

ELECTIVA IV

Ing. Jorge Luis Charco Aguirre, MSc.

Ciclo 2020-2021 / Online

COMPUTACIÓN MOVIL



PLATAFORMAS MÓVILES DE COMUNICACIÓN Y PROCESAMIENTO

Las nuevas
plataformas móviles,
han **abierto nuevas**
líneas para los usos y
aplicaciones
informáticas

Posibilitan el **acceso**
ubicua e
instantáneamente a
múltiples recursos
disponibles a través
de Internet

App Store
(funcionando desde
mediados de 2008)
incluye mas de
420.000 aplicaciones
para el iPhone

DISPOSITIVO MÓVIL

Es un sistema de reducido tamaño ("de bolsillo"), que típicamente dispone de una pantalla de visualización con una entrada táctil, y/o un teclado miniatura.



En las PDA ("Personal Digital Assistant") usualmente las entradas y salidas se encuentran combinadas en una misma pantalla táctil.



COMUNICACIÓN DE DATOS PARA PLATAFORMAS MÓVILES



Servicio celular de datos, tal como GSM, CDMA o GPRS, y más recientemente redes 3G tales como WCDMA, EDGE o CDMA2000.



Conexiones Wi-Fi, que ofrecen prestaciones mayores, pueden ser de redes privadas o de redes Wi-Fi públicas.



Acceso Internet por satélite, cuando no están disponibles las dos anteriores. Las conexiones se realizan a través de satélites geoestacionarios.



RETOS DE LAS PLATAFORMAS MÓVILES

Reducido ancho de banda.

Estándares de seguridad.

Consumo de potencia.

Interferencias

Posibles daños a la salud.

LAS PLATAFORMAS MÓVILES PUEDEN REQUERIR:

1

Tamaño (factor de forma), peso y consumo de energía muy reducidos.

2

Tolerancia a las vibraciones y golpes; así como a temperaturas extremas

3

Pantalla de gran luminosidad (para trabajar en exteriores).

CARACTERÍSTICAS DE LA TECNOLOGÍA MÓVIL

Movilidad

Amplio alcance

Ubicuidad

Comodidad

Conectividad instantánea

Personalización

Localización de productos y servicios.

CONTROLADORES DE LA TECNOLOGÍA MÓVIL



Amplia disponibilidad
de los dispositivos
móviles.



No se requiere una PC.



La cultura del auricular.



Los vendedores están
impulsando el comercio
móvil.



La reducción de los
precios y el aumento de
las funcionalidades.



Mejoramiento del
ancho de banda.

COMERCIO MÓVIL

01

Son actividades de comercio electrónico o negocio electrónico que se realizan en un entorno inalámbrico.

02

Es una extensión natural del e-commerce creando nuevas oportunidades de negocio.

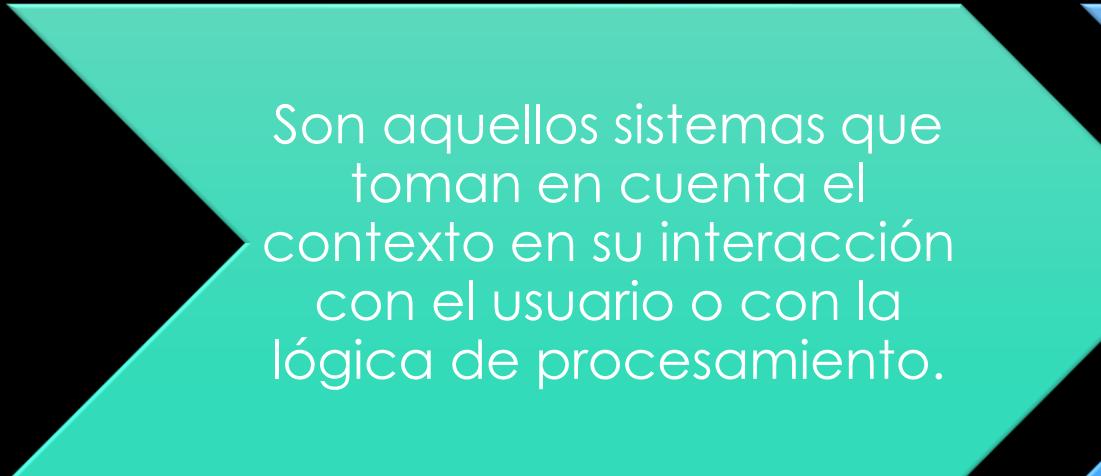
03

Algunas aplicaciones:
Gerencias de servicios de campo,
Localización de productos, Aplicaciones financieras.

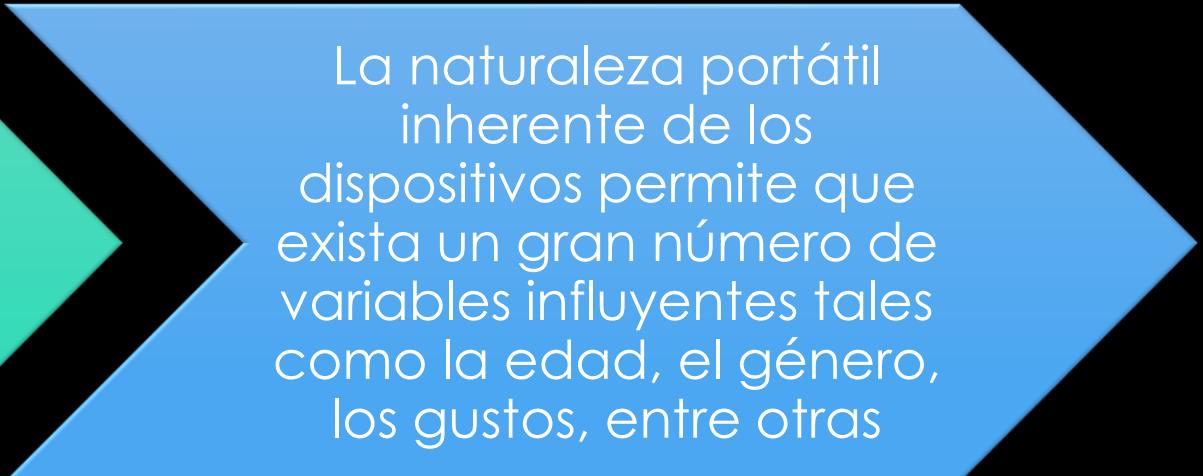
MODELAMIENTO DE APLICACIONES MÓVILES BASADAS EN SU CONTEXTO



MODELAMIENTO DE APLICACIONES MÓVILES BASADAS EN SU CONTEXTO



Son aquellos sistemas que toman en cuenta el contexto en su interacción con el usuario o con la lógica de procesamiento.



La naturaleza portátil inherente de los dispositivos permite que exista un gran número de variables influyentes tales como la edad, el género, los gustos, entre otras

CONTEXTO

1

Información que caracteriza al entorno en el cual se ejecuta una aplicación o servicio.

2

Otro rasgo destacado del contexto es su validez temporal.

3

Evoluciona deprisa. El contexto debe ser procesado en tiempo real.

CONTEXTO: ¿PARA QUÉ SIRVE?

Para enriquecer servicios

Ayuda adaptarse mucho mejor y de forma automática a las condiciones del entorno.

Proporciona un mejor servicio.

Extrae la información necesaria para la aplicación y lo usa para personalizarla.

GRADOS DE CONTEXTO



CONTEXTO
INDIVIDUAL



CONTEXTO DE GRUPO

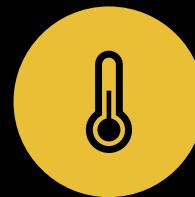


CONTEXTO
COLECTIVO

DONDE AQUIRIR EL CONTEXTO



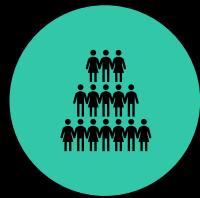
Contexto
espacial.



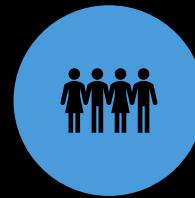
Contexto
temporal.



Contexto
ambiental



Contexto
personal.



Contexto
social

CONTEXTO VS PERFIL

01

SON CONCEPTOS
COMPLEMENTARIOS, Y
MUY RELACIONADOS.

02

UN PERFIL DE UN USUARIO
CONTIENE INFORMACIÓN
SOBRE RASGOS ESTABLES
DE UNA PERSONA, QUE EN
GENERAL SE SUPONE QUE
VARÍAN LENTAMENTE
CON EL TIEMPO

03

**EL CONTEXTO DE UN
USUARIO**, EN CAMBIO,
EVALÚA LA SITUACIÓN DE
LA PERSONA EN UN
MOMENTO CONCRETO

EN QUE ENTORNOS SON ÚTILES



Entornos urbanos, la situación de diferentes entidades (incluidos los usuarios) es dinámica y cambiante.



Gran impacto en la privacidad, ya que se debe ajustar al gusto y las preferencias del usuario de la mejor manera.

CASO 1: APLICACIONES MÓVILES BASADAS EN SU CONTEXTO



Aprendizaje de idiomas como apoyo a usuarios que viven en países extranjeros.



La aplicación Android se enfoca en estudiantes interesados en programas de intercambio entre estos dos países



La aplicación sugiere vocabulario, frases u oraciones que son útiles para el usuario, considerando atributos tales como el género del usuario, la geolocalización y el lenguaje nativo.

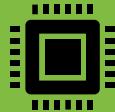
CASO 2: APLICACIONES MÓVILES BASADAS EN SU CONTEXTO



Sistema de aprendizaje situado capaz de entregar contenidos a estudiantes de ingeniería de acuerdo con el contexto.



Se fundamenta en la filosofía cliente-servidor. El cliente se configura para proporcionar un acceso offline y online.

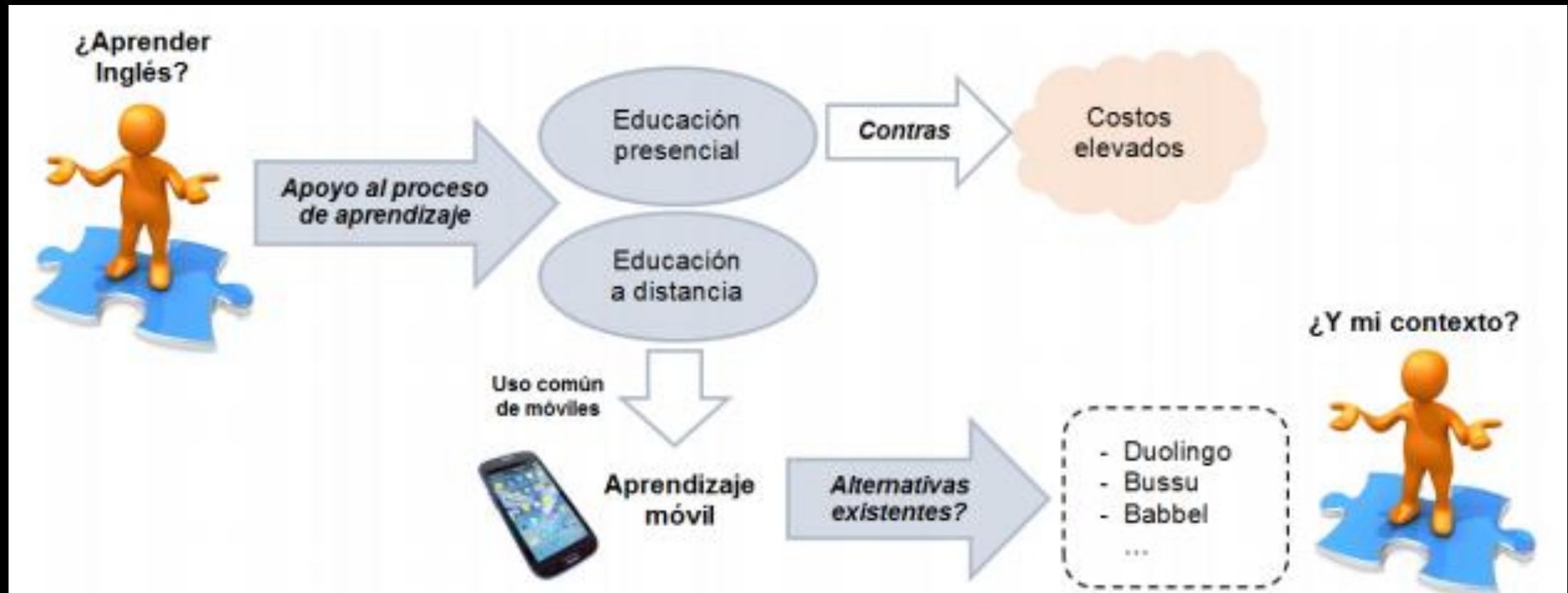


La configuración offline se hace uso de la realidad aumentada mediante objetos modelados en 3D que son almacenados-presentados en una computadora personal.



Mediante el acceso online el estudiante puede seleccionar a través de una aplicación móvil, el tipo de lectura a realizar: QR Code, mientras el servidor gestiona los contenidos respectivos.

NECESIDAD DEL USUARIO



ANDROID

ANDROID

1

Sistema operativo
para dispositivos
móviles

2

Núcleo basado en
el de Linux

3

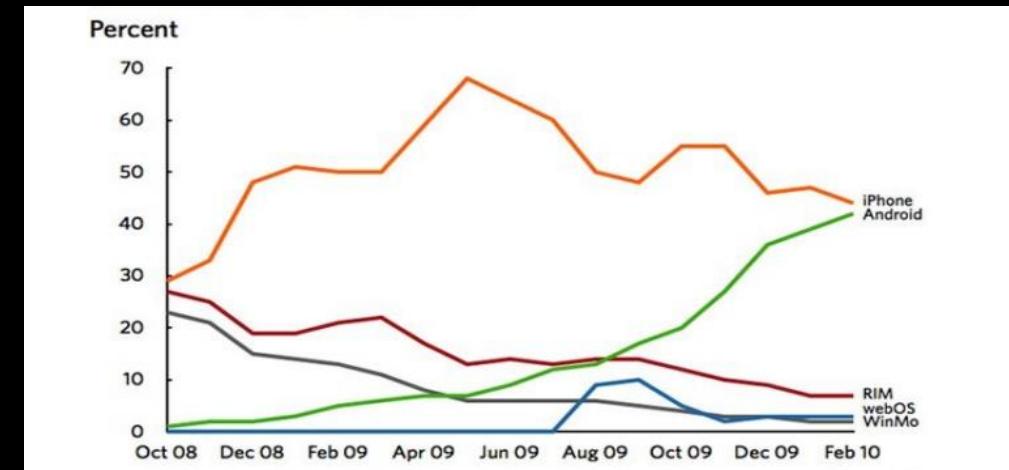
Programación de
aplicaciones en
Java

HISTORIA

- Android 1.1 se publica en febrero de 2009 (coincide con la proliferación de smartphones táctiles).

Licencia apache:

- ✓ Open Source
- ✓ Permite a los fabricantes añadir extensiones propietarias sin ponerlas en manos de la comunidad del software libre

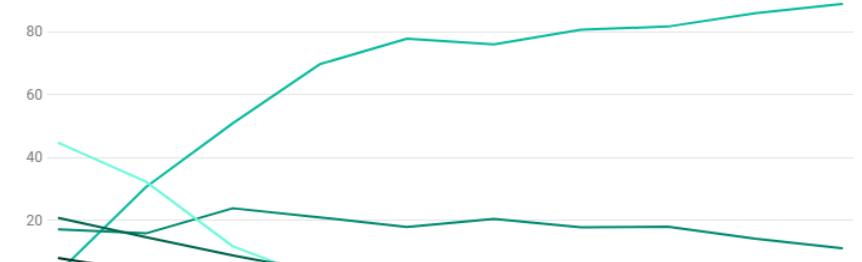


Diez años de crecimiento imparable

Evolución de la cuota de mercado de Android (2009-2018)

Cifras expresadas en porcentajes

■ Android ■ iOS ■ Symbian ■ RIM ■ Microsoft



ASPECTOS POSITIVOS

Código abierto

- Valor añadido para todos
- Mantenibilidad
- Seguridad informática
- Transparencia del uso de sensores

Servicios gratuitos de Google

ASPECTOS NEGATIVOS

01

Obligatorio log-in
con el ID de
Google
(dependencia)

02

Constante
intercambio de
datos con Google

03

Envío de
localización
(desactivable)

04

Aunque el SO sea
libre, gran parte de
su valor está en los
servicios gratuitos
de Google, que no
son libres.

ANDROID SDK

01

Licencias,
distribución y
desarrollo
gratuitos.

02

Acceso al
hardware de WiFi,
GPS, Bluetooth y
telefonía.

03

Control completo
de multimedia,
incluyendo la
cámara y el
micrófono.

04

APIs para los
sensores:
acelerómetros y
brújula.

ANDROID SDK

Almacenes de datos compartidos, proveedores de contenidos, SQLite, acceso a SD Card.

Aplicaciones y procesos en segundo plano

Widgets para la pantalla de inicio (escritorio).

Uso de mapas y sus controles desde las aplicaciones.

Aceleración gráfica por hardware, incluyendo OpenGL 2.0 para 3D.

CONSIDERACIONES PARA EL DESARROLLO



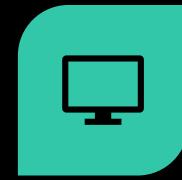
PEQUEÑA
CAPACIDAD DE
PROCESAMIENTO



MEMORIA RAM
LIMITADA



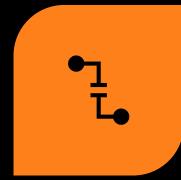
MEMORIA
PERMANENTE DE
POCA CAPACIDAD



PANTALLAS
PEQUEÑAS DE
POCA
RESOLUCIÓN



TRANSFERENCIAS
DE DATOS
COSTOSA Y LENTA
(EN TÉRMINOS DE
ENERGÍA Y
ECONÓMICOS)



INESTABILIDAD DE
LAS CONEXIONES
DE DATOS



BATERÍA MUY
LIMITADA



NECESIDAD DE
TERMINAR LA
APLICACIÓN EN
CUALQUIER
MOMENTO

TIPOS DE APLICACIONES

Primer plano
(activities)

- Segundo plano
- Servicios puros

Servicios
combinados
con actividades

Widgets de
escritorio

ENTORNO DE DESARROLLO

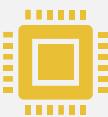


REQUERIMIENTOS DE SISTEMA



**Windows 7/8/10
(32-bit o 64-bit)**

MacOs 10 o superior
(Sistemas basados en Intel)
Linux con versión 2.19 o
superior de GNU C Library



**4GB de RAM mínimo (8 GB o
superior es preferible)**

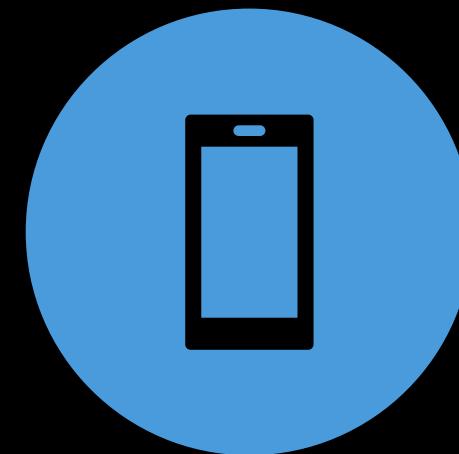
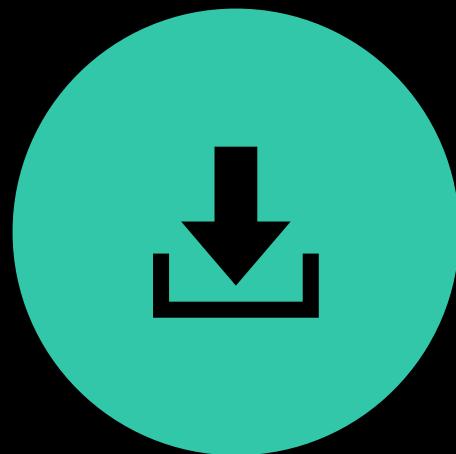


**Aproximadamente 4GB de
espacio disponible en disco**



**1280 x 800 de resolución de
pantalla**

PASOS A SEGUIR



PASO 1. DESCARGA E INSTALACIÓN DE
JAVA.

PASO 2. DESCARGA E INSTALACIÓN DE
ANDROID STUDIO Y EL SDK DE ANDROID.

PASO # 1 DESCARGAR JAVA

- Crear una nueva variable de entorno llamada `JAVA_HOME` y cuyo valor sea la ruta donde se instalo el JDK.

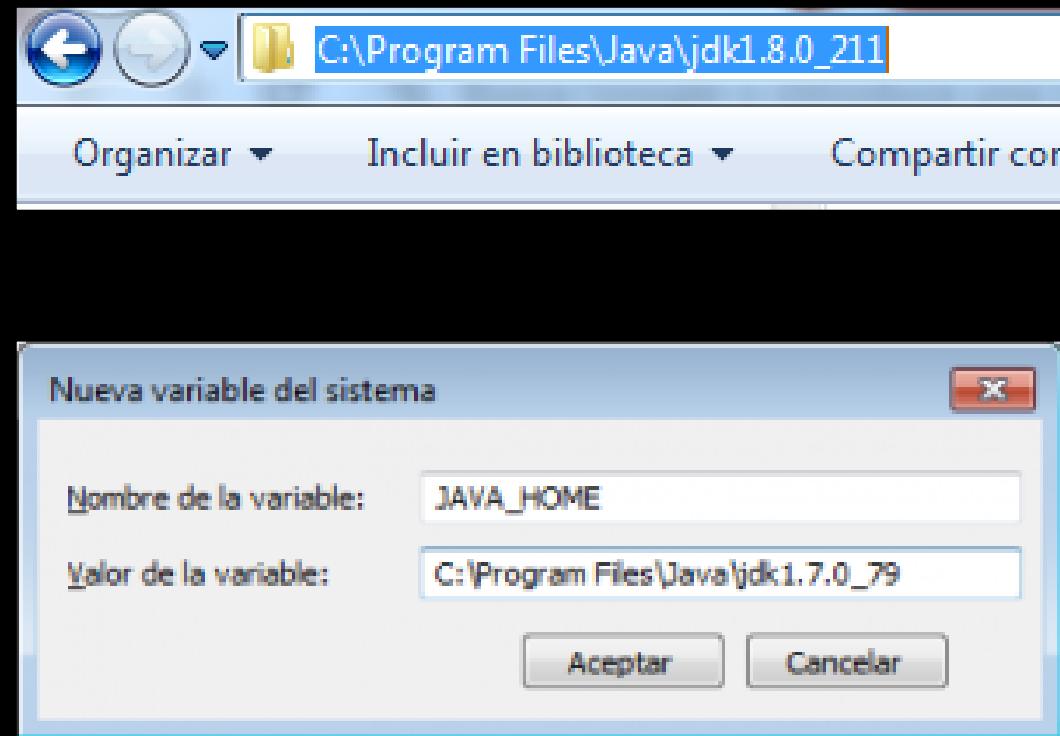
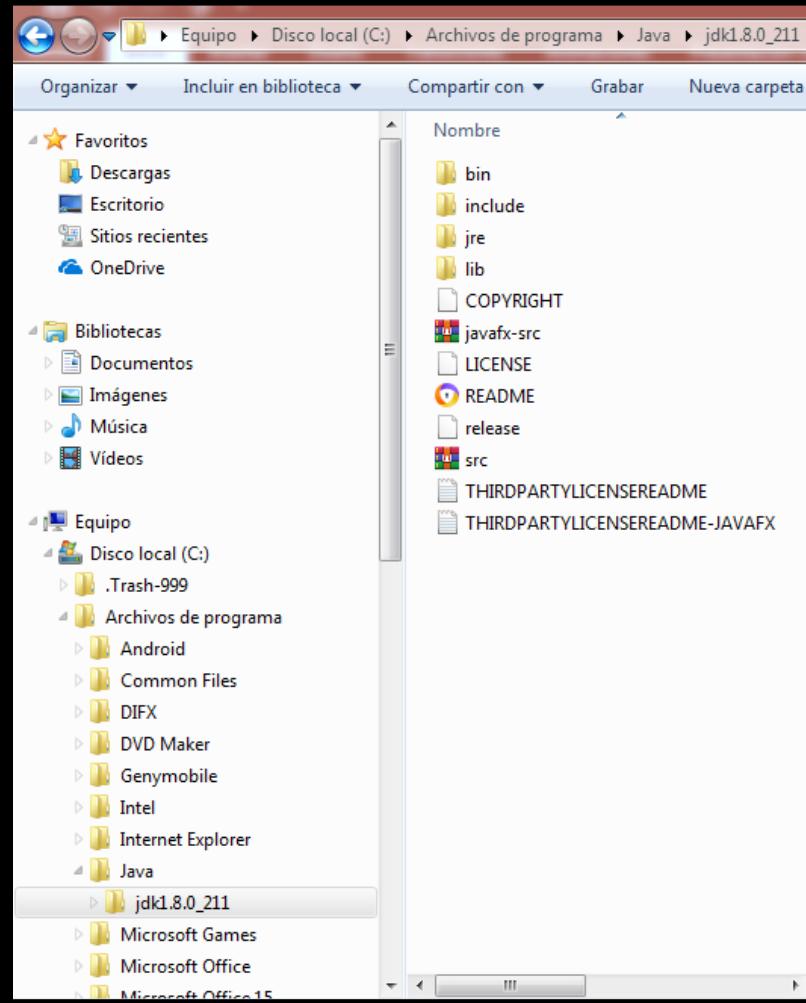
Java SE Development Kit 7u51

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Accept License Agreement Decline License Agreement

