PROGRAMACIÓN MULTIMEDIA Y DISPOSITIVOS MÓVILES

TEMA 4:

Primer Contacto Con El Código. Ciclo De Vida De Una Aplicación

UNION EUROPEA
Fondo Social Europeo
"Una manera de hacer Europa"

"Una manera de hacer Europa". Cofinanciación a cargo del Programa Operativo del FSE 2014-2020 para Extremadura gastos de Ciclos Formativos de Grados Medio y Superior.

ÍNDICE

PMDM

Primer Contacto Con El Código. Ciclo De Vida De Una Aplicación

1. Componentes de una aplicación Android

1.- Componentes de una aplicación Android

- En el tema anterior vimos la estructura de un proyecto Android y aprendimos dónde se colocan cada uno de los elementos que componen una aplicación, tanto elementos de software como recursos gráficos o de datos.
- En este punto vamos a ver los componentes que pueden formar parte de nuestras aplicaciones:

1.- Componentes de una aplicación Android

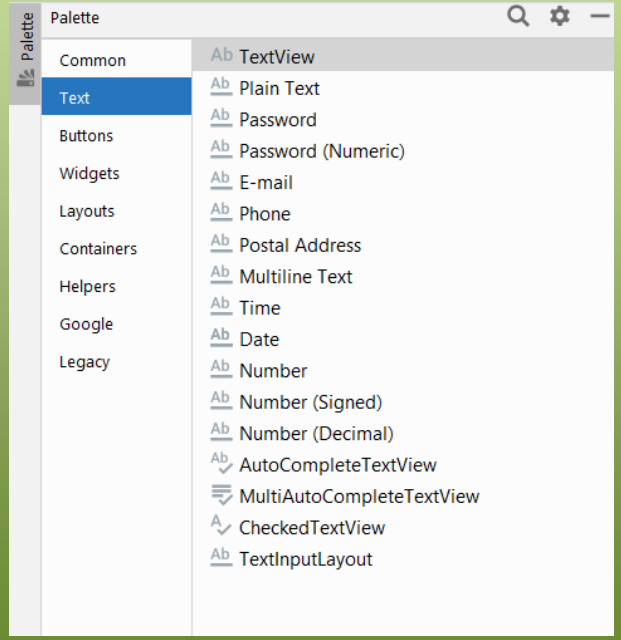
- **ACTIVITY:** Las actividades (activities) representan una pantalla única con una interfaz de usuario. Por ejemplo, una aplicación de correo electrónico puede tener una actividad que muestra una lista de correo electrónico, otra actividad para escribir un correo y otra actividad para leer los mensajes.

1.- Componentes de una aplicación Android

- Aunque las actividades trabajan conjuntamente para dar la sensación de una única aplicación, cada una de ellas es independiente de las otras. Por lo tanto, otra aplicación externa diferente podría iniciar cualquiera de estas actividades (si la aplicación, por ejemplo de correo electrónico, lo permite).
- Por ejemplo, una aplicación que gestiona los contactos podría iniciar la actividad que compone nuevos mensajes de correo indicando como destinatario del mensaje al contacto seleccionado.

1.- Componentes de una aplicación Android

- **VIEW:** Las vistas (view) son los componentes básicos con los que se construye la interfaz gráfica de la aplicación. Por ejemplo: cuadros de texto (***textView***), botones (***button***), listas (***ListView***), listas desplegables (***Spinner***), imágenes (***imageView***)…



1.- Componentes de una aplicación Android

- **SERVICE:** Los servicios (service) son componentes sin interfaz gráfica que se ejecutan en segundo plano.
- Los servicios pueden realizar cualquier tipo de acciones, por ejemplo actualizar datos, lanzar notificaciones, o incluso operaciones de larga duración, como por ejemplo escuchar música sin tener la App en primer plano.

1.- Componentes de una aplicación Android

- **CONTENT PROVIDER:** (Proveedores de Contenido) Es un mecanismo proporcionado por Android que nos permite compartir información entre diferentes aplicaciones.
- Por ejemplo, la aplicación Whatsapp accede a los datos de la aplicación Contactos para agregar contactos o consultarlos de ahí a través de los content provider que se hayan definido.

1.- Componentes de una aplicación Android

- **BROADCAST RECEIVER:** Un broadcast receiver es un componente que está destinado a recibir y responder ante eventos globales generados por el sistema (por ejemplo: "Batería baja", "SMS recibido", "Tarjeta SD insertada", "Descarga finalizada"...) y por otras aplicaciones.

1.- Componentes de una aplicación Android

- **WIDGET:** Los widgets son elementos visuales, normalmente interactivos, que pueden mostrarse en la pantalla principal (home screen) del dispositivo Android y recibir actualizaciones periódicas. Permiten mostrar información de la aplicación al usuario directamente sobre la pantalla principal como, por ejemplo, un reloj, la previsión del tiempo, últimos titulares, etc.

"Una manera de hacer Europa". Cofinanciación a cargo del Programa Operativo del FSE 2014-2020 para Extremadura gastos de Ciclos Formativos de Grados Medio y Superior.

1.- Componentes de una aplicación Android

- **INTENT:** Un intent es el elemento básico de comunicación entre los distintos componentes Android. Se pueden entender como los mensajes o peticiones que son enviados entre los distintos componentes de una aplicación o entre distintas aplicaciones.
- Mediante un intent se puede llamar a una actividad desde cualquier otra, iniciar un servicio, enviar un mensaje broadcast, iniciar otra aplicación, etc.

