U.T.1 ANÁLISIS DE TECNOLOGÍAS PARA DISPOSITIVOS MÓVILES.

Un **teléfono inteligente** (*smartphone*)

es un teléfono móvil construido sobre una plataforma informática,

- ★ con una mayor capacidad para almacenar datos,
- ★ realizar actividades semejantes a una mini computadora
- ★ mayor conectividad que un teléfono móvil convencional.
- ★ El término «inteligente» porque puede usarse como un ordenador de bolsillo

Serie iPhone de Apple, Serie Optimus de LG, Serie Nexus de Google, Serie Xperia de Sony Mobile Communications, Serie Galaxy de Samsung, ...

Un **sistema operativo móvil** o **SO móvil**

es un programa encargado de gestionar todos los recursos tanto hardware cómo software de un dispositivo móvil.

están más orientados a:

★ la conectividad inalámbrica,

★ los formatos multimedia para móviles

★ las diferentes formas de introducir informacion en ellos.

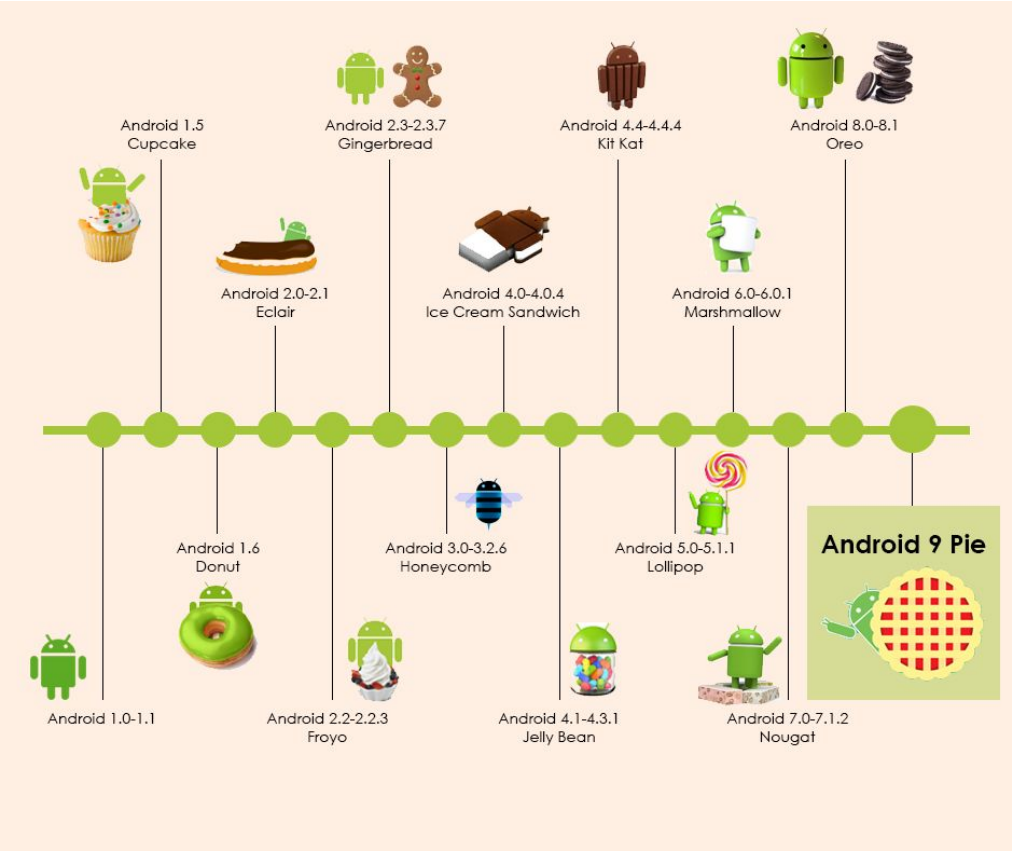
Android (de Google), iOS (de Apple), Windows Phone (de Microsoft).

Últimas **tecnologías** existentes en el mercado

- ★ Java Platform, Micro Edition (Java ME) proporciona un entorno robusto y flexible para aplicaciones que se ejecutan en dispositivos integrados y móviles en el Internet de las Cosas: microcontroladores, sensores, teléfonos móviles, PDA, decodificadores de TV, impresoras y más
- ★ iPhone es 100% Apple. Es programado con *Objective C*. Actualmente han desarrollado un nuevo lenguaje de programación **Swift** para desarrollo de aplicaciones móviles en los sistemas operativos de Apple, como iOS, macOS, watchOS y tvOS.
- ★ Android Es una plataforma de código abierto que utiliza **Linux** y **Java** como su punto de acceso con APIs. **Kotlin** ha ganado rápidamente popularidad como una alternativa moderna a Java para el desarrollo de aplicaciones Android.

Evolución de Android

Las sucesivas versiones de Android, casi desde los inicios de este sistema operativo, llevan los nombres de diferentes caramelos ordenados por orden alfabético .



Android Inc. fue creada en 2003, y comprada por Google en agosto de 2005. Un año después **apareció el primer prototipo de lo que sería un *smartphone* Android**, Fue en el año 2008 cuando nuestro sistema operativo apareció en el mercado con Android 1.5 hasta la versión de Android llamada oficialmente Android 9 Pie.

Las últimas versiones ya no llevan asociado nombre : Android 10, Android 11, Android 12...

CARACTERÍSTICAS DE ANDROID.

