

Tema 3

Activities

ÍNDICE



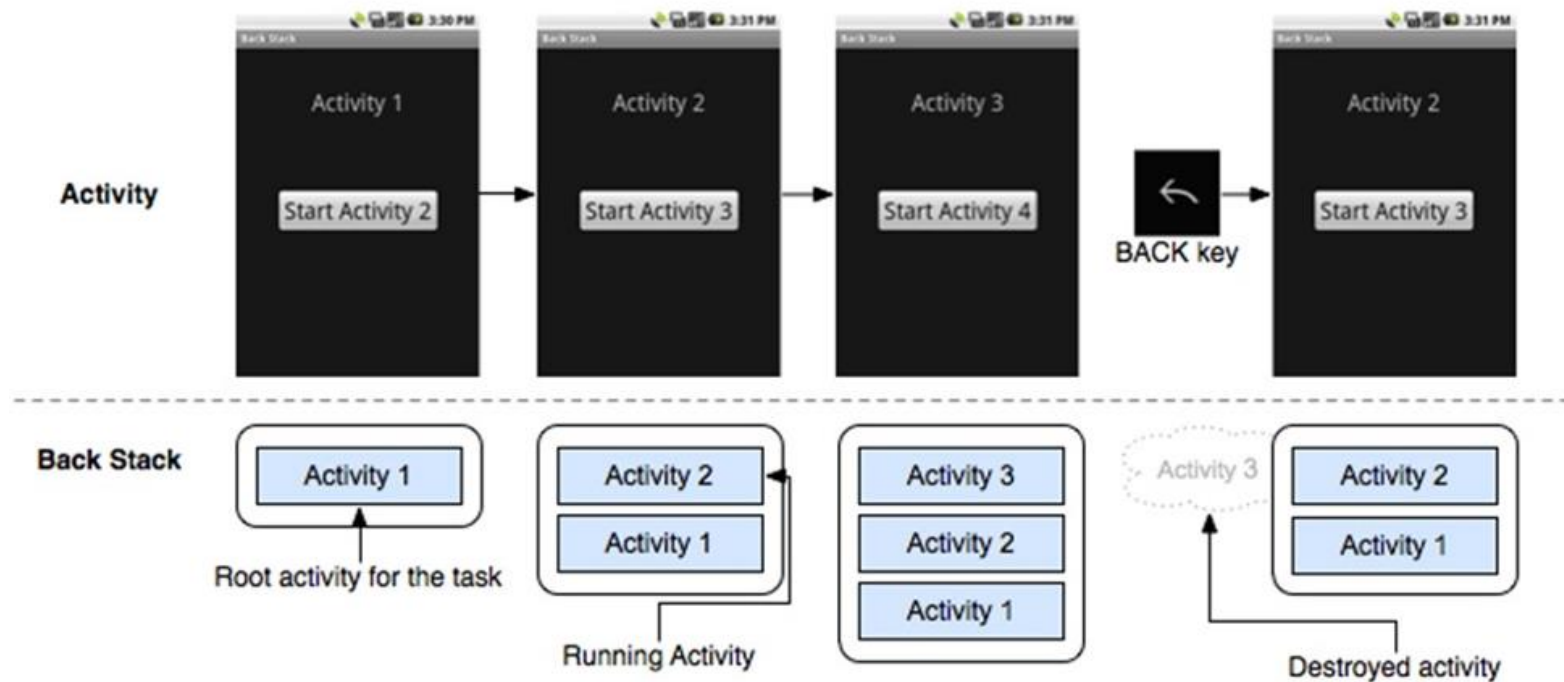
1. Activity

- Es el principal componente que nos encontramos en una aplicación Android.
- Hereda de la clase Activity y por lo tanto tendrá una pantalla.
- Se encarga de gestionar parte de las interacciones con el usuario.
- Nuestras aplicaciones están formadas por múltiples actividades que relacionan entre sí de manera flexible, es decir, podremos ir pasando de unas a otras(navegando) en función de lógica de aplicación que hayamos diseñado.

1. Activity

- Desde nuestra pantalla principal podremos ir desplazando a otras pantallas y conforme realizamos esta “navegación” nuestros dispositivos van recordando esta secuencia de “visitas”.
- Android guarda las posiciones de cada una de estas pantallas en la “pila”, Stack(Back Stack).
- Cuando una actividad inicia otra, esta nueva obtiene el foco y la anterior es empujada a la parte superior de la pila, pero detenida.
- Si el usuario regresa(Back), la actividad que tenía por foco se elimina, y la actividad previa(situada debajo de ella en la pila) se restaura y vuelve a coger el foco.

