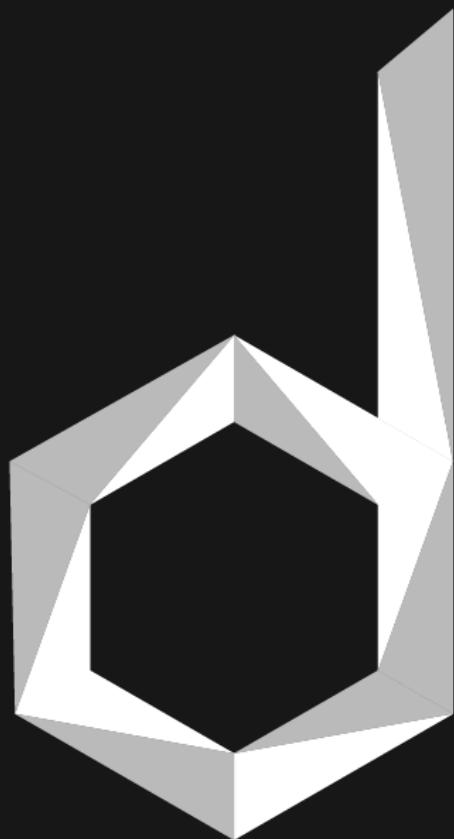


INGENIERÍA MECATRÓNICA



DI_CERO

DIEGO CERVANTES RODRÍGUEZ

DESARROLLO MÓVIL - ANDROID

ANDROID STUDIO

Kotlin: Android Studio

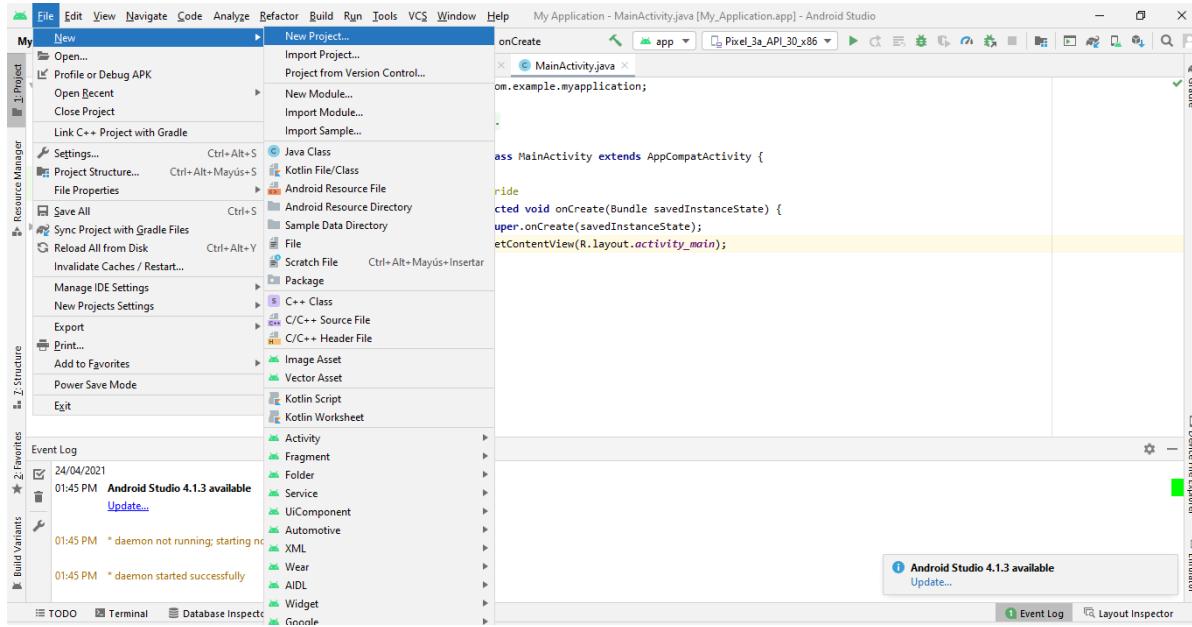
Contenido

Nuevo Proyecto en Android Studio	2
Android Studio – AVD y Modo desarrollador	5
Android Studio - Carpetas del proyecto.....	13
Estructura de Compilación - Proyecto Android Studio	20
Componentes de una Aplicación Android	22
Requisitos para Crear una Aplicación Android	22
Referencias:	26



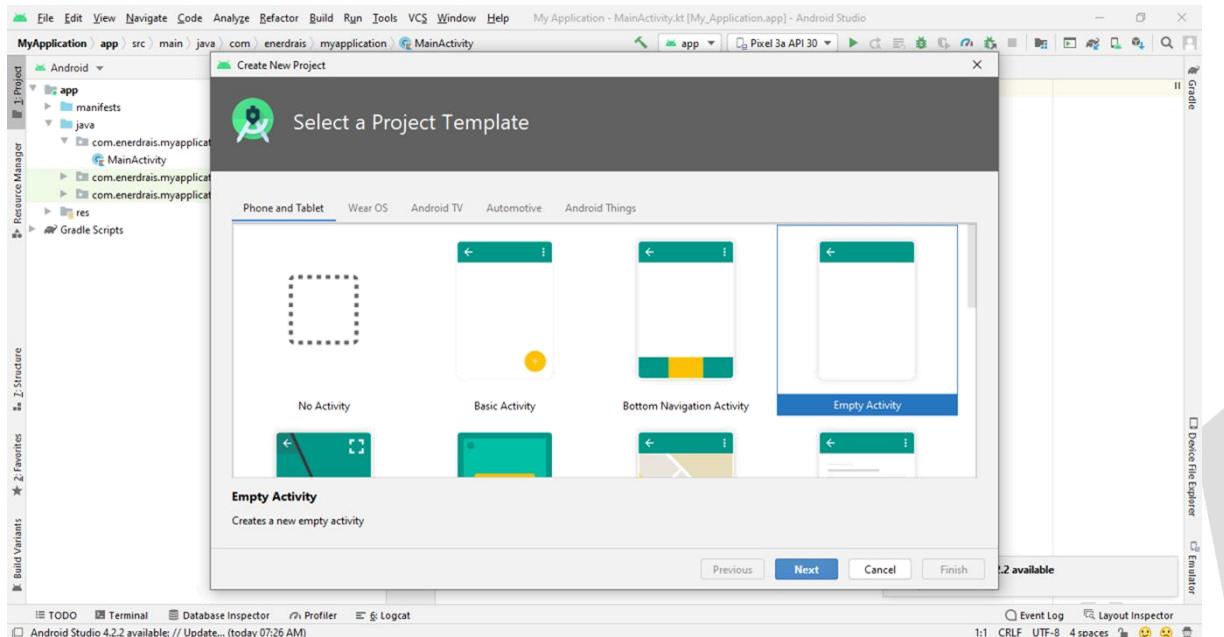
Nuevo Proyecto en Android Studio

Para crear un nuevo proyecto nos introducimos en la opción de File → New → New Project....

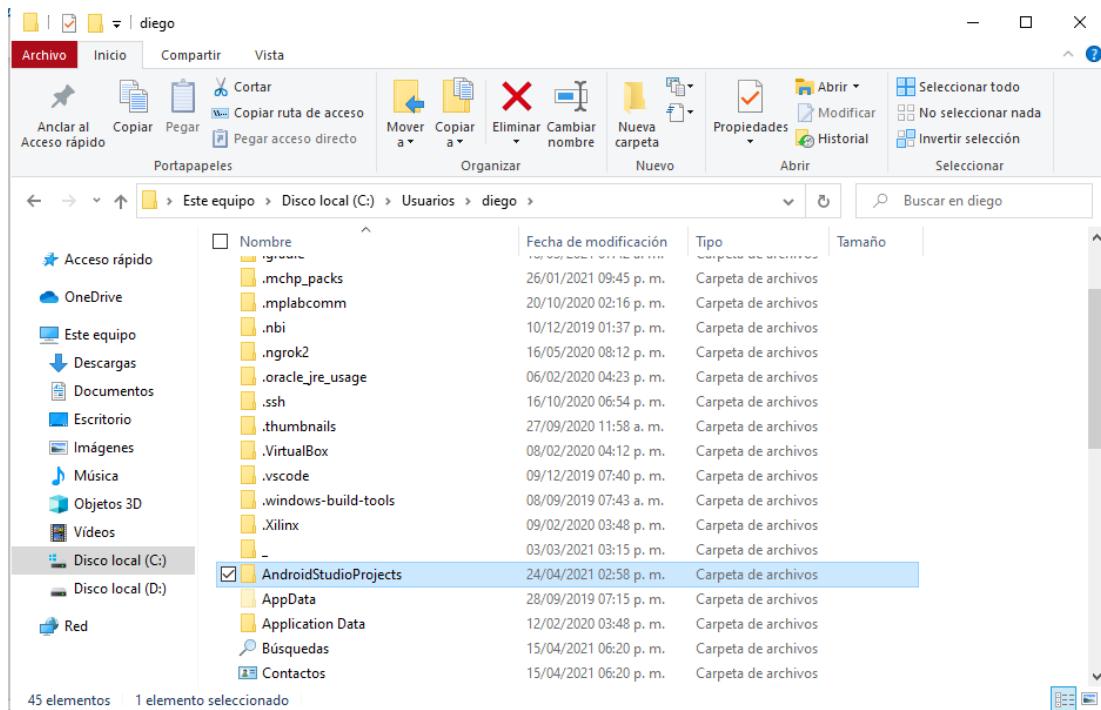


En esta parte se debe elegir qué tipo de proyecto se quiere crear, uno con un menú integrado, uno vacío, uno con el mapa de Google Maps, etc.

En este caso se elige uno vacío (Empty Activity) para diseñar desde cero la interfaz.

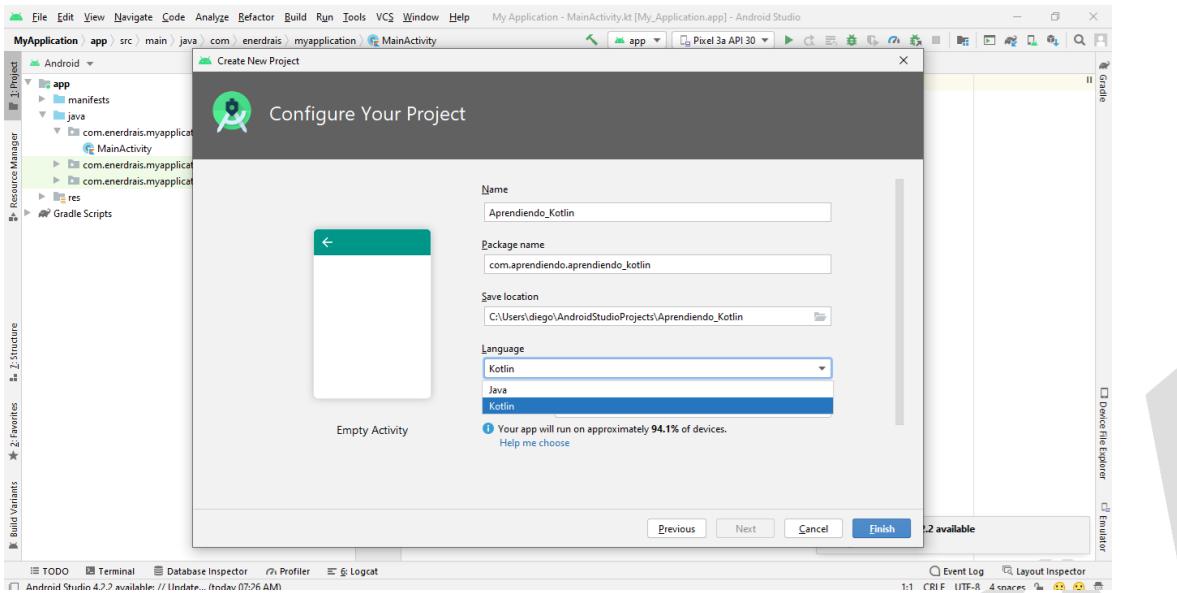


El proyecto se debe guardar en el directorio de C:\Users\diego\AndroidStudioProjects, donde se debe crear una carpeta nueva específica para el nuevo proyecto creado de Android. La carpeta se debe crear manualmente, se creará por sí sola cuando se haya creado el proyecto.

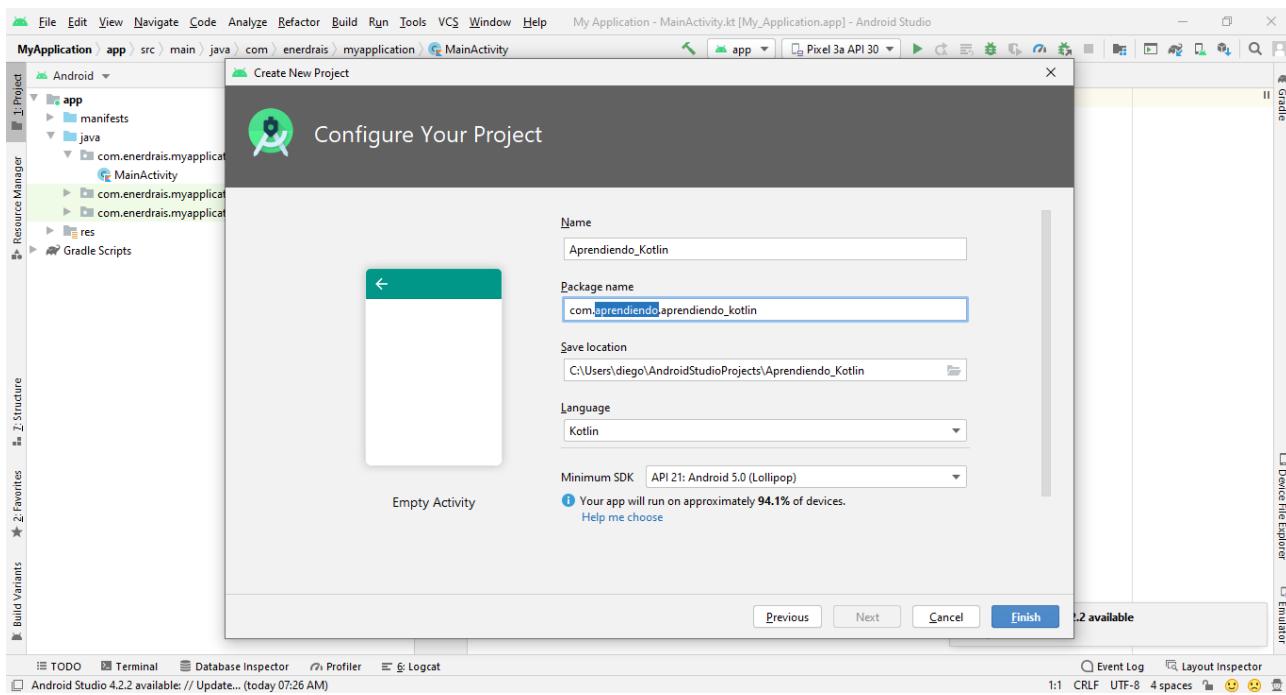


Además, se debe elegir el lenguaje que se utilizará en el proyecto, ya sea Java o Kotlin.

