



# **UD1 – Introducción a la programación móvil**

**2º CFGS  
Desarrollo de Aplicaciones Multiplataforma  
2022-23**

# 1.- Introducción

En **25 años** el mundo ha cambiado de una manera nunca imaginable.

Se podría decir que en la década de 1990 y principios de los 2000 el mundo se convirtió en **digital**.

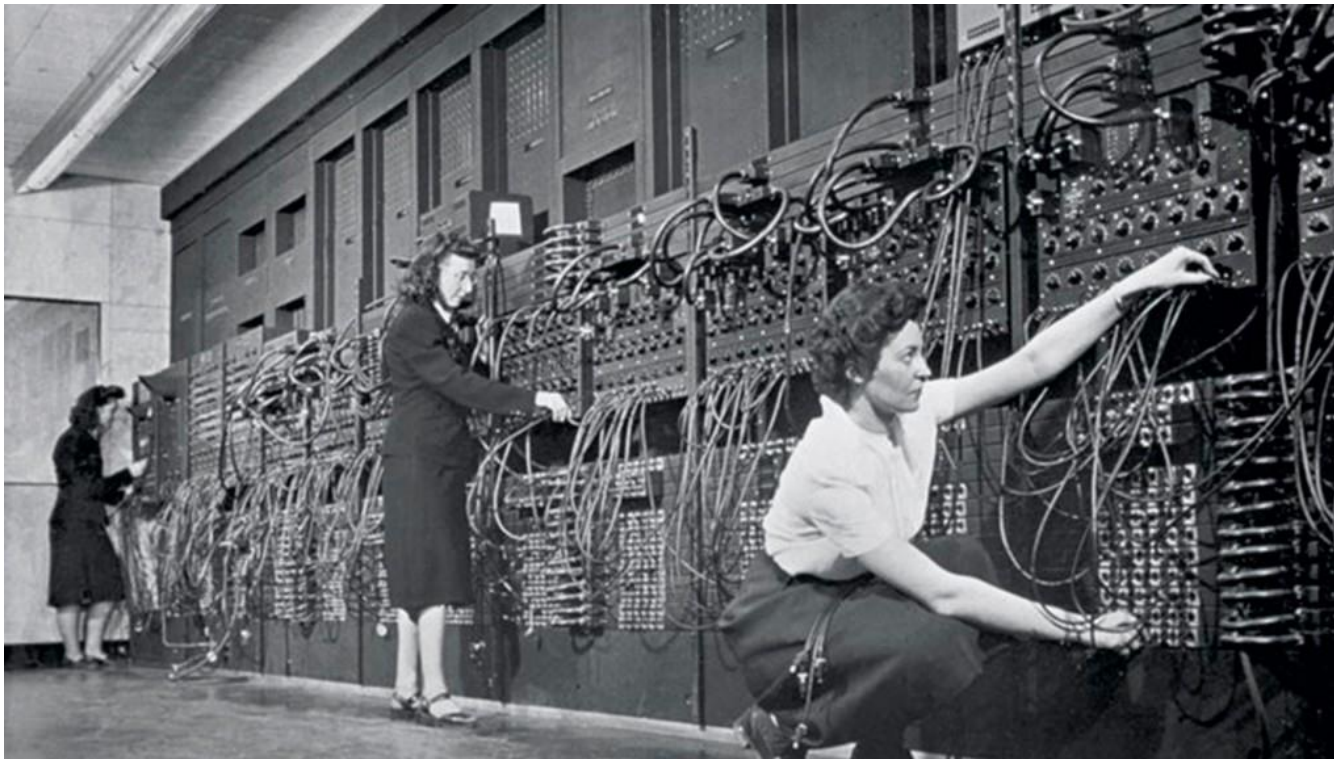
Hoy en día **el mundo es móvil**.

En España el móvil superó al PC como principal dispositivo de acceso a internet en 2017.

## 2.- Historia

Desde la invención de los ordenadores se ha intentado crear dispositivos más pequeños y manejables pero con **muchas más prestaciones**.

ENIAC (1946)



Portátil de 2022



## 2.- Historia

En el mundo de la **telefonía móvil** ha ocurrido exactamente lo mismo

Motorola DynaTAC 8000X (1984)



Móviles Nokia a lo largo del tiempo





## 2.- Historia

PDA PalmV  
1999



BlackBerry Pearl 8100  
2006



iPhone  
2007



## 2.- Historia

En **2007 Steve Jobs**, cofundador de **Apple**, presentó el **iPhone**.

Ya se habían presentado algunos smartphone antes que el iPhone.

Muchos de esos "smartphone" estaban orientados al uso de **oficina** como las PDA.

Pero iPhone fue una **revolución** en el mundo de la telefonía móvil.



## 2.- Historia

### ¿Por qué el iPhone fue una revolución?

- Pantalla táctil capacitiva – no necesita lápiz (resistiva)
- Precio – 500 €, menos de la mitad que una BlackBerry.
- Tamaño de pantalla – más de 3".
- Teclado completo (digital) – El del LG Prada era panel numérico.
- Sistema operativo diseñado en exclusiva.
- WiFi.
- iPod – reproductor de música.
- 4GB de RAM.
- Pantalla muy resistente.

## 2.- Historia

Como con cualquier sistema informático (PC, consolas...) en ese momento comenzó una **guerra** por ver quién **dominaba** el mercado de los sistemas operativos móviles.

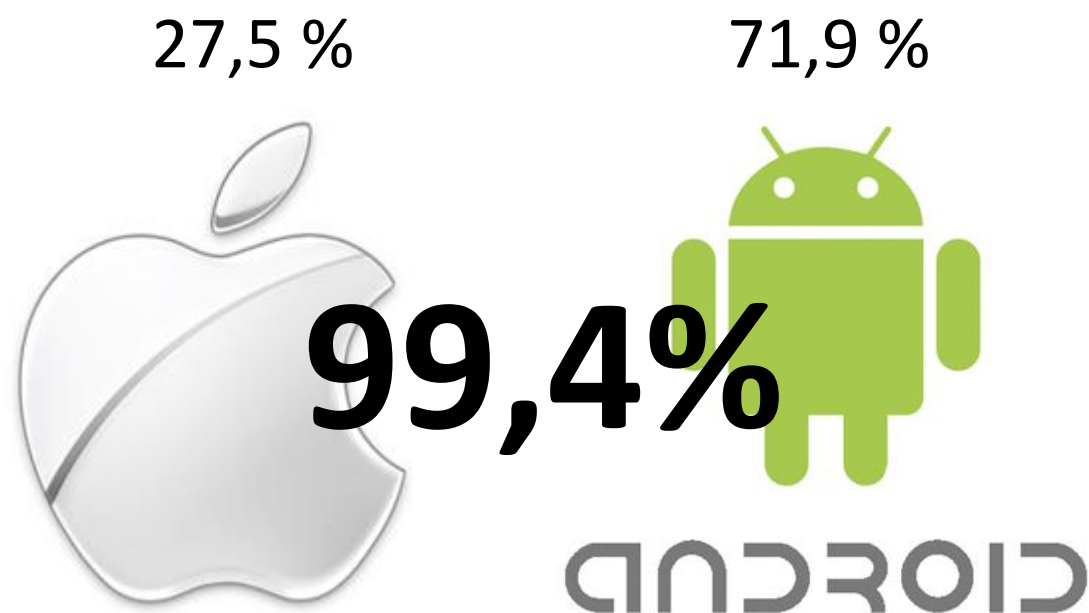
 **BlackBerry**





## 2.- Historia

Cuota de mercado mundial actual de SO en dispositivos móviles ([fuente](#)).



¿Quién ha ganado la batalla?

### 3.- Comparativa: iOS vs Android

iOS	Android
Ecosistema cerrado. La plataforma lo tiene todo bajo control.	Muy personalizable.
Más optimizado al tener el HW controlado.	Menos optimizado debido a las diferencias entre el HW de los dispositivos.
Poca fragmentación: La última versión se mantiene disponible durante mucho tiempo (iOS 15 está disponible hasta el iPhone 6s que es de 2015)	Mucha fragmentación: Las versiones de software disponibles para un dispositivo dependen de muchos factores como las características HW o el fabricante.
Como es un ecosistema cerrado no hay terceras partes que añadan capas al SO.	Los fabricantes suelen incorporar una última capa al SO.
Solo disponible en dispositivos Apple.	Cualquier fabricante puede comercializar un dispositivo con Android.
La App Store está muy supervisada de manera que es casi imposible instalar aplicaciones maliciosas.	Es muy fácil publicar en Google Play y también se pueden encontrar aplicaciones en internet (apk).
Se necesita un ordenador con macOS y el IDE Xcode para <b>publicar</b> aplicaciones en la App Store.	No se necesita HW ni SW específico para publicar aplicaciones en Google Play o directamente mediante su apk.

## 4.- Limitaciones

Existen unos factores determinantes a tener en cuenta cuando se desarrollan aplicaciones para dispositivos móviles:

- Desconexión.
- Seguridad.
- Memoria.
- Consumo de batería.
- Almacenamiento.

## 4.- Limitaciones

### Desconexión

Como dispositivo móvil en cualquier momento puede haber una desconexión ya sea por quedarse sin batería o perder la señal de datos.

Si la aplicación a desarrollar usa datos de un servidor debe tener en cuenta esto y ofrecer un mecanismo que en la medida de lo posible asegure que no se pierdan datos debido a esto.



