Tema 3. Nuestra primera aplicación

Prof. Manuel Castillo

Programación de Dispositivos Móviles

Escuela Profesional de Ciencias de la Computación

Facultad de Ciencias

Universidad Nacional de Ingeniería

Objetivos

- Saber crear un proyecto Android vacío.
- Saber elegir las versiones de Android para el proyecto.
- Conocer la estructura básica de un proyecto Android.
- Saber ejecutar una aplicación Android.

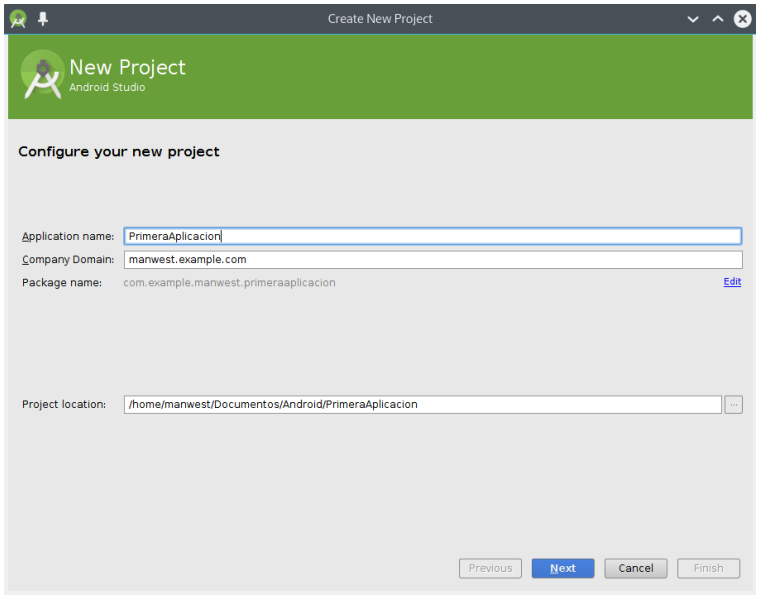
Índice de contenido

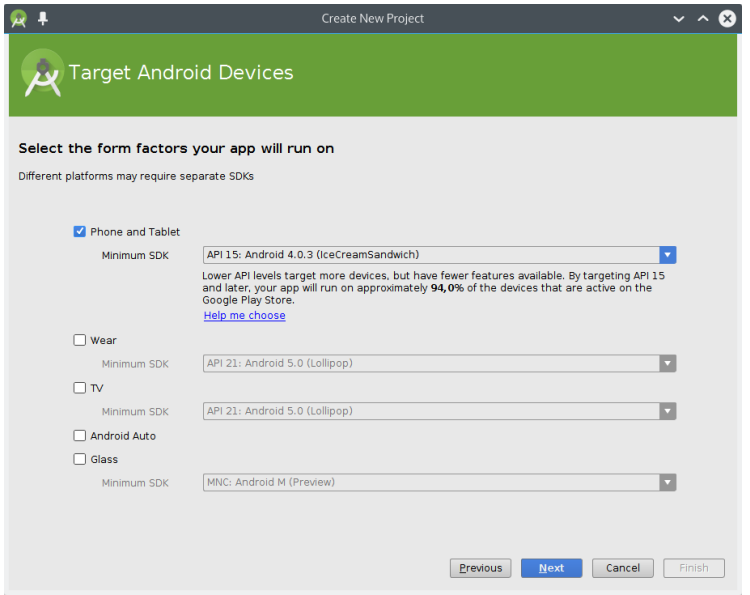
- Creando el Proyecto.
- Componentes del proyecto.
- “Hola Mundo” al detalle.
- Probando nuestra aplicación.