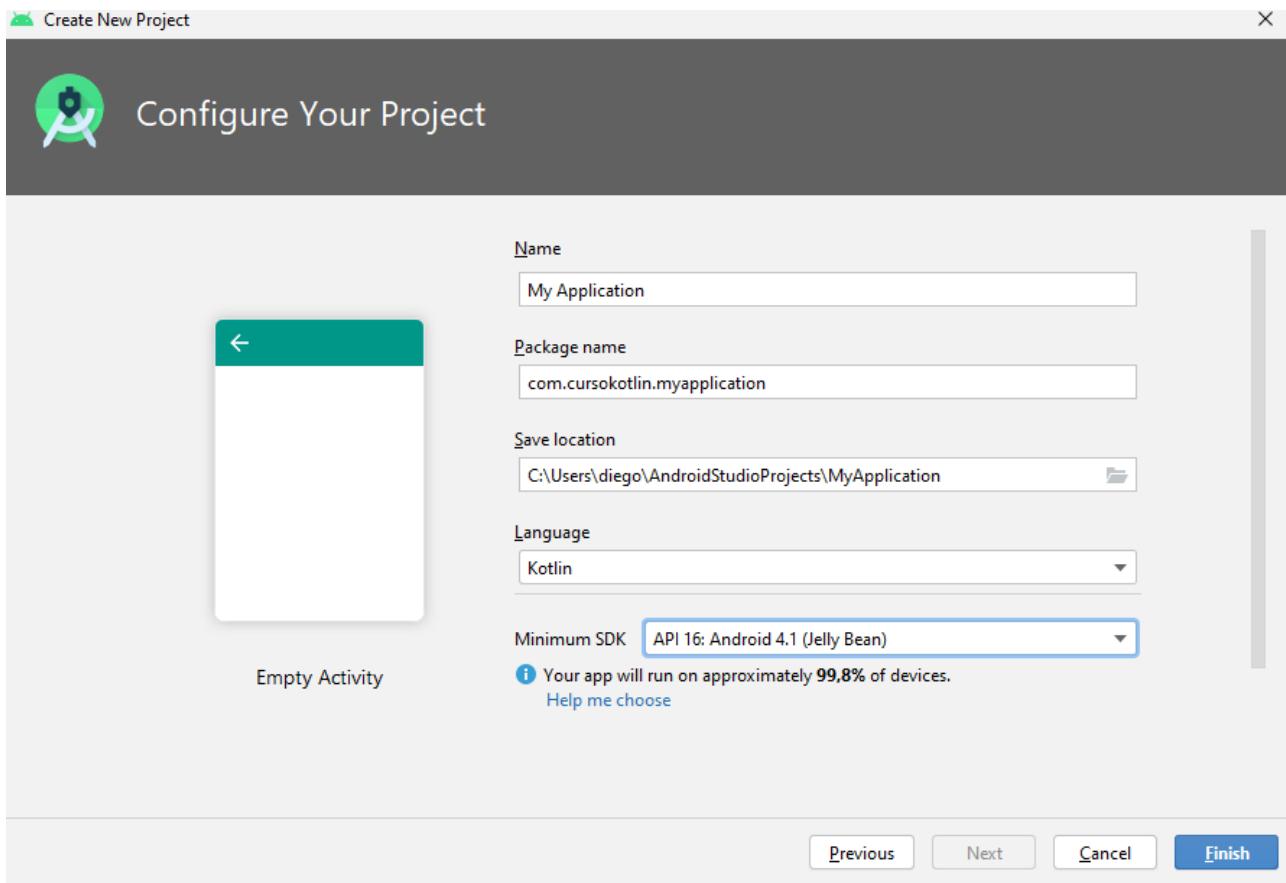
Kotlin fue creado por la empresa Jetbrains que se encarga de crear entornos de desarrollo como PHP Store, IntelliJ IDEA, etc. (IntelliJ IDEA es el precursor en el que se basó Jetbrains para la creación de Android Studio y sirve para crear aplicaciones de Kotlin que se ejecuten en consola), Kotlin es posible mezclarlo con Java para que grandes proyectos poco a poco se pudieran migrar a Kotlin.

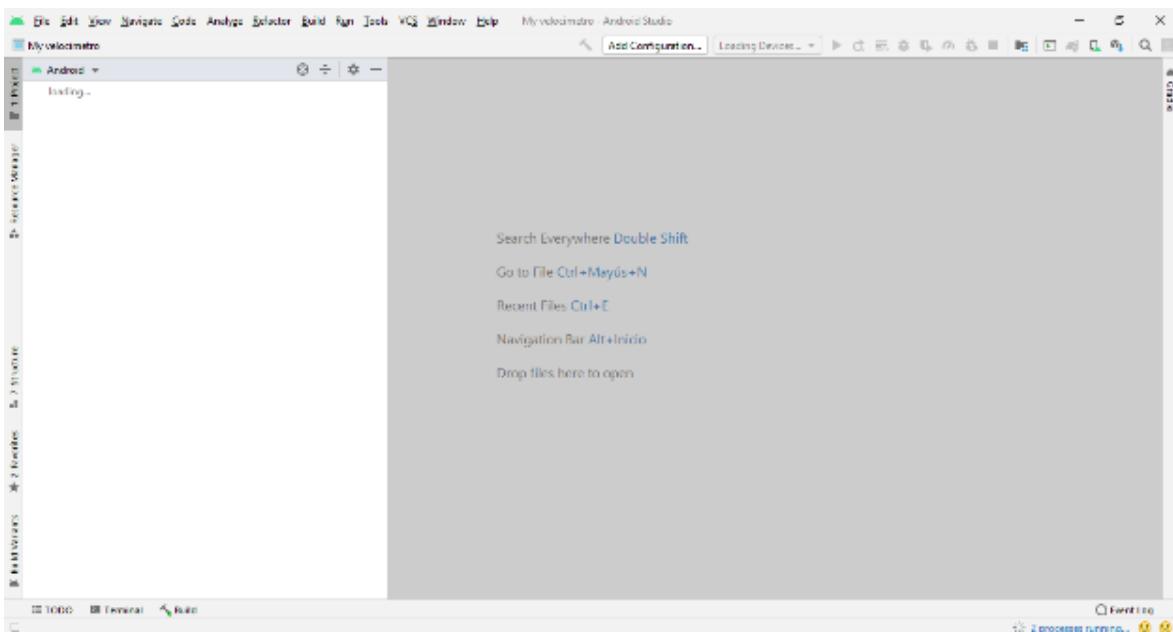


Al crear un nuevo proyecto, si este lo queremos lanzar a la Play Store en un futuro y que esté disponible para el público en general, debemos cambiar el paquete llamado example por uno propio. El Package name es básicamente el dominio que tendrá la aplicación Android.



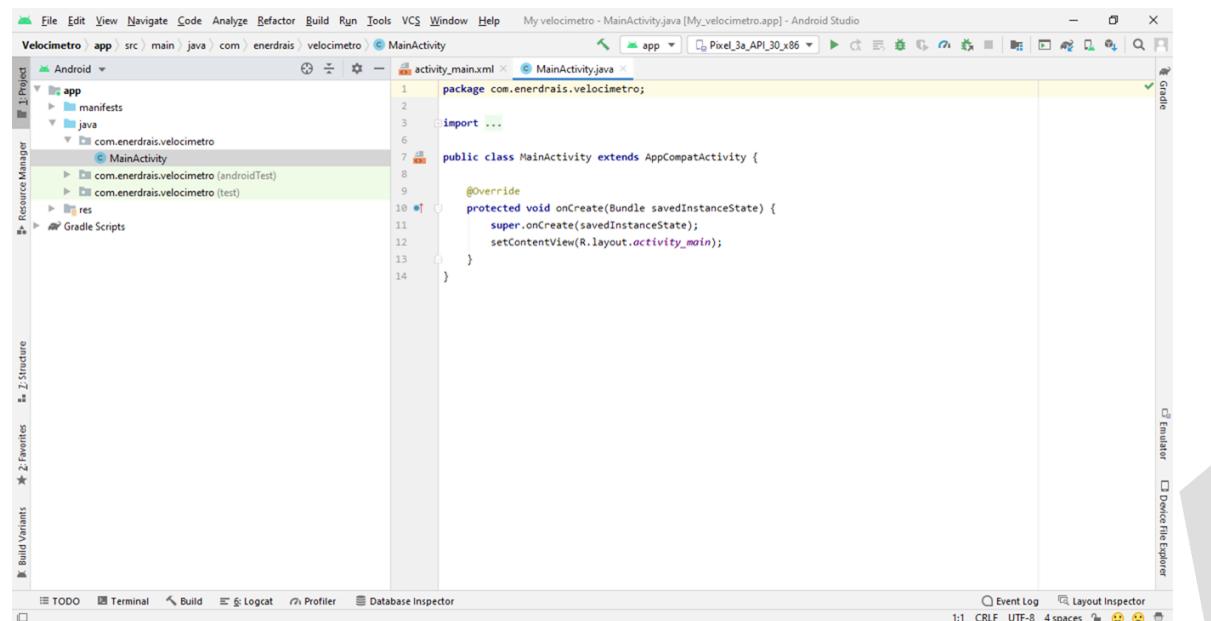
El SDK (Software Development Kit) elegido mostrará el porcentaje de dispositivos en donde se podrá ejecutar la aplicación, el mejor es el de API 16 porque se puede ejecutar en el 99.8% de los dispositivos, una vez acabado daré clic en Finish para que se empiece a crear el nuevo proyecto.





Mientras dice Loading lo que estará haciendo es crear las carpetas necesarias para el nuevo proyecto de Android, esto tardará un rato.

Ahora necesitamos una manera de correr la aplicación, ya que esta no se corre solamente en consola, para ello deberé usar un teléfono real o un emulador llamado AVD (Android Virtual Device) que simule un celular.



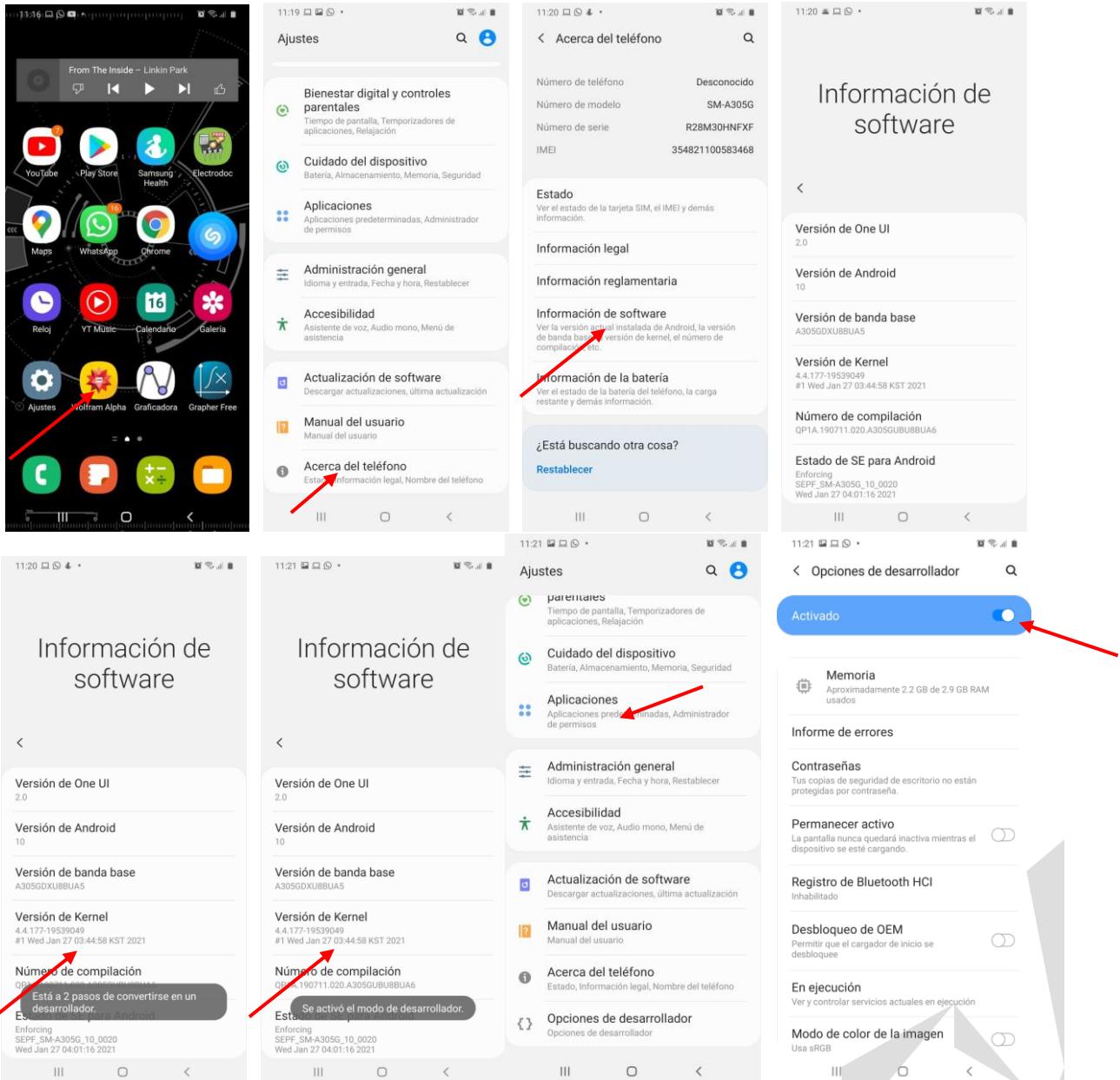
Android Studio – AVD y Modo desarrollador

Para poder visualizar los avances del código de la aplicación se tienen 2 caminos:

- Cambiar la configuración de algún dispositivo Android para poder conectarlo al ordenador y visualizar la aplicación desarrollada directamente en el celular.
- Ver el resultado del código por medio de un teléfono virtual (AVD).