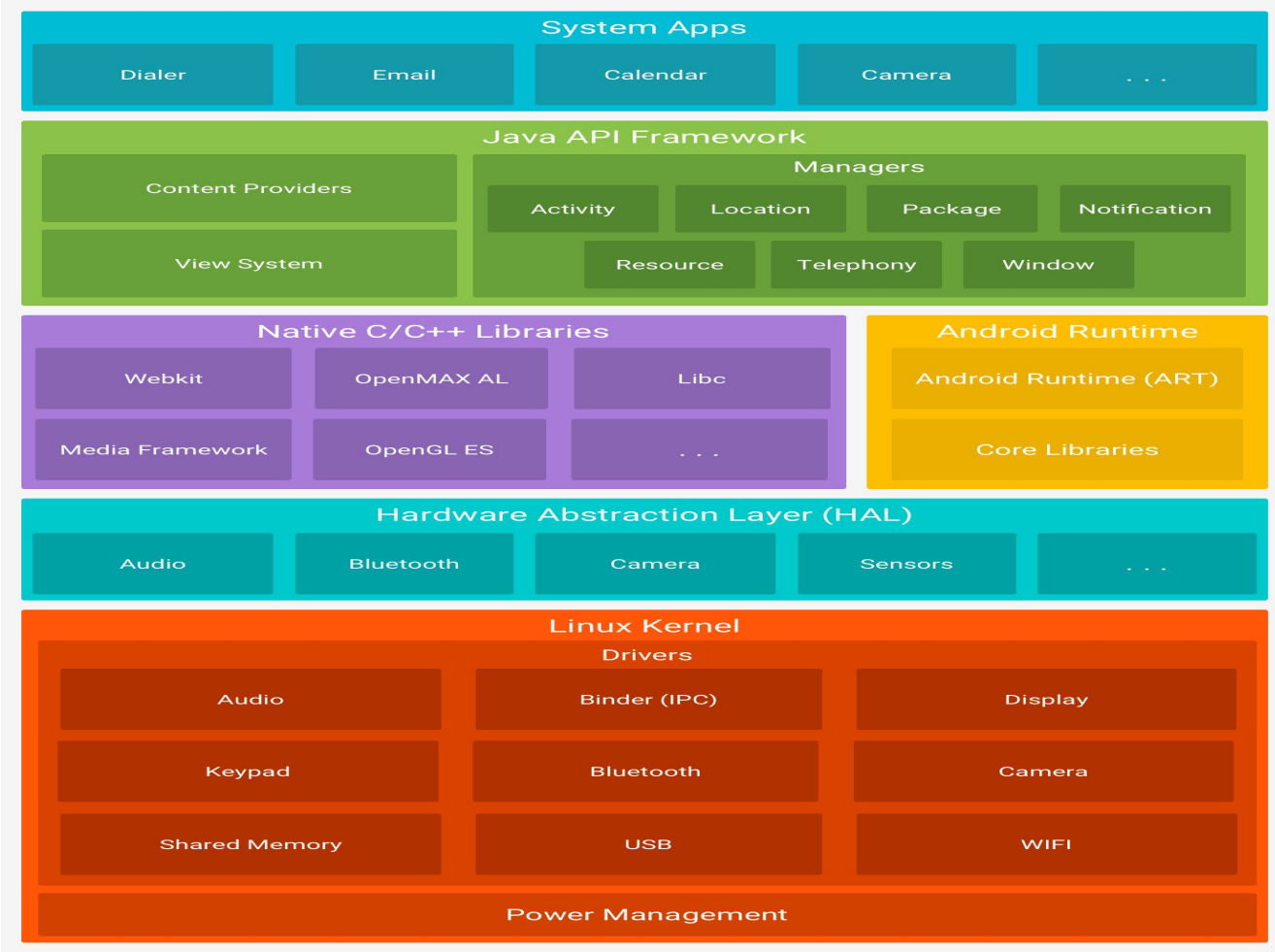
Las características más destacadas son:

- ★ Código abierto
- ★ Servicios: acceso al hardware (acelerómetro, GPS, ...). SQLITE como base de datos
- ★ Aplicaciones hechas de componentes. Adaptable a muchas pantallas y resoluciones.
- ★ Capacidades multimedia.
- ★ Seguridad (entre aplicaciones y para el usuario)
- ★ Navegador web incluido (webkit)
- ★ Programación en Java o kotlin (o en C/C++, aunque menos recomendable)
- ★ Google mantiene un mercado de aplicaciones predefinido (Google play).
- ★ Bluetooth
- ★ mensajería, notificaciones, multitarea real de aplicaciones.

La arquitectura de Android está estructurada en capas.

- ★ KERNEL LINUX : Núcleo de Android que contiene los driver necesarios para la pantalla, teclado, cámara, audio,..
- ★ Capa de abstracción de Hardware (HAL): contiene módulos de biblioteca para cada componente hardware: cámara, bluetooth,... Cuando se realiza una llamada para acceder al hardware del dispositivo, el sistema Android carga el módulo de biblioteca para el componente de hardware en cuestión.
- ★ LIBRERIAS : librerias nativas, están escritas en C/C++. Surface Manager: administrador de pantalla Open GL: Librerias para gráficos 3D. SGL: Librería para 2D. Media Framework: librería que contiene utilidades para reproducir videos. SQLite : Base de datos de android.
- ★ ANDROID RUNTIME: **Dalvik** Virtual Machine, Máquina virtual de Android en dispositivos anteriores a la versión de **Android 5.0 (Api 21)**, Actualmente **ART** entorno de ejecución Android Runtime, permite ejecutar varias máquinas virtuales optimizando el espacio de memoria empleado.
- ★ JAVA API FRAMEWORK capa intermedia que maneja los conceptos de activities, package, views: button, textview,... recursos, etc. Estas API son los cimientos que necesitas para crear apps de Android.
- ★ Apps del sistema En Android se incluye un conjunto de apps centrales para correo electrónico, mensajería SMS, calendarios, navegación en Internet y contactos, ...

Arquitectura de la plataforma Android

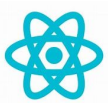


Herramientas para desarrollar aplicaciones móviles

Android Studio Entorno de Desarrollo Integrado para la creación rápida de aplicaciones en el sistema operativo móvil **Android** de Google. Utiliza java cómo lenguaje de programación.