1. Creando el proyecto

1.1. Pasos (I)

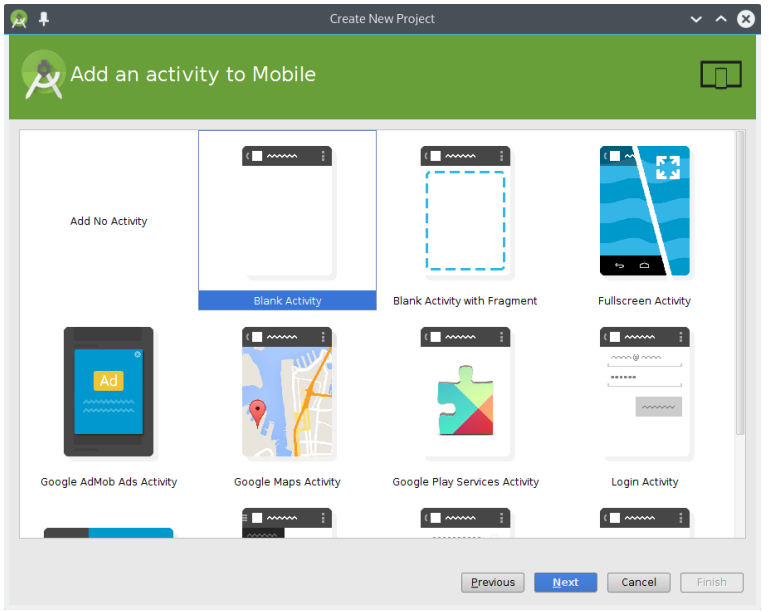
- New Project
  - Nombre de la Aplicación
  - Nombre de la compañía
    - Paquete de la app
  - Localización
- Plataformas/Factores
  - App: Phone/Tablet
  - TV: Android TV
  - Wear: Android Wear
  - Glass: Google Glass

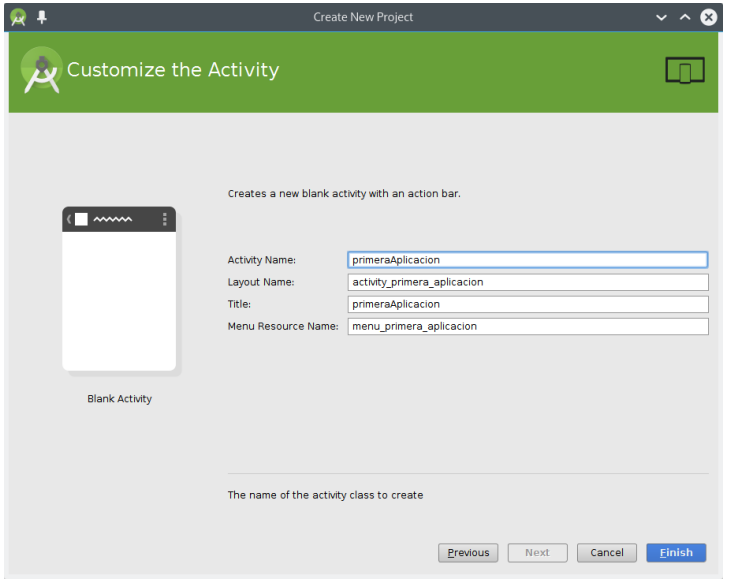




1.1. Pasos (II)

- Activity
  - *Blank Activity*
  - Next
- Nombre de la Clase
- Nombre del Layout: fichero *XML*.
- Título de la App.
- Finish
  - Entonces *Gradle* crea el proyecto y abre la ventana del proyecto.