1.- Componentes de una aplicación Android

- También, gracias a los intent, cualquier aplicación puede iniciar un componente de otra aplicación. Por ejemplo, si es necesario para la aplicación que tengamos abierta capturar una imagen con la cámara, seguramente ya exista otra aplicación que hace eso y que la aplicación inicial puede reutilizar en lugar de desarrollar el código necesario para capturar la foto. Únicamente hay que iniciar la actividad de la aplicación de la cámara de fotos y capturar la imagen. La sensación del usuario es como si la cámara formara parte de la aplicación inicial.

1.- Componentes de una aplicación Android

- Por lo tanto, las aplicaciones de Android no tienen un punto de entrada único (no hay función main()). Debido a que el sistema ejecuta cada aplicación en un proceso independiente con permisos restringidos, ésta no puede activar directamente un componente de otra aplicación, sino que es el sistema operativo Android el encargado de hacerlo.
- Para activar un componente de otra aplicación es necesario enviar un mensaje al sistema que especifica su intención (clase Intent) de iniciar un componente en particular y el sistema operativo es el encargado de activar el componente solicitado.

1.- Componentes de una aplicación Android

- Para más información, visita:
  - https://academiaandroid.com/componentes-aplicacion-android/
  - https://developer.android.com/guide/components/fundamentals?hl=es-419

PMDM

Primer Contacto Con El Código. Ciclo De Vida De Una Aplicación.

2. El componente Activity

2.- El componente Activity

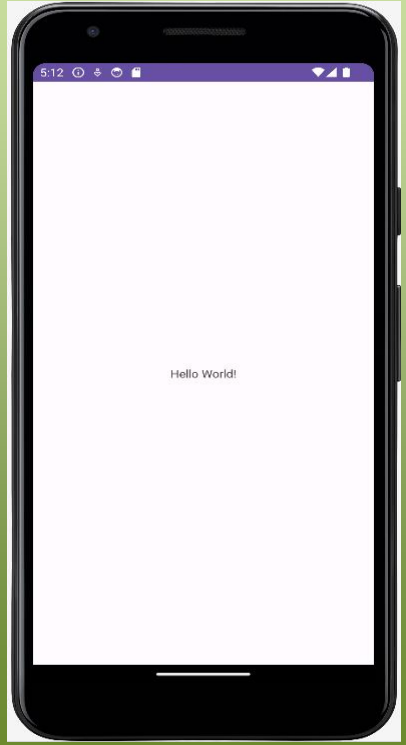
- Una Actividad (Activity) es un componente de Android que ofrece una pantalla con la que los usuarios pueden interactuar con la aplicación, como marcar el teléfono, sacar una foto, enviar un correo electrónico o ver un mapa.
- Por lo general, una aplicación de Android consta de múltiples actividades que están más o menos ligadas entre sí. Habitualmente, se define una actividad "principal", que es la que se presenta al usuario cuando se inicia la aplicación por primera vez. Desde una actividad se puede iniciar otra actividad con el fin de realizar diferentes operaciones.

2.- El componente Activity

- Por lo tanto si cambiamos de ventana, estamos cambiando de Activity.
- Ejemplo de Activity:

activity_main.xml | MainActivity.kt

```
1  package com.example.miprimeraaplicacion
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 }
```



2.- El componente Activity

- Una aplicación puede tener muchas actividades, si bien el usuario sólo interactúa con ellas **de una en una**.
- Fíjate en que la clase principal *ActividadPrincipal* de la aplicación **hereda** de la clase *AppCompatActivity* de Android.
- Android llama al método **onCreate()** cuando una actividad se inicia. En este método se lleva a cabo la inicialización de variables y la configuración de la interfaz de usuario.

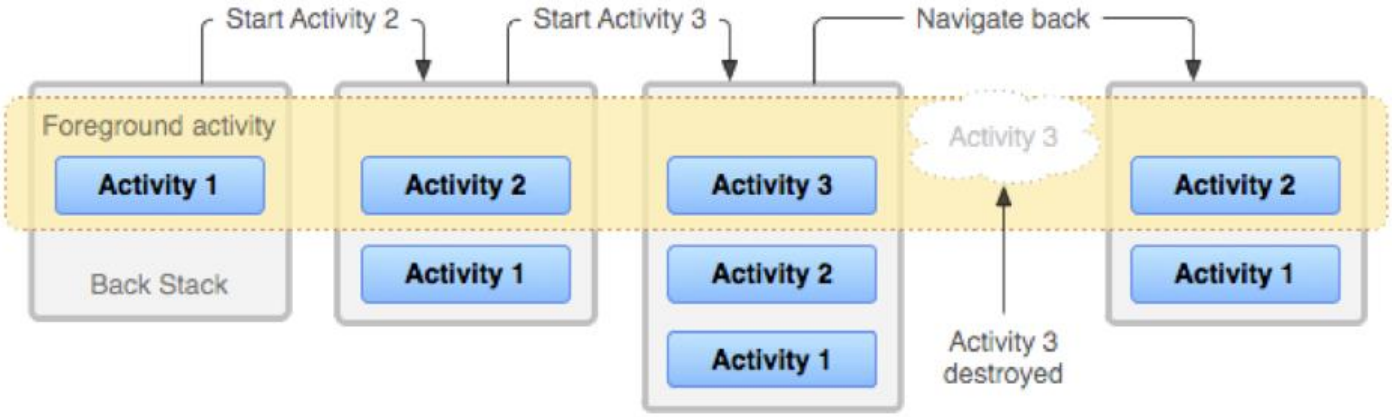
2.- El componente Activity

- ¿Cómo se comporta una Activity cuando otra la sustituye?