1. Activity



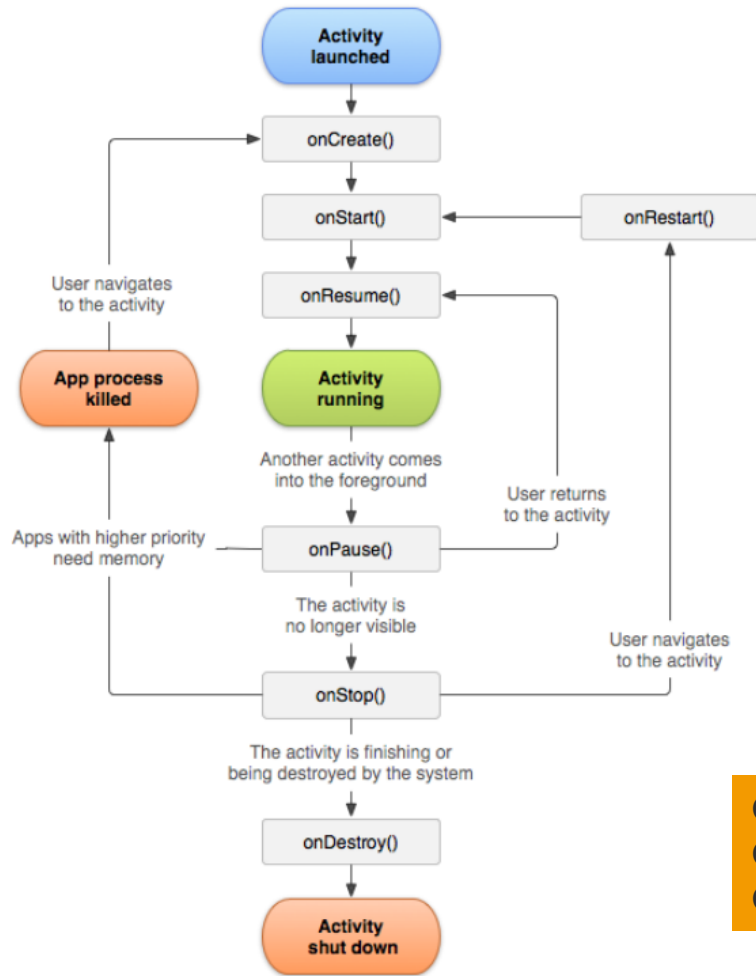
2. Ciclo de vida de una aplicación

Una Actividad pasará por diferentes períodos de funcionamiento que se denominarán estados, y dependerán del flujo de la aplicación, la interacción con el usuario y las necesidades de recursos del sistema.

ESTADO	EXPLICACIÓN
INACTIVA	Actividad que ha terminado su ejecución o no se ha iniciado.
DETENIDA	Sí ocupa memoria, aunque no es visible y, por tanto, tampoco es accesible.
PAUSADA	Es visible, pero no se puede interactuar con ella.
ACTIVA	Está en primer plano del dispositivo y el usuario interactúa con ella.

Estos estados suelen ir en sucesión creciente cuando la actividad se pone en marcha, y en sentido decreciente cuando desaparece y se destruye. A estas sucesiones de estados en los que se puede encontrar la actividad se les conoce como **ciclo de vida**.

2. Ciclo de vida de una aplicación



Ciclo de vida completo : onCreate() -> onDestroy()
Ciclo de vida visible: onStart() -> onStop()
Ciclo de vida en primer plano: onResume() -> onPause()