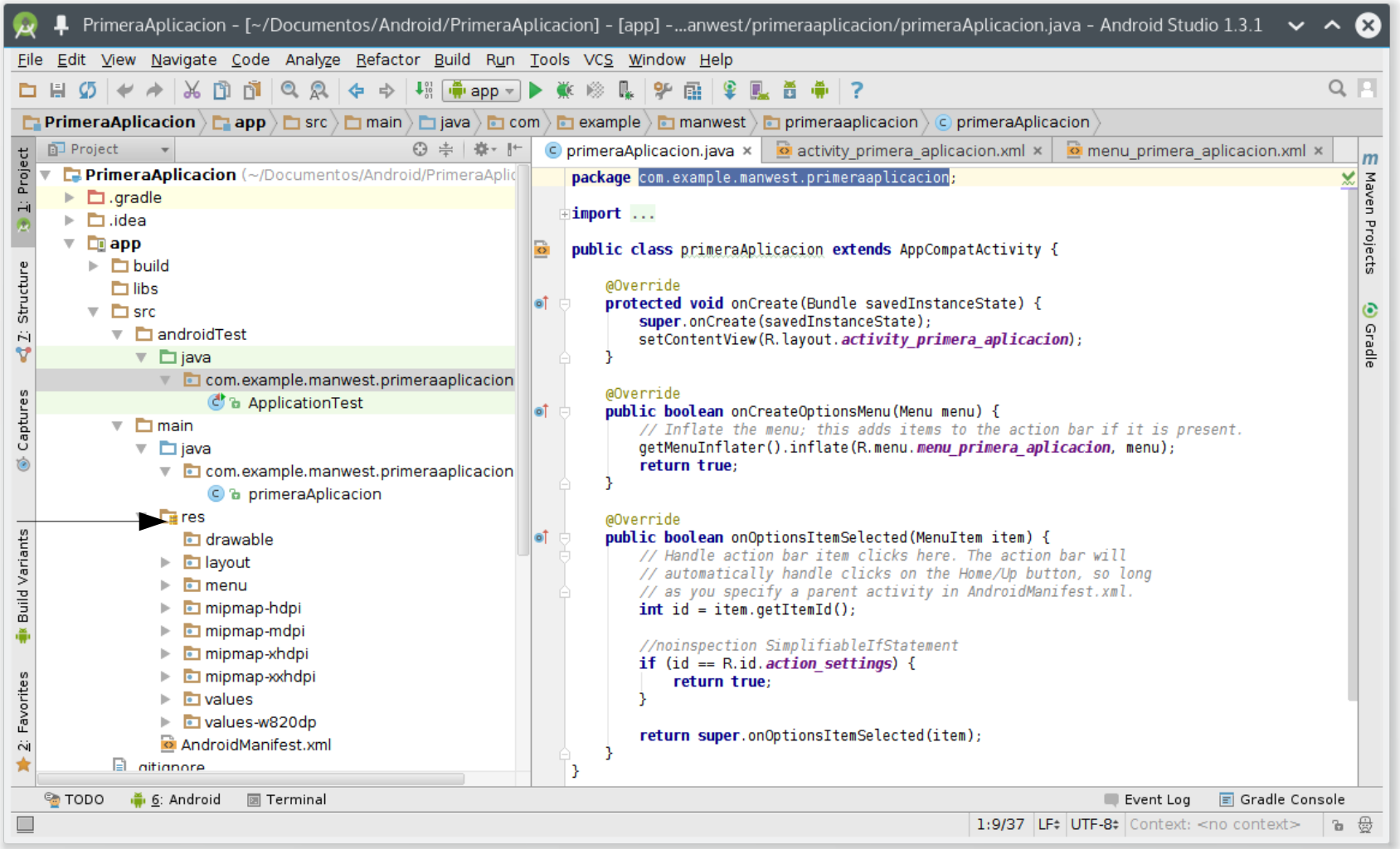


1.2. Estructura del proyecto (I)

- *.idea*: configuraciones del proyecto.
- *app*: directorio principal de desarrollo.
  - *build*: directorio de ficheros compilados.
  - *libs*: bibliotecas.
  - *src*: código fuente
    - *androidTest*: ficheros de pruebas.
    - *main*: ficheros principales ← **IMPORTANTE!!!!**
      - *java*: ficheros de programación. Código JAVA
      - *res*: ficheros de recursos.
      - *AndroidManifest*.xml: fichero de manifiesto.
- *build.gradle*: fichero de configuración de la compilación.

1.2. Estructura del proyecto (II)

Estructura principal del proyecto



1.2.1. build.gradle

primeraAplicacion.java × | app × | activity_primera_aplicacion.xml ×

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 22
    buildToolsVersion "23.0.0 rc3"

    defaultConfig {
        applicationId "com.example.manwest.primeraaplicacion"
        minSdkVersion 15
        targetSdkVersion 22
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:appcompat-v7:22.2.1'
}
```

Versiones a compilar

Especifica nivel mínimo de la API para ejecutar la aplicación (minSdkVersion), nivel en que se ha diseñado (targetSdkVersion).

A la hora de importar proyectos la modificación de este fichero es muy importante!!!!! Hacerlo desde File → Project Structure

2. Componentes de aplicaciones

2.1. Actividad

Empezamos por Actividad

- Componente central de la plataforma Android.
- Representa una tarea que puede realizar la aplicación y son independientes unas de otras.
- Lo normal es que haya al menos una, que represente una ventana para interactuar con el usuario. Heredan de la clase *AppCompactctivity*.
- Las actividades se componen de **fragmentos**, que facilitan la funcionalidad de la aplicación tanto para *tablets* como para *smartphones*.

