

Aplicaciones Móviles

SC09S INTRODUCCIÓN

Crear un proyecto con Android Studio

1. Conocer el ambiente de desarrollo integrado.
2. Conocer la interfase del usuario de Android Studio.
3. Conocer los diferentes componentes que se integran al desarrollo de una app.
4. Establecer la versión objetivo de desarrollo de una app.
5. Uso de dispositivos virtuales y físicos para crear la app.
6. Uso del gradle y las dependencias que se pueden integrar para el desarrollo de una app.

Que es Android?

- Android es un sistema operativo de código abierto, creado por Google específicamente para su uso en dispositivos móviles (teléfonos celulares y tabletas)
- Basado en Linux (kernel 2.6)
- Puede ser programado en C y C++ pero se hace más desarrollo de aplicaciones en Java (Java accesa a bibliotecas C vía JNI (Java native Interface))
- Soporta Bluetooth, Wi-Fi y redes 3G, 4G y 5G

Android y Java

Después de escribir un programa en Java para Android, hacemos clic en un botón y nuestro código se transforma en un nuevo código, el código que entiende Android.

Esta otra forma se denomina bytecode y el proceso de transformación se denomina compilación.

Luego, cuando el usuario instala nuestra aplicación, el código de bytecode es traducido por otro proceso conocido como Android Runtime (ART) en código máquina.

Este es el formato de ejecución más rápido posible, bytecode se cambia a código máquina que es rápido para el dispositivo. Este proceso de compilación añade a nuestra app la optimización necesaria para la administración de batería y memoria.

Bluetooth

- Open wireless technology
 - Desarrollada por Ericsson (1994)
 - Originalmente debían para reemplazar por cable RS-232
 - Corta distancia a través de bajo consumo, radio de corta distancia
- Permite la creación de redes de área personales.
 - Sobre todo para conectar dispositivos periféricos inalámbricos a un ordenador (ratones, auriculares, micrófonos, teclados...
- También puede utilizarse para comunicarse de forma inalámbrica entre dos equipos host (reemplazar cables seriales)

WiFi

Utilizado para productos certificados que pertenecen a una clase de red de área local inalámbrica basada en IEEE estándar de la marca 802.11

- Actualmente hay 3 versiones de 802.11 en uso común:
 1. B, cerca de 150 pies, 300 pies al aire libre
 2. G, 54 Mbits unos 150 pies, 300 pies al aire libre
 3. N, 600 Mbits, aproximadamente 1,5 millas al aire libre, utiliza MIMO (múltiple entradas y salidas antenas)
 4. WiFi 5: 802.11ac in 2013 @ 6800Mbps
 5. WiFi 6: 802.11ax in 2018 @ 11000Mbps

4G

- Proporciona una completa y segura solución basada en IP para telefonía basada en IP, internet de banda ancha ultra, servicios de juego y multimedia streaming.
- Tasa de datos máxima de 100 Mbit para dispositivos de alta movilidad y 1 GB para dispositivos de baja movilidad.

5G

5G es la quinta generación de tecnología celular. Está diseñada para aumentar la velocidad, reducir la latencia y mejorar la flexibilidad de los servicios inalámbricos. La tecnología 5G ofrece una velocidad máxima teórica de 20 Gbps, mientras que la velocidad máxima de la tecnología 4G es solo de 1 Gbps.

5G también ofrece menor latencia, lo que puede mejorar el rendimiento de las aplicaciones comerciales y de otras experiencias digitales (como juegos en línea, videoconferencias y automóviles con piloto automático).

Paquetes comúnmente usados

- Widgets y controles de interfaz de usuario
- Layout para interfase de usuario.
- Red segura y web browsing
- Storage estructurado y bases de datos relacionales (SQLite RDBMS)
- Gráficos 2D y 3D SGL and OpenGL
- Soporte de media visual y de audio
- Acceso a hardware opcional (GPS)

Ecosistema de Android

Una plataforma abierta móvil, para dispositivos móviles, embebidos (ejemplo, dispositivo Android para autos, TV's etc.) y usables en nuestro cuerpo.

- Google es el principal responsable de su actualización, aunque otras personas contribuyen al sistema.
- Cada fabricante de dispositivos móviles puede adaptar “customize” Android a sus necesidades.

Requerimientos Técnicos Windows

1. Microsoft® Windows® 7/8/10 (64-bit)
2. 4 GB RAM as a minimum; 8 GB RAM recommended
3. 2 GB of available disk space as a minimum; 4 GB recommended (500 MB for the Integrated Development Environment (IDE) + 1.5 GB for the Android Software Development Kit (SDK) and emulator system image)
4. 1,280 x 800 minimum screen resolution

Requerimientos Técnicos Mac

1. Mac® OS X® 10.10 (Yosemite) or higher, up to 10.14 (macOS Mojave)
2. 4 GB RAM as a minimum; 8 GB RAM recommended
3. 2 GB of available disk space as a minimum; 4 GB recommended (500 MB for the IDE + 1.5 GB for the Android SDK and emulator system image)
4. 1,280 x 800 minimum screen resolution

Mercado

1. Android es el sistema operativo más utilizado en el mundo.
2. Incluso Windows está rezagado con respecto a la adopción generalizada de la plataforma funcional de teléfonos inteligentes propiedad de Google. iOS ni siquiera está cerca.
3. Cuando se trata de dispositivos móviles, Apple ha ganado algo de terreno en los últimos años, pero Android todavía impulsa alrededor del 75% de todos los teléfonos inteligentes y tabletas.

fuentes: [Android Market Share and 20+ Statistics for 2024 \(techjury.net\)](#)

Mercado

1. Android tiene una cuota del 71,77%, mientras que iOS representó el 27,6% del mercado de sistemas operativos móviles.
2. Hay más de 2.500 millones de usuarios activos de Android en más de 190 países.
3. Un total de 1.570 millones de dispositivos Android vendidos en 2022, frente a los 0.980 millones de 2021.
4. Android representó el 44,86% del mercado de sistemas operativos móviles, mientras que iOS representó el 54,74% en enero de 2023.
5. El 97% del malware para teléfonos inteligentes se dirige a teléfonos Android.
6. Las ventas de Android están aumentando en Estados Unidos, Europa, Japón, India y China.
7. La cuota de mercado de Samsung en Estados Unidos fue del 28,57% en 2023, frente al 26,45% de 2022.
8. La participación de Android en todos los dispositivos conocidos fue del 70,97%.

Historia Inicial

- 30 Abril 2009, the official 1.5 (Cupcake) update para Android.
- 15 Septiembre 2009, the 1.6 (Donut) SDK released.
- 26 Octubre 2009 the 2.0 (Eclair) SDK released
- 3 Diciembre 2009 the 2.0.1 SDK released.
- 12 Enero 2010 the 2.1 SDK released.

Historia Inicial



Alpha

A



Beta

B



Cupcake

C



Donut

D



Eclair

E



Froyo

F



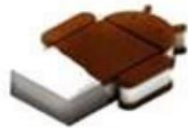
Gingerbread

G



Honeycomb

H



Ice Cream Sandwich

I



Jelly Bean

J



KitKat

K



Lollipop

L



Marshmallow

M



N

Características de Android

Re-uso y reemplazo de los componentes.

Dalvik virtual machine (Descontinuado). Optimiza el código Java para correr en móviles.

Android Runtime (ART). Sucesor de DVM, y usado a partir de Lollipop.

Browser integrado

Gráficas optimizadas.