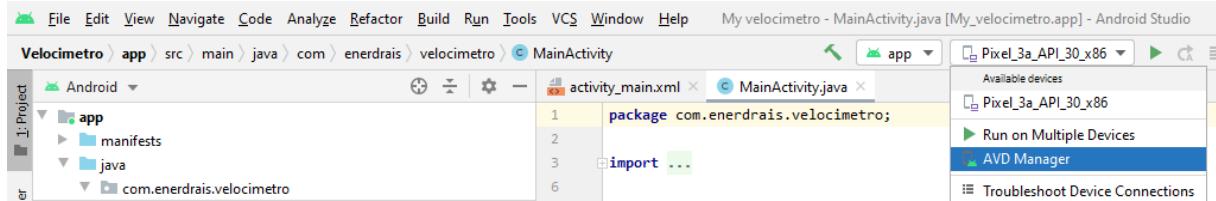
Para poder ejecutar la aplicación Android en un dispositivo móvil (modo desarrollador) debo introducirme a un celular con sistema operativo Android y seguir los siguientes pasos:

Ajustes → Acerca del teléfono → Información del software → Dar 4 clic en el botón de Número de compilación (al hacerlo aparecerá en pantalla cuántos clics me faltan para completar los 4) → Regresar al menú de Ajustes → Opciones de desarrollador (que antes no existía) → Activar el modo desarrollador.

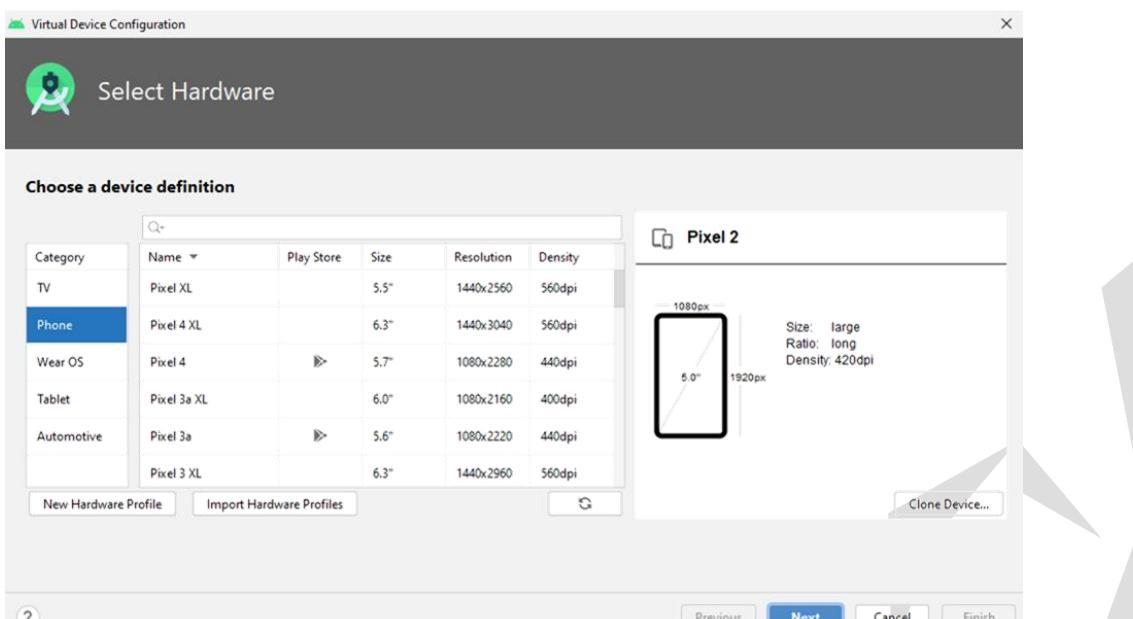
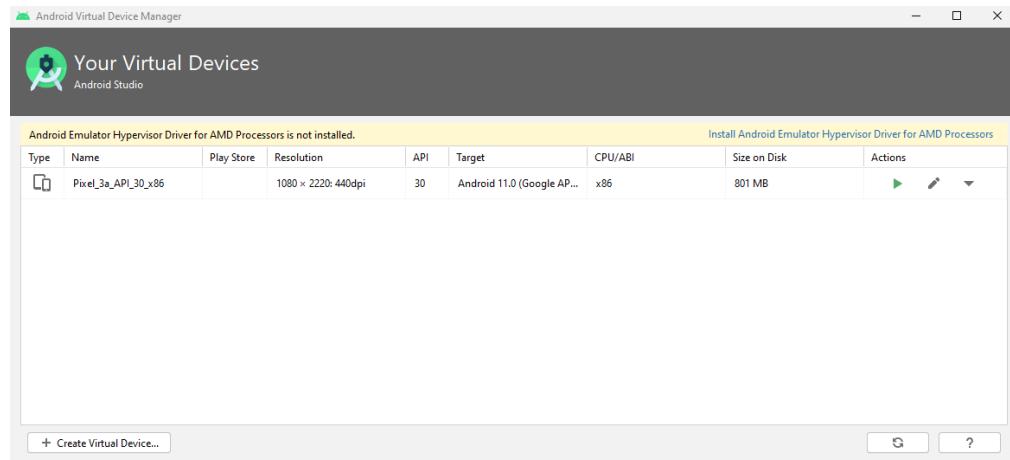


Para poder ejecutar el código de la aplicación en un teléfono virtual se creará un AVD, en este se deberá indicar a qué tipo de dispositivo está emulando. Usar un AVD es de gran utilidad para distintas simulaciones como la de seguir un recorrido en Google Maps, realizar una llamada, tomar una foto, etc.

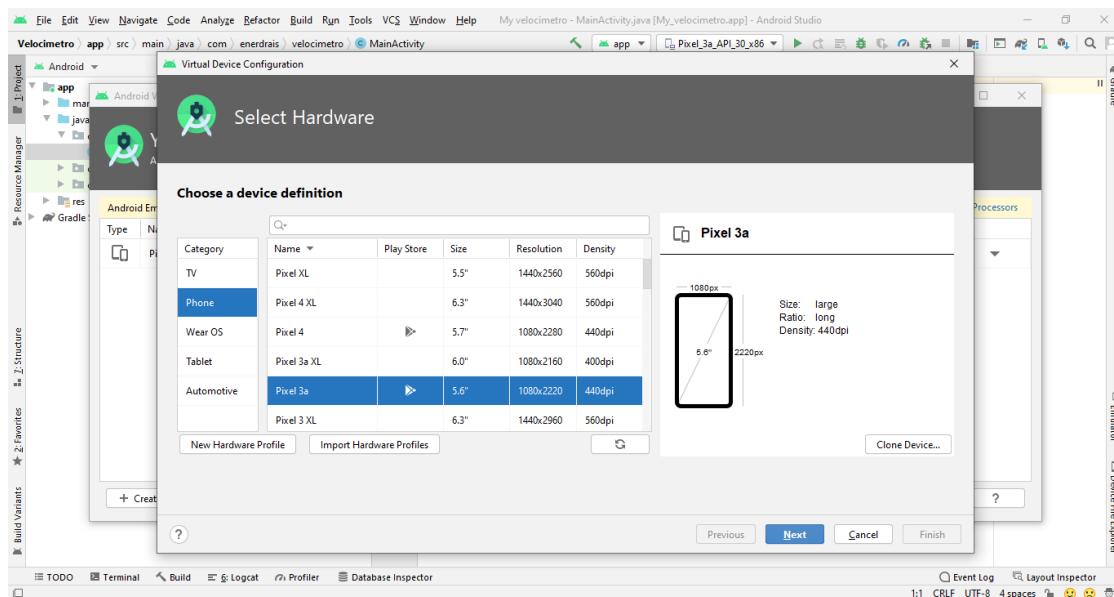
Para poder mostrar un AVD se necesita correr el AVD manager que se encuentra a la izquierda del botón de Play en la parte superior del área de trabajo.



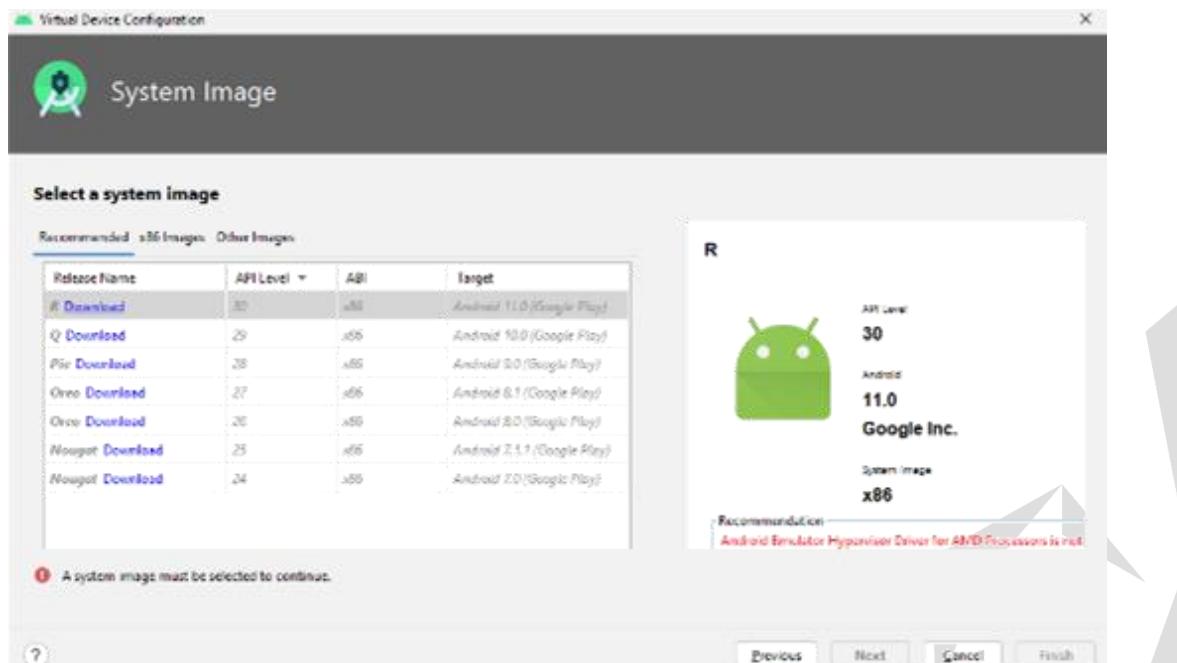
Cuando se dé clic en esta opción, saldrá una ventana donde aparecerán todos los AVD ya instalados (ya que puedo tener varios), al dar clic en el botón de Create Virtual Device, se me permitirá elegir si quiero que mi AVD sea una televisión, un teléfono, una tableta, etc.

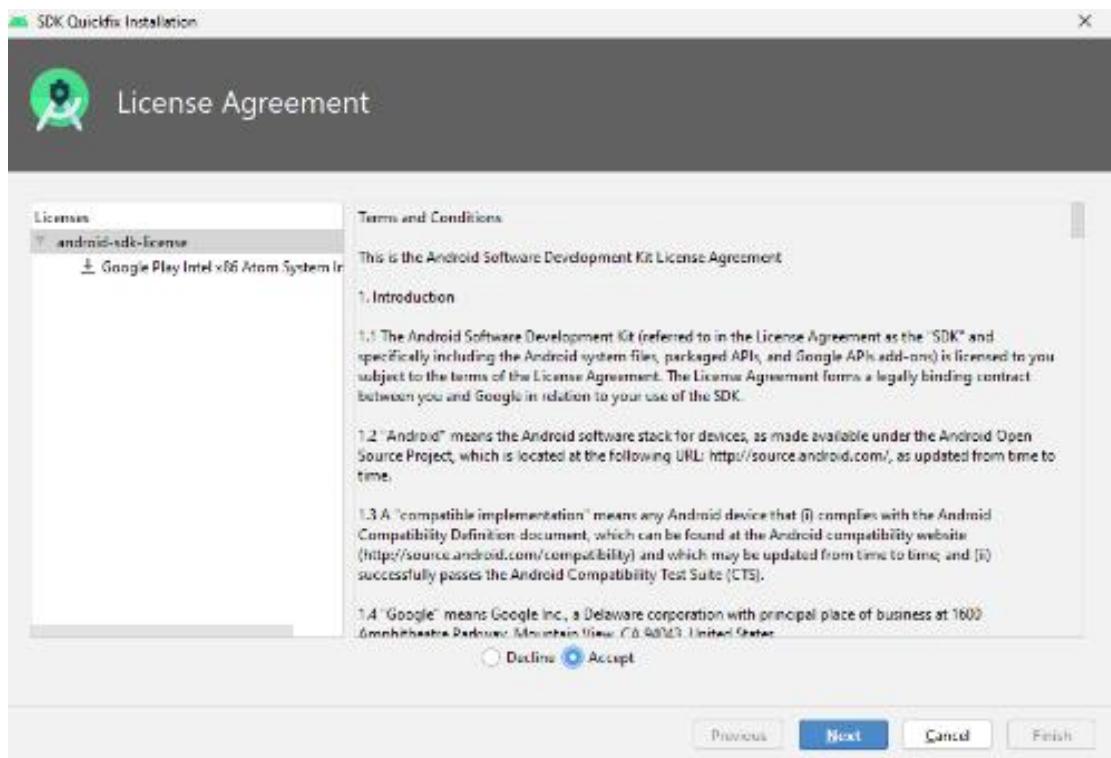


Las opciones de dispositivos son mostradas dando su nombre, el tamaño de su pantalla, si tiene ya instalada la Play Store o no, su resolución y densidad. En este caso vamos a elegir un teléfono Pixel 3a ya que el ícono que se encuentra en la columna que dice Play Store indica que ese AVD ya tiene la aplicación de Play Store instalada, los que no tienen el ícono no cuentan con esa app ya instalada, esto es importante porque si se quiere utilizar alguna librería que sea de Google como lo es Maps, necesariamente debe estar instalado en el AVD la Play Store.