## 4.- Limitaciones

### Seguridad

Al ser dispositivos pequeños son susceptibles de ser sustraídos.

En ocasiones se pueden conectar a redes poco seguras (free WiFi).

También existe la posibilidad de que se hayan instalado aplicaciones maliciosas que tengan acceso a los sensores del dispositivo.



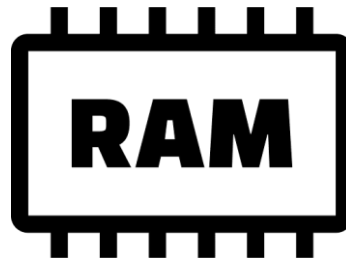


## 4.- Limitaciones

### Memoria

Una de las principales desventajas de los dispositivos móviles es la dificultad e incluso imposibilidad de cambiar el hardware.

Aunque se ha ampliado la cantidad de RAM con los años aún sigue siendo inferior a un PC.



## 4.- Limitaciones

### Consumo de batería

La batería de los dispositivos móviles es uno de los recursos más preciados.

Aunque la capacidad de las baterías ha aumentado también lo ha hecho el tamaño de las pantallas (componente que generalmente más batería usa).

En el desarrollo de aplicaciones móviles se debe tener en cuenta la liberación de recursos para así ahorrar el consumo de batería.



## 4.- Limitaciones

### Almacenamiento

Aunque muchos dispositivos permiten incorporar una tarjeta microSD, habitualmente este almacenamiento solo puede utilizarse para almacenar multimedia (fotos, vídeos, audios).

Así en el almacenamiento se suele estar las aplicaciones y es un recurso muy preciado y las aplicaciones deberían controlar su tamaño para dejar espacio a otras aplicaciones.



---

## 5.- Tipos de aplicaciones

### **Aplicaciones nativas (native app):**

Para su desarrollo se usan las herramientas que proporciona el fabricante.

### **Aplicaciones multiplataforma (cross-platform app, hybrid app)**

Para desarrollar estas aplicaciones se usan herramientas multipropósito.

## 5.- Tipos de aplicaciones

Igual que ocurre en otros mercados de aplicaciones, el de dispositivos móviles es muy diverso con **novedades cada mes**.

Los desarrollos se enfrentan al desafío de **elegir para qué plataforma desarrollar** sus aplicaciones.

No existe una fórmula mágica ni algoritmo que indique qué plataforma usar.

Como desarrollador de aplicaciones se deben conocer las diferencias para poder decidir qué tipo de aplicación se va a desarrollar.



## 5.- Tipos de aplicaciones

### Aplicaciones nativas – ventajas

- Se aprovecha todo el potencial del HW.  
Al usar la plataforma que facilita el fabricante los mecanismos de acceso a las funcionalidades y sensores están más optimizadas.
- Mayor optimización en las pantallas.  
La plataforma nativa ofrece más libertad a la hora de optimizar las pantallas, factor importante dada la gran cantidad de tamaños y relación de aspecto de las mismas.
- Mayor rendimiento.  
Al estar desarrolladas en expreso para una plataforma estas aplicaciones son muy rápidas y funcionan mejor.

## 5.- Tipos de aplicaciones

### Aplicaciones nativas – desventajas

- Incompatibilidad entre plataformas.

Una aplicación nativa está desarrollada expresamente para un sistema operativo por lo que esa aplicación no podrá ser usada en una plataforma diferente.

Se tendrá que desarrollar la aplicación tantas veces como plataformas donde se quiera ejecutar la aplicación.

- Incompatibilidad de versiones.

Las plataformas que facilitan los fabricantes intentan que las aplicaciones siempre se desarrollen para las últimas versiones por lo que desarrollar una aplicación con soporte para todas las versiones supone a veces un desafío.

## 5.- Tipos de aplicaciones

### Aplicaciones multiplataforma – ventajas

- Reutilización de código.  
Al usar una plataforma genérica la aplicación solo se desarrolla una vez (varias compilaciones).
- Compatibilidad de plataformas.  
Las plataformas de desarrollo incorporan mecanismos para no tener que conocer los detalles de cada plataforma a la hora de desarrollar funcionalidades.
- Curva de aprendizaje corta.  
No es necesario conocer los entresijos de cada plataforma al haber un único desarrollo.
- Rentabilidad.  
No se requiere ni SW ni HW específico por lo que no supondrá una inversión grande.
- Misma Interfaz y experiencia de usuario.  
En todas las plataformas la interfaz de la aplicación será la misma.

## 5.- Tipos de aplicaciones

### Aplicaciones multiplataforma – desventajas

- Peor rendimiento.  
Las plataformas de desarrollo deben aplicar capas extra para poder convertir las funcionalidades desarrolladas a cada plataforma, esto empeora el rendimiento.
- Problemas de compatibilidad  
Una misma funcionalidad puede no funcionar correctamente en diferentes plataformas.
- Menos flexibles.  
Al usar procedimientos genéricos en ocasiones no se permite un desarrollo concreto.
- Publicación costosa.  
A la hora de publicar una aplicación multiplataforma en los canales oficiales puede haber complicaciones.

## 5.- Tipos de aplicaciones

### ■ Aplicaciones nativas:

Plataforma destino	IDE	Lenguaje
Android	Android Studio	Java, Kotlin
Android	App Inventor	Lenguaje de bloques
Apple	XCode	Objective-C, Swift

### ■ Aplicaciones multiplataforma:

Compañía	Plataforma destino	IDE	Lenguaje
Microsoft	Android, iOS y Windows	Xamarin	C#
Unity Technologies	Escritorio, móviles, TV	Unity	C#
Google	Android, iOS, escritorio, web	Flutter	Dart
Open Source	Android, iOS, Windows, web	Ionic	JavaScript
Facebook	Android, iOS, escritorio, TV	React Native	JavaScript
Apache Cordova	Android, iOS	Apache Cordova	HTML5, CSS3 y JS



# Hoja de ruta

1ª Evaluación:

Desarrollo de aplicaciones nativas → Android

2ª Evaluación:

Desarrollo de aplicaciones multiplataforma → Unity

---

## 6.- Android

Sistema operativo **móvil**.

Desarrollo inicial por Android Inc. en **2003**.

Adquirida por **Google** en 2005 por 5 millones de dólares.



## 7.- Características de Android

### ■ **Plataforma abierta**

Android es una plataforma libre basada en Linux y de código abierto.

Se puede utilizar, modificar y adaptar sin pagar derechos de autor (royalties).

### ■ **Portabilidad garantizada**

Las aplicaciones Android se ejecutan sobre una especie de Máquina Virtual de Java lo cual ofrece un gran nivel de compatibilidad entre dispositivos actuales y futuros.

### ■ **Arquitectura basada en componentes inspirados en Internet**

Por ejemplo la interfaz de usuario se realiza con XML, esto permite el reescalado de la aplicación según el tamaño de la pantalla

## 7.- Características de Android

- **Filosofía dispositivo siempre conectado a internet**

- **Servicios integrados**

Android incorpora una gran cantidad de servicios integrados en la plataforma como:

Localización GPS

Bases de datos SQL

Reconocimiento y síntesis de voz

Navegador

...

- **Nivel de seguridad aceptable**

La máquina virtual donde se ejecutan las aplicaciones utiliza el concepto "ejecución dentro de una caja" lo cual aísla las aplicaciones unas de otras.

## 7.- Características de Android

- **Optimizado para bajo consumo y poca memoria**

La Máquina Virtual que utiliza Android está optimizada para la ejecución en dispositivos móviles.

- **Gráficos y sonido de alta calidad**

Android ofrece una gran cantidad de mecanismos para garantizar gráficos y sonidos de alta calidad:

- Gráficos vectoriales con anti-aliasing (suavizado)

- Gráficos 3D basados en OpenGL

- Animaciones inspiradas en Flash

- Códecs de vídeo y audio

- ...



## 8.- Arquitectura de Android

En un principio Android usaba la Máquina Virtual **Dalvik** (DVM) que es muy similar a la Máquina Virtual Java.

Las aplicaciones Android tienen el formato **.apk**.

Un archivo **apk** contiene clases Java compiladas en "DEX bytecode" a diferencia de Java que usa "Java bytecode".

Dalvik compila en **tiempo de ejecución** (just-in-time JIT) el DEX bytecode al lenguaje máquina del dispositivo donde se ejecuta la aplicación.

## 8.- Arquitectura de Android

A partir de Android KitKat 4.4 se utiliza **Android Runtime (ART)**.