SQLite

Soporte Multimedia

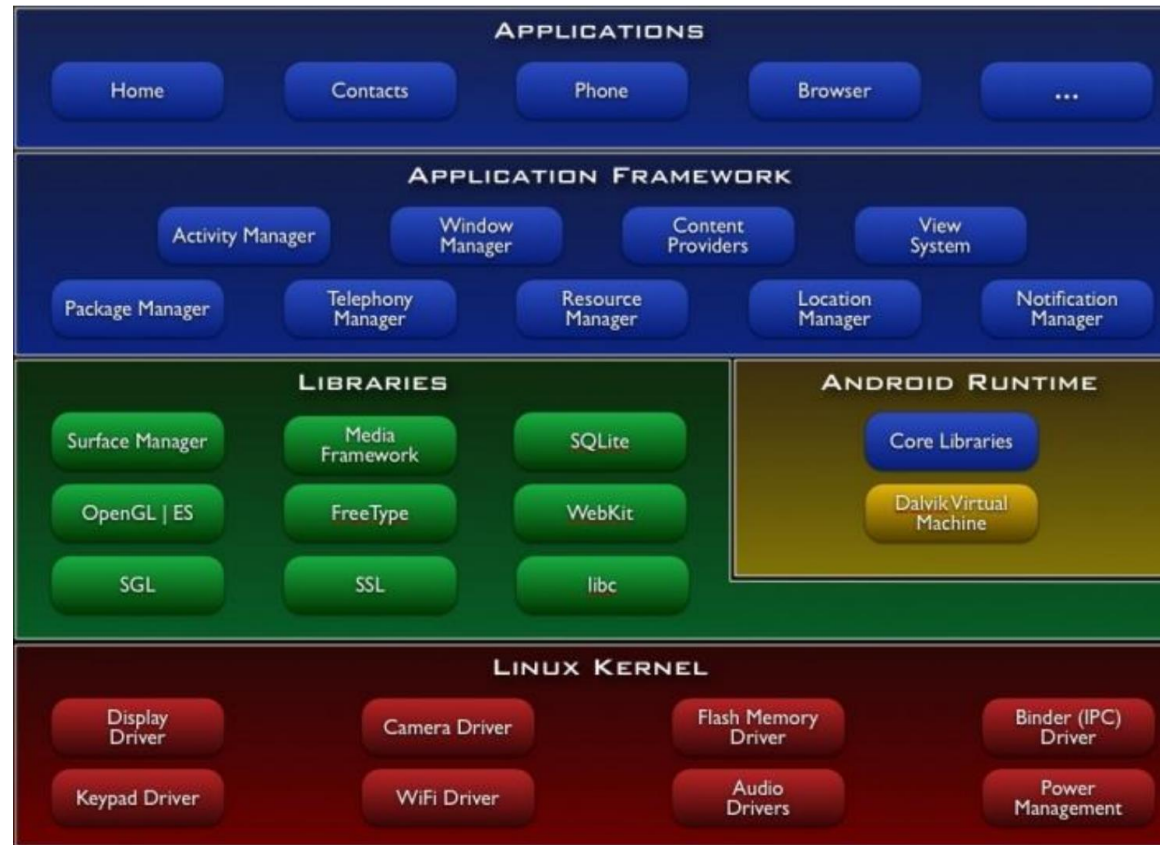
Telefonía GSM

Bluetooth, EDGE, 3G, and WiFi

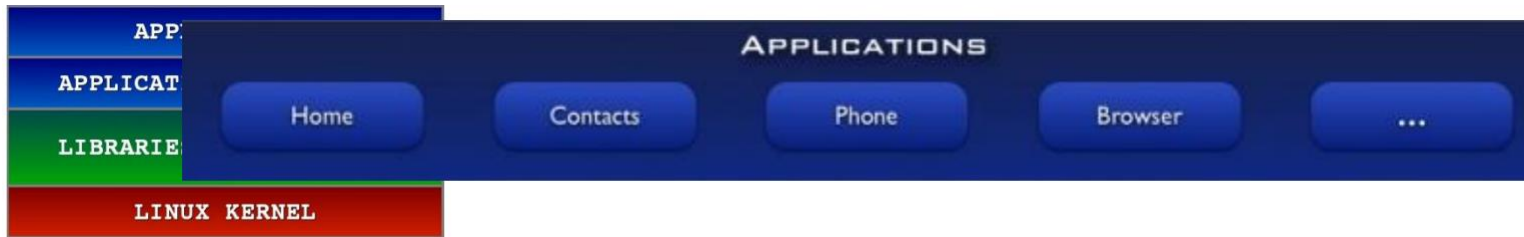
Cámara, GPS, compas, and acelerómetro

Ambiente rico de desarrollo.

Arquitectura de Android



Applications



- Android proporciona un conjunto de aplicaciones principales:
 - ✓ Email Client
 - ✓ SMS Program
 - ✓ Calendar
 - ✓ Maps
 - ✓ Browser
 - ✓ Contacts
 - ✓ et.
- Todas las aplicaciones están escritas usando el lenguaje Java

Application Framework



Permitir y simplificar la reutilización de componentes.

- desarrolladores tienen acceso completo al mismo marco API utilizadas por las aplicaciones base.
- Los usuarios pueden reemplazar componentes.

Application Framework

Elemento	Rol
View System	Usado para construir una aplicación, incluyendo lists, grids, text boxes, buttons, y web browser
Content Provider	Lo que permite aplicaciones para acceder a datos de otras aplicaciones o compartir sus propios datos
Resource Manager	Acceso a los recursos que no son de código (cadenas localizadas, gráficos y archivos de diseño)
Notification Manager	Permitir todas las aplicaciones mostrar al cliente alertas en la barra de estado.
Activity Manager	Gestión del ciclo de vida de las aplicaciones y proporcionar un backstack de navegación común

Application Framework



- Incluye un conjunto de bibliotecas en C y C++ que son utilizados por los componentes del sistema.
- Expuesto a los desarrolladores a través del marco de aplicaciones para Android.

Runtime Framework



- Core Libraries

- Proporciona la mayor parte de la funcionalidad disponible en las bibliotecas base del lenguaje java.

- APIs

- Estructura de Datos
 - Utilidades
 - File Access
 - NetworkAccess
 - Graphics

Runtime

- Dalvik Virtual Machine
 - Entorno en el que cada aplicación Android corre
 - Android cada aplicación se ejecuta en su propio proceso, con su propia instancia de la VM Dalvik.
 - Dalvik ha sido escrito tal que un dispositivo puede correr múltiples máquinas virtuales eficientemente.
 - Virtual machine basada en registros

Linux Kernel



- Basado en Linux Kernel 2.6 para los servicios del sistema core
 - memoria y proceso de gestión
 - red pila
 - Basado en modelo de controlador.
 - seguridad
- Proporcionando una capa de abstracción entre el hardware y el resto de la pila de SW

Fundamentos de una app

- Apps son escritas en Java
- Contenida en: Android Asset Packaging Tool
- Cada aplicación corre en su propio proceso Linux.
- Cada proceso tiene su propia Java Virtual Machine
- Para cada aplicación es asignada un único identificación de usuario Linux
- Apps pueden compartir el mismo usuario para poder compartir archivos.

Componentes Generales de una aplicación

- Actividades
 - Es una interface visual enfocada al usuario.
 - Ejemplo: una lista de items que los usuarios pueden escoger.
- Servicios
 - Corren en “background” por tiempo indefinido.
 - Ejemplo: calcular y proveer el resultado de las actividades que son necesarias.
- Broadcast Receivers
 - Reciben y toman acción a notificaciones.
 - Ejemplo: notificación de una noticia importante.

