**Asignatura**: Desarrollo de Aplicaciones Móviles Nativas.

**Tema**: Instalación de Android Studio.

**Introducción**.

Para iniciar las actividades con Android Studio se puede acceder al sitio siguiente:

https://developer.android.com/codelabs/basic-android-kotlin-compose-install-android-studio?hl=es-419#2

A través del enlace anterior se puede descargar, iniciar la instalación y configuración de la última versión de Android Studio que se denomina Narwhal 2025.1.2. Android Studio es el entorno de desarrollo integrado (IDE) oficial que se usa en el desarrollo de apps para Android. Basado en las herramientas de IntelliJ IDEA y que ofrece funciones que mejoran la productividad de las apps para Android.

Incluye las siguientes características:

Compilación flexible basado en Gradle. Emulador con varias funciones y de dispositivos físicos, en tiempo real. Integración con GitHub y plantillas de código para compilar las funciones de apps comunes e importar código muestra. Herramientas de Lint para identificar problemas de rendimiento, usabilidad y compatibilidad de versiones. Compatibilidad con C++ y NDK. Compatibilidad con Google Cloud Platform, que se integra con Google Cloud Messaging y App Engine.

Cada proyecto de Android Studio incluye uno o más módulos con archivos de código fuente y archivos de recursos. Cada módulo de app contiene las siguientes carpetas:

manifests: El archivo AndroidManifest.xml.

java: Los archivos de código fuente de Java y Kotlin, incluido el código de prueba JUnit.

res: Los recursos sin código, cadenas de IU e imágenes de mapa de bits.

Los archivos de compilación de Android Studio se denominan build.gradle.kts si se utiliza Kotlin, o build.gradle si se utiliza Groovy. Son archivos de texto sin formato que usan la sintaxis de Kotlin o Groovy para configurar la compilación con elementos que proporciona el complemento de Android para Gradle.

Cada proyecto tiene un archivo de compilación de nivel superior para todo el proyecto y archivos de compilación de nivel de módulo independientes para cada módulo. Para habilitar la depuración intercalada, en la ventana Debug, seleccionar Settings y Show Variable Values in Editor.

Los archivos ejecutables de Android se denominan archivos APK (Android Package Kit, paquete de aplicaciones de Android). Un archivo APK es un formato de archivo comprimido que contiene todos los elementos necesarios (código, recursos y certificados) para que una aplicación se instale y funcione correctamente en un dispositivo Android.

Cuando se compila un programa, Android Studio ejecutan automáticamente inspecciones de lint y otras inspecciones del IDE para identificar y corregir problemas con la calidad estructural del código. La herramienta lint comprueba los archivos de origen del proyecto de Android en busca de posibles errores y realizar mejoras de precisión, seguridad, rendimiento, usabilidad, accesibilidad e internacionalización.

Los archivos principales de un proyecto de Android son los siguientes:

* Archivos Java o Kotlin, con la lógica principal de la clase MainActivity.java con el código de la actividad principal.
* AndroidManifest.xml que define el núcleo de la aplicación e incluye permisos, como acceso a internet o a la cámara, e cuál es la actividad principal que se lanzará al abrir la aplicación.
* Archivos de recursos (res) con todos los elementos de la aplicación como imágenes, iconos o imágenes de la interfaz.
* Plantillas que definen la estructura visual de pantallas o cadenas de texto para interfaces de usuario, menús, etc.
* Bibliotecas de terceros o bibliotecas nativas escritas en lenguajes como C/C++.

Para ejecutar una app de Android en un emulador, se crea un Dispositivo Virtual de Android (AVD) en Android Studio, seleccionarlo en el menú de dispositivos de destino y digitar el botón Run. Para ejecutarla en un dispositivo real, primero se habilitan las Opciones de desarrollador y de depuración USB en el dispositivo físico, y conectarlo a la computadora con cable USB. En Android Studio, seleccionar el dispositivo físico en el menú de destino y digitar en Run para que la aplicación se compile e instale en él.

**Instalación.**

En el sitio indicado al inicio, descargar el IDE de Android Studio. Ejecutar el instalador digitando el icono de la figura 1:

**Figura 1**. Icono del instalador de Android Studio. **Figura 2**. Inicio de la instalación de la versión Narwhal.

Una vez que se inicia la instalación se muestra la figura 2. Enseguida, se sigue la secuencia de instalación con las siguientes figuras correspondientes, digitando en Next o Install, según sea el caso.

