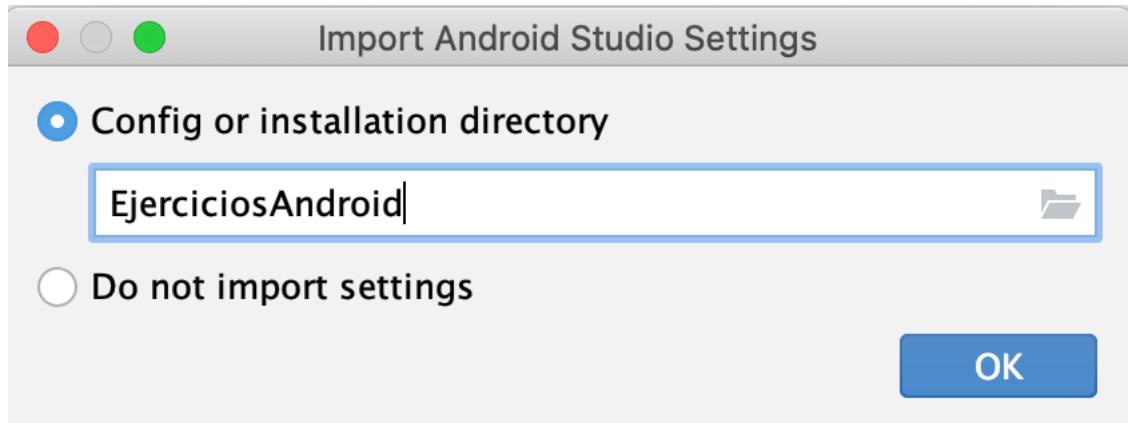
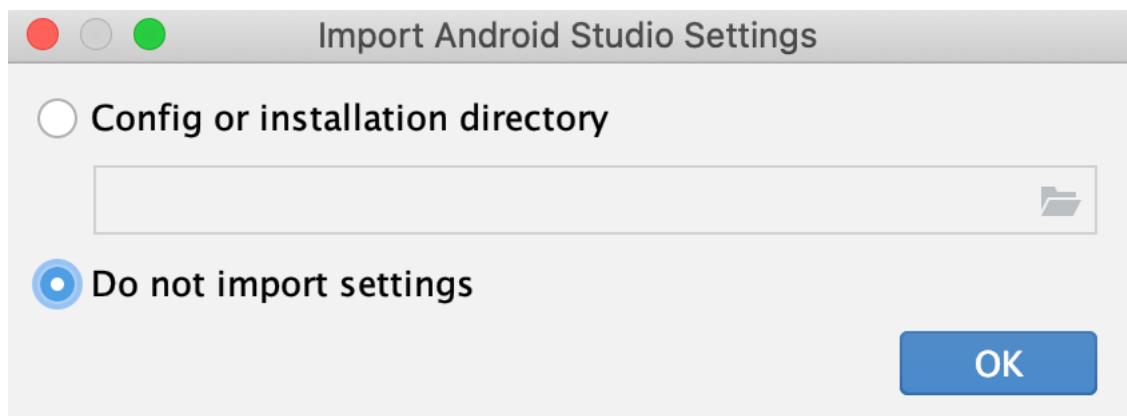