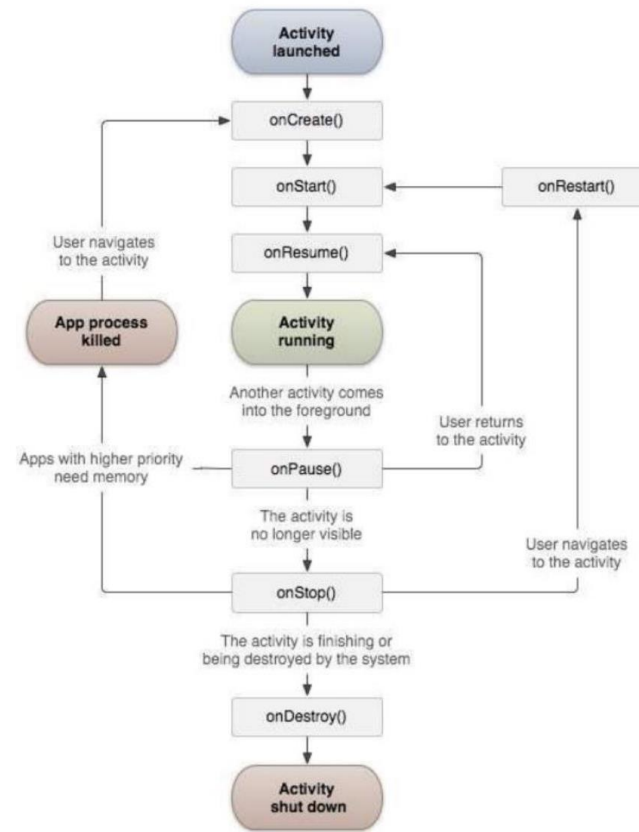
Componentes Generales de una aplicación

- Content Providers
 - Guarda y recupera información para hacerla accesible a todas las aplicaciones.
 - Ejemplo: Android provee con algunos proveedores de contenido para tipos comunes de tipos de datos. (ej., audio, video, imágenes, información personal, etc.)
- Intents
 - Mantiene el contenido de un mensaje
 - Ejemplo: actividad como agregar comentarios a una imagen presentada al usuario.

Actividad

- Una actividad representa una pantalla como una pantalla o un marco (“frame”) y esta es una subclase de la clase `ContextThemeWrapper`.
- Como en los lenguajes de programación C, C++ o Java que en su programa parte de la función `main()`. De manera muy similar, sistema Android inicia su programa con una actividad a partir de una llamada en el método `onCreate()`.
- Hay una secuencia de métodos de devolución de llamada que comienzan una actividad y una secuencia de métodos de devolución (“callback”) de llamada que quita una actividad como se muestra en el siguiente diagrama de ciclo de vida de actividad.

Ciclo de Vida de una actividad

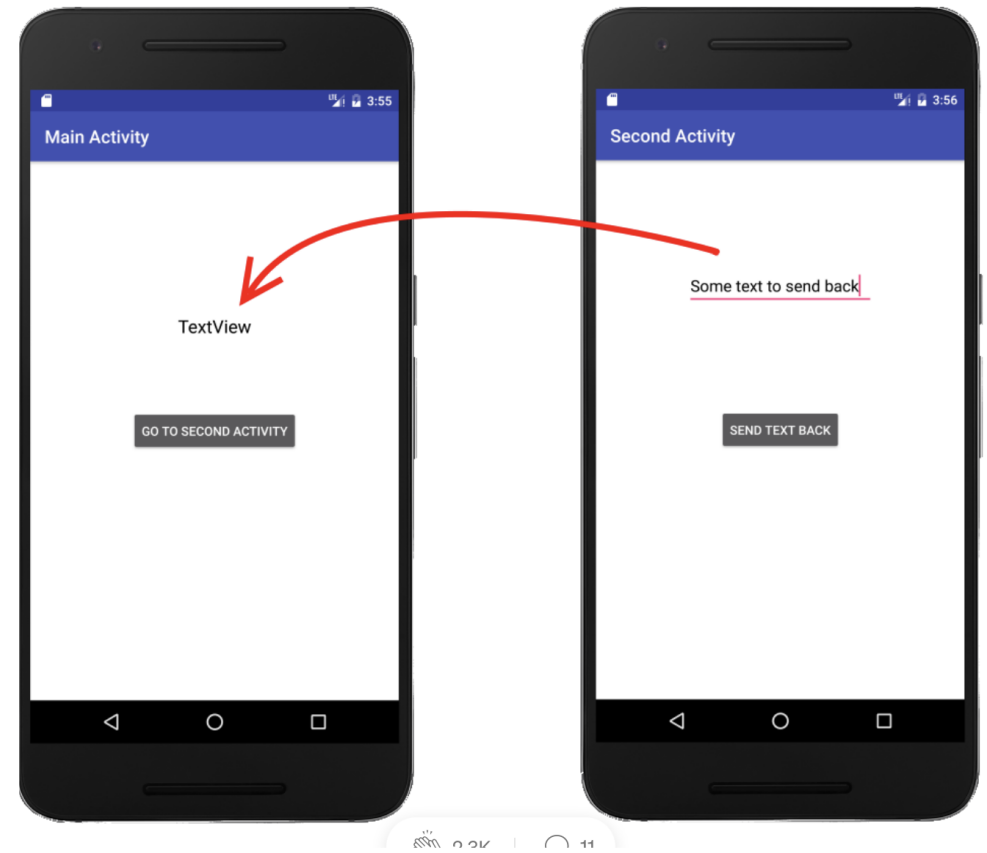


Que es una actividad en Android

Ejecuta acciones en pantalla, cualquier operación, lo haremos en la actividad.

Podemos crear múltiples actividades en un proyecto.

Se divide en la parte de interfase lógica (UI) y la parte lógica de la actividad.



Que es una API

Una API simplifica la programación del APP's usando interfases que nos ayudan a acceder los componentes necesarios que requerimos para nuestra aplicación.

Por ejemplo si queremos obtener la ubicación del dispositivo usaremos la siguiente interfase:

```
locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER)
```

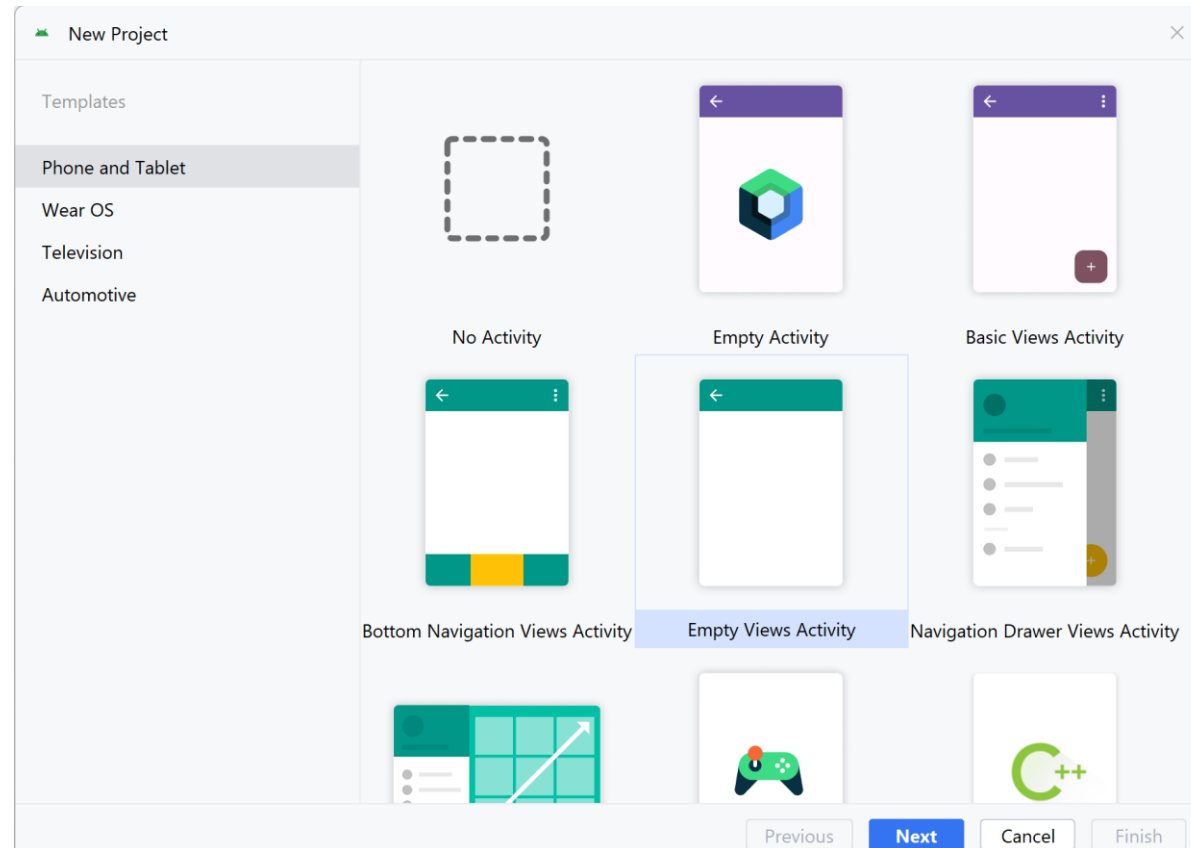
Se estiman entre 1 y 20 millones de líneas de este tipo.