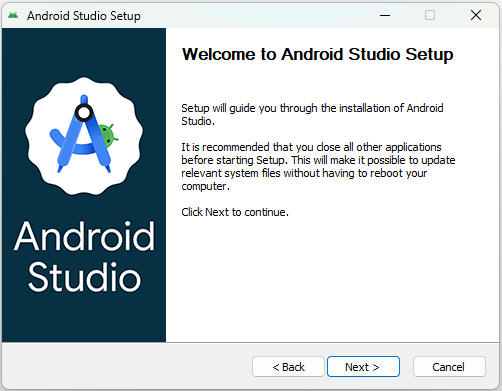
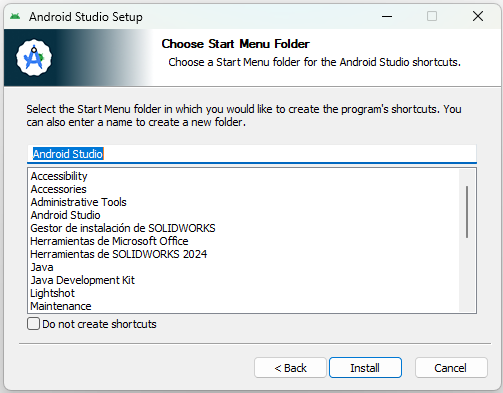
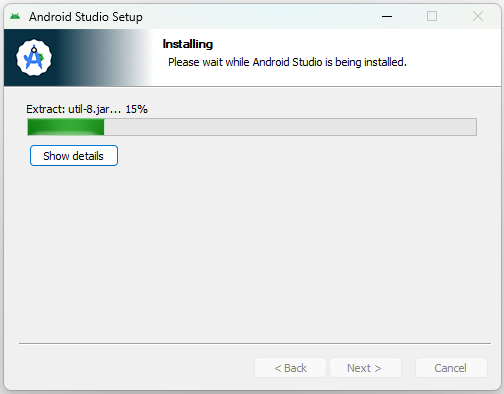
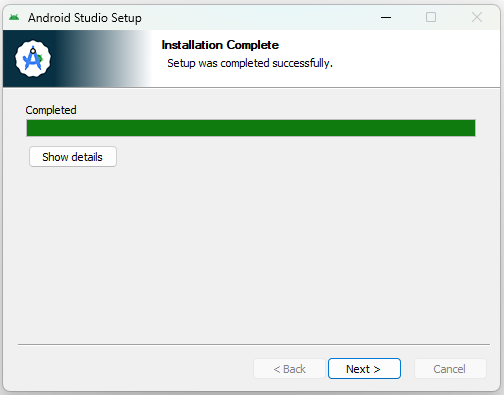
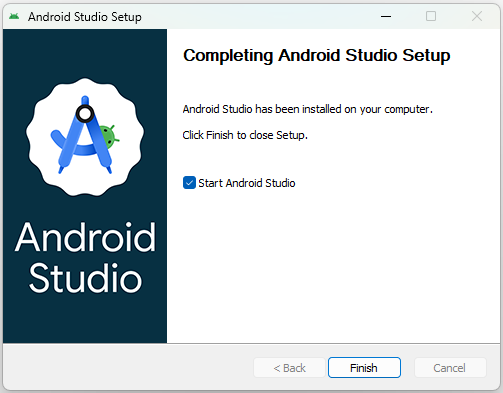
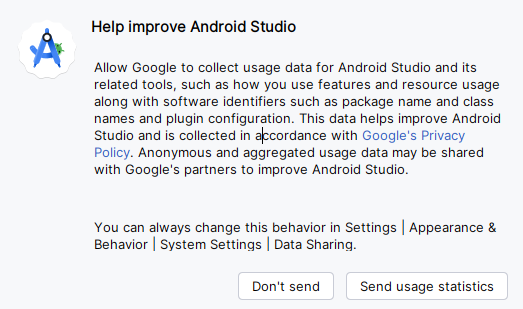
 