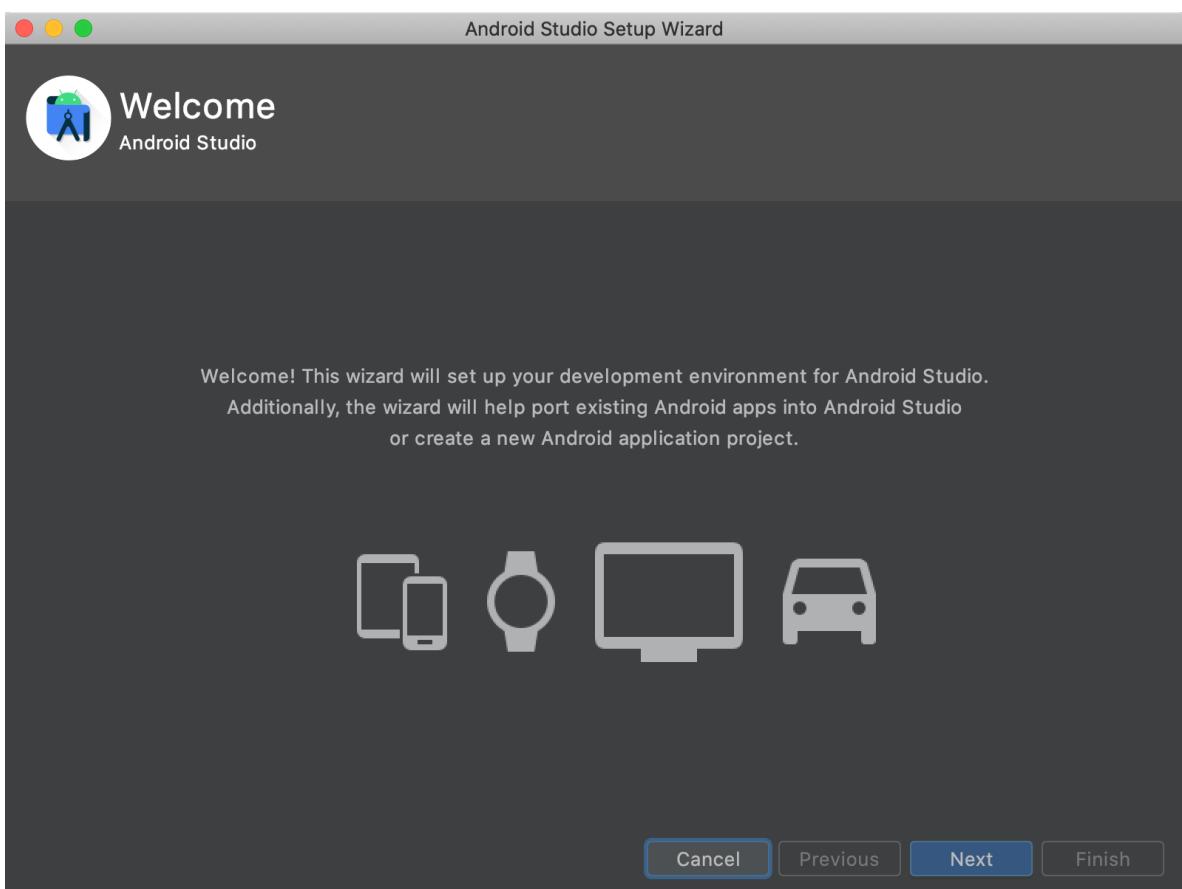
PROGRAMACIÓN PARA DISPOSITIVOS MÓVILES

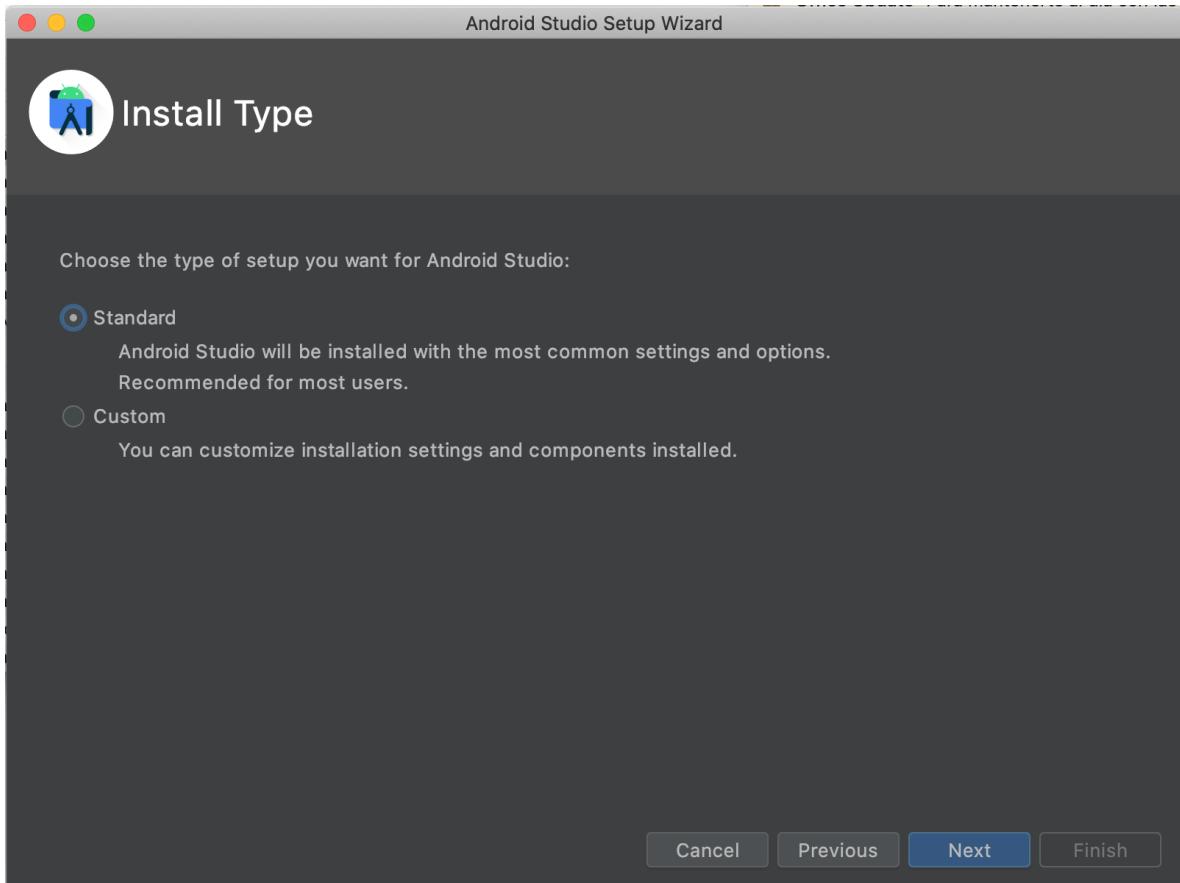
EJERCICIO. Instalación

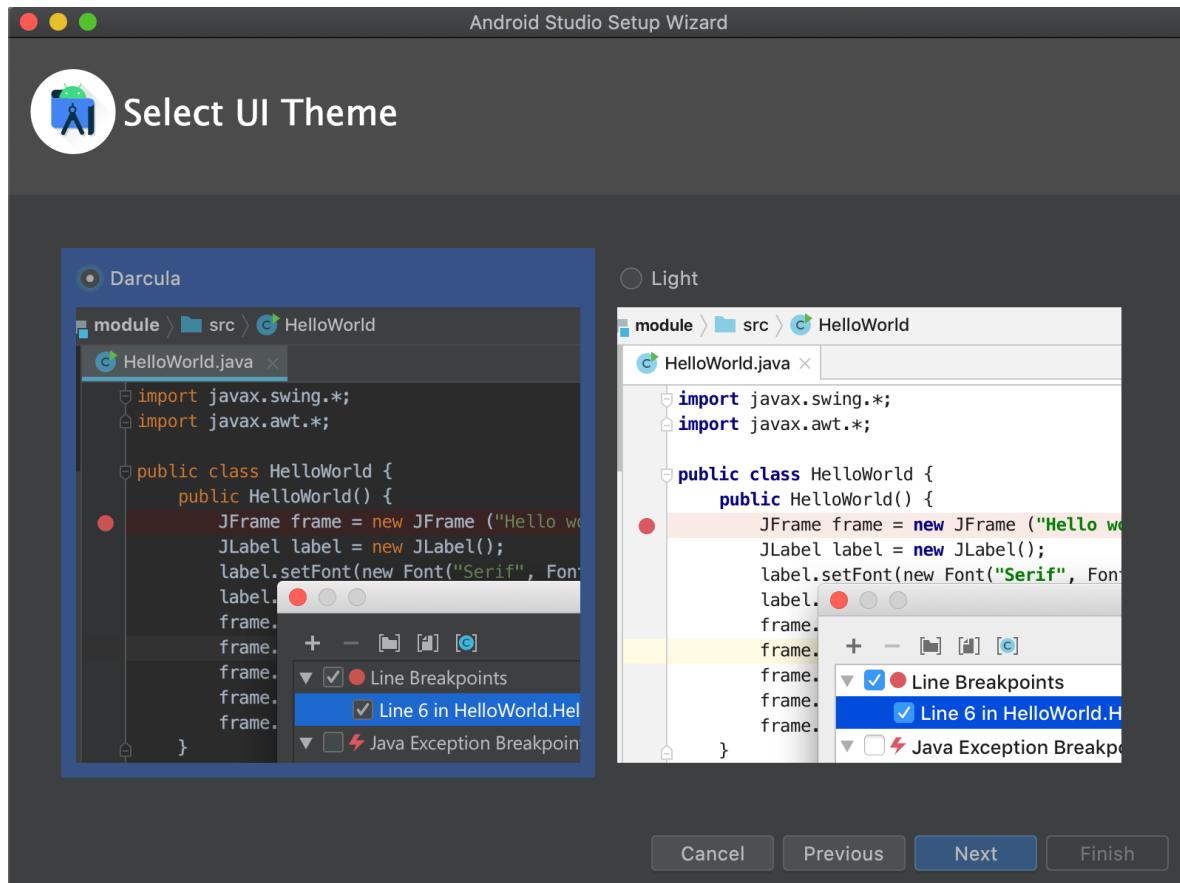


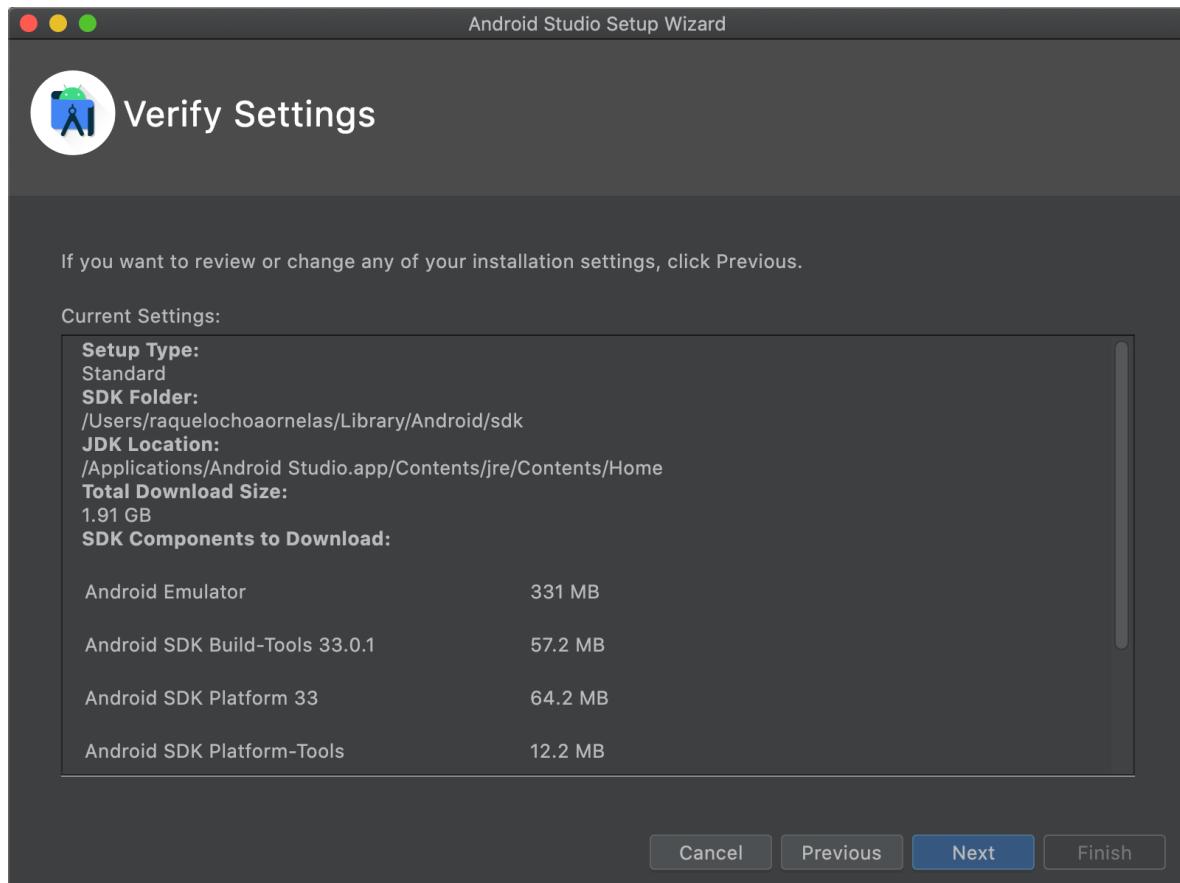
1. Seleccionar la plantilla predeterminada para celular o tableta.
2. Elegir la plantilla sencilla que no traiga código atogenerado (En blanco)
3. Proporcionar el nombre del proyecto.
4. Por defecto aparece example, hay que cambiarlo. Ya que representa la url que define la ubicación en la que se crea la aplicación. Y define la url de la aplicación para que se descargue desde una página web.
5. Seleccionar Lenguaje Java
6. Elegir mínimo SDK (las librerías actuales no son compatibles para versiones viejas) encontrar un equilibrio con la versión 4.1 que es compatible con el 99.8% de los dispositivos.