Java es un lenguaje OOP

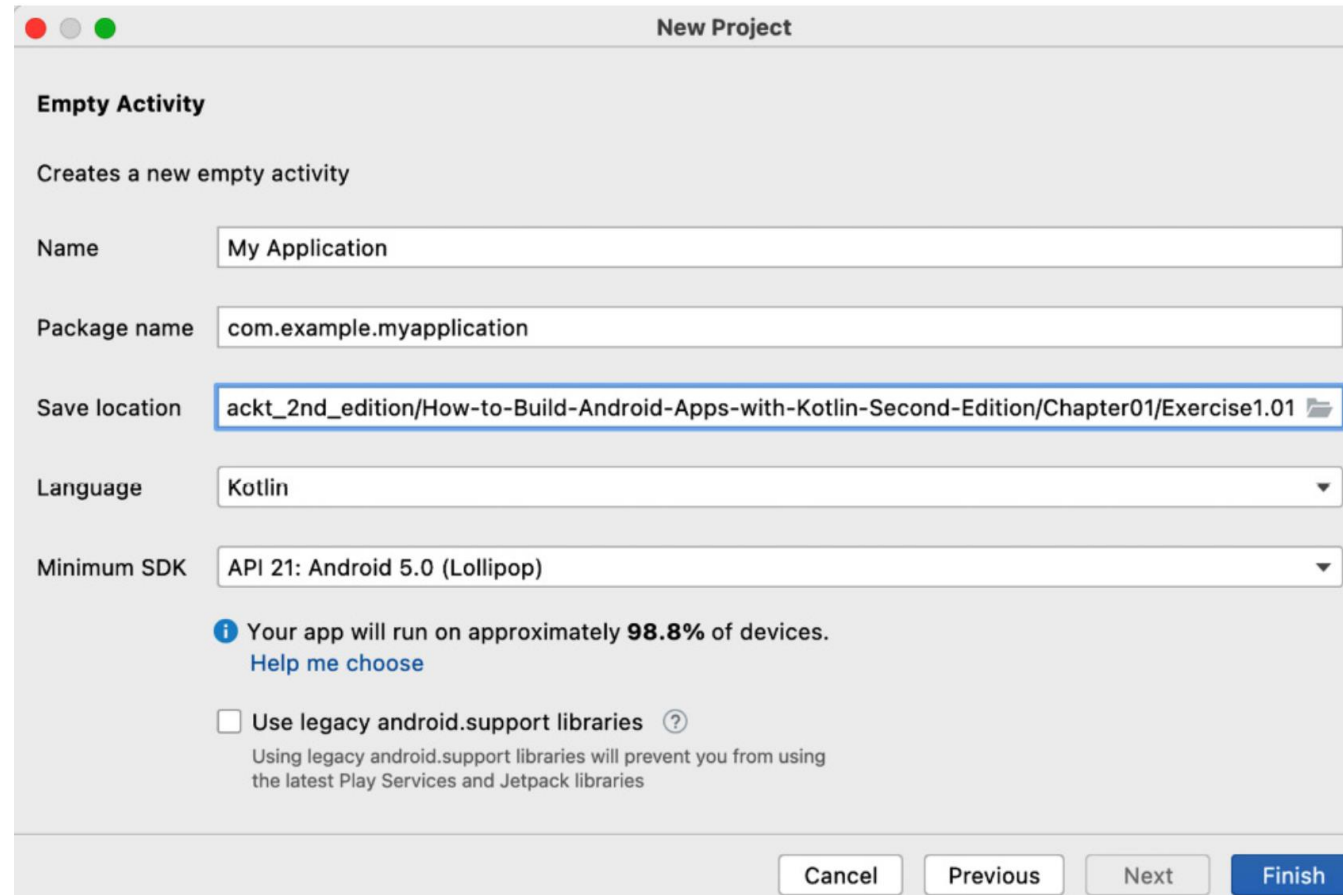
Java es un lenguaje de programación que existe desde hace mucho más tiempo que Android.

Es un lenguaje orientado a objetos. Esto significa que utiliza el concepto de objetos de programación reutilizables.

Como crear una APP



Como crear una APP



New Project

Empty Activity

Creates a new empty activity

Name: My Application

Package name: com.example.myapplication

Save location: ackt_2nd_edition/How-to-Build-Android-Apps-with-Kotlin-Second-Edition/Chapter01/Exercise1.01

Language: Kotlin

Minimum SDK: API 21: Android 5.0 (Lollipop)

i Your app will run on approximately **98.8%** of devices.
[Help me choose](#)

☐ Use legacy android.support libraries **?**
Using legacy android.support libraries will prevent you from using the latest Play Services and Jetpack libraries

Cancel Previous Next Finish

Opciones de creación de la APP

Name: Al igual que el nombre de tu proyecto de Android, este nombre aparecerá como el nombre predeterminado de tu aplicación cuando esté instalada en un teléfono y visible en Google Play.

Package name: Utiliza el patrón de nombre de dominio inverso estándar para crear un nombre. Se usará como identificador de dirección para el código fuente y los recursos de la aplicación. Lo mejor es que este nombre sea lo más claro y descriptivo posible y que esté lo más alineado posible con el propósito de la aplicación. Por lo tanto, probablemente sea mejor cambiar esto para usar uno o más subdominios (como `com.sample.shop.myshop`).

Save location: Esta es la carpeta local de su máquina donde se almacenará inicialmente la aplicación. Esto se puede cambiar en el futuro, por lo que probablemente pueda mantener el valor predeterminado o editarlo a algo diferente (como `Usuarios/MiUsuario/android/proyectos`). La ubicación predeterminada variará según el sistema operativo que esté utilizando. De forma predeterminada, el proyecto se guardará en una nueva carpeta con el nombre de la aplicación sin espacios.