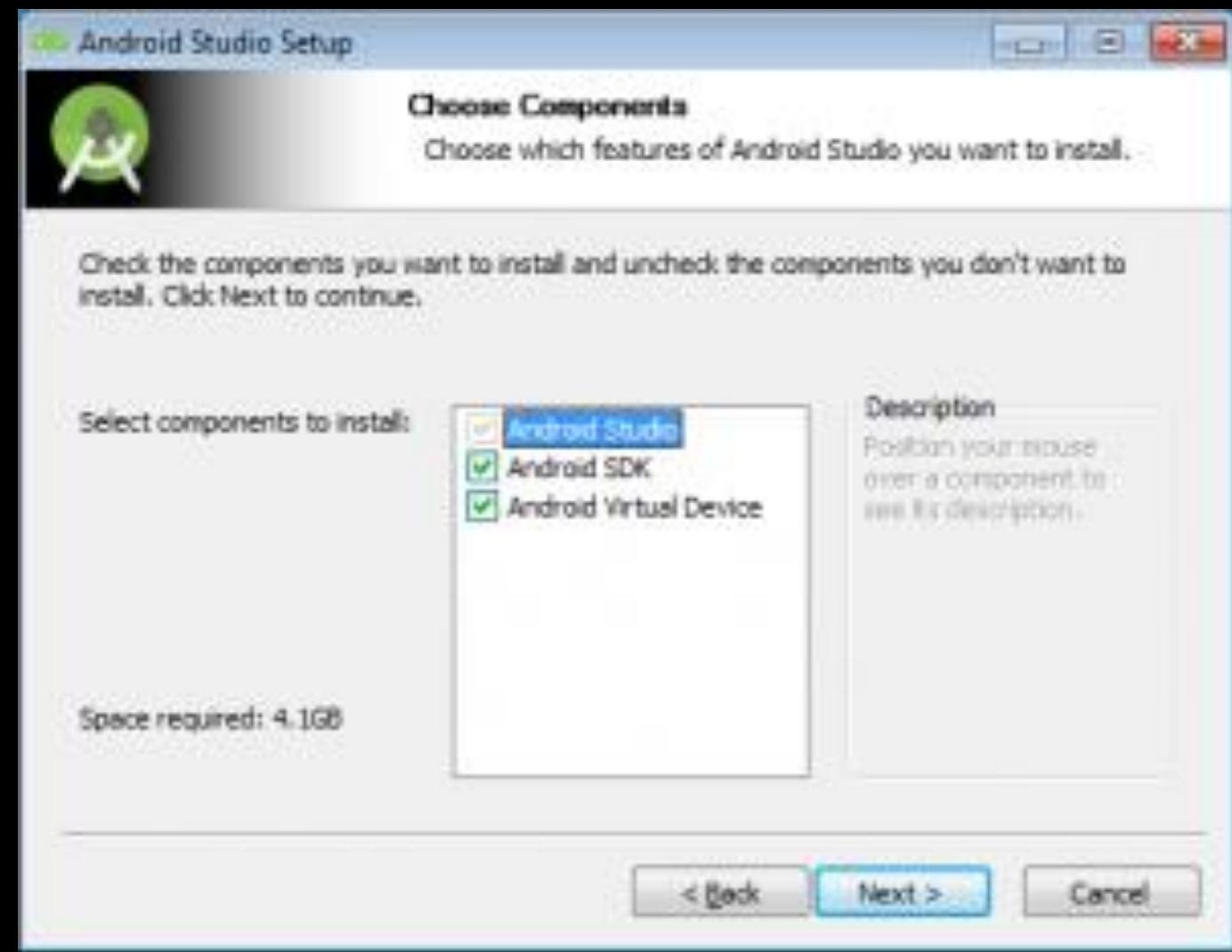
Product / File Description	File Size	Download
Linux ARM v6/v7 Hard Float ABI	67.7 MB	jdk-7u51-linux-arm-vfp-hflt.tar.gz
Linux ARM v6/v7 Soft Float ABI	67.68 MB	jdk-7u51-linux-arm-vfp-sflt.tar.gz
Linux x86	115.65 MB	jdk-7u51-linux-i586.rpm
Linux x86	132.98 MB	jdk-7u51-linux-i586.tar.gz
Linux x64	116.96 MB	jdk-7u51-linux-x64.rpm
Linux x64	131.8 MB	jdk-7u51-linux-x64.tar.gz
Mac OS X x64	179.49 MB	jdk-7u51-macosx-x64.dmg
Solaris x86 (SVR4 package)	140.02 MB	jdk-7u51-solaris-i586.tar.Z
Solaris x86	95.13 MB	jdk-7u51-solaris-i586.tar.gz
Solaris x64 (SVR4 package)	24.53 MB	jdk-7u51-solaris-x64.tar.Z
Solaris x64	16.28 MB	jdk-7u51-solaris-x64.tar.gz
Solaris SPARC (SVR4 package)	139.39 MB	jdk-7u51-solaris-sparc.tar.Z
Solaris SPARC	98.19 MB	jdk-7u51-solaris-sparc.tar.gz
Solaris SPARC 64-bit (SVR4 package)	23.94 MB	jdk-7u51-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	18.33 MB	jdk-7u51-solaris-sparcv9.tar.gz
Windows x86	123.64 MB	jdk-7u51-windows-i586.exe
Windows x64	125.46 MB	jdk-7u51-windows-x64.exe

AÑADIR VARIABLE DE ENTORNO



PASO # 2 DESCARGAR E INSTALAR ANDROID STUDIO

- Descarga la versión más reciente de Android studio de su sitio oficial
- Se tiene que indicar también las rutas donde se quiere instalar tanto Android Studio como el SDK de Android. Para evitar posibles problemas futuros la recomendación es seleccionar **rutas que no contengan espacios en blanco.**



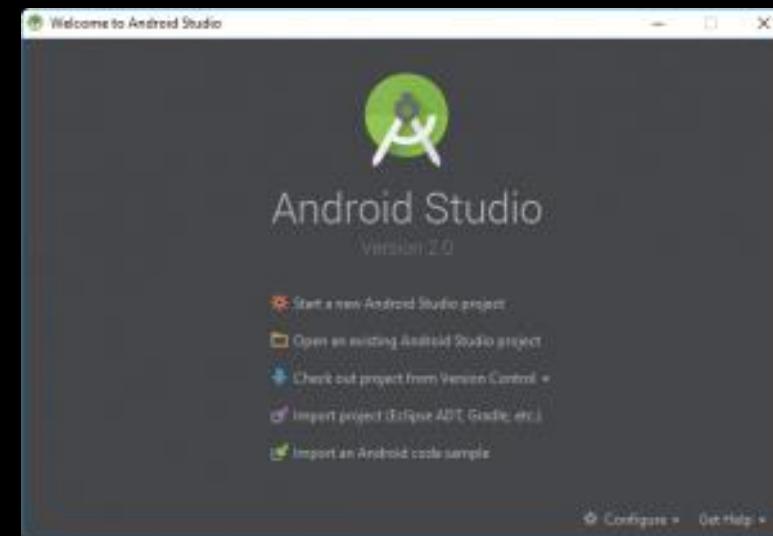
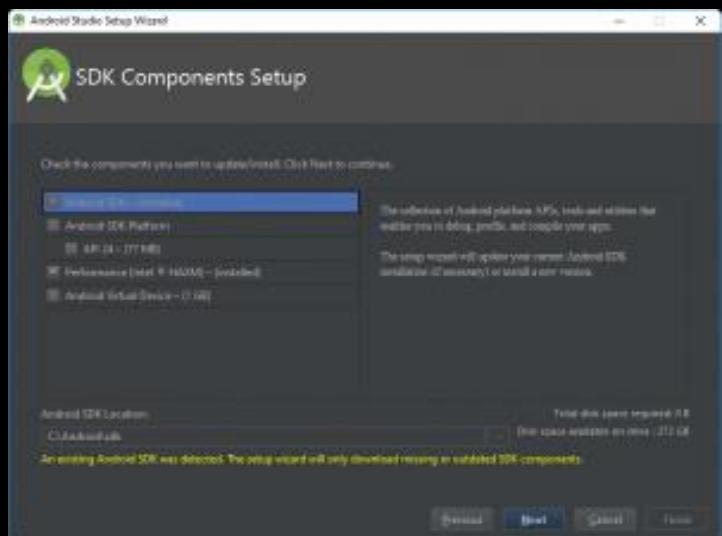
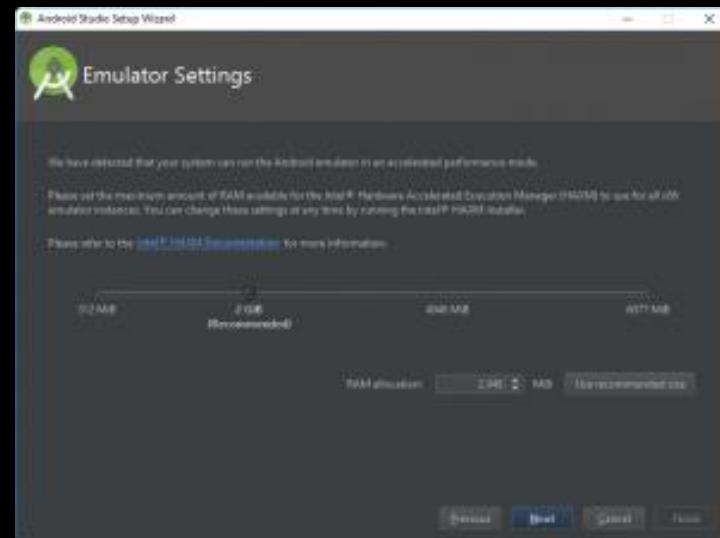
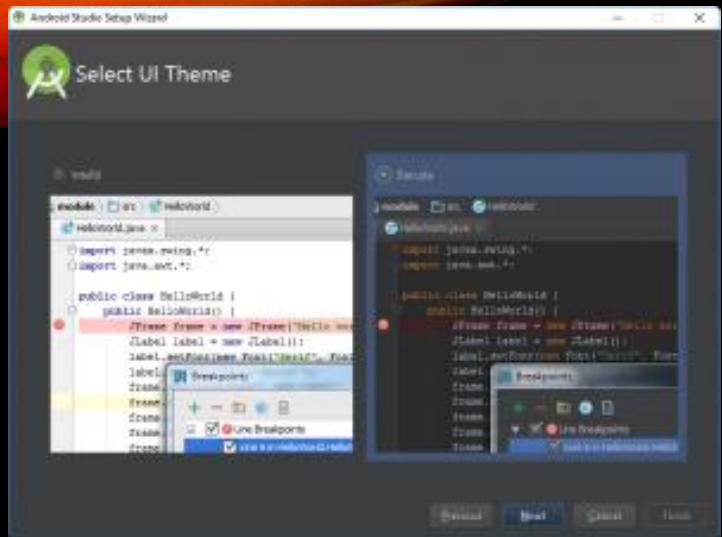
CONSIDERACIONES:

Durante la instalación se tiene que indicar también las rutas donde se quiere instalar tanto Android Studio como el SDK de Android.

"I do not have a previous versión" para una instalación nueva y sigas las indicaciones.

Tiene que decidir el *tema visual* que utilizará la aplicación.

Seleccione los componentes a instalar



CONFIGURACIÓN DEL ENTORNO DE DESARROLLO



PASO # 3 CONFIGURACIÓN INICIAL

Asegurar de que están correctamente configuradas las rutas a los SDK de Java y Android.



Para ello:

Ir a menú «Configure» de la pantalla de bienvenida, entrar en el submenú «Project Defaults» y pulsar sobre la opción «Project Structure».

En la ventana de opciones que aparece, revisar el apartado **SDK Location** asegurándonos de que tenemos correctamente configuradas las rutas al JDK y al SDK de Android

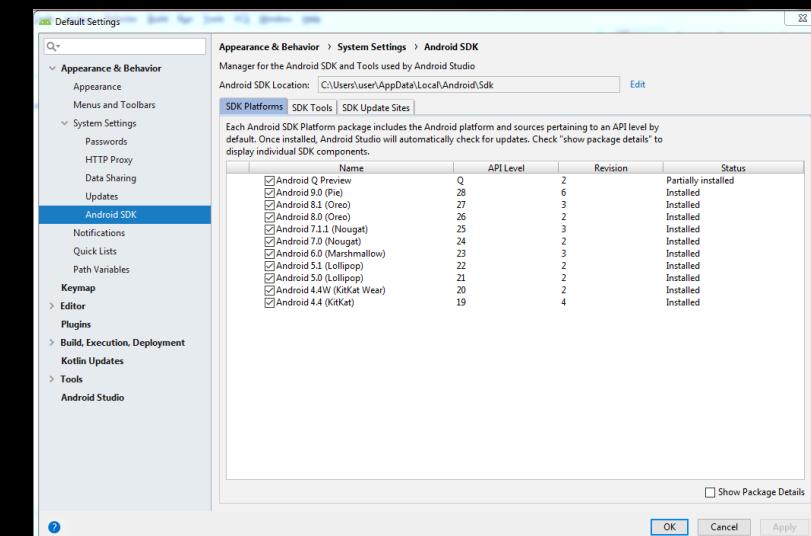
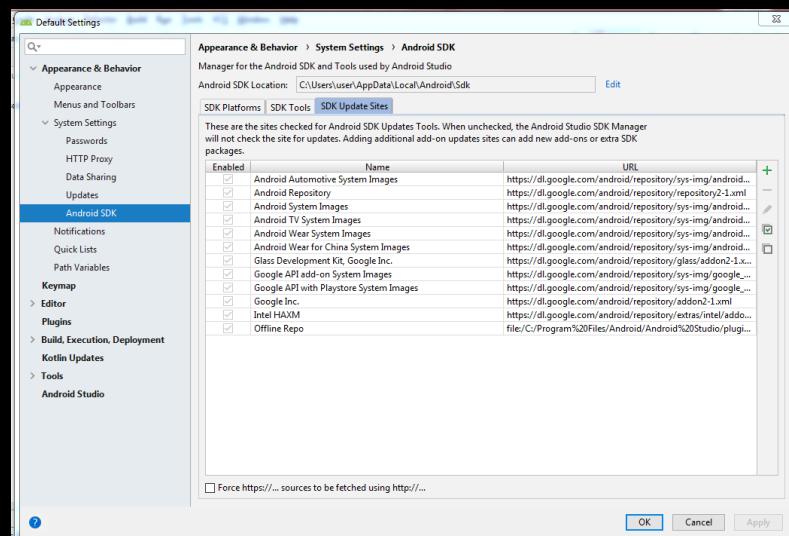
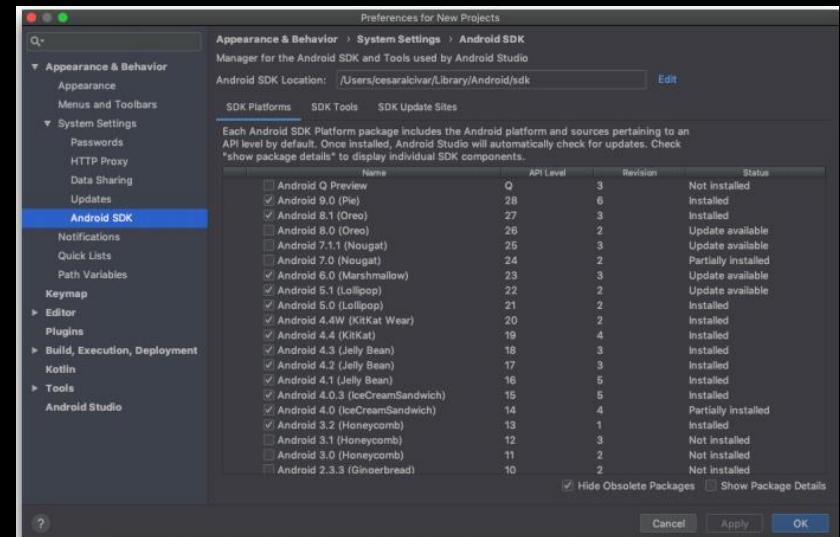
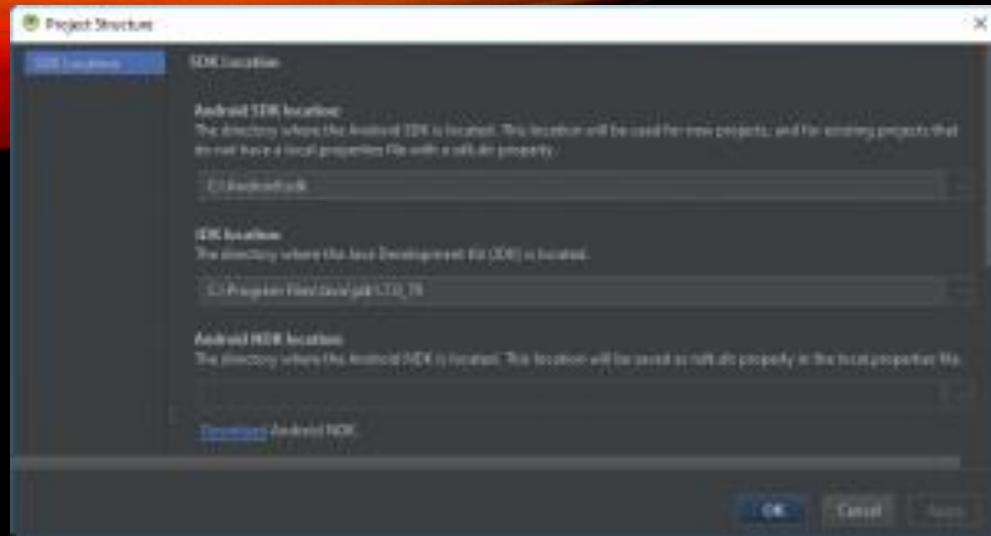
PASO #4 INSTALAR/ACTUALIZAR COMPONENTES DEL SDK DE ANDROID



SE DEBE ACCEDER DE NUEVO AL MENÚ «CONFIGURE» Y PULSAR ESTA VEZ SOBRE LA OPCIÓN «SDK MANAGER», LO QUE PERMITIRÁ ACCEDER AL SDK MANAGER DE ANDROID.



CON ESTA HERRAMIENTA SE PUEDE INSTALAR, DESINSTALAR, O ACTUALIZAR TODOS LOS COMPONENTES DISPONIBLES COMO PARTE DEL SDK DE ANDROID.



ANDROID SDK TOOLS

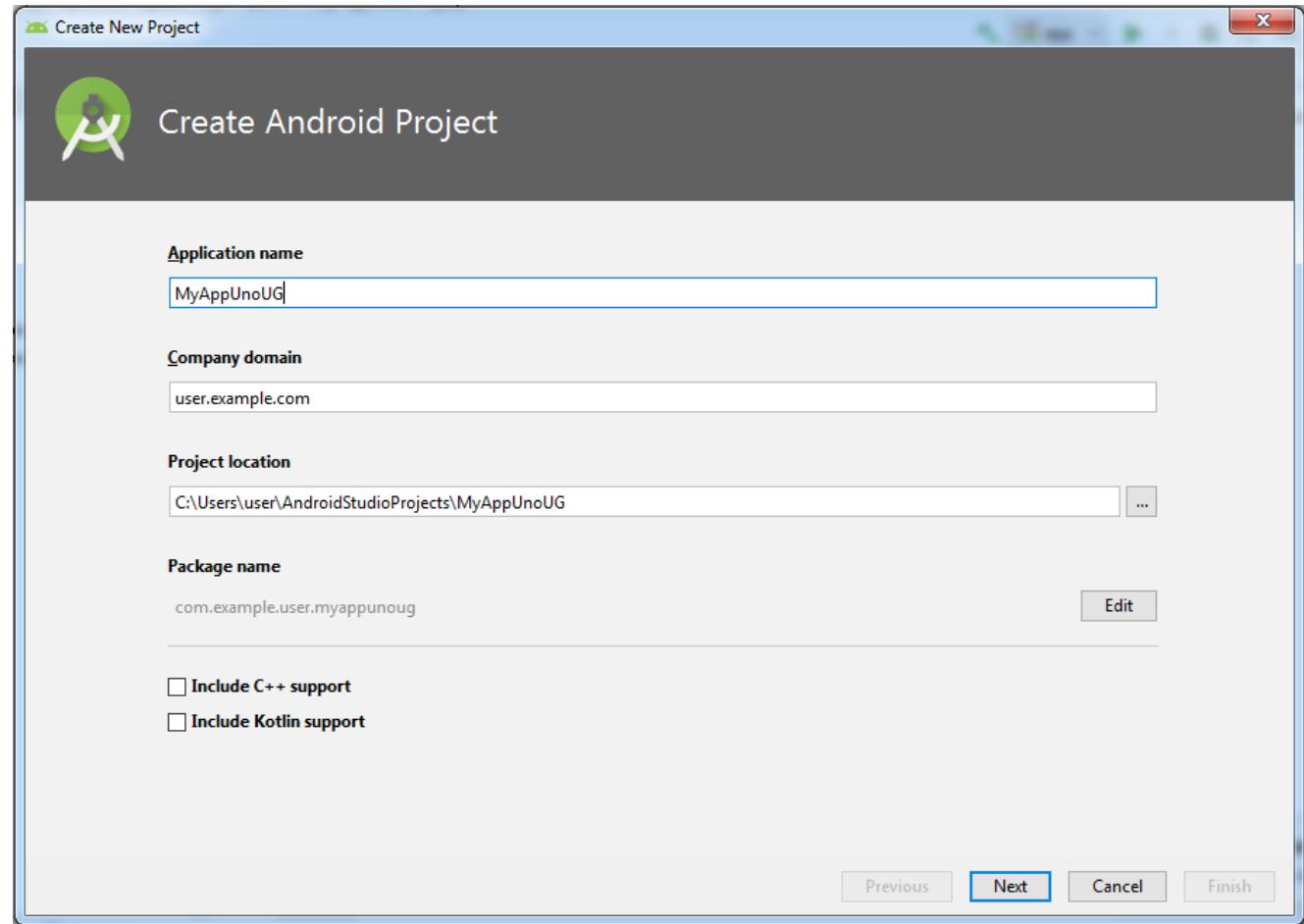


ESTRUCTURA DE UN PROYECTO ANDROID



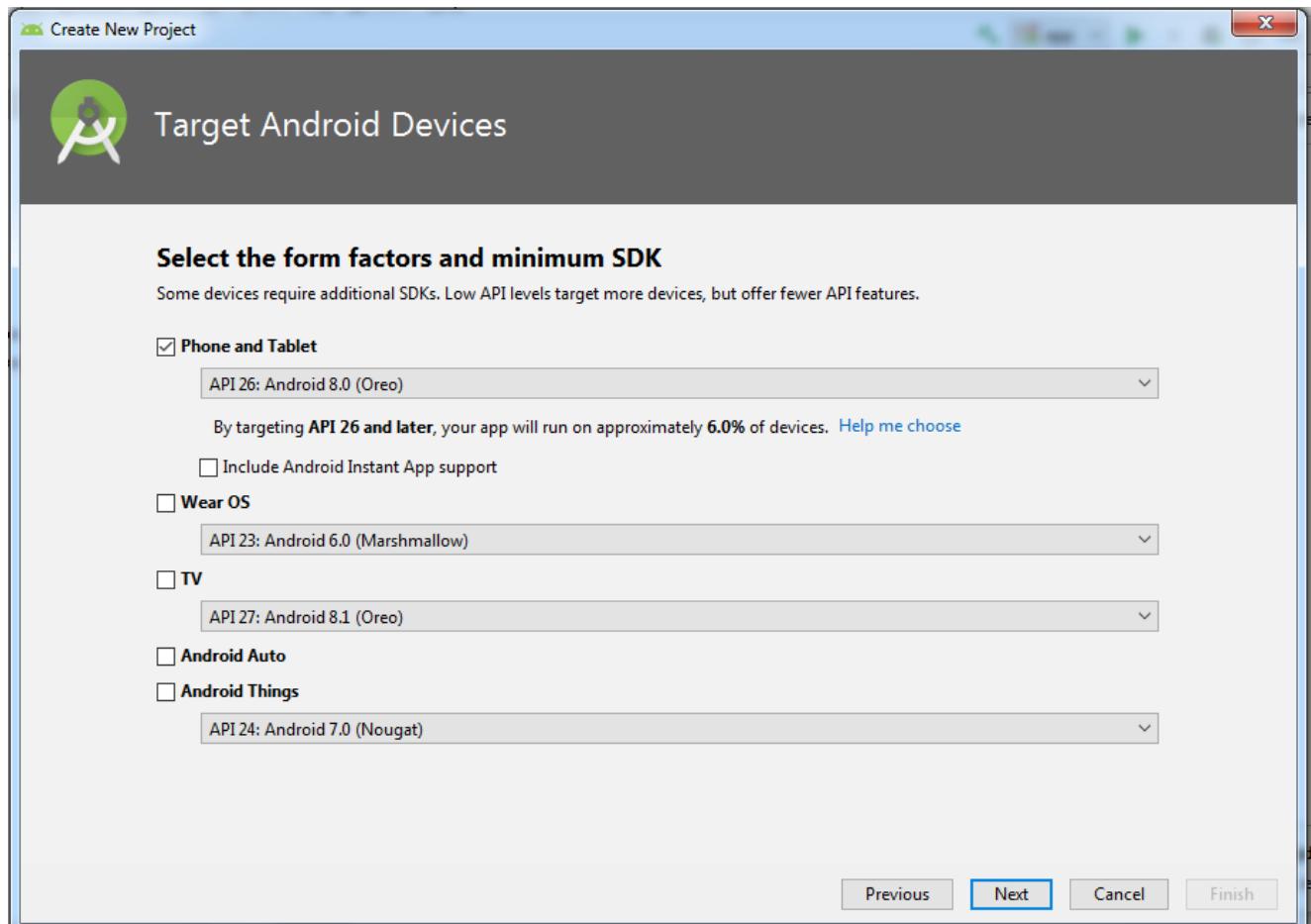
ANDROID

- En la primera pantalla indicaremos, por este orden, el nombre de la aplicación, el dominio de la compañía, y la ruta donde crear el proyecto.
- El segundo de los datos indicados tan sólo se utilizará como paquete de las clases java



ANDROID

- Se consideran aplicaciones para teléfonos y tablets.
- Seleccionar la API mínima (es decir, la versión mínima de Android) que soportará la aplicación.



ANDROID

01

La versión mínima que seleccionemos en esta pantalla implicará que nuestra aplicación se pueda ejecutar en más o menos dispositivos.

