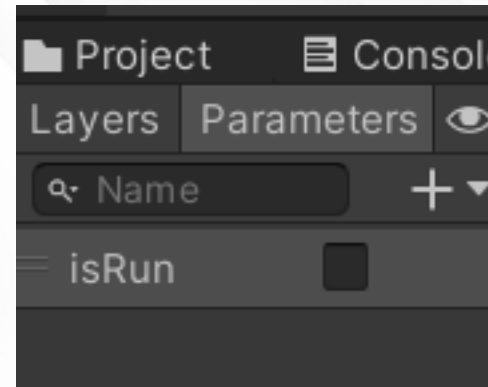




- PRIMER PROYECTO UNITY
- ANIMACIÓN DEL PERSONAJE
  - 2. ANIMATOR
  - Tendremos que configurar qué animación quiero cargar en cada momento y según lo que tenga que hacer el personaje
  - Arrastra la ventana de Animator abajo



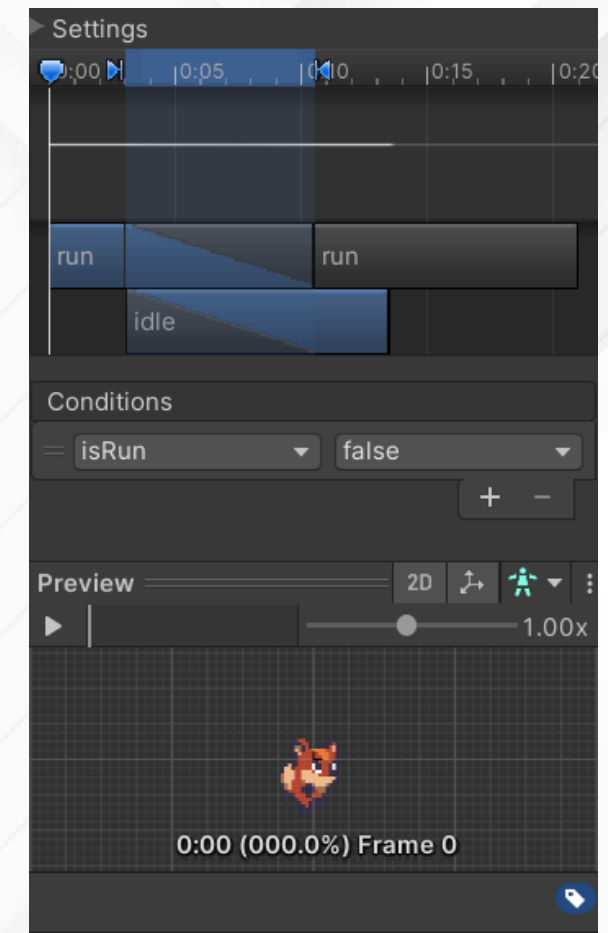
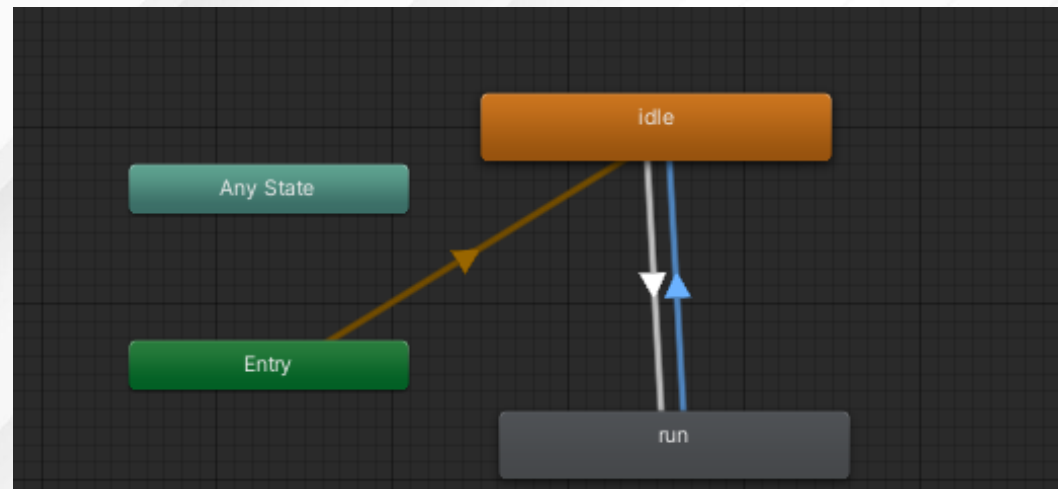
- PRIMER PROYECTO UNITY
- ANIMACIÓN DEL PERSONAJE
  - 2. ANIMATOR
  - Pestaña PARAMETER
  - Creamos variable BOOL isRun que inicialmente sea false
  - Clica la flecha de idle a run
  - En el panel lateral establecemos la condición para pasar de idle a r
  - un