Opciones de creación de la APP

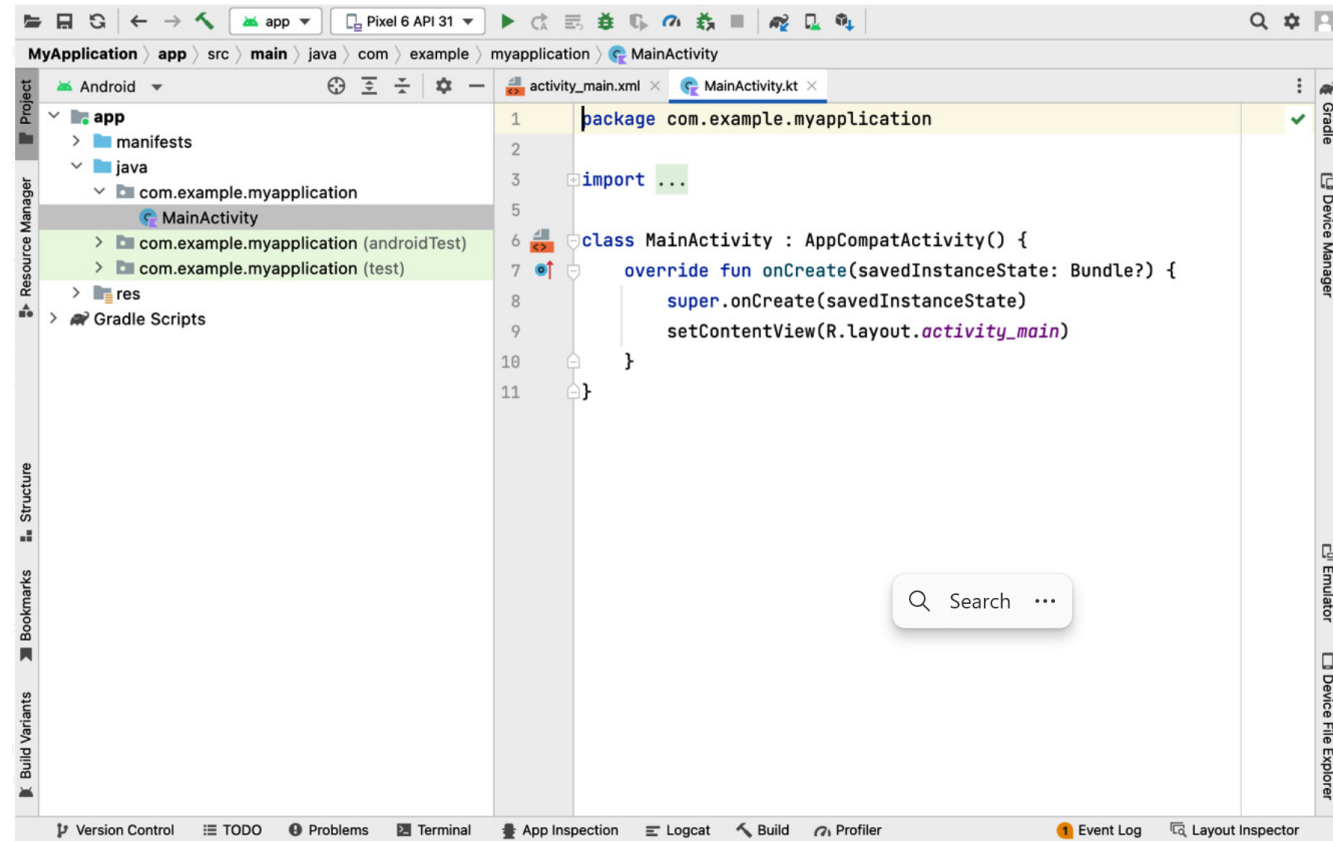
Language: Kotlin es el lenguaje preferido de Google para el desarrollo de aplicaciones Android. Como se explico usaremos Java y en excepciones usaremos Kotlin.

Minimum SDK: Según la versión de Android Studio que descargues, el valor predeterminado puede ser el mismo que se muestra en la en la lamina anterior o una versión diferente. Por favor mantén esto igual. La mayoría de las nuevas funciones de Android son compatibles con versiones anteriores, por lo que su aplicación funcionará bien en la gran mayoría de los dispositivos más antiguos. Sin embargo, si deseas dirigirte a dispositivos más nuevos, debería considerar aumentar el nivel mínimo de API. Hay un enlace [Help me Choose](#) a un cuadro de diálogo que explica el conjunto de funciones al que tiene acceso con vistas al desarrollo en diferentes versiones de Android y el porcentaje actual de dispositivos en todo el mundo que ejecutan cada versión de Android.

Opciones de creación de la APP

Use legacy android.support libraries: Dejar esto sin marcar. Utilizará las bibliotecas de AndroidX, que son el reemplazo de las bibliotecas de compatibilidad que se diseñaron para hacer que las funciones de las versiones más recientes de Android sean compatibles con versiones anteriores, pero proporciona mucho más que esto. También contiene nuevos componentes de Android llamados Jetpack, que, como su nombre indica, impulsa el desarrollo de Android y proporciona una gran cantidad de funciones enriquecidas que querrá usar en su aplicación, simplificando así las operaciones comunes.

Como crear una APP



Interfase de desarrollo

Una vez que haya completado todos estos detalles, seleccione Finish. Su proyecto se construirá y luego se le presentará la pantalla mostrada en la lamina anterior o similar. Puede ver inmediatamente la actividad que se ha creado (MainActivity) en una pestaña y el diseño utilizado para la pantalla en la otra pestaña (activity_main.xml). Las carpetas de la estructura de la aplicación se encuentran en el panel Izquierdo.

Interfase de desarrollo

Una vez que haya completado todos estos detalles, seleccione Finish. Su proyecto se construirá y luego se le presentará la pantalla mostrada en la lamina anterior o similar. Puede ver inmediatamente la actividad que se ha creado (MainActivity) en una pestaña y el diseño utilizado para la pantalla en la otra pestaña (activity_main.xml). Las carpetas de la estructura de la aplicación se encuentran en el panel Izquierdo.

Creando tu dispositivo virtual

Como parte de la instalación de Android Studio, al descargar e instalar los componentes más recientes del kit de desarrollo de software (SDK) de Android.

Estos deben incluir un emulador base, que configurará para crear un dispositivo virtual en el que ejecutar aplicaciones de Android.

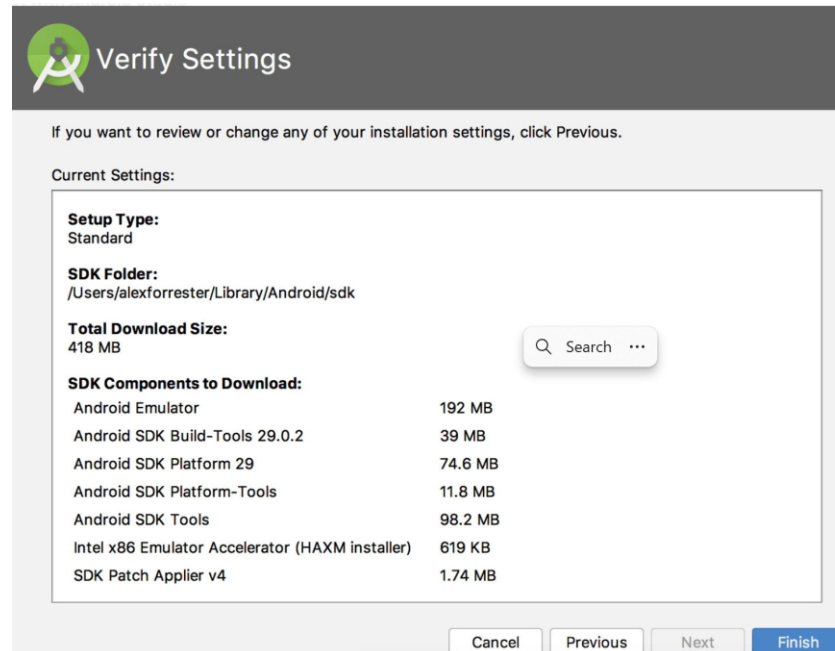
Un emulador imita las características y la configuración de hardware y software de un dispositivo real.

La ventaja es que puede realizar cambios y verlos rápidamente en el escritorio mientras desarrolla la aplicación.

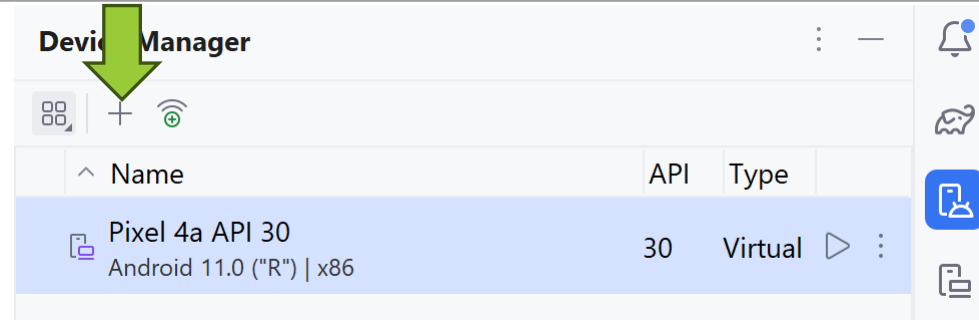
Aunque los dispositivos virtuales no tienen todas las características de un dispositivo real, el ciclo de retroalimentación suele ser más rápido que seguir los pasos para conectar un dispositivo real.