Titanium SDK es un framework de código abierto que permite la creación de aplicaciones móviles nativas en plataformas iOS y Android a partir de una única base de código **JavaScript**.

Flutter es un entorno de desarrollo de código abierto creado por Google para crear aplicaciones multiplataforma a partir de una sola base de código. Utiliza **Dart,** un lenguaje optimizado para aplicaciones rápidas en cualquier plataforma.



React Native, es un framework de código abierto creado por Meta Platforms, Inc. Se utiliza para desarrollar aplicaciones para Android, iOS, Web, Windows al permitir que los desarrolladores usen React con las características nativas de estas plataformas

ANDROID STUDIO

Android Studio es un entorno de desarrollo integrado para el sistema operativo **Android** lanzado por Google, diseñado para ofrecer herramientas para el desarrollo de aplicaciones.

Para descargarlo:

https://developer.android.com/studio

Android Studio proporciona todo lo necesario para comenzar a desarrollar aplicaciones para Android, incluyendo el Android Studio IDE, las herramientas del SDK de Android y emuladores para la ejecución de las aplicaciones.

Antes de configurar Android Studio, asegúrese de haber instalado JDK 7 o posterior ya que Android no lleva la máquina virtual de java (jdk), lleva una máquina virtual Dalvik o ART.

Instalación y configuración

★ Lanza el archivo .exe que acaba de descargar.

★ Siga el asistente de instalación para instalar **Android Studio**, las herramientas **SDK y Android Virtual Devices**.

El asistente nos dará unas rutas de archivos donde realizará la instalación. Para las SDK mejor elegir una ruta en la unidad de disco **C:** para que pueda ser compartida por todos los usuarios del equipo.

★ Si el script lanzador no encuentra donde está instalado Java, debemos establecer una variable de entorno que indique su ubicación.

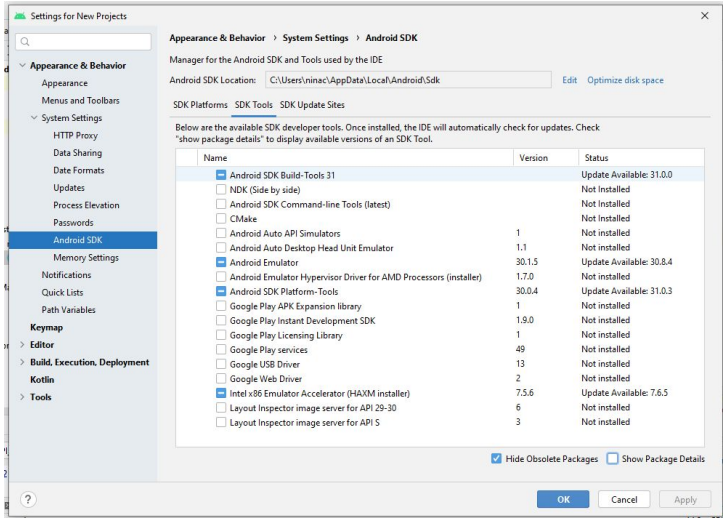
Seleccione **el menú Inicio> Equipo> Propiedades del sistema> Propiedades avanzada del sistema** . A continuación, abra la pestaña **Variables de entorno avanzados** y añadir la ruta jdk, por ej. C:\Program Files (x86)\Java\jdk1.8.0\bin o la versión jdk más actualizada de tu equipo

SDK Manager. Añadir paquetes SDK Tools

Para añadir paquetes o actualizar los existentes, debemos ejecutar el **SDK Manager de Android,** que proporciona las herramientas sdk, plataformas y componentes necesarios para desarrollar nuestras app.

Para abrir SDK Manager haz clic en **Tools > Android > SDK Manager** o en el icono **SDK Manager** . Abra **SDK Tools** y seleccione la última actualización de:

- Android SDK Tools
- Android SDK Platform-tools
- Android SDK Build tools
- Android Emulator



SDK Manager. Añadir paquetes SDK Platform

Platform SDK debes instalar al menos una versión de la plataforma de Android.

Cada versión proporciona varios paquetes diferentes.

Usa la versión más reciente de la plataforma como destino de tu compilación para ofrecer la mejor experiencia del usuario en los dispositivos más recientes.

Si deseas ver todos los paquetes disponibles para cada plataforma de Android, haz clic en **Show Package Details** en la parte inferior de la ventana.

Abra **SDK Platform** y seleccione:

Android SDK platform XX

Imágenes del sistema Intel o ARM

Una imagen del sistema para el emulador, como **ARM EABI Sistema v7a Imagen o Intel x86 Atom System Image** que es mucho más rápida siempre que nuestro PC tenga un procesador intel.

Añadir librerías de soporte y servicios de google play

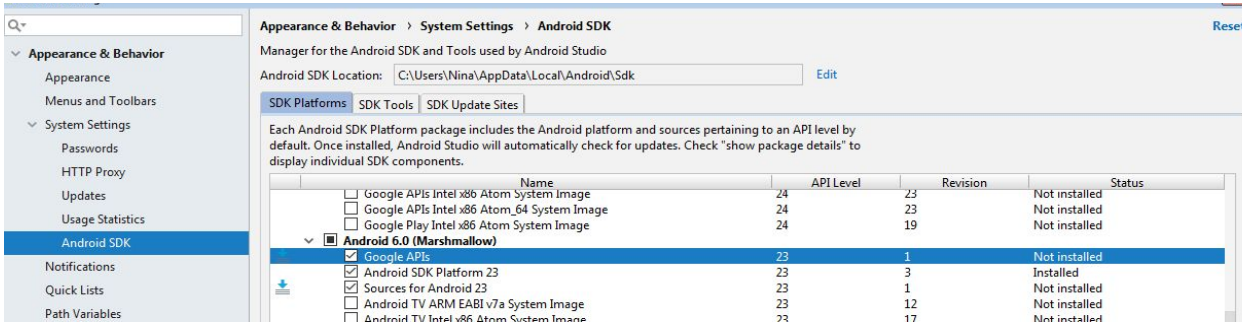
Las API de servicios de Google Play proporcionan una variedad de funciones y servicios para sus aplicaciones Android, como por ejemplo:

Autenticación de usuario

Reconocimiento de la ubicación,

Mapas de Google,

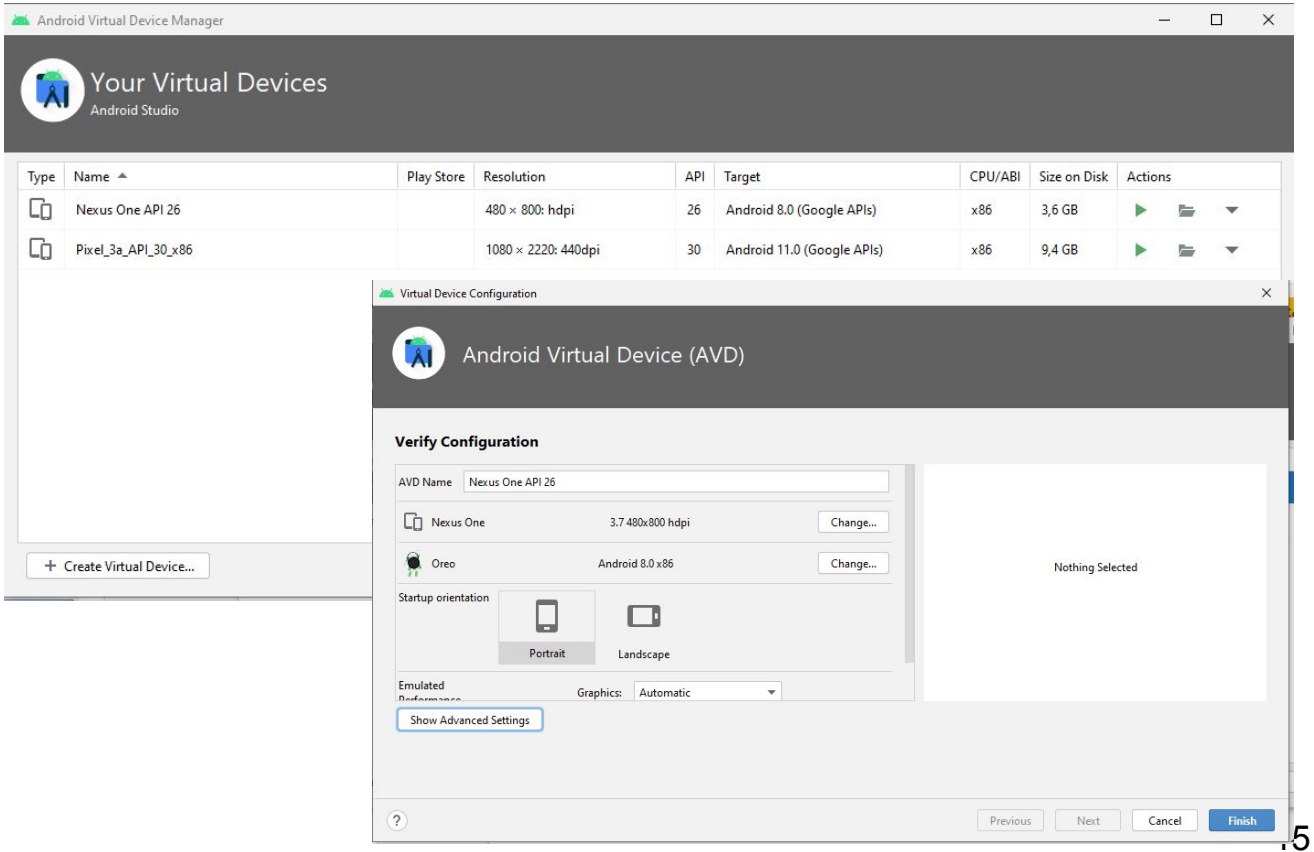
VideoLLamadas, ...



Si piensas usar algún servicio de la API de Google Play, debes descargar Google Apis de la plataforma Android elegida. Puedes hacerlo desde el Sdk manager o al crear el emulador

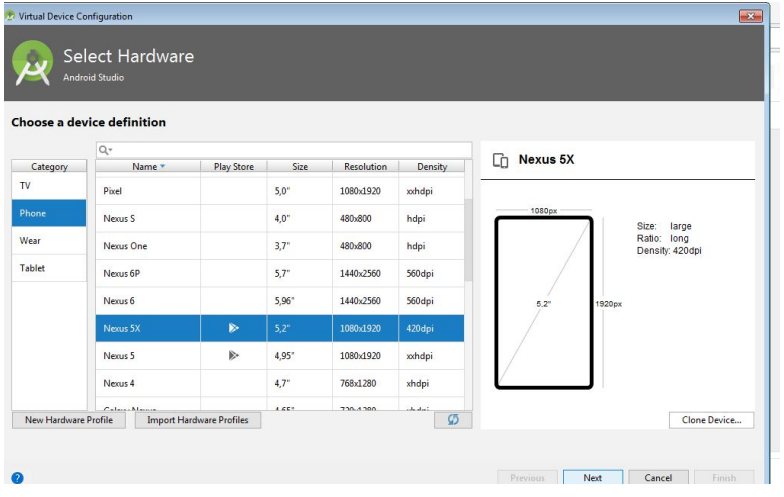
Gestión de AVDs. Dispositivo Virtual

AVD Manager es una herramienta que permite crear y gestionar dispositivos virtuales Android (AVDs), que definen configuraciones de los dispositivos para el emulador de Android .