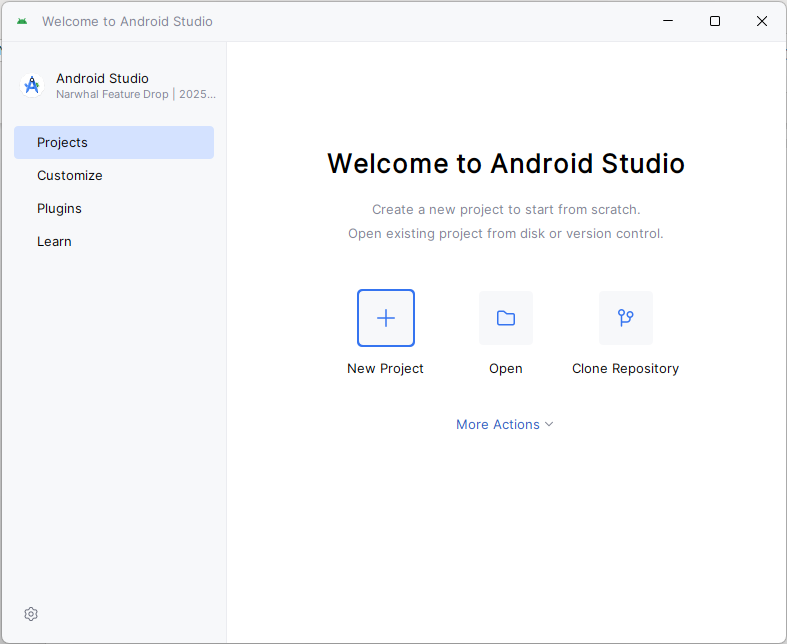
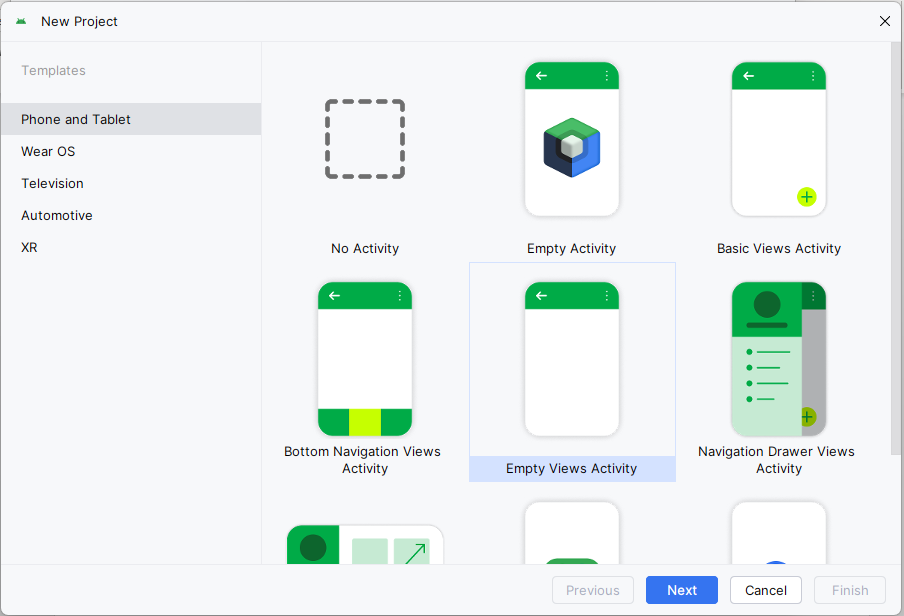
Figura 3. Bienvenida a la instalación. Figura 4. Selección de la carpeta de instalación.