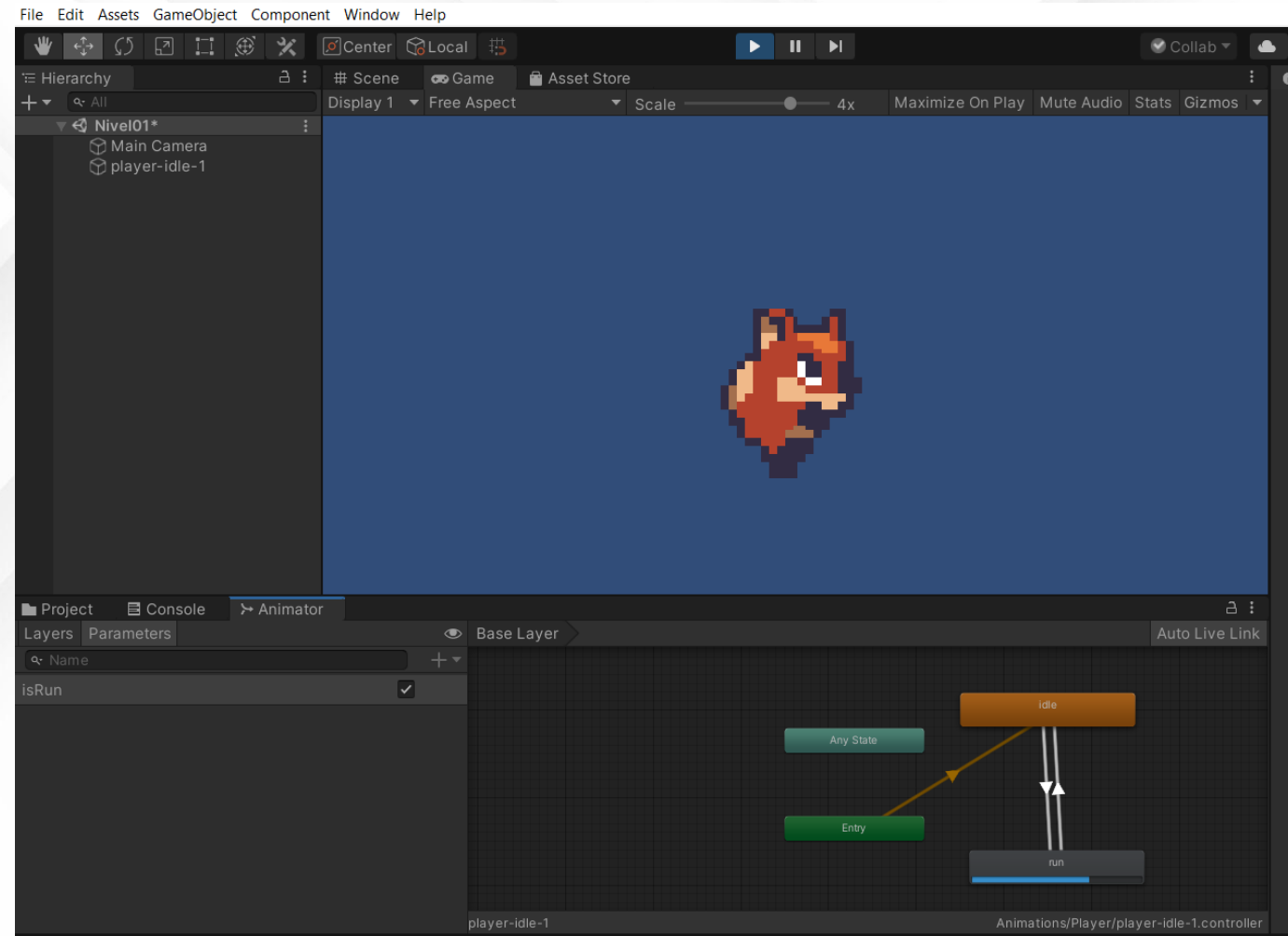


- PRIMER PROYECTO UNITY
- ANIMACIÓN DEL PERSONAJE
  - 2. ANIMATOR