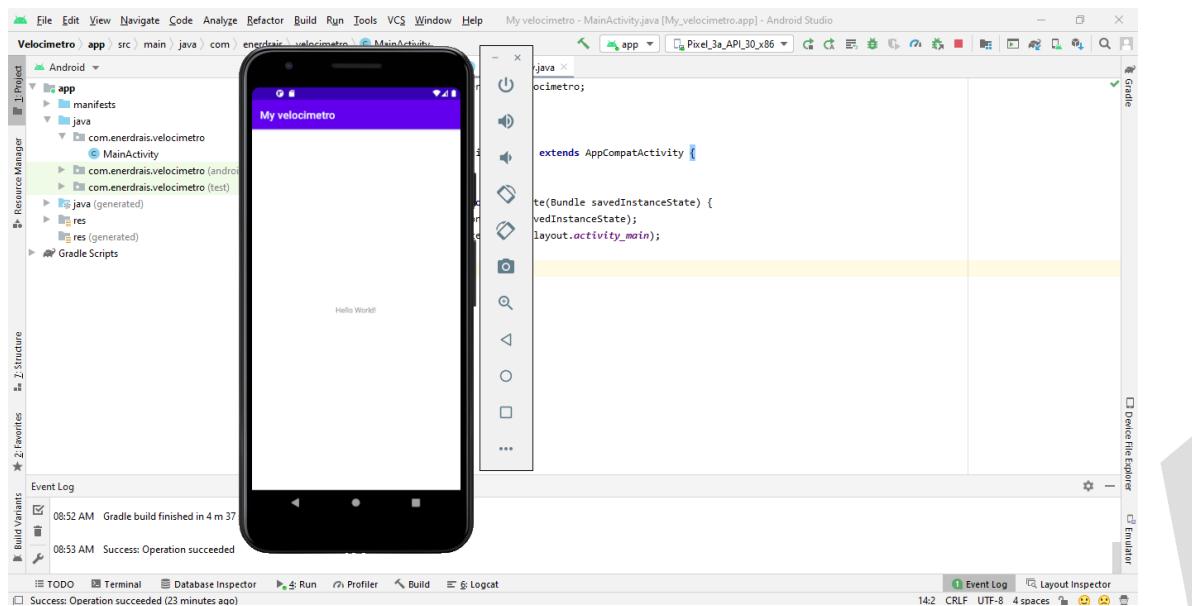


Después se elige un sistema de imagen, que son básicamente distintas ediciones del sistema operativo Android del teléfono virtual (osea el SDK y API, que es lo mismo que elegimos al crear el proyecto, pero en este caso es elegido específicamente para el AVD), es recomendable descargar la más reciente, para luego solo Aceptar los términos y condiciones y así descargar el sistema operativo que tendrá el AVD.



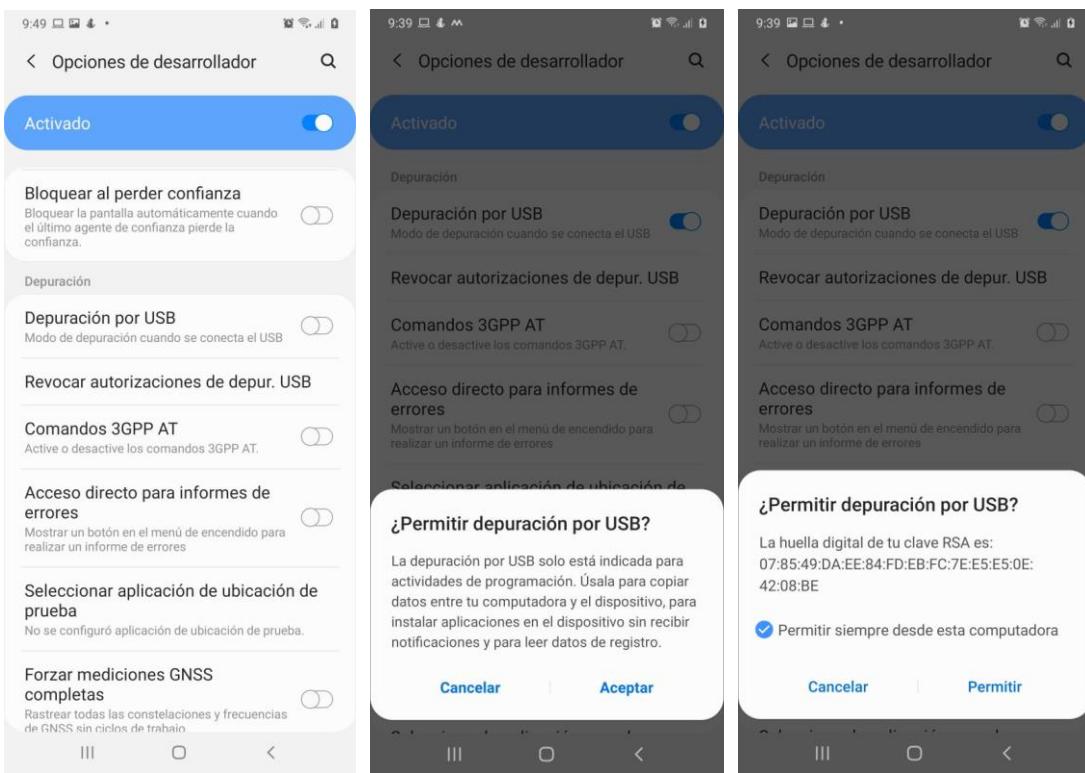


Ya habiendo descargado el AVD, si todavía no se ha escrito ninguna línea de código, al dar clic en el botón de Play que se encuentra alado del AVD Manager, se correrá el teléfono Virtual y se verá en él un Hola Mundo.

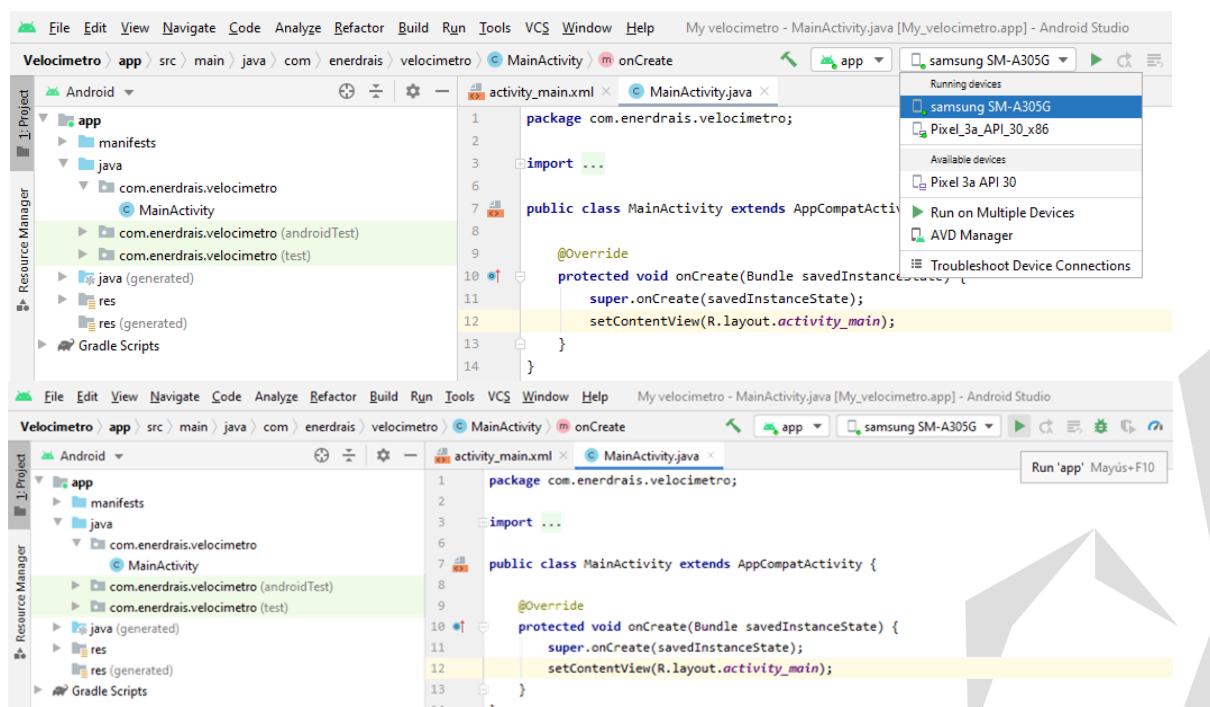


Para hacer pruebas del código desde el teléfono, se conecta el celular a la computadora por medio de USB y se sigue los siguientes pasos:

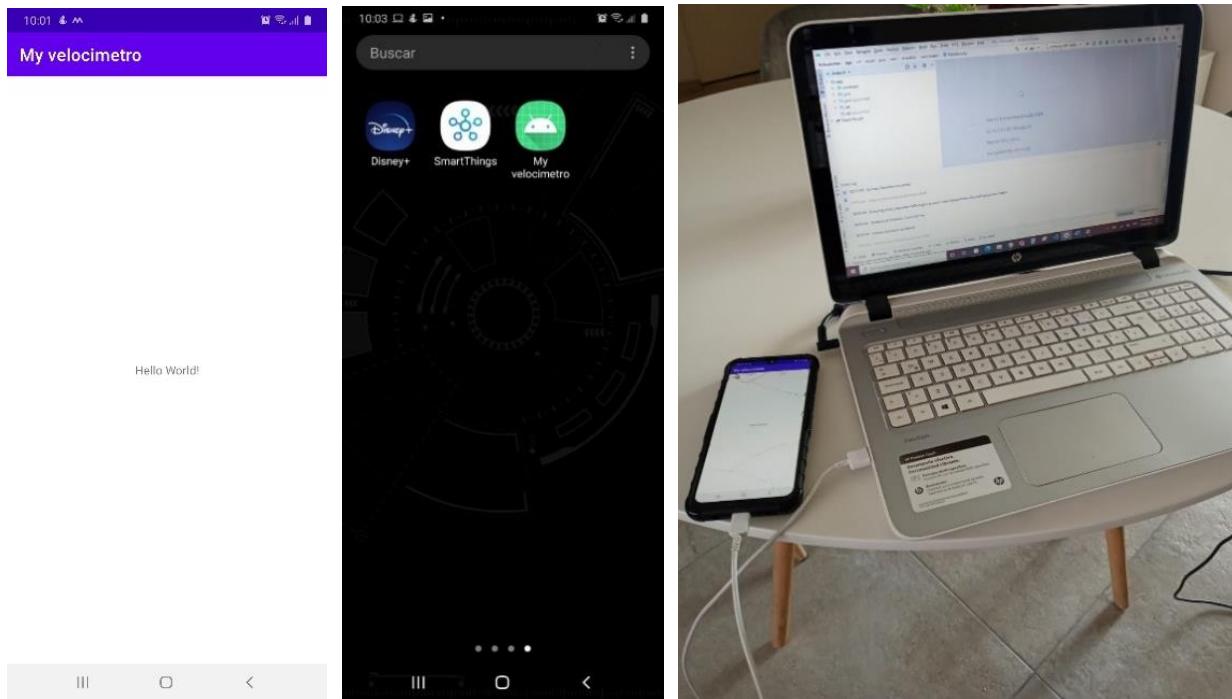
Ajustes → Opciones de desarrollador → Depuración por USB → Dar permiso a la computadora donde se conectó el teléfono Android.



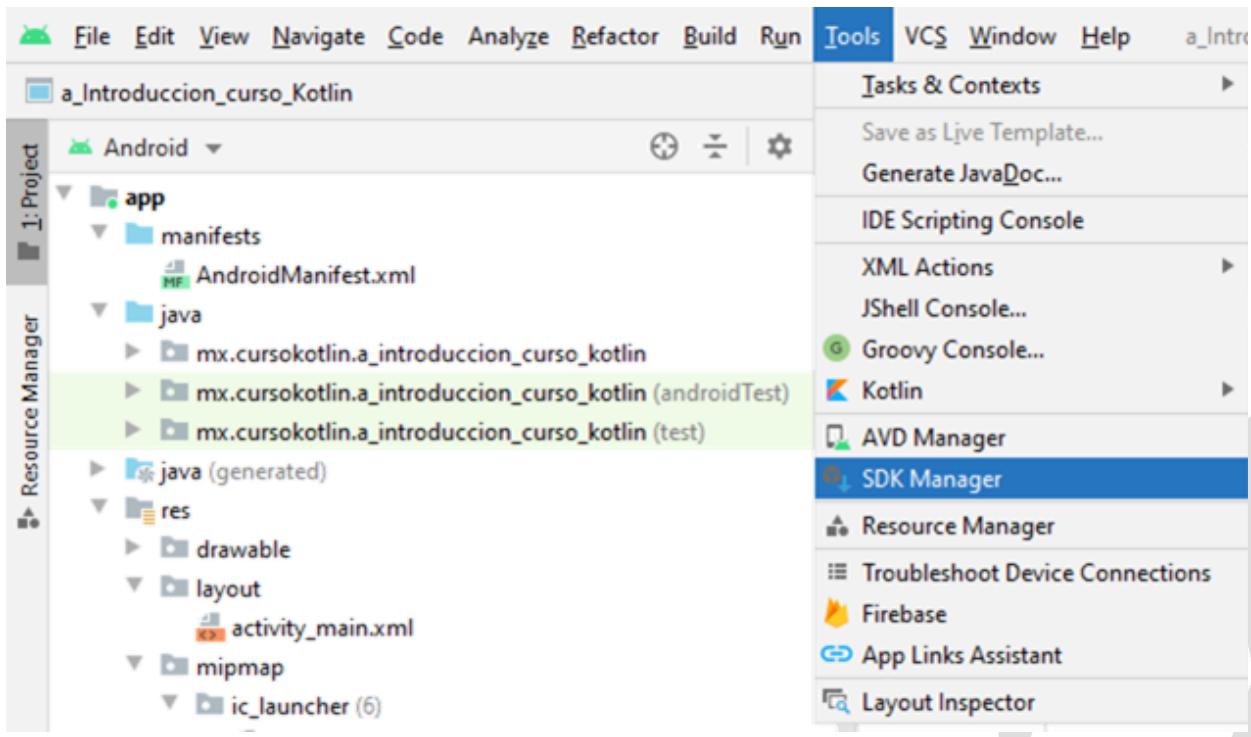
Ya habiendo habilitado y aceptado el modo de Depuración por USB en un celular con sistema operativo Android conectado al ordenador, se podrá elegir el celular conectado dentro de Android Studio en la parte donde se encuentra el AVD Manager y al dar clic en el botón de Play, el mismo Hola Mundo se podrá ver reflejado dentro del celular conectado, este proceso la primera vez tarda un poco de tiempo en ejecutarse.