**Figura 5**. Inicio de la descarga de los archivos. **Figura 6**. Fin de la descarga de archivos de instalación.

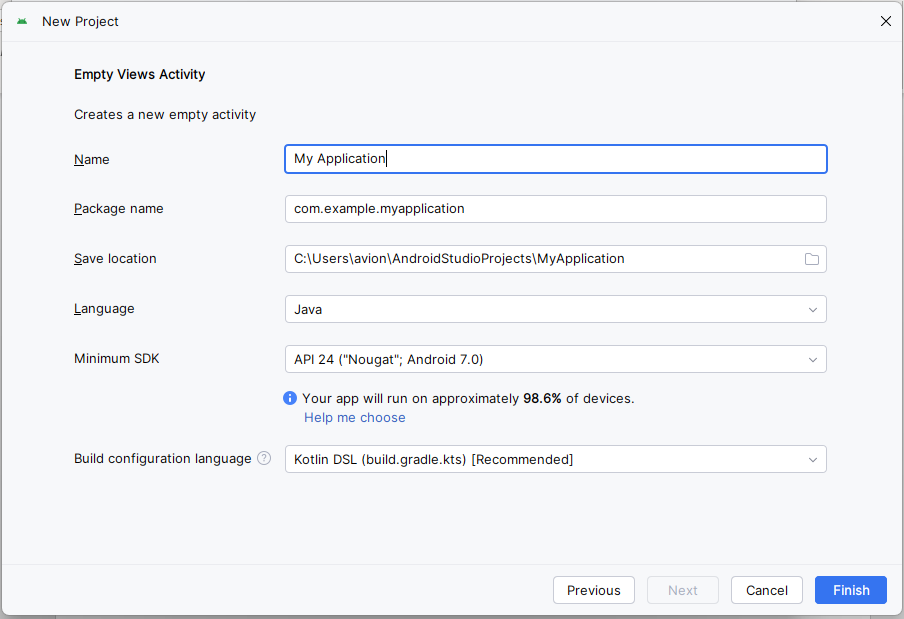
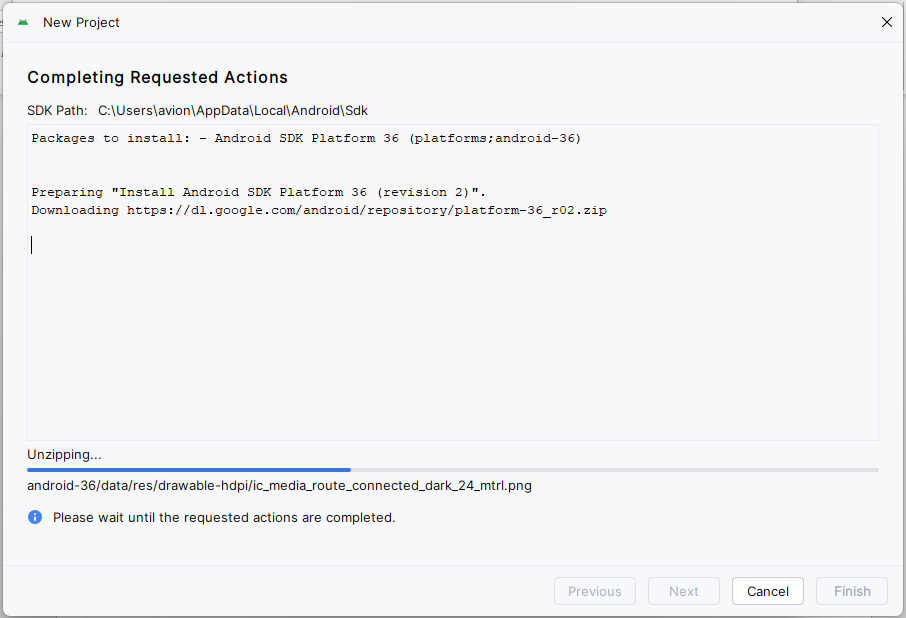
**Figura 7**. Inicio del IDE. **Figura 8**. Envío de notificaciones. Seleccionar Don’t send.

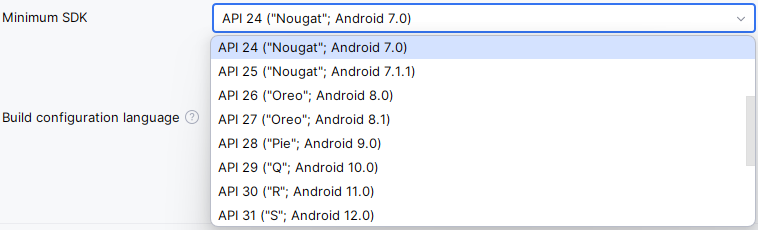
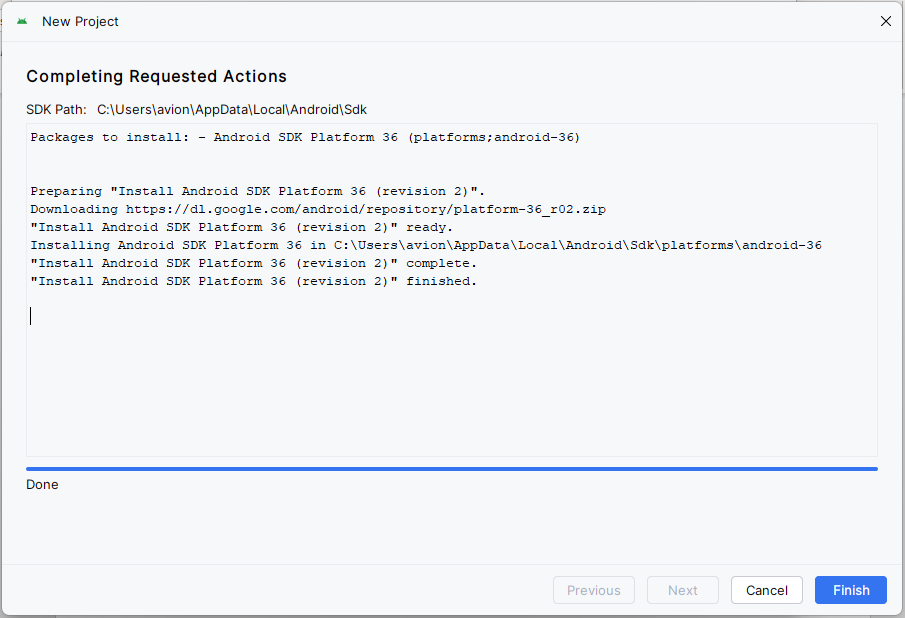
**Figura 9**. Pantalla de inicio de Android Studio. **Figura 10**. Seleccionar la opción Empty Views Activity.