  - Repetimos el proceso para la flecha de run a idle

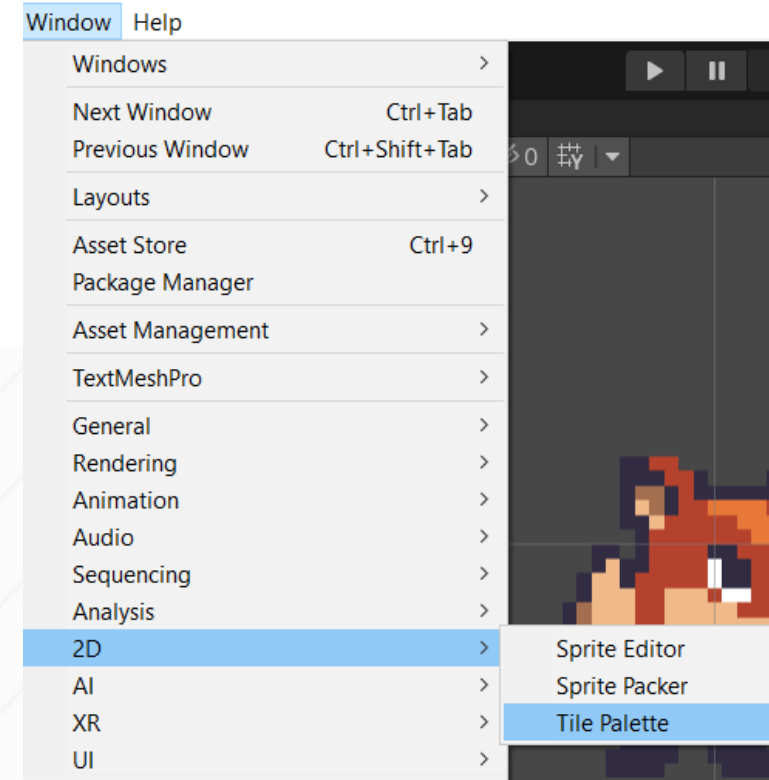
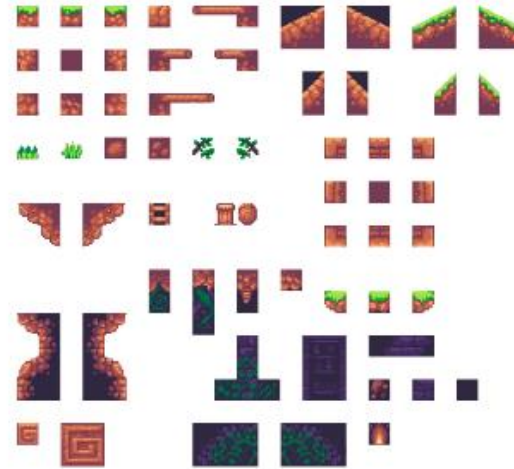