```
import android.support.v7.app.AppCompatActivity;

public class primeraAplicacion extends AppCompatActivity {
    //código omitido
}
```

2.2. Servicio

- Permiten ejecutar operaciones en segundo plano, siendo independientes de cualquier actividad. Heredan de la clase *Service*.

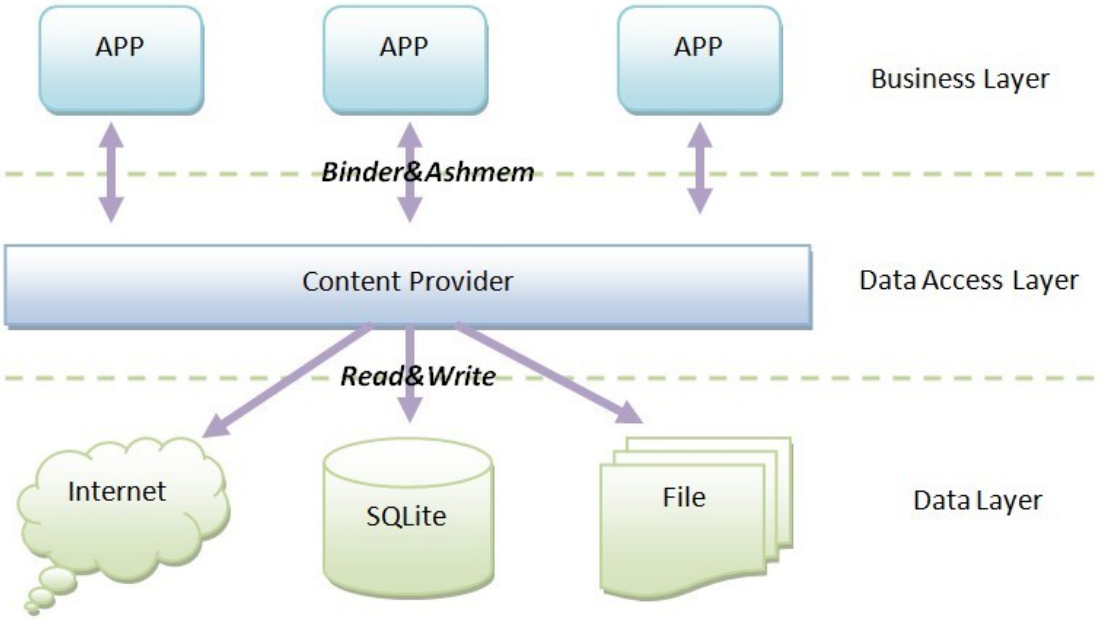
```
import android.app.Service;

public class MiServicio extends Service {
        //código omitido
}
```

- Se lanzan a través de la clase *Context*, gracias a los métodos *startService*() y *bindService*().

```
Context.startService();  //Ejecuta el servicio, indefinidamente
                                    //en segundo plano