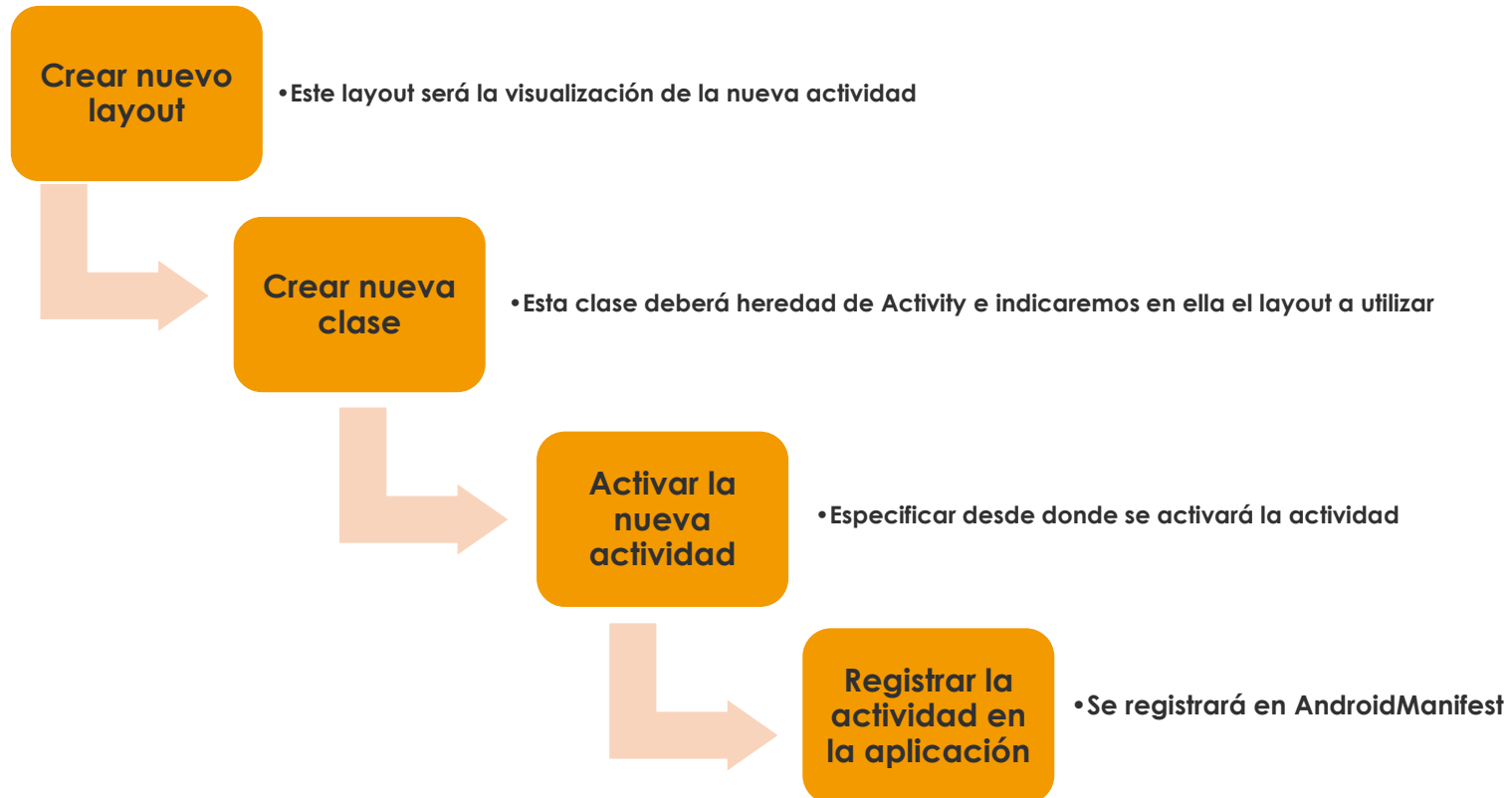
2. Ciclo de vida de una aplicación

ESTADO	EXPLICACIÓN
ONCREATE()	Es llamado cuando la actividad es invocada por primera vez. Es el momento en que se crean las vistas(interfaz de usuario), se hace una reserva de memoria para los datos.
ONSTART()	Sí ocupa memoria, aunque no es visible y, por tanto, tampoco es accesible.
ONRESUME()	Un método llamado cuando la actividad puede interactuar con el usuario. Sucede al anterior y precede al onPause(). En este periodo se suelen activar sensores, cámara, ejecutar animaciones, actualizar información...
ONPAUSE()	Ocurre cuando nuestra actividad pasa a un segundo plano, bien porque está en proceso de destrucción o porque otra actividad pasa a ocupar el primer plano. Es cuando se detienen procesos, animaciones... en este periodo la actividad es parcialmente visible. Puede ocurrir que vuelva a primer plano (onResume) o que sea parada onStop(), haciéndose visible para el usuario.

2. Ciclo de vida de una aplicación

ESTADO	EXPLICACIÓN
ONSTOP()	Este método es invocado cuando la actividad ya no es visible al usuario y otra actividad ha pasado a primer plano. Se pausan las animaciones, determinados servicios se detienen (GPS) y se minimizan los recursos consumidos por la actividad, aunque la actividad aún está en memoria. Si queremos reactivarla, se utiliza <code>onRestart()</code> , volviendo a primer plano o <code>onDestroy()</code> , si queremos eliminarla definitivamente.
ONRESTART()	Llamado cuando una actividad parada vuelve a primer plano. Siempre seguido de un <code>onStart()</code> .
ONDESTROY()	Es llamado cuando la actividad es definitivamente destruida y eliminada de memoria. En su interior se suele limpiar y liberar recursos, por ejemplo la cámara

