Aplicaciones Telemáticas

Tema 2.1.- Introducción a Android

J. E. López Patiño, F. J. Martínez Zaldívar

ETSIT-UPV

Índice

Índice

Información básica

Información general:

`http://www.android.com`

Desarrollo:

`http://developer.android.com/develop/index.html`

Herramientas:

`http://developer.android.com/intl/es/sdk/index.html`

Primeros pasos:

`http://developer.android.com/intl/es/training/index.html`

API:

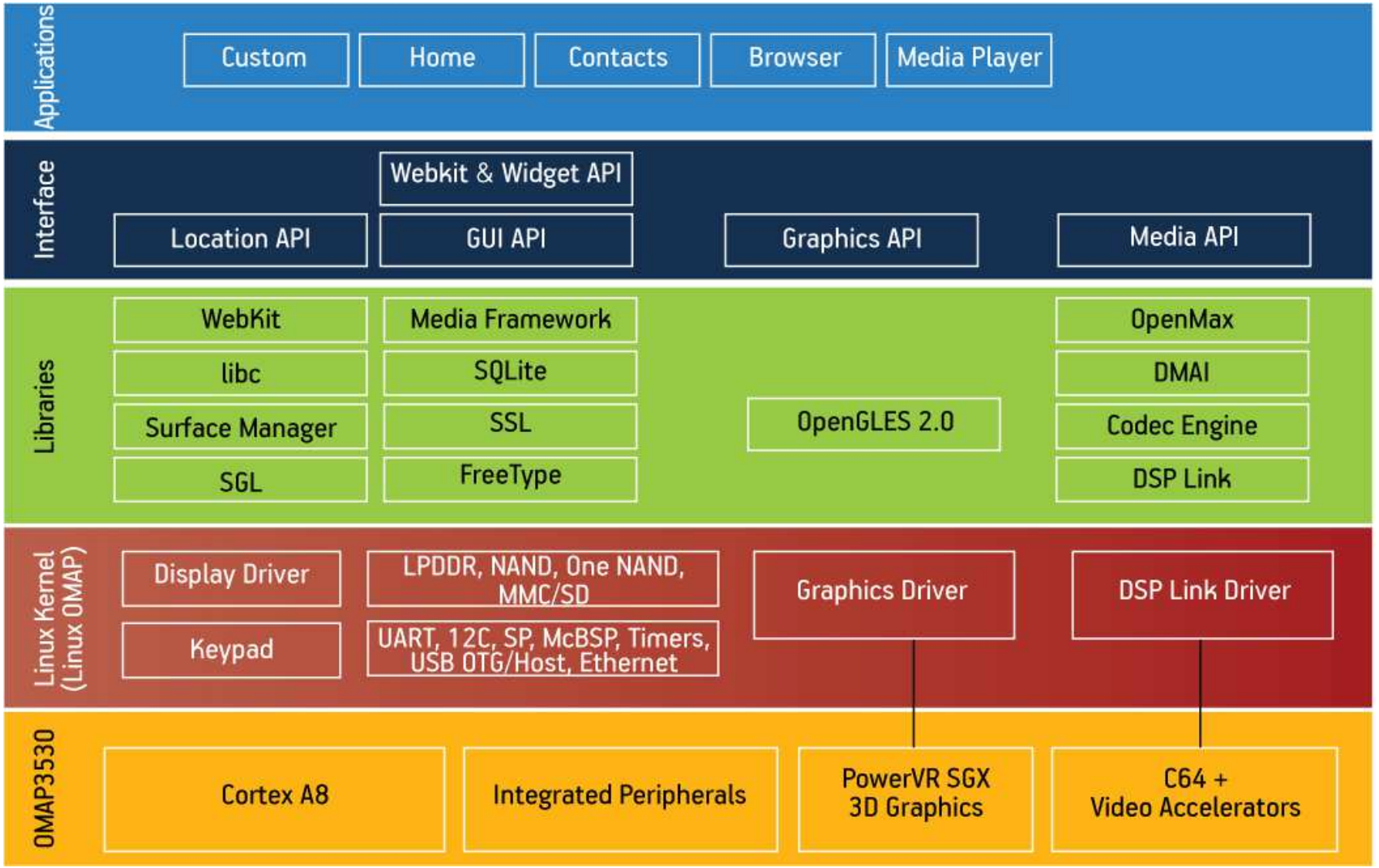
`http://developer.android.com/intl/es/reference/packages.html`

...

¿Qué es?

- Plataforma soft de Google y Open Handset Alliance
- Evolución:
  - 2005: Google compra Android Inc.
  - 2007: formación de Open Handset Alliance
    - Operadoras
    - Fabricantes terminales
    - Fabricantes semiconductores
    - Compañías de software
    - Comercializadoras
  - 2008: SDK 1.0, T-mobile G1
  - ...

Niveles de Android



Componentes de una aplicación Android

Activities *(actividades)*: equivalente a una ventana o caja de diálogo en apliación convencional de ordenador

Content providers *(proveedores de contenido)*: acceso de datos por parte de cualquier aplicación del sistema

Services *(servicios)*: aplicación sin interacción directa con usuario y con ciclo de vida típicamente más largo

Intents *(propósitos o intenciones)*: mensajes asíncronos que permite que unos componentes de Android utilicen otros componentes

Facilidades para el desarrollo

Almacenamiento: bases de datos, acceso a tarjeta SD, ficheros de datos, ...

Red: desde *sockets* de Java hasta *widgets* basados en WebKit

Multimedia: reproducción y grabación de sonidos, imágenes y video

GPS: localización y visualización en mapas...