Creando tu dispositivo virtual

Además, aunque debes asegurar de la aplicación se ejecute según lo esperado en diferentes dispositivos, puedes estandarizarla dirigiéndose a un dispositivo específico mediante la descarga de un perfil de dispositivo, incluso si no tienes un dispositivo real si este es un requisito de tu proyecto. La pantalla que habrás visto (o algo similar) al instalar Android Studio es la siguiente:

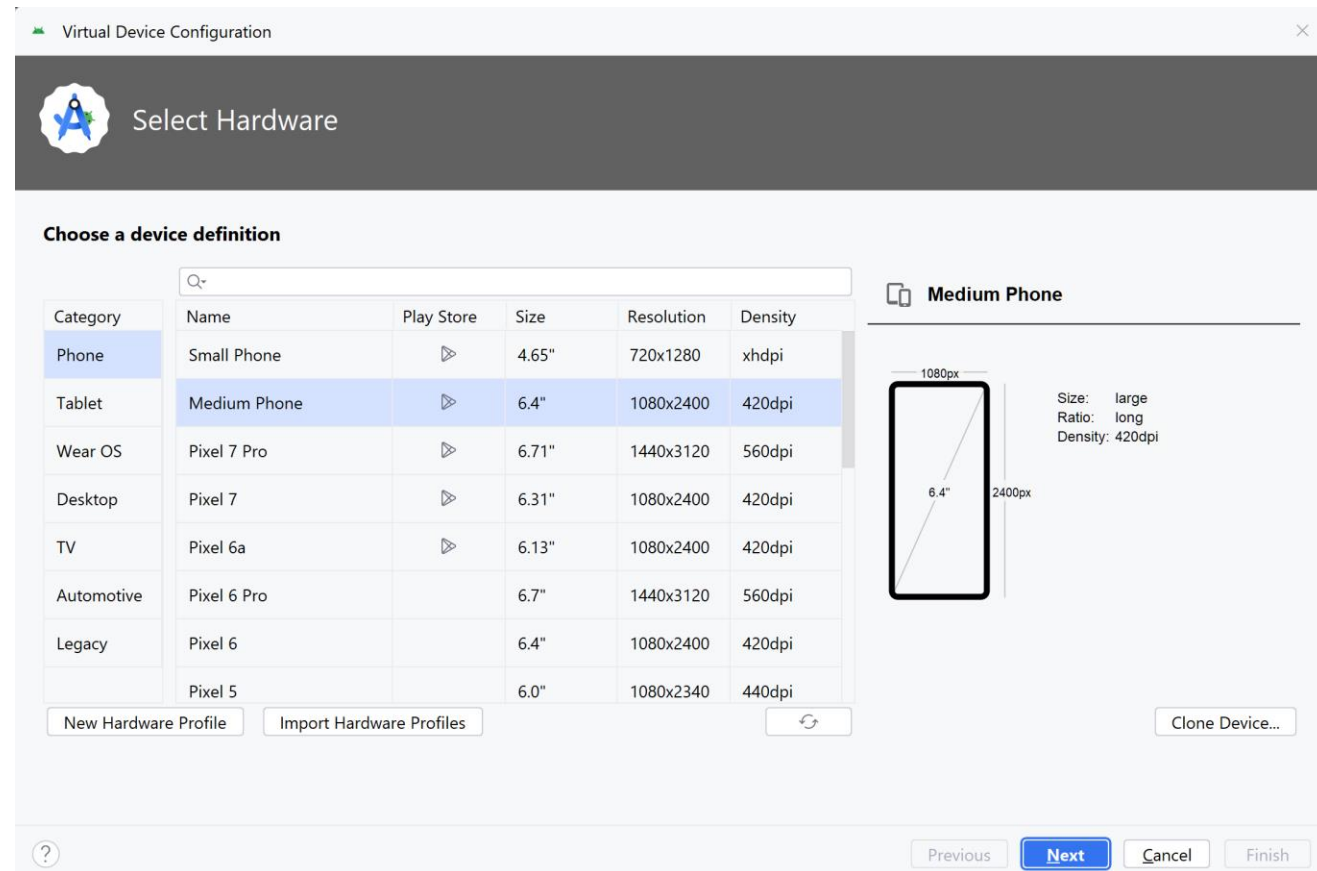


Creando tu dispositivo virtual



Seleccione el icono mostrado en la figura arriba y aparecerá el submenú de Device Manager. Utiliza el símbolo + para agregar un nuevo dispositivo virtual.

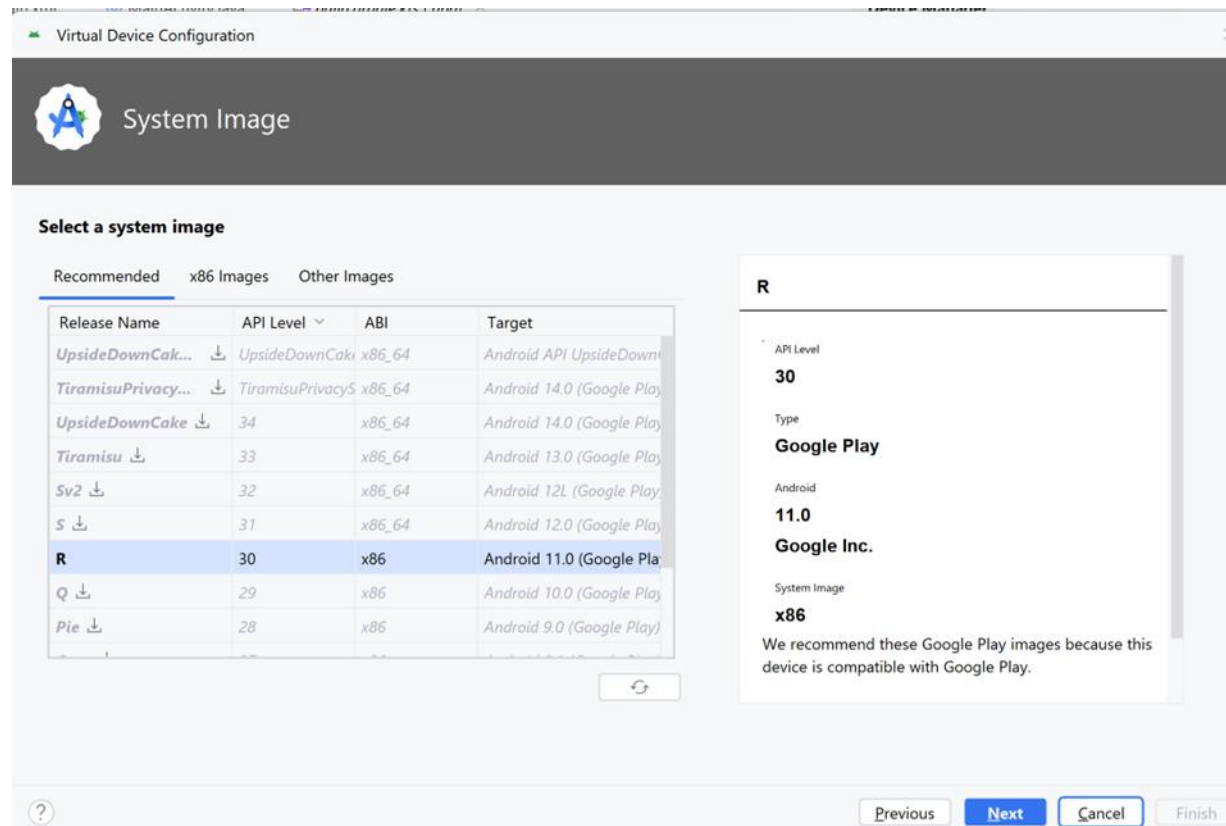
Seleccionando HW



Seleccionando HW

Seleccione Pixel 6, o en su defecto el Pixel 4^a, la razón de usar un pixel es que al ser desarrollado por Google tiene acceso a las últimas actualizaciones de la plataforma Android. Seleccionado el dispositivo aparecerá la siguiente pantalla.

Seleccionando HW



Seleccionando HW

El nombre de Tirimasu que se muestra aquí es el código o el nombre de la versión inicial de Android 13.


Seleccione esta imagen del sistema.

La columna Target también puede mostrar (Google Play) o (API de Google) en el nombre. Las API de Google significan que la imagen del sistema viene preinstalada con Google Play Services.