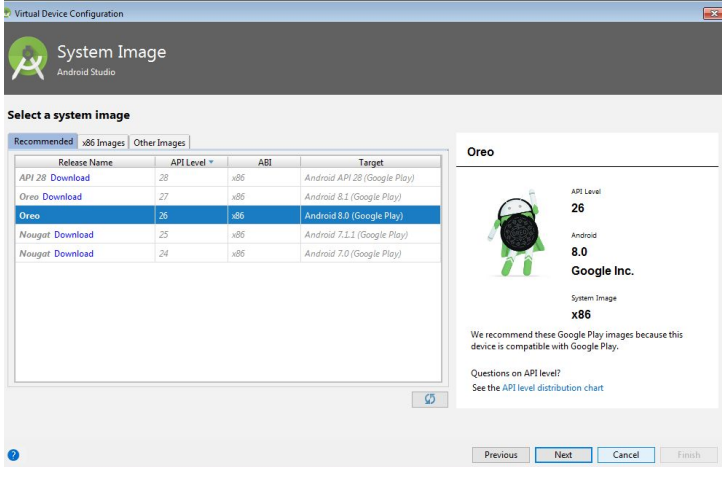


Crear un AVD

- Desde la pantalla principal, haga clic en **Crear dispositivo virtual** .
- En la ventana Seleccionar Hardware, seleccione una configuración de dispositivo, como Nexus 6, a continuación, haga clic en **Siguiente** .

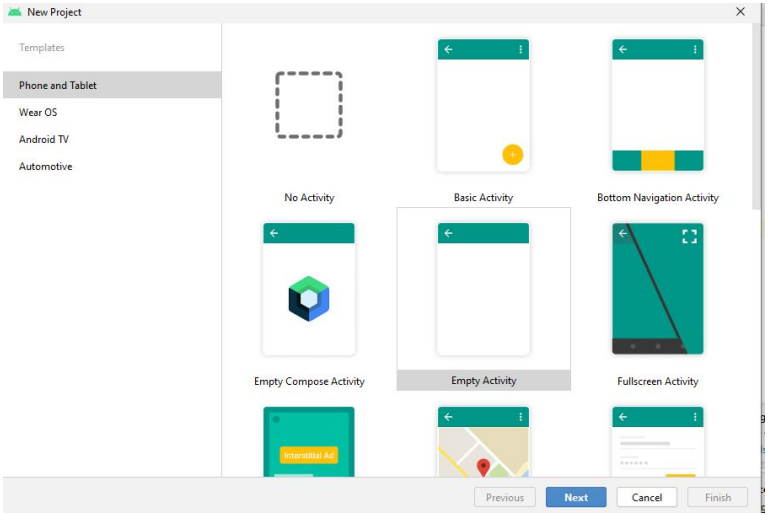


- Seleccione la versión del sistema que desee para la AVD y haga clic en **Siguiente**.
- Verifique los ajustes de configuración, a continuación, haga clic **en Finalizar**