Para configurar el proyecto, el nombre del proyecto se puede asignar con un nuevo nombre.

* Seleccionar el lenguaje de programación, en este caso Java.
* En la opción Mínimum SDK selecciona la API deseada, en este caso la API 24 de Nougat, verificando la lista desplegable, como se indica en la figura 13.
* Digitar el botón Help me choose para mostrar el porcentaje de dispositivos que podrán ejecutarse la app con esa API.
* Por último, seleccionar Finish para terminar la configuración del proyecto My Application.

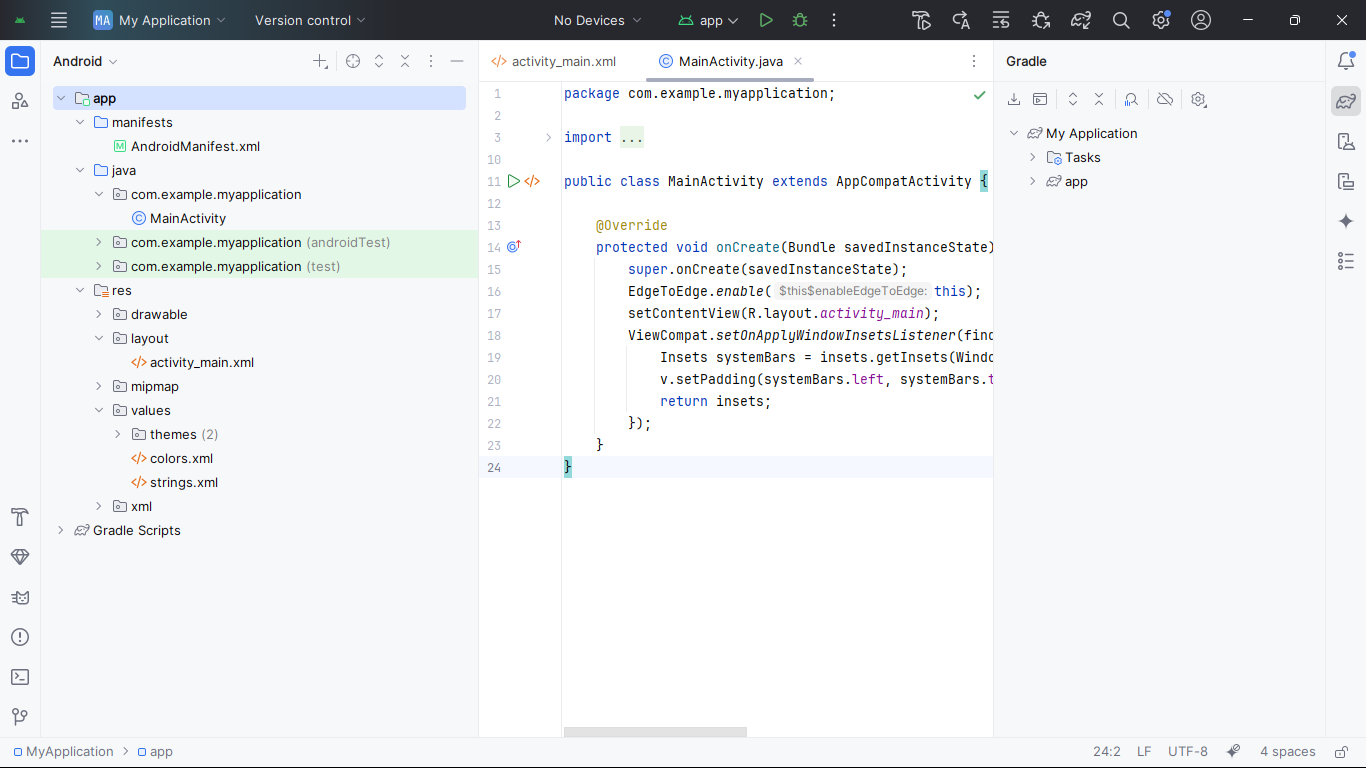
 

**Figura 11**. Ventana de configuración de proyecto. **Figura 12**. Descarga de bibliotecas de configuración.

**Figura 13**. Selección de una API. **Figura 14**. Terminación de la configuración del proyecto.

Enseguida, se muestra el ambiente de desarrollo del IDE de Android Studiio



**Figura 15**. La ventana principal del IDE Android Studio.

En la ventana principal se muestran los menús principales.