Cuando una Activity se inicia, comienza en la cima de la Pila de Actividades. La Activity que está en la cima de la pila es la que se está ejecutando y mostrando al usuario. El resto de Activities quedarán por debajo en la pila.

En el momento en que la Activity sobre la cima se desapila (deje de existir; por ejemplo al pulsar el botón de "atrás"), la que está por debajo será la que ahora ocupa la cima de la pila y por tanto la que estará al frente del usuario.

2.- El componente Activity



Referencias:

- http://jarroba.com/activity-entender-y-usar-una-actividad/
- http://developer.android.com/reference/android/app/Activity.html

EJERCICIOS

- **Ejercicio 01.- (<u>OBLIGATORIO</u>)** Crea un proyecto en Android Studio llamado "*Ejercicio1T4*" que contenga una única Activity.
- Esta aplicación tendrá una imagen de fondo y el texto "¡¡¡Hola mundo!!!" centrado en la pantalla del móvil, de tamaño 60 sp y color rojo.

EJERCICIOS

- Antes de nada, vamos a localizar los dos ficheros, a priori, más importantes de nuestra nueva app: *activity_main.xml* y *MainActivity.kt*.
  - **activity_main.xml** (*Project -> app -> src -> main -> res -> layout -> activity_main.xml*) Contendrá el diseño gráfico de nuestra activity principal.
  - **MainActivity.kt** (*Project -> app -> src -> main -> java -> [nombre del paquete] -> MainActivity.kt*) Contendrá el código Kotlin que establecerá el comportamiento de nuestra activity principal.

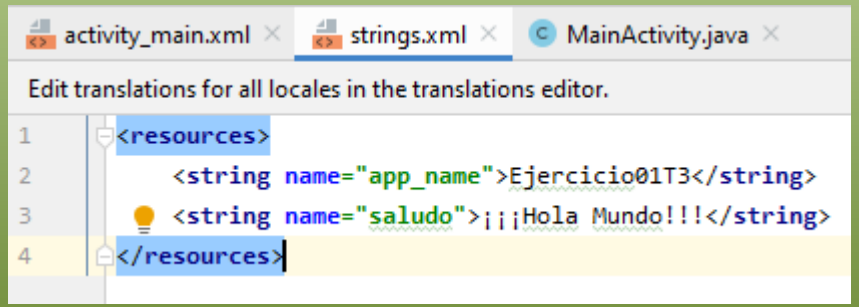
"Una manera de hacer Europa". Cofinanciación a cargo del Programa Operativo del FSE 2014-2020 para Extremadura gastos de Ciclos Formativos de Grados Medio y Superior.

EJERCICIOS

- Lo primero que vamos a cambiar es la asignación del texto de ***android:text="Hello World!"*** a "@string/saludo".
- Para ello, debemos crear una nueva cadena en *strings.xml* y, posteriormente, cambiarlo en el activity_main.xml o en las propiedades del TextView.

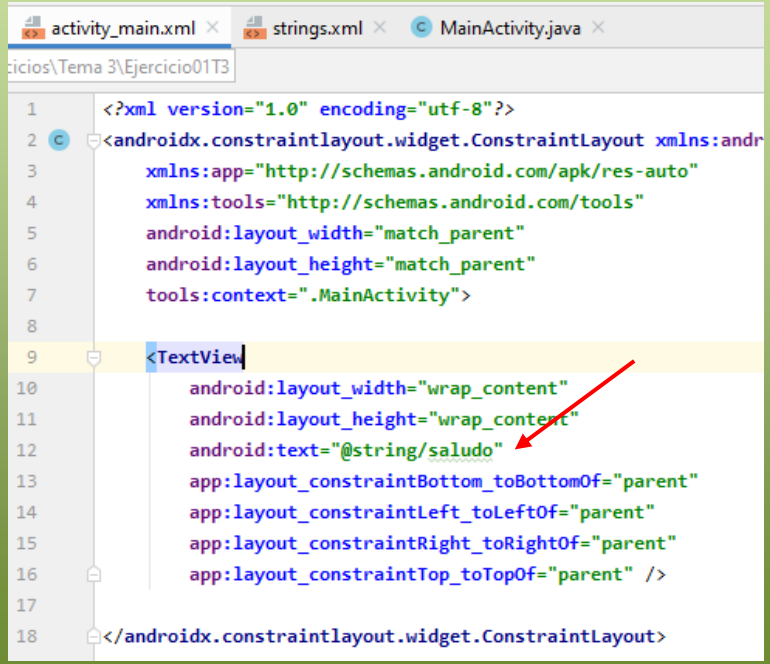
EJERCICIOS

- Abrimos strings.xml desde la ruta *Project -> app -> src -> main -> res -> values -> strings.xml* y añadimos una nueva línea, con el nombre de la nueva cadena y su valor:

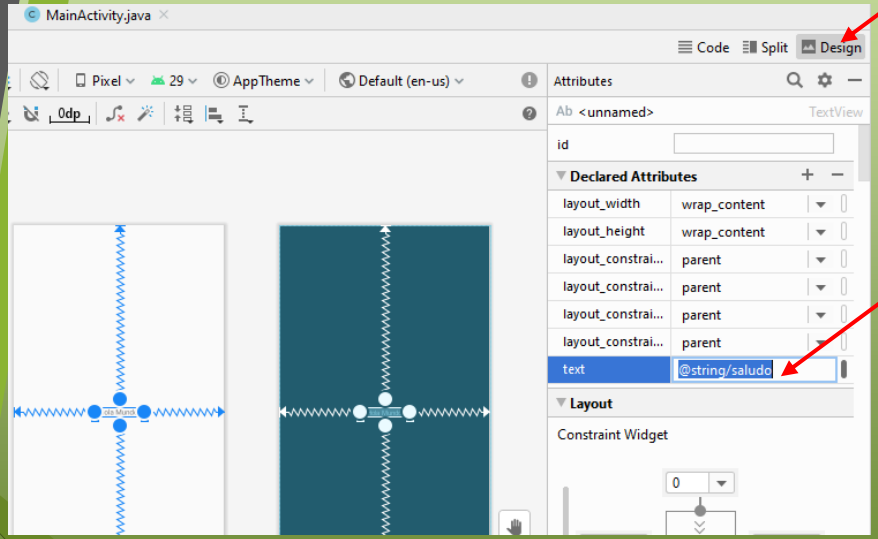


EJERCICIOS

- Ahora si, podemos abrir *activity_main.xml* y modificar el valor del textView.



EJERCICIOS



- O podemos también, desde el diseño (*Design*), modificar el valor de la propiedad *text* del textView.

EJERCICIOS

- Establecemos el rojo como color del texto (nuevamente podemos hacerlo desde código o desde las propiedades del diseño)

activity_main.xml | strings.xml | MainActivity.java

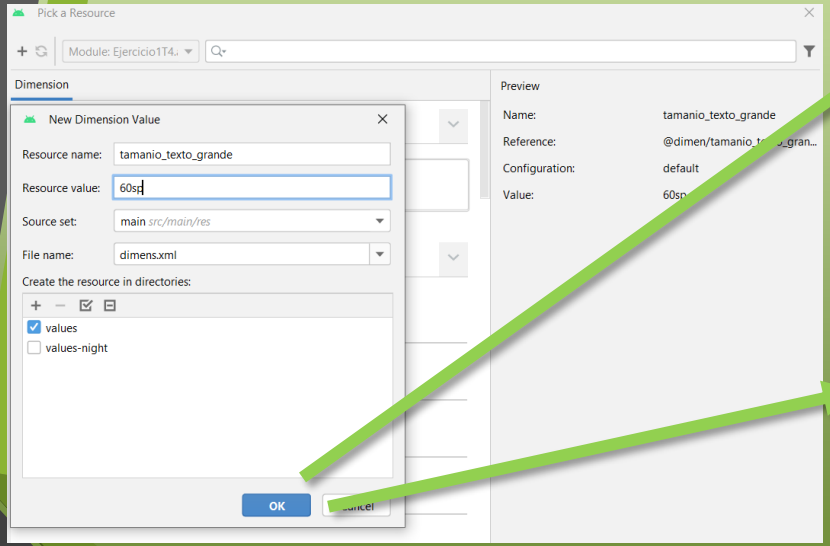
```
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout xmlns:andr
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      tools:context=".MainActivity">
8
9      <TextView
10         android:layout_width="wrap_content"
11         android:layout_height="wrap_content"
12         android:text="@string/saludo"
13         app:layout_constraintBottom_toBottomOf="parent"
14         app:layout_constraintLeft_toLeftOf="parent"
15         app:layout_constraintRight_toRightOf="parent"
16         app:layout_constraintTop_toTopOf="parent"
17         android:textColor="#ff0000"/>
18
19 </androidx.constraintlayout.widget.ConstraintLayout>
```

EJERCICIOS

- Vamos a crear un nuevo recurso para el tamaño de letra. Para ello buscamos en las propiedades *TextSize*, pulsamos los puntos suspensivos y en la ventana que aparece pulsamos el botón + -> *Dimension Value*.
- En *Resource name* ponemos el nombre tamanio_texto_grande y en *Resource value* 6osp.
- Centramos el texto con la propiedad *Gravity* a center.

EJERCICIOS


¡¡¡Hola Mundo!!!
textLocale
textScaleX
textSize
@dimen/tamanio_texto_grande
textStyle
normal
theme


Pick a Resource
Module: Ejercicio1T4..
Dimension
Preview
New Dimension Value
Resource name: tamanio_texto_grande
Resource value: 60sp
Source set: main src/main/res
File name: dimens.xml
Create the resource in directories:
values
values-night
OK
Name: tamanio_texto_grande
Reference:
Configuration: default
Value:

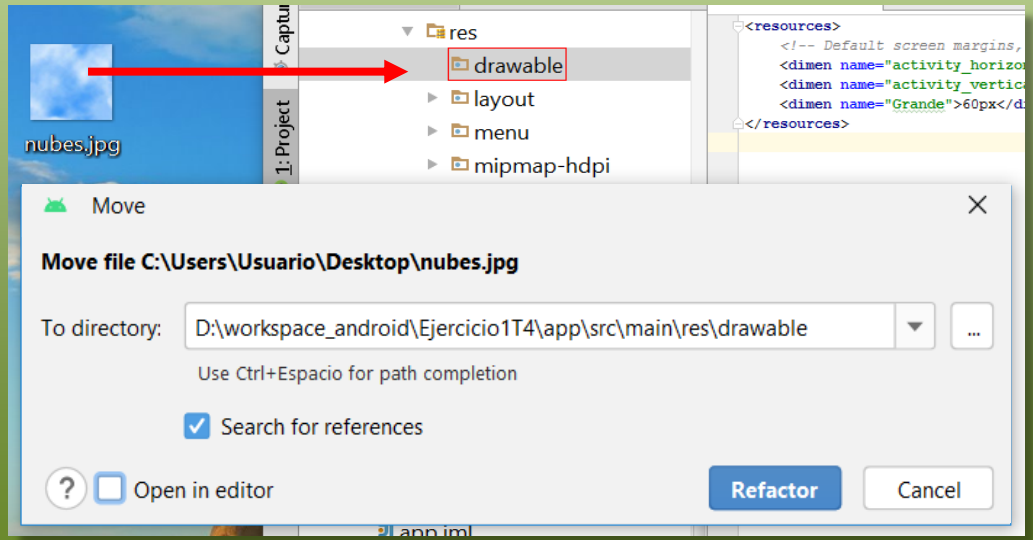

activity_main.xml
MainActivity.kt
dimens.xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
<dimen name="tamanio_texto_grande">60sp</dimen>
</resources>

EJERCICIOS

- Unidades de medida que tenemos disponibles en Android:
  - **dp** (Density-independent Pixels): Es una unidad abstracta que se basa en la densidad física de la pantalla. Esta unidad es equivalente a un píxel en una pantalla con una densidad de 160 dpi. Cuando se está ejecutando en una pantalla de mayor densidad, se aumenta el número de píxeles utilizados para dibujar 1dp según los dpi's de la pantalla. Por otro lado, si la pantalla es de menor densidad, el número de píxeles utilizados para 1dp se reducirán. Utilizar las unidades dp en lugar de píxeles es la solución más simple para tratar los diferentes tamaños de pantalla de los dispositivos.
    - **sp** (Scale-independent Pixels): Esta unidad es como la anterior, pero se escala según el tamaño de fuente configurada. Se recomienda utilizar esta unidad si se especifican tamaños de fuente, por lo que se ajusta tanto para la densidad de pantalla y como a las preferencias del usuario.
    - **px** (Pixels): Corresponde a un píxel real en la pantalla. Esta unidad de medida no se recomienda porque la representación real puede variar según el dispositivo en el que se ejecute, ya que cada uno de ellos puede tener un número diferente de píxeles por pulgada y pueden tener más o menos píxeles totales disponibles en la pantalla.
    - **mm** (Milímetros): Son milímetros reales según el tamaño físico de la pantalla.
    - **in** (Pulgadas): Son pulgadas reales según el tamaño físico de la pantalla.
    - **pt** (Points): Es un 1/72 de una pulgada, según el tamaño físico de la pantalla

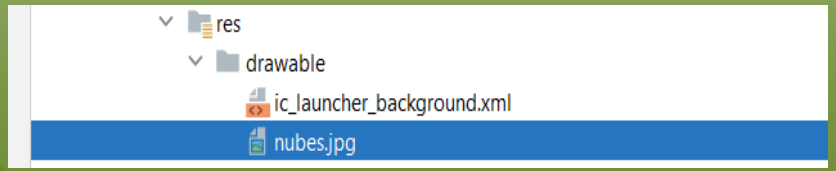
EJERCICIOS

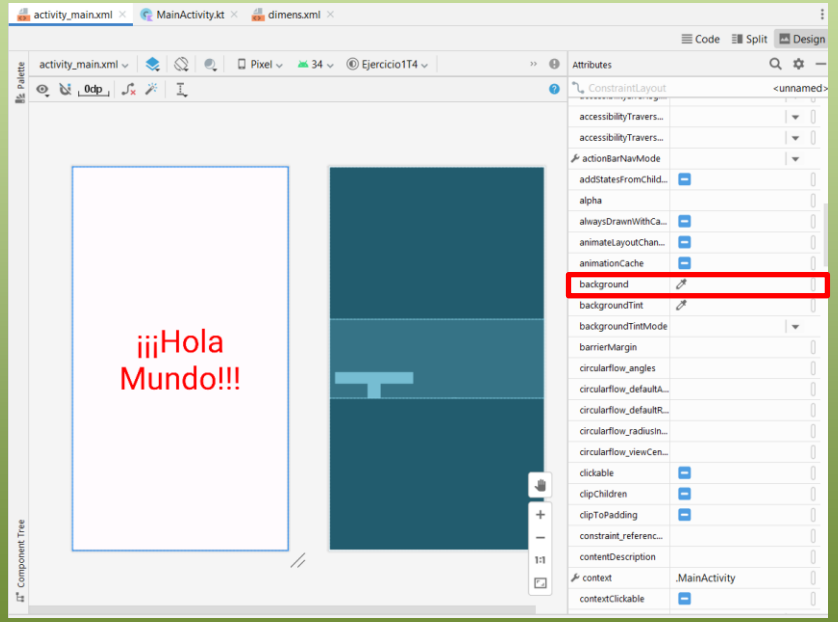
- Vamos a añadir un fondo a nuestra aplicación. Para ello arrastramos el jpg a la carpeta "drawable" de nuestro proyecto:



EJERCICIOS

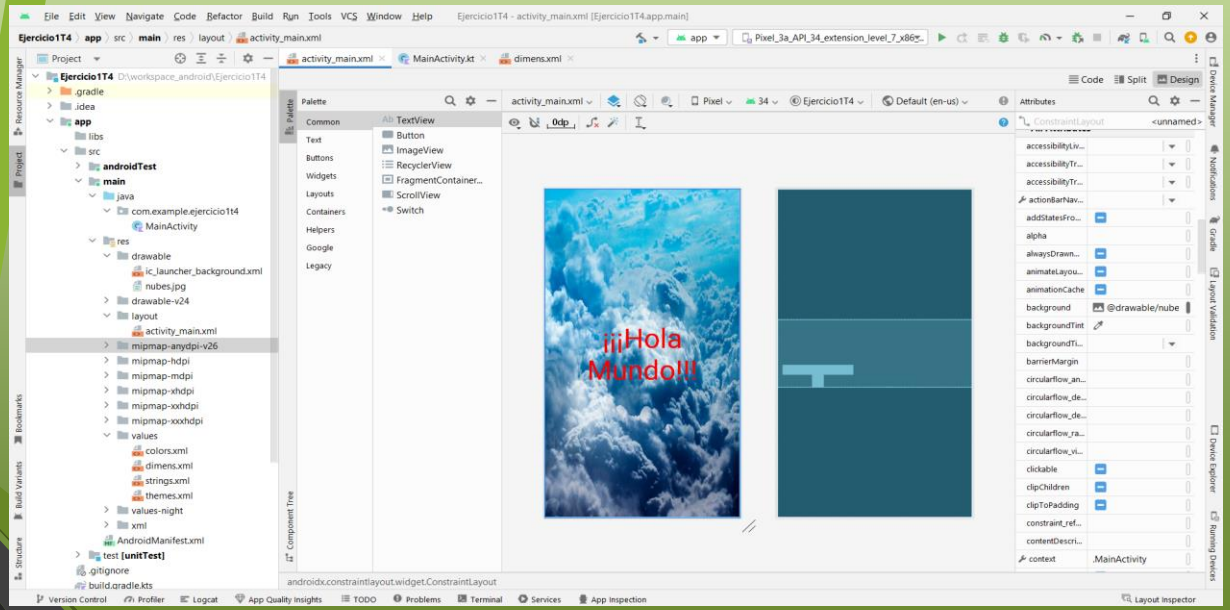
- Nos vamos a la vista diseño, pinchamos en el fondo, elegimos "background" y seleccionamos nuestro fichero:





EJERCICIOS

- Resultado:



EJERCICIOS

- **Ejercicio 02.- (<u>OBLIGATORIO</u>)** Crea un proyecto en Android Studio llamado "*Ejercicio2T4*" que contenga una única Activity.
- Esta aplicación tendrá una imagen de fondo y dos textos: "Bienvenido a mi app" (en azul y con tamaño 60sp) y, justo debajo, tu nombre (en el color que elijas y con tamaño 32sp).

EJERCICIOS

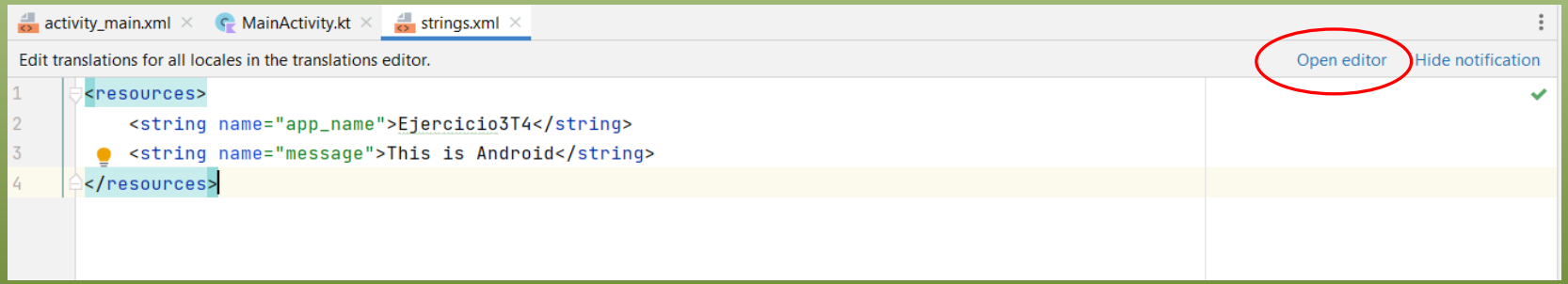
- **Ejercicio 03.- (OBLIGATORIO):** Crea un proyecto en Android Studio llamado "*Ejercicio3T4*" que contenga una única Activity.
- Este proyecto tendrá un texto en el centro de la pantalla con color verde y tamaño 32sp. Pero, en este ejercicio, mejoraremos algunos aspectos respecto a los anteriores: lo crearemos en dos idiomas distintos, en inglés (por defecto) y en castellano.