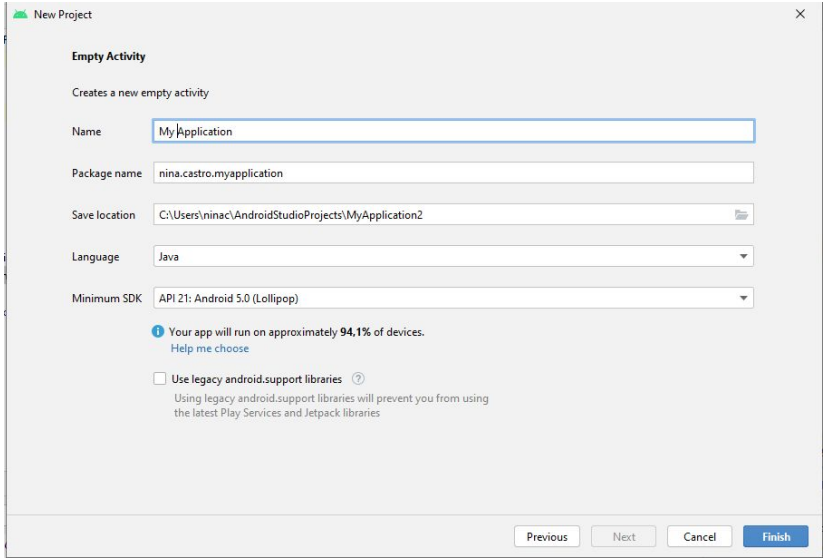
Crear un proyecto android

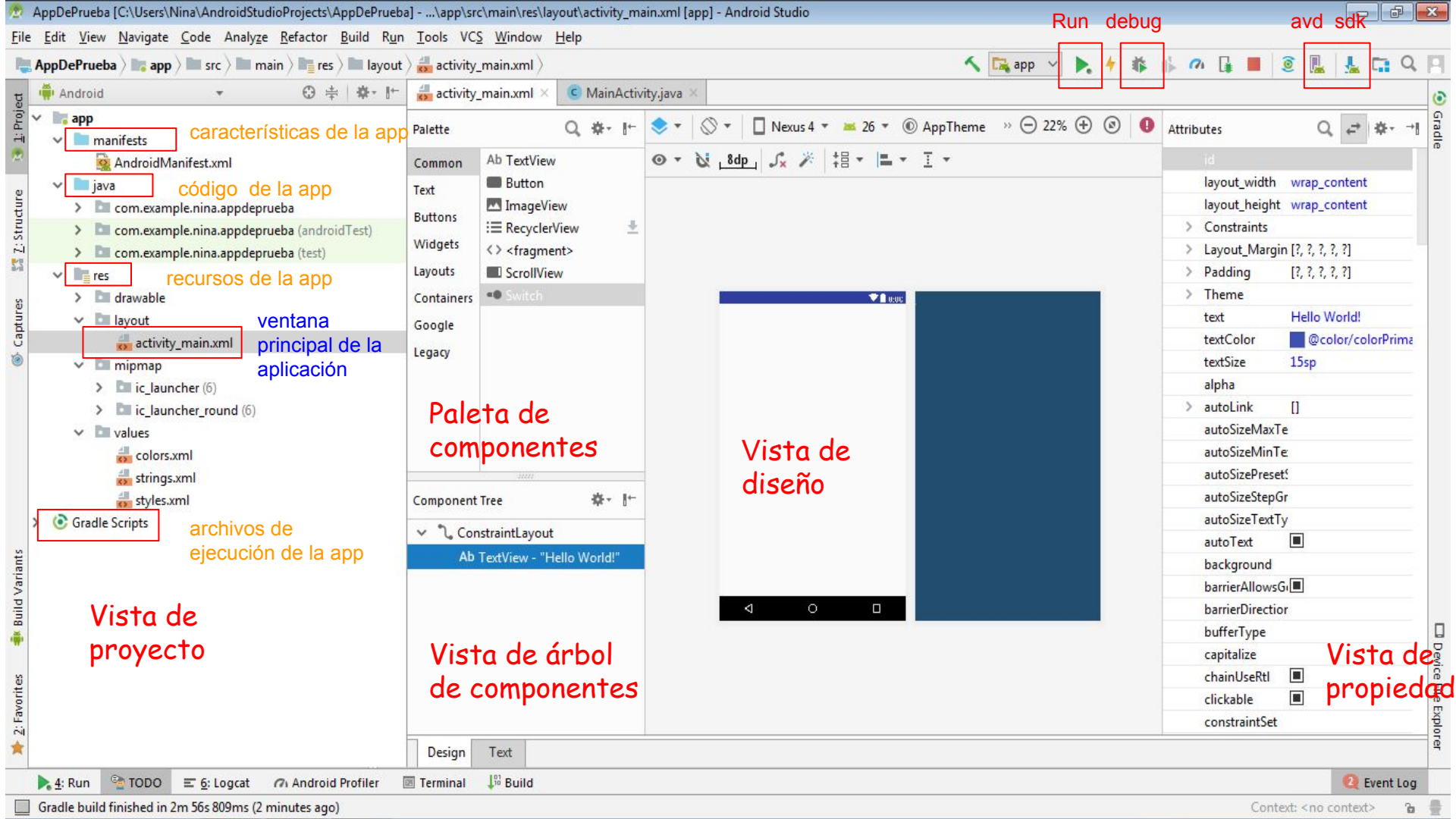
1. Selecciona File - new Proyect



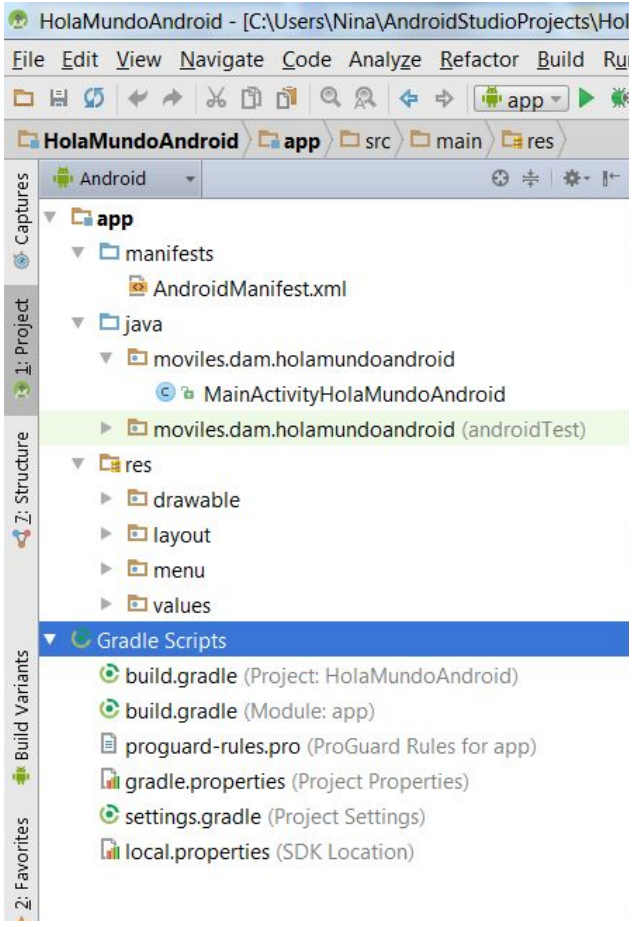
2. Selecciona Blank Activity - Next

3. Escribe el nombre del proyecto - Elige el api mínima de android donde podrá ejecutarse la aplicación - Finist




AppDePrueba [C:\Users\Nina\AndroidStudioProjects\AppDePrueba] - ...\app\src\main\res\layout\activity_main.xml [app] - Android Studio
Run
debug
avd
sdk
características de la app
código de la app
recursos de la app
ventana principal de la aplicación
archivos de ejecución de la app
Vista de proyecto
Paleta de componentes
Vista de diseño
Vista de árbol de componentes
Vista de propiedad