3. Crear una nueva actividad

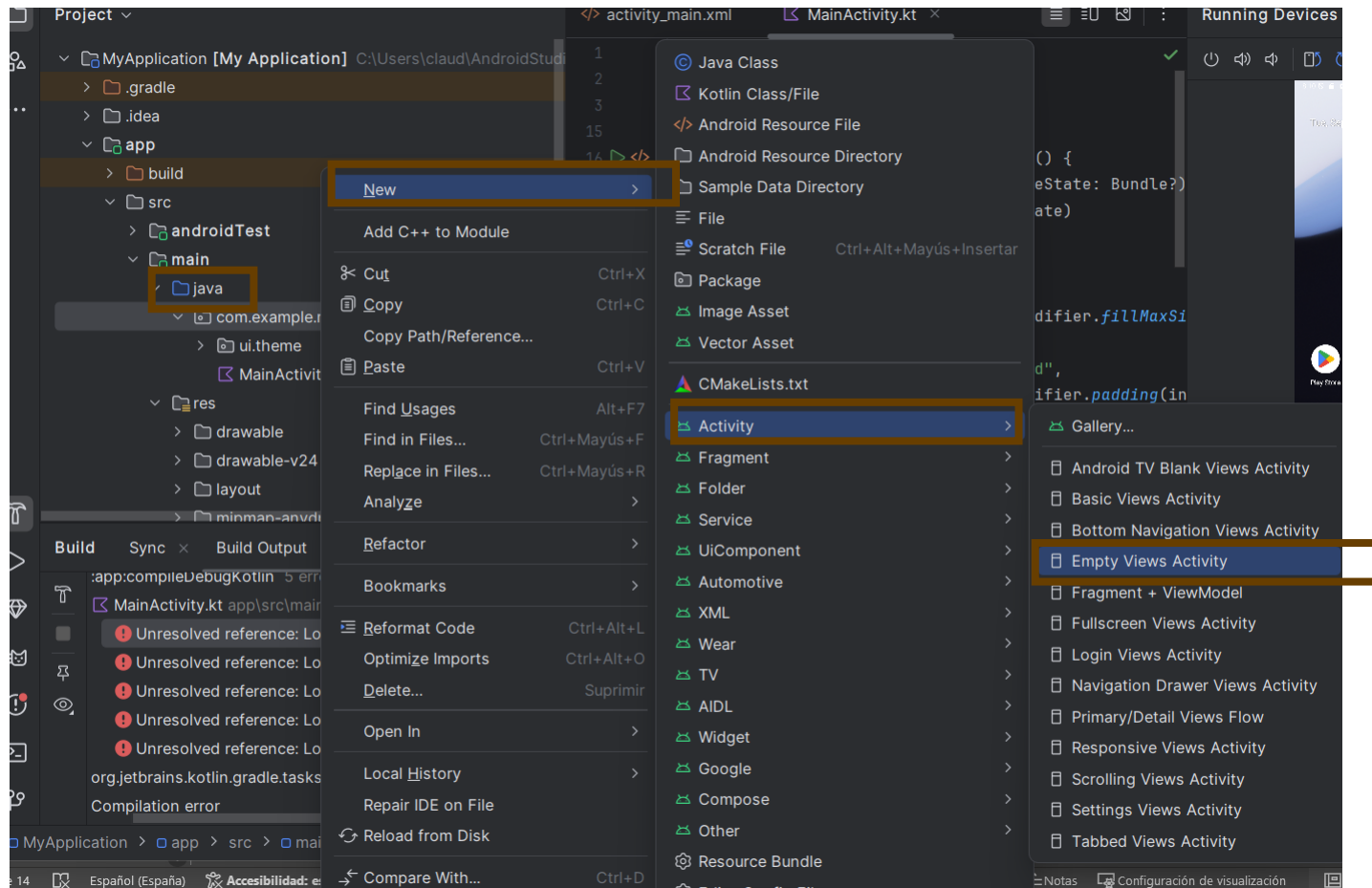


3. Crear una nueva actividad

Antes de modificar el código de nuestra actividad principal, vamos a crear una nueva actividad para la segunda pantalla de la aplicación análoga a ésta primera, a la que llamaremos **SaludoActivity**.

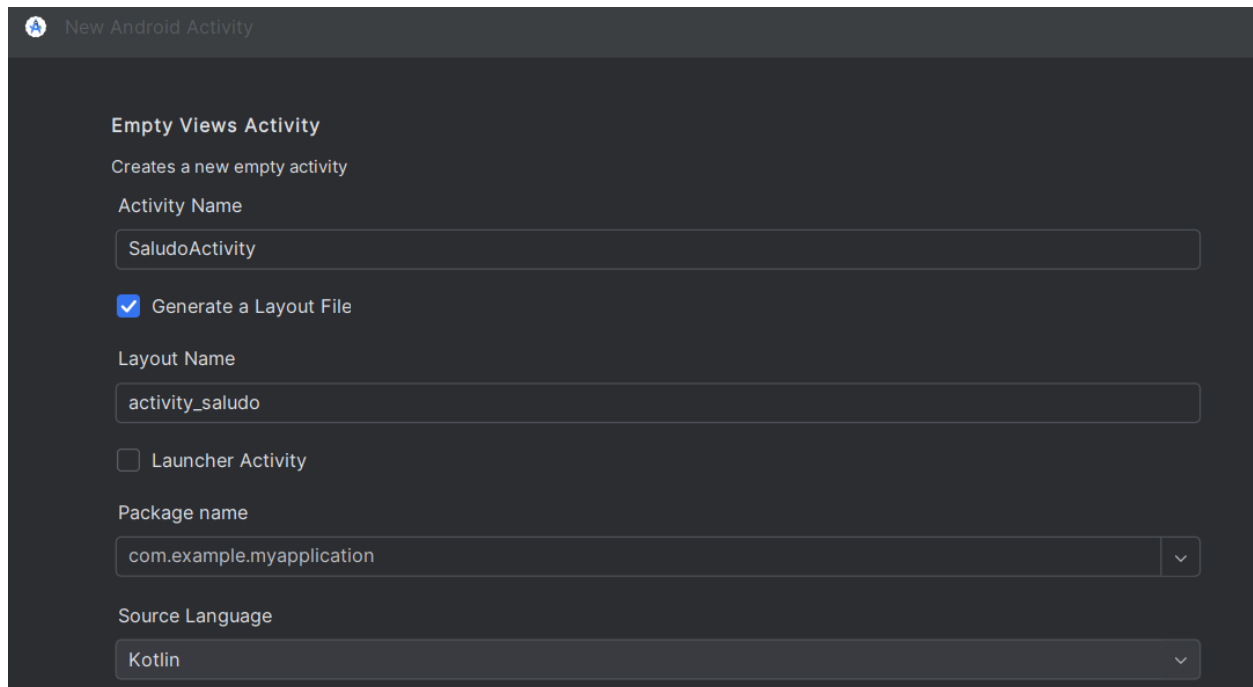
Para ello, pulsaremos el botón derecho sobre la carpeta `/src/main/java/tu.paquete.java/` y seleccionaremos la opción de menú **New / Activity / Empty Activity**.