EJERCICIOS

- El fichero strings.xml te deberá quedar algo parecido a esto:

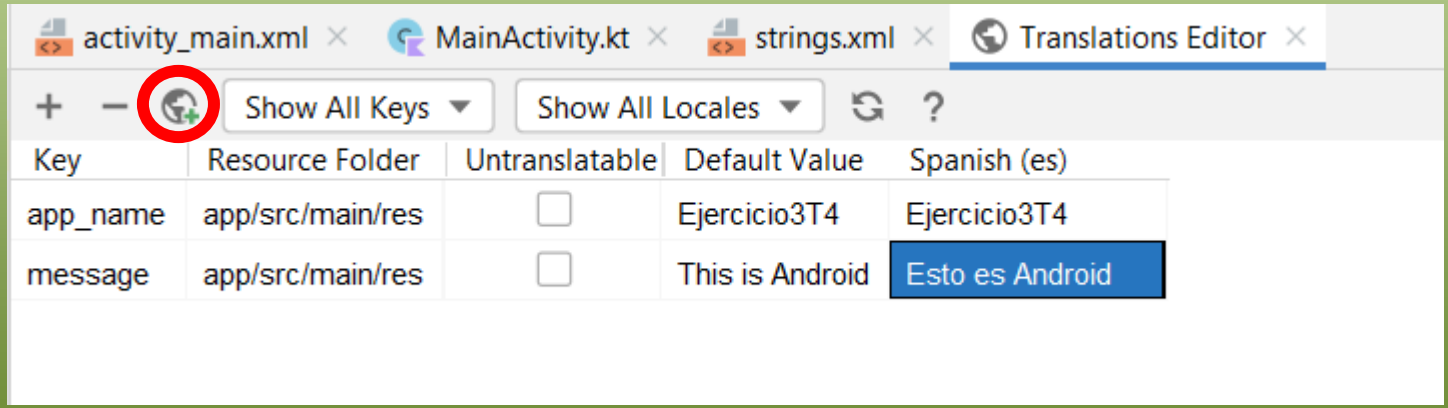
EJERCICIOS

- Ahora añadiremos a nuestra aplicación las cadenas en castellano con ayuda del editor de traducciones:



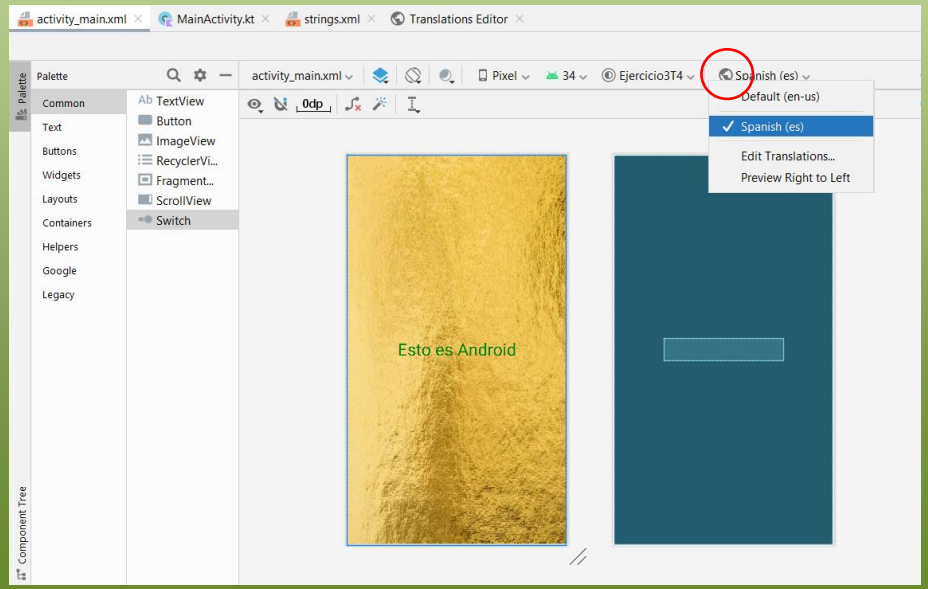
EJERCICIOS

- Añadimos una localización como por ejemplo Spanish (es) y luego vamos rellenando la tabla:



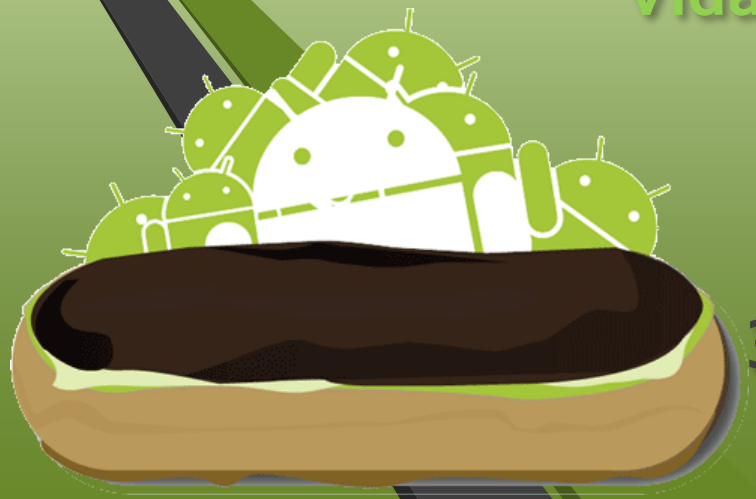
EJERCICIOS

- Una vez creado el idioma, podemos cargar nuestra app en un lenguaje u otro eligiendo el que queramos:



PMDM

Primer Contacto Con El Código. Ciclo De Vida De Una Aplicación.