02

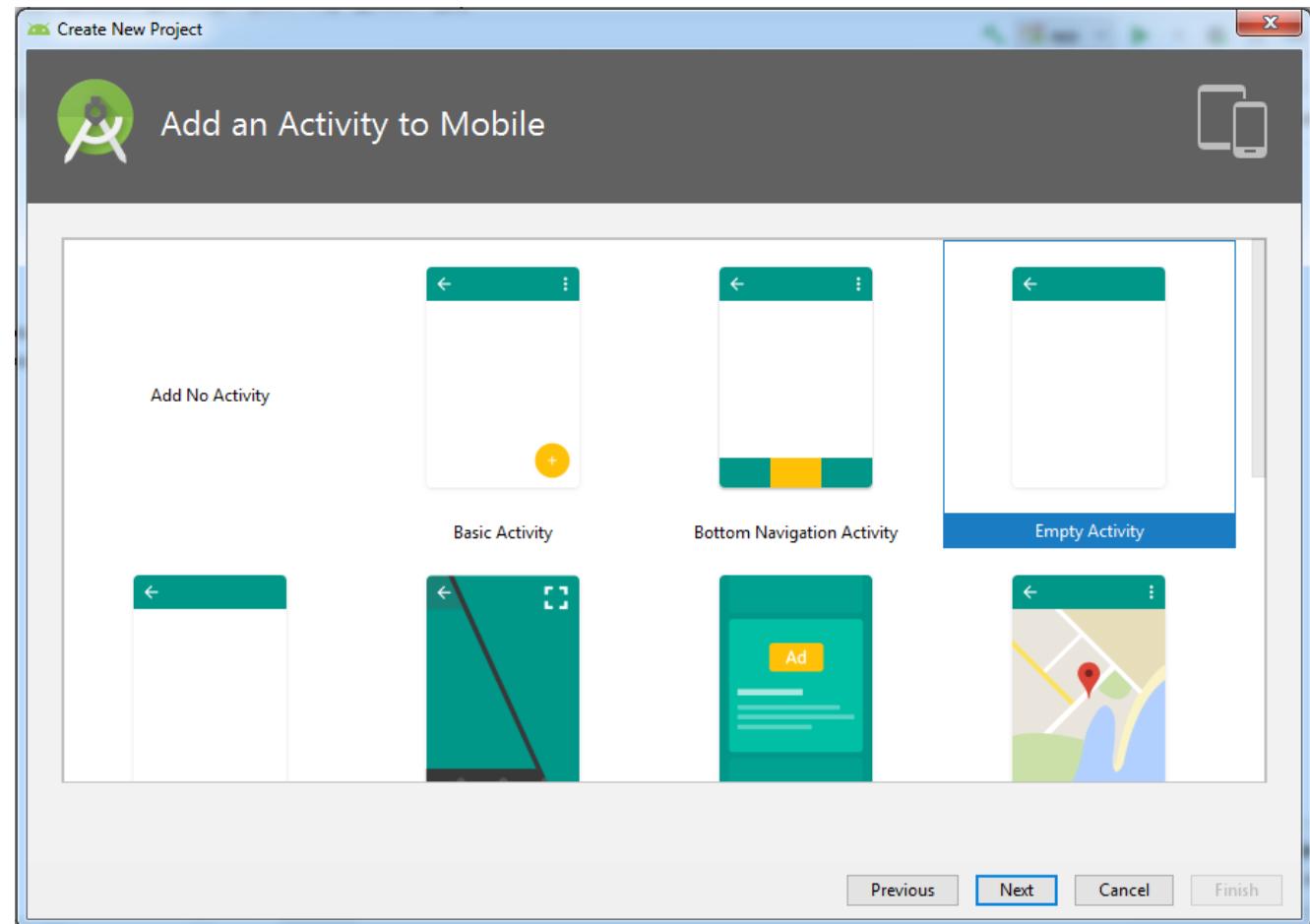
De esta forma, cuanto menor sea ésta, a más dispositivos podrá llegar nuestra aplicación, pero más complicado será conseguir que se ejecute correctamente en todas las versiones de Android.

03

Por ejemplo, en el momento de escribir este artículo, si seleccionamos como API mínima la 15 conseguiríamos cubrir un 94,0% de los dispositivos actuales

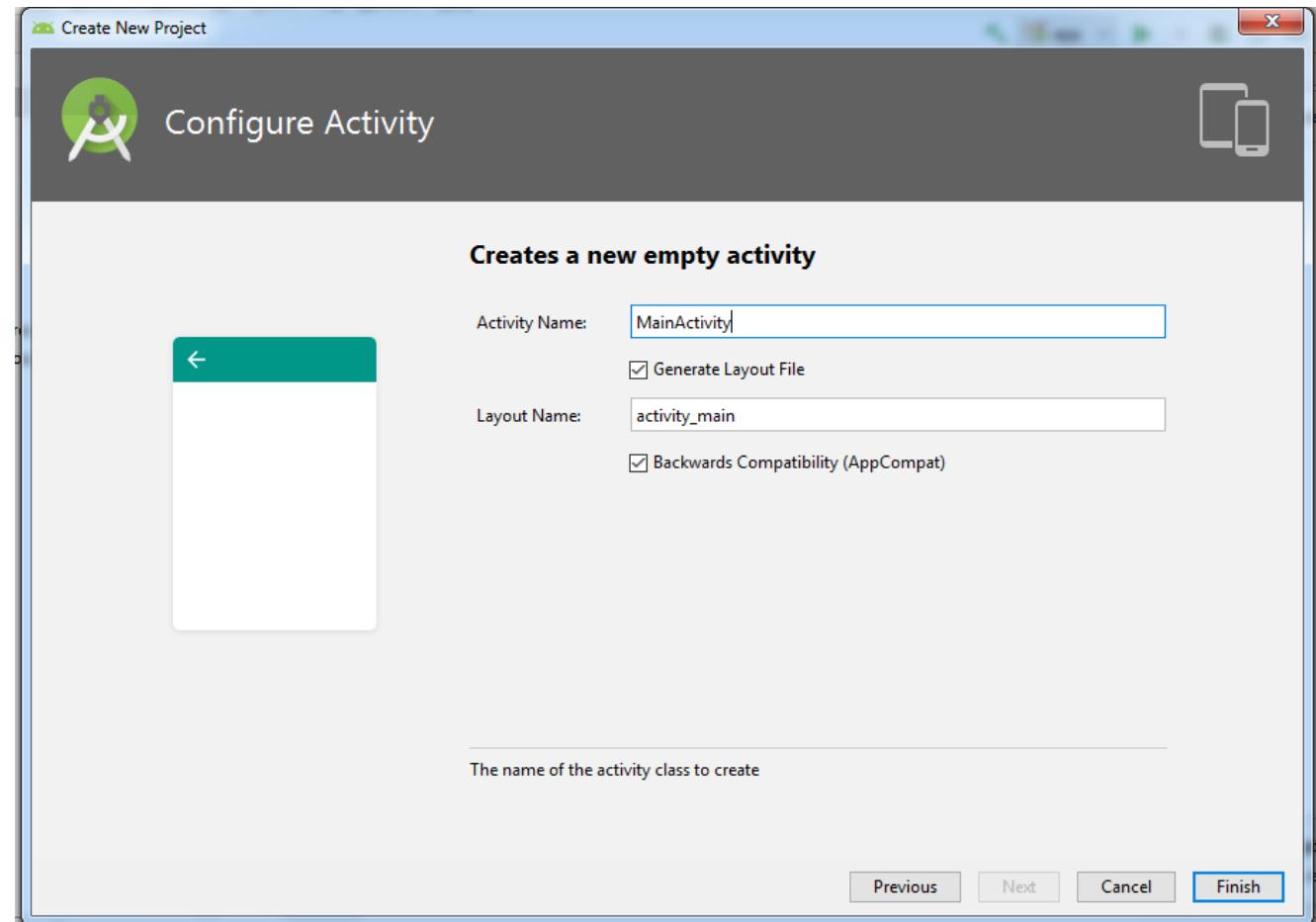
ANDROID

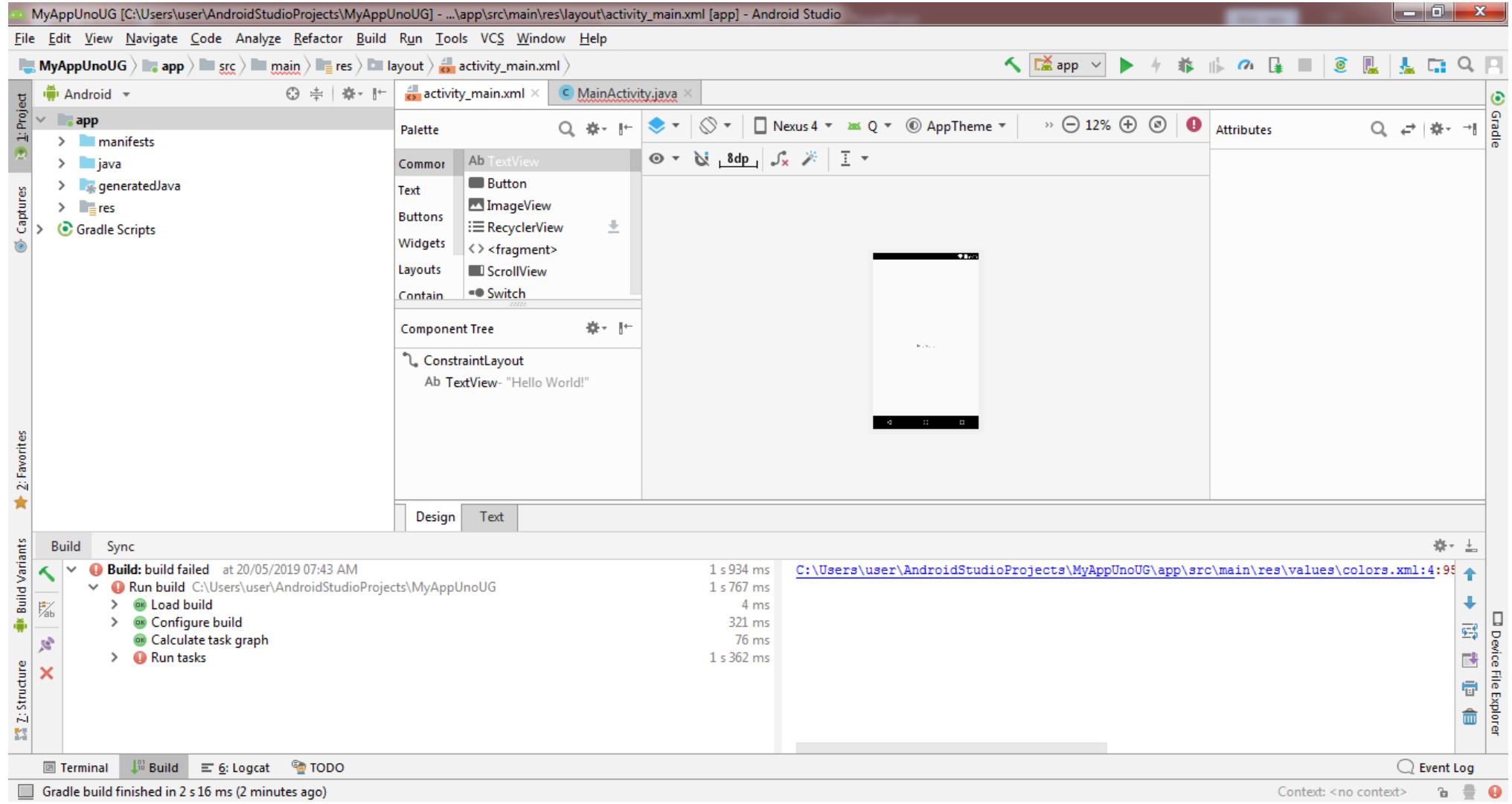
- Entenderemos por ahora que una *actividad* es una “ventana” o “pantalla” de la aplicación.
- Seleccionaremos *Empty Activity*, que es el tipo más sencillo.



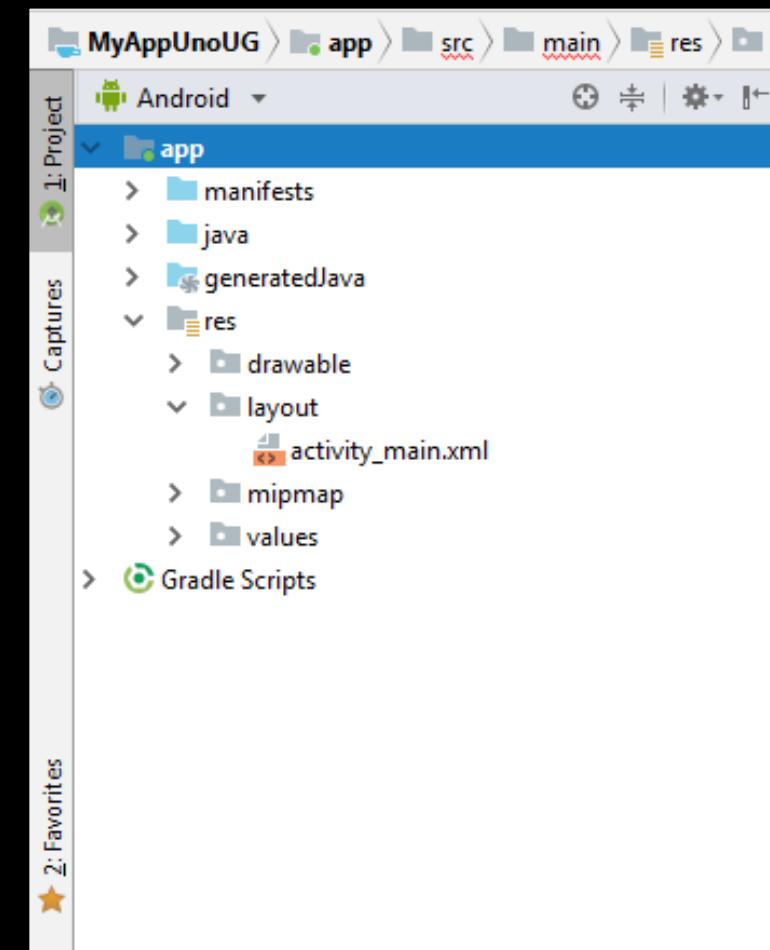
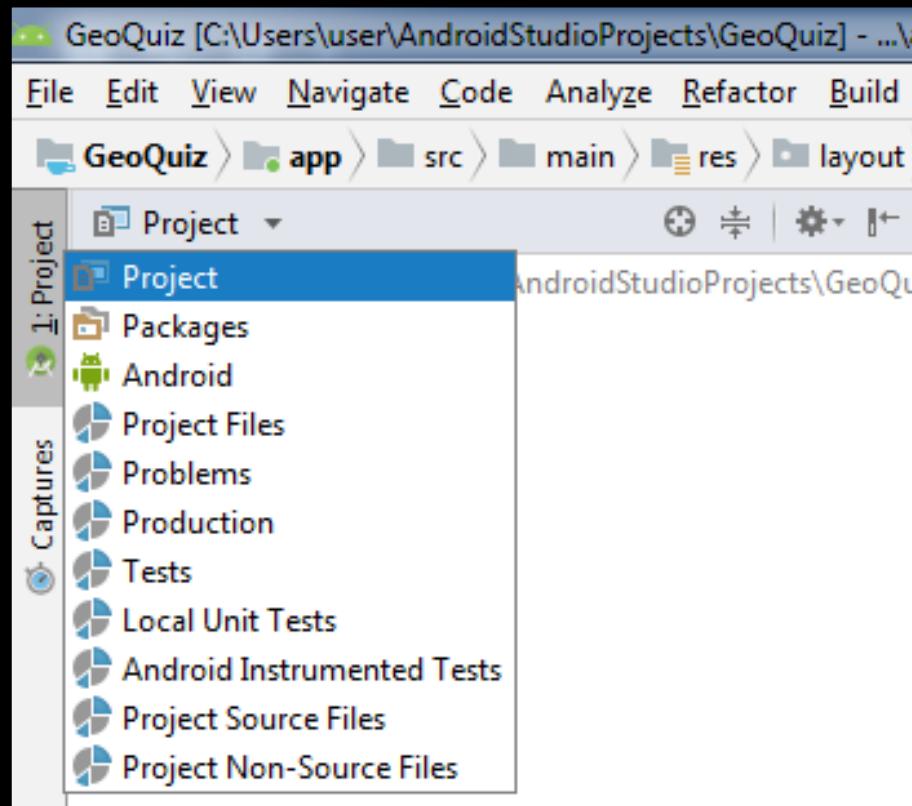
ANDROID

- Se indicará los datos asociados a esta actividad principal que se acaba de elegir, indicando el nombre de su clase java asociada (*Activity Name*) y el nombre de su *layout xml*.
- Puede optar por dejar los valores por defecto.





ANDROID: (VISTA PROJECT) – (VISTA ANDROID)



ANDROID: ELEMENTOS PRINCIPALES DE LA ESTRUCTURA

01

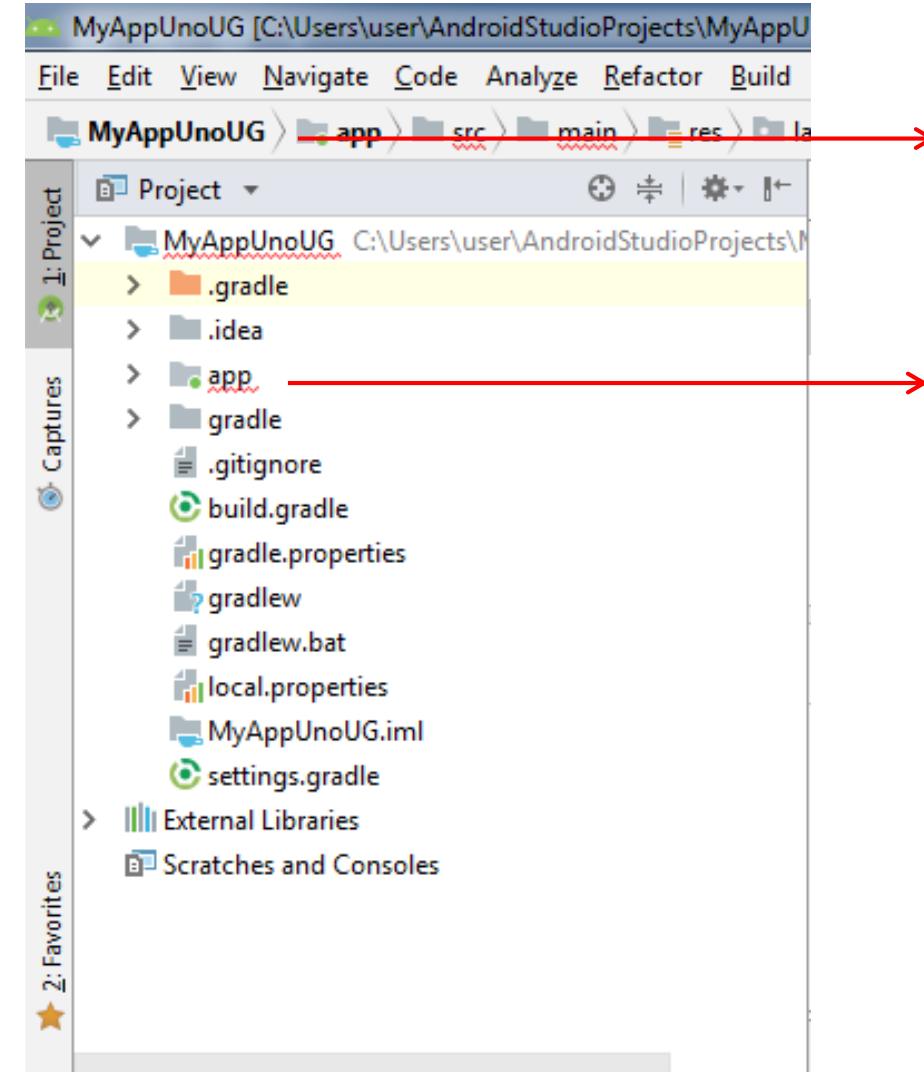
Se debe distinguir son los conceptos de **proyecto** y **módulo**

02

La entidad **proyecto** es única, y engloba a todos los demás elementos.

03

Dentro de un proyecto se pueden incluir varios **módulos**, que pueden representar aplicaciones distintas, versiones diferentes de una misma aplicación



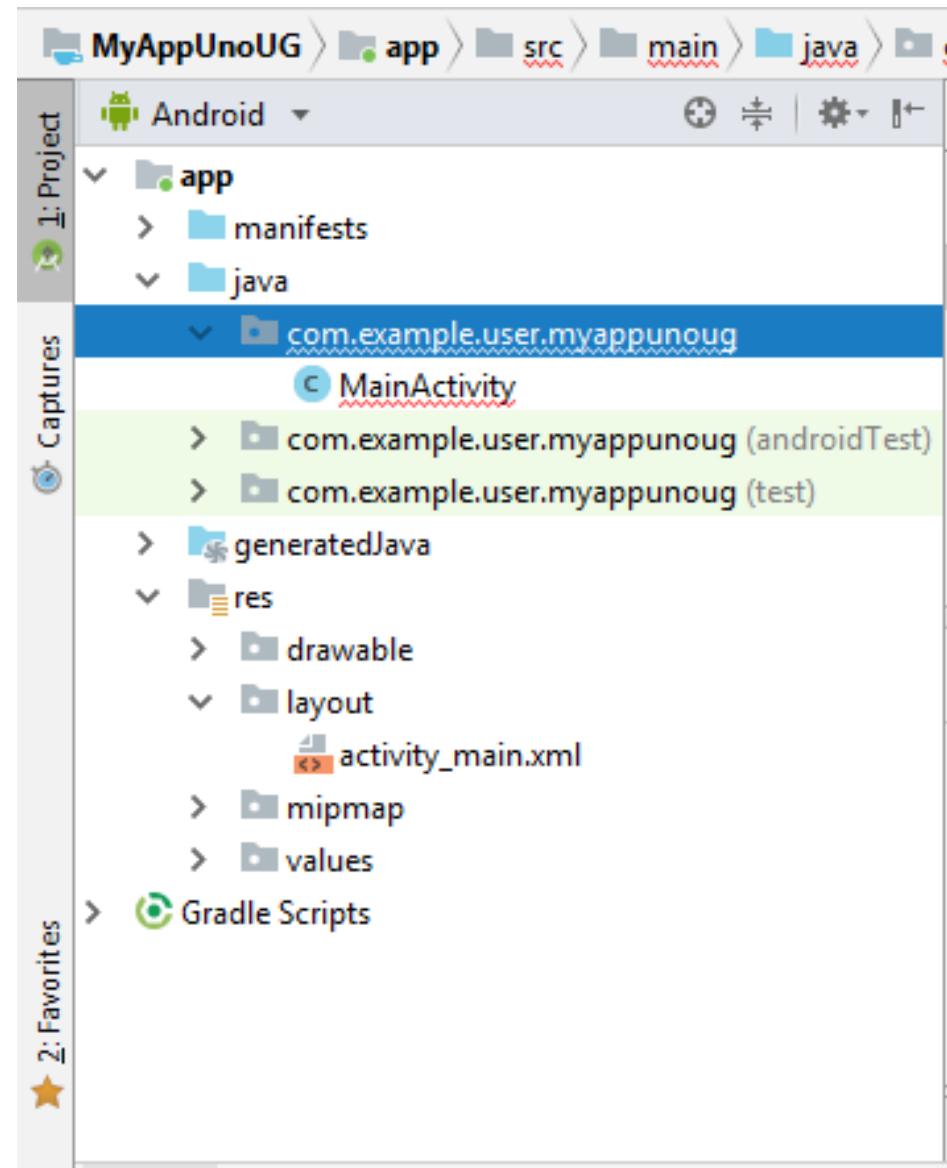
Proyecto

Módulo

ANDROID: CONTENIDOS PRINCIPALES DEL MÓDULO

Carpeta /app/src/main/java

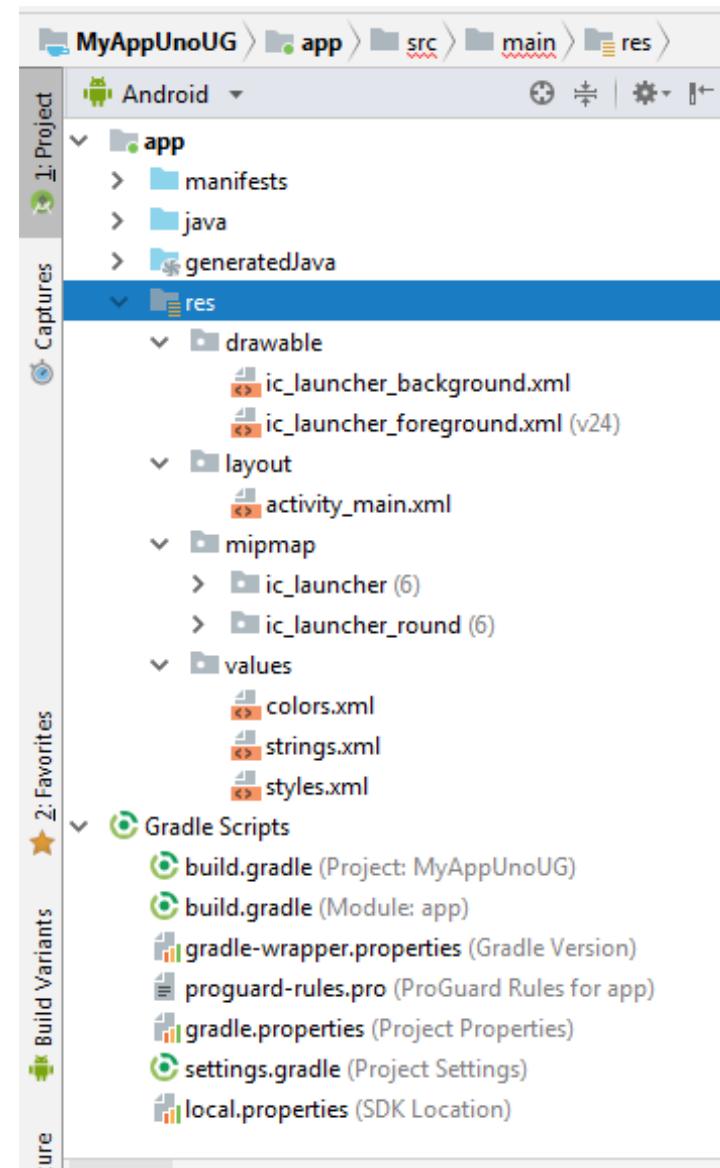
- Esta carpeta contendrá todo el **código fuente de la aplicación**, clases auxiliares, etc. Inicialmente, Android Studio creará por nosotros el código básico de la pantalla (*actividad* o *activity*) principal de la aplicación



ANDROID: CONTENIDOS PRINCIPALES DEL MÓDULO

Carpeta /app/src/main/res/

- Contiene todos los **ficheros de recursos necesarios para el proyecto**: imágenes, layouts, cadenas de texto, etc.