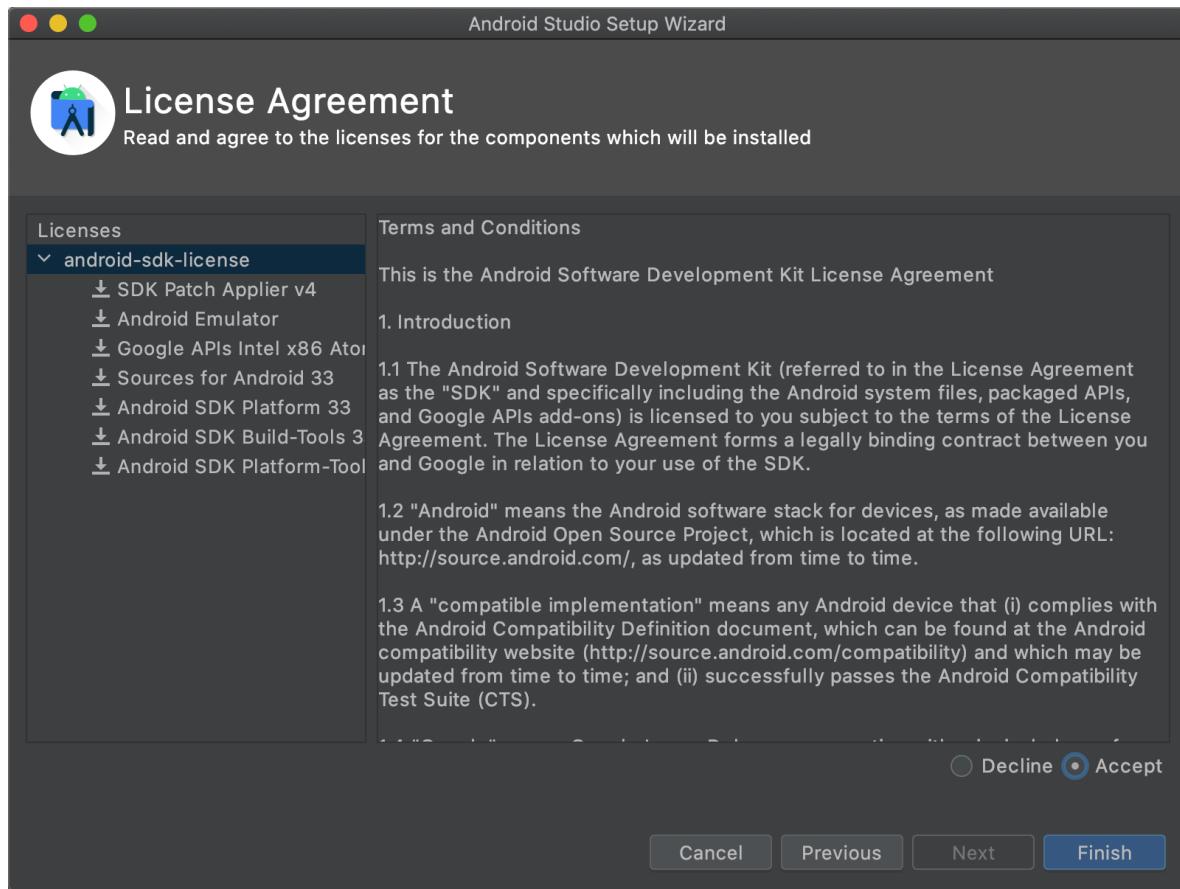


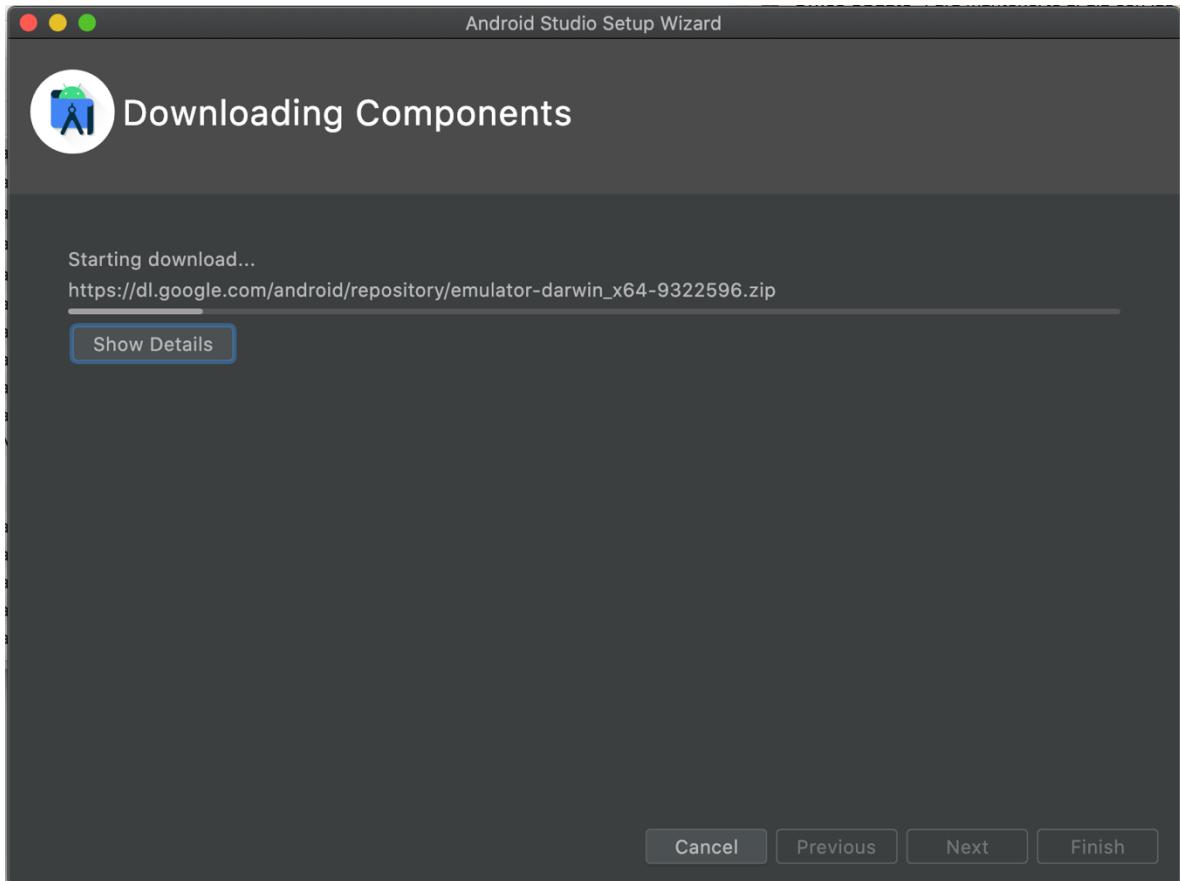


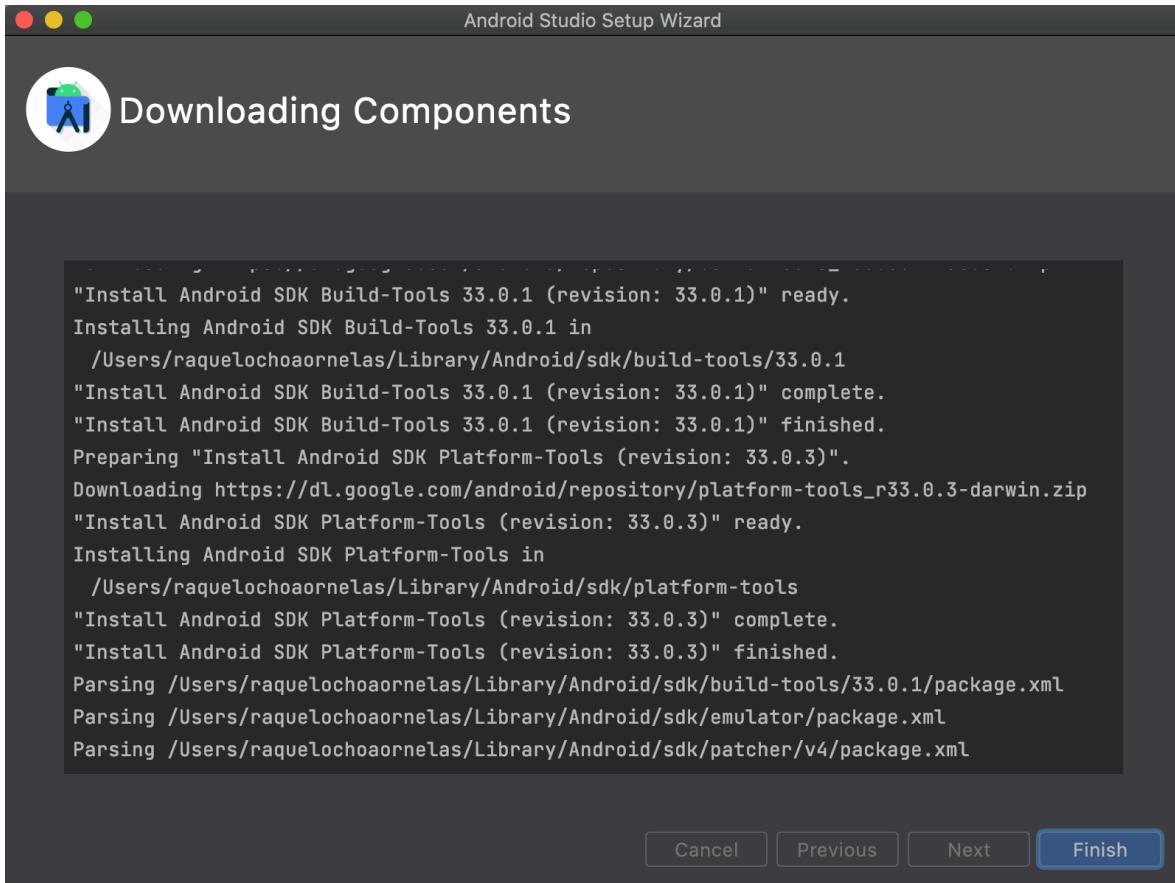


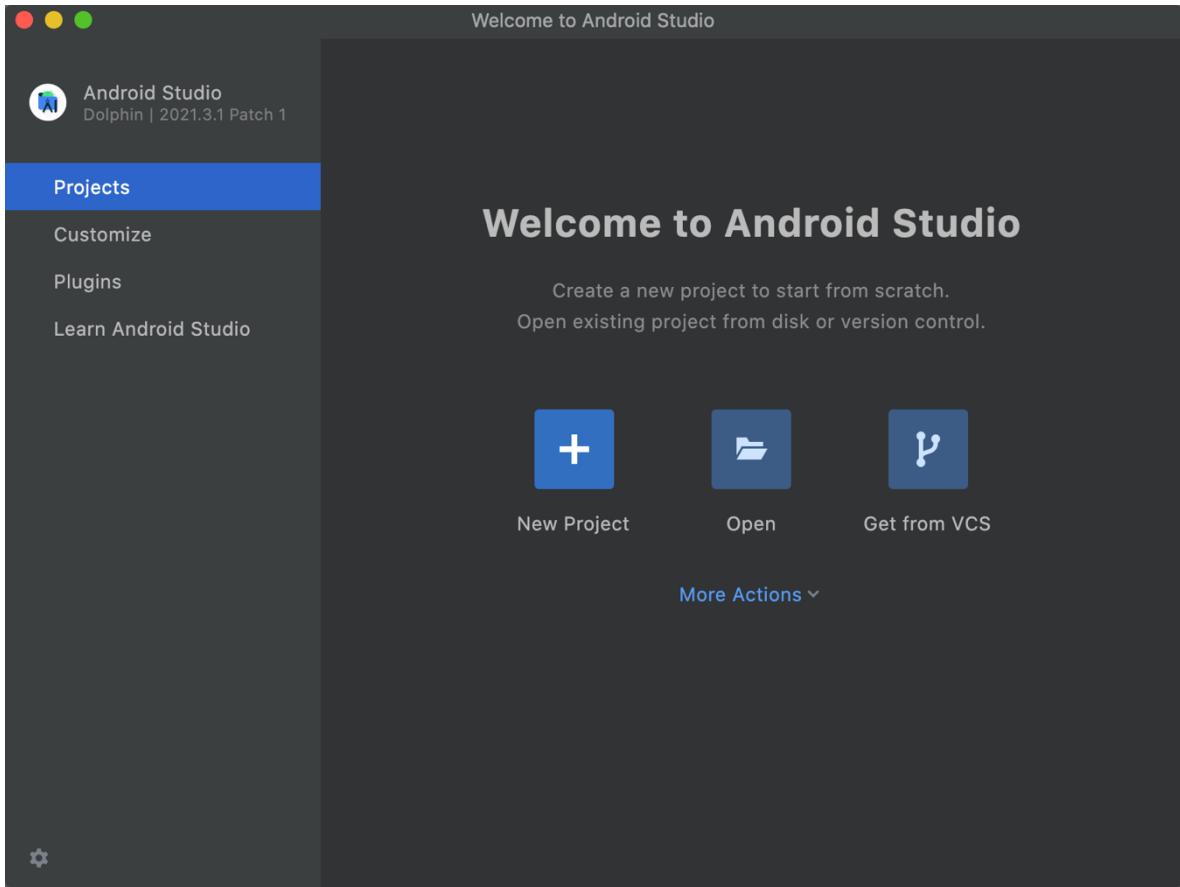


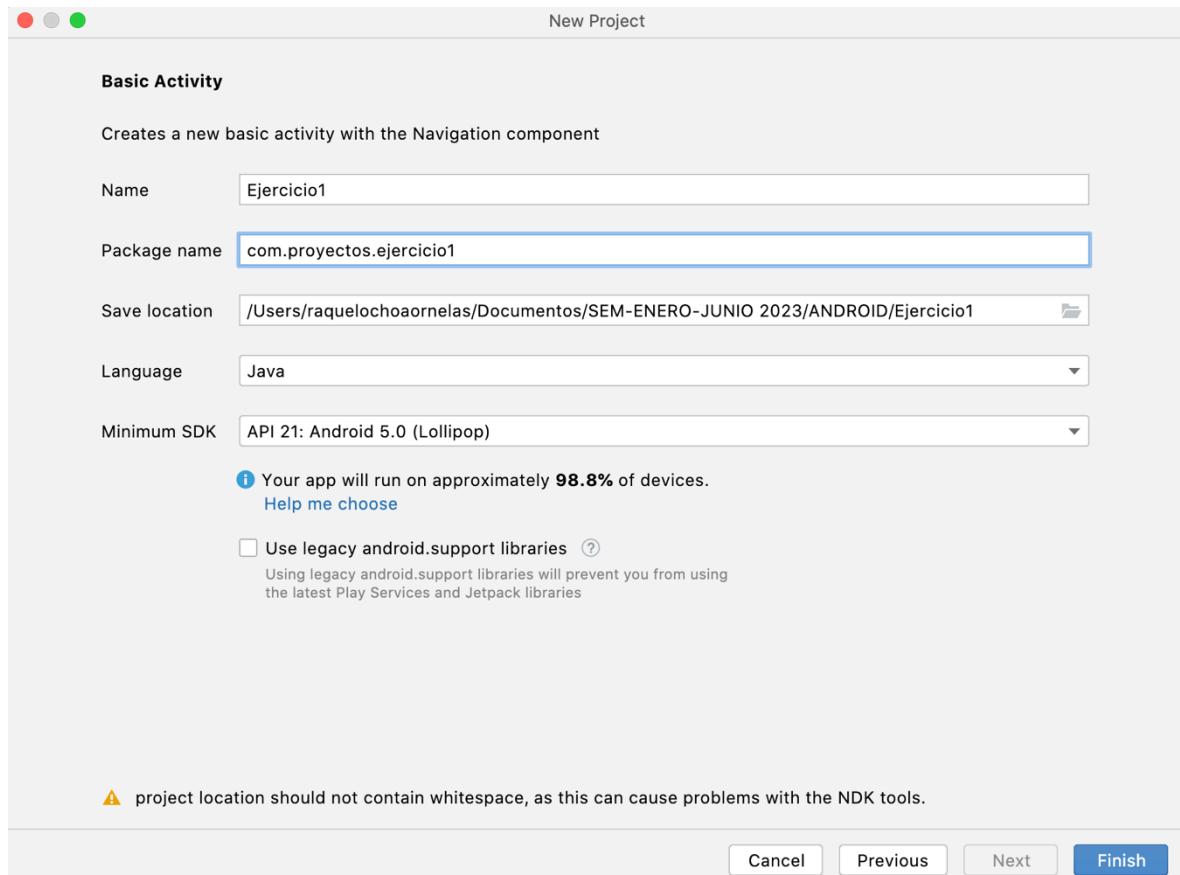


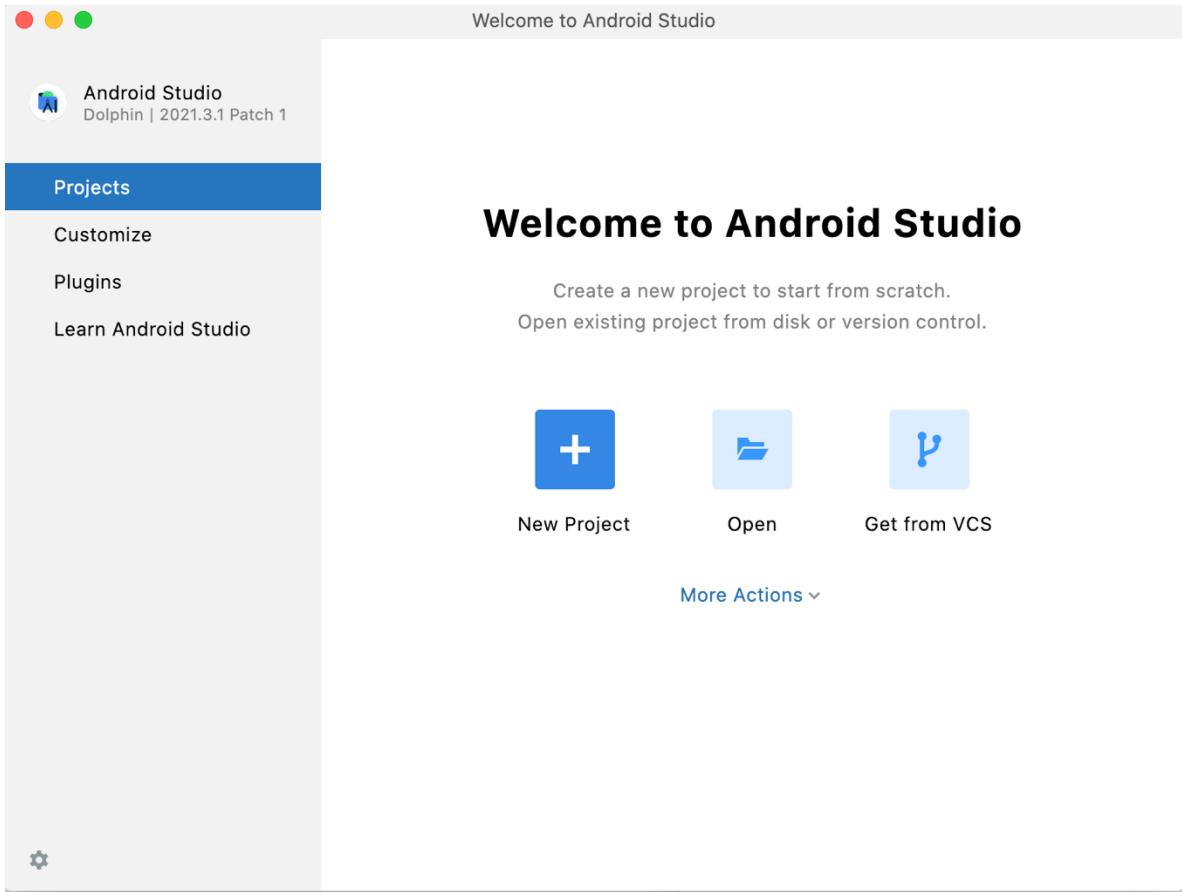


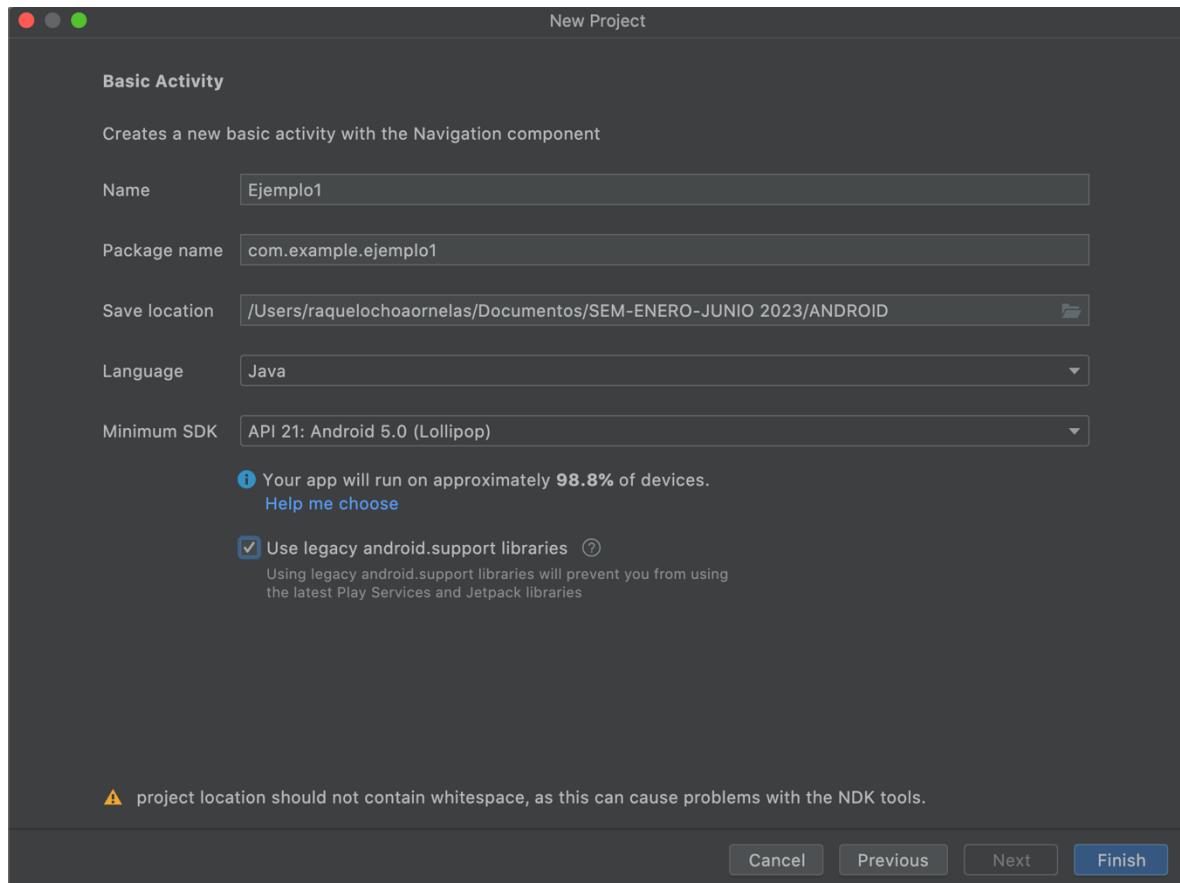


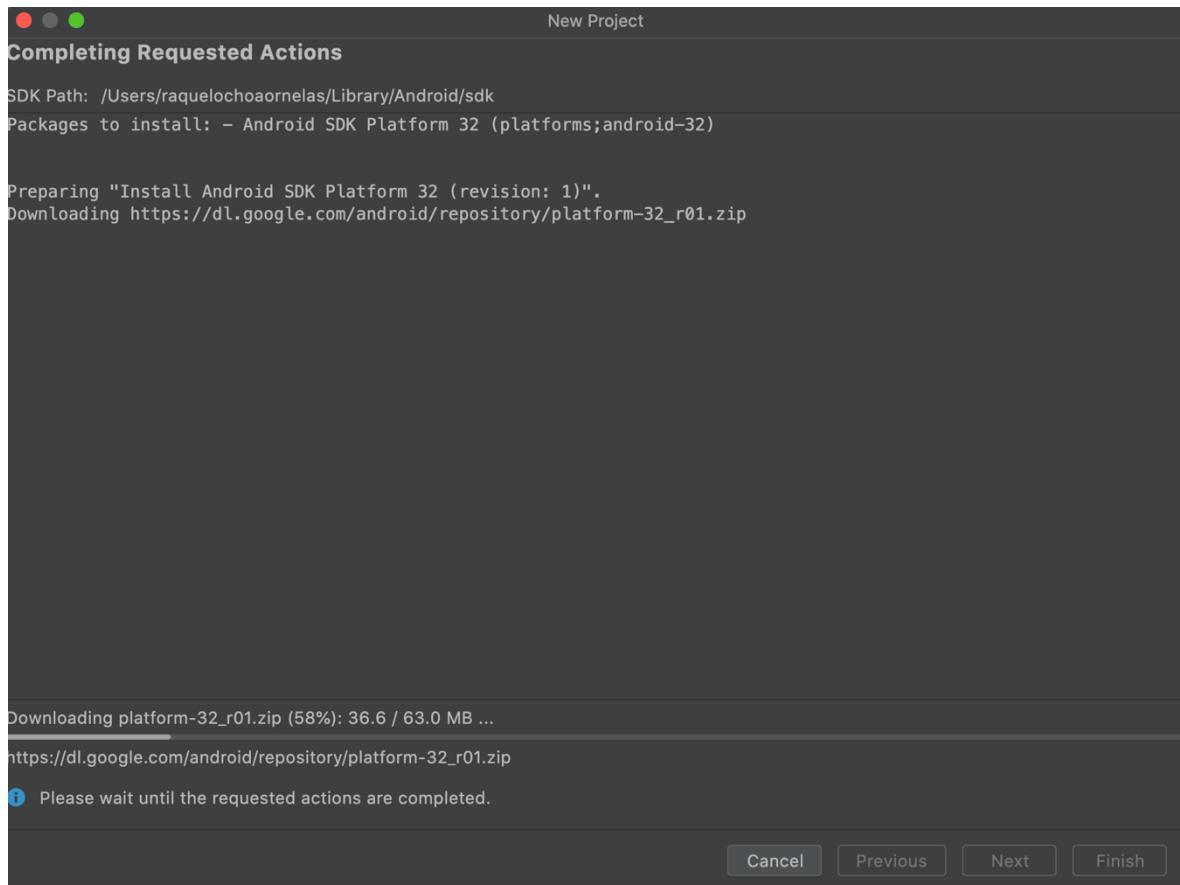