3. Crear una nueva actividad



3. Crear una nueva actividad

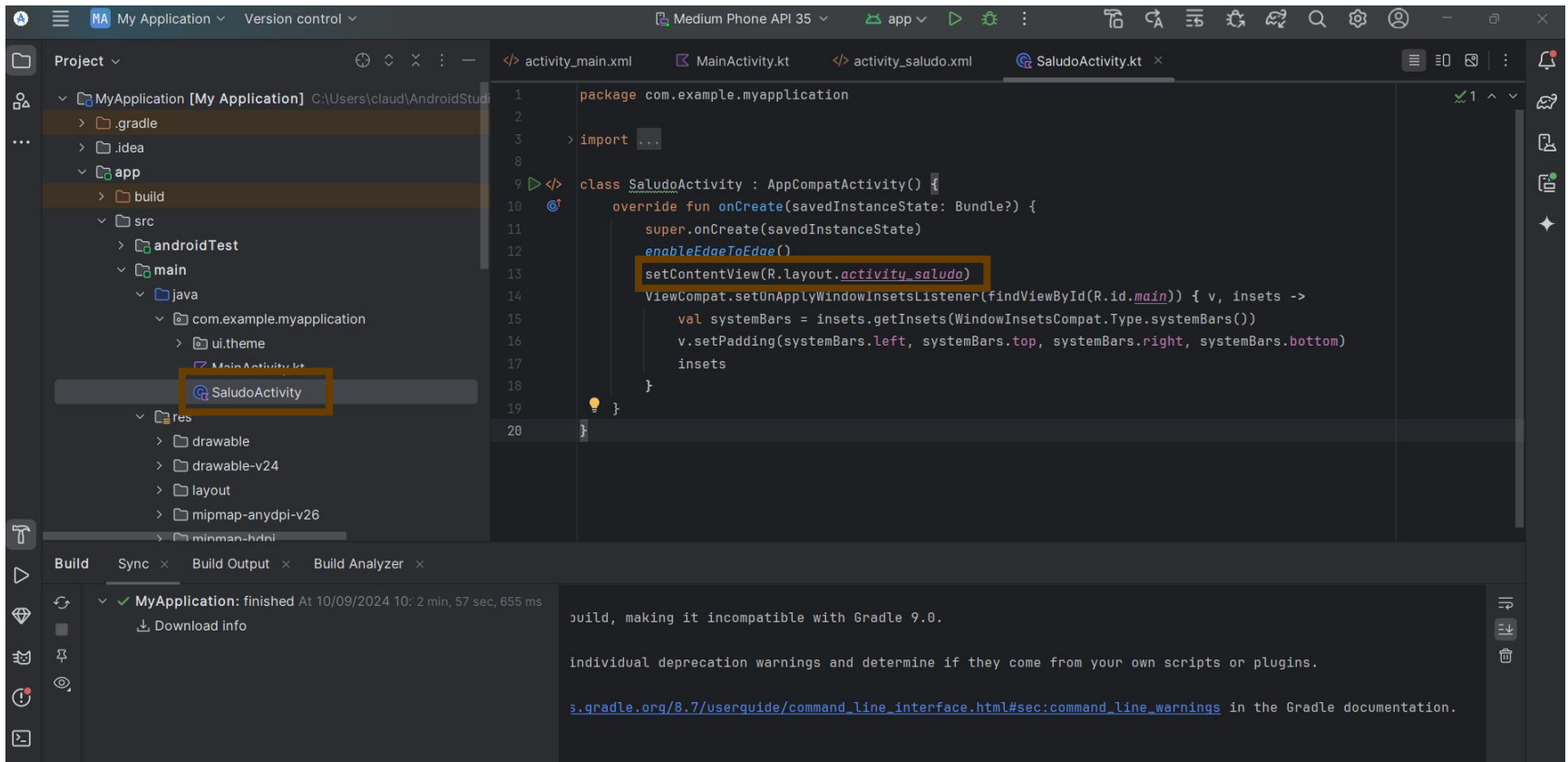
En el cuadro de diálogo que aparece indicaremos el nombre de la actividad, en nuestro caso **SaludoActivity**, el nombre de su layout XML asociado (Android Studio creará al mismo tiempo tanto el layout XML como la clase java), que llamaremos **activity_saludo**, y el nombre del paquete java de la actividad, donde podemos dejar el valor por defecto.