ANDROID: CONTENIDOS PRINCIPALES DEL MÓDULO

- Carpeta **/app/src/main/res/**

Carpeta	Descripción
<code>/res/drawable</code>	Contiene las imágenes y otros elementos gráficos usados por la aplicación. Para poder definir diferentes recursos dependiendo de la resolución y densidad de la pantalla del dispositivo se suele dividir en varias subcarpetas:

	<ul style="list-style-type: none">• <code>/drawable</code> (recursos independientes de la densidad)• <code>/drawable-ldpi</code> (densidad baja)• <code>/drawable-mdpi</code> (densidad media)• <code>/drawable-hdpi</code> (densidad alta)• <code>/drawable-xhdpi</code> (densidad muy alta)• <code>/drawable-xxhdpi</code> (densidad muy muy alta :)
--	---

ANDROID: CONTENIDOS PRINCIPALES DEL MÓDULO

- Carpeta **/app/src/main/res/**

Carpeta	Descripción
<code>/res/mipmap/</code>	Contiene los iconos de lanzamiento de la aplicación (el icono que aparecerá en el menú de aplicaciones del dispositivo) para las distintas densidades de pantalla existentes. Al igual que en el caso de las carpetas <code>/drawable</code> , se dividirá en varias subcarpetas dependiendo de la densidad de pantalla: <ul style="list-style-type: none">• <code>/mipmap-mdpi</code>• <code>/mipmap-hdpi</code>• <code>/mipmap-xhdpi</code>• ...
<code>/res/layout/</code>	Contiene los ficheros de definición XML de las diferentes pantallas de la interfaz gráfica. Para definir distintos <i>layouts</i> dependiendo de la orientación del dispositivo se puede dividir también en subcarpetas: <ul style="list-style-type: none">• <code>/layout</code> (vertical)• <code>/layout-land</code> (horizontal)

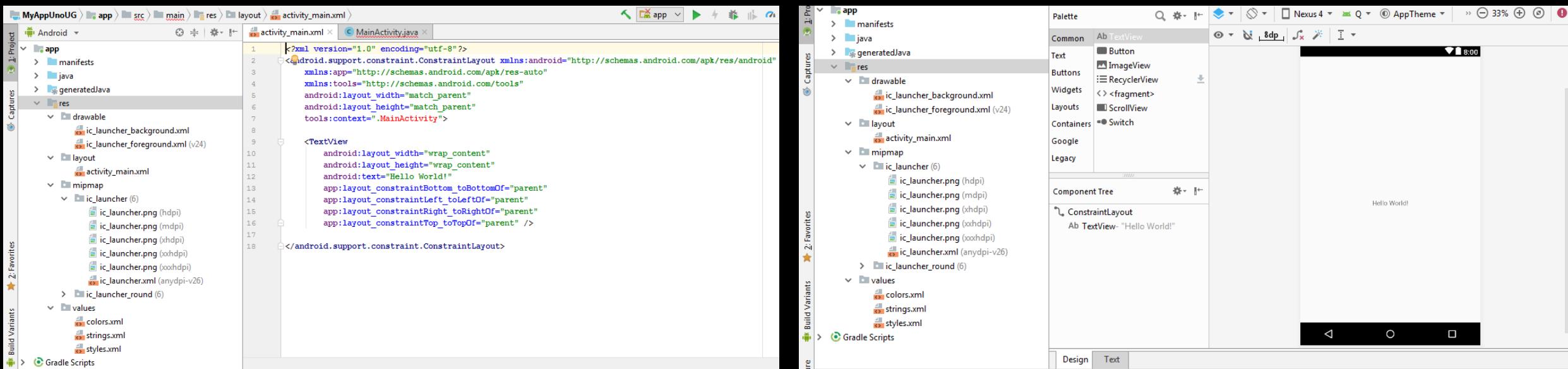
ANDROID: CONTENIDOS PRINCIPALES DEL MÓDULO

- Carpeta /app/src/main/res/

Carpeta	Descripción
/res/anim/ /res/animator/	Contienen la definición de las animaciones utilizadas por la aplicación.
/res/color/	Contiene ficheros XML de definición de listas de colores según estado.
/res/menu/	Contiene la definición XML de los menús de la aplicación.
/res/xml/	Contiene otros ficheros XML de datos utilizados por la aplicación.
/res/raw/	Contiene recursos adicionales, normalmente en formato distinto a XML, que no se incluyan en el resto de carpetas de recursos.
/res/values/	Contiene otros ficheros XML de recursos de la aplicación, como por ejemplo cadenas de texto (<i>strings.xml</i>), estilos (<i>styles.xml</i>), colores (<i>colors.xml</i>), arrays de valores (<i>arrays.xml</i>), tamaños (<i>dimens.xml</i>), etc.

ANDROID: CONTENIDOS PRINCIPALES DEL MÓDULO

«Design» y «Text» podremos alternar entre el editor gráfico y el editor XML



ANDROID: CONTENIDOS PRINCIPALES DEL MÓDULO

Fichero /app/src/main/AndroidManifest.xml



Contiene la definición en XML de muchos de los aspectos principales de la aplicación, como por ejemplo su identificación (nombre, icono, ...), sus componentes (pantallas, servicios, ...), o los permisos necesarios para su ejecución.

ANDROID: CONTENIDOS PRINCIPALES DEL MÓDULO

FICHERO /APP/BUILD.GRADLE

CONTIENE INFORMACIÓN NECESARIA PARA LA COMPILACIÓN DEL PROYECTO, POR EJEMPLO LA VERSIÓN DEL SDK DE ANDROID UTILIZADA PARA COMPILAR, LA MÍNIMA VERSIÓN DE ANDROID QUE SOPORTARÁ LA APLICACIÓN, REFERENCIAS A LAS LIBRERÍAS EXTERNAS UTILIZADAS, ETC.

ANDROID: CONTENIDOS PRINCIPALES DEL MÓDULO

Carpeta /app/libs

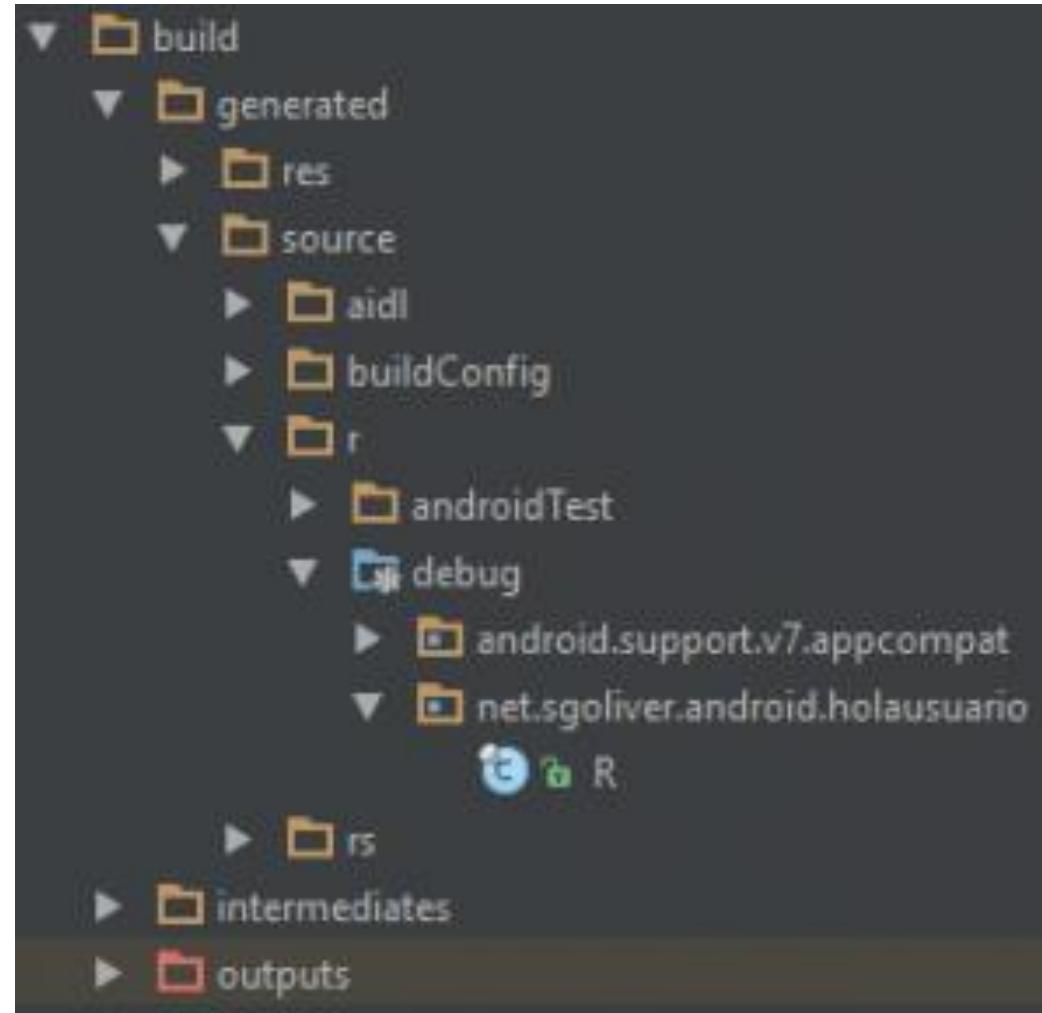


Puede contener las *librerías java externas* (ficheros .jar) que utilice la aplicación.

Normalmente no incluiremos directamente aquí ninguna librería, sino que haremos referencia a ellas en el fichero *build.gradle*

ANDROID: CONTENIDOS PRINCIPALES DEL MÓDULO

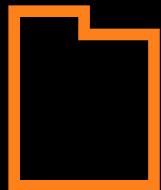
- **Carpeta /app/build/**
- Contiene una serie de elementos de código generados automáticamente al compilar el proyecto.
- **Importante:** no se modifican manualmente bajo ninguna circunstancia.



COMPONENTES DE UNA APLICACIÓN ANDROID



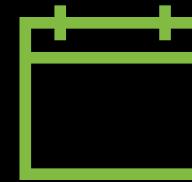
ELEMENTOS BÁSICOS DE UNA APLICACIÓN



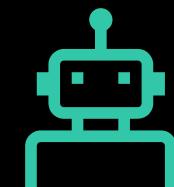
Ventana



Control



Eventos

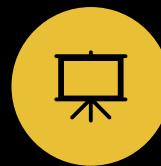


Servicios como los elementos básicos en la construcción de una aplicación.

COMPONENTES DE UNA APLICACIÓN ANDROID



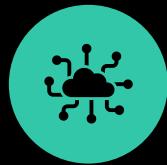
Activity



View



Service



Content Provider



Broadcast Receiver



Widget



Intent

ACTIVITY



Las actividades (*activities*) representan el componente principal de la interfaz gráfica de una aplicación Android.

Se puede pensar como una ventana o pantalla en cualquier otro lenguaje visual.

VIEW

1

Las vistas (**view**) son los componentes básicos con los que se construye la interfaz gráfica de la aplicación.

2

Controles básicos, como cuadros de texto, botones, listas desplegables o imágenes

SERVICE

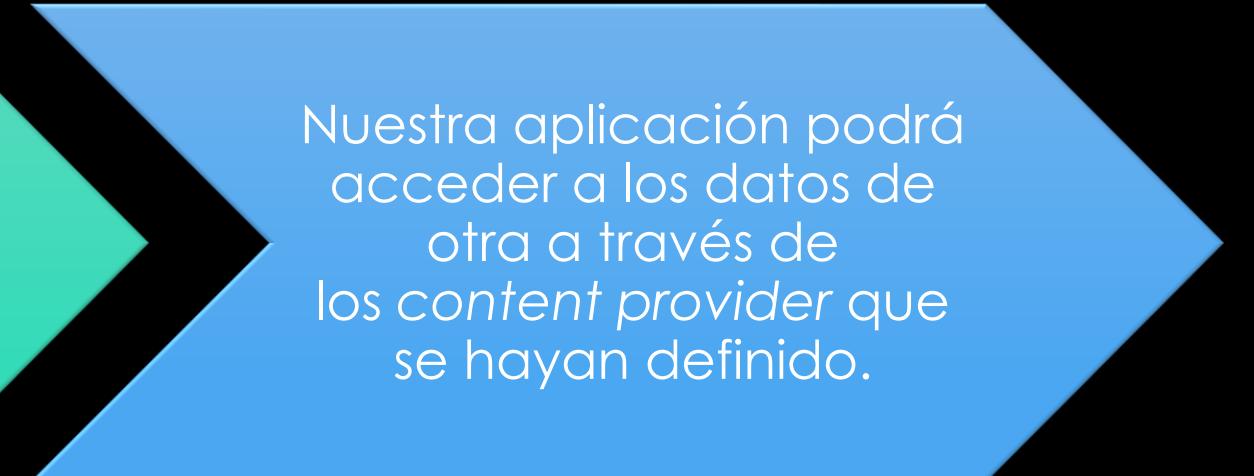
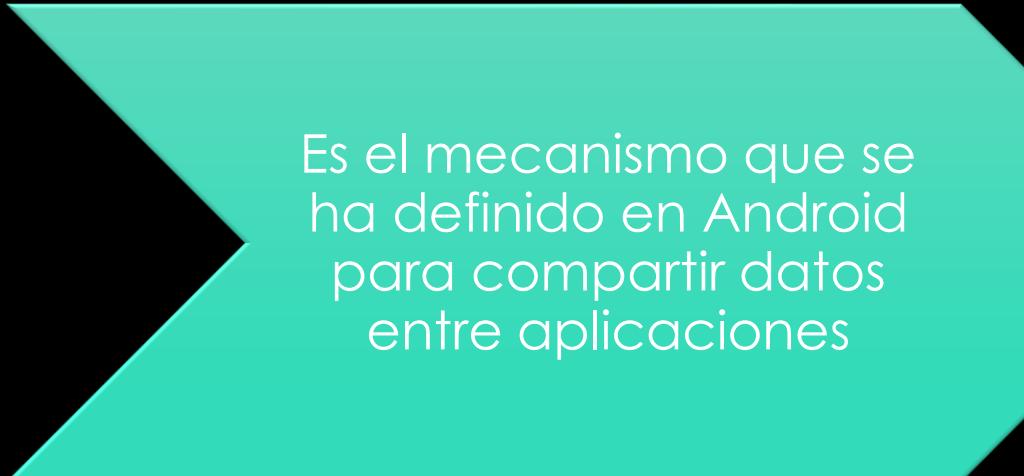
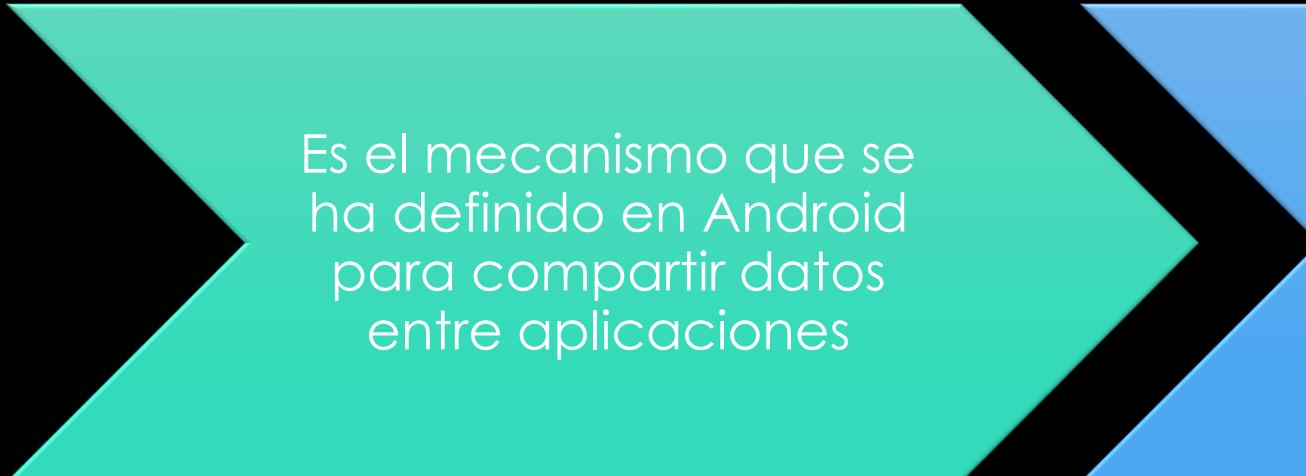
01

Los servicios (**service**) son componentes sin interfaz gráfica que se ejecutan en segundo plano.

02

Los servicios pueden realizar cualquier tipo de acciones, por ejemplo actualizar datos, lanzar notificaciones

CONTENT PROVIDER



Es el mecanismo que se ha definido en Android para compartir datos entre aplicaciones

Nuestra aplicación podrá acceder a los datos de otra a través de los *content provider* que se hayan definido.

BROADCAST RECEIVER

Un *broadcast receiver* es un componente destinado a detectar y reaccionar ante determinados mensajes o **eventos globales generados por el sistema** (por ejemplo: “Batería baja”, “SMS recibido”, “Tarjeta SD insertada”, ...)



Broadcast, es decir, no dirigidos a una aplicación concreta sino a cualquiera que quiera escucharlo).

WIDGET

1

Los **widgets** son elementos visuales, normalmente interactivos, que pueden mostrarse en la pantalla principal y recibir actualizaciones periódicas.

2

Permiten mostrar información de la aplicación al usuario directamente sobre la pantalla principal.

INTENT

01

Un *intent* es el elemento básico de comunicación entre los distintos componentes Android

02

Son los mensajes o peticiones que son enviados entre los distintos componentes de una aplicación o entre distintas aplicaciones



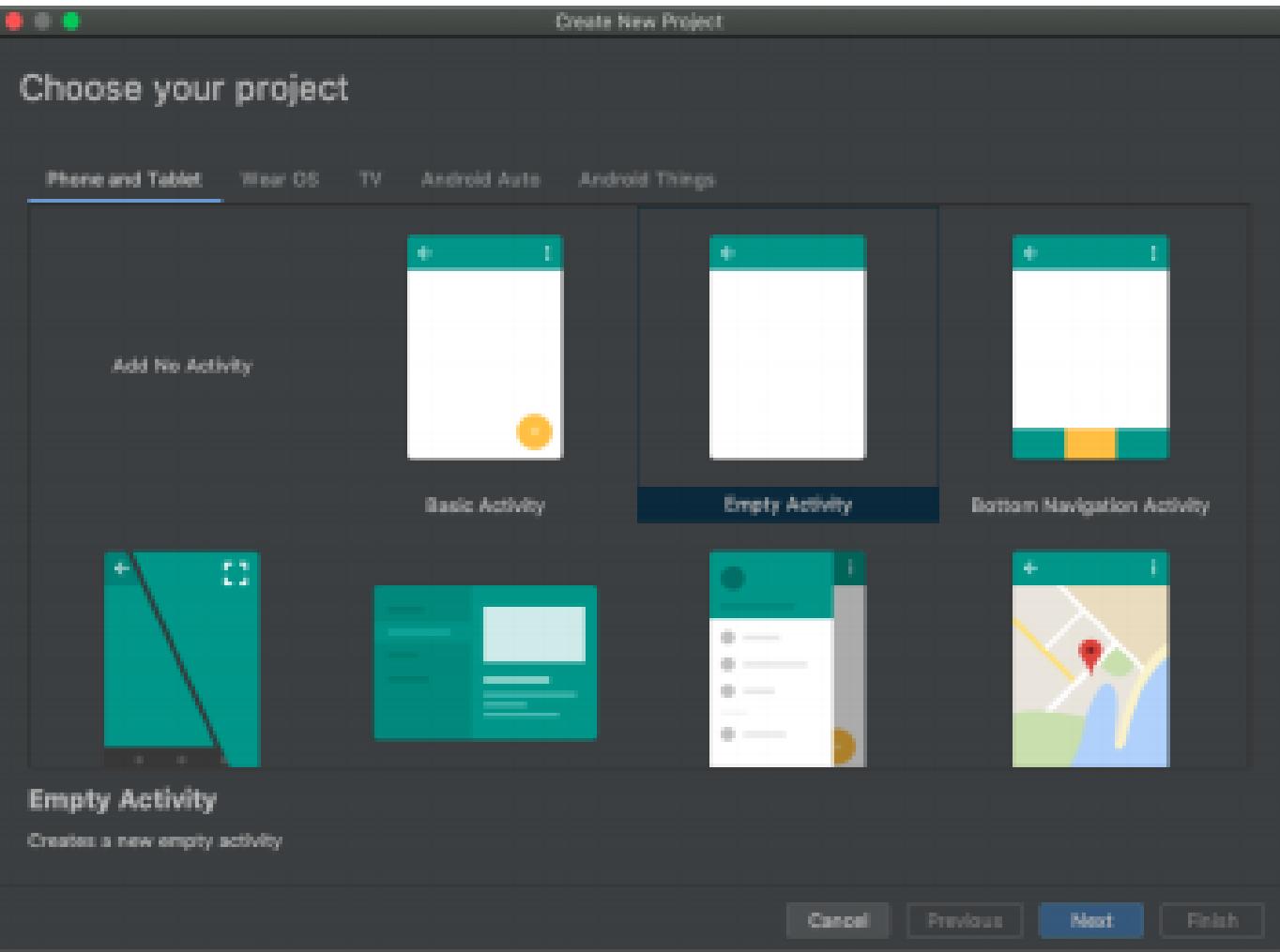
Clases, Layouts Vistas y Recursos

- Aplicación en Android Studio

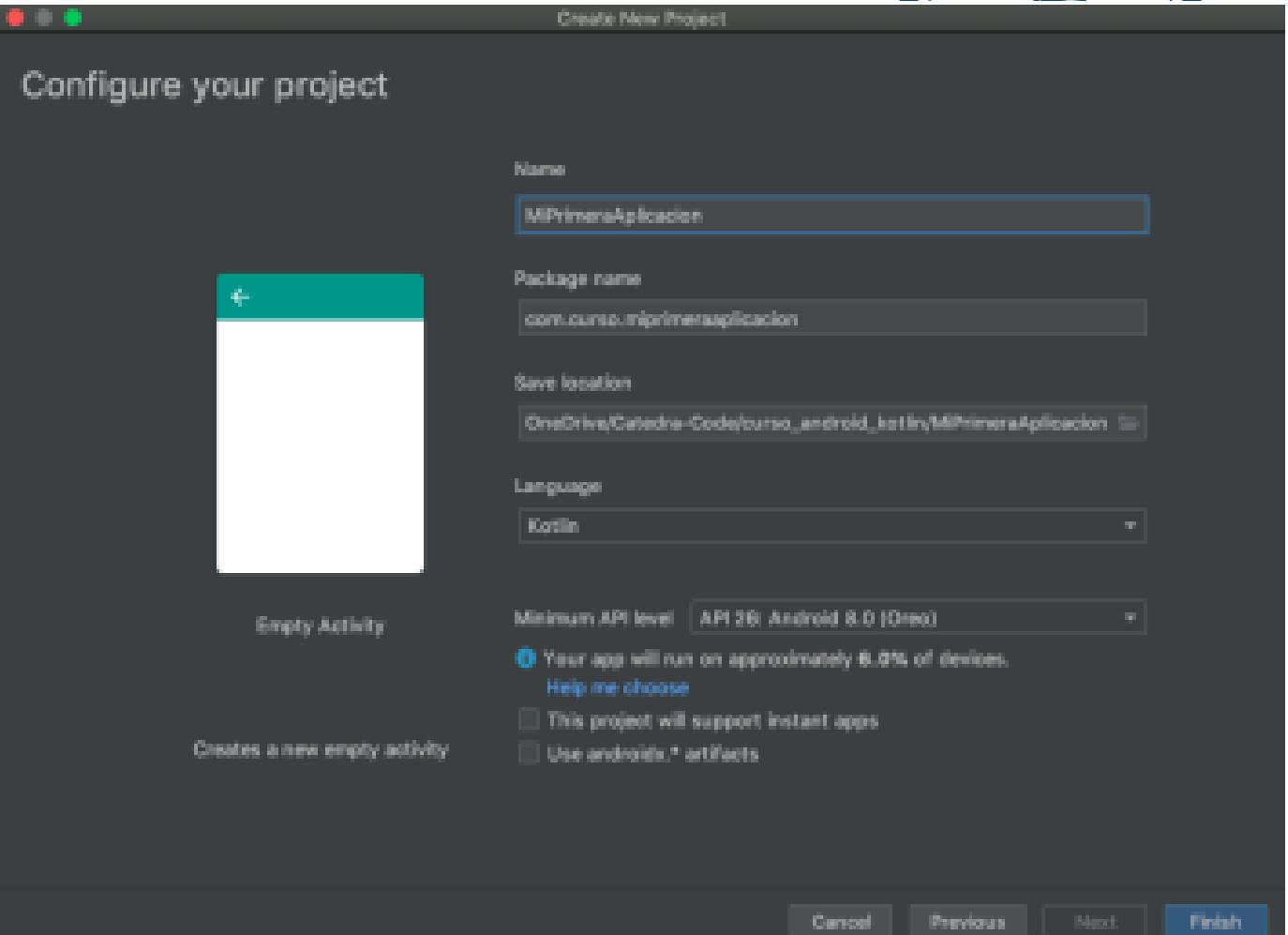
Aplicación

- GeoQuiz es una aplicación que evalúa tus conocimientos de la geografía ecuatoriana a través de preguntas con respuestas de “Verdadero” o “Falso”
- Este proyecto nos introducirá en algunos aspectos claves del desarrollo de Android, tales como buenas prácticas de componente y arquitectura

Aplicación



Aplicación



Cosas a tener en cuenta

- **Package name** es un identificador único en el ecosistema android. Puede ser cualquier nombre, pero tradicionalmente se establece una URL en reverso de tu dominio, por ejemplo, www.miempresa.com, entonces sería com.miempresa.aplicación.
- **Save Location** establece la localización del proyecto.
- **Language** seleccionamos Java
- **Minimum API Level** seleccionamos API 26: Android 8.0 (Oreo).