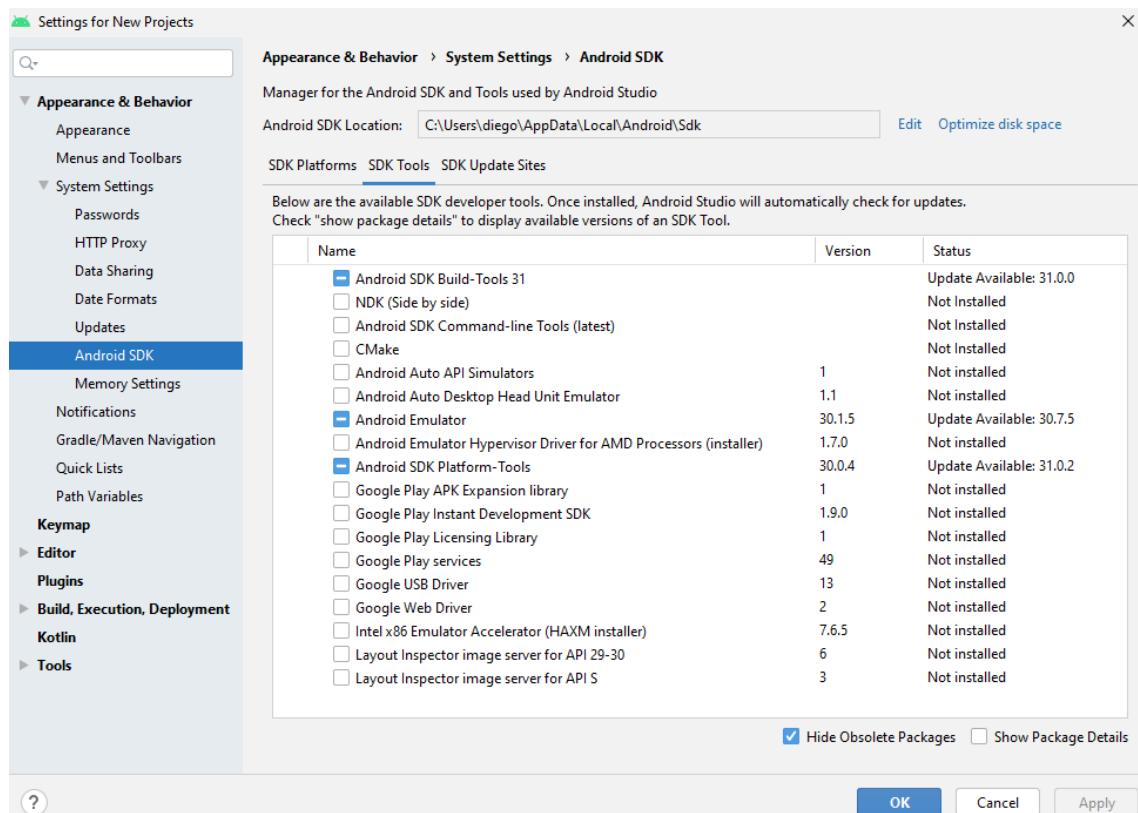


Hola mundo en celular con sistema operativo Android conectado por USB al ordenador:

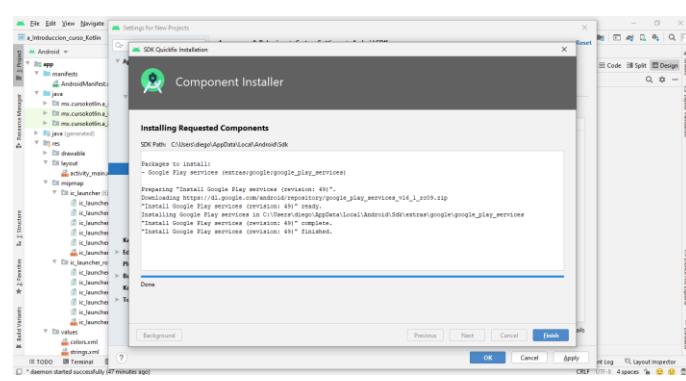
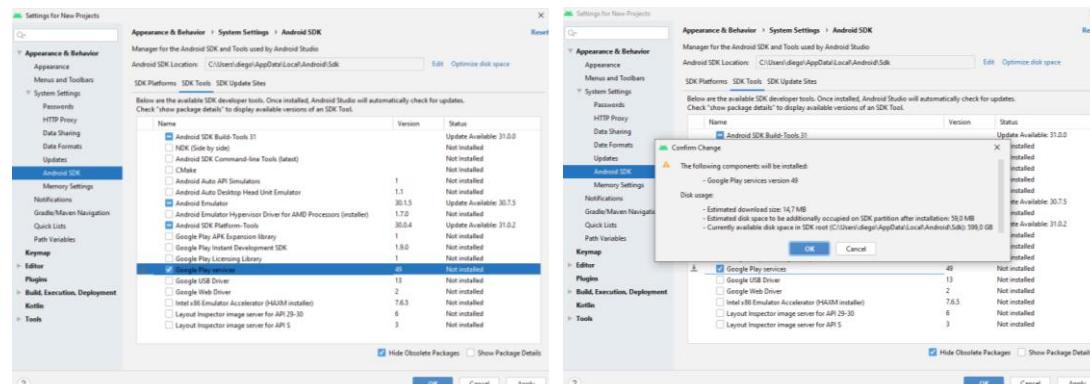


Cuando se quiera utilizar servicios como Google Maps, vamos a dirigirnos a la pestaña de Tools → SDK Manager → SDK Tools → Google Play services, seleccionaremos el checkbox y daremos clic en el botón de Ok para que se actualice ese paquete del SDK (Software Development Kit).



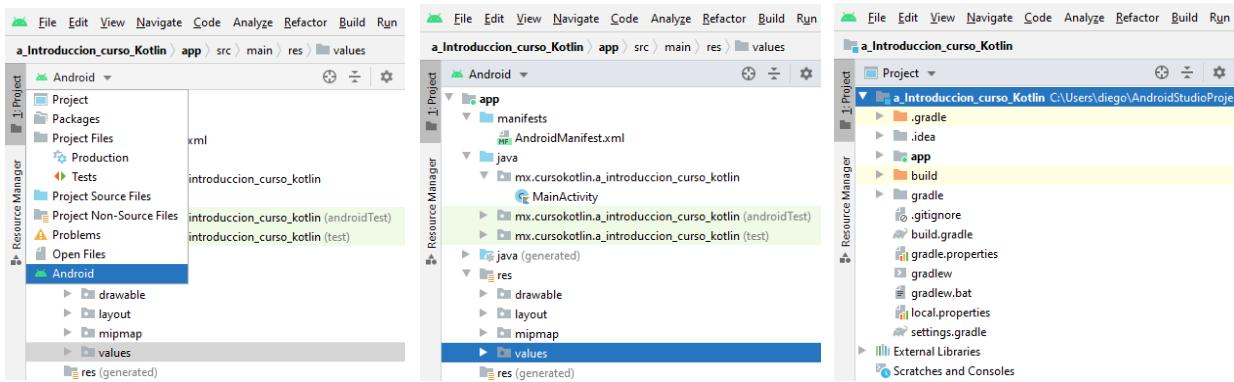


Al descargar el Google Play service podremos hacer uso de herramientas de Google como lo es Maps o Firebase dentro de la app Android.



Android Studio - Carpetas del proyecto

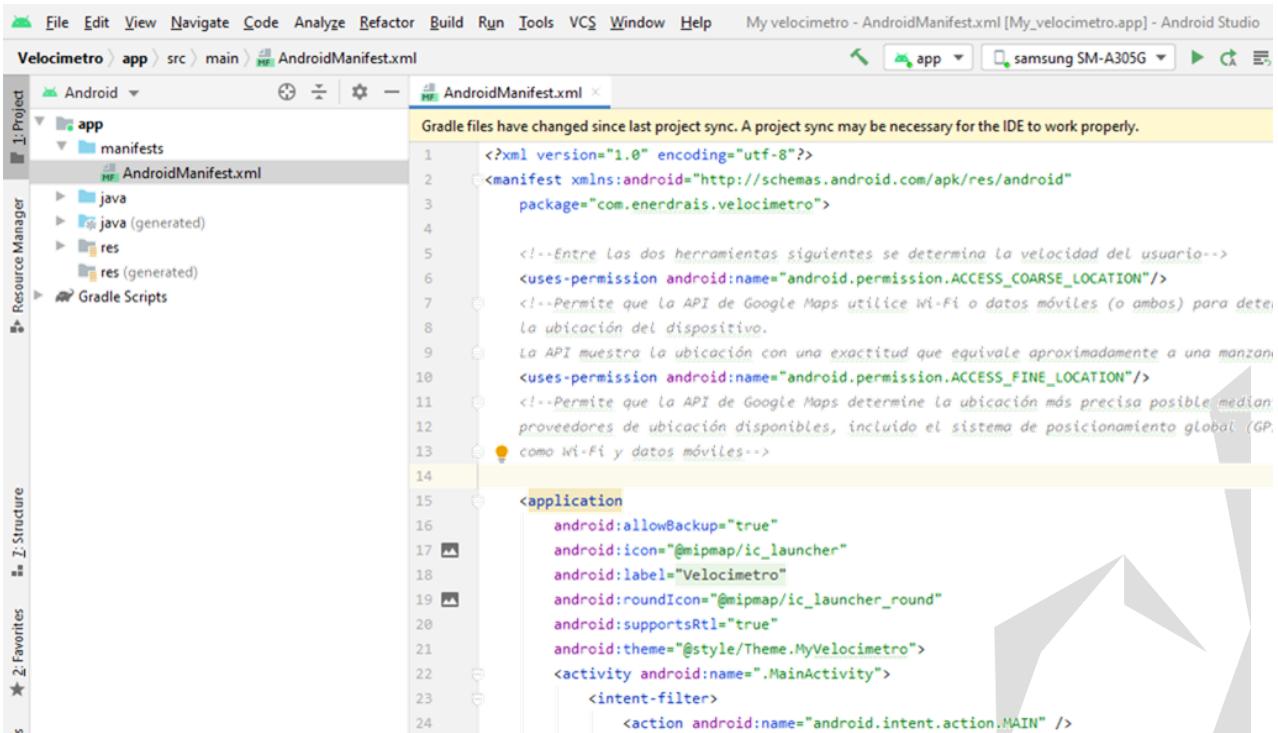
En la esquina superior izquierda de Android Studio podemos ver varias opciones para visualizar el proyecto, en específico las opciones de Project y Android sirven para visualizar las distintas carpetas del proyecto, pero se observan las mismas carpetas de forma distinta.



En esencia ambas opciones muestran lo mismo, pero es más utilizada la opción de Android, aunque la opción de proyecto muestra las carpetas como en realidad se encuentran dentro del ordenador.

Las carpetas importantes del proyecto donde se afectan sus distintos aspectos estéticos y funcionales son:

- **app → manifest → AndroidManifest.xml:** En este archivo del proyecto se enlistan los permisos necesarios para utilizar las distintas herramientas del celular que requiera la aplicación para funcionar, como sensores, cámara, etc.



- En el archivo manifest también se indican todas las actividades que vayan a conformar la aplicación, en este caso como se acaba de crear el proyecto, solo está indicada la Main activity.

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="mx.curso.kotlin.a_introduccion_curso_kotlin">

    <!--En esta parte del archivo manifest del proyecto se enlistan los permisos necesarios para
    utilizar las distintas herramientas del celular que requiera la aplicación para funcionar, como
    sensores, cámara, etc.-->

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="ISIA - Tienda en linea"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportRtl="true"
        android:theme="@style/Theme.A_Introduccion_curso_Kotlin">
        <!--Dentro del archivo manifest también se indican las actividades (osea las pantallas) que
        conforman la aplicación, si en el flujo de ejecución de la app se utiliza alguna actividad
        que no esté registrada en el archivo manifest, la aplicación se cerrará-->
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>

```

- **app → java → com.nombreDominio.nombreProyecto (dominio de la aplicación declarada al crear el proyecto) → MainActivity.java o MainActivity.kt:** En este archivo del proyecto de Android Studio se encuentra el Main Activity, es un archivo de extensión y lenguaje Java o Kotlin que funciona como el inicio de la ejecución del código y maneja todas las funcionalidades del proyecto. Dentro de esta carpeta se desarrollan las funciones pertenecientes a cada activity que conforma al proyecto Android, en estos archivos se crean solo las funciones de las activities, pero no las interfaces gráficas, esas se crean en la carpeta **app/java/res/layout** por medio de archivos XML (la carpeta siempre se llama Java, aunque el proyecto esté hecho con Kotlin).

```

package com.enerdrais.velocimetro;

import ...