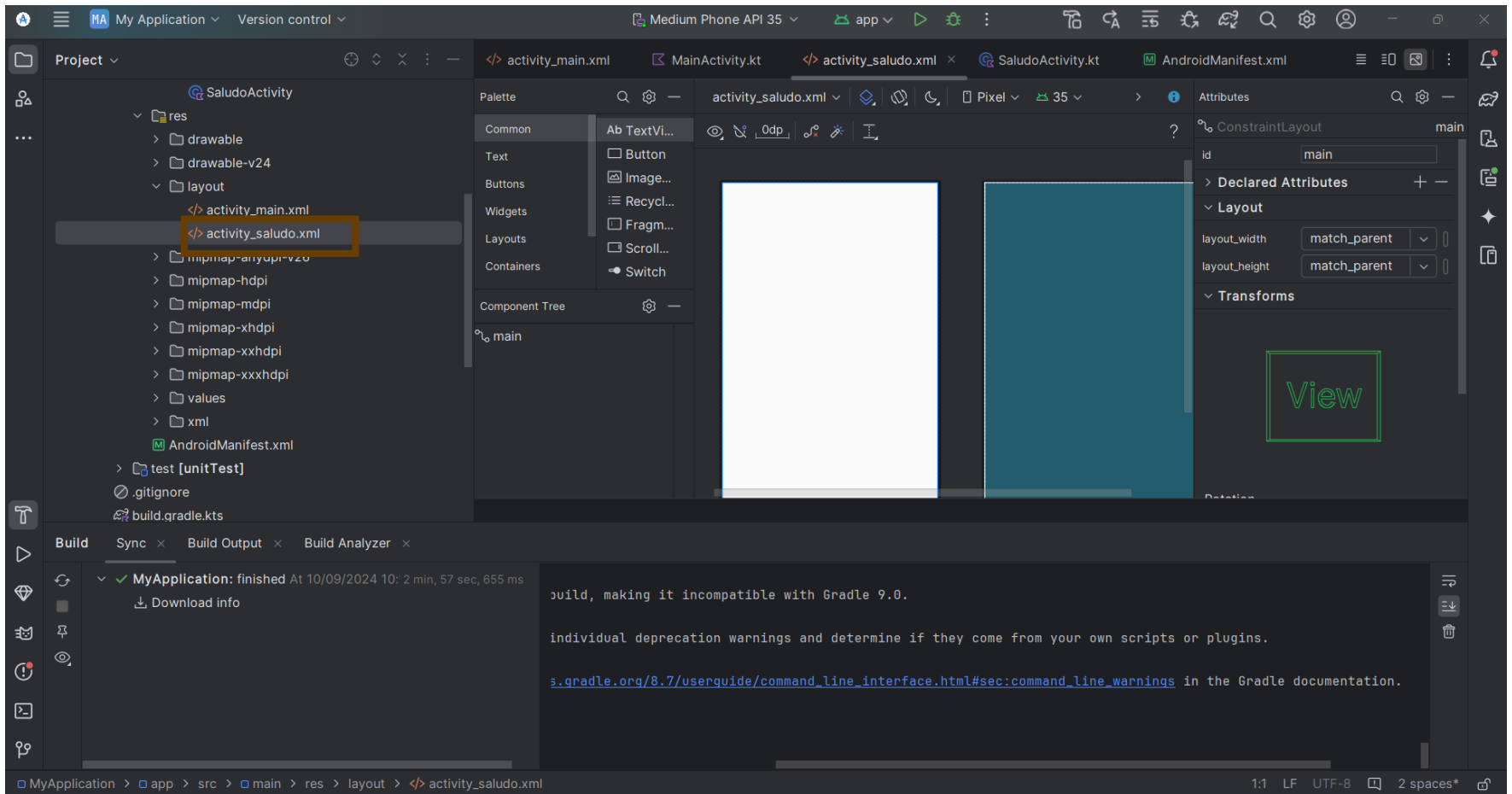
The screenshot shows the 'New Android Activity' dialog box in Android Studio. The dialog is titled 'New Android Activity' and contains the following fields and options:

- Empty Views Activity**: A section header.
- Creates a new empty activity**: A description of the activity type.
- Activity Name**: A text input field containing 'SaludoActivity'.
- Generate a Layout File**: A checked checkbox.
- Layout Name**: A text input field containing 'activity_saludo'.
- Launcher Activity**: An unchecked checkbox.
- Package name**: A text input field containing 'com.example.myapplication'.
- Source Language**: A dropdown menu set to 'Kotlin'.