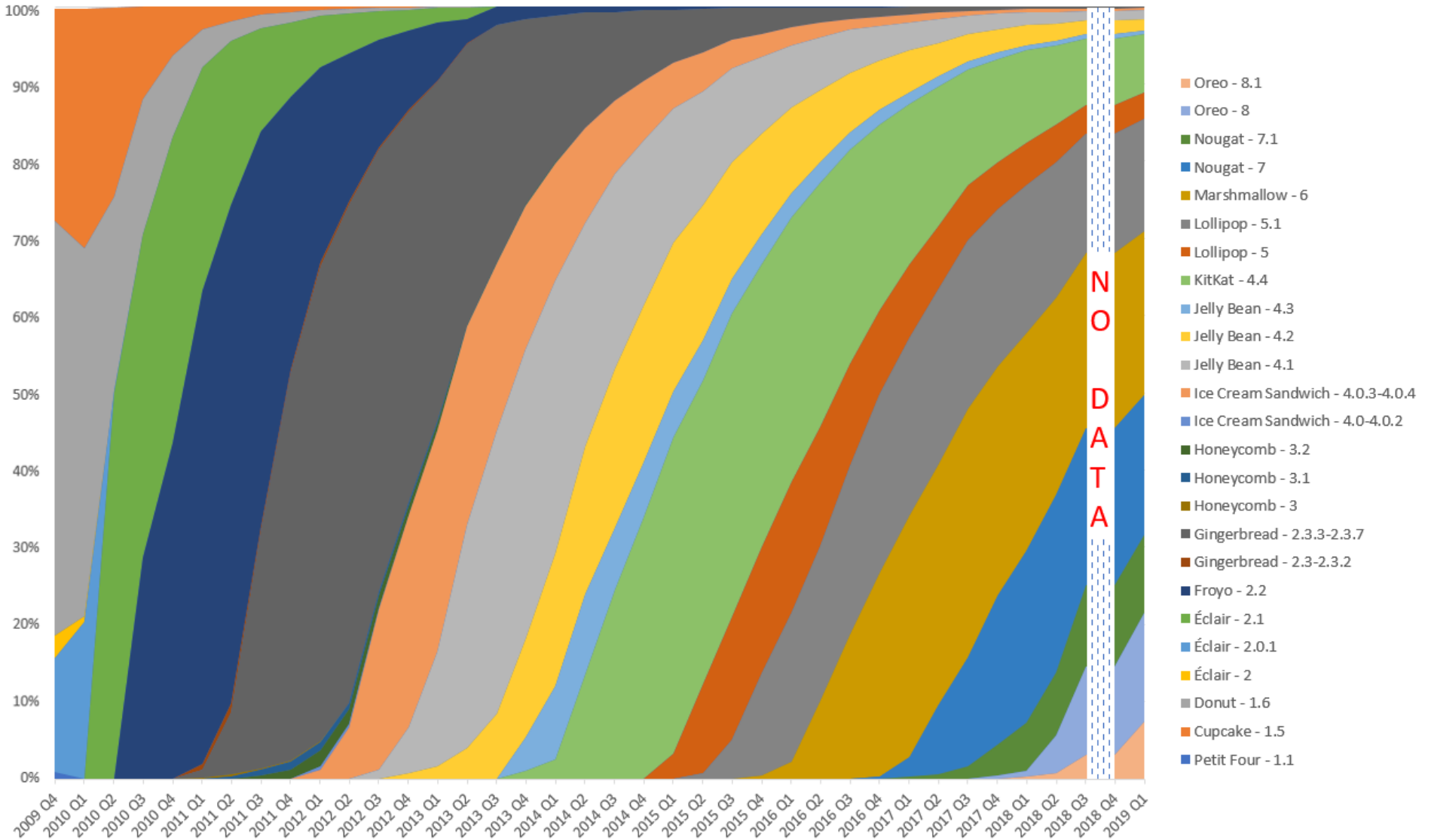
Servicios de teléfono: iniciar llamadas, enviar/recibir SMS, ...

Versión Android - nivel API

API *(Application Programming Interface)*:

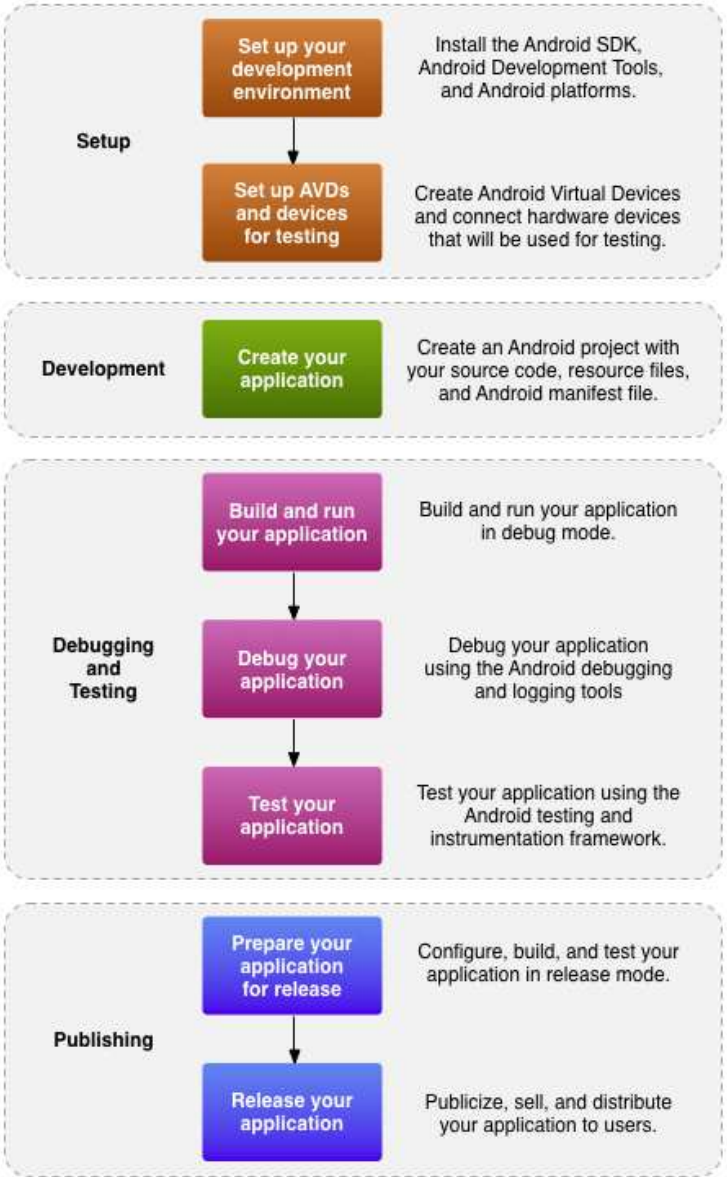
Version	Nivel API	VERSION_CODE
Android 10.0	29	'Q'
Android 9.0	28	PIE
Android 8.0 – 8.1	26 – 27	OREO
Android 7.0 – 7.1.2	24 – 25	NOUGAT
Android 6.0	23	MARSHMALLOW
Android 5.1	22	LOLLIPOP
Android 5.0 – 5.0.2	21	LOLLIPOP
Android 4.4W – 4.4W2	20	KITKAT WEARABLE
Android 4.4 – 4.4.4	19	KITKAT
Android 4.3 – 4.3.1	18	JELLY_BEAN_MR2
Android 4.2 – 4.2.2	17	JELLY_BEAN_MR1
Android 4.1, 4.1.2	16	JELLY_BEAN
Android 4.0.3 – 4.0.4	15	ICE_CREAM_SANDWICH_MR1
Android 4.0 – 4.0.2	14	ICE_CREAM_SANDWICH
Android 3.2 – 3.2.6	13	HONEYCOMB_MR2
Android 3.1	12	HONEYCOMB_MR1
Android 3.0	11	HONEYCOMB
Android 2.3.3 – 2.3.7	10	GINGERBREAD_MR1
Android 2.3 – 2.3.2	9	GINGERBREAD
Android 2.2 – 2.2.3	8	FROYO
Android 2.1	7	ECLAIR_MR1
Android 2.0.1	6	ECLAIR_0_1
Android 2.0	5	ECLAIR
Android 1.6	4	DONUT
Android 1.5	3	CUPCAKE
Android 1.1	2	BASE_1_1
Android 1.0	1	BASE

Versiones Android: proporciones



Índice

Flujo de desarrollo



Herramientas

Imprescindible: SDK *(Software Development Kit)*. Herramientas para:

- Creación de proyectos
- Compilación
- Emulación
- Depuración

Ejecución en línea de comandos

Opcional: alternativa a línea de comandos

- IDE *(Integrated Development Environment)*:
  - `Android Studio`: basado en `IntelliJ IDEA`
  - Anteriormente: `Eclipse`

Disponible para múltiples sistemas operativos: Windows, Linux, OS-X, ...