Vista de Proyecto Android

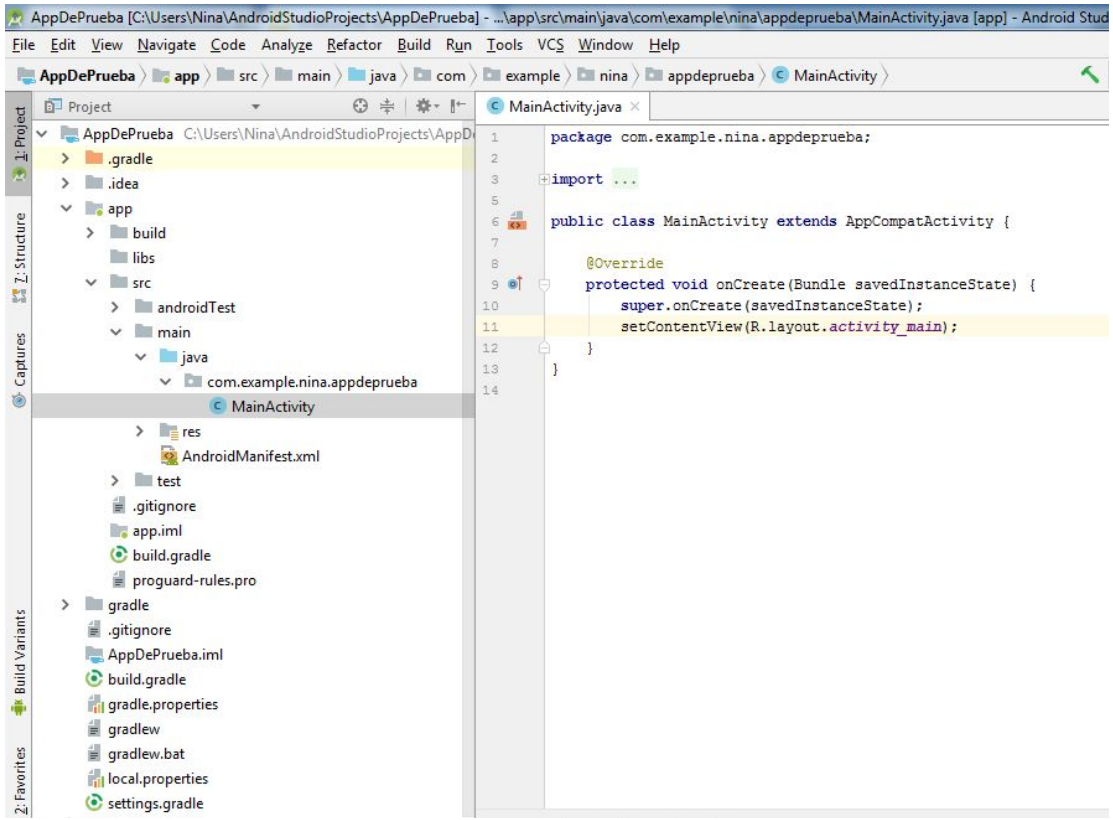


Muestra las 4 carpetas siguientes:

- **java /** archivos de código fuente de la aplicación.
- **Manifest/** - archivo de manifiesto de la aplicación.
- **res /** - Archivos de recursos de la aplicación.
- **Scripts Gradle/** - Archivos de creación y de propiedades

Código fuente de la aplicación: Carpeta **Java**

java/ Contiene todos los ficheros .java del proyecto.

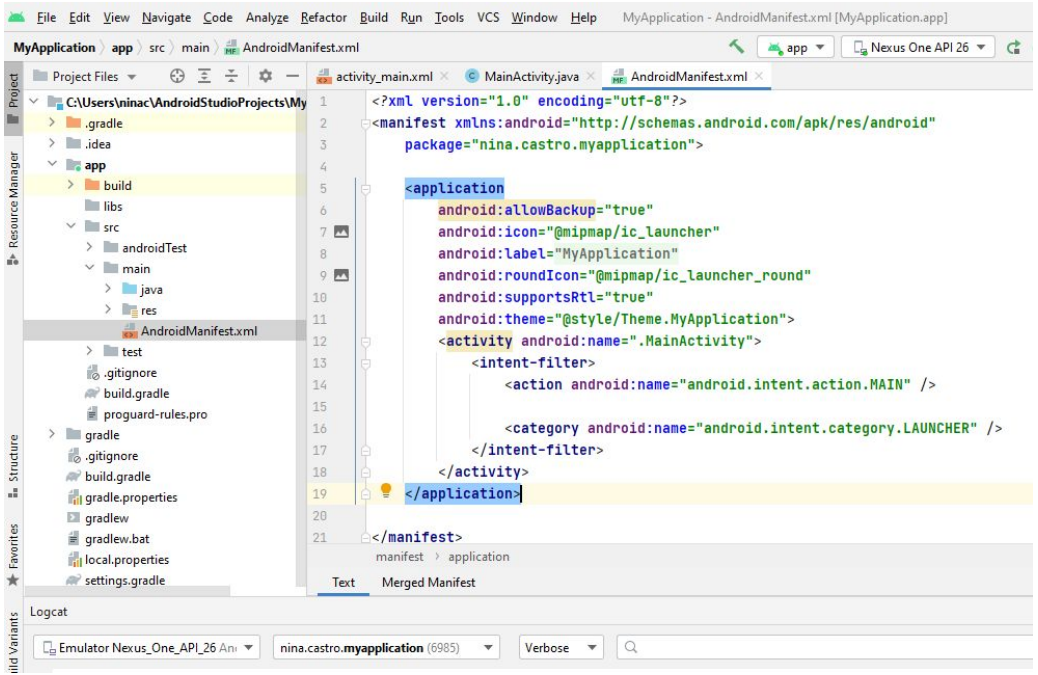


MainActivity.java es el fichero java de la ventana principal de la aplicación.