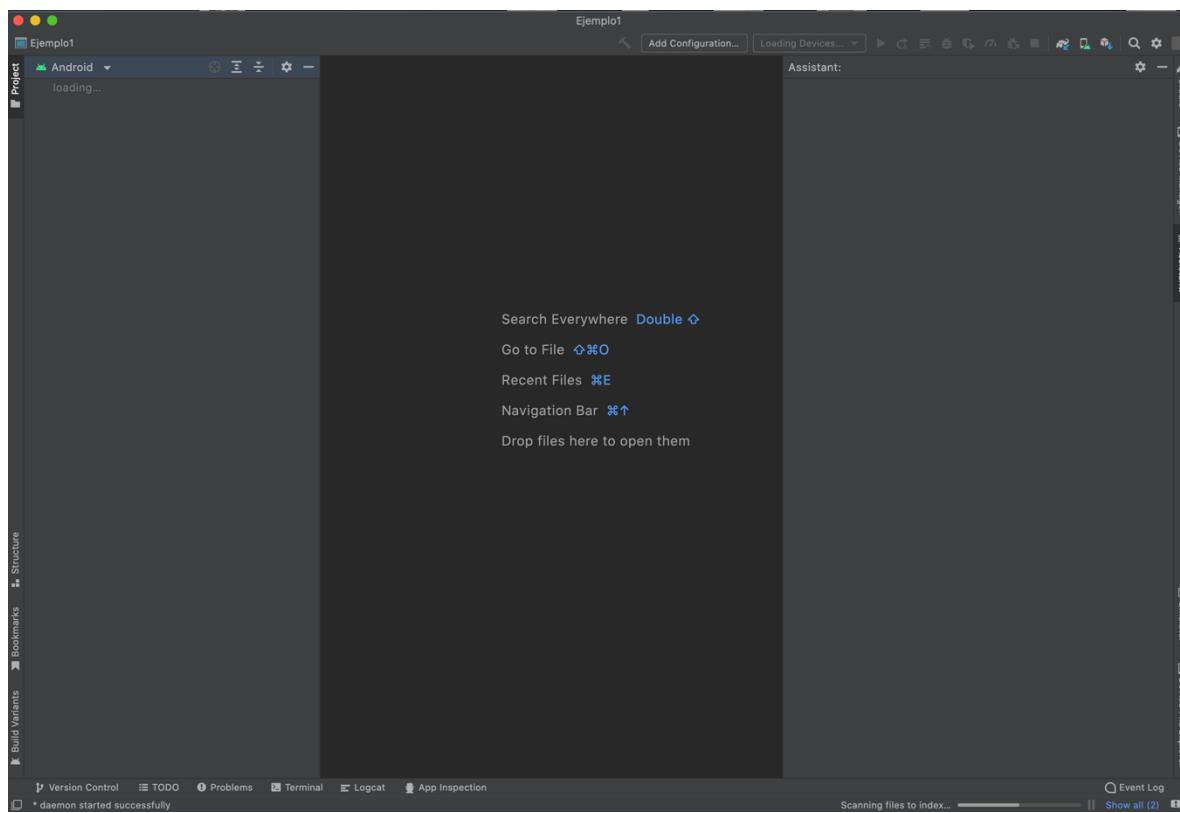
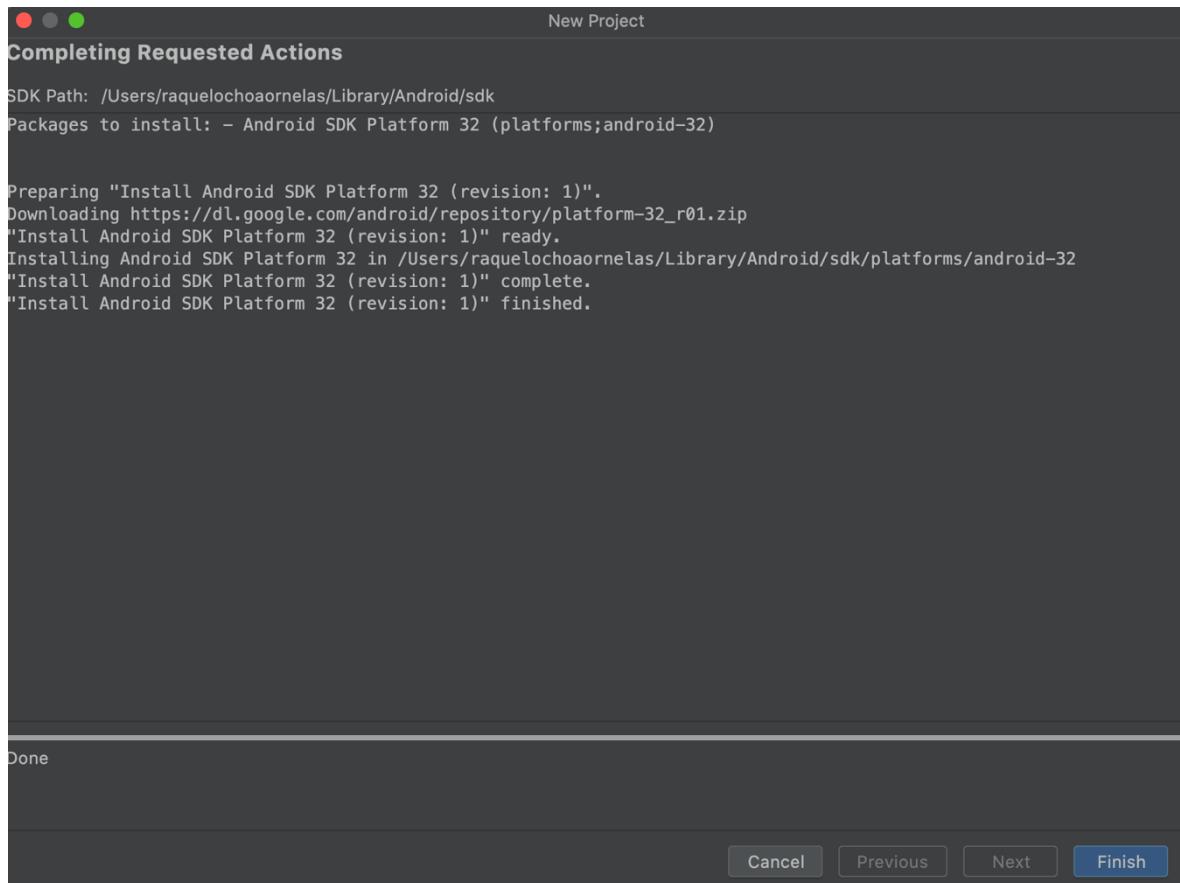




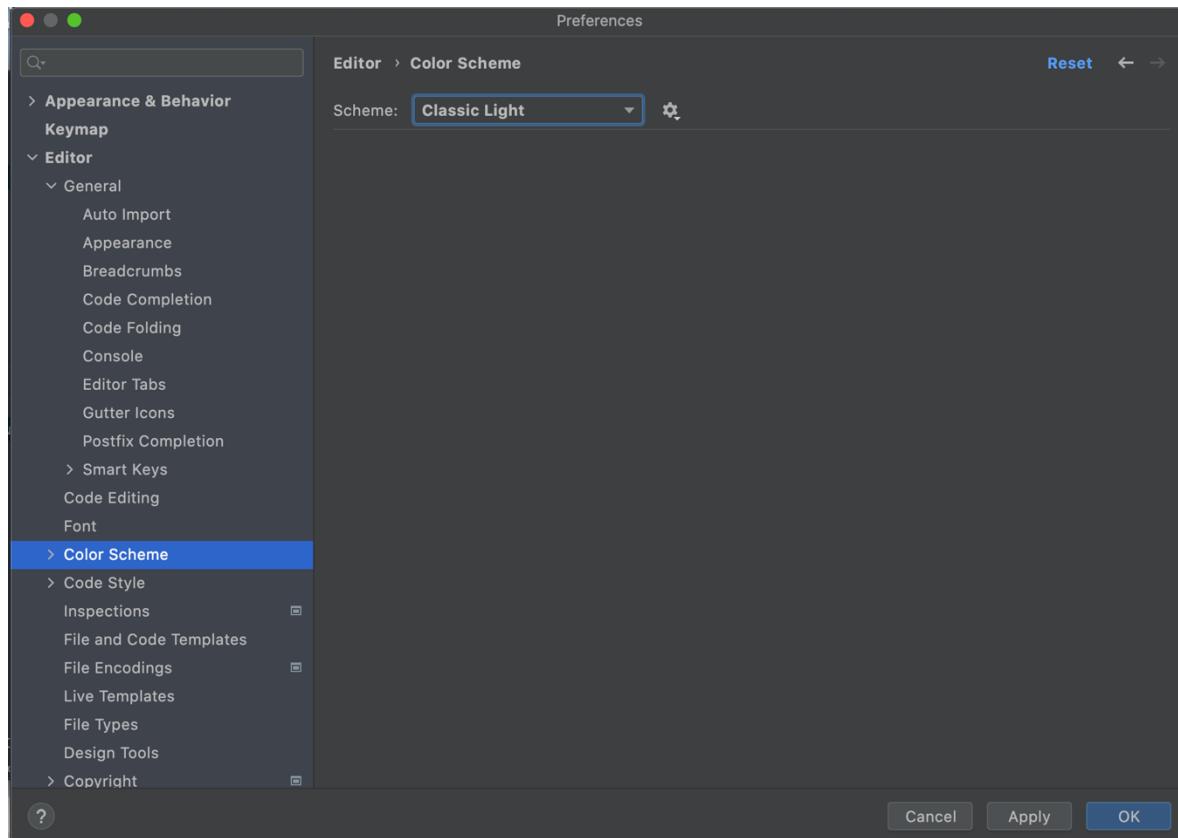






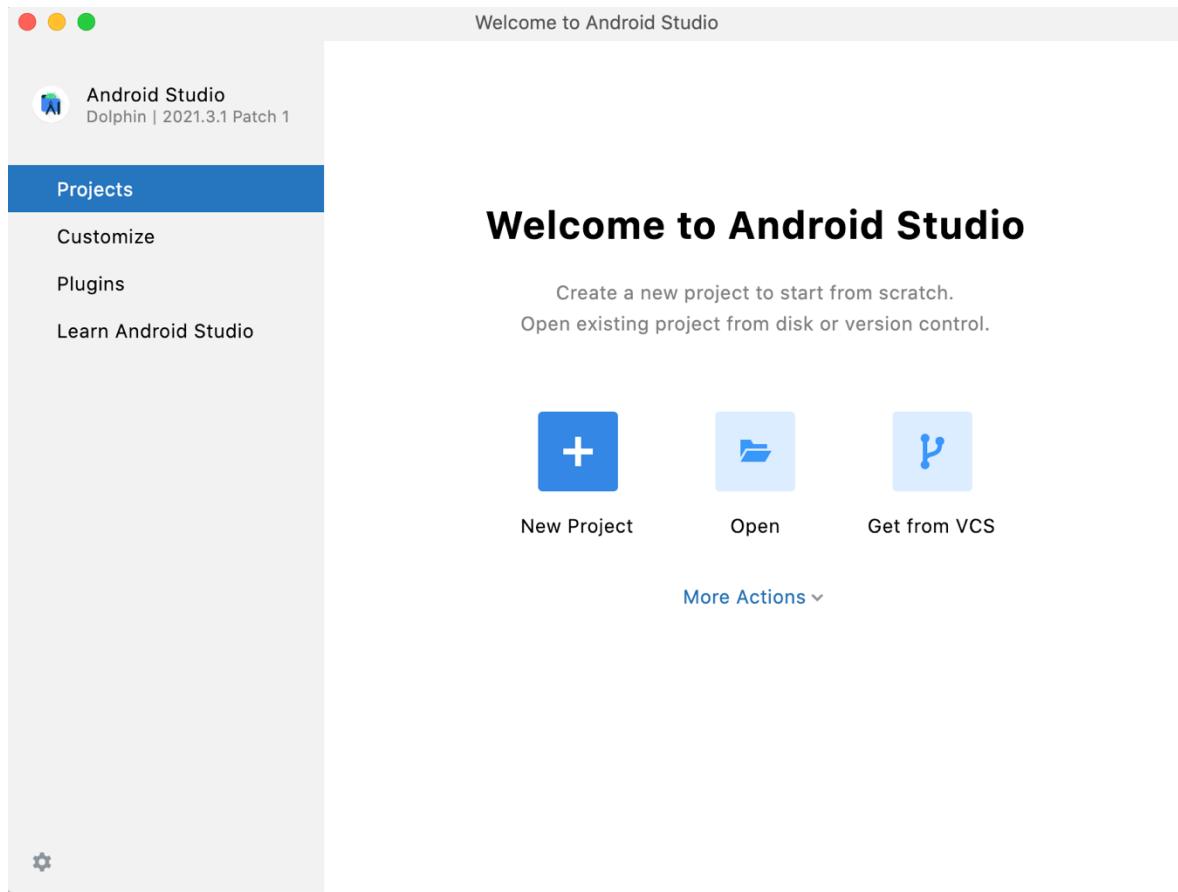


Para cambiar el color de esquema dar clic en la equina superior izquierda del MainActivity y seleccionar Configure Editor Tabs y posteriormente:

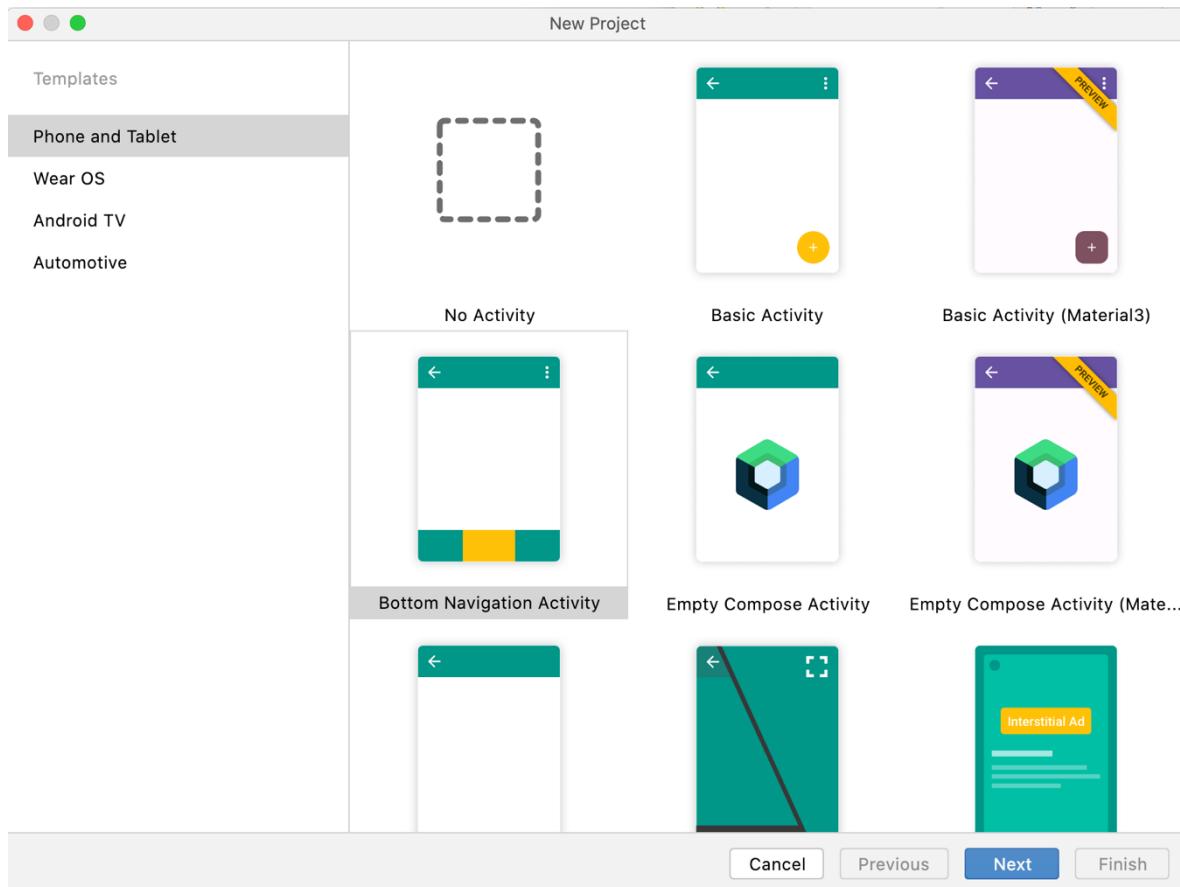


EJEMPLO 1.

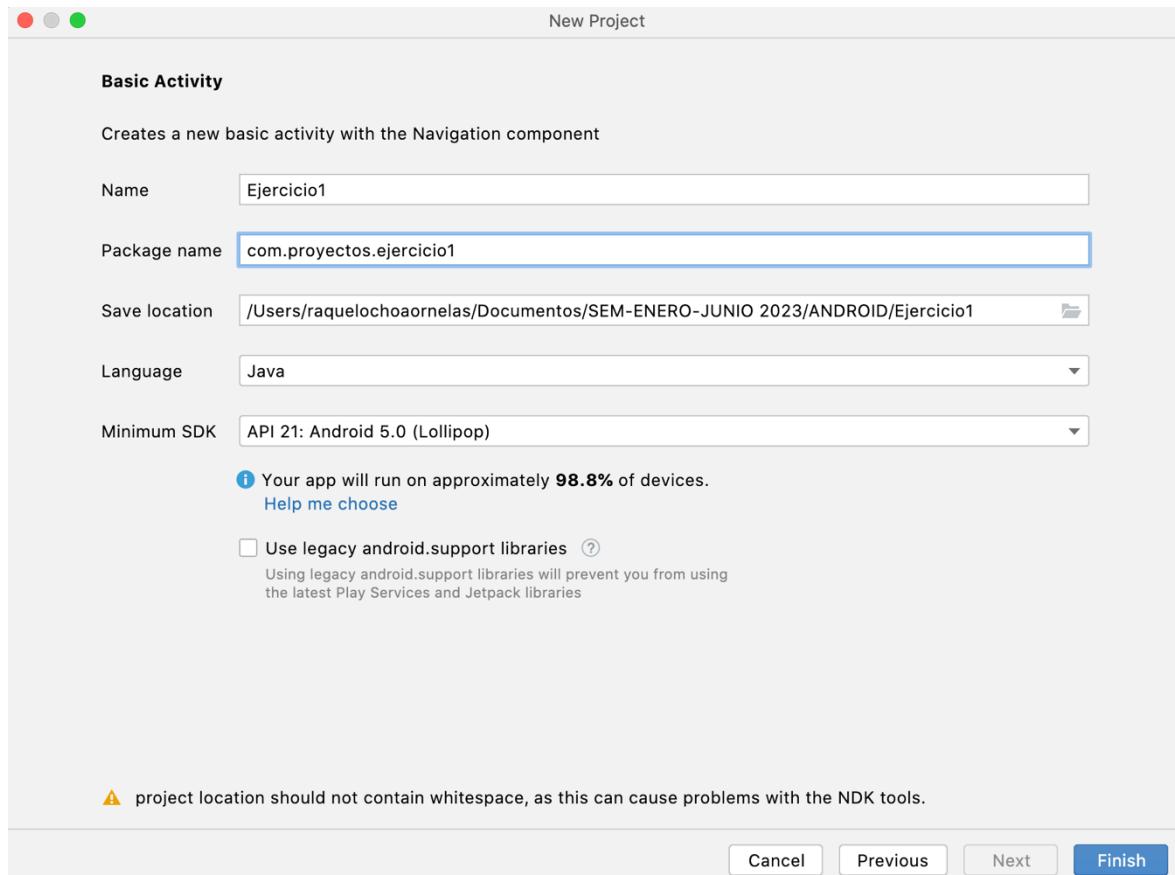
1. Seleccionar nuevo proyecto



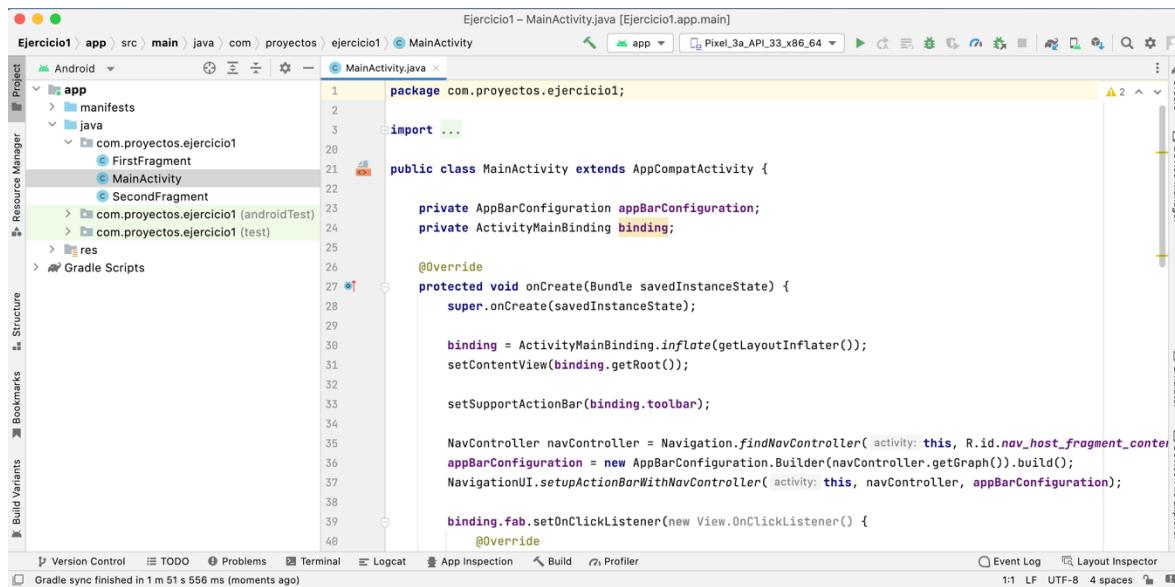
2. Seleccionar plantilla básica



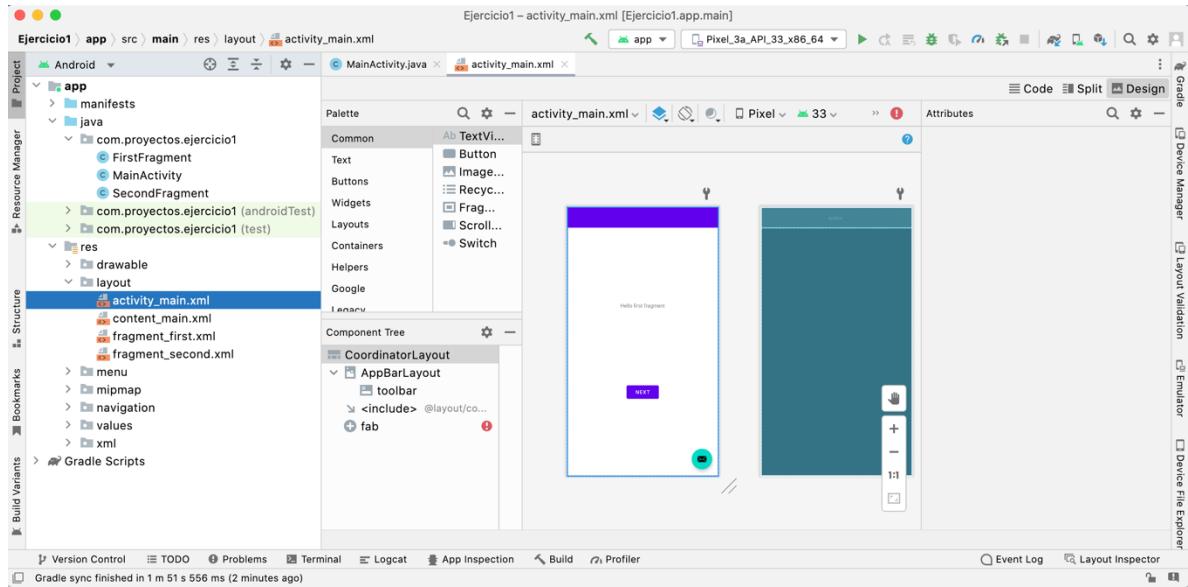
3. Asignar el nombre del proyecto y paquete:



4. Se presenta la estructura del proyecto

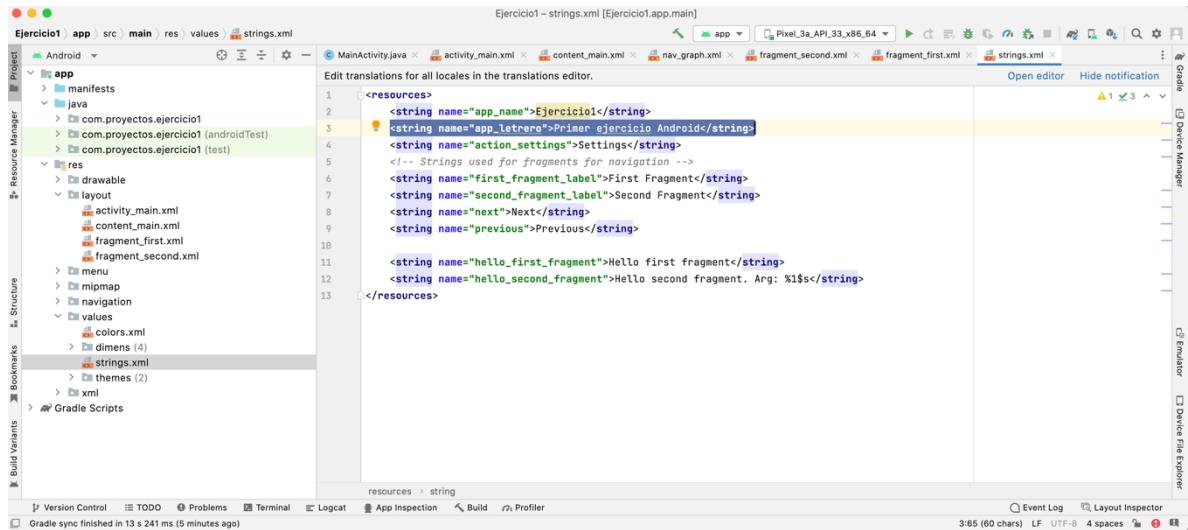


5. En la carpeta res se localiza layout, abrirla y seleccionar activity_main.xml:

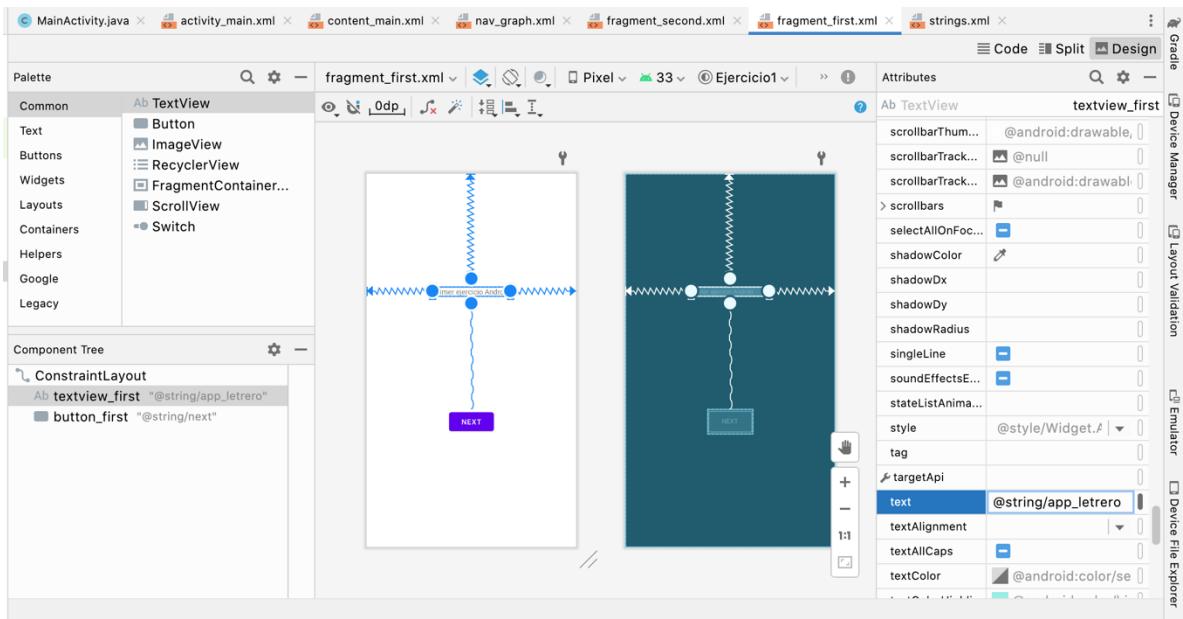


Se pueden agregar y anclar nuevos elementos en código o en diseño. Si se hace en diseño, seleccionar a los elementos en la sección azul para que sean anclados y no queden ubicados de manera aleatoria.