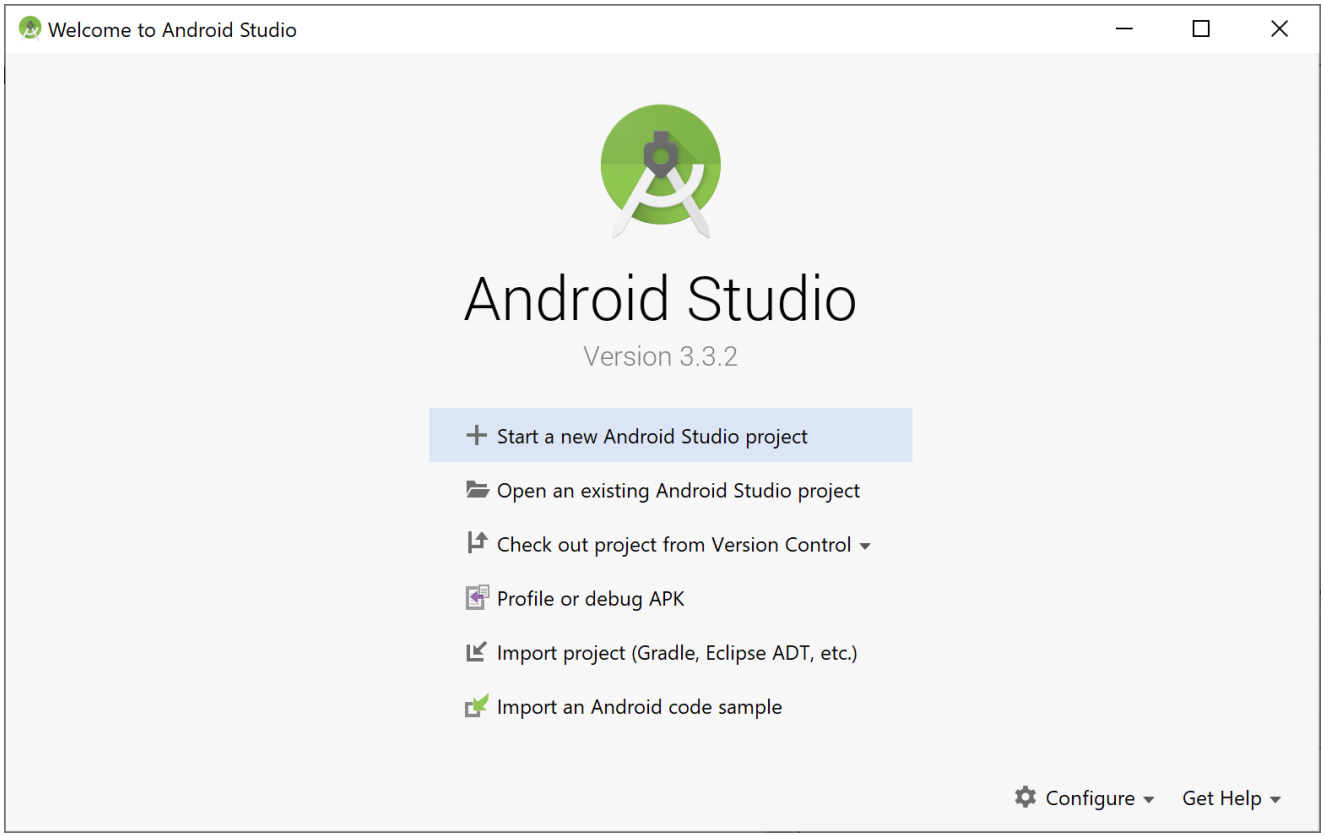
Android Studio

- Instalación de `Android Studio` desde:

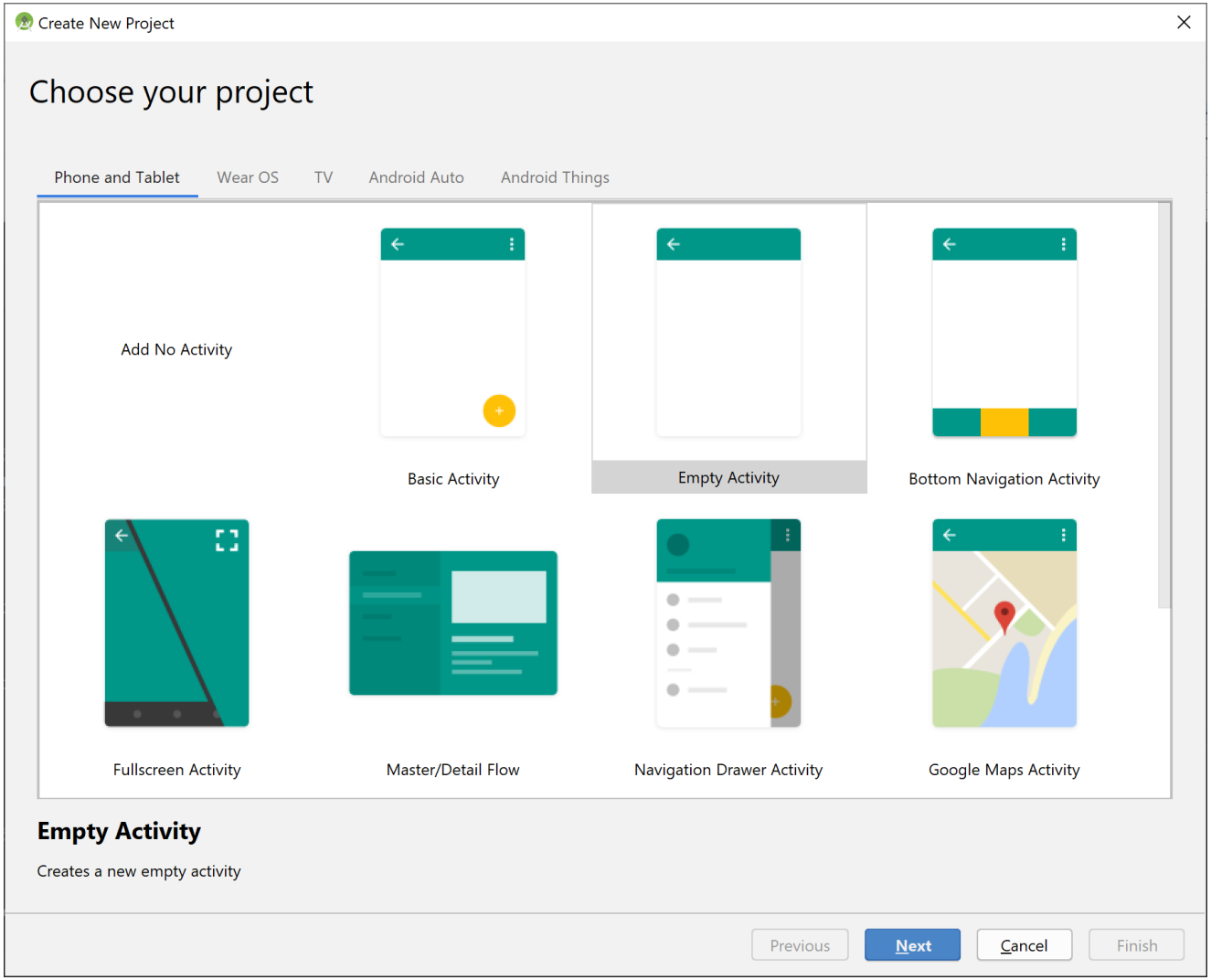
  `https://developer.android.com/studio`