Cosas a tener en cuenta

- Android Studio permite trabajar con versiones antiguas, muchas de los aspectos de seguridad y privacidad fueron introducidos en el API 25.
- Para mejorar la seguridad, Google anuncio que a partir de agosto 2018 solo permitirán publicar nuevas aplicación utilizando el API 26 o superior

Cosas a tener en cuenta

- La vista de la **izquierda** es la **project tool window** y a la **derecha** tenemos el **editor de código**

The screenshot shows the Android Studio interface. On the left, the Project Tool Window displays the project structure under the 'app' folder, including 'manifests', 'java' (with 'com.example.user.geoquiz' containing 'MainActivity' and 'Pregunta'), 'generatedJava', 'res', and 'Gradle Scripts'. On the right, the Code Editor shows the 'MainActivity.java' file. The code defines a class 'MainActivity' extending 'AppCompatActivity'. It contains a private array 'preguntas' with several questions about Ecuador. The code also initializes buttons for 'True' (btnVerdadero), 'False' (btnFalso), 'Next' (btnSiguiente), 'Previous' (btnAtras), and a question text view (textpregunta). The 'onCreate' method sets the content view to 'activity_main' and initializes the button variables.

```
Quiz [C:\Users\user\AndroidStudioProjects\GeoQuiz] - ...\\app\\src\\main\\java\\com\\example\\user\\geoquiz\\MainActivity.java [app] - Android Studio
  View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
  app > strings.xml > MainActivity.java > colors.xml > styles.xml > Preguntas
  Android > strings.xml > MainActivity.java > colors.xml > styles.xml > Preguntas

  app
    manifests
    java
      com.example.user.geoquiz
        MainActivity
        Pregunta
        > com.example.user.geoquiz (androidTest)
        > com.example.user.geoquiz (test)
        generatedJava
        res
    Gradle Scripts

  10
  11 public class MainActivity extends AppCompatActivity {
  12
  13     private Pregunta[] preguntas = {
  14         new Pregunta("Quito es la capital del Ecuador", "responde"),
  15         new Pregunta("Guayaquil es conocida como la perla del",
  16             "norte"),
  17         new Pregunta("Ecuador esta ubicado en America del Nor",
  18             "oeste"),
  19         new Pregunta("Cuenca es una ciudad del sur de Ecuador",
  20             "responde"),
  21         new Pregunta("Manta pertenece a la provincia del Guay",
  22             "nape"),
  23         new Pregunta("Manta pertenece a la provincia del Guay",
  24             "nape");
  25
  26     int posicionActual = 0;
  27     private Button btnVerdadero;
  28     private Button btnFalso;
  29     private ImageButton btnSiguiente;
  30     private ImageButton btnAtras;
  31     private TextView textpregunta;
  32
  33     @Override
  34     protected void onCreate(Bundle savedInstanceState) {
  35         super.onCreate(savedInstanceState);
  36         setContentView(R.layout.activity_main);
  37
  38         btnVerdadero = findViewById(R.id.button_verdadero);
  39         btnFalso = findViewById(R.id.button_falso);
  40         btnSiguiente = findViewById(R.id.button_siguiente);
  41         textpregunta = findViewById(R.id.text_pregunta);
  42         btnAtras = findViewById(R.id.button_atras);
  43
  44     }
  45
  46     MainActivity > onCreate()
```

DDO 6: Logcat Build Terminal

all files and settings

GeoQuiz [C:\Users\user\AndroidStudioProjects\GeoQuiz] - ...app\src\main\java\com\example\user\geoquiz\MainActivity.java [app] - Android Studio

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

GeoQuiz app src main java com example user geoquiz MainActivity

1: Project

2: Captures

3: Structure

4: Build Variants

5: Favorites

6: Android

7: strings.xml

8: MainActivity.java

9: colors.xml

10: styles.xml

11: Pregunta.java

12: ic_adelante.png

13: Gradle

14: 1: Project

15: 2: Captures

16: 3: Structure

17: 4: Build Variants

18: 5: Favorites

19: 6: Android

20: 7: strings.xml

21: 8: MainActivity.java

22: 9: colors.xml

23: 10: styles.xml

24: 11: Pregunta.java

25: 12: ic_adelante.png

26: 13: Gradle

27: 14: 1: Project

28: 15: 2: Captures

29: 16: 3: Structure

30: 17: 4: Build Variants

31: 18: 5: Favorites

32: 19: 6: Android

33: 20: 7: strings.xml

34: 21: 8: MainActivity.java

35: 22: 9: colors.xml

36: 23: 10: styles.xml

37: 24: 11: Pregunta.java

38: 25: 12: ic_adelante.png

39: 26: 13: Gradle

40: 27: 14: 1: Project

41: 28: 2: Captures

42: 29: 3: Structure

43: 30: 4: Build Variants

44: 31: 5: Favorites

45: 32: 6: Android

46: 33: 7: strings.xml

47: 34: 8: MainActivity.java

48: 35: 9: colors.xml

49: 36: 10: styles.xml

50: 37: 11: Pregunta.java

51: 38: 12: ic_adelante.png

52: 39: 13: Gradle

53: 54: 14: 1: Project

55: 56: 2: Captures

57: 58: 3: Structure

59: 60: 4: Build Variants

61: 62: 5: Favorites

63: 64: 6: Android

65: 66: 7: strings.xml

67: 68: 8: MainActivity.java

69: 70: 9: colors.xml

71: 72: 10: styles.xml

73: 74: 11: Pregunta.java

75: 76: 12: ic_adelante.png

77: 78: 13: Gradle

79: 80: 14: 1: Project

81: 82: 2: Captures

83: 84: 3: Structure

85: 86: 4: Build Variants

87: 88: 5: Favorites

89: 90: 6: Android

91: 92: 7: strings.xml

93: 94: 8: MainActivity.java

95: 96: 9: colors.xml

97: 98: 10: styles.xml

99: 100: 11: Pregunta.java

101: 102: 12: ic_adelante.png

103: 104: 13: Gradle

105: 106: 14: 1: Project

107: 108: 2: Captures

109: 110: 3: Structure

111: 112: 4: Build Variants

113: 114: 5: Favorites

115: 116: 6: Android

117: 118: 7: strings.xml

119: 120: 8: MainActivity.java

121: 122: 9: colors.xml

123: 124: 10: styles.xml

125: 126: 11: Pregunta.java

127: 128: 12: ic_adelante.png

129: 130: 13: Gradle

131: 132: 14: 1: Project

133: 134: 2: Captures

135: 136: 3: Structure

137: 138: 4: Build Variants

139: 140: 5: Favorites

141: 142: 6: Android

143: 144: 7: strings.xml

145: 146: 8: MainActivity.java

147: 148: 9: colors.xml

149: 150: 10: styles.xml

151: 152: 11: Pregunta.java

153: 154: 12: ic_adelante.png

155: 156: 13: Gradle

157: 158: 14: 1: Project

159: 160: 2: Captures

161: 162: 3: Structure

163: 164: 4: Build Variants

165: 166: 5: Favorites

167: 168: 6: Android

169: 170: 7: strings.xml

171: 172: 8: MainActivity.java

173: 174: 9: colors.xml

175: 176: 10: styles.xml

177: 178: 11: Pregunta.java

179: 180: 12: ic_adelante.png

181: 182: 13: Gradle

183: 184: 14: 1: Project

185: 186: 2: Captures

187: 188: 3: Structure

189: 190: 4: Build Variants

191: 192: 5: Favorites

193: 194: 6: Android

195: 196: 7: strings.xml

197: 198: 8: MainActivity.java

199: 200: 9: colors.xml

201: 202: 10: styles.xml

203: 204: 11: Pregunta.java

205: 206: 12: ic_adelante.png

207: 208: 13: Gradle

209: 210: 14: 1: Project

211: 212: 2: Captures

213: 214: 3: Structure

215: 216: 4: Build Variants

217: 218: 5: Favorites

219: 220: 6: Android

221: 222: 7: strings.xml

223: 224: 8: MainActivity.java

225: 226: 9: colors.xml

227: 228: 10: styles.xml

229: 230: 11: Pregunta.java

231: 232: 12: ic_adelante.png

233: 234: 13: Gradle

235: 236: 14: 1: Project

237: 238: 2: Captures

239: 240: 3: Structure

241: 242: 4: Build Variants

243: 244: 5: Favorites

245: 246: 6: Android

247: 248: 7: strings.xml

249: 250: 8: MainActivity.java

251: 252: 9: colors.xml

253: 254: 10: styles.xml

255: 256: 11: Pregunta.java

257: 258: 12: ic_adelante.png

259: 260: 13: Gradle

261: 262: 14: 1: Project

263: 264: 2: Captures

265: 266: 3: Structure

267: 268: 4: Build Variants

269: 270: 5: Favorites

271: 272: 6: Android

273: 274: 7: strings.xml

275: 276: 8: MainActivity.java

277: 278: 9: colors.xml

279: 280: 10: styles.xml

281: 282: 11: Pregunta.java

283: 284: 12: ic_adelante.png

285: 286: 13: Gradle

287: 288: 14: 1: Project

289: 290: 2: Captures

291: 292: 3: Structure

293: 294: 4: Build Variants

295: 296: 5: Favorites

297: 298: 6: Android

299: 300: 7: strings.xml

301: 302: 8: MainActivity.java

303: 304: 9: colors.xml

305: 306: 10: styles.xml

307: 308: 11: Pregunta.java

309: 310: 12: ic_adelante.png

311: 312: 13: Gradle

313: 314: 14: 1: Project

315: 316: 2: Captures

317: 318: 3: Structure

319: 320: 4: Build Variants

321: 322: 5: Favorites

323: 324: 6: Android

325: 326: 7: strings.xml

327: 328: 8: MainActivity.java

329: 330: 9: colors.xml

331: 332: 10: styles.xml

333: 334: 11: Pregunta.java

335: 336: 12: ic_adelante.png

337: 338: 13: Gradle

339: 340: 14: 1: Project

341: 342: 2: Captures

343: 344: 3: Structure

345: 346: 4: Build Variants

347: 348: 5: Favorites

349: 350: 6: Android

351: 352: 7: strings.xml

353: 354: 8: MainActivity.java

355: 356: 9: colors.xml

357: 358: 10: styles.xml

359: 360: 11: Pregunta.java

361: 362: 12: ic_adelante.png

363: 364: 13: Gradle

365: 366: 14: 1: Project

367: 368: 2: Captures

369: 370: 3: Structure

371: 372: 4: Build Variants

373: 374: 5: Favorites

375: 376: 6: Android

377: 378: 7: strings.xml

379: 380: 8: MainActivity.java

381: 382: 9: colors.xml

383: 384: 10: styles.xml

385: 386: 11: Pregunta.java

387: 388: 12: ic_adelante.png

389: 390: 13: Gradle

391: 392: 14: 1: Project

393: 394: 2: Captures

395: 396: 3: Structure

397: 398: 4: Build Variants

399: 400: 5: Favorites

401: 402: 6: Android

403: 404: 7: strings.xml

405: 406: 8: MainActivity.java

407: 408: 9: colors.xml

409: 410: 10: styles.xml

411: 412: 11: Pregunta.java

413: 414: 12: ic_adelante.png

415: 416: 13: Gradle

417: 418: 14: 1: Project

419: 420: 2: Captures

421: 422: 3: Structure

423: 424: 4: Build Variants

425: 426: 5: Favorites

427: 428: 6: Android

429: 430: 7: strings.xml

431: 432: 8: MainActivity.java

433: 434: 9: colors.xml

435: 436: 10: styles.xml

437: 438: 11: Pregunta.java

439: 440: 12: ic_adelante.png

441: 442: 13: Gradle

443: 444: 14: 1: Project

445: 446: 2: Captures

447: 448: 3: Structure

449: 450: 4: Build Variants

451: 452: 5: Favorites

453: 454: 6: Android

455: 456: 7: strings.xml

457: 458: 8: MainActivity.java

459: 460: 9: colors.xml

461: 462: 10: styles.xml

463: 464: 11: Pregunta.java

465: 466: 12: ic_adelante.png

467: 468: 13: Gradle

469: 470: 14: 1: Project

471: 472: 2: Captures

473: 474: 3: Structure

475: 476: 4: Build Variants

477: 478: 5: Favorites

479: 480: 6: Android

481: 482: 7: strings.xml

483: 484: 8: MainActivity.java

485: 486: 9: colors.xml

487: 488: 10: styles.xml

489: 490: 11: Pregunta.java

491: 492: 12: ic_adelante.png

493: 494: 13: Gradle

495: 496: 14: 1: Project

497: 498: 2: Captures

499: 500: 3: Structure

501: 502: 4: Build Variants

503: 504: 5: Favorites

505: 506: 6: Android

507: 508: 7: strings.xml

509: 510: 8: MainActivity.java

511: 512: 9: colors.xml

513: 514: 10: styles.xml

515: 516: 11: Pregunta.java

517: 518: 12: ic_adelante.png

519: 520: 13: Gradle

521: 522: 14: 1: Project

523: 524: 2: Captures

525: 526: 3: Structure

527: 528: 4: Build Variants

529: 530: 5: Favorites

531: 532: 6: Android

533: 534: 7: strings.xml

535: 536: 8: MainActivity.java

537: 538: 9: colors.xml

539: 540: 10: styles.xml

541: 542: 11: Pregunta.java

543: 544: 12: ic_adelante.png

545: 546: 13: Gradle

547: 548: 14: 1: Project

549: 550: 2: Captures

551: 552: 3: Structure

553: 554: 4: Build Variants

555: 556: 5: Favorites

557: 558: 6: Android

559: 560: 7: strings.xml

561: 562: 8: MainActivity.java

563: 564: 9: colors.xml

565: 566: 10: styles.xml

567: 568: 11: Pregunta.java

569: 570: 12: ic_adelante.png

571: 572: 13: Gradle

573: 574: 14: 1: Project

575: 576: 2: Captures

577: 578: 3: Structure

579: 580: 4: Build Variants

581: 582: 5: Favorites

583: 584: 6: Android

585: 586: 7: strings.xml

587: 588: 8: MainActivity.java

589: 590: 9: colors.xml

591: 592: 10: styles.xml

593: 594: 11: Pregunta.java

595: 596: 12: ic_adelante.png

597: 598: 13: Gradle

599: 600: 14: 1: Project

601: 602: 2: Captures

603: 604: 3: Structure

605: 606: 4: Build Variants

607: 608: 5: Favorites

609: 610: 6: Android

611: 612: 7: strings.xml

613: 614: 8: MainActivity.java

615: 616: 9: colors.xml

617: 618: 10: styles.xml

619: 620: 11: Pregunta.java

621: 622: 12: ic_adelante.png

623: 624: 13: Gradle

625: 626: 14: 1: Project

627: 628: 2: Captures

629: 630: 3: Structure

631: 632: 4: Build Variants

633: 634: 5: Favorites

635: 636: 6: Android

637: 638: 7: strings.xml

639: 640: 8: MainActivity.java

641: 642: 9: colors.xml

643: 644: 10: styles.xml

645: 646: 11: Pregunta.java

647: 648: 12: ic_adelante.png

649: 650: 13: Gradle

651: 652: 14: 1: Project

653: 654: 2: Captures

655: 656: 3: Structure

657: 658: 4: Build Variants

659: 660: 5: Favorites

661: 662: 6: Android

663: 664: 7: strings.xml

665: 666: 8: MainActivity.java

667: 668: 9: colors.xml

669: 670: 10: styles.xml

671: 672: 11: Pregunta.java

673: 674: 12: ic_adelante.png

675: 676: 13: Gradle

677: 678: 14: 1: Project

679: 680: 2: Captures

681: 682: 3: Structure

683: 684: 4: Build Variants

685: 686: 5: Favorites

687: 688: 6: Android

689: 690: 7: strings.xml

691: 692: 8: MainActivity.java

693: 694: 9: colors.xml

695: 696: 10: styles.xml

697: 698: 11: Pregunta.java

699: 700: 12: ic_adelante.png

701: 702: 13: Gradle

703: 704: 14: 1: Project

705: 706: 2: Captures

707: 708: 3: Structure

709: 710: 4: Build Variants

711: 712: 5: Favorites

713: 714: 6: Android

715: 716: 7: strings.xml

717: 718: 8: MainActivity.java

719: 720: 9: colors.xml

721: 722: 10: styles.xml

723: 724: 11: Pregunta.java

725: 726: 12: ic_adelante.png

727: 728: 13: Gradle

729: 730: 14: 1: Project

731: 732: 2: Captures

733: 734: 3: Structure

735: 736: 4: Build Variants

737: 738: 5: Favorites

739: 740: 6: Android

741: 742: 7: strings.xml

743: 744: 8: MainActivity.java

745: 746: 9: colors.xml

747: 748: 10: styles.xml

749: 750: 11: Pregunta.java

751: 752: 12: ic_adelante.png

753: 754: 13: Gradle

755: 756: 14: 1: Project

757: 758: 2: Captures

759: 760: 3: Structure

761: 762: 4: Build Variants

763: 764: 5: Favorites

765: 766: 6: Android

767: 768: 7: strings.xml

769: 770: 8: MainActivity.java

771: 772: 9: colors.xml

773: 774: 10: styles.xml

775: 776: 11: Pregunta.java

777: 778: 12: ic_adelante.png

779: 780: 13: Gradle

781: 782: 14: 1: Project

783: 784: 2: Captures

785: 786: 3: Structure

787: 788: 4: Build Variants

789: 790: 5: Favorites

791: 792: 6: Android

793: 794: 7: strings.xml

795: 796: 8: MainActivity.java

797: 798: 9: colors.xml

799: 800: 10: styles.xml

801: 802: 11: Pregunta.java

803: 804: 12: ic_adelante.png

805: 806: 13: Gradle

807: 808: 14: 1: Project

809: 810: 2: Captures

811: 812: 3: Structure

813: 814: 4: Build Variants

815: 816: 5: Favorites

817: 818: 6: Android

819: 820: 7: strings.xml

821: 822: 8: MainActivity.java

823: 824: 9: colors.xml

825: 826: 10: styles.xml

827: 828: 11: Pregunta.java

829: 830: 12: ic_adelante.png

831: 832: 13: Gradle

833: 834: 14: 1: Project

835: 836: 2: Captures

837: 838: 3: Structure

839: 840: 4: Build Variants

841: 842: 5: Favorites

843: 844: 6: Android

845: 846: 7: strings.xml

847: 848: 8: MainActivity.java

849: 850: 9: colors.xml

851: 852: 10: styles.xml

853: 854: 11: Pregunta.java

855: 856: 12: ic_adelante.png

857: 858: 13: Gradle

859: 860: 14: 1: Project

861: 862: 2: Captures

863: 864: 3: Structure

865: 866: 4: Build Variants

867: 868: 5: Favorites

869: 870: 6: Android

871: 872: 7: strings.xml

873: 874: 8: MainActivity.java

875: 876: 9: colors.xml

877: 878: 10: styles.xml

879: 880: 11: Pregunta.java

881: 882: 12: ic_adelante.png

883: 884: 13: Gradle

885: 886: 14: 1: Project

887: 888: 2: Captures

889: 890: 3: Structure

891: 892: 4: Build Variants

893: 894: 5: Favorites

895: 896: 6: Android

897: 898: 7: strings.xml

899: 900: 8: MainActivity.java

901: 902: 9: colors.xml

903: 904: 10: styles.xml

905: 906: 11: Pregunta.java

907: 908: 12: ic_adelante.png

909: 910: 13: Gradle

911: 912: 14: 1: Project

913: 914: 2: Captures

915: 916: 3: Structure

917: 918: 4: Build Variants

919: 920: 5: Favorites

921: 922: 6: Android

923: 924: 7: strings.xml

925: 926: 8: MainActivity.java

927: 928: 9: colors.xml

929: 930: 10: styles.xml

931: 932: 11: Pregunta.java

933: 934: 12: ic_adelante.png

935: 936: 13: Gradle

937: 938: 14: 1: Project

939: 940: 2: Captures

941: 942: 3: Structure

943: 944: 4: Build Variants

945: 946: 5: Favorites

947: 948: 6: Android

949: 950: 7: strings.xml

951: 952: 8: MainActivity.java

953: 954: 9: colors.xml

955: 956: 10: styles.xml

957: 958: 11: Pregunta.java

959: 960: 12: ic_adelante.png

961: 962: 13: Gradle

963: 964: 14: 1: Project

965: 966: 2: Captures

967: 968: 3: Structure

969: 970: 4: Build Variants

971: 972: 5: Favorites

973: 974: 6: Android

975: 976: 7: strings.xml

977: 978: 8: MainActivity.java

979: 980: 9: colors.xml

981: 982: 10: styles.xml

983: 984: 11: Pregunta.java

985: 986: 12: ic_adelante.png

987: 988: 13: Gradle

989: 990: 14: 1: Project

991: 992: 2: Captures

993: 994: 3: Structure

995: 996: 4: Build Variants

997: 998: 5: Favorites

999: 1000: 6: Android

La vista de la izquierda es la project tool window

La vista derecha tenemos el editor de código

Cosas a tener en cuenta



A screenshot of an Android Studio code editor showing the XML layout file `activity_main.xml`. The code defines a vertical linear layout containing a text view and two buttons. A red arrow points from the bottom of the slide towards the 'Text' tab in the bottom navigation bar.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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"
    android:gravity="center"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/text_pregunta"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="24dp" />

    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" >

        <Button
            android:id="@+id/button_verdadero"
            android:text="Verdadero"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />

        <Button
            android:id="@+id/button_falso"
            android:text="Falso"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
    

```

The bottom navigation bar shows tabs for 'Design' and 'Text'. A red arrow points to the 'Text' tab.

GeoQuiz [C:\Users\user\AndroidStudioProjects\GeoQuiz] - ...app\src\main\res\layout\activity_main.xml [app] - Android Studio

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

GeoQuiz app src main res layout activity_main.xml

1: Project Captures 2: Structure 3: Build Variants 4: Favorites

MainActivity.java layout\activity_main.xml land\activity_main.xml Pregunta.java strings.xml activity_ver_respuesta.xml

Preview Gradle Device File Explorer

1 app manifests java generatedJava res drawable layout activity_main (2) activity_ver_respuesta.xml mipmap values

Gradle Scripts build.gradle (Project: GeoQuiz) build.gradle (Module: app) gradle-wrapper.properties (Gradle Version) proguard-rules.pro (ProGuard Rules for app) gradle.properties (Project Properties) settings.gradle (Project Settings) local.properties (SDK Location)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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"

    android:gravity="center"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/text_pregunta"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="24dp" />

    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" >

        <Button
            android:id="@+id/button_verdadero"
            android:text="Verdadero"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />

        <Button
            android:id="@+id/button_falso"
            android:text="Falso"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
    

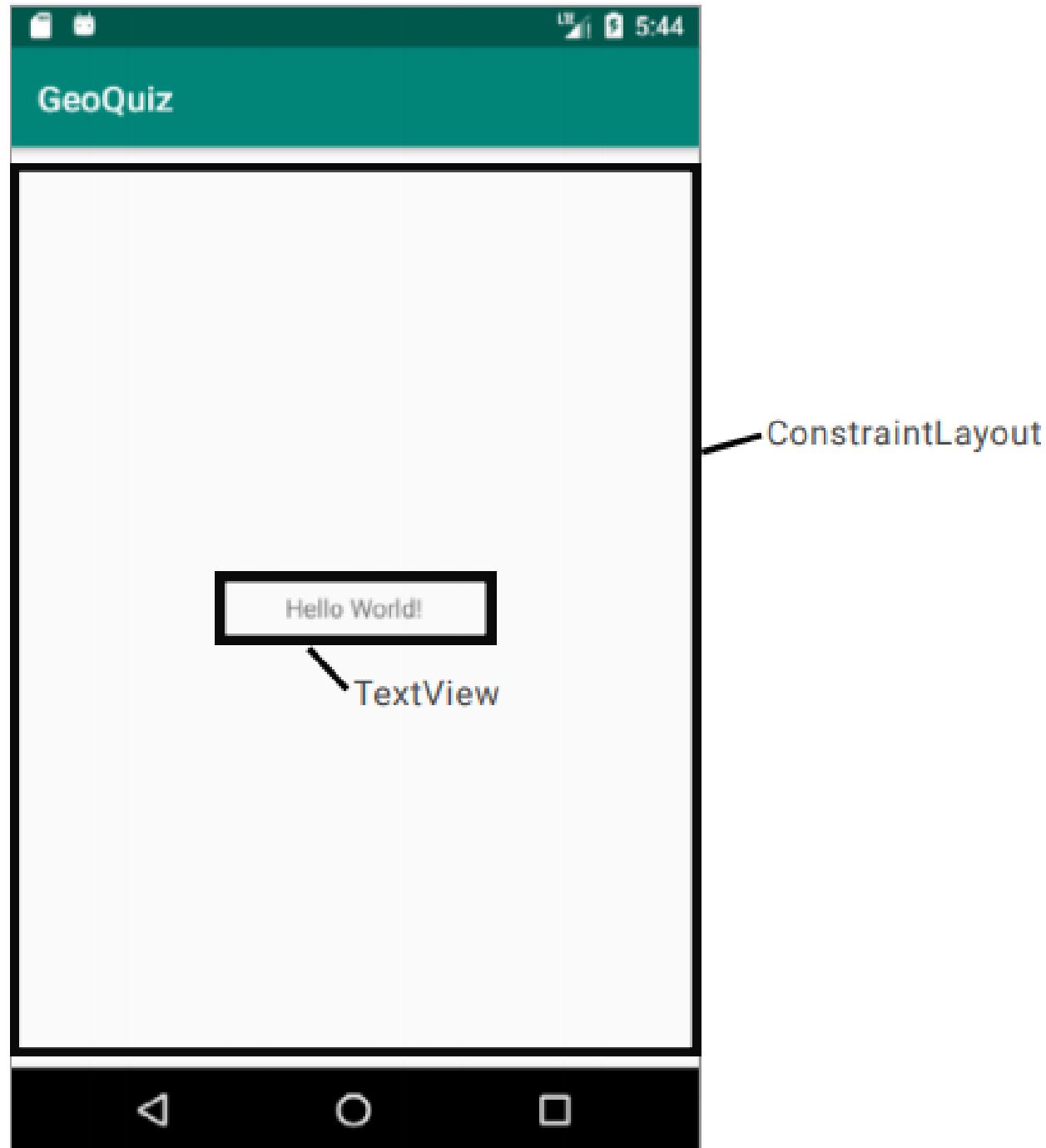
```