**ART** usa la **compilación anticipada** (ahead-of-time AOT).

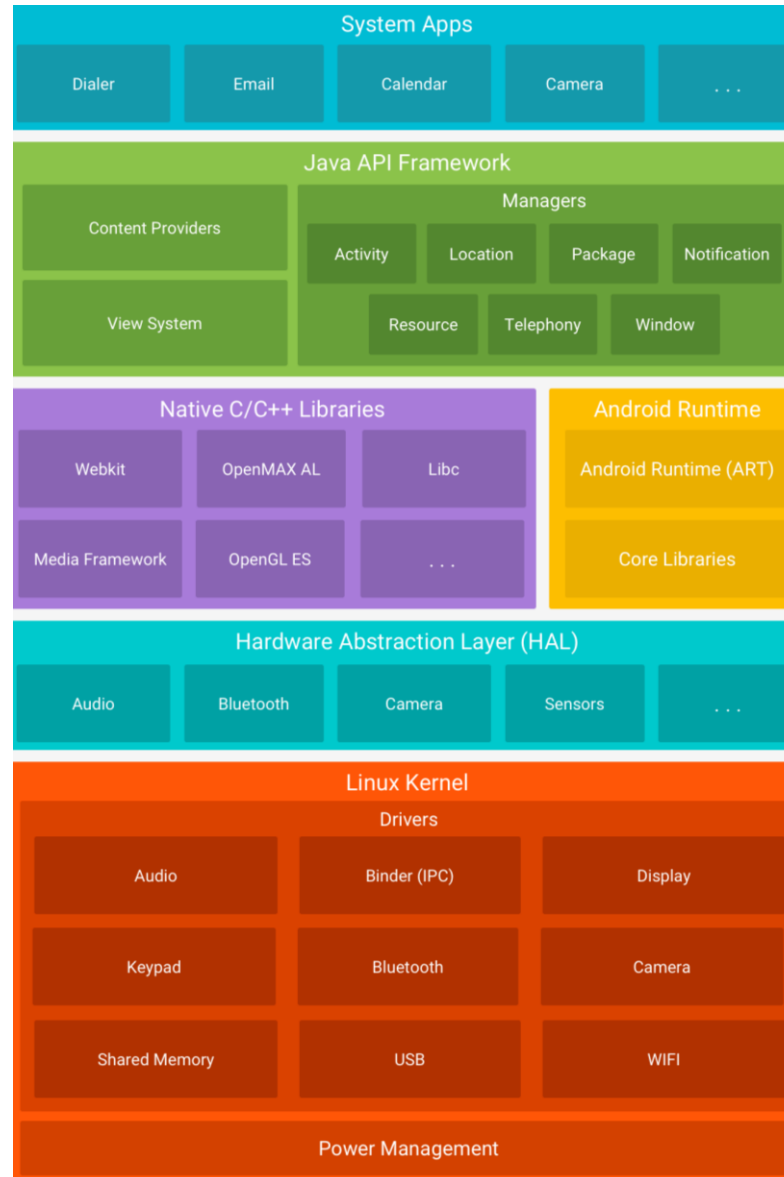
Mediante esta tecnología Android compila el DEX bytecode **durante la instalación de la aplicación**.

De esta manera aunque la instalación de aplicaciones necesita más tiempo y recursos, la posterior ejecución de aplicaciones es mucho más rápida.

ART también ofrece mejoras en el recolector de basura y en el desarrollo y depuración de aplicaciones.

# 8.- Arquitectura de Android

[Web oficial](#)



## 8.- Arquitectura de Android

Dada la base de Android basada en Java, gracias a la ejecución en máquina virtual es muy sencillo desarrollar aplicaciones para cualquier tipo de dispositivo.

Hoy en día Se puede encontrar Android en muchos tipos de dispositivos:

- SmartPhone
- Tablets
- SmartWatch
- TV
- Automóviles
- ...
- En PC gracias al proyecto Android-x86

## 8.- Arquitectura de Android

Este es uno de los principales problemas del desarrollador de Android.

- Tamaños de pantalla
- Densidad de píxeles
- Relaciones de aspecto
- Orientaciones de pantalla
- Cantidad de memoria RAM
- Cantidad de almacenamiento
- Sensores

A la hora de desarrollar una aplicación Android se deberá elegir en qué tipo de dispositivo se quiere ejecutar y para cuántas pantallas se van a diseñar.

## 9.- Fragmentación en Android

Como en cualquier sistema operativo (SO), Android evoluciona añadiendo mejoras y funcionalidades que se implementan en las diferentes versiones que se crean.

Se conoce como **fragmentación** de SO al hecho de que en un momento dado existen instaladas diferentes versiones del SO en los diferentes dispositivos que se usan.

La fragmentación en Android depende principalmente de dos factores:

- **Predisposición del fabricante** – ¿Obsolescencia programada?
- **HW del dispositivo**

Al desarrollar aplicaciones Android se debe tener en cuenta tanto la versión del SO como el **nivel de API**. (Existen bibliotecas de compatibilidad con versiones antiguas).

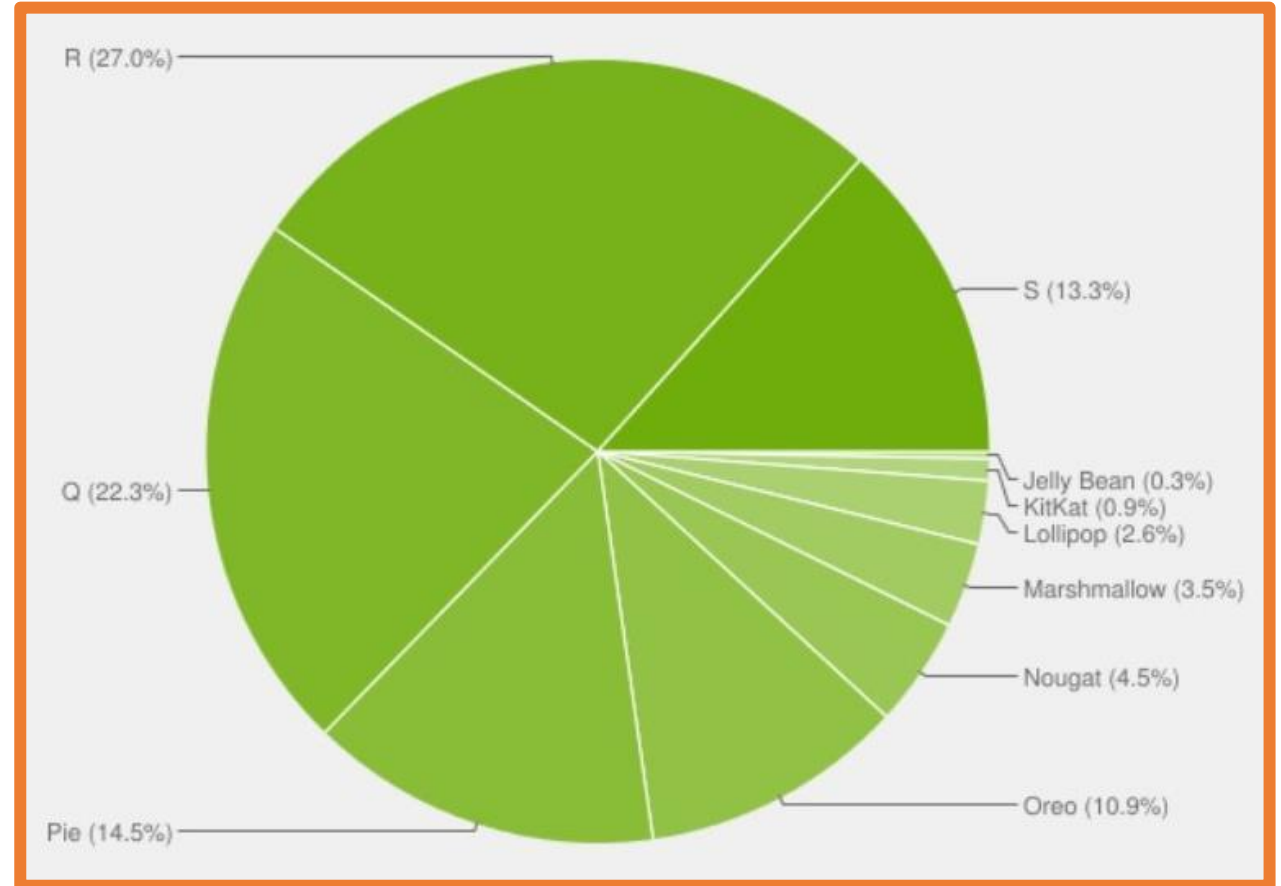
## 9.- Fragmentación en Android

Versión		Número de versión	Fecha de lanzamiento	Nivel de API	
<b>A</b>	Apple Pie	1.0	23 de septiembre de 2008	1	Antigua sin soporte
<b>B</b>	Banana Bread	1.1	9 de febrero de 2009	2	
<b>C</b>	Cupcake	1.5	25 de abril de 2009	3	
<b>D</b>	Donut	1.6	15 de septiembre de 2009	4	
<b>E</b>	Eclair	2.0 – 2.1	26 de octubre de 2009	5 – 7	
<b>F</b>	Froyo	2.2 – 2.2.3	20 de mayo de 2010	8	
<b>G</b>	Gingerbread	2.3 – 2.3.7	6 de diciembre de 2010	9 – 10	
<b>H</b>	Honeycomb	3.0 – 3.2.6	22 de febrero de 2011	11 – 13	
<b>I</b>	Ice Cream Sandwich	4.0 – 4.0.5	18 de octubre de 2011	14 – 15	
<b>J</b>	Jelly Bean	4.1 – 4.3.1	9 de julio de 2012	16 – 18	
<b>K</b>	KitKat	4.4 – 4.4.4	31 de octubre de 2013	19 – 20	
<b>L</b>	Lollipop	5.0 – 5.1.1	12 de noviembre de 2014	21 – 22	
<b>M</b>	Marshmallow	6.0 – 6.0.1	5 de octubre de 2015	23	
<b>N</b>	Nougat	7.0 – 7.1.2	15 de junio de 2016	24 – 25	
<b>O</b>	Oreo	8.0 – 8.1	21 de agosto de 2017	26 – 27	
<b>P</b>	Pie	9.0	6 de agosto de 2018	28	
<b>Q</b>	10 (Quince Tart)	10.0	3 de septiembre de 2019	29	Antigua con soporte
<b>R</b>	11 (Red Velvet Cake)	11.0	8 de septiembre de 2020	30	
<b>S</b>	12 (Snow Cone)	12.0 - 12L	4 de octubre de 2021	31 - 32	Vigente
<b>T</b>	13 (Tiramisú)	13.0	15 de agosto de 2022	33	Novedad

## 9.- Fragmentación en Android

Fragmentación agosto 2022

Versión		Distribución
J	Jelly Bean	0,30%
K	KitKat	0,90%
L	Lollipop	2,60%
M	Marshmallow	3,50%
N	Nougat	4,50%
O	Oreo	10,90%
P	Pie	14,50%
Q	10 (Quince Tart)	22,30%
R	11 (Red Velvet Cake)	27,00%
S	12 (Snow Cone)	13,30%
T	13 (Tiramisú)	---





## 10.- Integrated Development Environment (IDE)

Para desarrollar aplicaciones nativa en Android existen tres opciones:

- **App Inventor**

Entorno de desarrollo basado en un **lenguaje de bloques** similar a Scratch.  
Ofrece un nivel de abstracción muy alto.