- Bienvenida: configuración y actualización del `SDK Manager`

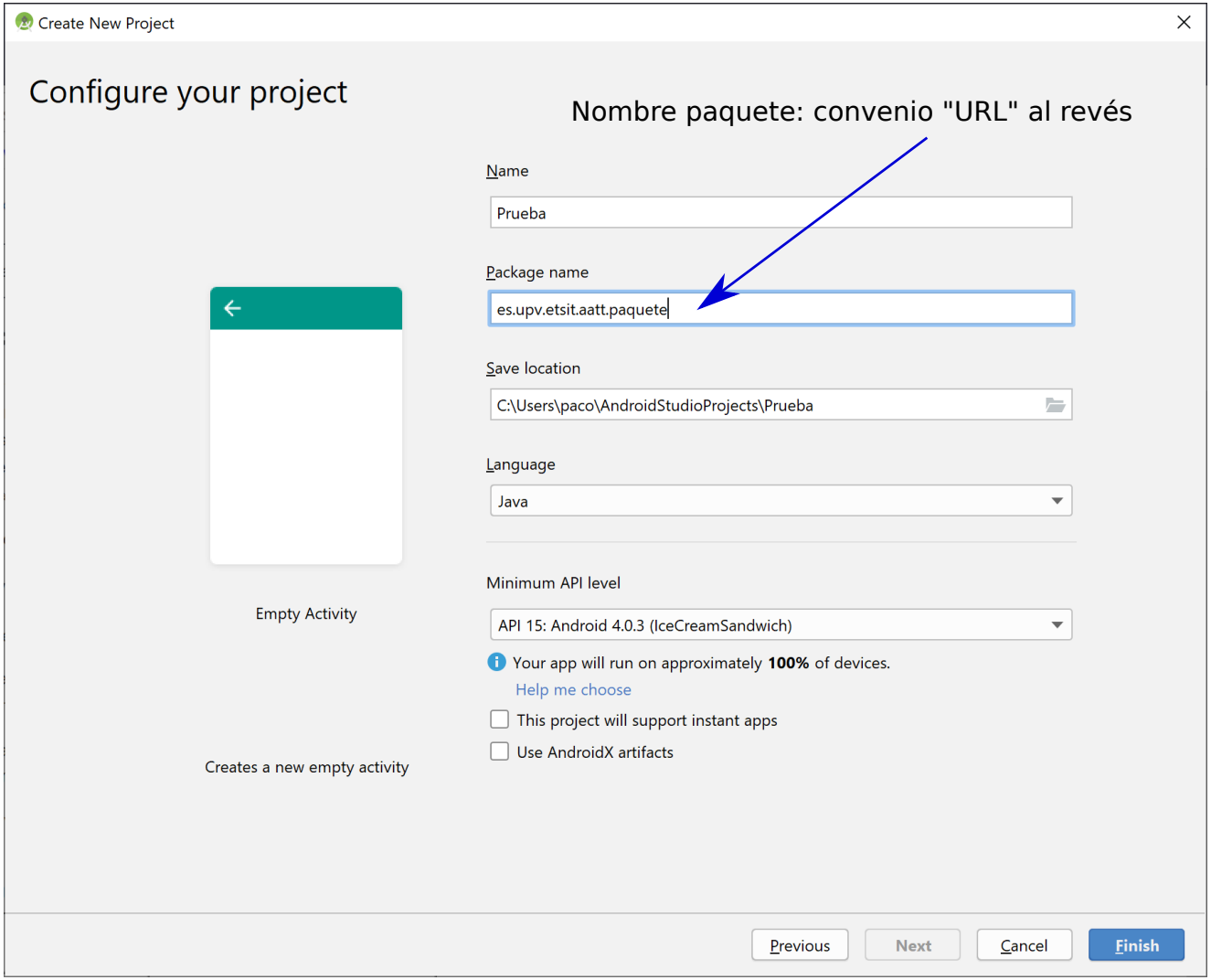
Creación proyecto



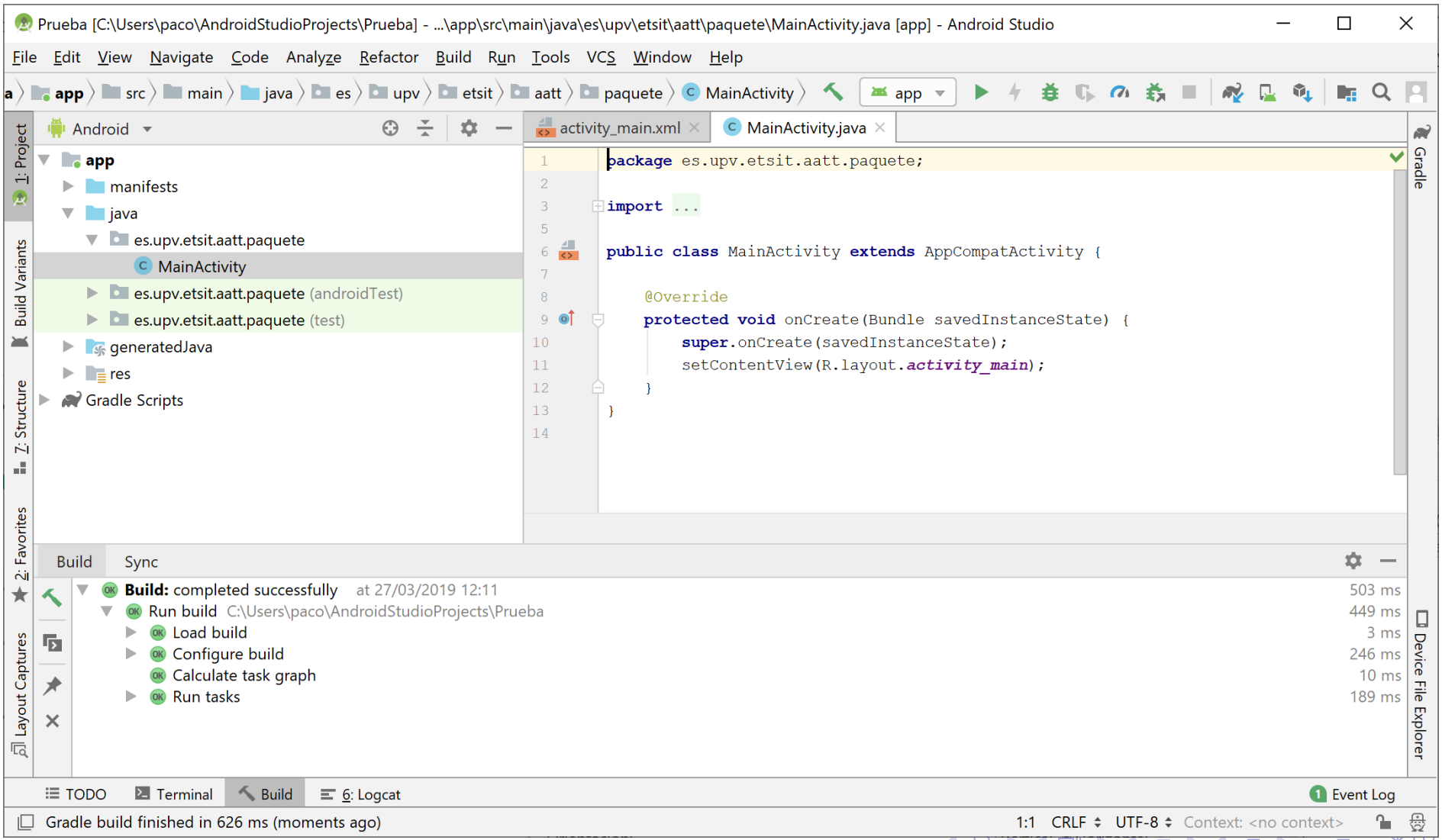
Elección de plantilla o tipo de proyecto