Design Text

JShell: JDK version is 8. JDK 9 or higher is needed to run JShell. (8 minutes ago)

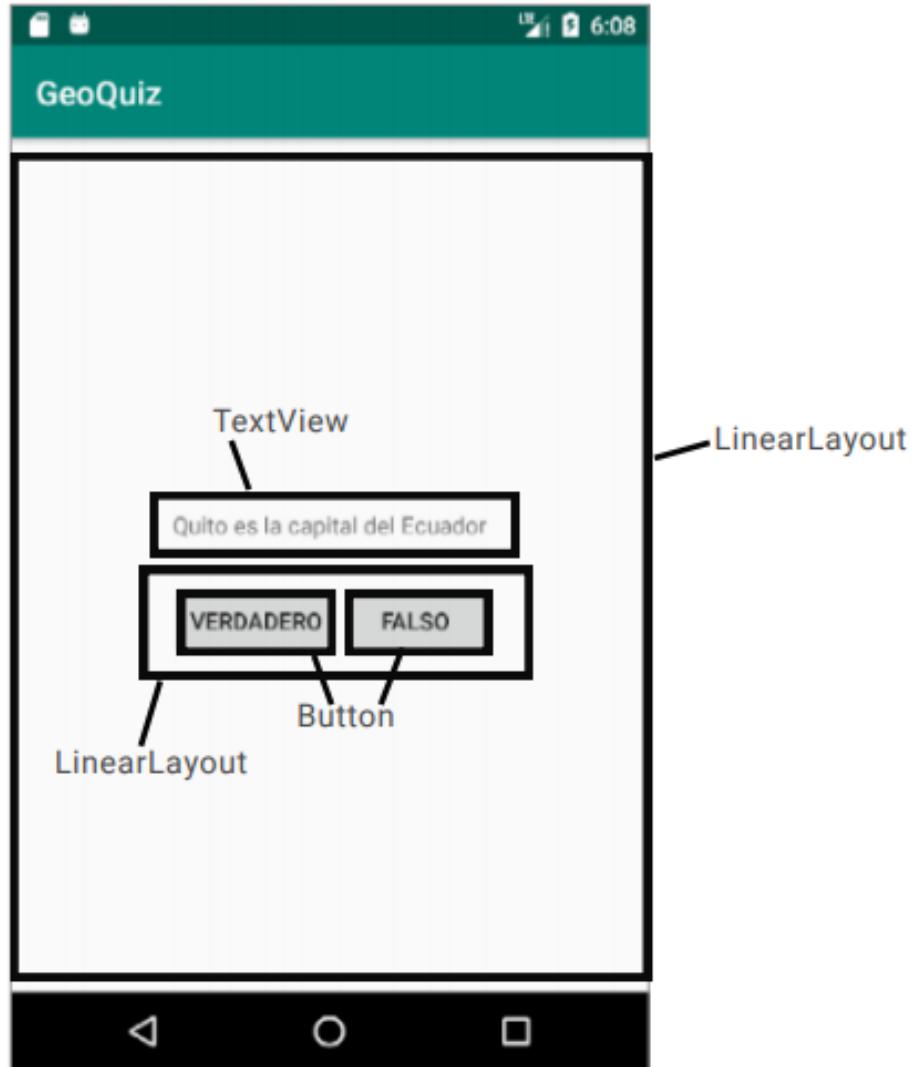
61:28 CRLF: UTF-8 Context: <no context>

Cosas a tener en cuenta





Cosas a tener en cuenta





Jerarquía de Componentes

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center">

    <TextView
        android:text="¿Es Quito la capital de Ecuador?"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="24dp"/>

    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Verdadero"/>

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Falso"/>
    </LinearLayout>
</LinearLayout>
```

1. **android:layout_width y android:layout_height**
 - **match_parent:** la view será tan grande como su parent
 - **wrap_content:** la view será tan grande como su contenido
2. **android:padding:** espacio interior del elemento
3. **LinearLayout:**
 - **android:orientation:** determina como los widgets serán mostrados en pantalla (vertical u horizontal)

Atributos de los widges

1. **android:layout_width** y **android:layout_height**
 - **match_parent**: la view será tan grande como su parent
 - **wrap_content**: la view será tan grande como su contenido
2. **android:padding**: espacio interior del elemento
3. **LinearLayout**:
 - **android:orientation**: determina como los widgets serán mostrados en pantalla (vertical u horizontal)

Creando recursos string

```
<resources>
    <string name="app_name">GeoQuiz</string>
    <string name="pregunta_text">Quito es la capital del Ecuador</string>
    <string name="verdadero_button">Verdadero</string>
    <string name="falso_button">Falso</string>
</resources>
```

- Todos los proyectos incluyen el archivo **res/values/ string.xml**
- Para obtener el valor de un recurso string desde cualquier xml, se lo referencia como **@string/nombre_recurso**
- En la proyecto creamos 3 recursos string:

Creando recursos string

- Todos los proyectos incluyen el archivo
res/values/ string.xml
- Para obtener el valor de un recurso string desde cualquier xml, se lo referencia como
@string/nombre_recurso
- En la proyecto creamos 3 recursos string:

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



Código



Código

```
import android.support.v7.app.AppCompatActivity  
import android.os.Bundle  
  
class MainActivity : AppCompatActivity() {  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
    }  
}
```

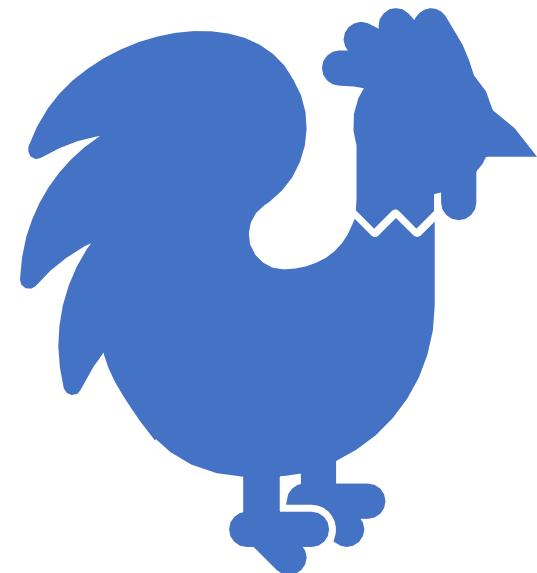
AppCompatActivity es una subclase de la clase Activity.

El método **onCreate(Bundle)** es llamado cuando se instancia una subclase Activity.

setContentView(int layoutResID) infla el layout. Cuando un layout se infla, cada widget es instanciado

Código

- **AppCompatActivity** es una subclase de la clase **Activity**.
- El método **onCreate(Bundle)** es llamado cuando se instancia una subclase Activity.
- **setContentView(int layoutResID)** infla el layout. Cuando un layout se infla, cada widget es instanciado





Conectando widgets con el código

```
<LinearLayout ...>  
    <TextView .../>  
    <LinearLayout ...>  
        <Button  
            android:id="@+id/button_verdadero"  
            android:layout_width="wrap_content"  
            android:layout_height="wrap_content"  
            android:text="@string/verdadero_button"/>  
        <Button  
            android:id="@+id/button_falso"  
            android:layout_width="wrap_content"  
            android:layout_height="wrap_content"  
            android:text="@string/falso_button"/>  
    </LinearLayout>  
</LinearLayout>
```

Crear componentes en .xml



Conectando widgets con el código

```
package com.curso.miprimeraaplicacion  
  
import android.support.v7.app.AppCompatActivity  
import android.os.Bundle  
import android.widget.Button  
  
class MainActivity : AppCompatActivity() {  
  
    var btnVerdadero: Button? = null  
    var btnFalso: Button? = null  
    ...  
}
```

Definir variables en java



Conectando widgets con el código

```
package com.curso.miprimeraaplicacion

import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import android.widget.Button

class MainActivity : AppCompatActivity() {

    var btnVerdadero: Button? = null
    var btnFalso: Button? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        btnVerdadero = findViewById(R.id.btn_verdadero) as Button
        btnFalso = findViewById(R.id.btn_falso)
    }
}
```

Relación entre el diseño y el código

Inflar Variables



Conectando widgets con el código

Cuando se le asigna un Id utilizando **android:id="+id/control"**.

android:id="@+id/button_verdadero"

se puede acceder a este elemento utilizando la función **findViewById**.

btnVerdadero = findViewById(R.id.btn_verdadero)

```
<LinearLayout ...>
    <TextView .../>
    <LinearLayout ...>
        <Button
            android:id="@+id/button_verdadero"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/verdadero_button"/>
        <Button
            android:id="@+id/button_falso"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/falso_button"/>
    </LinearLayout>
</LinearLayout>
```

```
package com.curso.miprimeraaplicacion
import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import android.widget.Button
class MainActivity : AppCompatActivity() {
    var btnVerdadero: Button? = null
    var btnFalso: Button? = null
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        btnVerdadero = findViewById(R.id.btn_verdadero) as Button
        btnFalso = findViewById(R.id.btn_falso)
    }
}
```

Eventos en Android

```
class MainActivity : AppCompatActivity(), View.OnClickListener  
{  
    override fun onClick(v: View?) {  
    }  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        btnVerdadero?.setOnClickListener {  
        }  
        btnVerdadero?.setOnClickListener(View.OnClickListener()  
        )  
        btnVerdadero?.setOnClickListener(this)  
    }  
}
```

Identificar y describir
cada uno de los
métodos eventos

Método agrupa un conjunto de instrucciones
por ejemplo una función

Evento es una actividad por la que pasa un
componente. Ejemplo evento clicked,
eliminar, etc

Eventos en Android

```
package com.curso.miprimeraaplicacion

import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import android.widget.Button

class MainActivity : AppCompatActivity() {

    var btnVerdadero: Button? = null
    var btnFalso: Button? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        btnVerdadero = findViewById(R.id.btn_verdadero) as Button
        btnFalso = findViewById(R.id.btn_falso)

        btnVerdadero?.setOnClickListener {
            }

        btnFalso?.setOnClickListener{
            }
    }
}
```

Identificar y
describir cada uno
de los métodos
eventos

Creando Android Virtual Device en Android Studio

Sobre AVD

- AVDs son esencialmente emuladores que permiten probar las aplicaciones Android con la necesidad de utilizar un dispositivo físico.
- Un AVD puede ser configurado para emular una variedad de hardware de diferentes características tales como tamaño de la pantalla, capacidad de memoria y otras características como la cámara.



Your Virtual Devices

Android Studio

Type	Name	Play Store	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	Nexus 5X API 22		1080 × 1920: 420dpi	22	Android 5.1 (Google APIs)	x86	3.5 GB	
	Nexus 5X API 27		1080 × 1920: 420dpi	27	Android 8.1 (Google Play)	x86	9.3 GB	
	Nexus 5X API 27 2		1080 × 1920: 420dpi	27	Android 8.1 (Google Play)	x86	8.8 GB	
	Nexus 5X API 28		1080 × 1920: 420dpi	28	Android 9.0 (Google Play)	x86	8.8 GB	
	Nexus 5X API 28 2		1080 × 1920: 420dpi	28	Android 9.0 (Google Play)	x86	513 MB	
	Nexus 5X API 28 3		1080 × 1920: 420dpi	28	Android 9.0 (Google Play)	x86	513 MB	

Creando un nuevo AVD

- Para crear un nuevo dispositivo virtual vamos al menú de la ventana principal **Tools -> Android -> AVD Manager**.

Ejecutando

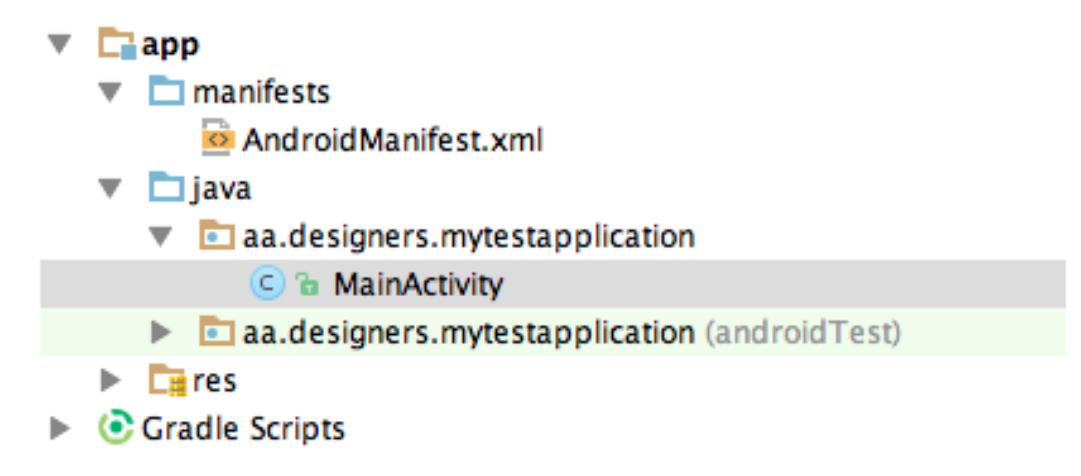




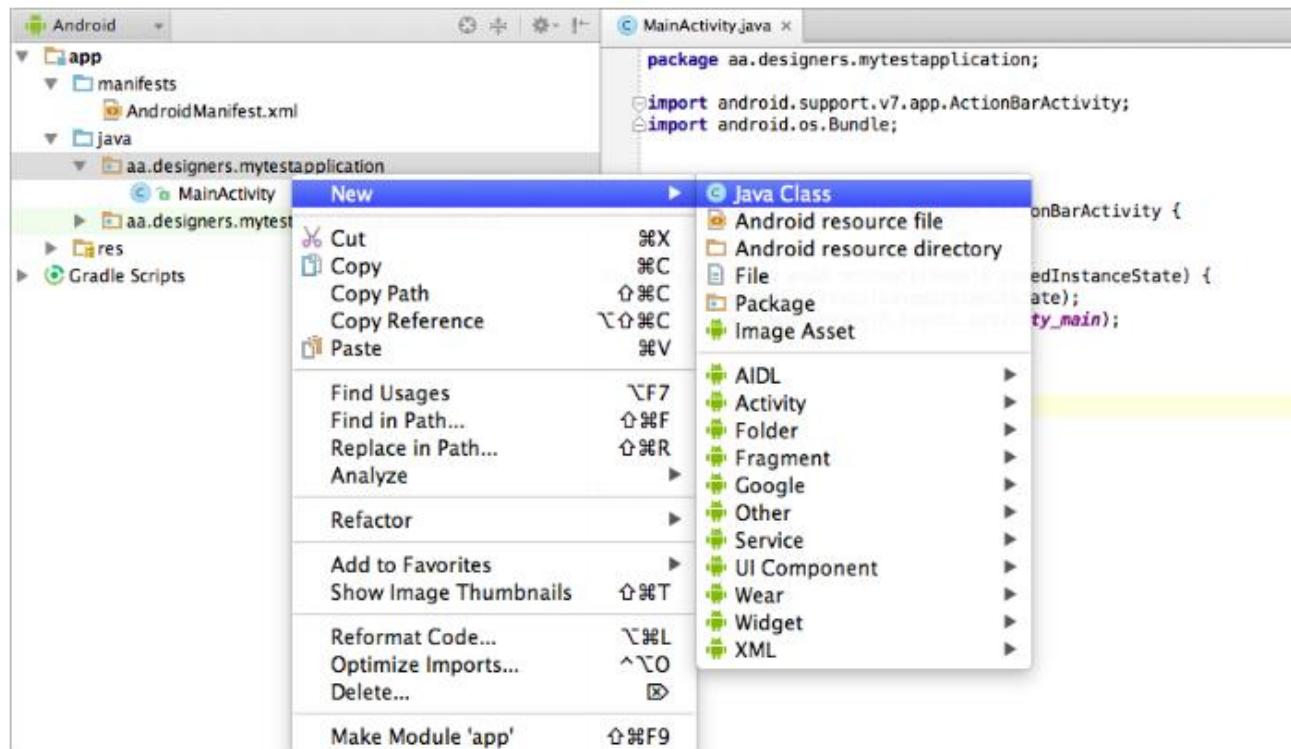
Activity, ciclo de vida y flujo de navegación

Activity

- Un activity es cada una de las pantallas o vistas que forman una aplicación.

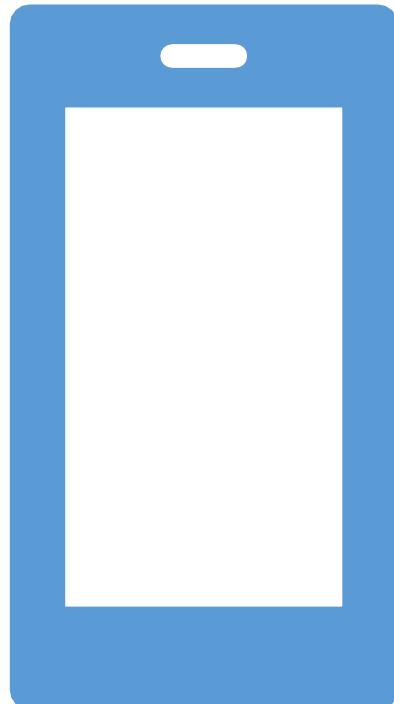


Activity



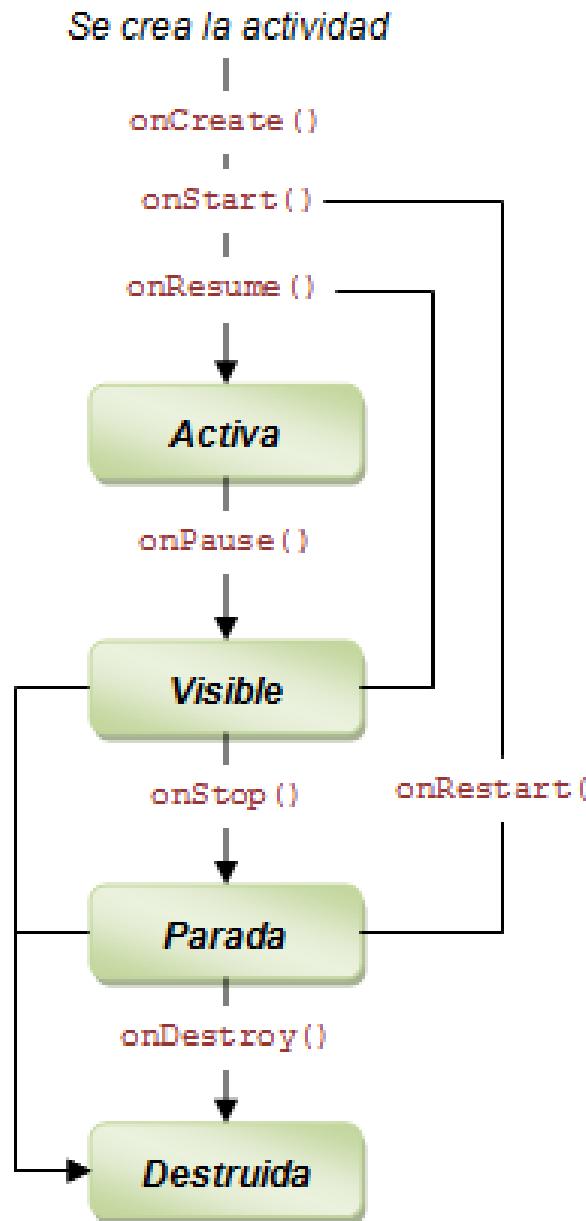
- Se puede **añadir un nuevo *activity*** haciendo click derecho sobre el paquete que deseemos, New > Java Class, como vemos en la figura adjunta.
- Esto **creará un nuevo archivo.java** desde el cual podemos manejar un nuevo *activity*.

ESTADOS DE UNA ACTIVIDAD



Una actividad en Android puede estar en uno de estos cuatro estados:

- **Activa (Running)**: La actividad está encima de la pila, lo que quiere decir que **es visible y tiene el foco**.
 - **Visible (Paused)**: La actividad **es visible pero no tiene el foco**. Se alcanza este estado cuando **pasa a activa otra actividad con alguna parte transparente o que no ocupa toda la pantalla**. Cuando una actividad está tapada por completo, pasa a estar parada.
 - **Parada (Stopped)**: Cuando la actividad **no es visible**. El programador debe guardar el estado de la interfaz de usuario, preferencias, etc.
 - **Destruida (Destroyed)**: Cuando la **actividad termina al invocarse el método `finish()`**, o es matada por el sistema.
- Cada vez que una actividad cambia de estado se van a generar eventos que podrán ser capturados por ciertos métodos de la actividad. A continuación se muestra un esquema que ilustra los métodos que capturan estos eventos.



Métodos de un activity

- **onCreate(Bundle)**: Se llama en la creación de la actividad. Se utiliza para realizar todo tipo de inicializaciones, como la creación de la interfaz de usuario o la inicialización de estructuras de datos.
- **onStart()**: Nos indica que la actividad está a punto de ser mostrada al usuario.
- **onResume()**: Se llama cuando la actividad va a comenzar a interactuar con el usuario. Es un buen lugar para lanzar las animaciones y la música.
- **onPause()**: Indica que la actividad está a punto de ser lanzada a segundo plano, normalmente porque otra actividad es lanzada. Es el lugar adecuado para detener animaciones, música o almacenar los datos que estaban en edición.
- **onStop()**: La actividad ya no va a ser visible para el usuario. ¡Ojo si hay muy poca memoria! es posible que la actividad se destruya sin llamar a este método.
- **onStart()**: Indica que la actividad va a volver a ser representada después de haber pasado por `onStop()`.
- **onDestroy()**: Se llama antes de que la actividad sea totalmente destruida. Por ejemplo, cuando el usuario pulsa el botón de volver o cuando se llama al método `finish()`. ¡Ojo si hay muy poca memoria, es posible que la actividad se destruya sin llamar a este método.

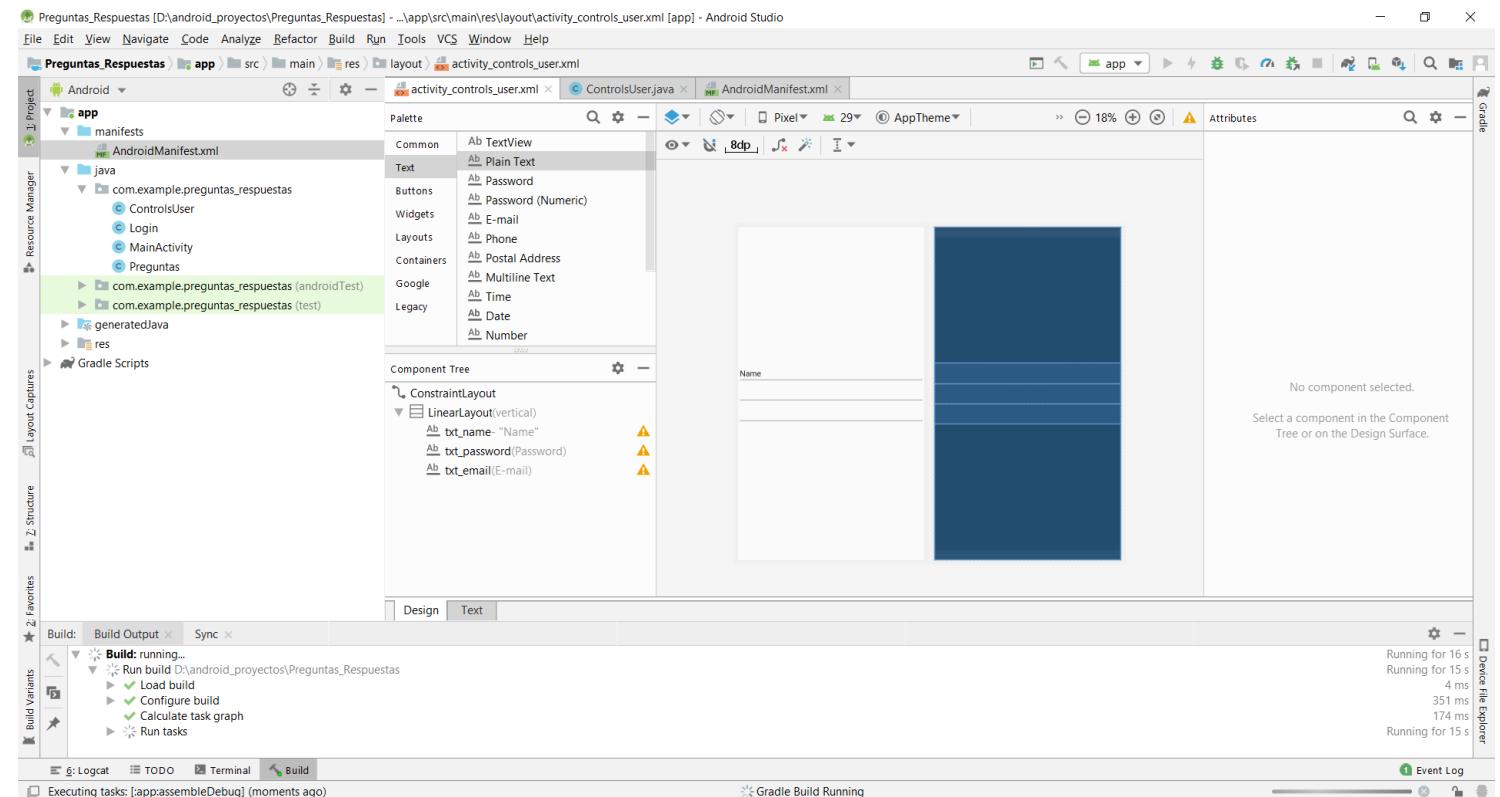
Controles de usuario: Input Controls, Alertas, y Pickers

Controles de entrada

- ▶ Son del tipo `inputType`
 - ▶ `EditText`
 - ▶ `Spinner`

Controles de entrada - Text

- ▶ Android ofrece algunas entradas para el manejo de texto tales como:
 - ▶ Texto normal
 - ▶ Contraseñas
 - ▶ Correos electrónicos
 - ▶ Teléfonos



Project

Resource Manager

Layout Captures

Structure

Favorites

Build Variants

Build: Build Output Sync

- Build: completed successfully at 28/7/2020 13:23
- Run build D:\android_proyectos\Preguntas_Respuestas
- Load build
- Configure build
- Calculate task graph
- Run tasks

4: Run 6: Logcat TODO Terminal Build Profiler

Gradle

activity_controls_user.xml ControlsUser.java AndroidManifest.xml

Palette

Common

- TextView
- Plain Text
- Password
- Password (Numeric)
- E-mail
- Phone
- Postal Address
- Multiline Text
- Time
- Date
- Number

Text

Buttons

Widgets

Layouts

Containers

Google

Legacy

Component Tree