* A la izquierda de la región central se muestra el repositorio con las carpetas principales y los archivos del desarrollo.
* En la región central se muestra el editor para los archivos de los lenguajes Java, Kotlin y XML en particular.
* En la región derecha se muestran diferentes herramientas, en particular las opciones de los emuladores.

En el editor se muestra el código Java de la clase MainActivity.java y el código del archivo XML activity\_mainl.xml predeterminados, como se indica enseguida.

El archivo MainActivity.java:

package com.example.myapplication;

import android.os.Bundle;

import androidx.activity.EdgeToEdge;

import androidx.appcompat.app.AppCompatActivity;

import androidx.core.graphics.Insets;

import androidx.core.view.ViewCompat;

import androidx.core.view.WindowInsetsCompat;

public class MainActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

EdgeToEdge.enable(this);

setContentView(R.layout.activity\_main);

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {

Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());

v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);

return insets;

});

}

}

El archivo activity\_main.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>

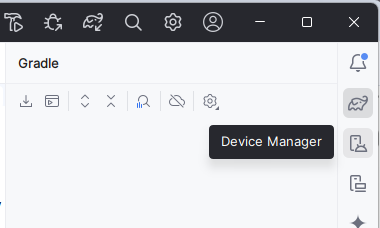
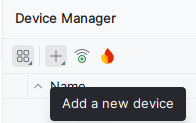
Por último, si se ha configurado un emulador o un dispositivo real, se puede ejecutar este ejemplo predeterminado.

**Configuración de un emulador o dispositivo virtual.**

Para configurar un dispositivo virtual (Virtual Device) se puede realizar mediante las siguientes opciones:

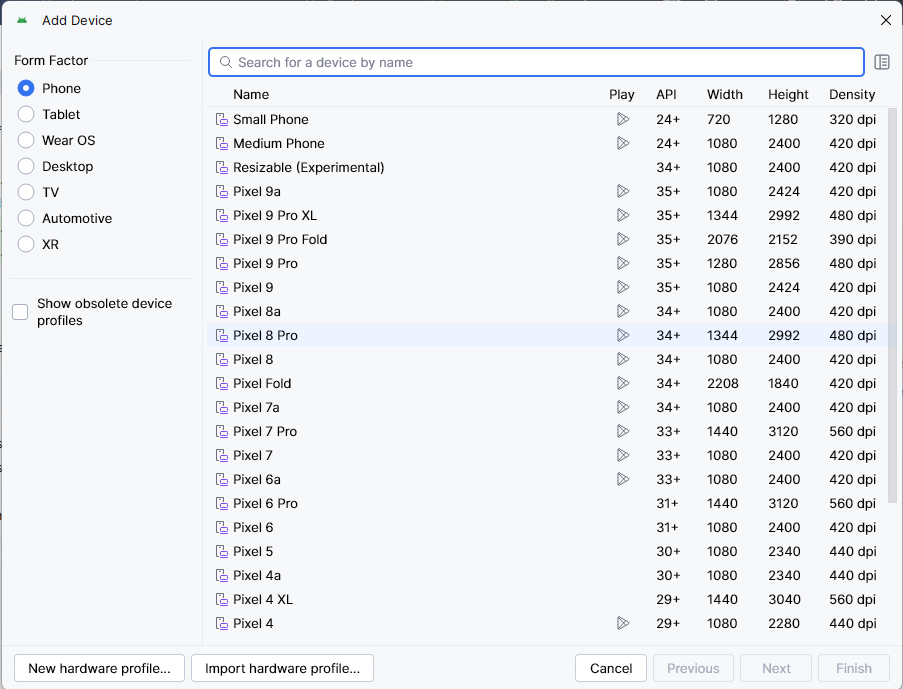
Seleccionar **View > Tool Windows > Device Manager** en la barra de menú principal. luego digitar en el botón con el símbolo **+**, y después en **Create Virtual Device**.

O también, seleccionar lo indicado en las siguientes figuras 16, 17 o 18.