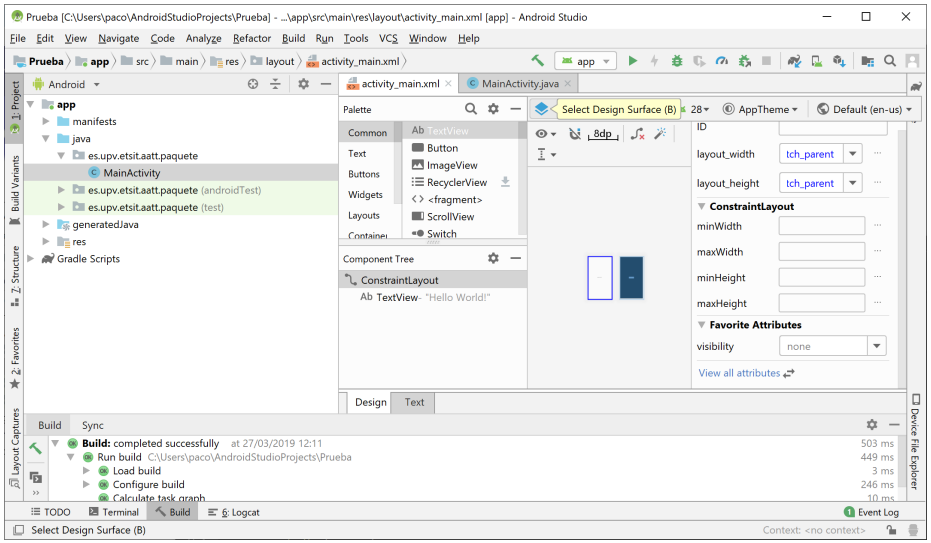
Configuración del proyecto

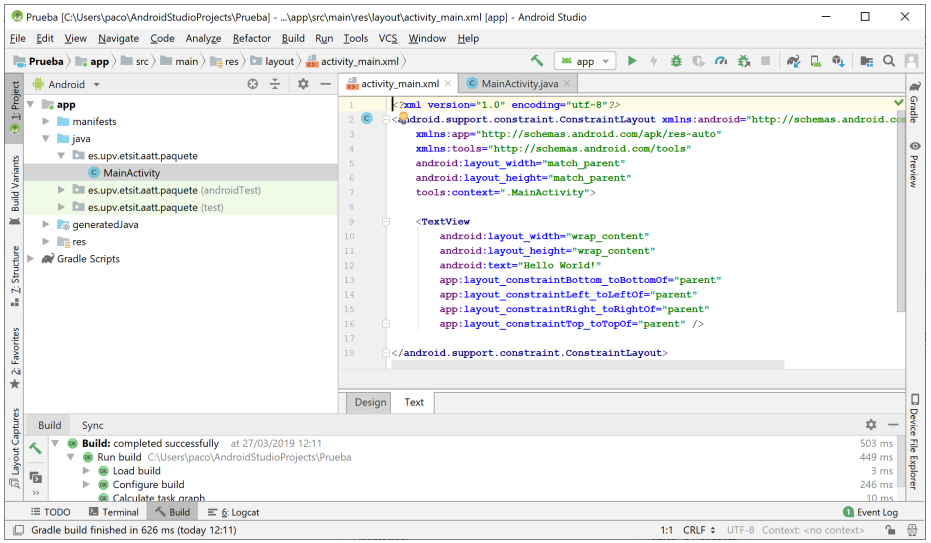


IDE Android Studio: primera aplicación (MainActivity.java)



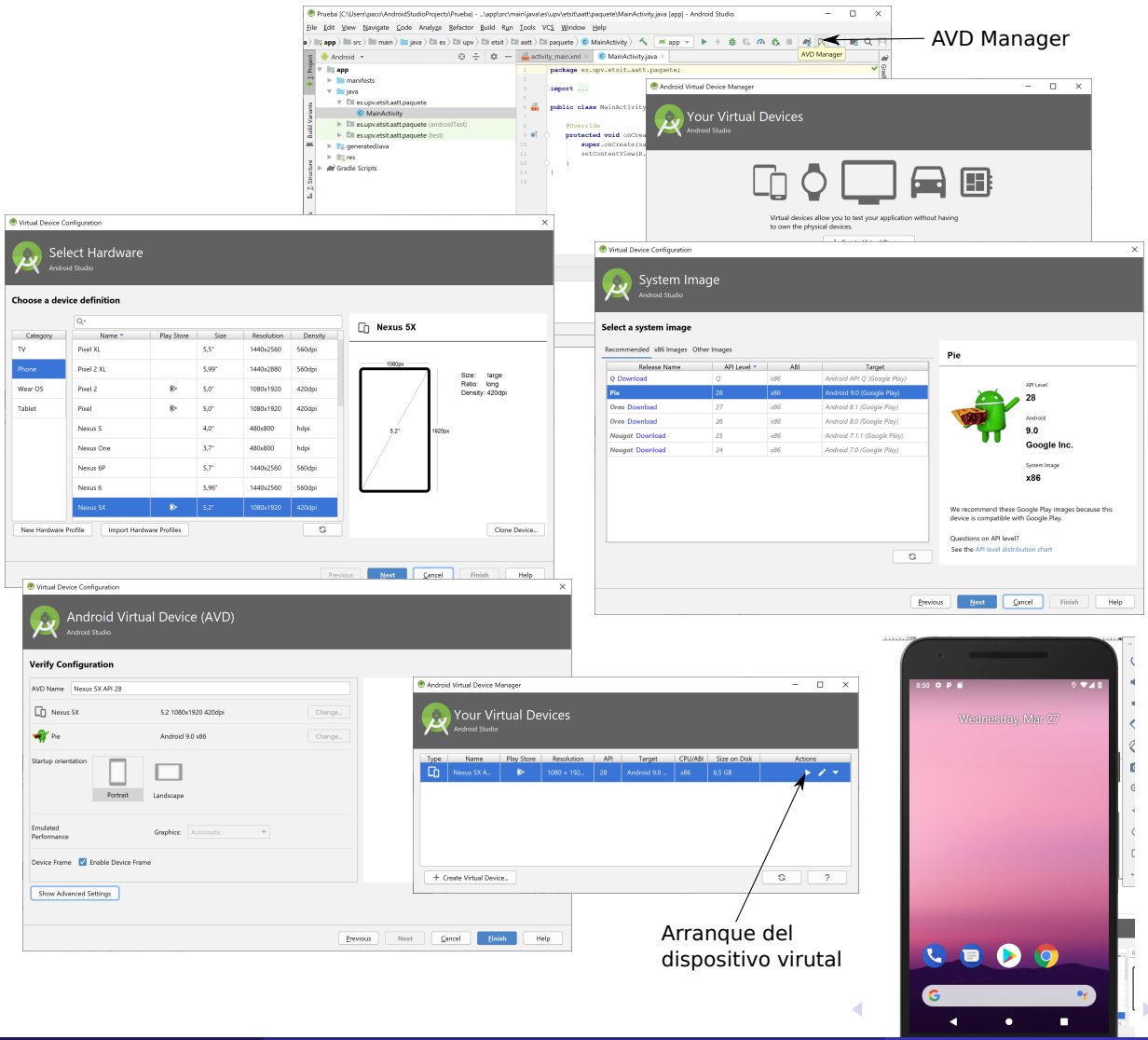
IDE Android Studio: primera aplicación (`activity_main.xml`)