Se trata de un amplio conjunto de funciones de las API de Google y las aplicaciones de Google que tu aplicación puede utilizar y con las que puede interactuar.

Al ejecutar la aplicación por primera vez, verá aplicaciones como Maps y Chrome en lugar de una imagen de emulador simple. Una imagen del sistema de Google Play significa que, además de las API de Google, también se instalará la aplicación Google Play.


Android Virtual Device

 Android Virtual Device (AVD)

Verify Configuration


AVD Name

Pixel 7 API 30

 Pixel 7

6.31 1080x2400 420dpi


Change...


 R

Android 11.0 x86

Change...

Startup orientation

 Portrait

 Landscape

Emulated Performance

Graphics: Automatic

Show Advanced Settings

AVD Name

The name of this AVD.

?

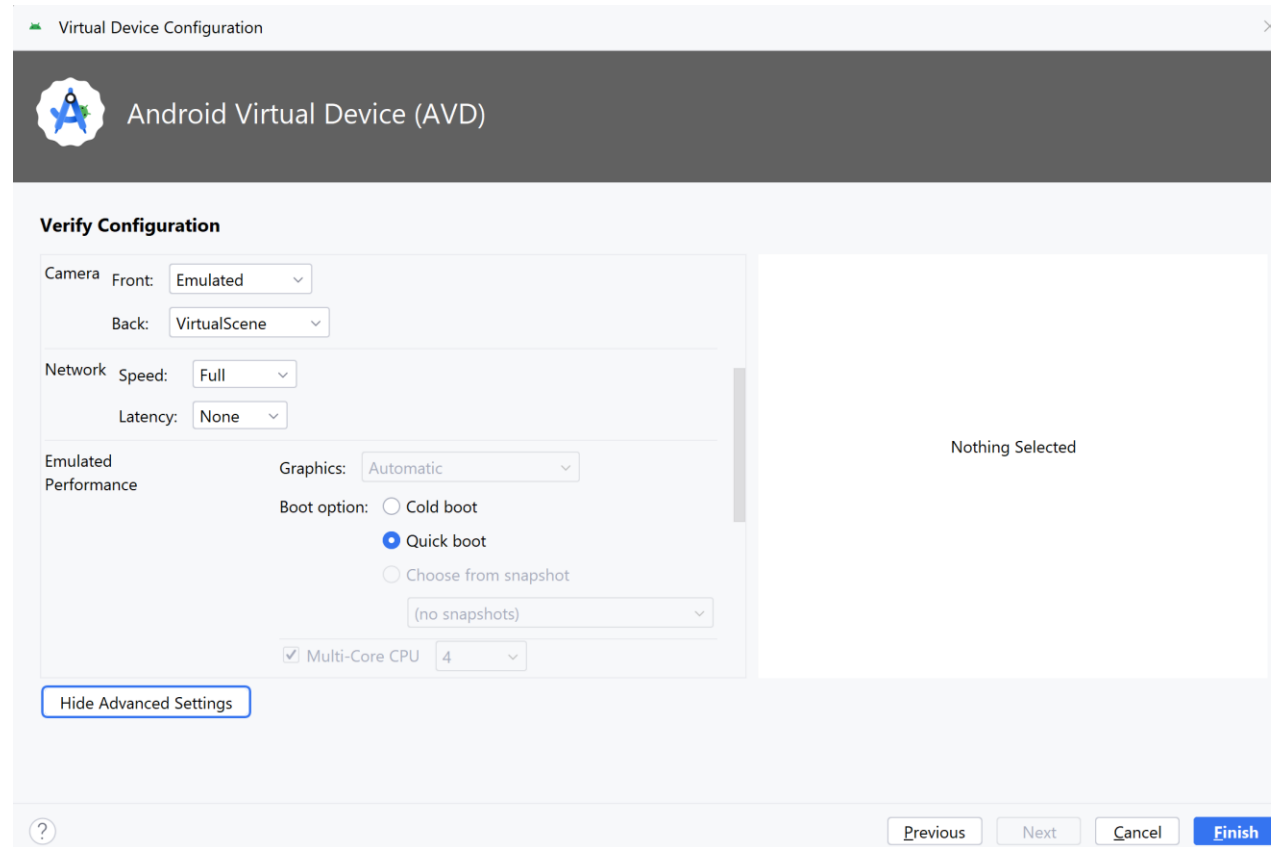
Previous

Next

Cancel

Finish

Android Virtual Device



Android Manifest

La aplicación que acabas de crear, aunque simple, abarca los bloques de construcción básicos que usarás en todos los proyectos que crees. La aplicación se controla desde el archivo AndroidManifest.xml, un archivo de manifiesto que detalla el contenido de la aplicación. Se encuentra en la aplicación | Manifiestos | AndroidManifest.xml

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

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportRtl="true"
        android:theme="@style/Theme.MyApplication"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

Android Manifest

Un archivo de manifiesto típico, en términos generales, es un archivo de nivel superior que describe los archivos adjuntos u otros datos y metadatos asociados que forman un grupo o unidad.

El manifest de Android aplica este concepto a tu app de Android como un archivo XML. Cada aplicación de Android tiene una clase de aplicación que le permite configurar la aplicación. Una vez que `<application>` se abre el elemento, se definen los componentes de la aplicación. Como acabamos de crear nuestra app, solo contiene la primera pantalla que se muestra en el siguiente código:`</application>`

```
<activity android:name=".MainActivity">
```

Que es metada?

Básicamente es una opción adicional para almacenar información a la que se puede acceder a través de todo el proyecto. En este caso, `<meta-data>` se define la etiqueta externa `<activity>` y la `<application>` etiqueta interna.`</application></activity></meta-data>`

Android Manifest

El siguiente sub-componente en el XML, se especifica como sigue:

`<intent-filter>`

Android usa intents como un mecanismo para interactuar con las apps y los componentes del sistema.

Los intents se envían y el filtro de intents registra la capacidad de tu app para reaccionar a estos intents. `<android.intent.action.MAIN>` es el punto de entrada principal a la aplicación, que, tal y como aparece en el XML adjunto de . MainActivity, especifica que esta pantalla se iniciará cuando se inicie la aplicación.

`Android.intent.category.LAUNCHER` indica que tu app aparecerá en el selector del dispositivo de tu usuario.`</android.intent.action.MAIN>`

Android Manifest

Como ha creado la aplicación a partir de una plantilla, tiene un manifiesto básico que iniciará la aplicación y mostrará una pantalla inicial al inicio a través de un componente de actividad.

Según las otras funciones que quieras agregar a tu app, es posible que debas agregar permisos en el archivo de manifiesto de Android. Los permisos se agrupan en tres categorías diferentes: normales, de firma y peligrosos:

Normal: Estos permisos incluyen el acceso al estado de la red, Wi-Fi, Internet y Bluetooth. Por lo general, se permiten sin solicitar el consentimiento del usuario en tiempo de ejecución.

Signature: Estos permisos son compartidos por el mismo grupo de aplicaciones que deben estar firmadas con el mismo certificado. Esto significa que estas aplicaciones pueden compartir datos libremente, pero otras aplicaciones no pueden acceder.

Dangerous: Estos permisos se centran en el usuario y su privacidad, como el envío de SMS, el acceso a cuentas y ubicación, y la lectura y escritura en el sistema de archivos y contactos.

Android Manifest

Estos permisos deben aparecer en el manifiesto y, en el caso de permisos peligrosos, a partir de la API de Android Marshmallow 23 (Android 6 Marshmallow) en adelante, también debes pedirle al usuario que otorgue los permisos en tiempo de ejecución.

La documentación detallada sobre este archivo se puede encontrar en:

<https://developer.android.com/guide/topics/manifest/manifest-intro>.

Gradle

En el transcurso de la creación de este proyecto, ha utilizado principalmente el SDK de la plataforma Android. Las bibliotecas de Android necesarias se descargaron cuando instalaste Android Studio.