- **Eclipse**

Primer IDE oficial para Android.

- **Android Studio**

IDE ofrecido por **Google** que reemplaza a Eclipse.  
Permite elegir entre los lenguajes **Java** y [Kotlin](#).  
Acceso total a los recursos del dispositivo.

## 10.- Integrated Development Environment (IDE)

Para la marcha del curso se necesitará principalmente:

- IDE Android Studio que incluye SDK de Android con API y bibliotecas Java.
- Driver USB del dispositivo donde se realizarán las pruebas.
- Emulador Android externo.

# 11.- Android Studio

## Windows

- 64-bit Microsoft® Windows® 8/10
- x86\_64 CPU architecture; 2nd generation Intel Core or newer, or AMD CPU with support for a Windows Hypervisor
- 8 GB RAM or more
- 8 GB of available disk space minimum (IDE + Android SDK + Android Emulator)
- 1280 x 800 minimum screen resolution

## Mac

- MacOS® 10.14 (Mojave) or higher
- ARM-based chips, or 2nd generation Intel Core or newer with support for Hypervisor.Framework
- 8 GB RAM or more
- 8 GB of available disk space minimum (IDE + Android SDK + Android Emulator)
- 1280 x 800 minimum screen resolution

## Linux

- Any 64-bit Linux distribution that supports Gnome, KDE, or Unity DE; GNU C Library (glibc) 2.31 or later.
- x86\_64 CPU architecture; 2nd generation Intel Core or newer, or AMD processor with support for AMD Virtualization (AMD-V) and SSSE3
- 8 GB RAM or more
- 8 GB of available disk space minimum (IDE + Android SDK + Android Emulator)
- 1280 x 800 minimum screen resolution

## Chrome OS

For information on recommended devices and specifications, as well as Android Emulator support, visit [chromeos.dev](https://chromeos.dev).

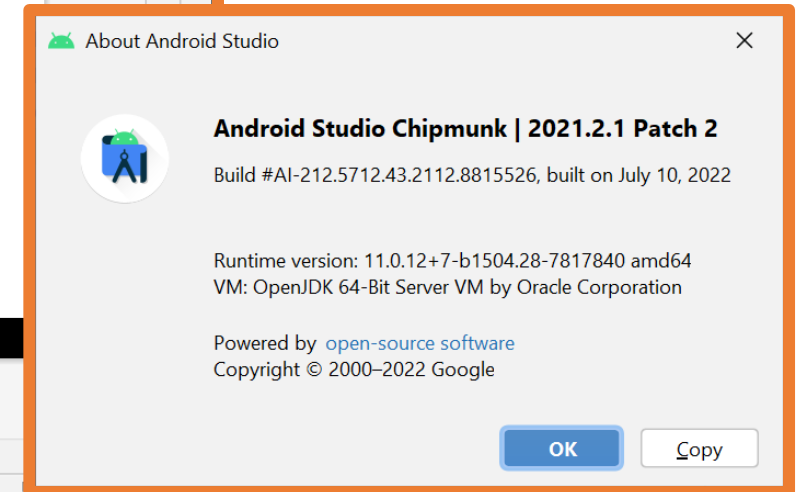
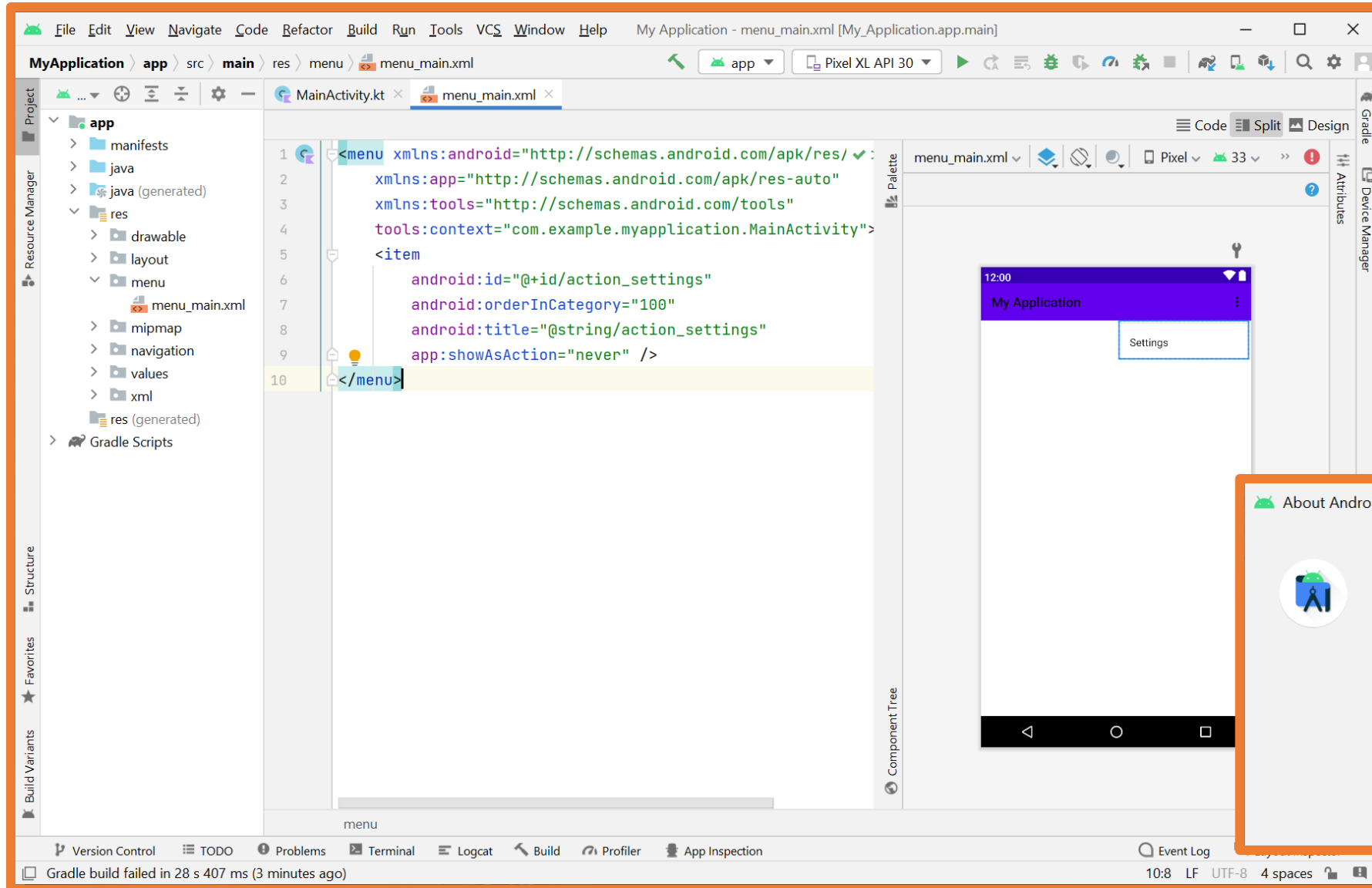
## 11.- Android Studio

Se puede usar cualquier sistema operativo para desarrollar para Android.

Windows es el que menos problemas tiene al reconocer el dispositivo móvil real a través de USB con el driver ADB (Android Device Bridge) lo cual es indispensable para la realización de simulaciones reales.

Si las simulaciones en dispositivo real fallan se podrá usar un emulador.

# 11.- Android Studio



# Práctica

## **Actividad 1:**

Preparando el entorno de programación y pruebas

## **Actividad 2:**

Primera app: Hello there

## 12.- Emulador Android

Android Studio dispone de un emulador incorporado donde poder ejecutar las aplicaciones que se están desarrollando.

El emulador permite crear dispositivos virtuales **AVD** (Android Virtual Device) con las características que se necesiten:

Hardware: memoria

tamaño de pantalla

resolución de pantalla

densidad de píxeles...

Software: versión de Android y/o API

También permite elegir de una lista de dispositivos ya creados e importar nuevos.

# Práctica

## **Actividad 3:**

Usando el emulador



## 13.- Probando aplicaciones en dispositivos reales

El emulador de Android Studio es una buena herramienta para poder probar las aplicaciones que se están desarrollando.

Una mejor opción es **probar las aplicaciones en dispositivos reales**.

Incluso es recomendable probar una misma aplicación en dispositivos con diferentes características:

- Versión de Android
- Tamaño de pantalla
- Cantidad de RAM
- ...