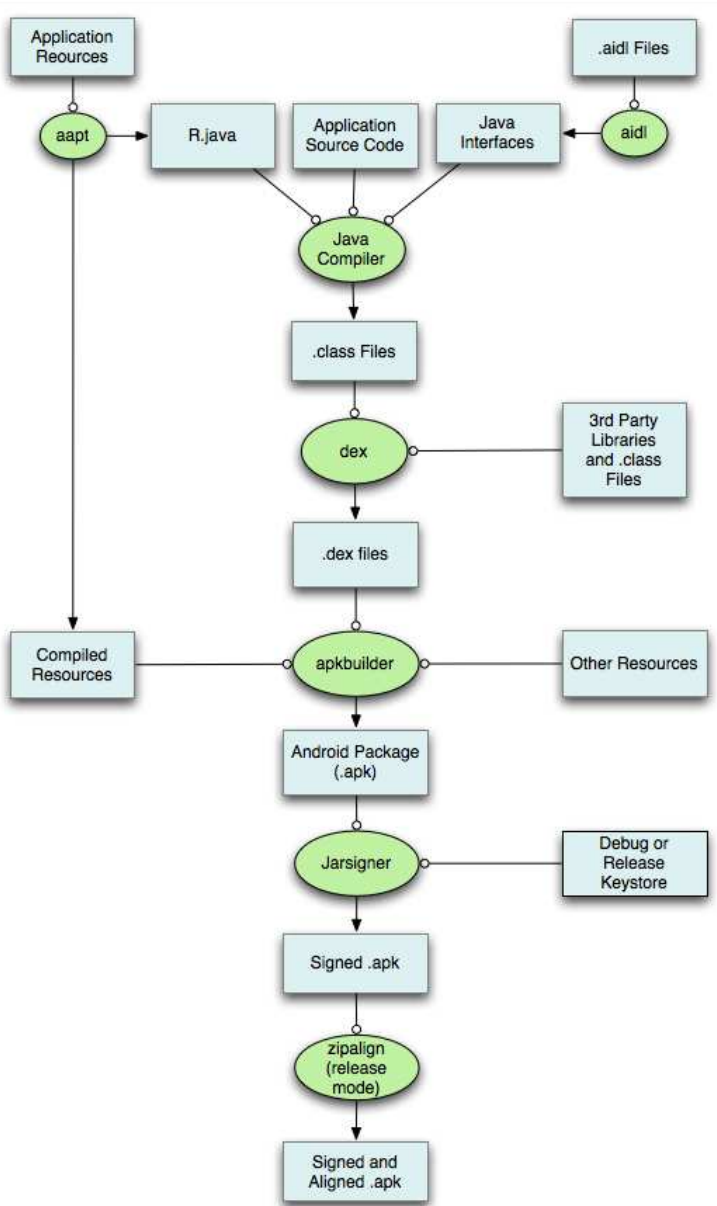


Configuración

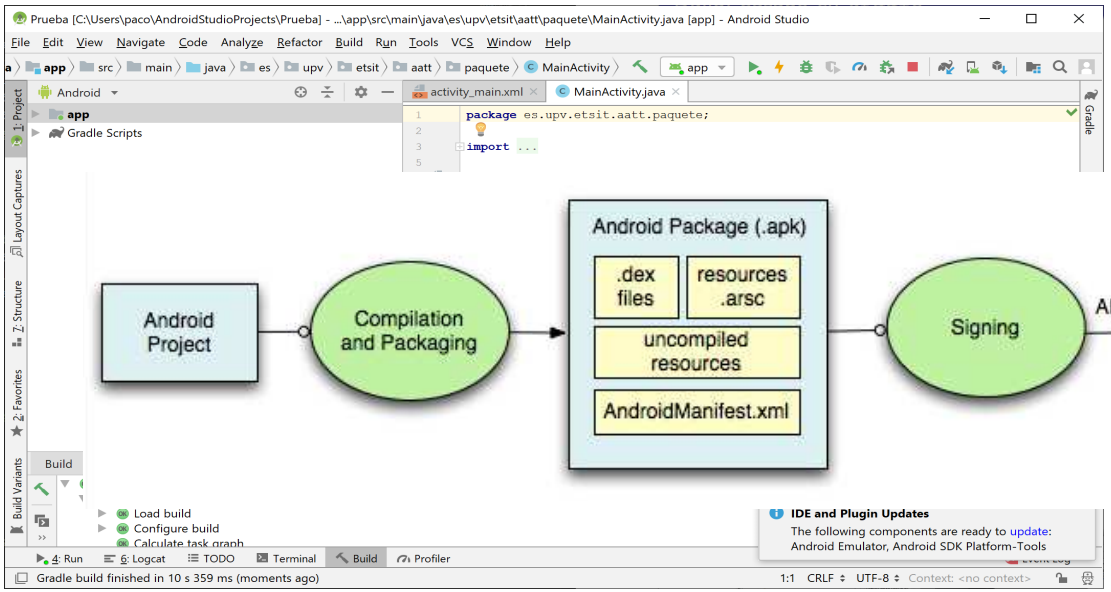
- Elección/configuración del *Virtual Device*