3. Ciclo de vida de una Activity

3.- Ciclo de vida de una Activity

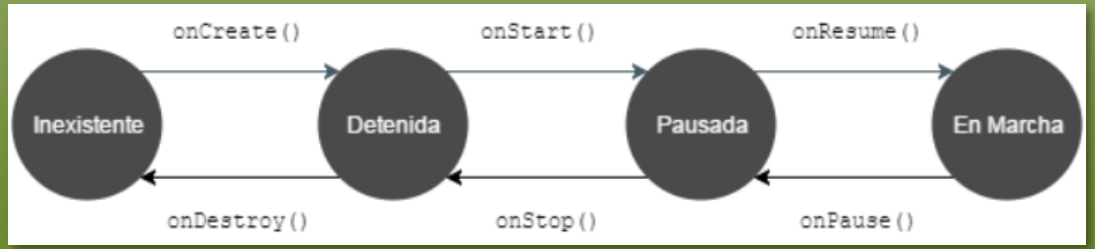
- Debido al diseño del sistema operativo Android que hablamos anteriormente, una actividad puede atravesar los siguientes estados para mantener el flujo entre apps y eventos del sistema:
  - Inexistente
  - Detenida
  - Pausada
  - En marcha
- La siguiente tabla nos muestra qué sucede en memoria, UI y el proceso de primer plano, cuando la actividad pasa por su ciclo de vida:

3.- Ciclo de vida de una Activity

Estado	¿En memoria?	¿Visible al usuario?	¿En primer plano?
Inexistente	No	No	No
Detenida	Si	No	No
Pausada	Si	Si	No
En Marcha	Si	Si	Si

http://www.hermosaprogramacion.com/2018/09/actividades-ciclo-de-vida/

- En este diagrama podemos ver los nombres de dichos métodos cuando una actividad va desde la construcción hasta su destrucción:



3.- Ciclo de vida de una Activity

- Métodos del ciclo de vida de una Actividad:
  - **onCreate()**
    - Se ejecuta cuando se crea una nueva instancia en memoria de la actividad. Aquí se inicializan las variables.
  - **onStart()**
    - Raramente utilizado. Se le llama después del onCreate() y en él se escriben sentencias relacionadas con la UI, ya que la actividad es ya visible al usuario.

3.- Ciclo de vida de una Activity

- **onResume()**
  - Se ejecuta antes de que la actividad se ponga *En Marcha* para interactuar con el usuario en primer plano. Normalmente en su interior se suelen ejecutar animaciones, iniciar la aplicación cámara, etc.
- **onPause()**
  - Es llamado cuando el sistema quita del primer plano a la actividad y así entrar en el estado Pausada.

3.- Ciclo de vida de una Activity

- **onStop()**
  - Es llamado antes de llegar al estado Detenida donde la actividad no será visible para el usuario.
- **onDestroy()**
  - El sistema llama a este método antes de destruir la instancia de la actividad.

3.- Ciclo de vida de una Activity

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        //Nuestro código a ejecutar en este momento
    }
    override fun onStart() {
        super.onStart()
        //Nuestro código a ejecutar en este momento
    }
    override fun onRestart() {
        super.onRestart()
        //Nuestro código a ejecutar en este momento
    }
    override fun onResume() {
        super.onResume()
        //Nuestro código a ejecutar en este momento
    }
    override fun onPause() {
        super.onPause()
        //Nuestro código a ejecutar en este momento
    }
    override fun onStop() {
        super.onStop()
        //Nuestro código a ejecutar en este momento
    }
    override fun onDestroy() {
        super.onDestroy()
        //Nuestro código a ejecutar en este momento
    }
}
```

- Podemos añadir opcionalmente cada uno de los métodos para sobrescribirlos (esto lo indica el override) a la clase padre Activity, no siendo obligatorio ninguno.
- Dentro escribiremos el código que queramos que se ejecute, y cada método será llamado por la clase Activity en el momento indicado anteriormente.
- A continuación escribimos todos los métodos posibles, los que no necesitemos ni los escribimos. No ponerlos no les priva de existencia, siguen existiendo en la clase Activity, por tanto en el ciclo de vida de nuestra Actividad, pero al ejecutarse a su momento estarán vacíos.

3.- Ciclo de vida de una Activity

- Para más información, visita:
  - http://www.androidcurso.com/index.php/tutoriales-android/37-unidad-6-multimedia-y-ciclo-de-vida/158-ciclo-de-vida-de-una-actividad
  - Ciclo de vida de una aplicación en Android: https://youtu.be/E1vAobJ8Fxo

PMDM

Primer Contacto Con El Código. Ciclo De Vida De Una Aplicación

4. Acceso a recursos

4.- Acceso a recursos

- Una vez que proporciones un recurso en tu aplicación, puedes aplicarlo haciendo referencia a su ID de recurso.
- Todos los ID de recursos se definen en la *clase R* de tu proyecto. Esta clase contiene los ID de recurso de todos los recursos de tu directorio *res/*.
- Para cada tipo de recurso hay una subclase R (por ejemplo, *R.drawable* para todos los recursos de elementos de diseño), y para cada recurso de ese tipo hay un valor entero estático (por ejemplo, R.drawable.icon). Ese valor entero es el ID del recurso que puedes usar para recuperar tu recurso.

4.- Acceso a recursos

- El ID de recurso siempre está compuesto por lo siguiente:
  - El tipo de recurso: Cada recurso se agrupa en un "tipo", como string, drawable y layout.
  - El nombre del recurso.

4.- Acceso a recursos

- Existen dos maneras de acceder a un recurso:
  - En código (Kotlin): Utilizando el método **getResources()**.

    val saludo: String = getResources().getString(R.string.saludo);

  - En XML: Usando una sintaxis XML que también corresponde al ID de recurso definido en tu clase R.

    @string/saludo