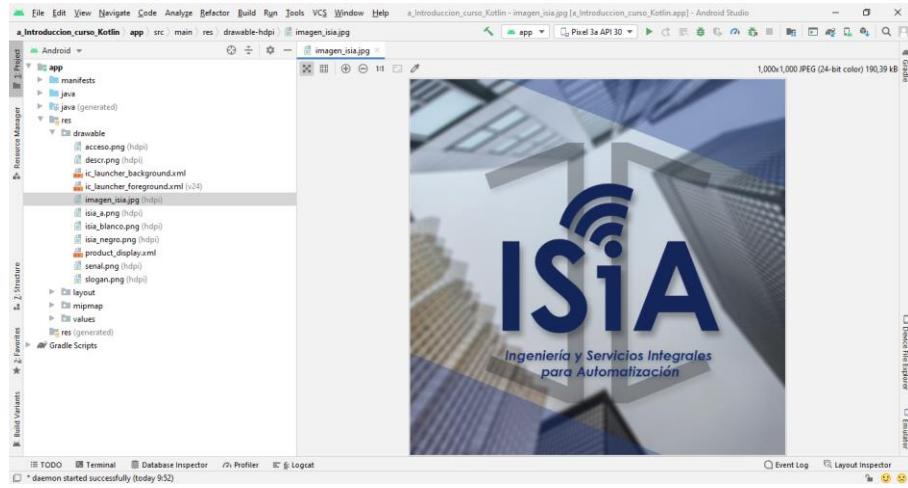
public class MainActivity extends AppCompatActivity {

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

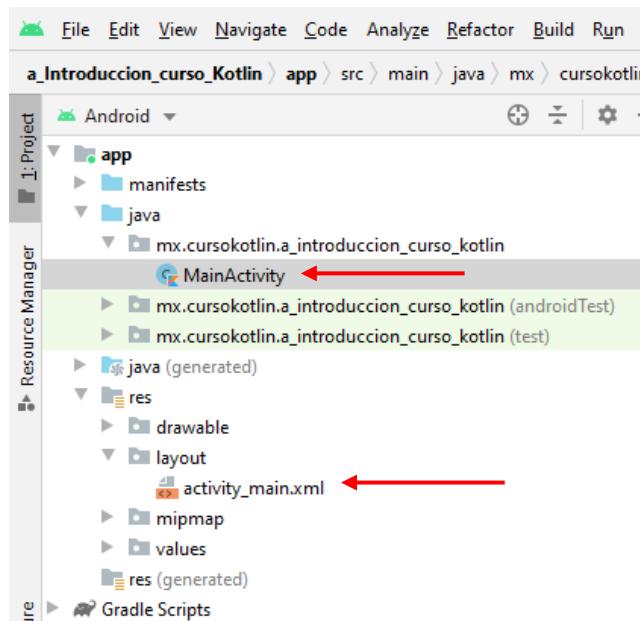
```

- Las otras carpetas que se encuentran dentro de la carpeta **java** que tienen entre paréntesis la palabra (androidTest) y (test) sirven para hacer pruebas unitarias, pero no para crear y diseñar la app.
- **app → java → res → drawable:** La carpeta **res** es llamada **Recursos** y contiene todos los elementos gráficos de la aplicación, desde las imágenes, colores, textos, interfaces gráficas, etc.

En específico en la carpeta de drawable se encuentran todas las imágenes del proyecto, ya sea en formato .jpg, .png, .gif, etc.

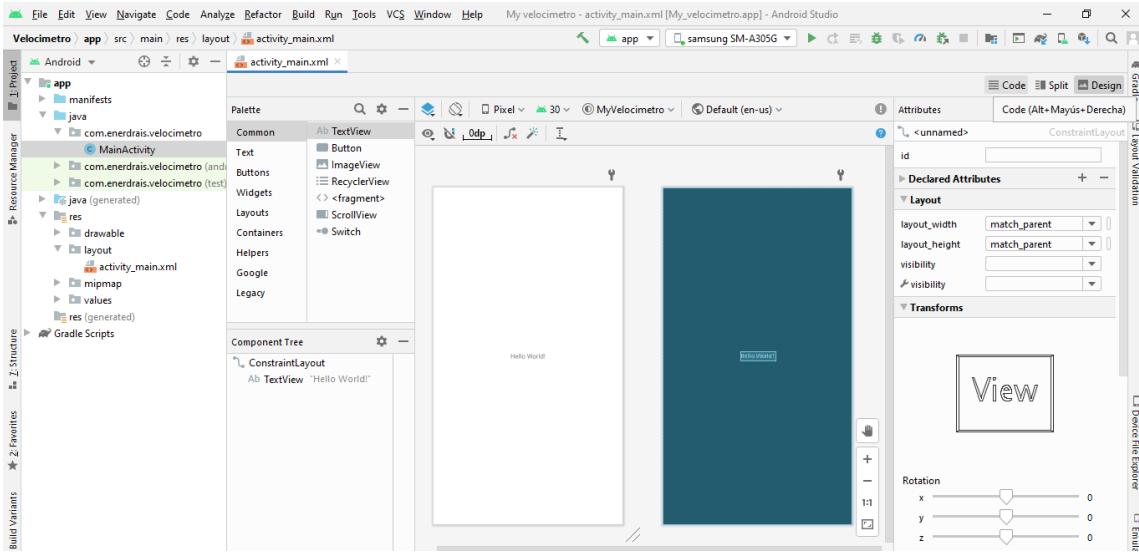


- **app → java → res → layout → activity_main.xml:** En la carpeta **app/java/res/layout** se crean todos los archivos XML que representen la interfaz gráfica de todas las pantallas del proyecto, el nombre de los elementos creados en la carpeta **res** deben estar hechos con todas las letras minúsculas, sin caracteres raros como ñ o acentos, sin espacios ni empezar con un número.
En el archivo **activity_main.xml** del proyecto es donde se editan los aspectos visuales de la interfaz gráfica del proyecto que está ligada al archivo **MainActivity.kt** que está hecho con Kotlin y le da sus funcionalidades a la interfaz de la activity.
Aunque XML es un lenguaje de marcado que principalmente sirve para la trasmisión de datos, en Android se utiliza para colocar elementos gráficos llamados **Views**, como botones, áreas de texto, etc.

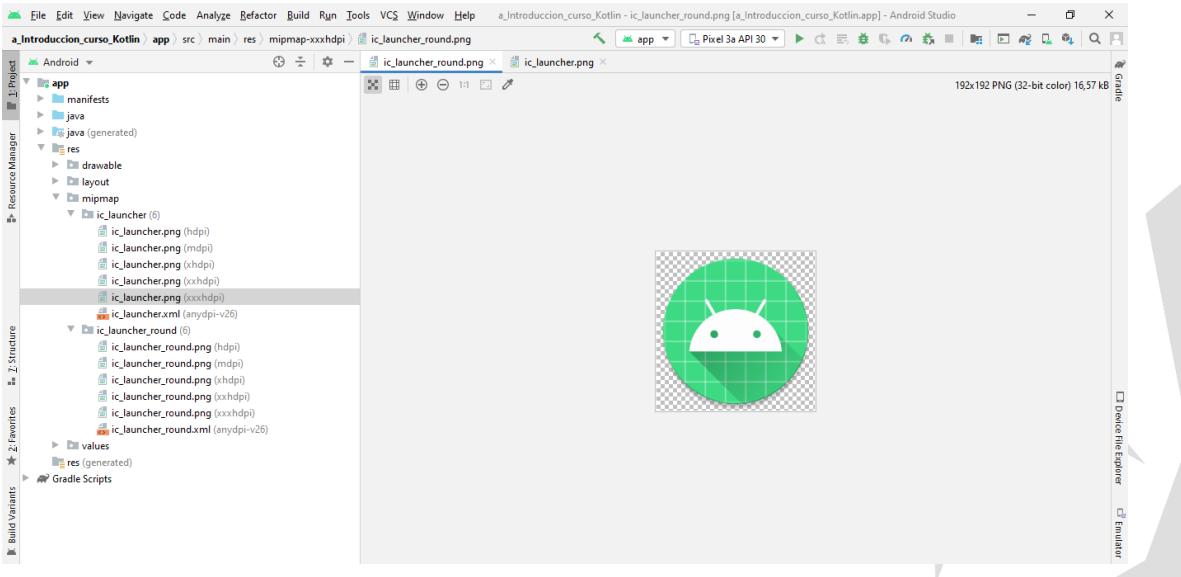


Para ver el código XML de los elementos gráficos que se están creando en la interfaz se selecciona el botón **Code** que se encuentra en la esquina superior derecha.

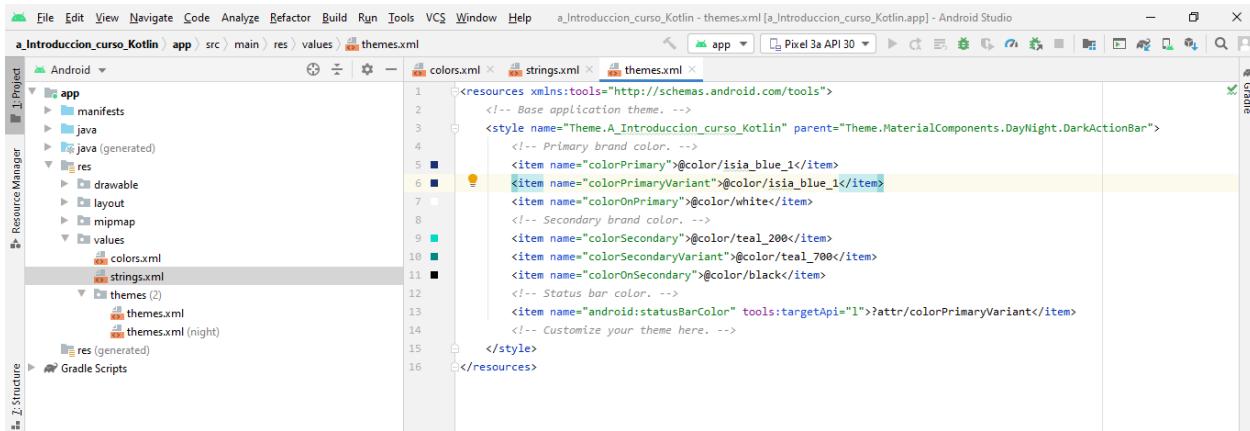
- En el área de diseño hay dos ventanas, a la blanca se le llama Design y a la azul se le llama Blueprint y muestran el acomodo de los distintos componentes que conforman la interfaz de la aplicación.
 - **Design:** Muestra los elementos que se van a ver en la aplicación en todo momento.
 - **Blueprint:** Muestra los componentes que conforman la interfaz, ya sean visibles en la interfaz de la aplicación durante todo momento o no.
- Cada uno de los componentes colocados en la interfaz de la aplicación es llamado **View** y al contenedor que puede incluir varios Views se le llama **Layout**.



- **app → java → res → mipmap:** En la carpeta mipmap se encuentran los íconos de la aplicación, por defecto las apps tienen el ícono de Android, por lo que aparecerán así en el dispositivo móvil donde sean probados. Los íconos se colocan de varios tamaños (hdpi, mdpi, xhdpi, xxhdpi, xxxhdpi) para que el ícono sea responsive, osea adaptable al tamaño de pantalla del dispositivo donde sea usada la app.



- **app → java → res → values:** En la carpeta values del proyecto se declaran valores estáticos (constantes) que se van a reutilizar en la aplicación, como lo son colores o cadenas de caracteres (Strings), estos podrán ser reutilizados en todas las partes del proyecto, en la carpeta de themes por ejemplo se podrán utilizar para dar color a la barra superior de la app Android.

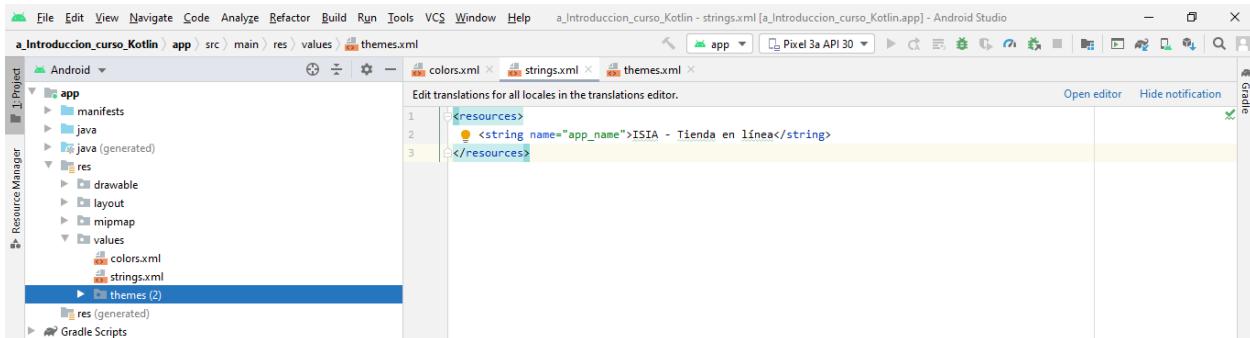


```

<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Theme.A_Introduccion_curso_Kotlin" parent="Theme.MaterialComponents.DayNight.DarkActionBar">
        <!-- Primary brand color. -->
        <item name="colorPrimary">@color/isia_blue_1</item>
        <item name="colorPrimaryVariant">@color/isia_blue_1</item>
        <item name="colorOnPrimary">@color/white</item>
        <!-- Secondary brand color. -->
        <item name="colorSecondary">@color/teal_200</item>
        <item name="colorSecondaryVariant">@color/teal_700</item>
        <item name="colorOnSecondary">@color/black</item>
        <!-- Status bar color. -->
        <item name="android:statusBarColor" tools:targetApi="l">?attr/colorPrimaryVariant</item>
        <!-- Customize your theme here. -->
    </style>
</resources>

```

- **app → java → res → values → strings.xml:** En este archivo del proyecto se indica el texto que aparece hasta arriba de la pantalla en la aplicación Android o se pueden poner textos de la aplicación que se traducirán por sí solas a diferentes lenguajes.

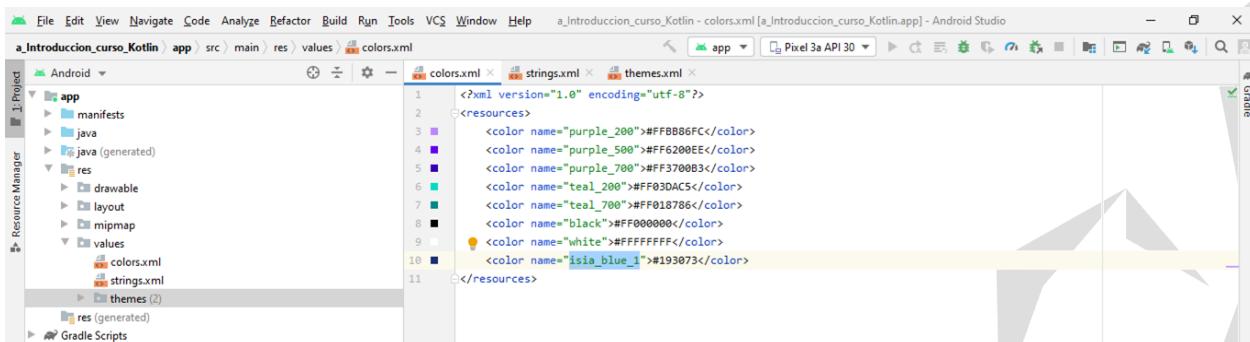