**Figura 16**, En el menú lateral. **Figura 17**. Botón **+** y Add a new device**. Figura 18**. Add a new device.

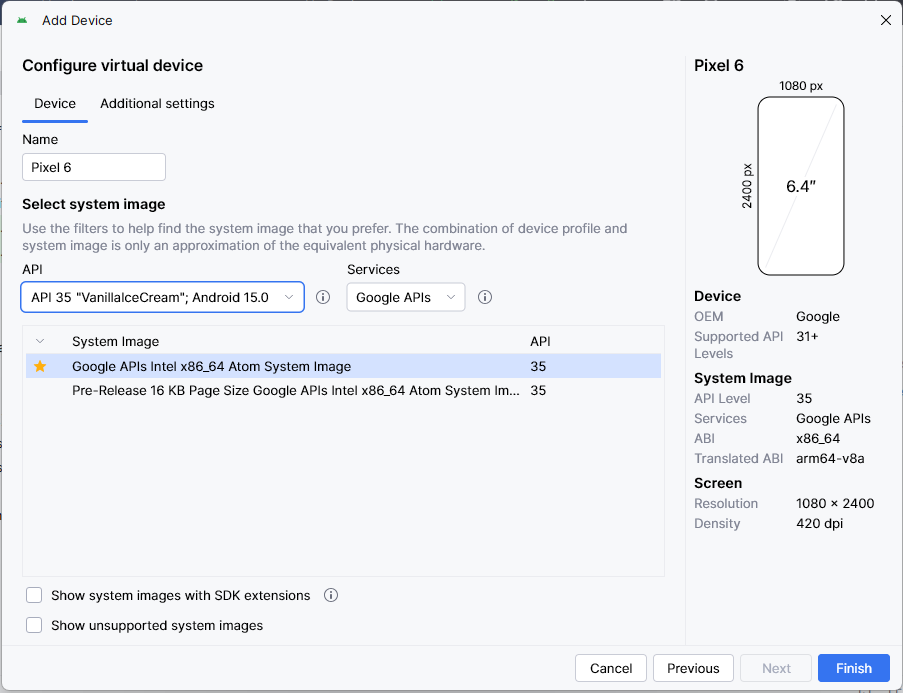
Enseguida, se muestra la ventana Add Device con los dispositivos virtuales disponibles predeterminados en Android Studio, como se indica en la figura 19.



**Figura 19**. Dispositivos virtuales disponibles en Android Studio.

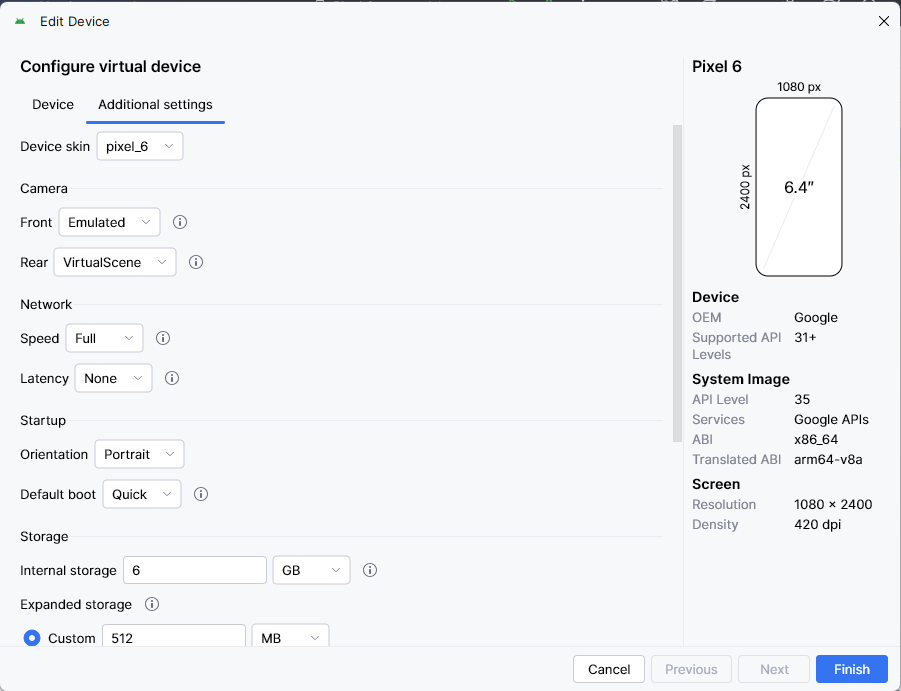
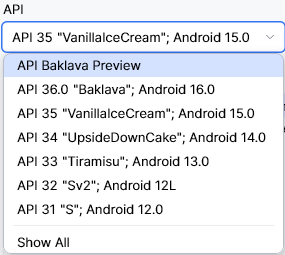
En la columna Form Factor, seleccionar la opción Phone. En la lista Name de dispositivos, seleccionar Pixel 6 y enseguida el botón Next.