Context.bind.Service(); //Ejecuta el servicio, permitiendo controlar
                                  //su ejecución desde una actividad
```

2.3. Proveedor de contenido

- Se encargan de almacenar datos y gestionar el acceso a ellos, de forma similar a una base de datos relacional, permitiendo compartir datos entre aplicaciones.
- Podemos crear nuestro propio proveedor extendiendo la clase abstracta *ContentProvider*.



2.4. Receptor de eventos

- Recibe y reacciona ante notificaciones emitidas por el sistema o la misma aplicación.
- Podemos crear nuestro propio receptor de eventos extendiendo la clase *BroadcastReceiver*.
- Se activan mediante los *Intents*, objetos que contienen los datos del mensaje que pueden activar también a servicios o actividades.

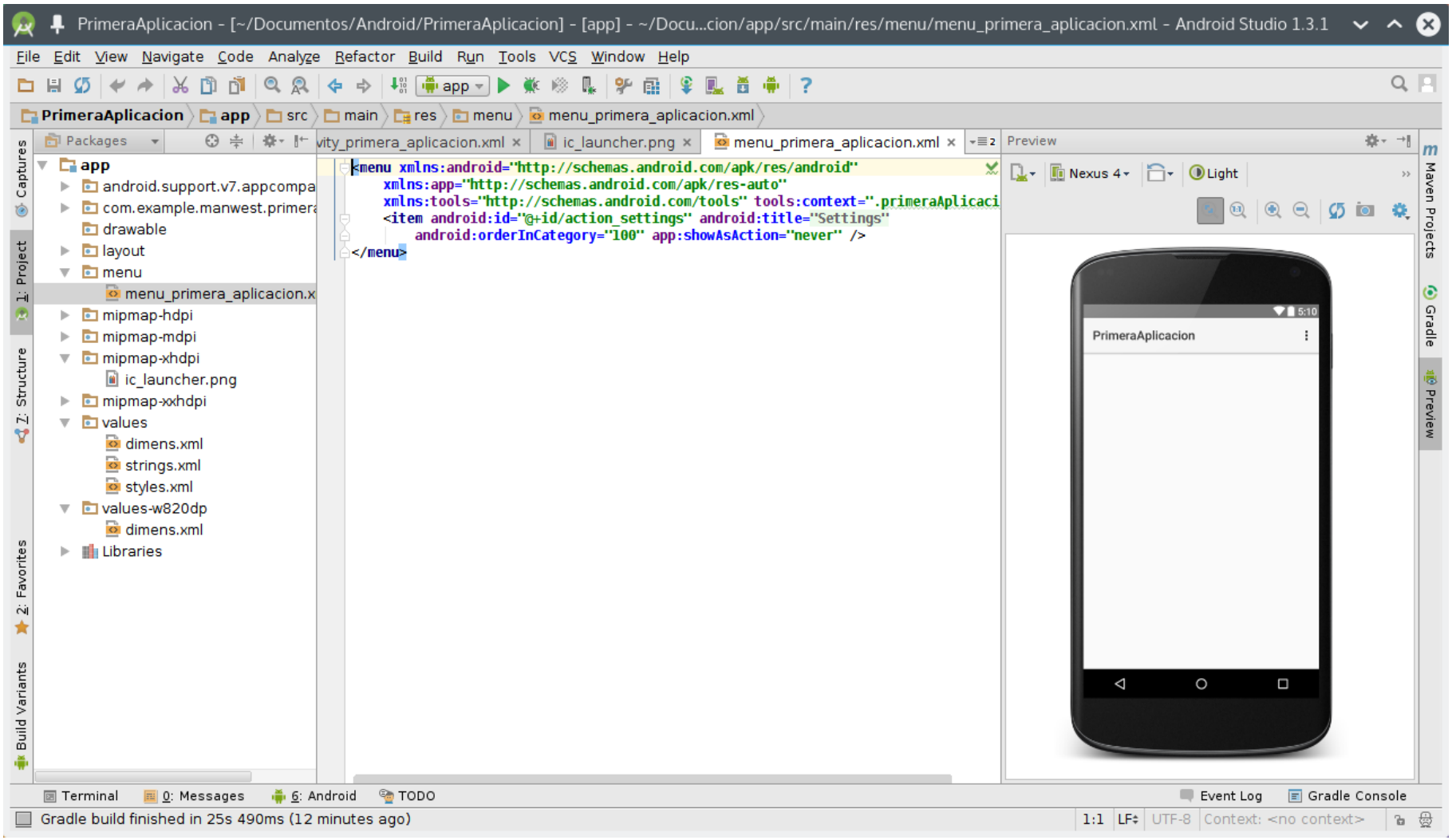
```
import android.content.BroadcastReceiver;

public class ReceptorSMS extends BroadcastReceiver{
      //código omitido
}
```

3. Componentes del proyecto

3.1. Ficheros/Directorios (I)

- AndroidManifest.xml: fichero de configuración principal de la app.
- Activities: controladores principales del pantallas, código Java.
- Layouts: Configuración de la Vista del proyectos, pantalla, ficheros *XML*.
- Drawable: Fichero de imagen, *jpg*, *png*, etc. Dividida dependiendo de la resolución/puntos por pulgada.
- Values: ficheros de configuración de vista general y traducciones, *XML*.
- Menu: configuración de menús y barras de navegación, *XML*.

3.1. Ficheros/Directorios (II)



3.2. Proceso de ejecución de una app

1. El Manifiesto identifica el paquete principal y las Actividades.
2. Las actividades marcan cual es la actividad principal y se carga al pulsar sobre el icono.
3. Se carga la Actividad principal.
4. Desde la Actividad principal se carga el *Layout* o el diseño.
5. La aplicación queda pendiente de las acciones que puede realizar el usuario.

3.3. AndroidManifest

Etiqueta xml del fichero y paquete principal

Información sobre nuestra app, por ejemplo permite backup, icono...

Corresponde a la clase Java cuando se carga esa Activity

Título en la barra superior

Define como activity principal a la actividad y su ejecución al arrancarla

Application: nos dice el número de aplicaciones que hemos definido

primeraAplicacion.java × | app × | activity_primera_aplicacion.xml × | AndroidManifest.xml ×

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.manwest.primeraaplicacion" >

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="PrimeraAplicacion"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".primeraAplicacion"
            android:label="PrimeraAplicacion" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Maven Projects | Gradle

3.4. Activity JAVA (I)

- Clase Java que hereda de *AppCompatActivity* (antiguamente *Activity*).
- Dispone de un método principal llamado *onCreate*, que se ejecutará siempre que se cargue la actividad.
  - Dentro del método se carga la vista mediante un *setContentView*.
    - Se pasa como parámetro la vista que se desea cargar mediante una referencia.
      - Dicha referencia se basa en la clase *R*.

3.4. Activity JAVA (II)

Herencia principal

primeraAplicacion.java × | app × | activity_primera_aplicacion.xml ×