Así, se recomienda probar las aplicaciones tanto en el emulador con diferentes dispositivos virtuales como en diferentes dispositivos reales.

## 13.- Probando aplicaciones en dispositivos reales

Para poder probar las aplicaciones en dispositivos reales es necesario instalar los drivers del dispositivo.

Se pueden usar los drivers:

- del fabricante del dispositivo.
- genéricos de Google.
- universales ADB (Android Debug Bridge).

Es importante configurar el **modo desarrollador** en los dispositivos reales para poder usar el modo depurador en ellos desde Android Studio.



# Práctica

## **Actividad 4:**

Probando en dispositivos reales

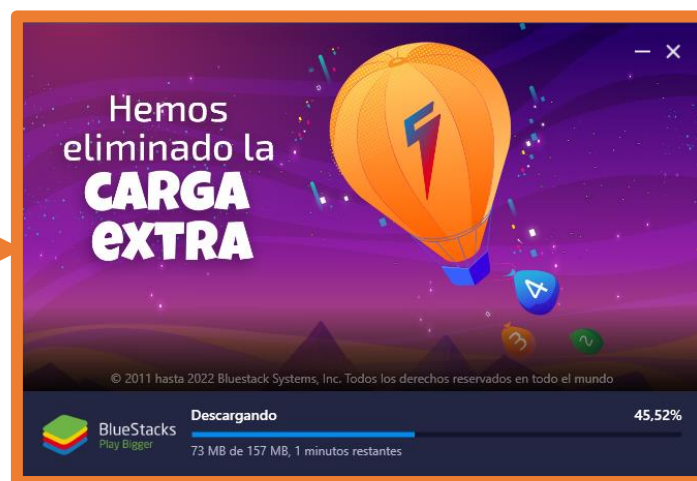
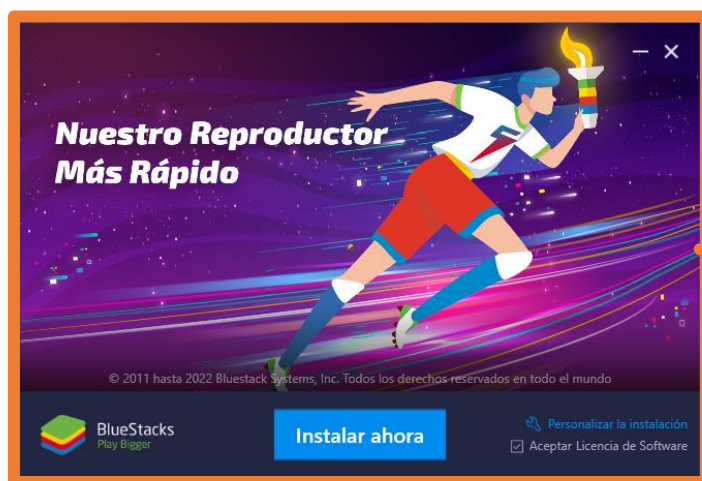
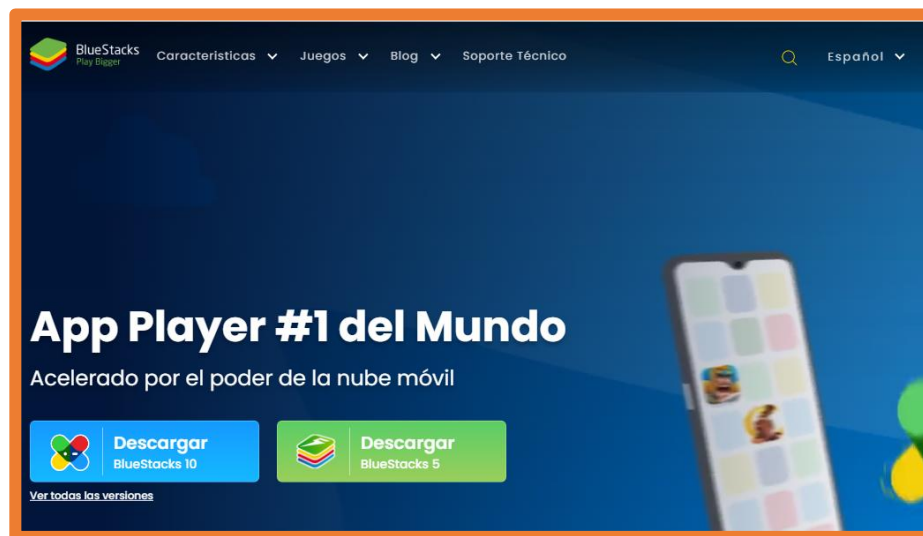
## 14.- Emuladores de terceros

Además del emulador de Android Studio, existen emuladores de otras empresas.

En ocasiones esos emuladores se crearon para poder ejecutar aplicaciones de Android, principalmente juegos, en sistemas operativos de escritorio.

Este es el caso de BlueStacks: <https://www.bluestacks.com/es/index.html>

# 14.- Emuladores de terceros

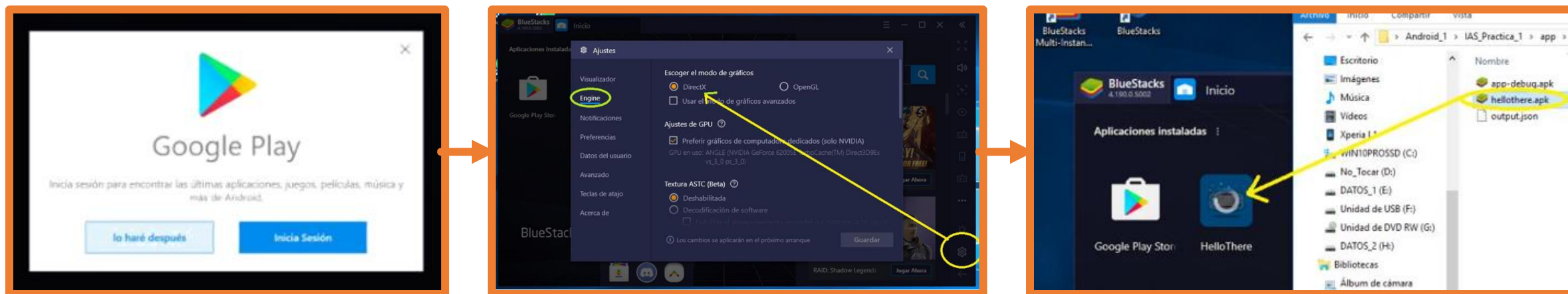


## 14.- Emuladores de terceros

No es necesario iniciar sesión en Google Play.

Si hay problemas con la ejecución se deberá cambiar a DirectX como modo de gráficos.

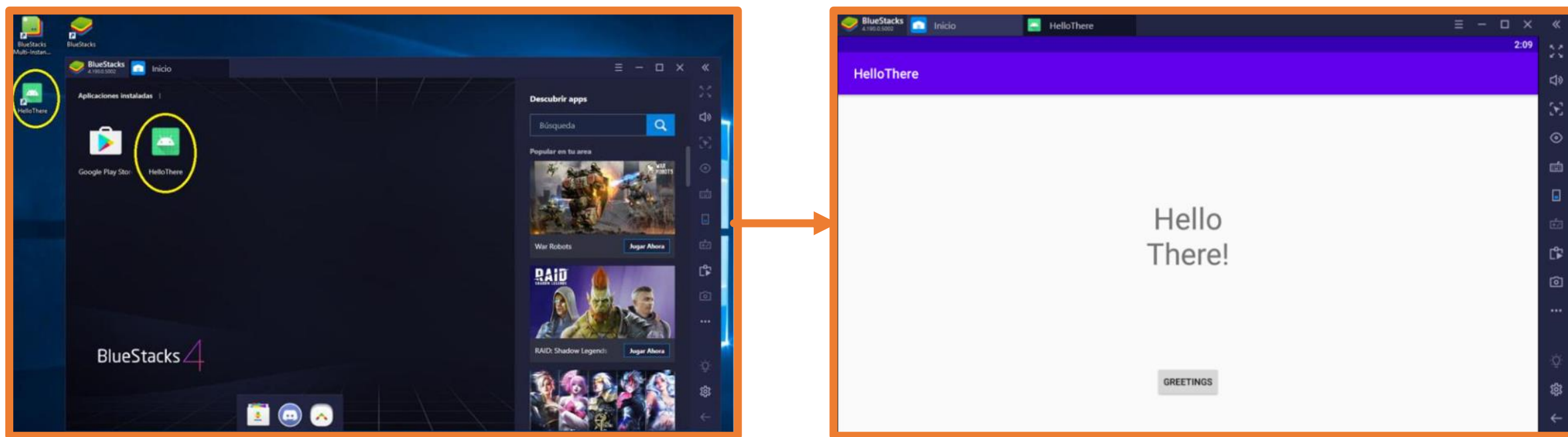
Una vez cargado BlueStacks se debe arrastrar el archivo .apk a la ventana principal del programa.



## 14.- Emuladores de terceros

Al arrastrar el archivo .apk a BlueStacks también se creará un acceso directo en el escritorio.

En este punto ya se puede ejecutar las aplicación en el emulador.



## 14.- Emuladores de terceros

Otro emulador muy conocido es [Genymotion](#).

Los desarrolladores indican que es mucho más rápido y eficiente que el emulador oficial de Android Studio.

En el siguiente enlace puedes ver un manual de [instalación y uso](#).