```

ConstraintLayout
  └ LinearLayout(vertical)
    └ editText5 (Phone)
  
```

Design Text

Name

Attributes

Attribute	Value
match_parent	▼
wrap_content	▼
10	
editText5	
phone	

Properties

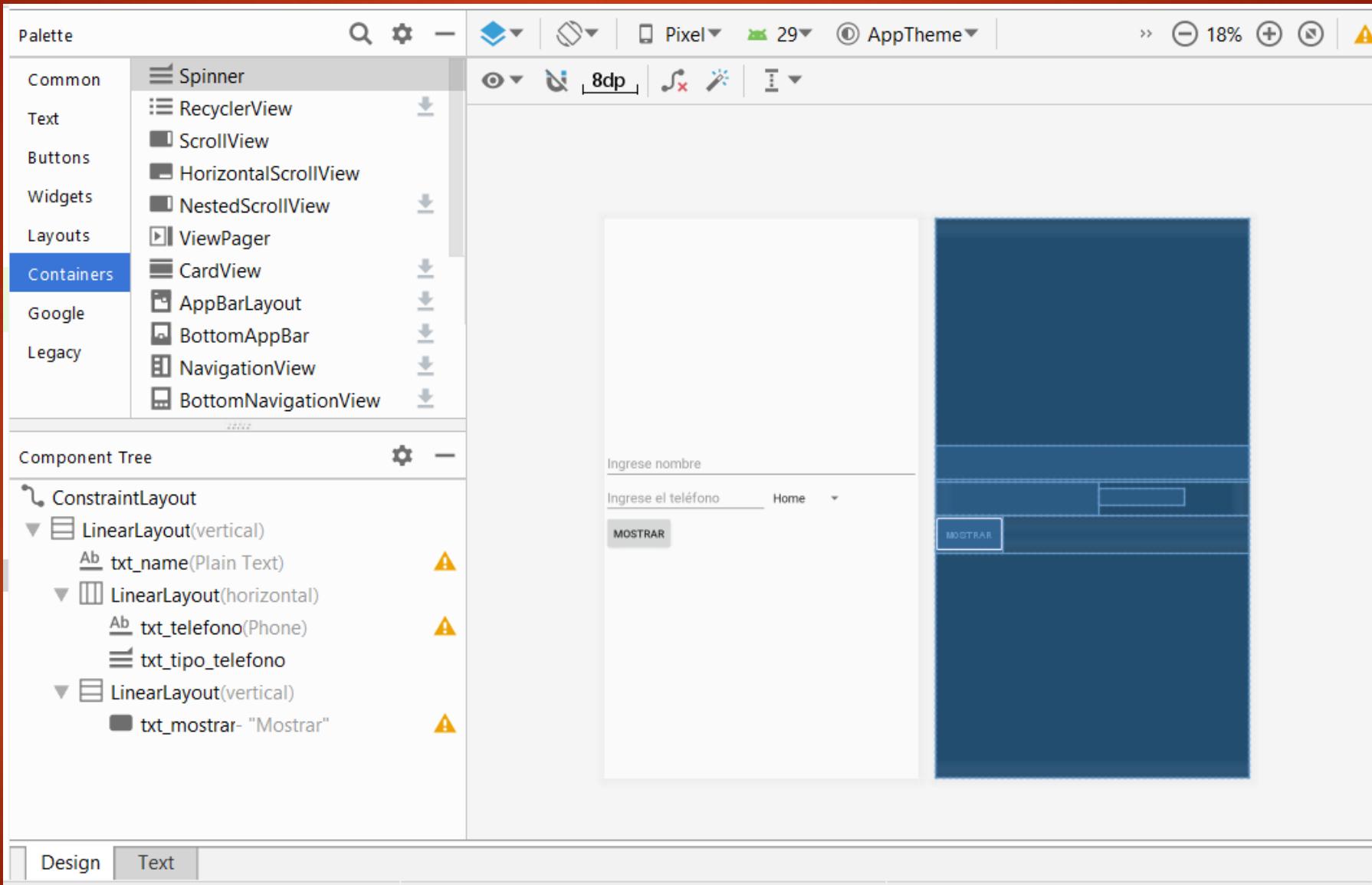
Property	Value
number	<input checked="" type="checkbox"/>
text	<input checked="" type="checkbox"/>
textPhonetic	<input type="checkbox"/>
textCapSentences	<input type="checkbox"/>
textPassword	<input type="checkbox"/>
textAutoComplete	<input type="checkbox"/>
textImeMultiLine	<input type="checkbox"/>
textPostalAddress	<input type="checkbox"/>
numberDecimal	<input type="checkbox"/>
textEmailAddress	<input type="checkbox"/>
numberPassword	<input type="checkbox"/>
textCapWords	<input type="checkbox"/>
textEmailSubject	<input type="checkbox"/>
textCapCharacters	<input type="checkbox"/>
time	<input type="checkbox"/>
textWebPassword	<input type="checkbox"/>

Device File Explorer

Event Log

Gradle build finished in 7 s 415 ms (2 minutes ago)

Spinner y Alertas



Spinner

En el archivo strings.xml

```
<string-array name="labels_array">
    <item>Home</item>
    <item>Work</item>
    <item>Mobile</item>
    <item>Other</item>
</string-array>
```

```
Spinner spinner = (Spinner) findViewById(R.id.spn_tipo_telefono);

// Create ArrayAdapter using the string array and default spinner layout.
ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource( context: this, R.array.labels_array,
    android.R.layout.simple_spinner_item);

// Especifique el diseño que se utilizará cuando aparezca la lista de opciones.
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);

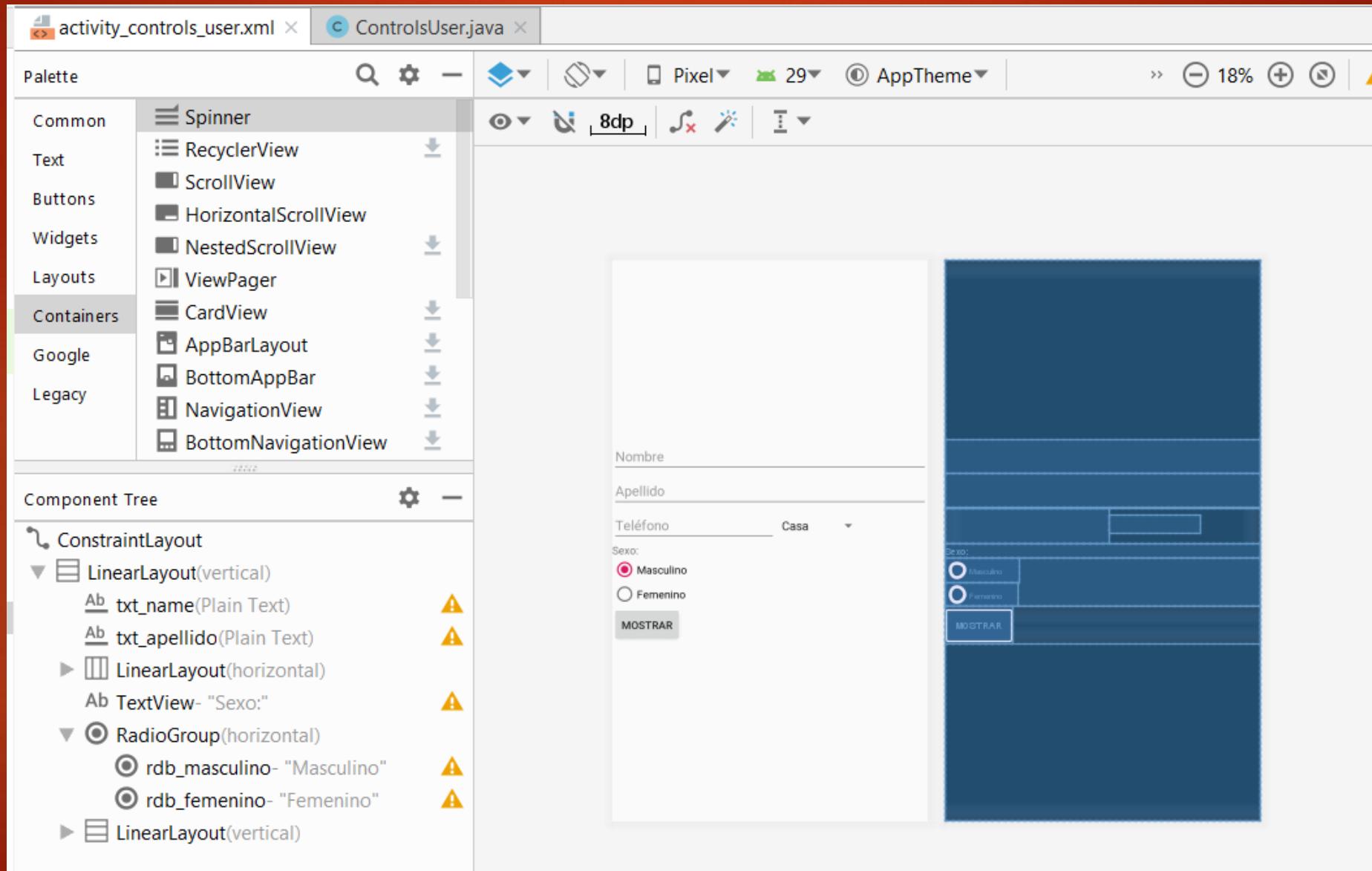
//Aplicar el adaptador al spinner
spinner.setAdapter(adapter);

spinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
        mSpinnerLabel = parent.getItemAtPosition(position).toString();
    }

    @Override
    public void onNothingSelected(AdapterView<?> parent) {

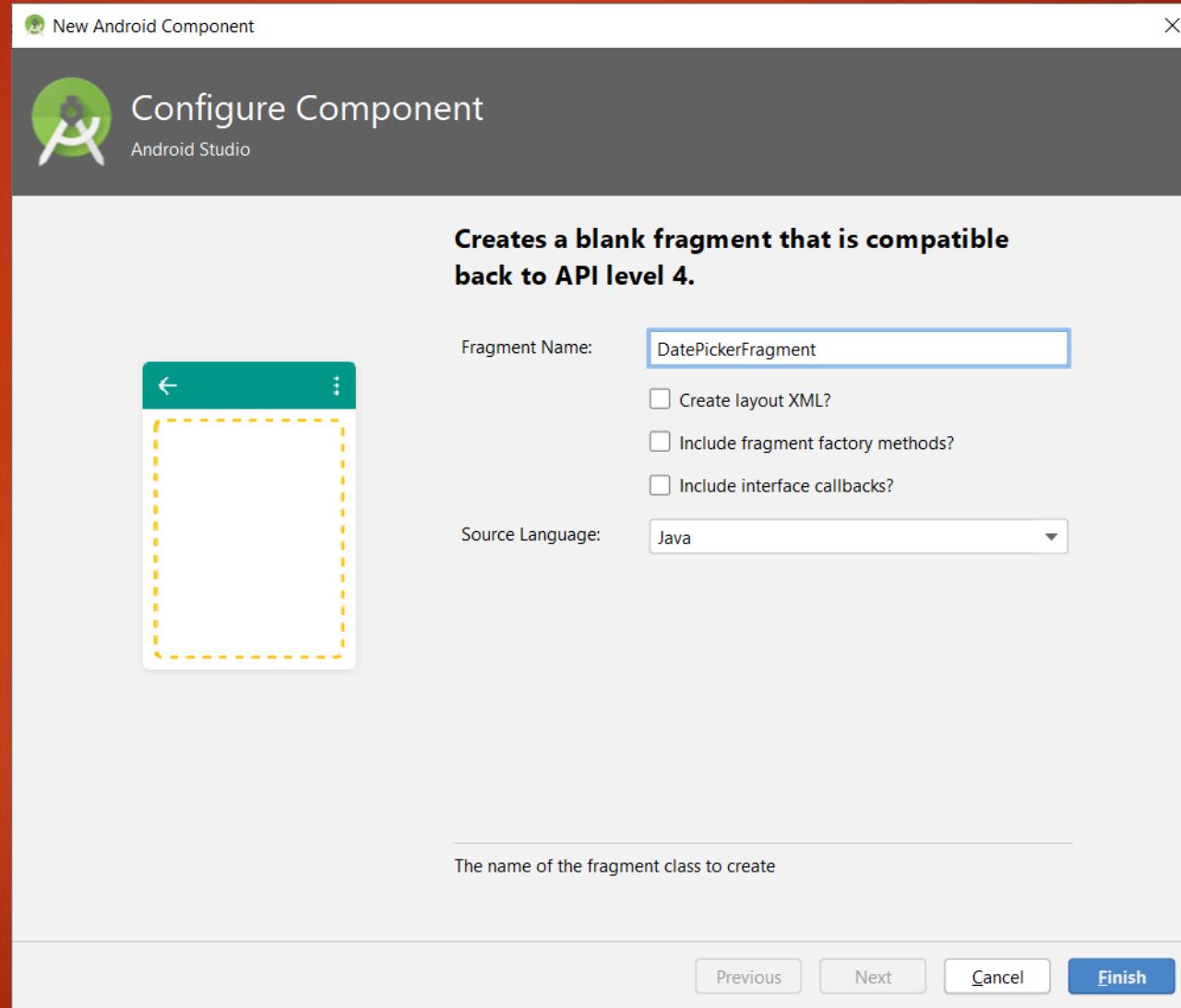
    }
});
});
```

Radio Button



```
public void onRadioButtonClicked(View view) {
    boolean checked = ((RadioButton) view).isChecked();
    // Check which radio button was clicked
    switch (view.getId()) {
        case R.id.rdb_masculino:
            if (checked)
                // Masculino
                Toast.makeText(getApplicationContext(), text: "Seleccionó Masculino", Toast.LENGTH_SHORT).show();
            break;
        case R.id.rdb_femenino:
            if (checked)
                // Femenino
                Toast.makeText(getApplicationContext(), text: "Seleccionó Femenino", Toast.LENGTH_SHORT).show();
            break;
    }
}
```

DatePickerFragment



Consideraciones:

- ▶ Implementar los métodos haciendo uso de los @override
 - ▶ onDateSet()
 - ▶ `public void onDateSet(DatePicker view, int year, int month, int day)`
- ▶ Quitar la contructor vacio DatePickerFragment.
- ▶ Reemplazar **onCreateView()** con **onCreateDialog()** que retorna **Dialog**, and anota **onCreateDialog()** with **@NonNull** para indicar que no puede retornar nulo.

Código

```
public class DatePickerFragment extends DialogFragment implements DatePickerDialog.OnDateSetListener {

    private DatePickerDialog.OnDateSetListener listener;

    public static DatePickerFragment newInstance(DatePickerDialog.OnDateSetListener listener) {
        DatePickerFragment fragment = new DatePickerFragment();
        fragment.setListener(listener);
        return fragment;
    }

    public void setListener(DatePickerDialog.OnDateSetListener listener) {
        this.listener = listener;
    }

    @NotNull
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        // Use the current date as the default date in the picker.
        final Calendar c = Calendar.getInstance();
        int year = c.get(Calendar.YEAR);
        int month = c.get(Calendar.MONTH);
        int day = c.get(Calendar.DAY_OF_MONTH);
        // Create a new instance of DatePickerDialog and return it.
        return new DatePickerDialog(getActivity(), listener, year, month, day);
    }

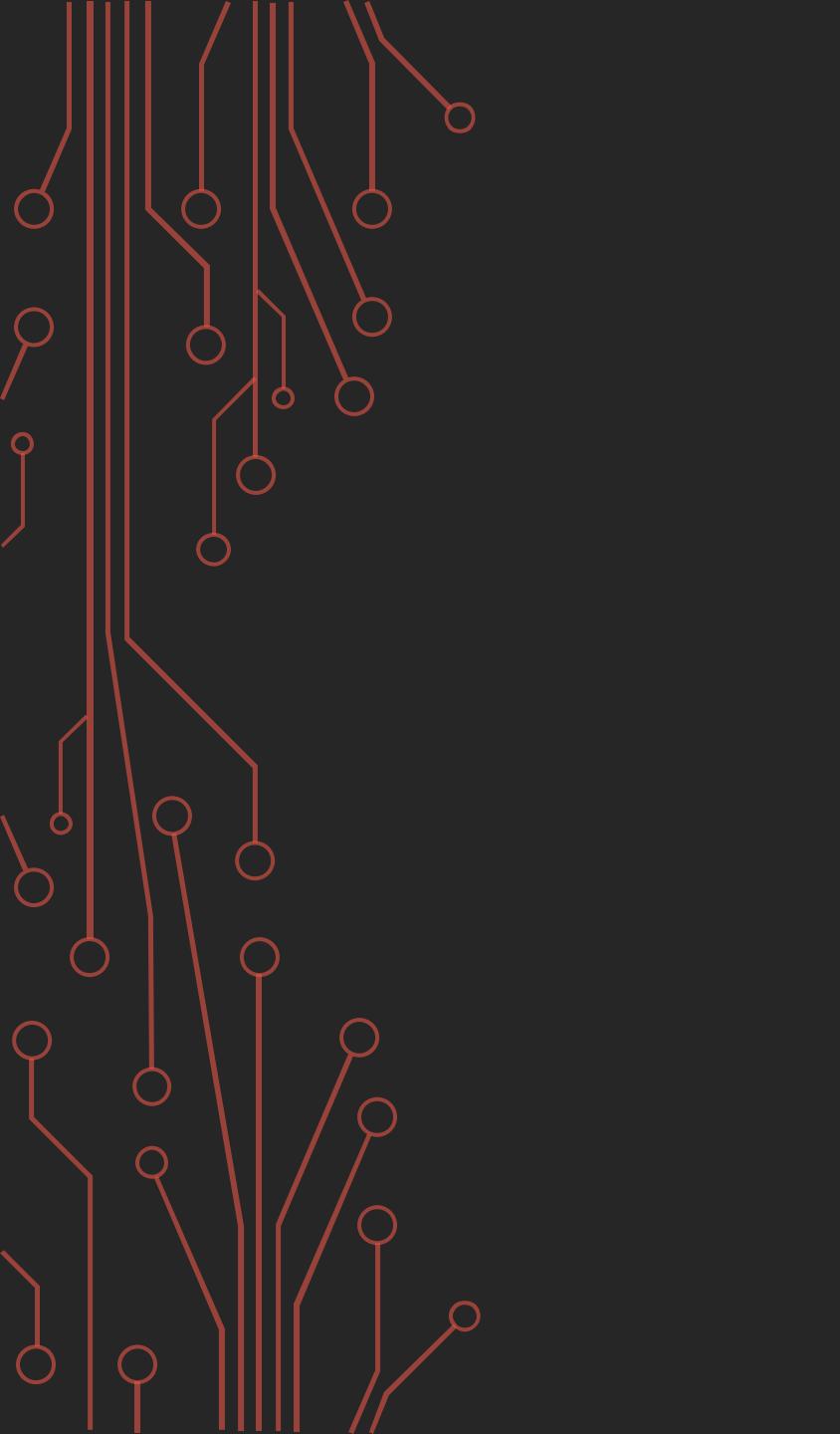
    @Override
    public void onDateSet(DatePicker view, int year, int month, int day) {
    }
}
```

En el activity principal

```
public void showDatePickerDialog(View v) {
    final TextView etPlannedDate = (TextView) findViewById(R.id.txt_fecha);

    DatePickerFragment newFragment = DatePickerFragment.newInstance(new DatePickerDialog.OnDateSetListener() {
        @Override
        public void onDateSet(DatePicker datePicker, int year, int month, int day) {
            // +1 because January is zero
            final String selectedDate = day + "/" + (month+1) + "/" + year;
            etPlannedDate.setText(selectedDate);
        }
    });