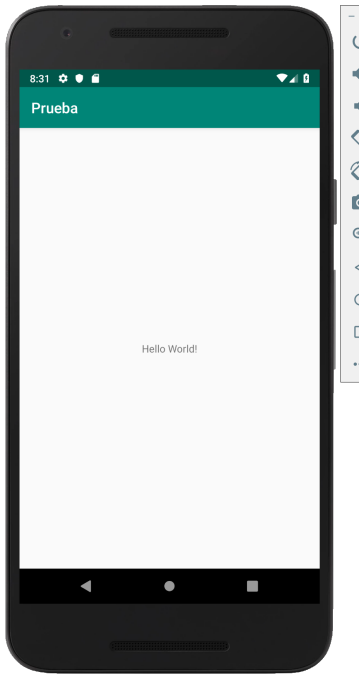


Build: basado en Gradle

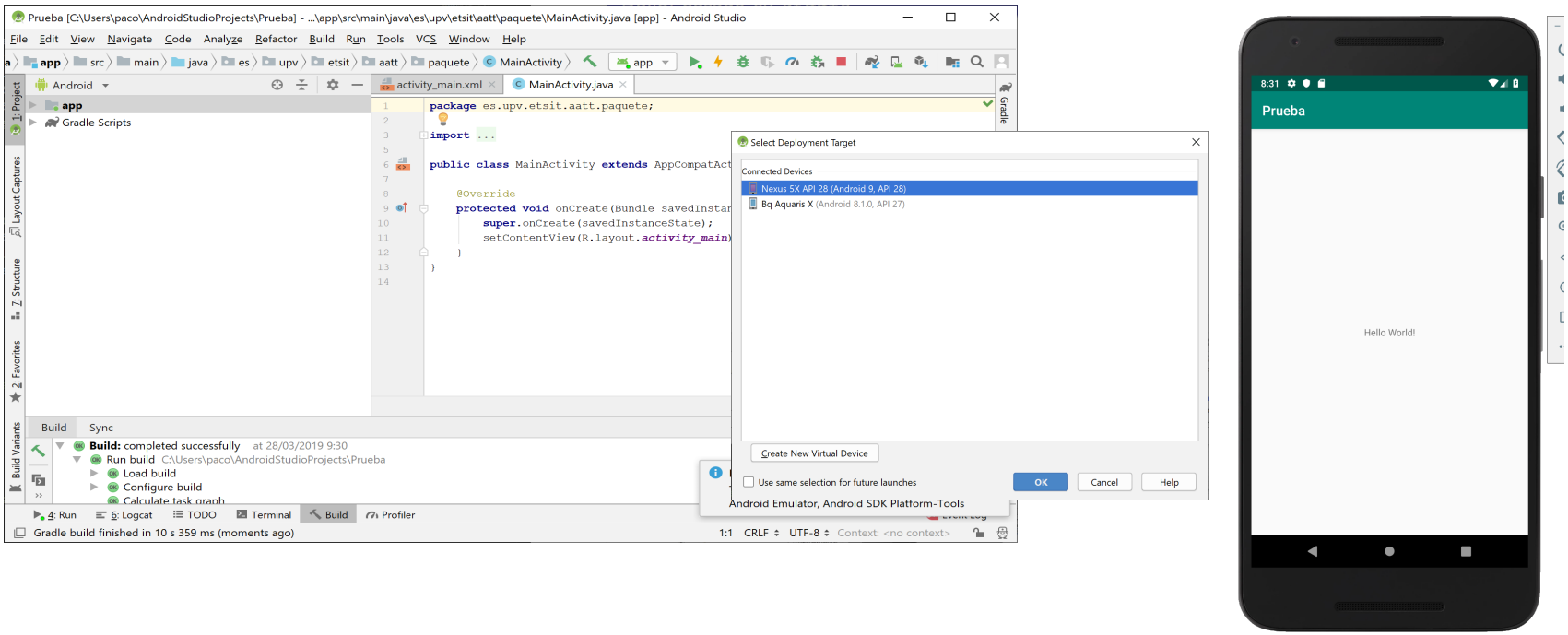


Ejecución: ADB (Android Debug Bridge)



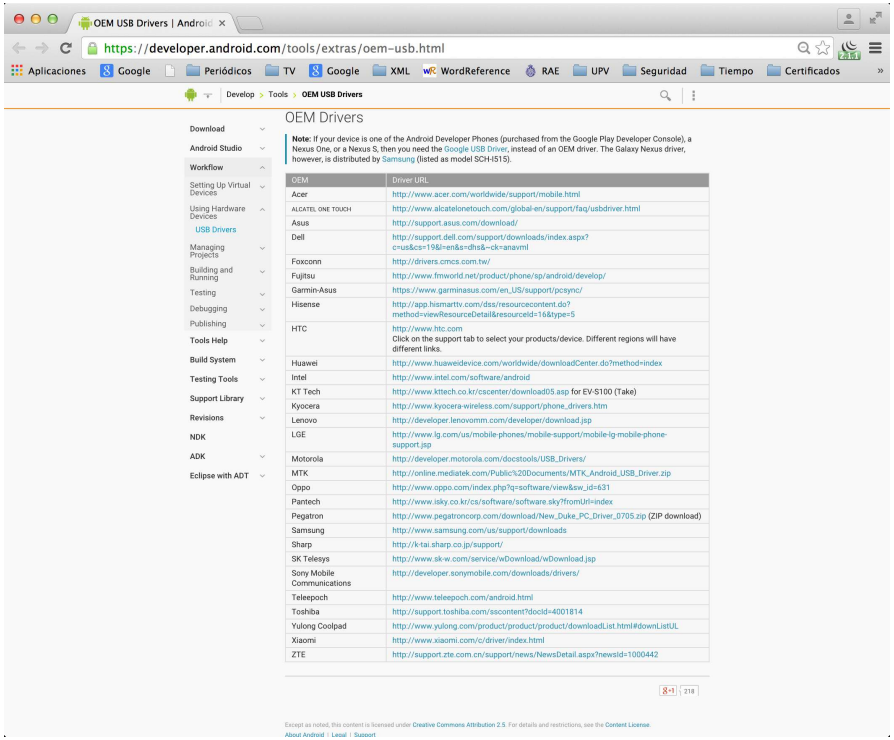


Ejecución en AVD: elección de dispositivo



Ejecución en dispositivo real

- Conexión de dispositivo Android a PC vía USB
- Instalación de driver
  `https://developer.android.com/tools/extras/oem-usb.html`



- El driver de USB de dispositivos Google ubicado en
  `<sdk>\extras\google\usb_driver\`.
  donde probablemente `<sdk>=C:\Users\paco\AppData\Local\Android\sdk`

Ejecución en dispositivo real

- Habilitar *USB debugging* en el dispositivo Android
  - $<$ Android 4.2: `Settings -> Applications -> Development`
  - $\geq$ Android 4.2:
    - `Settings -> About phone`, tocar `Build number` 7 veces
    - Aparecerá `Developer options` en pantalla anterior: habilitar *USB debugging*
- Elección de dispositivo (uno más)

Modos `debug` y `release`

	`debug`	`release`
Fase	Desarrollo	Final
Optimización de código	No	Sí
Generación información símbolos depuración	Sí	No
Tamaño relativo de código	Mayor	Menor
Velocidad relativa de ejecución	Menor	Mayor

Firma de las aplicaciones

- Cualquier aplicación debe ser firmada par que pueda instalarse en el emulador Android o en el terminal Android real.
- En modo `debug` se crea una clave asimétrica de depuración (no apta para publicar aplicación).
  - Proceso de firma es automático
- En modo `release` se debe firmar con clave privada (puede provenir de un certificado autofirmado)

Índice

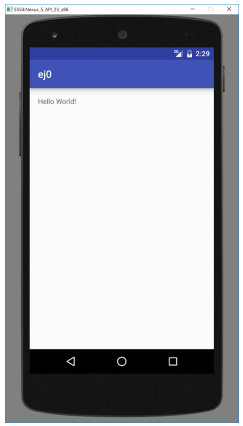
Aplicaciones y actividades

- Una aplicación Android puede constar o consistir en uno/a o más:
  - Activities *(Actividades)*
  - Content providers *(proveedores de contenido)*
  - Servicios *(services)*
  - Intents *(propósitos o intenciones)*