## 15.- Generación de archivos .apk

Una vez desarrollada una aplicación se debe generar el archivo **.apk** que será el que habrá que distribuir cargándolo en Play Store o por internet.

Existen dos tipos de apk:

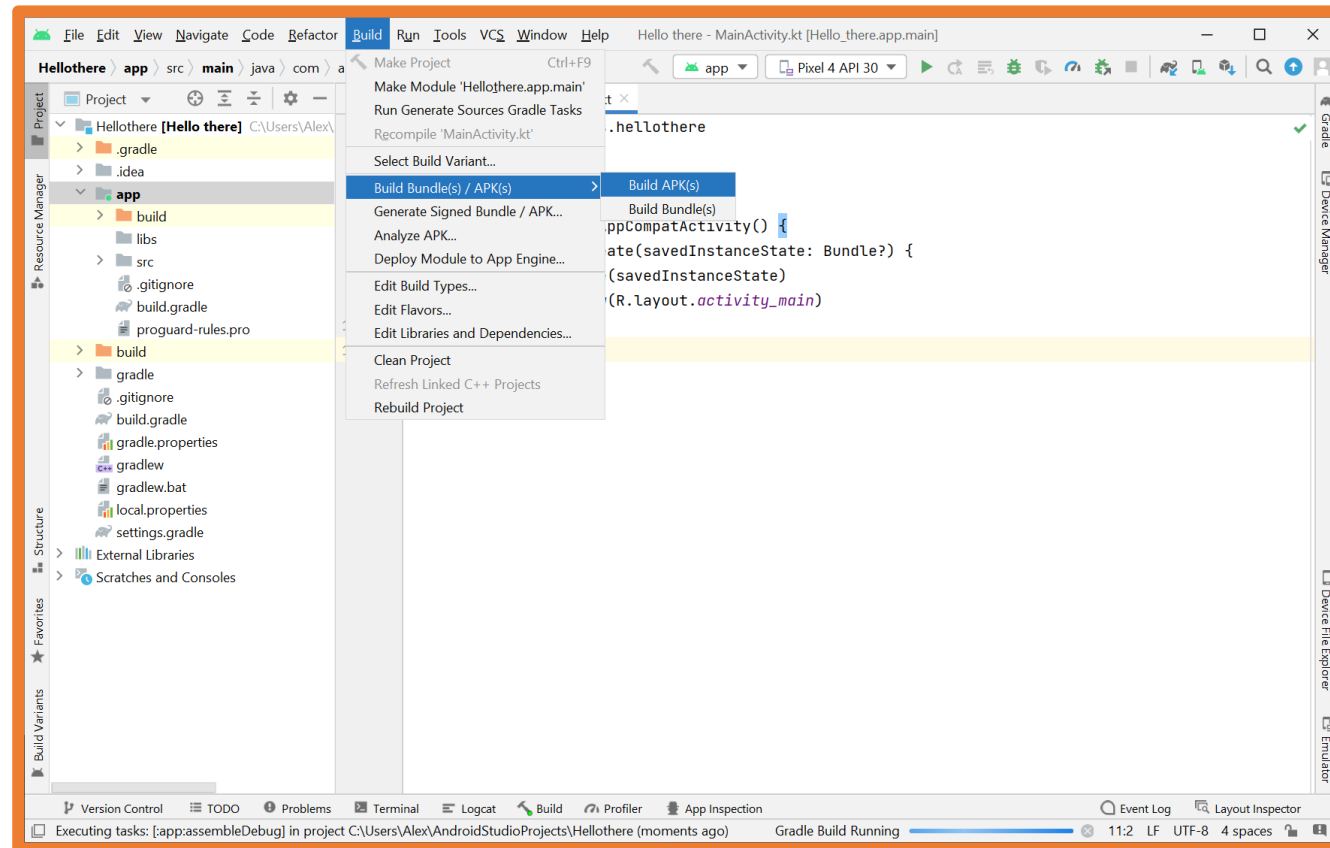
- **APK de depuración:** su uso es para pruebas, no se pueden poner a disposición del público.
- **APK firmado:** aplicación ya probada y depurada que se carga en Play Store para su distribución.

# 15.- Generación de archivos .apk

## APK de depuración

Generar estos archivos es muy sencillo, con el proyecto abierto:

Build → Build Bundle(s) / APK(s) → Build APK(s)



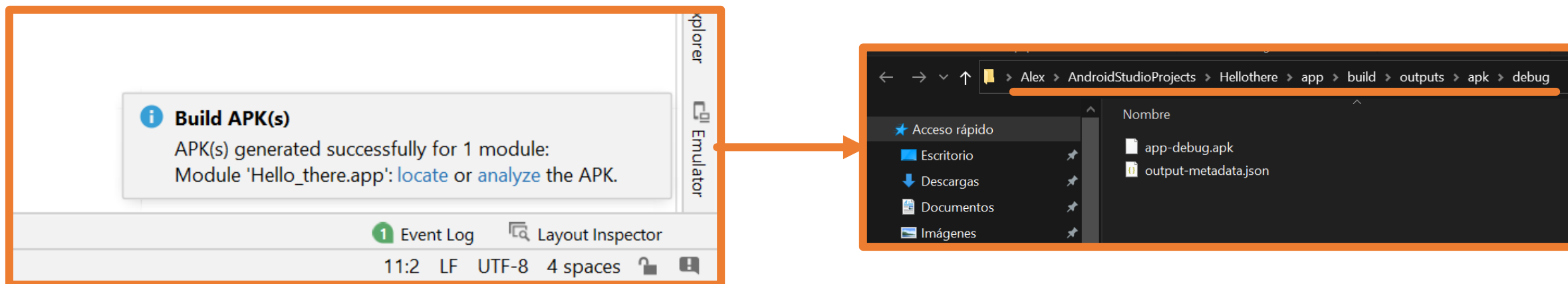
## 15.- Generación de archivos .apk

### APK de depuración

Una vez finalizada la creación del APK se muestra una notificación en la esquina inferior derecha de Android Studio.

Al hacer clic en **locate** se abrirá el directorio con el APK de depuración.

Si pierdes la notificación puedes buscar el APK en el directorio del proyecto.

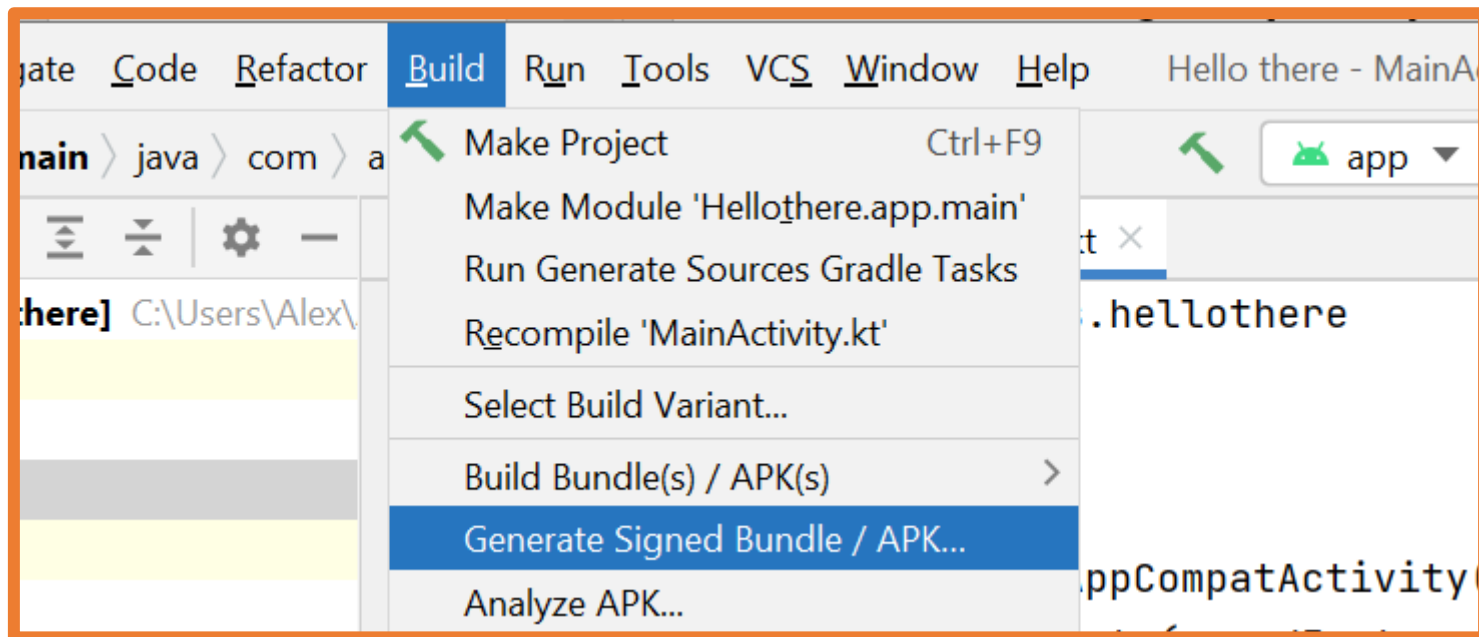


## 15.- Generación de archivos .apk

### APK firmado

Para generar un APK firmado también es sencillo aunque la primera vez hay que realizar algunos pasos como la creación de un almacén de claves.