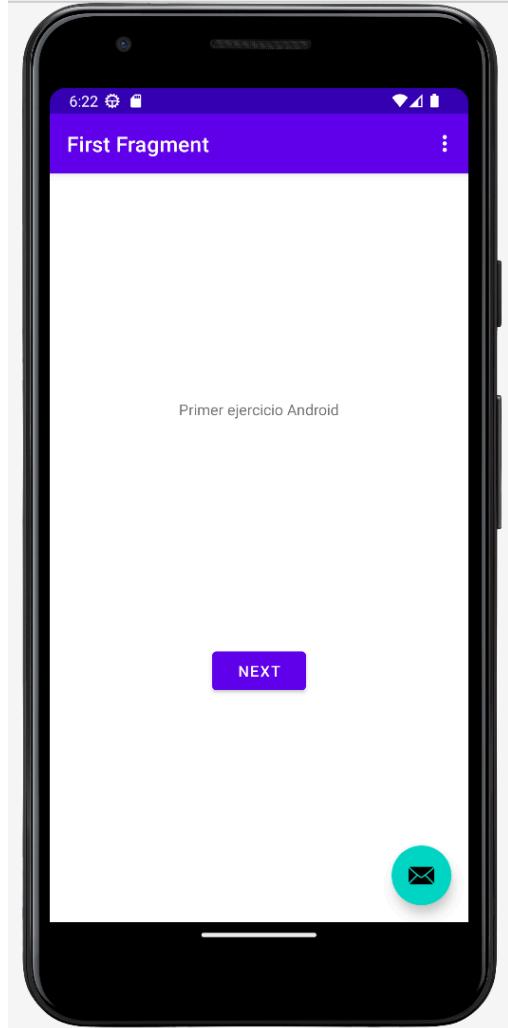
- En values->string se declaran las constantes. Ir a esta sección y agregar la siguiente constante:



- Asignar a la propiedad Text del textView el valor de app_letrero definida como constante anteriormente:



8. Ejecutar la aplicación:



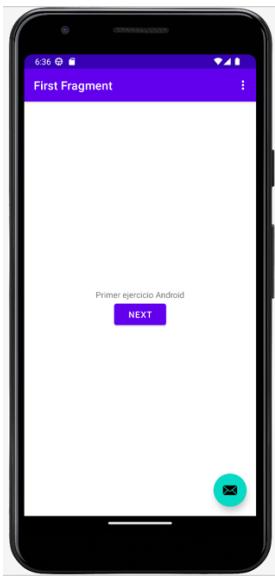
9. Cambiar ConstraintLayout por LinearLayout, en split editar el Código y cambiarlo:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".FirstFragment">
    <Text...>
        No documentation found.
    <Text...
        android:id="@+id/textview_first"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Primer ejercicio Android"
        app:layout_constraintBottom_toTopOf="@+id/button_fi
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <Button
        android:id="@+id/button_first"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Next"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</LinearLayout>
```

10. Ejecutarlo nuevamente y ver que los elementos se cargan sin ningún orden.



11. Seleccionar el LinearLayout y asignar en la propiedad orientation el valor de vertical y en gravity el valor de center. Ejecutar de nuevo.



EJERCICIO 2 LinearLayout

Si se quiere combinar varios elementos de tipo vista se tiene que utilizar un objeto de tipo Layout. Un Layout es un contenedor de una o más vistas y controla su comportamiento y posición. Hay que destacar que un Layout puede contener a otro Layout y que es un descendiente de la clase View.

La siguiente lista describe los Layout más utilizados en Android:

LinearLayout: Dispone los elementos en una fila o en una columna.

TableLayout: Distribuye los elementos de forma tabular.

RelativeLayout: Dispone los elementos en relación a otro o al padre.

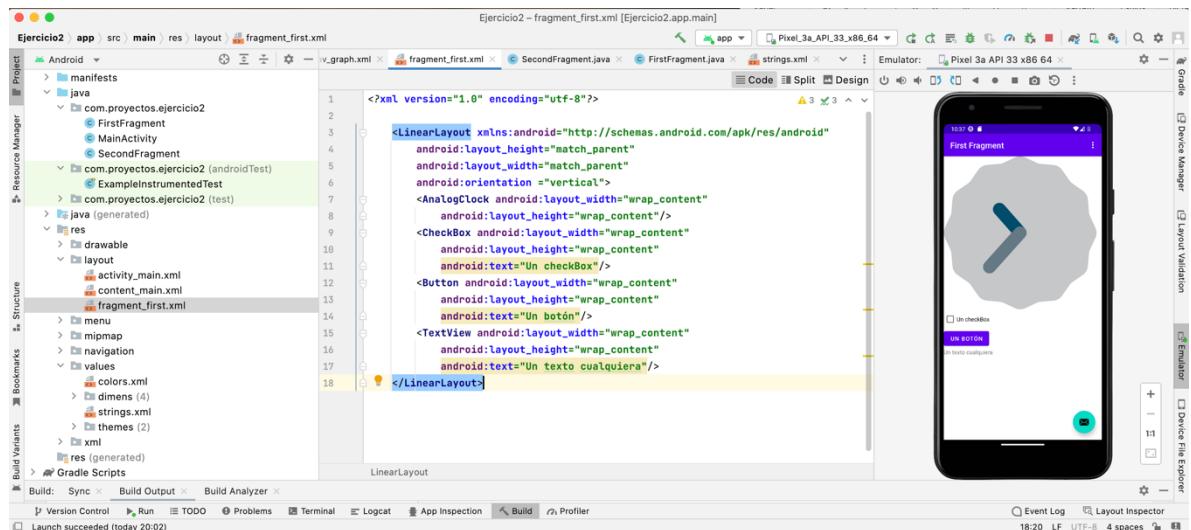
AbsoluteLayout: Posiciona los elementos de forma absoluta.

FrameLayout: Permite el cambio dinámico de los elementos que contiene.

ConstraintLayout: Versión mejorada de RelativeLayout, que permite una edición visual desde el editor y trabajar con porcentajes.

Este layout apila uno tras otro todos sus elementos hijos de forma horizontal o vertical según se establezca su propiedad android:orientation.

1. Crear un nuevo proyecto con nombre Ejercicio2
2. Eliminar de res->layout el archivo fragment_second.xml
3. Escribir el siguiente código en res->layout del archivo fragment_first.xml



4. Eliminar el código relacionado con el fragmento eliminado en el archivo FirstFragment.java:

```
package com.proyectos.ejercicio2;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
```

```

import android.view.ViewGroup;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;
import androidx.navigation.fragment.NavHostFragment;

import com.proyectos.ejercicio2.databinding.FragmentFirstBinding;

public class FirstFragment extends Fragment {

    private FragmentFirstBinding binding;

    @Override
    public View onCreateView(
        LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState
    ) {

        binding = FragmentFirstBinding.inflate(inflater, container,
false);
        return binding.getRoot();

    }

    public void onViewCreated(@NonNull View view, Bundle
savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);

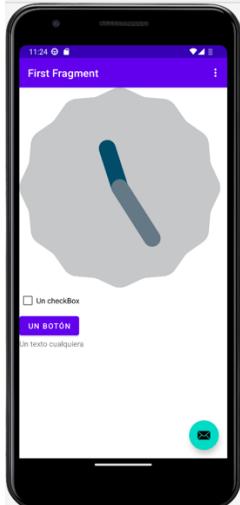
    }

    @Override
    public void onDestroyView() {
        super.onDestroyView();
        binding = null;
    }

}

```

5. Ejecutar la aplicación.



EJERCICIO 3 TableLayout

Un TableLayout permite distribuir sus elementos hijos de forma tabular, definiendo las filas y columnas necesarias, y la posición de cada componente dentro de la tabla.

La estructura de la tabla se define de forma similar a como se hace en HTML, es decir, indicando las filas que compondrán la tabla (objetos TableRow), y dentro de cada fila las columnas necesarias, con la salvedad de que no existe ningún objeto especial para definir una columna (algo así como un TableColumn) sino que directamente se insertan los controles necesarios dentro del TableRow y cada componente insertado (que puede ser un control sencillo o incluso otro ViewGroup) corresponderá a una columna de la tabla. De esta forma, el número final de filas de la tabla se corresponderá con el número de elementos TableRow insertados, y el número total de columnas quedará determinado por el número de componentes de la fila que más componentes contenga.

Por norma general, el ancho de cada columna se corresponderá con el ancho del mayor componente de dicha columna, pero existen una serie de propiedades que ayudarán a modificar este comportamiento:

android:stretchColumns: Indicará las columnas que pueden expandir para absorver el espacio libre dejado por las demás columnas a la derecha de la pantalla.

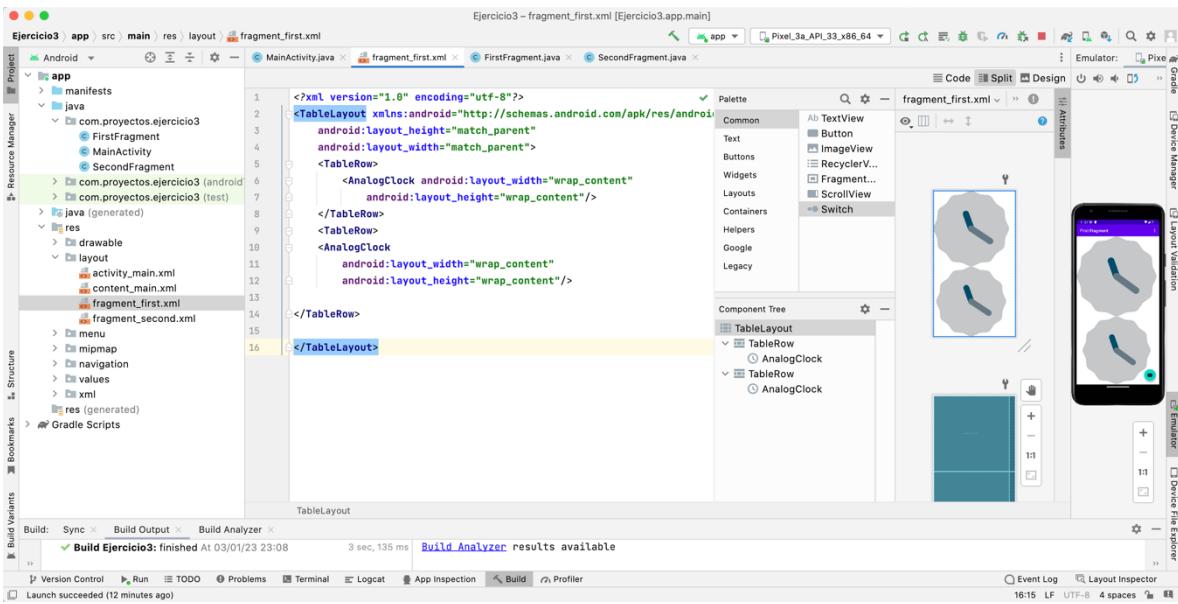
android:shrinkColumns: Indicará las columnas que se pueden contraer para dejar espacio al resto de columnas que se puedan salir por la derecha de la pantalla.

android:collapseColumns: Indicará las columnas de la tabla que se quieren ocultar completamente.

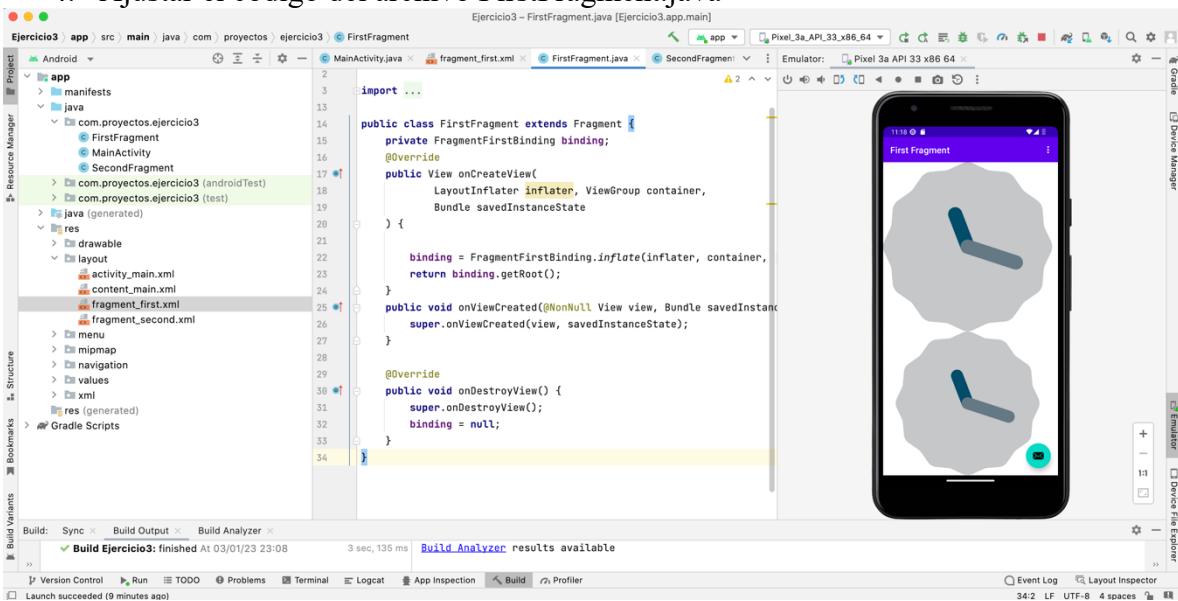
Todas estas propiedades del TableLayout pueden recibir una lista de índices de columnas separados por comas (ejemplo: android:stretchColumns="1,2,3") o un asterisco para indicar que debe aplicar a todas las columnas (ejemplo: android:stretchColumns="*").

Otra característica importante es la posibilidad de que una celda determinada pueda ocupar el espacio de varias columnas de la tabla (análogo al atributo colspan de HTML). Esto se indicará mediante la propiedad android:layout_span del componente concreto que deberá tomar dicho espacio.

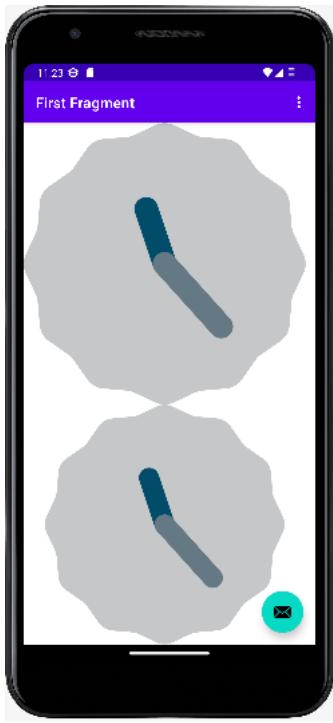
1. Crear un nuevo proyecto de nombre Ejercicio3.
2. Eliminar de res->layout el archivo fragment_second.xml
3. Escribir el siguiente código en res->layout del archivo fragment_first.xml



4. Ajustar el código del archivo FirstFragment.java



5. Ejecutar la aplicación.



EJERCICIO 4. TableLayout

1. Crear un nuevo proyecto de nombre Ejercicio4.
2. Eliminar de res->layout el archivo fragment_second.xml
3. Escribir el siguiente código en res->layout del archivo fragment_first.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_height="match_parent"
    android:layout_width="match_parent">
    <TableRow
        android:id="@+id/TableRow01">
        <TextView
            android:id="@+id/TextView01"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Nombre"
            android:width="100sp" />
        <EditText
            android:id="@+id/campo_nombre"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:maxLength="10"
            android:hint="Texto de entrada"
            />
    </TableRow>
    <TableRow>
```