    newFragment.show(getSupportFragmentManager(), tag: "Calendario");
}
```



MENU - ANDROID

INTRODUCCIÓN

Los menús son parte importante en nuestras aplicaciones

A partir de Android 3.0 (nivel de API 11), los dispositivos con Android ya no tienen que proporcionar un botón Menú exclusivo. Con este cambio, las apps de Android dejarán de depender de los paneles de menú tradicionales de 6 elementos

TIPOS DE MENÚS

- Menú de opciones y barra de app
- Menú contextual y modo de acción contextual
- Menú emergente

MENÚ DE OPCIONES Y BARRA DE APP

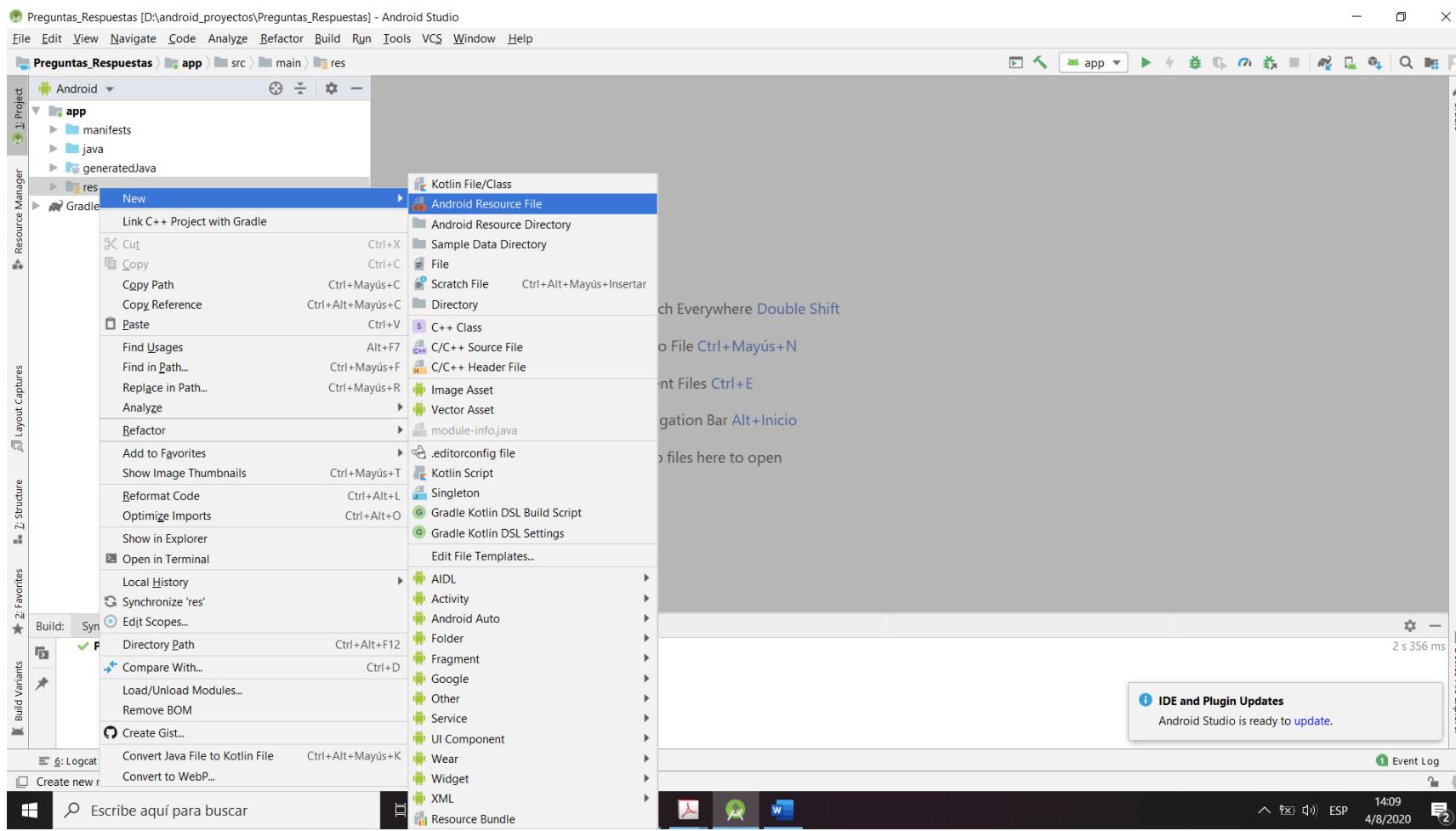
- Estos tipos de menú se encuentran diseñados para ser implementados en un Activity, dentro se colocan acciones concretas que el usuario en algún momento pueda solicitar, como Ajustes, About, Buscar etc...

MENÚ CONTEXTUAL Y MODO DE ACCIÓN CONTEXTUAL

- Este tipo de menú es un menú flotante como editar, compartir, eliminar que solo aparece cuando el usuario realizo un clic prolongado en algún elemento.

MENÚ EMERGENTE

- Un menú emergente muestra una lista de elementos en una lista vertical que está anclada a la vista que invocó el menú. Es adecuado para proporcionar una ampliación de acciones relacionadas con contenido específico o para proporcionar opciones en una segunda parte de un comando.



New Resource File

File name:

Resource type:

Root element:

Source set:

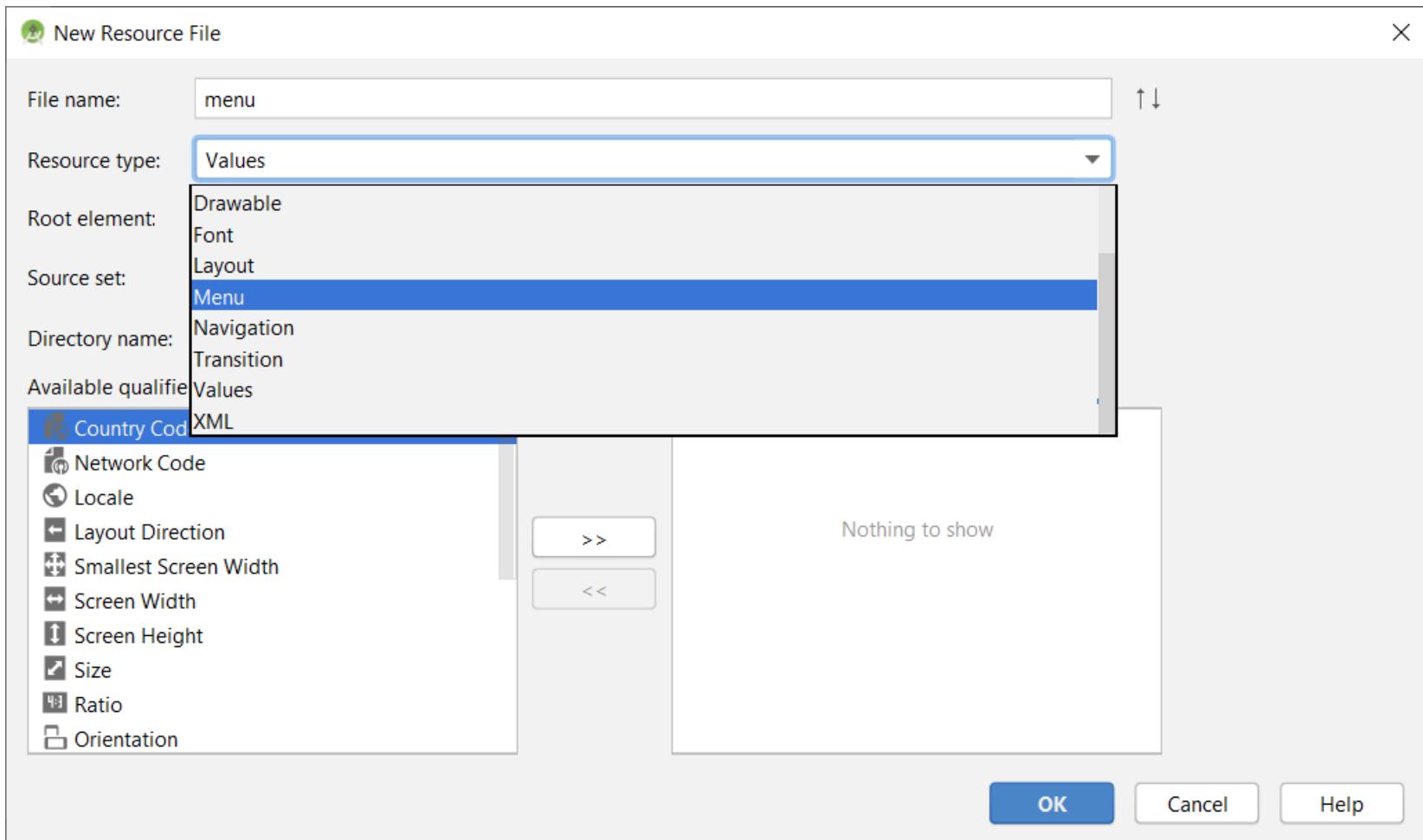
Directory name:

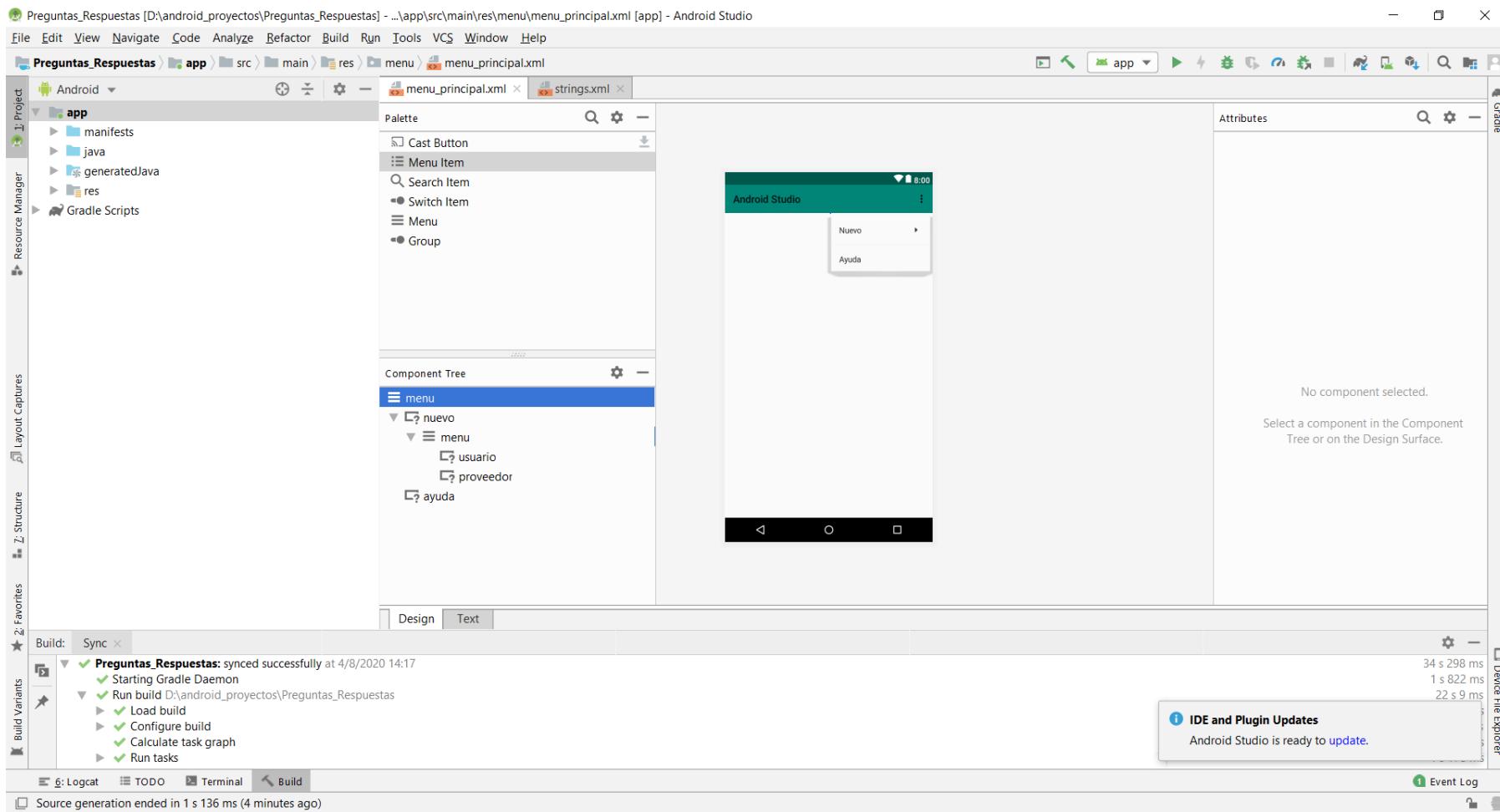
Available qualifiers:

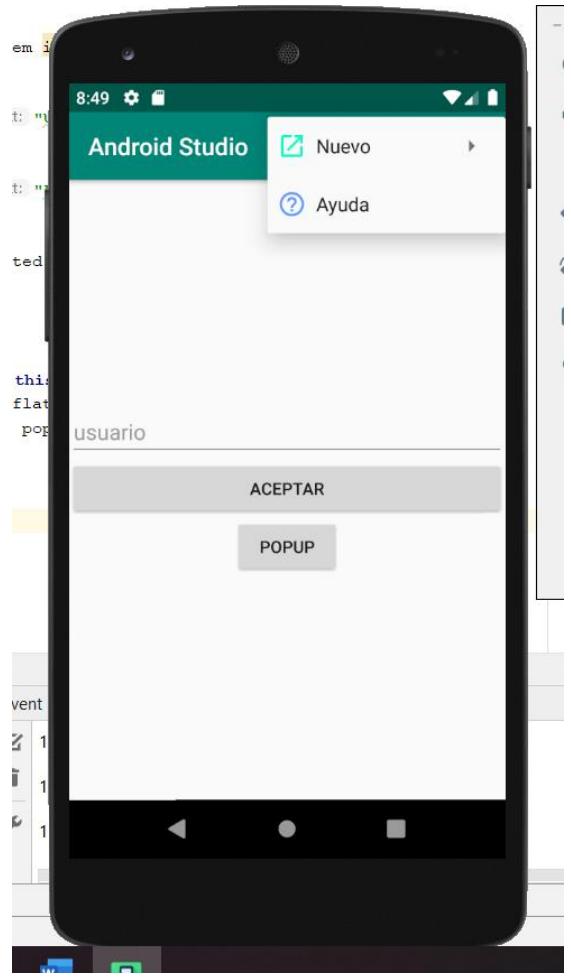
- Country Code
- Network Code
- Locale
- Layout Direction
- Smallest Screen Width
- Screen Width
- Screen Height
- Size
- Ratio
- Orientation

Chosen qualifiers:

Nothing to show





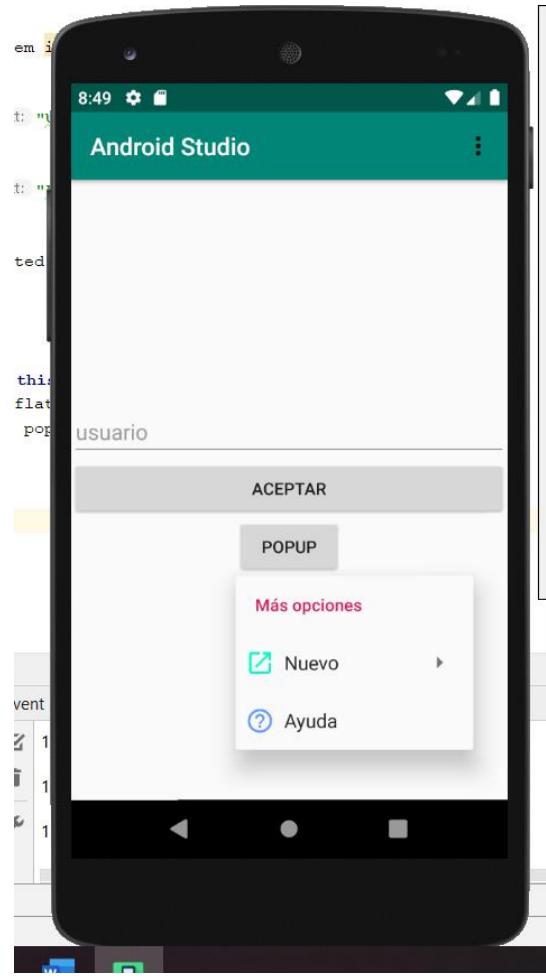


```
<item
    android:id="@+id/mas"
    android:icon="@drawable/ic_more"
    android:title="Ver Mas"
    app:showAsAction="always">
<menu>
    <item
        android:id="@+id/nuevo"
        android:icon="@drawable/ic_open_in_new_black_24dp"
        android:title="Nuevo">

    <menu>
        <item
            android:id="@+id/usuario"
            android:icon="@drawable/ic_people"
            android:title="Usuario" />
        <item
            android:id="@+id/proveedor"
            android:icon="@drawable/ic_domain_black_24dp"
            android:title="Proveedor" />
    </menu>
</item>
<item
    android:id="@+id/ayuda"
    android:icon="@drawable/ic_help_24dp"
    android:title="Ayuda" />
</menu>
</item>
```

```
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.usuario:
            Toast.makeText( context: this, text: "Usuario", Toast.LENGTH_LONG ).show();
            return true;
        case R.id.proveedor:
            Toast.makeText( context: this, text: "Proveedor", Toast.LENGTH_LONG ).show();
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}

public void showPopup(View v) {
    PopupMenu popup = new PopupMenu( context: this, v);
    MenuInflater inflater = popup.getMenuInflater();
    inflater.inflate(R.menu.menu_emergente, popup.getMenu());
    popup.show();
}
```



RECYCLERVIEW

RECYCLERVIEW

- La clase RecyclerView nos permite **mostrar un listado (o bien una grilla)** de elementos.
- Lleva este nombre porque a medida que se renderizan los elementos de la lista, los elementos que dejan de observarse se reciclan para mostrar los elementos siguientes.

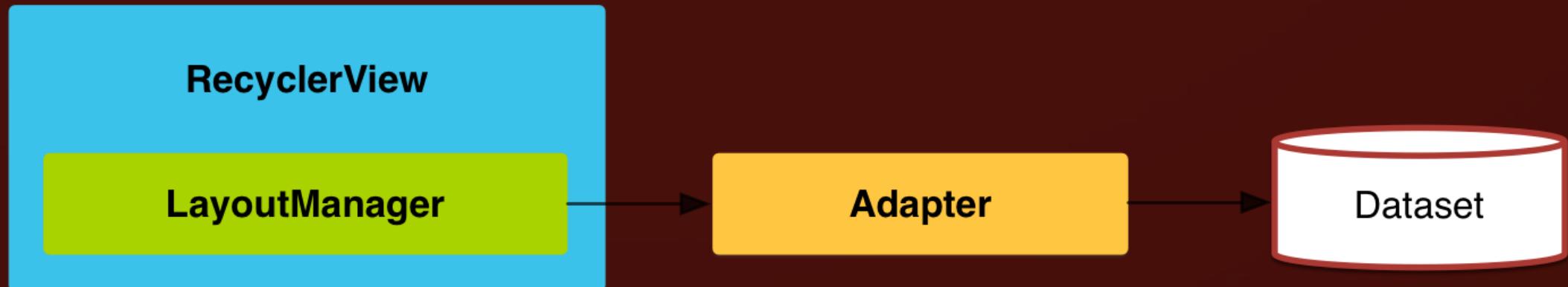
RECYCLERVIEW

- RecyclerView es una versión mejorada de la clase ListView, principalmente cuando el número de elementos es variable, y/o los datos cambian continuamente.

QUE ES UN CARDVIEW

- Si bien un **RecyclerView** representa una lista de elementos, cada elemento debe tener una UI definida.
- Al usar Material Design se suele usar la clase **CardView para definir la apariencia de cada elemento** de un listado, en la mayoría de los casos.

DIAGRAMA



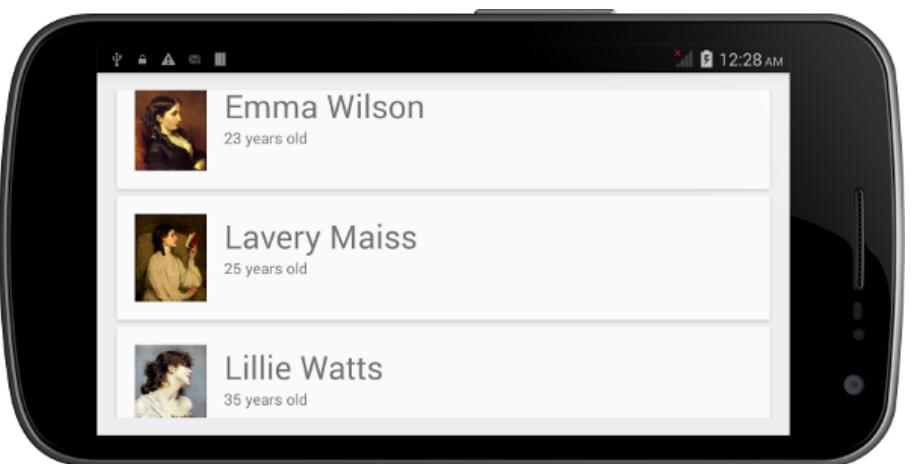
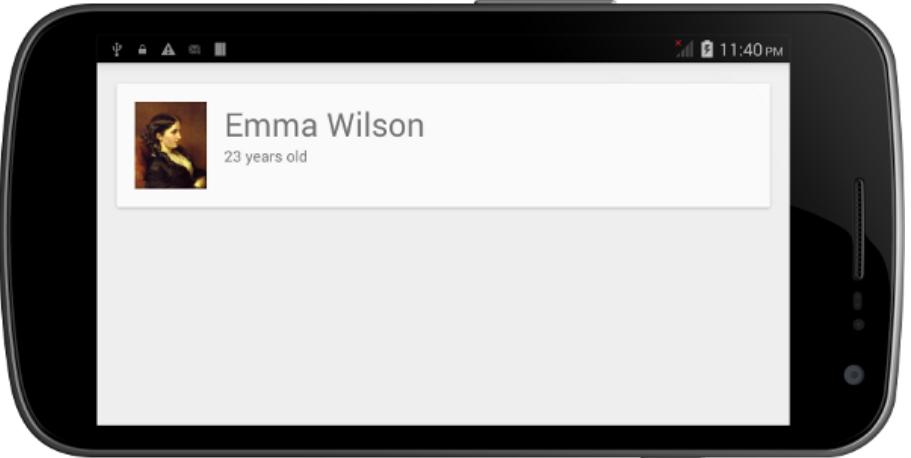
RECYCLERVIEW + CARDVIEW, ¿SIEMPRE JUNTOS?

No es obligatorio que se usen en conjunto, pero es usual hacerlo.

- ¿En qué casos no se usarían en conjunto?
 - Hay ocasiones en que se desea mostrar algo puntual. Entonces un elemento puede ser tan simple como un TextView, sin usar un CardView como contenedor.
 - Es posible usar CardViews directamente sobre un Activity o Fragment, sin situarlos al interior de un RecyclerView.
- Es posible usar cualquier otro componente visual para representar el layout de cada elemento del RecyclerView.

RESULTADO

- La idea es reciclar los elementos de la lista dinámica que no están visibles para el usuario, logrando un mejor rendimiento del aplicativo cuando existe una gran cantidad de elementos a mostrar.



Temas y Estilos

introducción

- ▶ Un estilo es una colección de propiedades que especifican que aspecto ha de tener un objeto View o una ventana. Con los estilos podemos definir propiedades como la altura, relleno, color del texto, fondo etc. Los estilos en Android comparten la filosofía de las hojas de estilo en cascada (CSS), permitiendo separar el diseño del contenido.

Código más limpio

Sin estilos:

```
<textview android:layout_width="fill_parent"  
         android:layout_height="wrap_content"  
         android:textColor="#00FF00"  
         android:typeface="monospace"  
         android:text="@string/hello" />
```

Con estilos:

```
<textview style="@style/CodeFont"  
         android:text="@string/hello" />
```

Definir estilos

- ▶ Los estilos se definen en un fichero de recursos distinto al layout, colocado en el directorio `/res/values`. Éstos ficheros no tienen porqué tener un nombre en concreto, pero por convención se suelen llamar **`styles.xml`** ó **`themes.xml`**

Definir estilos

- ▶ Para cada estilo hay que definir un elemento Y un atributo *name* para ese estilo (es obligatorio), después, añadiremos un elemento <item> para cada propiedad del estilo, que debe tener obligatoriamente el atributo ***name*** que declara la propiedad del estilo y su valor.

- ▶ Los valores para <item> puede ser una palabra clave, valor hexadecimal, una referencia a un recurso u otro valor dependiendo de la propiedad del estilo

```
<style name="CodeFont" parent="@android:style/TextAppearance.Medium">
    <item name="android:layout_width">fill_parent</item>
    <item name="android:layout_height">wrap_content</item>
    <item name="android:textColor">#00FF00</item>
    <item name="android:typeface">monospace</item>
</style>
```

- ▶ En tiempo de compilación, los elementos se convierten en un recurso que podremos referenciar posteriormente mediante el atributo **name** del estilo, como vimos en el primer ejemplo (**@style/CodeFont**).
- ▶ El atributo **parent** es opcional y especifica el ID de otro estilo del cual queremos heredar sus propiedades, pudiendo así sobreescribirlas.

Herencia

- ▶ El atributo parent sirve para heredar propiedades de otros estilos, podemos heredar tanto de estilos del sistema como de los nuestros propios.

Del sistema:

```
<style name="GreenText" parent="@android:style/TextAppearance">
    <item name="android:textColor">#00FF00</item>
</style>
```

De nuestros propios estilos:

```
<style name="CodeFont.Red">
    <item name="android:textColor">#FF0000</item>
</style>
<style name="CodeFont.Red.Big">
    <item name="android:textSize">30sp</item>
</style>
```

Aplicar estilos y temas a la interfaz gráfica

Hay dos formas de aplicar estilos a la UI:

- A una View individual, añadiendo el atributo style a un elemento del layout.
- A una aplicación o actividad completa, mediante el atributo **android:theme** del elemento `<activity>` o `<application>` en el Android manifest.

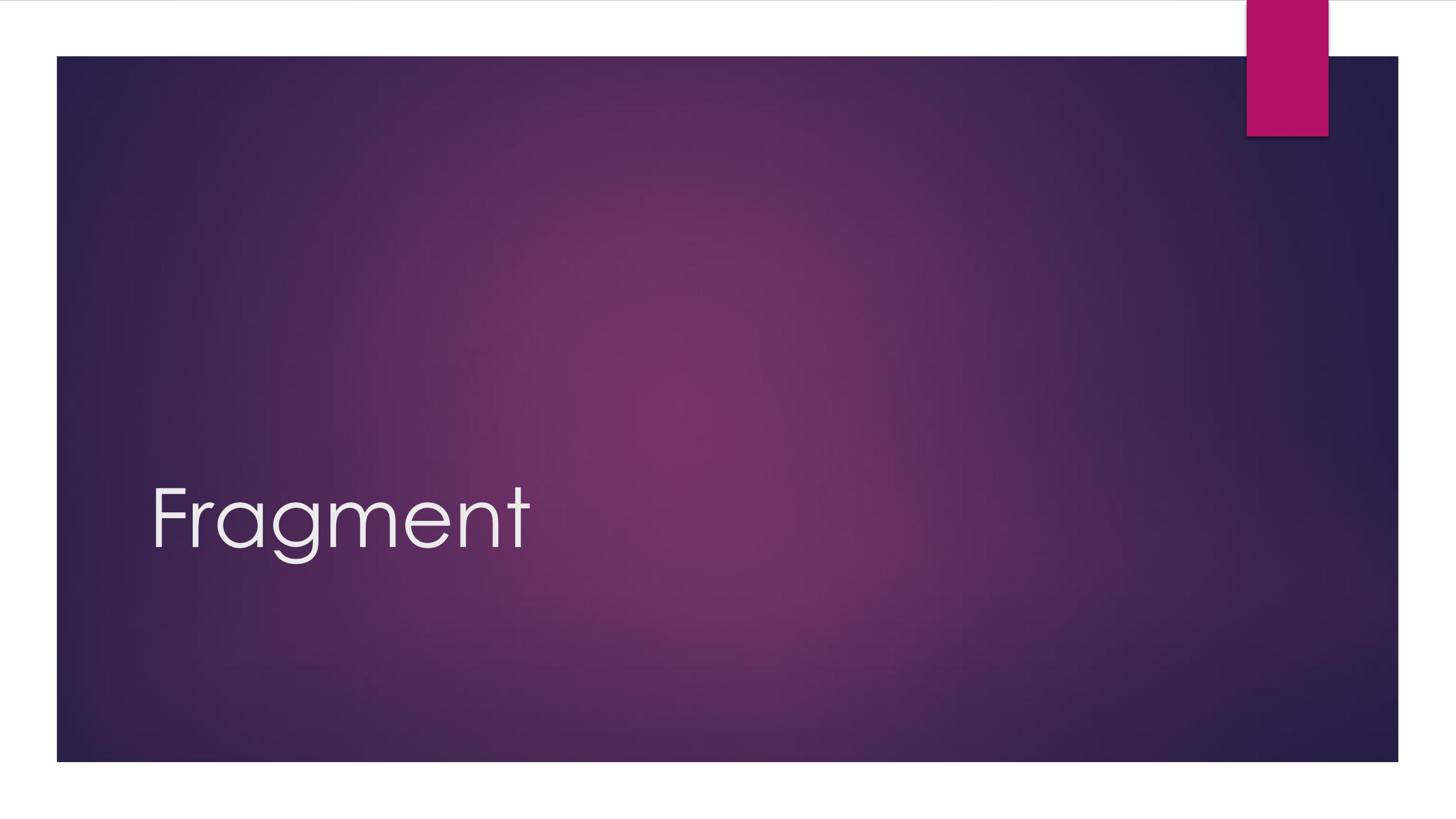
Como vimos al principio, para aplicar un estilo a una View concreta usamos **style="@style/NombreDelEstilo**

Para aplicar un tema a una actividad o aplicación usaremos:

```
<application android:theme="@style/CustomTheme">  
    </application>
```

Para aplicarlos sobre actividades, usamos:

```
<activity android:theme="@android:style/Theme.Dialog"></activity>  
<activity android:theme="@android:style/Theme.Translucent"></activity>
```



Fragment

Introducción

- ▶ Representa un comportamiento o una parte de la interfaz de usuario en una FragmentActivity. Puedes combinar varios fragmentos en una sola actividad para crear una IU multipanel y volver a usar un fragmento en diferentes actividades.
- ▶ Puede pensar en un fragmento como una sección modular de una actividad que tiene un ciclo de vida propio, que recibe sus propios eventos de entrada

Importante

- ▶ Un fragmento siempre debe estar alojado en una actividad y el ciclo de vida del fragmento se ve afectado directamente por el ciclo de vida de la actividad anfitriona.
- ▶ *Cuando la actividad está pausada, también lo están todos sus fragmentos, y cuando la actividad se destruye, lo mismo ocurre con todos los fragmentos.*