Sin embargo, estas no son las únicas bibliotecas que se usan para crear la aplicación. Para configurar y compilar tu proyecto o app de Android, se usa una herramienta de compilación llamada Gradle.

Gradle es una herramienta de compilación multipropósito que Android Studio usa para compilar tu app. De forma predeterminada, Android Studio usa Groovy, un lenguaje de máquina virtual Java (JVM) de tipo dinámico, para configurar el proceso de compilación y permite una fácil administración de dependencias para que puedas agregar bibliotecas a tu proyecto y especificar las versiones.

Gradle

Cuando creas tu app por primera vez, hay dos archivos build.gradle, uno en el nivel raíz o superior del proyecto y otro específico para tu app en la carpeta del módulo de la app. Si puedes ver ambas fueron construidas en Kotlin, antes se usaba un sw llamado Groovy

Versión Kotlin

```
plugins {  
    id("com.android.application") version "8.2.1" apply false  
    id("org.jetbrains.kotlin.android") version "1.9.22" apply false  
}
```

Versión Java

```
// Top-level build file where you can add configuration options common to all sub-projects/modules.  
plugins {  
    id("com.android.application") version "8.2.1" apply false  
}
```

Gradle

Gradle funciona en un sistema de complementos, por lo que puedes escribir tu propio complemento que realice una tarea o una serie de tareas y conectarlo a tu canalización de compilación. Los tres complementos enumerados anteriormente hacen lo siguiente:

com.android.application: Esto agrega compatibilidad para crear una aplicación de Android.

com.android.library: Esto permite que los subproyectos/módulos sean bibliotecas de Android.

org.jetbrains.kotlin.android: Esto proporciona integración y compatibilidad con el lenguaje para Kotlin en el proyecto

Gradle app level

La app build.gradle es específica para la configuración de tu proyecto:

```
plugins {
    id("com.android.application")
}

android {
    namespace = "com.example.myapplication"
    compileSdk = 34

    defaultConfig {
        applicationId = "com.example.myapplication"
        minSdk = 24
        targetSdk = 34
        versionCode = 1
        versionName = "1.0"

        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            isMinifyEnabled = false
            proguardFiles(
                getDefaultProguardFile("proguard-android-optimize.txt"),
                "proguard-rules.pro"
            )
        }
    }
    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_1_8
        targetCompatibility = JavaVersion.VERSION_1_8
    }
}

dependencies {
    implementation("androidx.appcompat:appcompat:1.6.1")
    implementation("com.google.android.material:material:1.11.0")
    implementation("androidx.constraintlayout:constraintlayout:2.1.4")
    testImplementation("junit:junit:4.13.2")
    androidTestImplementation("androidx.test.ext:junit:1.1.5")
    androidTestImplementation("androidx.test.espresso:espresso-core:3.5.1")
}
```

Gradle app Level

Los complementos para Android y Kotlin, que se detallan en el archivo raíz `build.gradle`, se aplican a tu proyecto aquí por ID en las líneas de complementos.

El bloque `android`, proporcionado por el complemento `com.android.application`, es donde configuras los ajustes de configuración específicos de Android:

namespace: Esto se establece a partir del nombre del paquete que especificó al crear el proyecto. Se utilizará para generar identificadores de compilación y recursos.

compileSdk: Esto se usa para definir el nivel de API con el que se ha compilado la aplicación, y la aplicación puede usar las características de esta API y versiones anteriores.

defaultConfig: Esta es la configuración base de la aplicación.

applicationId: Se establece en el paquete de la aplicación y es el identificador de la aplicación que se usa en Google Play para identificar de forma única la aplicación. Se puede cambiar para que sea diferente del nombre del paquete si es necesario.

Gradle app Level

El bloque android, proporcionado por el complemento com.android.application, es donde configuras los ajustes de configuración específicos de Android:

minSdk: Este es el nivel mínimo de API que admite tu aplicación. De este modo, se evitará que tu aplicación se muestre en Google Play para dispositivos con versiones inferiores a esta.

targetSdk: Este es el nivel de API al que te diriges. Este es el nivel de API que está diseñado para funcionar la aplicación compilada y con el que se ha probado.

versionCode: Esto especifica el código de versión de la aplicación. Cada vez que es necesario realizar una actualización en la aplicación, el código de versión debe aumentarse en uno o más.

versionName: Un nombre de versión fácil de usar que normalmente sigue el control de versiones semántico de X.Y.Z, donde X es la versión principal, Y es la versión secundaria y Z es la versión de revisión, por ejemplo, 1.0.3.

testInstrumentationRunner: Este es el ejecutor de pruebas que se usará para las pruebas de interfaz de usuario.

Gradle app Level

El bloque android, proporcionado por el complemento com.android.application, es donde configuras los ajustes de configuración específicos de Android:

buildTypes: En buildTypes, se agrega una versión que configura la aplicación para crear una compilación de versión.

El valor minifyEnabled, si se establece en true, reducirá el tamaño de la aplicación al quitar el código no utilizado y ofuscar la aplicación.

Este paso de ofuscación cambia el nombre de las referencias del código fuente a valores como a.b.c(). Esto hace que el código sea menos propenso a la ingeniería inversa y reduce aún más el tamaño de la aplicación compilada.

compileOptions: Este es el nivel de lenguaje del código fuente de Java (sourceCompatibility) y el código de bytes (targetCompatibility).

kotlinOptions: Esta es la biblioteca jvm que debe usar el complemento de Gradle de kotlin.

Gradle app Level

El bloque de dependencias especifica las bibliotecas que usa tu app sobre el SDK de la plataforma Android, como se muestra aquí (con comentarios agregados):

```
dependencies {  
  
    implementation("androidx.appcompat:appcompat:1.6.1")  
    implementation("com.google.android.material:material:1.11.0")  
    implementation("androidx.constraintlayout:constraintlayout:2.1.4")  
    testImplementation("junit:junit:4.13.2")  
    androidTestImplementation("androidx.test.ext:junit:1.1.5")  
    androidTestImplementation("androidx.test.espresso:espresso-core:3.5.1")  
}
```

Diferencias en Kotlin vs Android

KOTLIN es un lenguaje de programación multiplataforma, de tipos estáticos, de propósito general con inferencia de tipos. KOTLIN está diseñado para interoperar completamente con Java, pero la inferencia de tipos permite que su sintaxis sea más concisa. KOTLIN está patrocinado por JetBrains y Google a través de la Fundación Kotlin.

JAVA es un lenguaje de programación orientado a objetos desarrollado por JAMES GOSLING y sus colegas de SUN MICRO SYSTEMS en 1991. El lenguaje se llamó inicialmente OAK. Fue desarrollado como un lenguaje de programación completo en el que se pueden realizar los mismos tipos de tareas y resolver problemas similares que se pueden hacer en otros lenguajes de programación como BASIC, C ++, etc.

La razón más importante para introducir el avance de Kotlin en Android fue disminuir la cantidad de líneas de código y hacer que el desarrollo sea más conveniente. Todo lo que se puede hacer con Java se puede hacer con Kotlin para el desarrollo de Android.

Estructura de una aplicación de Android

KOTLIN es un lenguaje de programación multiplataforma, de tipos estáticos, de propósito general con inferencia de tipos. KOTLIN está diseñado para interoperar completamente con Java, pero la inferencia de tipos permite que su sintaxis sea más concisa. KOTLIN está patrocinado por JetBrains y Google a través de la Fundación Kotlin.

JAVA es un lenguaje de programación orientado a objetos desarrollado por JAMES GOSLING y sus colegas de SUN MICRO SYSTEMS en 1991. El lenguaje se llamó inicialmente OAK. Fue desarrollado como un lenguaje de programación completo en el que se pueden realizar los mismos tipos de tareas y resolver problemas similares que se pueden hacer en otros lenguajes de programación como BASIC, C ++, etc.

La razón más importante para introducir el avance de Kotlin en Android fue disminuir la cantidad de líneas de código y hacer que el desarrollo sea más conveniente. Todo lo que se puede hacer con Java se puede hacer con Kotlin para el desarrollo de Android.