Primera actividad

- Empty Activity (extensión de: `AppCompatActivity`) (ALT+INTRO: importación automática)

```
package es.upv.etsit.aatt.ej0;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

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



- Segundo ejemplo. Extensión de `Activity` en vez de `AppCompatActivity`: cambio de aspecto

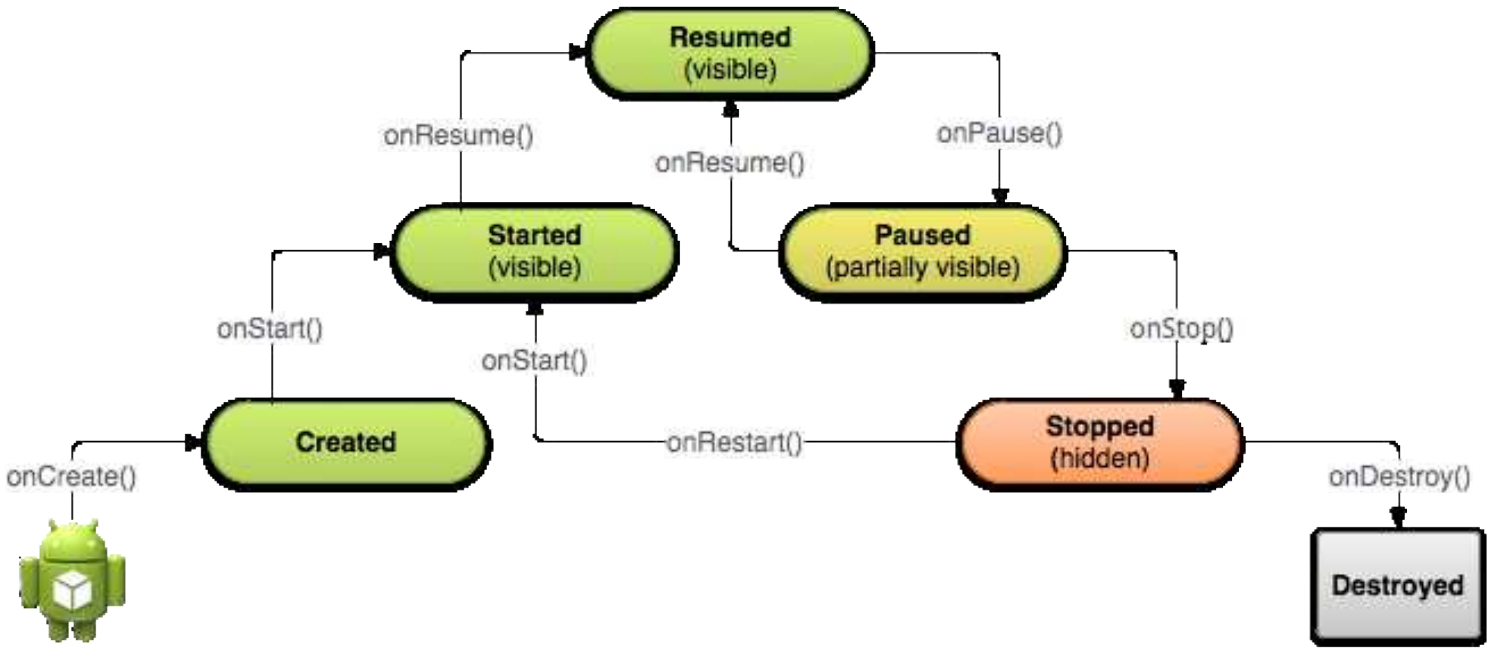
```
package es.upv.etsit.aatt.ej0;

import android.app.Activity;
import android.os.Bundle;

public class MainActivity extends Activity {

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

Estados de una actividad

- Reanudada/activa *(Resumed)*:
  - La actividad fue arrancada por el usuario
  - Está ejecutándose
  - Es visible en primer plano
  - El usuario puede interactuar
- Pausada *(Paused)*:
  - La actividad fue arrancada
  - Se está ejecutando
  - Es parcialmente visible (ej.: solapamiento por notificación de otra actividad).
  - El usuario no puede interactuar
- Parada *(Stopped)*:
  - La actividad fue arrancada
  - Se está ejecutando
  - Está oculta por otras actividades
  - No hay interacción con el usuario
  - Puede haber comunicación por notificaciones
- Muerta *(Destroyed)*:
  - La actividad nunca fue arrancada
  - O fue finalizada por falta de recursos
- Estados transitorios
  - Creada *(Created)*
  - Arrancada *(Started)*

Ciclo de vida de una actividad

- Gestionado enteramente por el sistema
- Métodos de *callback* donde el programador puede intervenir



Transiciones: métodos de *callback*

- `onCreate()`:
  - Arranque por primera vez (desde *system restart*)
  - Paro de la actividad
  - Cambio en los rescursos (de vertical a apaisado)
- `onDestroy()`:
  - Llamada a `finish()`
  - Necesidad de recursos (puede no llamarse si la necesidad es urgente)
- `onStart()`:
  - Paso previo a poder interactuar, desde bien su creación o su vuelta desde estado parado
- `onRestart()`:
  - Tras recuperación de paro
- `onStop()`:
  - Paso previo a estado parado
- `onResume()`:
  - Vuelta a visibilidad e interacción
  - Refresco de UI *(User Interface)*
- `onPause()`:
  - La actividad dejará de ser completamente visible e interactuable
  - Tras `onPause()` existe la posibilidad de que no se ejecute `onStop()` ni `onDestroy`

Test de estados

Ejemplo con mensajes de log:

- Clase a importar:

```
import android.util.Log;
```

- Ejemplo: `Log.d(<ETIQUETA>, <MENSAJE>);`

```
Log.d("TestClase", "En bucle. Iteracion" + i );
```

- Clases de log (orden por criticidad creciente):
  - `Log.v()`: `VERBOSE`
  - `Log.d()`: `DEBUG`
  - `Log.i()`: `INFO`
  - `Log.w()`: `WARN`
  - `Log.e()`: `ERROR`
- Situaciones:
  - `VERBOSE`: sólo debe estar presente en fase de desarrollo
  - `DEBUG`: eliminado automaticamente en tiempo de ejecución
  - `INFO`, `WARN`, `ERROR`: se mantienen siempre
- El clásico

```
System.out.println("Mensaje");
```

  funciona como

```
Log.i("system.out", "Mensaje");
```

- Logs en `Android Monitor (parte inferior) -> Logcat`

Test de estados (cont.)

https://github.com/AATT-ETSIT/U2T1-Ej1-Estados_actividad

```
package es.upv.etsit.aatt.ej2;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;

public class MainActivity extends AppCompatActivity {

    final String ETIQUETA = "ETIQUETA";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Log.d( ETIQUETA, "En onCreate()");
    }

    @Override
    protected void onStart( ) { Log.d( ETIQUETA, "En onStart()" ); }

    @Override
    protected void onRestart( ) { Log.d( ETIQUETA, "En onRestart()" ); }

    @Override
    protected void onResume( ) { Log.d( ETIQUETA, "En onResume()" ); }

    @Override
    protected void onPause( ) { Log.d( ETIQUETA, "En onPause()" ); }

    @Override
    protected void onStop( ) { Log.d( ETIQUETA, "En onStop()" ); }

    @Override
    protected void onDestroy( ) { Log.d( ETIQUETA, "En onDestroy()" ); }

    //... ¡¡¡CRASH!!!: super.onXxxx();

}
```

Ejemplo adicional de depuración...