```

<resources>
    <string name="app_name">ISIA - Tienda en línea</string>
</resources>

```

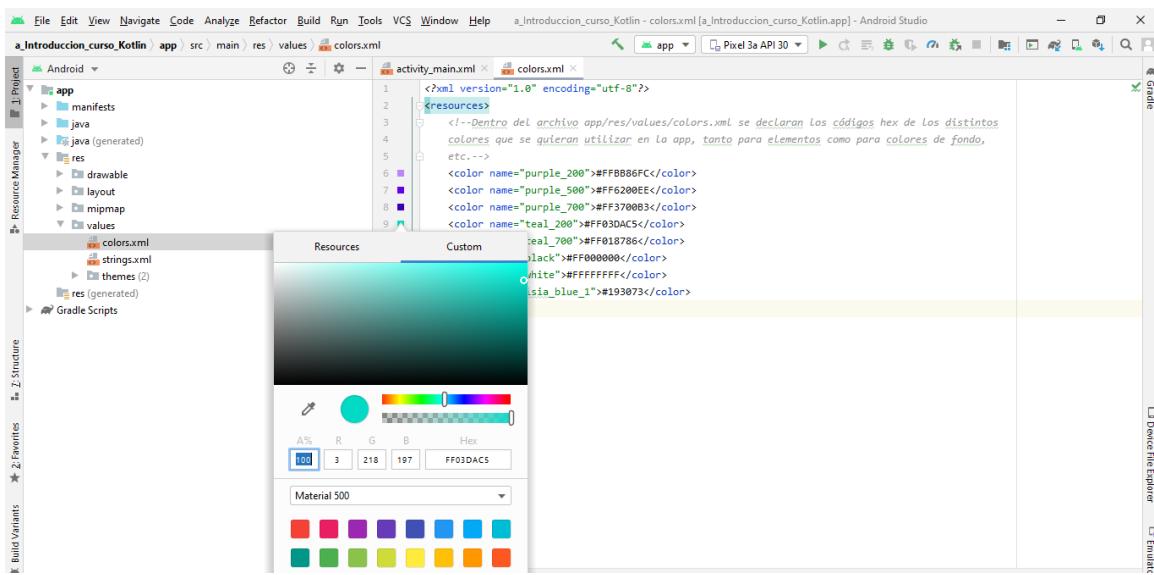
- **app → java → res → values → colors.xml:** En este archivo del proyecto se indican los colores en forma hexadecimal (o hex) que conforman a la aplicación Android, en la parte derecha de la línea de código que representa cada color se muestra un cuadrado con el color que representa, si damos clic en ese cuadrado aparecerá una ventana donde se puede editar el color como queramos, cambiando la transparencia, el mismo color, etc.



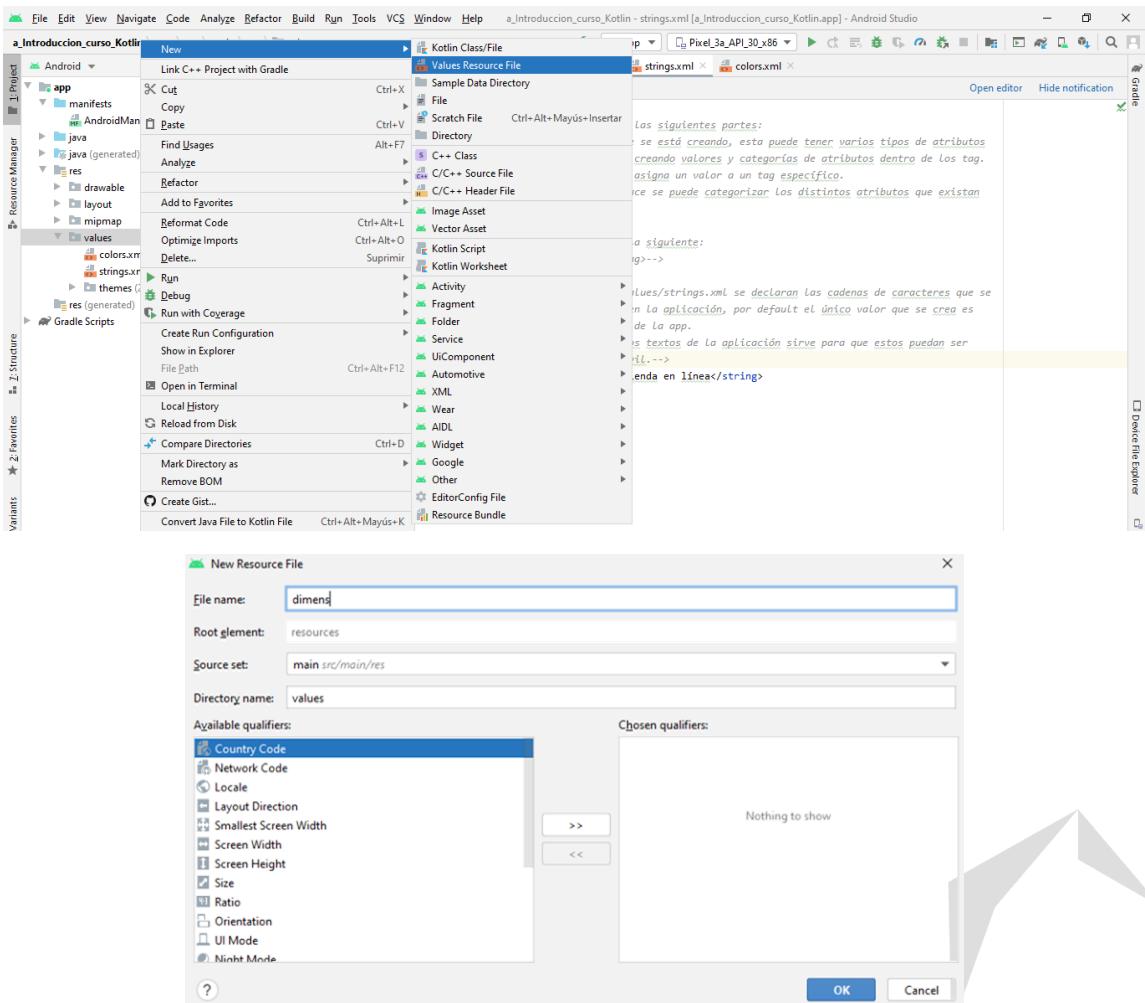
```

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="purple_200">#FFBB86FC</color>
    <color name="purple_500">#F6208E</color>
    <color name="purple_700">#F3700B</color>
    <color name="teal_200">#F03DAC5</color>
    <color name="teal_700">#FB1878C</color>
    <color name="black">#F000000</color>
    <color name="white">#FFFFFF</color>
    <color name="isia_blue_1">#193073</color>
</resources>

```



- **app → java → res → values → dimens.xml:** normalmente este archivo **no se crea por default en Android Studio**, para crearlo debemos dar clic derecho en la carpeta de values y seleccionar la opción de New → Values Resource File → Nombre: dimens.



En este archivo se indican dimensiones compartidas, como por ejemplo los tamaños de letra, los anchos de una pantalla, el alto (height) o ancho (width) de una imagen, etc. Esto se hace a través de la etiqueta con el tag `<dimen name = "id_atributo">valor</dimen>`, esto se hace para que se pueda utilizar ciertos tamaños de imagen o texto y que, al cambiarlo en este archivo, el valor se cambie en todos lados donde fue usado.

```

<?xml version="1.0" encoding="utf-8?>
<resources>
    <!--Las etiquetas XML se componen de las siguientes partes:
        - tag: Es el tipo de etiqueta XML que se está creando, esta puede tener varios tipos de atributos
            con su respectivo valor y namespace, creando valores y categorías de atributos dentro de los tag.
        - atributo: Es la forma en la que se asigna un valor a un tag específico.
        - namespace: Por medio de los namespace se puede categorizar los distintos atributos que existan
            dentro de un tag.

    La sintaxis de las etiquetas XML es la siguiente:>
    <tag namespace="atributo="valor"></tag>-->

    <!--La etiqueta utilizada para indicar Las dimensiones tiene el tag dimen, se diferencian entre
        ellos por el atributo name y su valor se pone dentro de las etiquetas de apertura y cierre con
        tag dimen.
        - sp: Con la unidad de medida sp se indica el tamaño de un texto.
        - dp: Con la unidad de medida dp -->
        <dimen name="tamaño_texto">14dp</dimen>
</resources>

```

- **Gradle scripts → build.gradle (Project: NombreProyecto.app):** Dentro del archivo build.gradle (Project: NombreProyecto.app) se tiene el archivo de configuración del proyecto completo.

DIFERENCIA DE PROJECT VS MODULE:

- **Project (proyecto):** Se refiere a todas las carpetas que conforman al proyecto android, incluyendo la carpeta de app y todas las que contiene, además de la de Gradle Scripts.
- **Module (módulo):** Se refiere a los distintos módulos que se pueden crear de una misma aplicación para utilizarla en distintos dispositivos Android como lo son una televisión, un teléfono, una tableta, un smartwatch, un automóvil, etc. Si ese fuera el caso, no existiría solo la carpeta app, sino que se crearía una carpeta nueva por cada dispositivo donde se fuera a ejecutar el proyecto, cada una de ellas representaría un módulo. La carpeta app en específico representa el módulo que puede ejecutar el código Kotlin en teléfonos y tabletas.

```

You can configure Gradle wrapper to use distribution with sources. It will provide IDE with Gradle API/DSL documentation. Hide the tip Ok, apply suggestion!
Gradle files have changed since last project sync. A project sync may be necessary for the IDE to work properly. Sync Now Ignore these changes
buildscript {
    ext.kotlin_version = "1.4.32"
    repositories {
        google()
        jcenter()
    }
    dependencies {
        classpath "com.android.tools.build:gradle:4.1.0"
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"
    }
}

// NOTE: Do not place your application dependencies here; they belong
// in the individual module build.gradle files

```

- **Gradle scripts → build.gradle (Module: NombreProyecto.app):** En este archivo del proyecto se importan las librerías que se quiera usar dentro de la aplicación Android, específicamente se colocan en la parte del código donde dice dependencies.
- Graddle es el sistema de compilación de Android, administrando todos los archivos necesarios para poder correr el proyecto en este módulo en específico, osea en la carpeta app creada para que el proyecto se ejecute en celulares o tabletas.

```

18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41

```

```

buildTypes {
    release {
        minifyEnabled 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:1.2.0'
    implementation 'com.google.android.material:material:1.3.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
    testImplementation 'junit:junit:4.+'
    androidTestImplementation 'androidx.test.ext:junit:1.1.2'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'