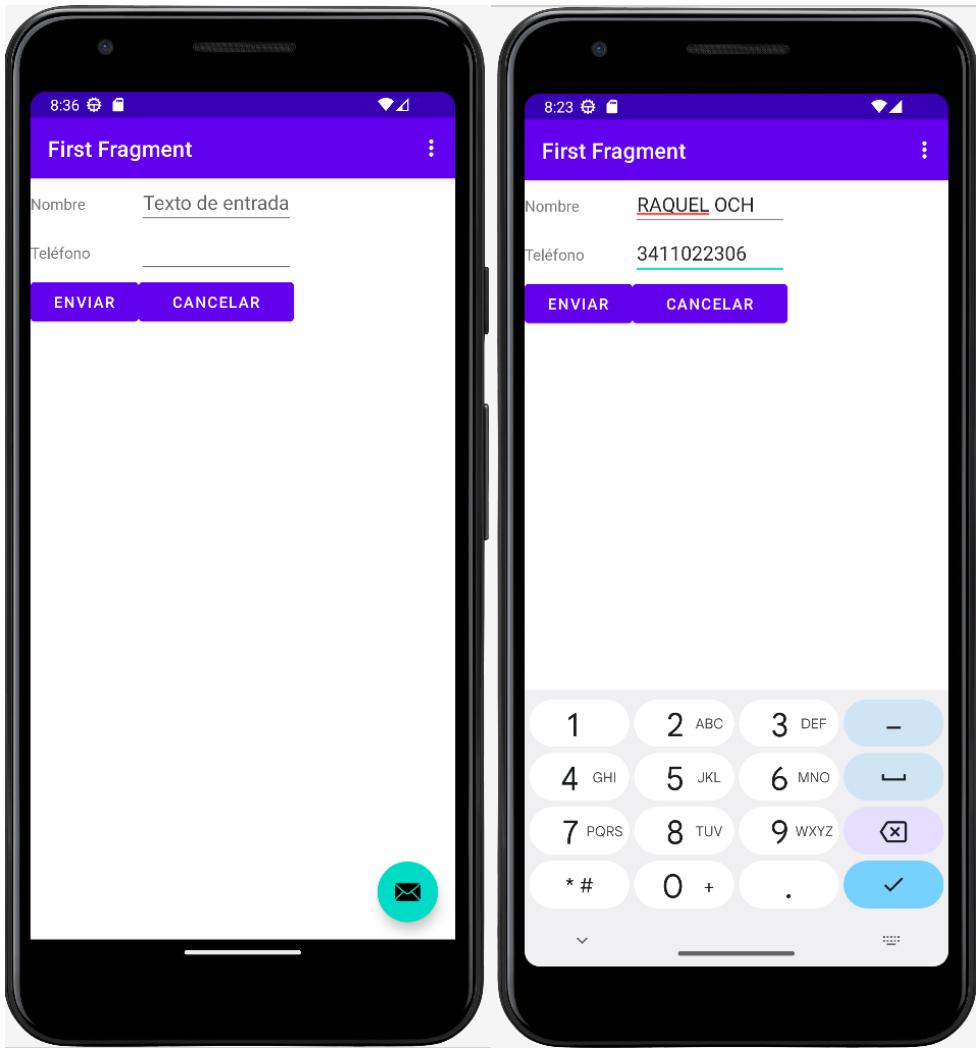
```

    android:id="@+id/TableRow02">
        <TextView
            android:id="@+id/TextView02"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Teléfono"

        />
        <EditText
            android:id="@+id/campo_tel"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:inputType="phone"
            android:maxLength="10"
        />
    </TableRow>
    <TableRow android:id="@+id/TableRow03">
        <Button android:id="@+id/Button01"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Enviar"
        />
        <Button android:id="@+id/Button02"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Cancelar"
        />
    </TableRow>
</TableLayout>

```

4. Eliminar el archivo SecondFragment.
5. Ajustar el código del archive FirstFragment.java
6. Ejecutar la aplicación:



EJERCICIO 5 RelativeLayout

Este layout permite especificar la posición de cada elemento de forma relativa a su elemento padre o a cualquier otro elemento incluido en el propio layout. De esta forma, al incluir un nuevo elemento X se puede indicar por ejemplo que debe colocarse debajo del elemento Y y alineado a la derecha del layout padre. Veamos esto en el ejemplo siguiente:

RelativeLayout es un grupo de vistas que muestra vistas secundarias en posiciones relativas. La posición de cada vista puede especificarse como relativa a elementos del mismo nivel (como a la izquierda de otra vista o por debajo de ella) o en posiciones relativas al área RelativeLayout superior (como alineada a la parte inferior, izquierda o central).

Nota: Para un mejor rendimiento y asistencia con las herramientas, debes crear tu diseño con ConstraintLayout.



RelativeLayout es una utilidad muy eficaz para diseñar una interfaz de usuario porque puede eliminar grupos de vistas anidados y conservar la estabilidad de la jerarquía de diseño, lo que mejora el rendimiento. Si se usa varios grupos de LinearLayout anidados, quizás se pueda reemplazar por un grupo RelativeLayout individual.

Cómo posicionar vistas

RelativeLayout permite que vistas secundarias especifiquen su posición relativa a la vista superior o entre sí (especificada por ID). De esta manera, puedes alinear dos elementos por el borde derecho o hacer que uno esté por debajo del otro, en el centro de la pantalla, en el centro a la izquierda, y así sucesivamente. De manera predeterminada, todas las vistas secundarias se dibujan en la esquina superior izquierda del diseño, por lo que debes definir la posición de cada vista utilizando las diversas propiedades de diseño disponibles en `RelativeLayout.LayoutParams`.

Entre algunas de las muchas propiedades de diseño disponibles para las vistas de un RelativeLayout, se incluyen las siguientes:

`android:layout_alignParentTop`

Si el valor es "true", el borde superior de esta vista coincidirá con el del elemento superior.

`android:layout_centerVertical`

Si el valor es "true", centra este elemento secundario en posición vertical dentro de su elemento superior.

android:layout_below

Posiciona el borde superior de esta vista debajo de la vista especificada con un ID de recurso.

android:layout_toRightOf

Posiciona el borde izquierdo de esta vista a la derecha de la vista especificada con un ID de recurso.

Estos son solo algunos ejemplos. Todos los atributos de diseño están documentados en `RelativeLayout.LayoutParams`.

El valor de cada propiedad de diseño es un valor booleano que habilita una posición de diseño relativa al elemento `RelativeLayout` superior o un ID que hace referencia a otra vista en el diseño en el que se debe posicionar la vista.

En tu diseño XML, las dependencias frente a otras vistas en el diseño se pueden declarar en cualquier orden. Por ejemplo, puedes declarar que "view1" se posicione debajo de "view2", incluso si "view2" es la última vista declarada de la jerarquía. El siguiente ejemplo demuestra una situación de este tipo.

Al igual que estas tres propiedades, en un `RelativeLayout` se cuenta con un sinfín de propiedades para colocar cada control donde se requiera. Veamos las principales [creo que sus propios nombres explican perfectamente la función de cada una]:

Posición relativa a otro control:

android:layout_above.

android:layout_below.

android:layout_toLeftOf.

android:layout_toRightOf.

android:layout_alignLeft.

android:layout_alignRight.

android:layout_alignTop.

android:layout_alignBottom.

android:layout_alignBaseline.

Posición relativa al layout padre:

android:layout_alignParentLeft.

android:layout_alignParentRight.

android:layout_alignParentTop.

android:layout_alignParentBottom.

android:layout_centerHorizontal.

android:layout_centerVertical.

android:layout_centerInParent.

Opciones de margen (también disponibles para el resto de layouts):

android:layout_margin.

android:layout_marginBottom.

android:layout_marginTop.

android:layout_marginLeft.

android:layout_marginRight.

Opciones de espaciado o padding (también disponibles para el resto de layouts):

android:padding.

android:paddingBottom.

android:paddingTop.

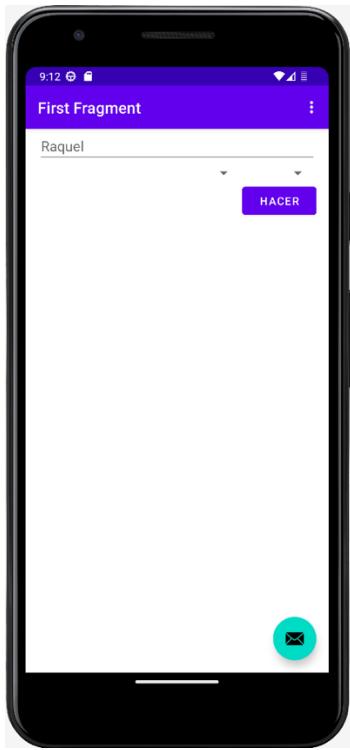
android:paddingLeft.

android:paddingRight.