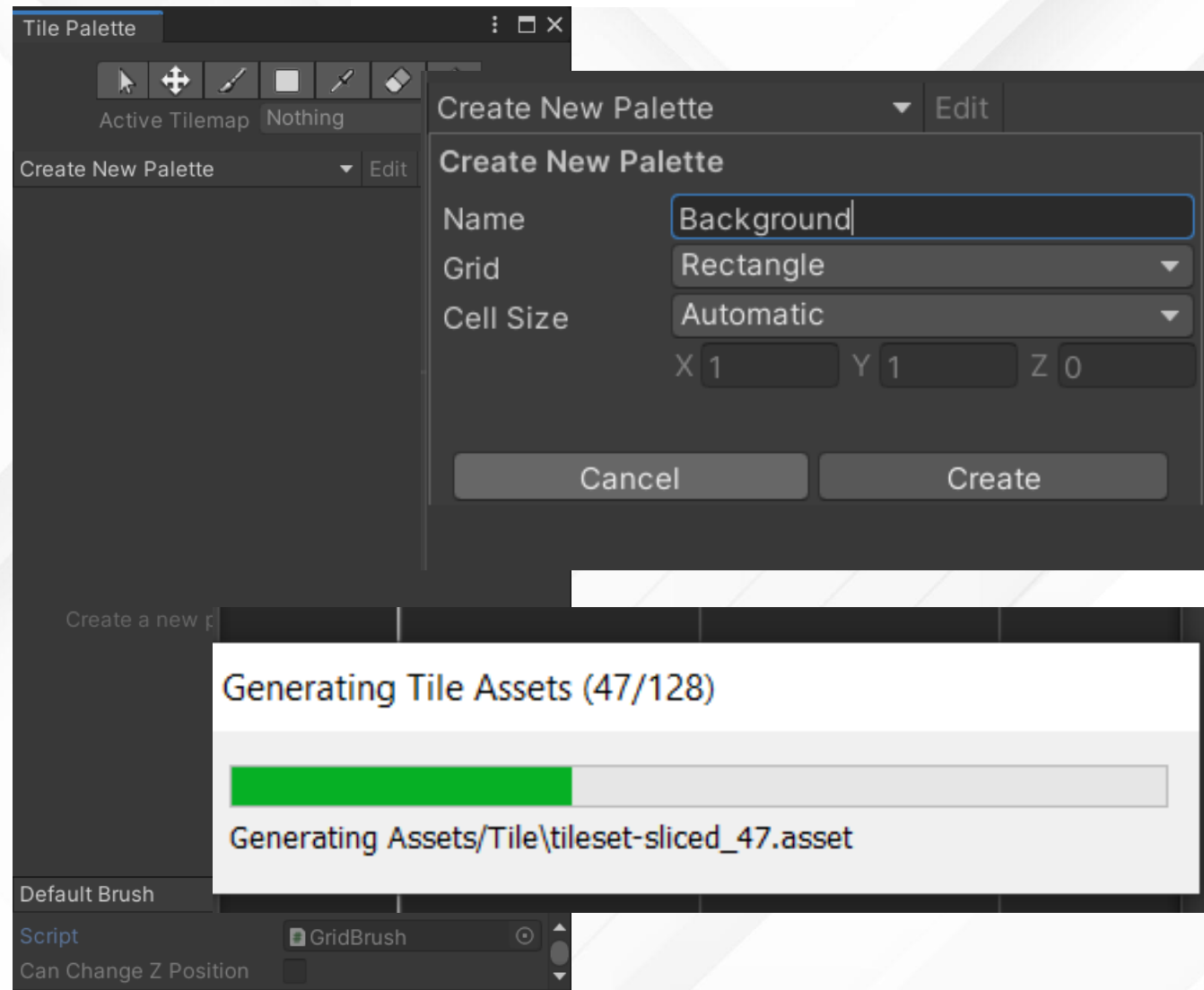
- PRIMER PROYECTO UNITY
- ANIMACIÓN DEL PERSONAJE
  - 2. ANIMATOR
  - Si ejecutamos el proyecto podemos ver que por defecto el personaje carga la animación idle, pero si marco la casilla de la variable isRun, cambia de animación.