La ventana se actualiza con la imagen siguiente para la configuración del dispositivo seleccionado, como se indica en la figura 20.



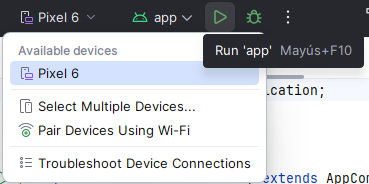
**Figura 20**. Configuración del dispositivo virtual.

En el campo Name se puede cambiar el nombre del dispositivo. En la opción API se puede cambiar la selección, por ejemplo, la opción API 35 VanillaIceCream, como se indica en la figura 22. Se puede seleccionar la pestaña Additional settings para verificar las características predeterminadas del dispositivo seleccionado, como se indica en la figura 21. Para terminar, seleccionar Finish.

**Figura 21**. La pestaña Additional settings. **Figura 22**. La API 35 VanillaIceCream.

Para ejecutar la aplicación, en el menú principal seleccionar en la lista el dispositivo configurado Pixel 6. Enseguida digitar el botón de flecha verde Run ‘app’ o Mayús+F10, para iniciar la ejecución, como se indica en la figura 23.



**Figura 23** La selección del dispositivo Pixel 6 y la opción Run ‘app’.

Para el llevar a cabo la ejecución del proyecto, primero se compila, luego se instala y por último, si no muestra algún error, se muestra la pantalla principal de la aplicación. Esto puede llevar a cabo algunos breves minutos si la computadora posee el suficiente espacio de memoria, como se indica en la figura 24.



Figura 24. La compilación de la aplicación y la espera para mostrar el dispositivo virtual.

En las figuras 25, 26 y 27 se muestra la secuencia de ejecución y las pantallas del dispositivo móvil. La pantalla principal muestra la etiqueta central con el mensaje Hola ESCOM!, el cual se cambió por Hello World!. La cadena textual se modificó en el archivo activity\_main.xml en la línea indicada del código siguiente:

<TextView

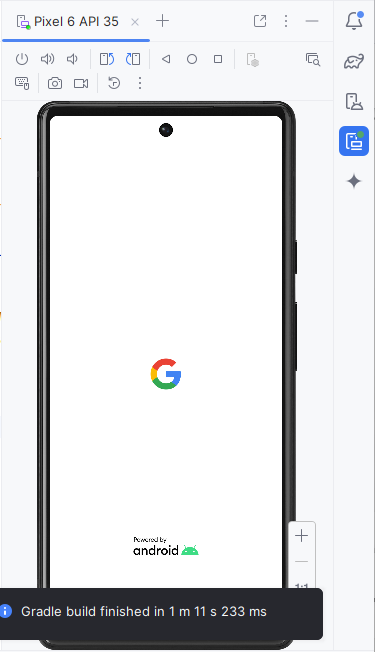
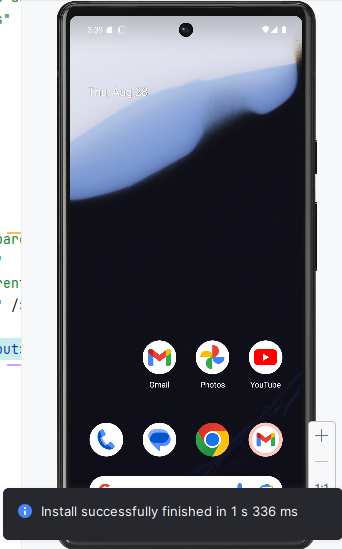
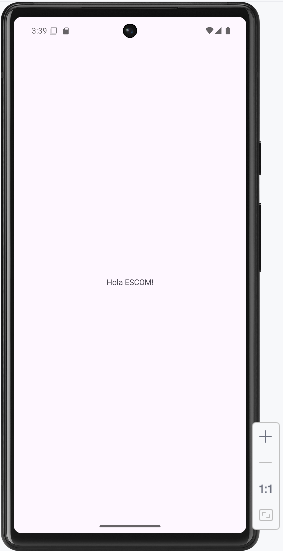
android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="**Hola ESCOM!**"

:

\>

**Figura 25**. Termina la compilación. **Figura 26**. Termina la instalación. **Figura 27**. Hola ESCOM!.

**Nota**. Cualquier duda acerca del presente documento comentarlo con el profesor.