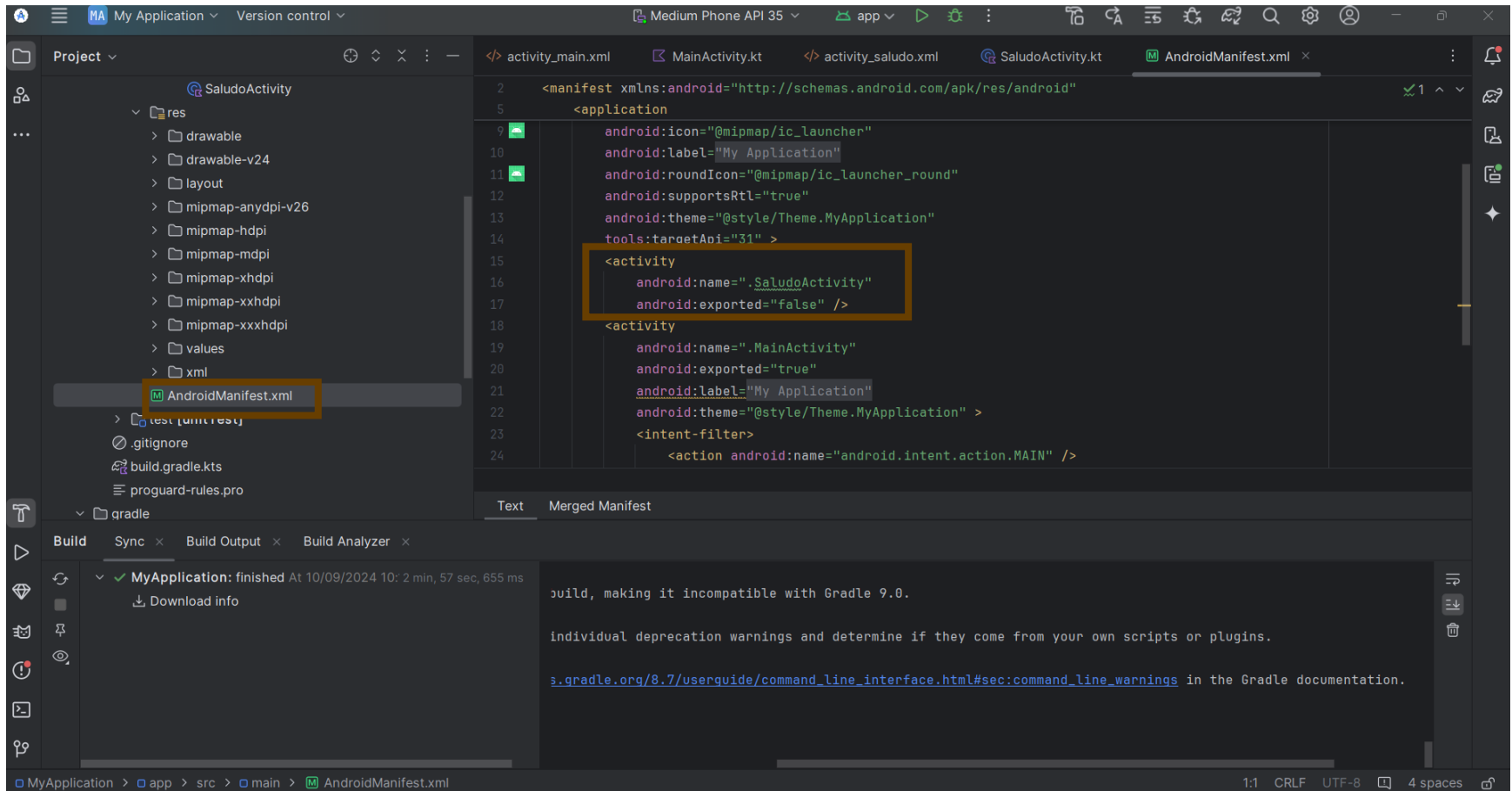
3. Crear una nueva actividad



3. Crear una nueva actividad



3. Crear una nueva actividad



3. Crear una nueva actividad

Ejercicio 2: Navegar entre dos actividades

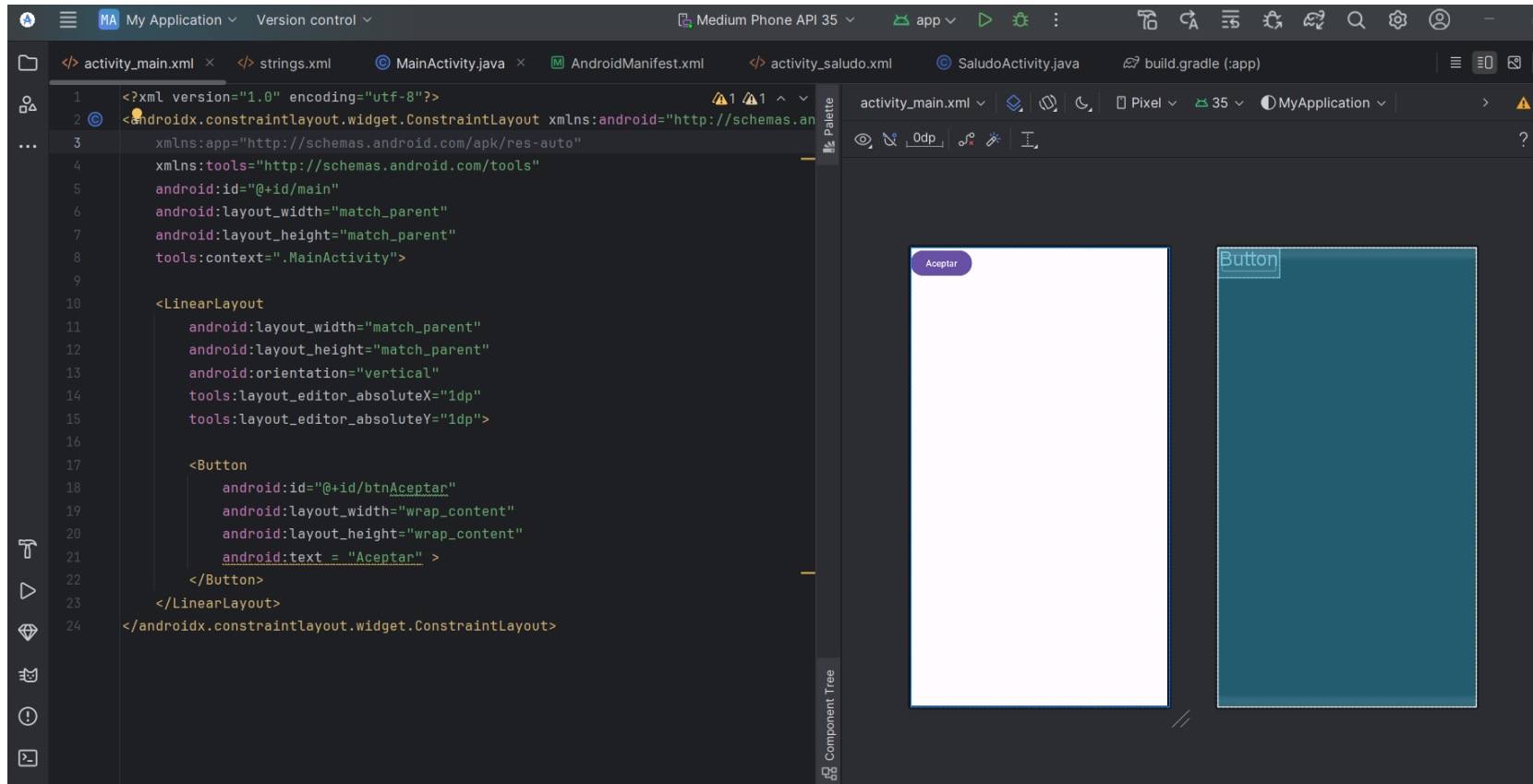
3. Crear una nueva actividad

Vamos a modificar en primer lugar el aspecto de la ventana principal de la aplicación añadiendo los controles (views) que vemos en el esquema mostrado al principio del apartado. Para ello, vamos a sustituir el contenido del fichero **activity_main.xml** por el siguiente:

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:layout_editor_absoluteX="1dp"
    tools:layout_editor_absoluteY="1dp">

    <Button
        android:id="@+id/btnAceptar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text = "Aceptar" >
    </Button>
</LinearLayout>
```

3. Crear una nueva actividad



3. Crear una nueva actividad

Añadiremos a la clase **MainActivity** un **botón**, obteniendo una referencia a los diferentes controles de la interfaz que necesitemos manipular, en nuestro caso sólo el botón. Para ello utilizaremos el método **findViewById()** indicando el ID de cada control, definidos como siempre en la clase R. Todo esto lo haremos dentro del método **onCreate()** de la clase **MainActivity**, justo a continuación de la llamada a **setContentView()** que ya comentamos.

```
public class MainActivity extends AppCompatActivity {  
  
    private Button btnAceptar; 2 usages  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        //Obtenemos una referencia a los controles de la interfaz  
        btnAceptar = (Button)findViewById(R.id.btnAceptar);  
    }  
}
```

3. Crear una nueva actividad

Ya sólo nos queda implementar las acciones a tomar cuando pulsemos el botón de la pantalla.

Implementaremos el evento **onClick** de dicho botón. Este botón tendrá que ocuparse de abrir la actividad SaludoActivity.