- PRIMER PROYECTO UNITY
- CREAR UN ESCENARIO DE JUEGO
  - Vamos a usar los sprites que se han diseñado para SunnyIsland
  - Assets/Sunnyland / artwork/ Environment
  - Creamos el fondo en Window/2d/ Tile Palette: permite crear paleta a partir de una imagen
  - Vamos a pintar el escenario como si lo estuviéramos pintando

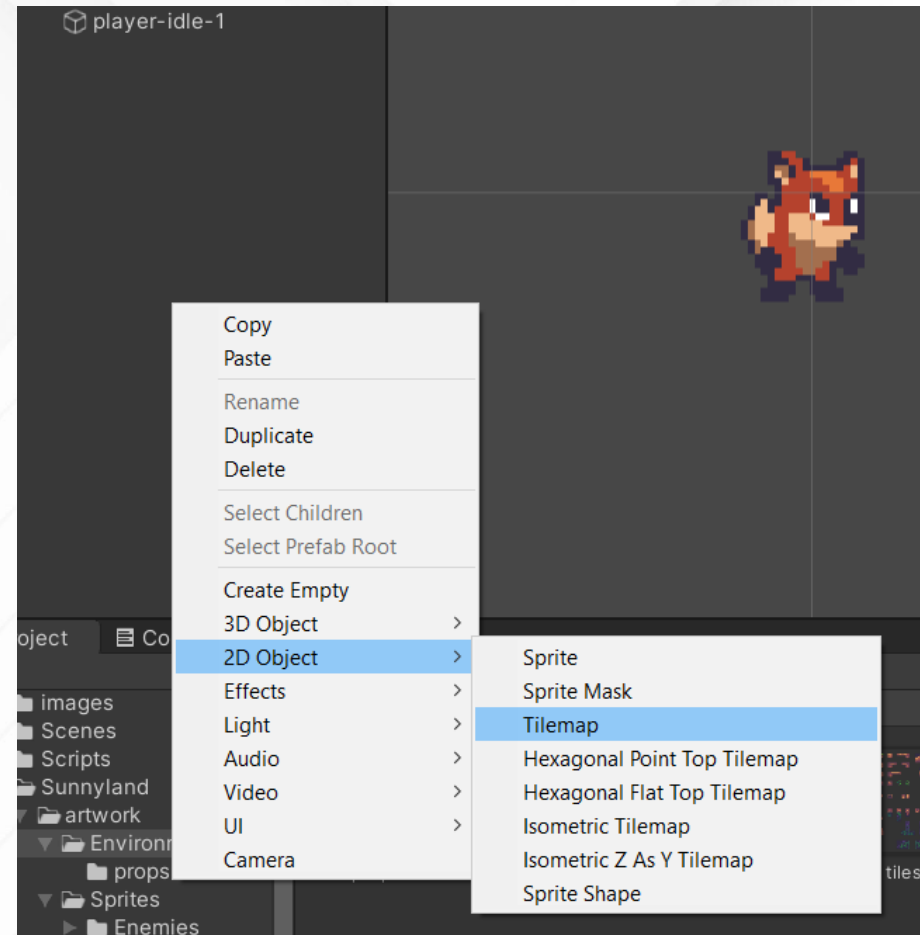


- PRIMER PROYECTO UNITY
- CREAR UN ESCENARIO DE JUEGO
  - Creamos una nueva paleta
    - Nombre: Background
    - Grid: rectagle
    - Cell size: automatic
  - La guardamos en Assets, creamos una nueva carpeta Tile
  - Ahora arrastramos el fichero tileset slice (ya recortado) aunque podríamos recortarlo a partir del otro





- PRIMER PROYECTO UNITY
- CREAR UN ESCENARIO DE JUEGO
  - Así nos quedaría la paleta para poder ir pintando en el juego
  - Usaremos el icono del pincel
  - Necesitaríamos crear un TileMap primero para poder insertar los elementos o tiles





- PRIMER PROYECTO UNITY
- CREAR UN ESCENARIO DE JUEGO
  - Como vemos, los cuadrados no encajan con el tamaño de los cuadrados, por lo que tenemos que pinchar en Grid (Hierarchy) y configurar el tamaño de la celda
  - X e Y tamaño 0.15