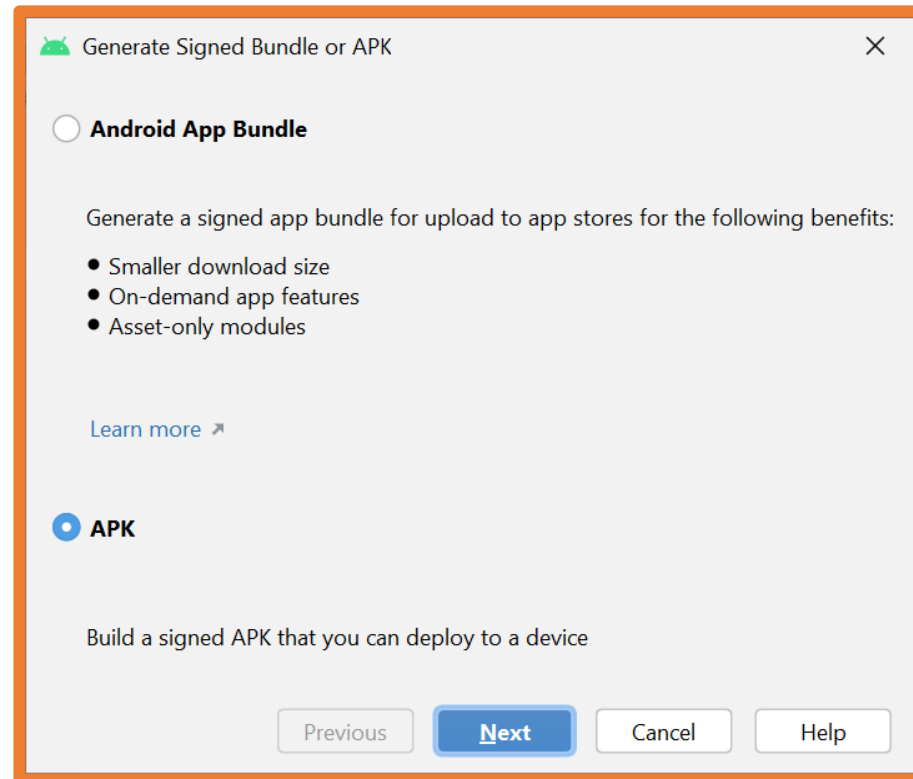
Build → Generate Signed Bundle / APK...



# 15.- Generación de archivos .apk

## APK firmado

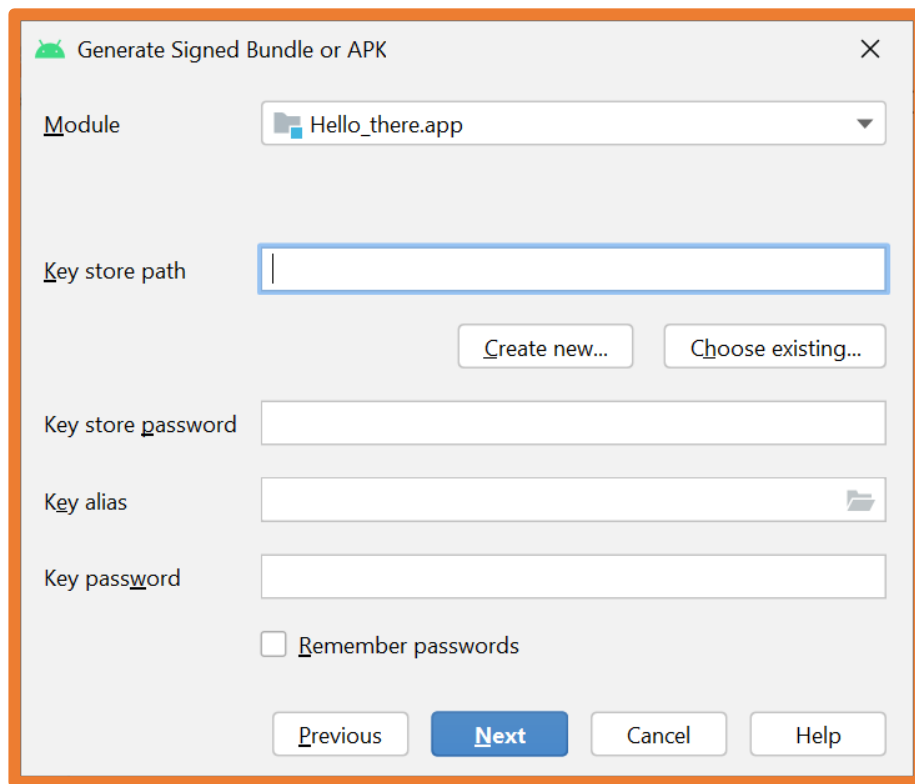
Se abrirá una ventana en la que se debe marcar la opción **APK**.



# 15.- Generación de archivos .apk

## APK firmado

En el siguiente paso se debe indicar la ruta al almacén y a la contraseña del almacén de claves, y también un alias y una contraseña para la clave.



The screenshot shows the 'Generate Signed Bundle or APK' dialog box. It has a title bar with an Android icon and a close button. The 'Module' dropdown is set to 'Hello\_there.app'. The 'Key store path' field is empty, with 'Create new...' and 'Choose existing...' buttons to its right. Below this are three text input fields: 'Key store password', 'Key alias', and 'Key password'. The 'Remember passwords' checkbox is unchecked. At the bottom are four buttons: 'Previous', 'Next' (highlighted in blue), 'Cancel', and 'Help'.

Field	Value / Action
Module	Hello_there.app
Key store path	[Empty]   Create new...   Choose existing...
Key store password	[Empty]
Key alias	[Empty]
Key password	[Empty]
Remember passwords	<input type="checkbox"/>
Navigation	Previous   Next   Cancel   Help

## 15.- Generación de archivos .apk

### **APK firmado**

Las claves se utilizan para acceder a la aplicación una vez cargada en Play Store, por ejemplo al actualizar la app.

Si se pierde la clave no se podrá acceder a la aplicación publicada ni se podrá actualizar.

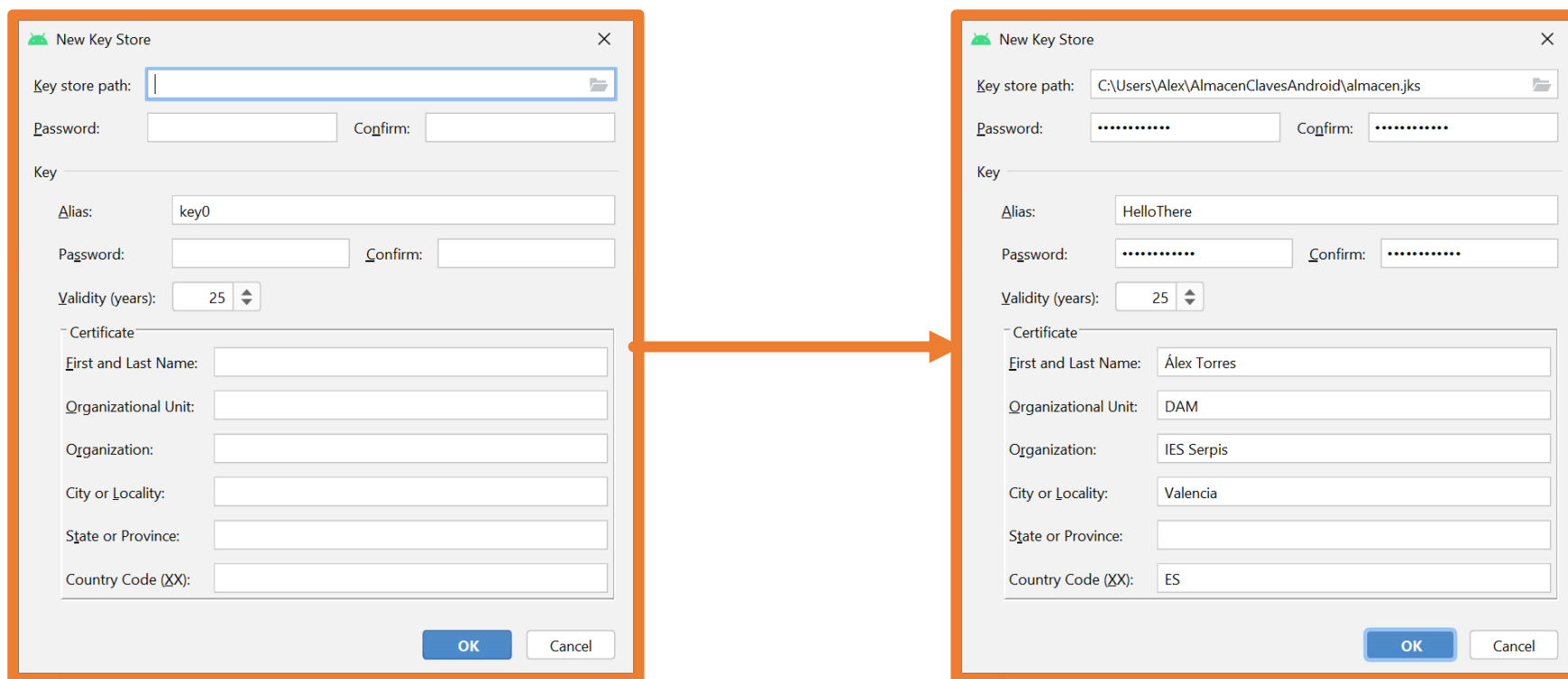
Si no se tiene almacén de claves se deberá crear uno.