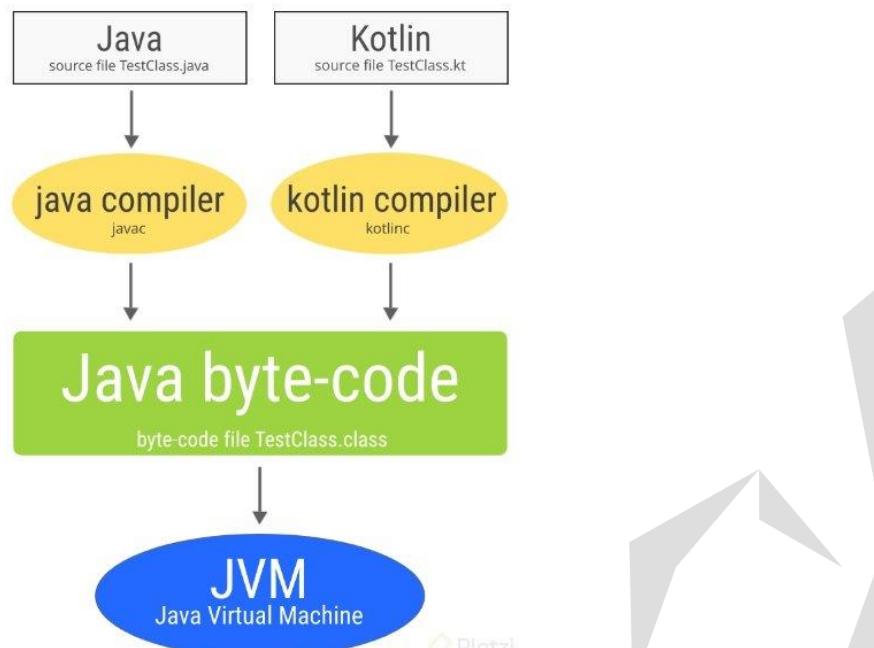
    //Librerías agregadas
    implementation 'com.google.android.gms:play-services-location:17.0.0'
}

```

Estructura de Compilación - Proyecto Android Studio

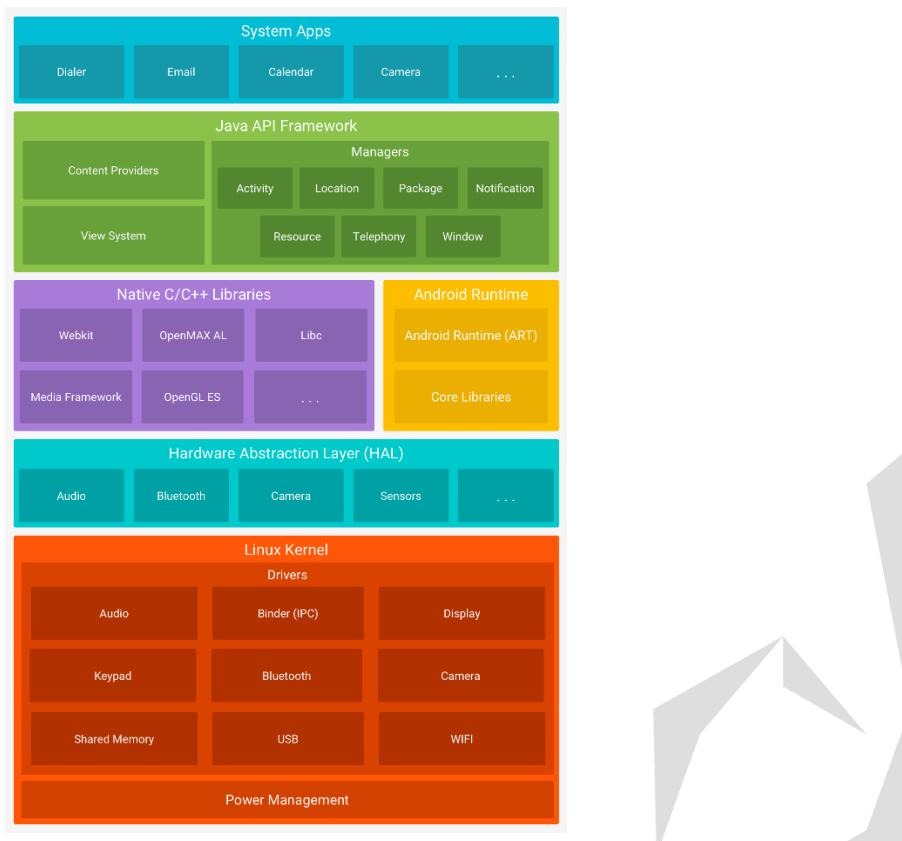
Como Android está basado en el lenguaje de programación Java (aunque se programe con Kotlin), debemos tener instalado el JDK (Java Development Kit) en la computadora ya que Java es un lenguaje multiplataforma y este software le permite al ordenador interpretar el código creado.

Tanto Java como Kotlin son lenguajes multiplataforma, esto significa que para correr su código se crea un archivo intermedio llamado **bytecode** con extensión **.class**, el cual permite que su código pueda ser ejecutado en cualquier sistema operativo, ya sea Windows, IOS o Linux.



La arquitectura de los proyectos Android es la siguiente, indicando las capas y distintos lenguajes de programación de los que está compuesto un proyecto Android.

- **Linux Kernel:** El sistema operativo Android está basado en Linux, Android es una modificación del sistema operativo Linux utilizada y especializada en dispositivos móviles.
- **Hardware:** Es la capa donde se tiene acceso al Audio, Bluetooth, Cámara, Sensores, etc. del dispositivo móvil.
- **Native C/C++ Libraries & Android Runtime (ART):**
 - Las librerías de C y C++ se utilizan para interactuar de forma directa con los elementos de hardware del teléfono y algunas aplicaciones Android se pueden programar en estos lenguajes para hacer más rápido el tiempo de ejecución de una app Android, ya que de esta manera nos saltamos la ejecución del JVM (Java Virtual Machine).
 - El Android Runtime, también llamado ART es una máquina virtual que forma parte de un conjunto de máquinas virtuales internas que sirven para compilar el código hecho con Java o Kotlin a código byte codes con extensión .class (que está hecho para funcionar con laptops o PC) y luego a un archivo tipo .dex que pueda interpretar el dispositivo móvil. **La máquina virtual que compila el código Java o Kotlin a .class es la JVM (Java Virtual Machine) y la máquina virtual que compila el código byte codes a .dex para que el dispositivo móvil lo pueda interpretar es la ART.**
- **Java API Framework:** En esta capa es donde se desarrolla el código de Java o Kotlin para crear los elementos de la interfaz de la aplicación Android, sus funciones, etc. En esta parte es donde entra el desarrollo hecho en Android Studio.
- **System Apps:** Son las aplicaciones ya descargadas en nuestro teléfono bajadas de la Play Store.



Componentes de una Aplicación Android

- **Activity:** A cada página (o pantalla) de la aplicación Android se le conoce como Activity, entre ellas existe la main Activity, que es donde empieza a ejecutarse el programa y de ahí se llama a las demás que conformen la aplicación.

Las distintas páginas de las aplicaciones Android se pueden hacer por medio de Activities o a través de fragmentos.

- **Fragment:** Los fragmentos son elementos dinámicos de las aplicaciones, esto implica que es un cuadro estático que cambia su contenido de forma dinámica, como sucede con los cuadros de Instagram que pueden mostrar distintas fotografías, pero mantienen sus dimensiones y posición, los cuadritos de YouTube que muestran distintos videos, etc.

En la actualidad en vez de usar actividades se está optando por usar fragmentos para crear las distintas páginas de las aplicaciones, pero el fragmento en estos casos ocupa toda la pantalla de la aplicación Android y es llamado **Dialog Fragment**.

Las páginas creadas en el entorno de desarrollo llamado Android Studio son responsivas, esto implica que la interfaz de la aplicación se puede adaptar a los distintos grosores de la pantalla donde se esté viendo la aplicación, ya sea si se está viendo en un celular con pantalla grande, uno con pantalla chica o hasta en una tableta.

- **Services:** Son procesos que ayudan a colocar y ejecutar una aplicación en segundo plano, osea que se siga ejecutando aun cuando el teléfono esté bloqueado. Cuando se quiera hacer que la app se siga ejecutando en segundo plano se debe crear y hacer uso de un servicio, pero los servicios nunca tendrán una interfaz de usuario, son puro código.
- **Evento:** Es una interacción que el usuario puede tener con el dispositivo, ya sea la acción de desbloquear el teléfono, bloquear el teléfono, la llegada de un mensaje de texto, la llegada o salida de una llamada telefónica, el nivel de batería baja, diversas notificaciones, etc.
- **Broadcast receivers:** Es la detección de eventos como un clic, pasar el mouse sobre un elemento, etc.
- **Content providers:** Su función es almacenar datos y compartirlos con cualquier aplicación, ya sea para acceder y compartir los contactos del celular, imágenes, videos, música, etc. Es un tipo de base de datos del teléfono o tableta.

Requisitos para Crear una Aplicación Android

- **Idea del proyecto:** Las funciones que ejecutará.
- **Tecnologías a utilizar:** Las librerías o APIs que deberán ser importadas en los archivos de build.gradle (Project: NombreProyecto) y build.gradle (Module: NombreProyecto.app) localizados en la carpeta de Gradle Scripts.
 - **XML:** Es el lenguaje de marcado utilizado para crear la interfaz de usuario de la aplicación.
 - **Kotlin:** Es el lenguaje de programación empleado para dar funciones a los distintos elementos de las interfaces gráficas de la aplicación Android.

- **Material Design:** Material Design es una librería que asiste en el diseño visual, proporcionando componentes para hacer interactivas y llamativas las aplicaciones desarrolladas con Android Studio.
- **Android Jetpack:** Jetpack es un conjunto de bibliotecas Android (arquitectura, interfaz de usuario, comportamiento y fundación) que ayudan a los desarrolladores a seguir las prácticas recomendadas, reducir el código estándar y escribir código que funcione de manera coherente en los dispositivos y las distintas versiones de Android.
- **Cloud Firestore:** Es una herramienta de base de datos perteneciente a Firebase que sirve para poder almacenar y consumir datos de la nube para mostrarlos en la aplicación Android.
- **Google Maps:** Es una parte de Google Play services que nos permite desplegar, personalizar y mostrar un mapa de Google Maps.
- **Pruebas de la aplicación:** Antes de desarrollar las interfaces se debe probar el funcionamiento de la aplicación terminada.
- **Mockups o Wireframes:** Es la maquetación de las pantallas a desarrollar, tomando en cuenta su flujo de ejecución. Se considera además la creación de imágenes de fondo y elementos estéticos que conformarán la aplicación.

Unos de los software gratuitos que se pueden usar para esto es el de Marvel donde se pueden hacer los diseños hasta a mano:

<https://marvelapp.com/>

Otro software gratuito donde se puede hacer el maquetado de pantallas es Draw.io en la opción de Blank Diagram → More Shapes → iOS UI → iPhone (Portrait):

<https://app.diagrams.net/>

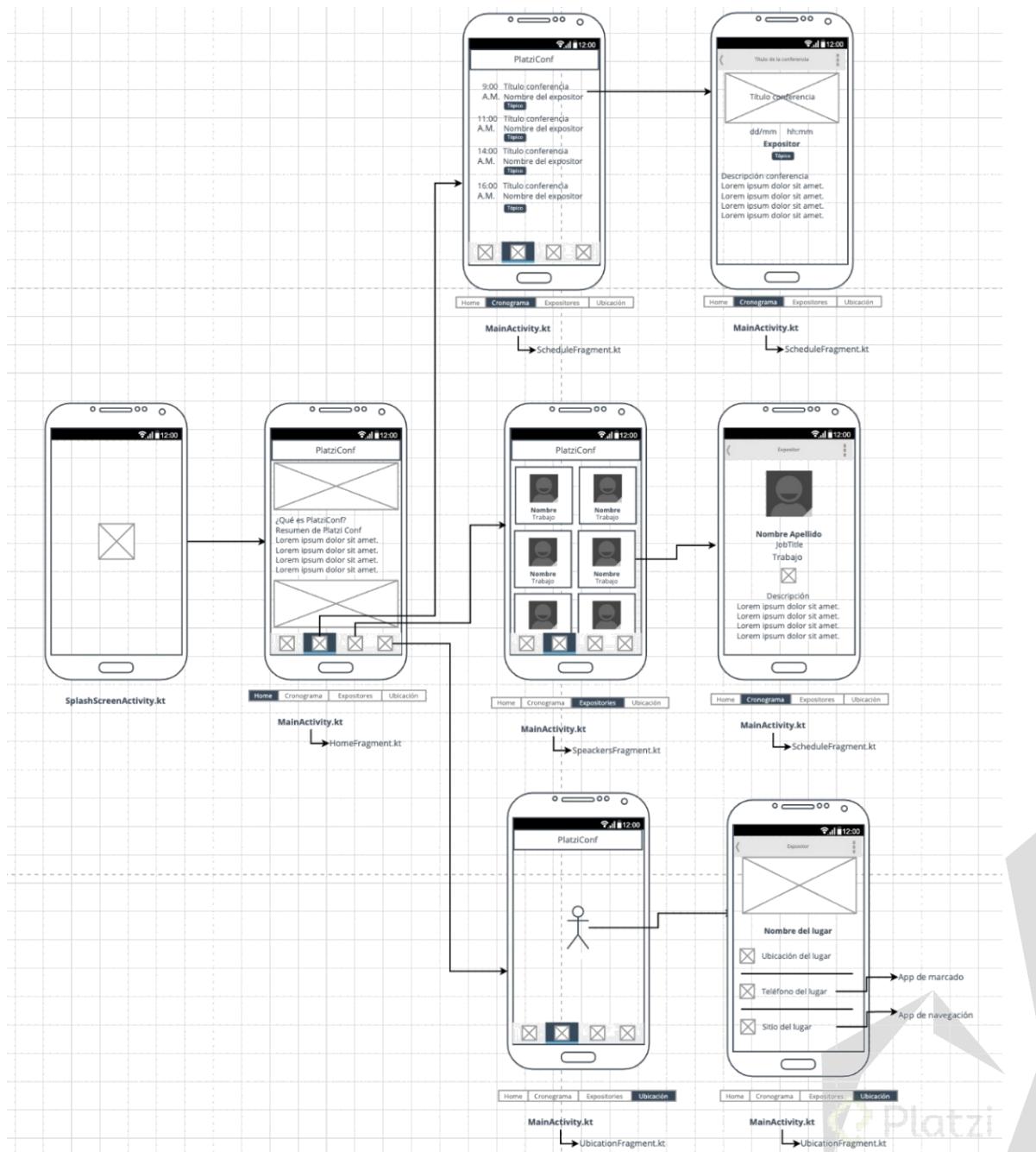
La forma en la que se estructuran los nombres y acomodo de las pantallas diseñadas en el maquetado de pantallas es la siguiente:

- Por buenas prácticas todas las **Activities** de la carpeta `app/java/com.nombreDominio.nombreProyecto` que representen las funciones de las **interfaces** creadas en la carpeta `app/res/layout` se deben nombrar empezando con una letra mayúscula, sin espacios y utilizando la palabra **Activity** hasta el final:
 - Ejemplo: `LoginActivity.kt`.
- Normalmente la forma en la que se crean las apps es por medio de dos Activities de la siguiente manera:
 1. La primera pantalla es una **Activity**, puede ser la pantalla de Login o una que tenga una animación del logo de la empresa.
 2. Luego el usuario se introduce en la aplicación donde ya se muestra el menú que te permite navegar por medio de la aplicación, **en esta segunda Activity de la aplicación es donde se muestran las distintas partes de la aplicación en forma de Fragments**, que son subpantallas contenidas en una Activity y harán que se muestren las distintas pantallas a las que te dirige el menú de la app.
 - Por buenas prácticas todos los **Fragments** de la carpeta `app/java/com.nombreDominio.nombreProyecto` que representen las

funciones de las subpantallas creadas para una Activity dentro de la carpeta app/res/layout se deben nombrar empezando con una letra mayúscula, sin espacios y utilizando la palabra Fragment hasta el final:

- Ejemplo: HomeFragment.kt

En los diagramas de Mockups o Wireframes utilizados para mostrar el flujo de la aplicación Android se coloca una imagen de la pantalla mostrando más o menos que es lo que hace o contiene cada una, debajo se muestra el nombre de su Activity y si esta contiene Fragments, se muestra el nombre del fragmento debajo del nombre de la Activity a la que pertenece, ya que recordemos que el Fragmento es una subpantalla de la Activity.





<https://www.figma.com>

Referencias:

Platzi, Gustavo Lizárraga, “Curso de Kotlin para Android”, [Online], Available: <https://platzi.com/clases/1836-kotlin-android/26986-por-que-desarrollar-para-android-usando-kotlin/>