Archivo AndroidManifest.xml: Carpeta **Manifest**

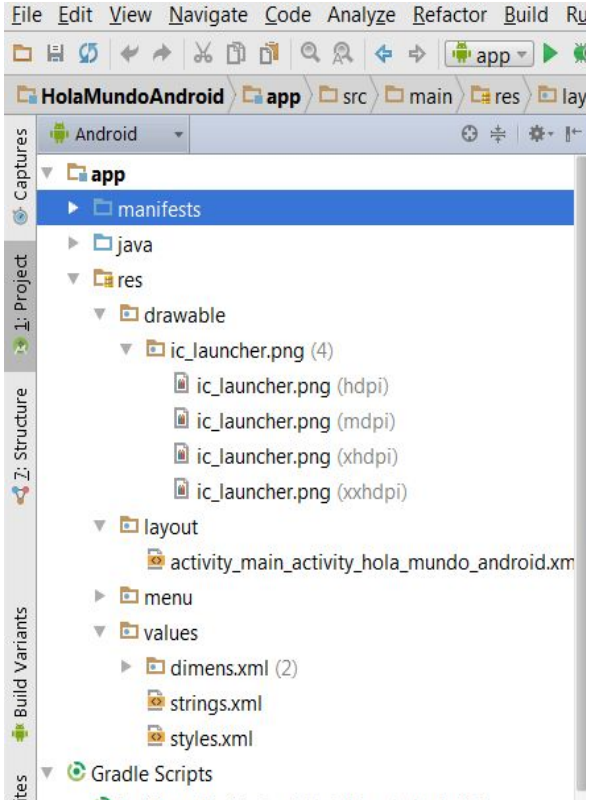
El archivo AndroidManifest.xml mantiene una descripción en XML de las características básicas de la aplicación.

(nombre, versión, icono, …), sus componentes (actividades, pantallas, mensajes, …), las librerías auxiliares utilizadas, o los permisos necesarios para su ejecución.



Recursos de la aplicación: Carpeta **res**

res/ Contiene todos los ficheros de recursos necesarios para el proyecto.



Imágenes, vídeos, cadenas de texto, etc.

Los diferentes tipos de recursos se distribuyen entre las siguientes subcarpetas: **/res/drawable/, /res/layout/, /res/values/…**

Entre los recursos creados por defecto, está el layout "*activity_main.xml*": interfaz gráfica de la pantalla principal de la aplicación.

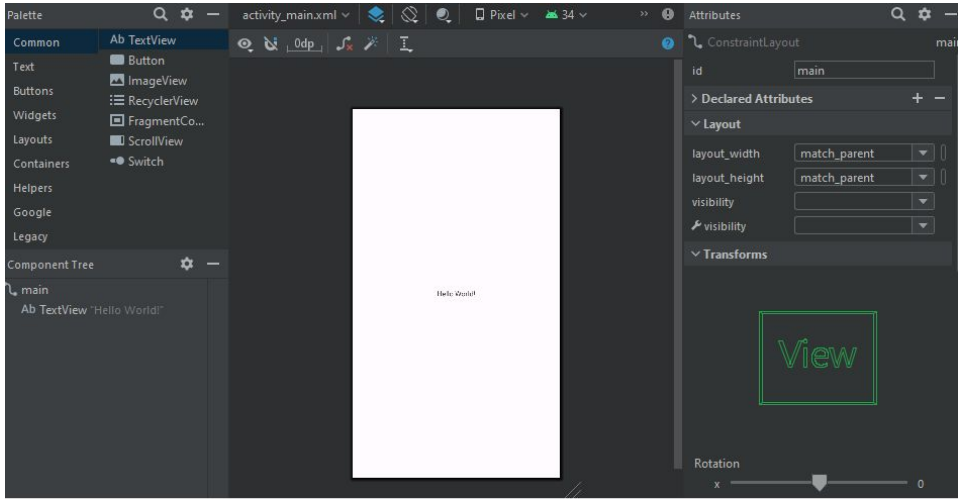
Para hacer uso de estos recursos desde el código de nuestro proyecto utilizamos el **ID** (identificador) del recurso.

Vista gráfica de la aplicación. Carpeta *res/layout*

El archivo activity_main.xml: Es el layout principal de la aplicación.

Contiene la definición de la interfaz gráfica de la pantalla principal de la aplicación.

Vista de diseño del layout



Vista del código xml del layout

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout 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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Compilar y ejecutar tu app

Para compilar y ejecutar tu app, haz clic en **Run** Android Studio compila tu app con **Gradle**, solicita que selecciones un destino de implementación (un emulador o un dispositivo conectado) y, luego, implementa la app en dicho destino.

Vista de la ejecución de la app en el emulador.

Dispositivos para ejecutar tu app

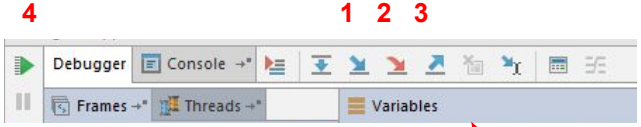
Si deseas usar Android Emulator para ejecutar tu app, debes preparar un Android Virtual Device (AVD). Si todavía no creaste uno, haz clic en **Run** y luego en **Create New Emulator**, en el diálogo **Select Deployment Target**. Sigue las instrucciones del asistente Virtual Device Configuration para definir el tipo de dispositivo que deseas emular.

Si usas un dispositivo Android físico, debes habilitar la depuración USB en el dispositivo. Para obtener más información, consulta Ejecutar apps en un dispositivo de hardware.

Conectarte a tu dispositivo mediante Wi-Fi A partir de Android 11 se permite la implementación y depuración de tu app de forma inalámbrica desde tu estación de trabajo mediante Android Debug Bridge (adb).

Depurar una app

Puedes implementar tu app en el modo de depuración. Para ello, haz clic en Debug. La ejecución de tu app en el modo de depuración te permite configurar puntos de interrupción en el código, examinar variables y evaluar expresiones en el tiempo de ejecución, y también ejecutar herramientas de depuración. Para obtener más información, consulta Depuración de tu app.



1. Para avanzar a la siguiente línea del código **Step Over** .
2. Para avanzar a la primera línea dentro de una llamada a un método, **Step Into** .
3. Para avanzar a la siguiente línea fuera del método actual, haz clic en **Step Out** .
4. Para continuar ejecutando la app normalmente, **ResumeProgram** .