# 15.- Generación de archivos .apk

## APK firmado

A la hora de crear el almacén de claves también se puede crear una clave para la aplicación actual.

Para cada clave se genera un certificado con toda la información del desarrollador.



The image shows two screenshots of the 'New Key Store' dialog box in Android Studio, illustrating the process of creating a new key and certificate. An orange arrow points from the first screenshot to the second.

**Left Screenshot (Initial State):**

- Key store path:** (Empty text field)
- Password:** (Empty text field)
- Confirm:** (Empty text field)
- Key:**
  - Alias:** key0
  - Password:** (Empty text field)
  - Confirm:** (Empty text field)
  - Validity (years):** 25
- Certificate:**
  - First and Last Name:** (Empty text field)
  - Organizational Unit:** (Empty text field)
  - Organization:** (Empty text field)
  - City or Locality:** (Empty text field)
  - State or Province:** (Empty text field)
  - Country Code (XX):** (Empty text field)

**Right Screenshot (Filled State):**

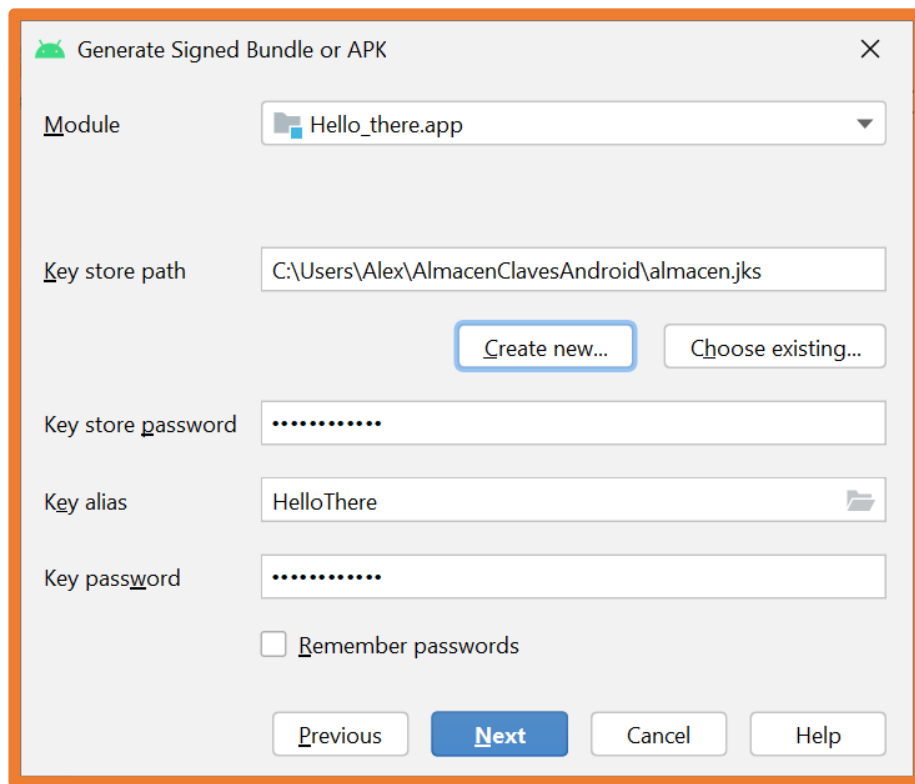
- Key store path:** C:\Users\Alex\AlmacenClavesAndroid\almacen.jks
- Password:** (Masked with dots)
- Confirm:** (Masked with dots)
- Key:**
  - Alias:** HelloThere
  - Password:** (Masked with dots)
  - Confirm:** (Masked with dots)
  - Validity (years):** 25
- Certificate:**
  - First and Last Name:** Álex Torres
  - Organizational Unit:** DAM
  - Organization:** IES Serpis
  - City or Locality:** Valencia
  - State or Province:** (Empty text field)
  - Country Code (XX):** ES



# 15.- Generación de archivos .apk

## APK firmado

Tras crear el almacén los campos en la ventana anterior se rellenarán automáticamente.



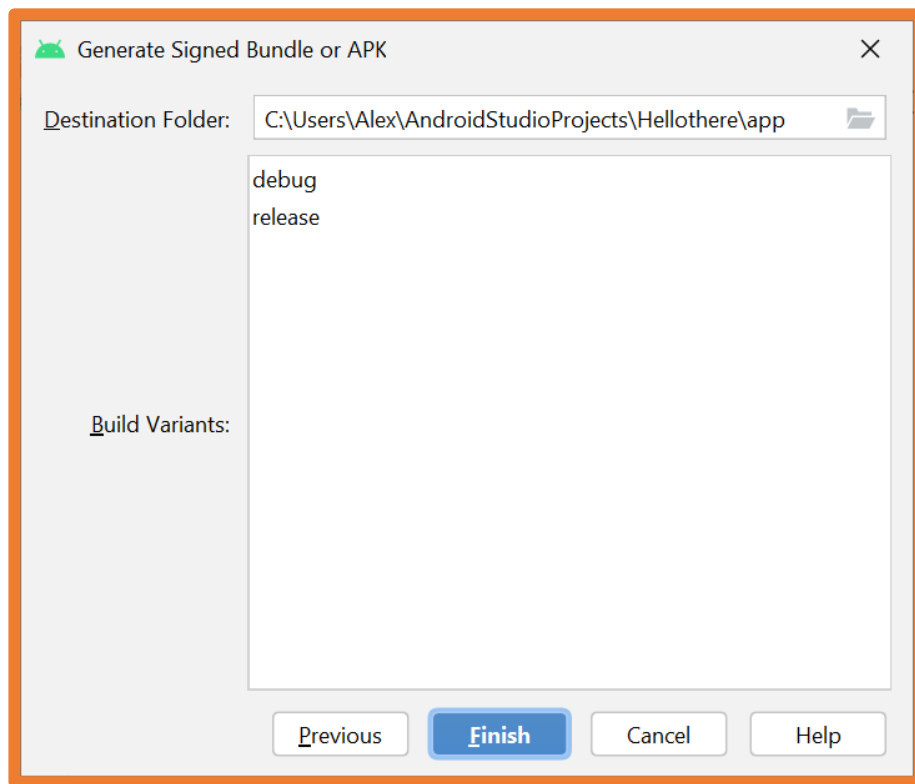
The screenshot shows the 'Generate Signed Bundle or APK' dialog box. The 'Module' dropdown is set to 'Hello\_there.app'. The 'Key store path' is 'C:\Users\Alex\AlmacenClavesAndroid\almacen.jks', with 'Create new...' and 'Choose existing...' buttons below it. The 'Key store password' and 'Key password' fields are masked with dots. The 'Key alias' is 'HelloThere'. The 'Remember passwords' checkbox is unchecked. The 'Next' button is highlighted in blue.

Field	Value
Module	Hello_there.app
Key store path	C:\Users\Alex\AlmacenClavesAndroid\almacen.jks
Key store password	.....
Key alias	HelloThere
Key password	.....
Remember passwords	<input type="checkbox"/>

## 15.- Generación de archivos .apk

### APK firmado

En la última ventana se pregunta qué versión de la aplicación se quiere firmar. Se selecciona **release** que es la versión que se lanzará.



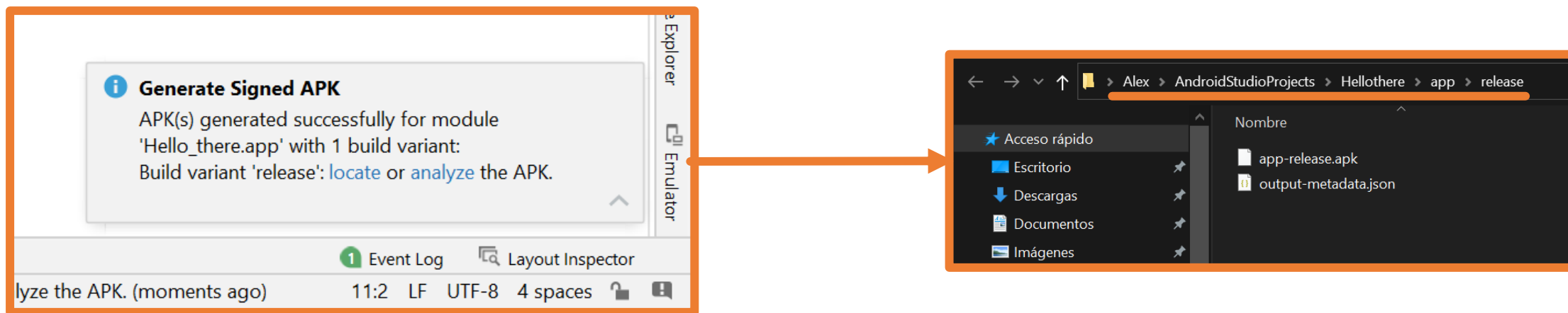
## 15.- Generación de archivos .apk

### APK de depuración

Una vez finalizada la creación del APK se muestra una notificación.

Al hacer clic en **locate** se abrirá el directorio con el APK firmado.

Si pierdes la notificación puedes buscar el APK en el directorio del proyecto.



## 15.- Generación de archivos .apk

Como se puede observar tanto el .apk de depuración como el firmado se generan con un nombre genérico: app-debug.apk y app-release.apk.

Este nombre se puede cambiar, por ejemplo:

HelloThere-debug.apk y HelloThere.apk

Si se quiere probar la aplicación en un emulador no hace falta que el .apk esté firmado.