```
package com.example.manwest.primeraaplicacion;

import ...

public class primeraAplicacion extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_primera_aplicacion);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_primera_aplicacion, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }
}
```

Sobreescribimos el método del padre. Parámetro *Bundle* como configuración del arranque de la activity.
***Super*: Ejecuta *onCreate* de la clase padre.**
SetContentView: Carga automáticamente el diseño de ese diseño

***onCreateOptionsMenu*(): inicializa el contenido del menú de opciones estándar de una actividad. Puede tener como parámetro un objeto Menu.**

Conoce el elemento del MENÚ que han seleccionado el usuario.
El id se seleccionado por el usuario

3.5. Layout (I)

- Referenciados desde la clase ***R.layout.****.
- Marca el diseño de la pantalla.
- Es un fichero XML localizado en el directorio *res/layout*
- Tiene un elemento principal, normalmente una etiqueta de tipo **Layout*.
  - Tiene definidos el alto y el ancho por defecto.
  - Puede incluir otras vistas, por ejemplo un *TextView*, es decir un texto en pantalla no editable.
  - Dentro del *TextView* podemos colocar cadenas de caracteres directamente o haciendo referencia a un *value*, un literal.
    - Los literales están definidos en *res/values/strings.xml*

3.5. Activity xml (II)

Coloca elementos en la posición que queramos de manera relativa a la pantalla

primeraAplicacion.java × | app × | activity_primera_aplicacion.xml ×

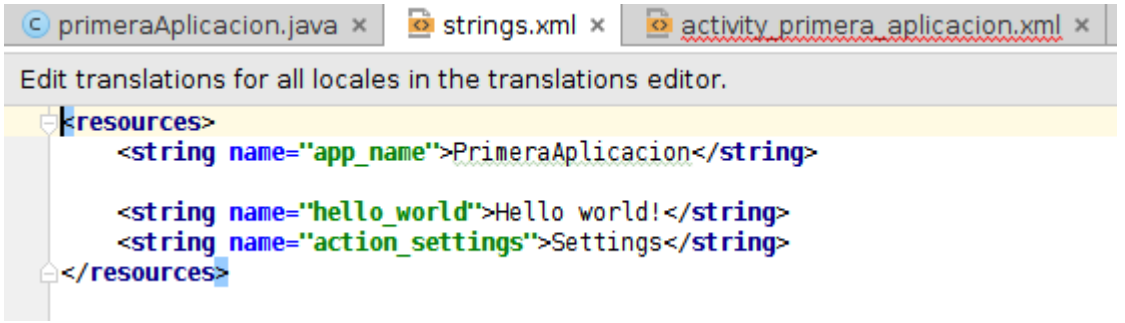
```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    android:paddingBottom="16dp"
    tools:context=".primeraAplicacion">

    <TextView android:text="Hello world!"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</RelativeLayout>
```

TextView: Etiqueta de texto no editable

Vemos que el texto está en una variable string de carpeta *values* → *strings.xml* "@string"

3.6. values/strings

- Define literales que luego pueden utilizarse desde el código Java y desde los *Layout*.
- La etiqueta más básica es <*string*> el atributo *name* define el nombre genérico, un literal,
  - El valor que va dentro de la etiqueta es el texto asociado a ese literal.



3.7. menu (I)

- Fichero xml del menú de la barra de acciones (*ActionBar*).
- Colocar un item por cada elemento, en nuestro caso uno solo que se llama "*Settings*".
  - También hace una referencia a una *string*.
- Las propiedades e información son importantes.

aplicacion.xml × | AndroidManifest.xml × | menu_primera_aplicacion.xml ×

```
<menu 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"
    tools:context=".primeraAplicacion">

    <item android:id="@+id/action_settings"
        android:title="Settings"
        android:orderInCategory="100"
        app:showAsAction="never" />
</menu>
```

3.8. Nuestra app

Layout

Menu