1. Crear un nuevo proyecto de nombre Ejercicio5.
2. Eliminar de res->layout el archivo fragment_second.xml
3. Escribir el siguiente código en res->layout del archivo fragment_first.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp" >
    <EditText
        android:id="@+id/name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Raquel" />
    <Spinner
        android:id="@+id/dates"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentLeft="true"
        android:layout_toLeftOf="@+id/times" />
    <Spinner
        android:id="@+id/times"
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentRight="true" />
    <Button
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@+id/times"
        android:layout_alignParentRight="true"
        android:text="Hacer" />
</RelativeLayout>
```

4. Ejecutar la aplicación.

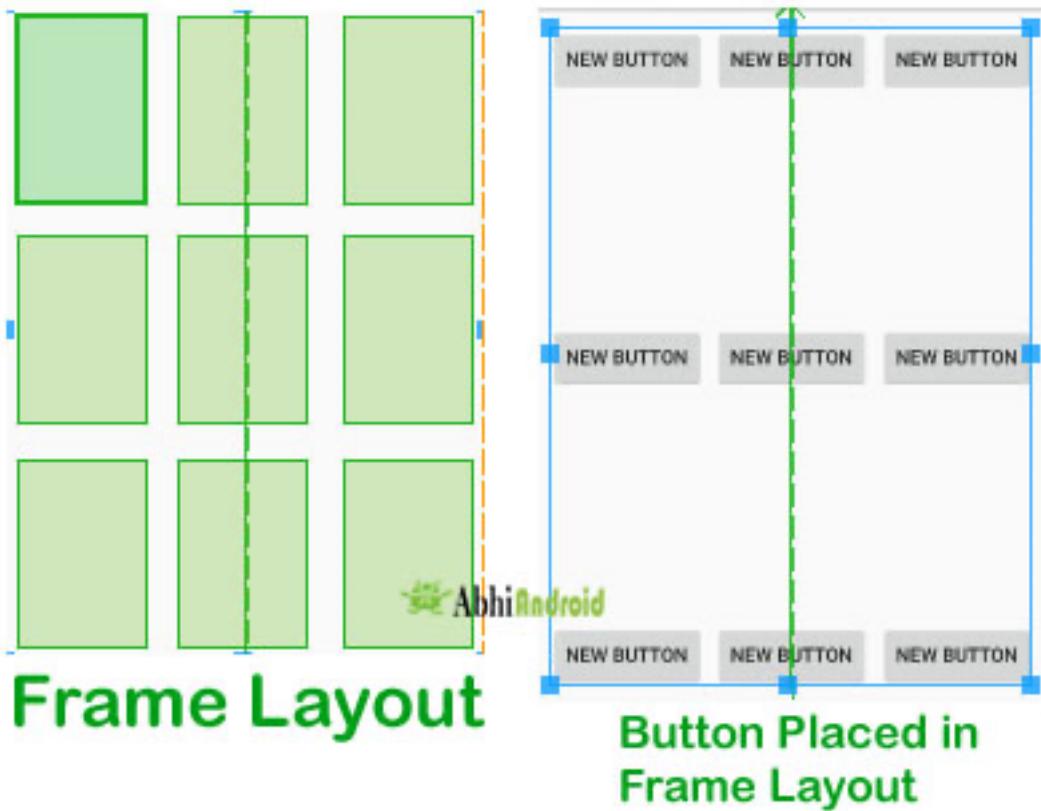


EJERCICIO 6 FrameLayout

Un FrameLayout coloca todos sus controles hijos alineados con su esquina superior izquierda, de forma que cada control quedará oculto por el control siguiente (a menos que éste último tenga transparencia). Por ello, suele utilizarse para mostrar un único control en su interior, a modo de contenedor (placeholder) sencillo para un sólo elemento sustituible, por ejemplo una imagen. Los componentes incluidos en un FrameLayout podrán establecer sus propiedades android:layout_width y android:layout_height, que podrán tomar los valores “fill_parent” (para que el control hijo tome la dimensión de su layout contenedor) o “wrap_content” (para que el control hijo tome la dimensión de su contenido).

Frame Layout es uno de los diseños más simples para organizar los controles de vista. Están diseñados para bloquear un área en la pantalla. El diseño de marco debe usarse para mantener la vista secundaria, ya que puede ser difícil mostrar vistas individuales en un área específica de la pantalla sin superponerse entre sí.

Podemos agregar varios elementos secundarios a un FrameLayout y controlar su posición asignando gravedad a cada elemento secundario mediante el atributo **android:layout_gravity**.



1. Crear un nuevo proyecto de nombre Ejercicio6.
2. Eliminar de res->layout el archivo fragment_second.xml
3. Escribir el siguiente código en res->layout del archivo fragment_first.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <ImageView
        android:id="@+id/imgvw1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:scaleType="centerCrop"
        android:src="@drawable/barco" />
    <TextView
        android:id="@+id/txtvw1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="40dp"
        android:background="#4C374A"
        android:padding="10dp"
        android:text="Hallstatt, Austria"
        android:textColor="#FFFFFF"
        android:textSize="20sp" />
    <TextView
```

```

        android:id="@+id/txtvw2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="right|bottom"
        android:background="#AA000000"
        android:padding="10dp"
        android:text="4/Enero/2023"
        android:textColor="#FFFFFF"
        android:textSize="18sp" />
    </FrameLayout>

```

4. Eliminar fragment_second.xml
5. Realizar ajustes en el archivo FirstFragment.java

```

package com.proyectos.ejercicio6;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;
import androidx.navigation.fragment.NavHostFragment;

import com.proyectos.ejercicio6.databinding.FragmentFirstBinding;

public class FirstFragment extends Fragment {

    private FragmentFirstBinding binding;

    @Override
    public View onCreateView(
        LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState
    ) {
        binding = FragmentFirstBinding.inflate(inflater, container,
false);
        return binding.getRoot();
    }

    public void onViewCreated(@NonNull View view, Bundle
savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);
    }

    @Override
    public void onDestroyView() {
        super.onDestroyView();
        binding = null;
    }
}

```

6. Realizar ajustes en SecondFragment.java

```
package com.proyectos.ejercicio6;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;
import androidx.navigation.fragment.NavHostFragment;

public class SecondFragment extends Fragment {

    public void onViewCreated(@NonNull View view, Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);

    }

    @Override
    public void onDestroyView() {
        super.onDestroyView();

    }
}
```

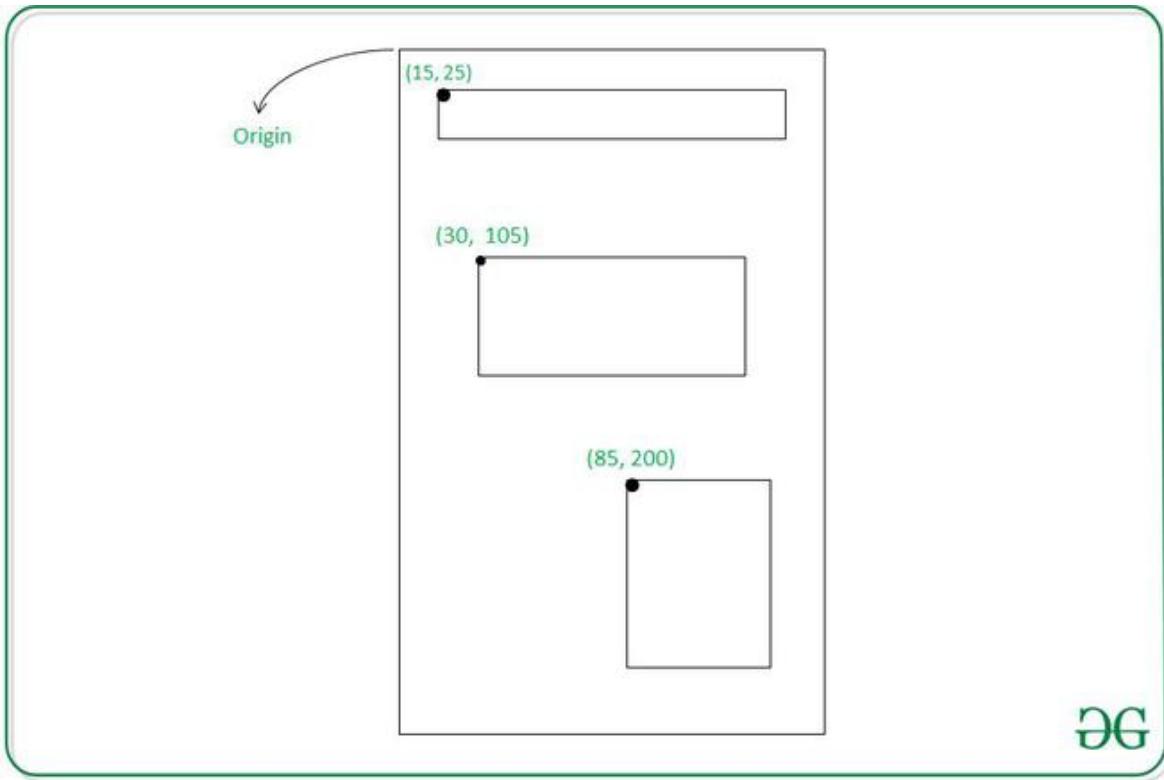
7. Ejecutar la aplicación



EJERCICIO 7. AbsoluteLayout

AbsoluteLayout permite indicar las coordenadas (x,y) donde queremos que se visualice cada elemento. No es recomendable utilizar este tipo de Layout. La aplicación que estamos diseñando tiene que visualizarse correctamente en dispositivos con cualquier tamaño de pantalla. Para conseguir esto, no es una buena idea trabajar con coordenadas absolutas. De hecho, este tipo de Layout ha sido marcado como obsoleto.

Un **diseño absoluto** le permite especificar la ubicación exacta, es decir, las coordenadas X e Y de sus hijos con respecto al origen en la esquina superior izquierda del diseño. El diseño absoluto es menos flexible y más difícil de mantener para diferentes tamaños de pantallas, por eso no se recomienda. Aunque Absolute Layout está obsoleto ahora.



DE

Algunos de los atributos importantes de Absolute Layout son los siguientes:

1. **android:id** : especifica de forma única el diseño absoluto
2. **android:layout_x**: especifica la coordenada X de las vistas (los posibles valores de esto están en densidad-píxel o píxel)
3. **android:layout_y**: especifica la coordenada Y de las vistas (los valores posibles de esto están en dp o px)

Escribir el siguiente código con fragment_first.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    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:layout_width="fill_parent"
    android:layout_height="fill_parent"
    tools:context=".MainActivity">

    <!--Setting up TextViews-->
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_x="100px"
        android:layout_y="300px"
        android:text="Primer texto"/>
```

```

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_x="120px"
    android:layout_y="350px"
    android:text="Segundo texto"/>

</AbsoluteLayout>

```

Realizar los ajustes en el archivo FirstFragment.java

```

package com.proyectos.ejercicio7;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;
import androidx.navigation.fragment.NavHostFragment;

import com.proyectos.ejercicio7.databinding.FragmentFirstBinding;

public class FirstFragment extends Fragment {

    private FragmentFirstBinding binding;

    @Override
    public View onCreateView(
        LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState
    ) {
        binding = FragmentFirstBinding.inflate(inflater, container,
false);
        return binding.getRoot();
    }

    public void onViewCreated(@NonNull View view, Bundle
savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);

    }

    @Override
    public void onDestroyView() {
        super.onDestroyView();
        binding = null;
    }
}

```

Ejecutar la aplicación



EJERCICIO 8: ScrollView

1. Crear un nuevo proyecto de nombre Ejercicio8.
2. Eliminar de res->layout el archivo fragment_second.xml
3. Escribir el siguiente código en res->layout del archivo fragment_first.xml

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

    <LinearLayout
        android:id="@+id/layout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="16dp"
        tools:context=".MainActivity">

        <TextView
            android:id="@+id/headline_4"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:text="Ejemplos"

            android:textAppearance="@style/TextAppearance.MaterialComponents.Headline
4"
            android:textColor="@color/black" />
    
```

```
<TextView
    android:id="@+id/textView"
    style="@style/ExampleText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="1. TextView " />

<TextView
    android:id="@+id/hello_world_text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World! " />

<TextView
    android:id="@+id/textView2"
    style="@style/ExampleText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="2. Cambiar color" />

<TextView
    android:id="@+id/text_color"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Texto color Material Indigo 500"
    android:textColor="#3F51B5" />

<TextView
    android:id="@+id/textView3"
    style="@style/ExampleText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="3. Cambiar tamaño" />

<TextView
    android:id="@+id/text_size_14sp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Tamaño de 14sp"
    android:textSize="14sp" />

<TextView
    android:id="@+id/text_size_16sp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Tamaño de 16sp"
    android:textSize="16sp" />

<TextView
    android:id="@+id/text_size_20sp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Tamaño de 20sp"
    android:textSize="20sp" />

<TextView
    android:id="@+id/text_size_24sp"
```

```
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Tamaño de 24sp"
        android:textSize="24sp" />

<TextView
    android:id="@+id/textView4"
    style="@style/ExampleText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="4. Cambiar estilo de fuente" />

<TextView
    android:id="@+id/text_style_normal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Estilo normal"
    android:textStyle="normal" />

<TextView
    android:id="@+id/text_style_bold"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Estilo en negrilla"
    android:textStyle="bold" />

<TextView
    android:id="@+id/text_style_italic"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Estilo itálico"
    android:textStyle="italic" />

<TextView
    android:id="@+id/text_style_combination"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Estilos itálico y negrilla"
    android:textStyle="bold|italic" />

<TextView
    android:id="@+id/textView5"
    style="@style/ExampleText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="5. Cambiar tipo de fuente" />

<TextView
    android:id="@+id/typeface_sans"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Tipo de fuente sans"
    android:typeface="sans" />

<TextView
    android:id="@+id/typeface_serif"
    android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:text="Tipo de fuente serif"
        android:typeface="serif" />

<TextView
    android:id="@+id/typeface_monospace"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Tipo de fuente monospace"
    android:typeface="monospace" />

<TextView
    android:id="@+id/textView6"
    style="@style/ExampleText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="6. Convertir urls en links clickeables" />

<TextView
    android:id="@+id/autolink_url"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:autoLink="web"
    android:text="https://develou.com"
    android:textColorLink="#F48FB1" />

<TextView
    android:id="@+id/textView7"
    style="@style/ExampleText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="7. Convertir a mayúsculas" />

<TextView
    android:id="@+id/text_all_caps"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="www.develou.com"
    android:textAllCaps="true" />

<TextView
    android:id="@+id/textView8"
    style="@style/ExampleText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="8. Cambiar familia tipográfica" />

<TextView
    android:id="@+id/font_family1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:fontFamily="sans-serif-condensed-medium"
    android:text="Sans Serif Condensed Medium "
    android:textSize="16sp" />

<TextView
    android:id="@+id/font_family2"
```

```

        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:fontFamily="serif-monospace"
        android:text="Serif Monospace "
        android:textSize="16sp" />

    <TextView
        android:id="@+id/font_family3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:fontFamily="casual"
        android:text="Casual"
        android:textSize="16sp" />

    <TextView
        android:id="@+id/custom_font"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:fontFamily="@font/cinzel"
        android:text="Texto que usa la fuente Cinzel"
        android:textSize="16sp" />

    <TextView
        android:id="@+id/textView9"
        style="@style/ExampleText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="9. Crear TextView programáticamente" />

</LinearLayout>
</ScrollView>

```

- Ajustar el programa de FirstFragment.java

```

package com.proyectos.ejercicio8;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;
import androidx.navigation.fragment.NavHostFragment;

import com.proyectos.ejercicio8.databinding.FragmentFirstBinding;

public class FirstFragment extends Fragment {

    private FragmentFirstBinding binding;

    @Override
    public View onCreateView(
        LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState
    ) {

```

```

        binding = FragmentFirstBinding.inflate(inflater, container,
false);
        return binding.getRoot();

    }

    public void onViewCreated(@NonNull View view, Bundle
savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);

    }

    @Override
    public void onDestroyView() {
        super.onDestroyView();
        binding = null;
    }
}

```

5. Incluir otros TextView en FirstFragment.xml con la finalidad de probar el Scroll en la pantalla del emulador. Cuidar que no se dupliquen los valores en el atributo id de cada TextView.
6. Ejecutar la aplicación.