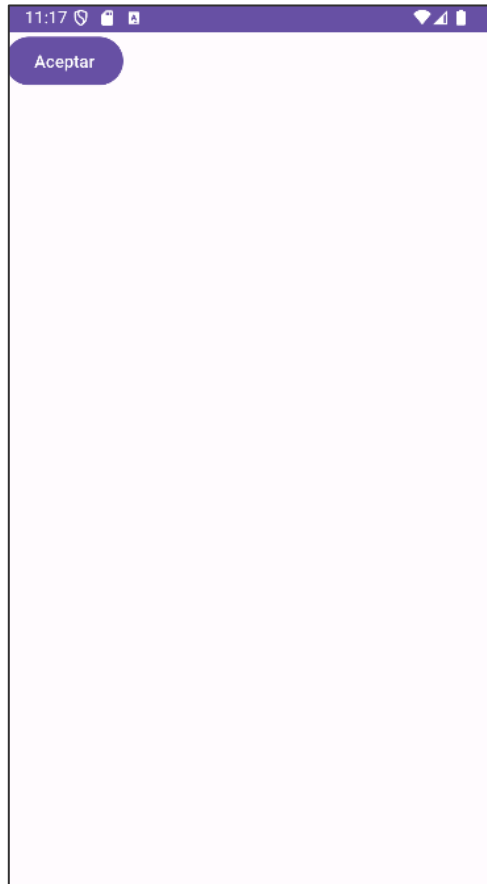
```
public class MainActivity extends AppCompatActivity {

    private Button btnAceptar;

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

        //Implementamos el evento click del botón
        btnAceptar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //Creamos el Intent
                Intent intent =
                    new Intent(getApplicationContext(), SaludoActivity.class);
                startActivity(intent);
            }
        });
    }
}
```

3. Crear una nueva actividad



3. Crear una nueva actividad

Ejercicio 3: Pasar datos entre actividades

3. Crear una nueva actividad

Vamos a modificar en primer lugar el aspecto de la ventana principal de la aplicación añadiendo los controles (views) que vemos en el esquema mostrado al principio del apartado. Para ello, vamos a sustituir el contenido del fichero **activity_main.xml** por el siguiente:

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:layout_editor_absoluteX="1dp"
    tools:layout_editor_absoluteY="1dp">

    <TextView android:id="@+id/lblNombre"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Escribe tu nombre:" >
    </TextView>

    <EditText
        android:id="@+id/txtNombre"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="text">
    </EditText>

    <Button
        android:id="@+id/btnAceptar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text = "Aceptar" >
    </Button>
</LinearLayout>
```


3. Crear una nueva actividad

En la etiqueta y el botón hemos establecido la propiedad `android:text`, que indica el texto que aparece en el control. Y aquí nos vamos a detener un poco, ya que tenemos dos alternativas a la hora de hacer esto:

Definir una nueva cadena de texto en el fichero de recursos `/src/main/res/values/strings.xml`:

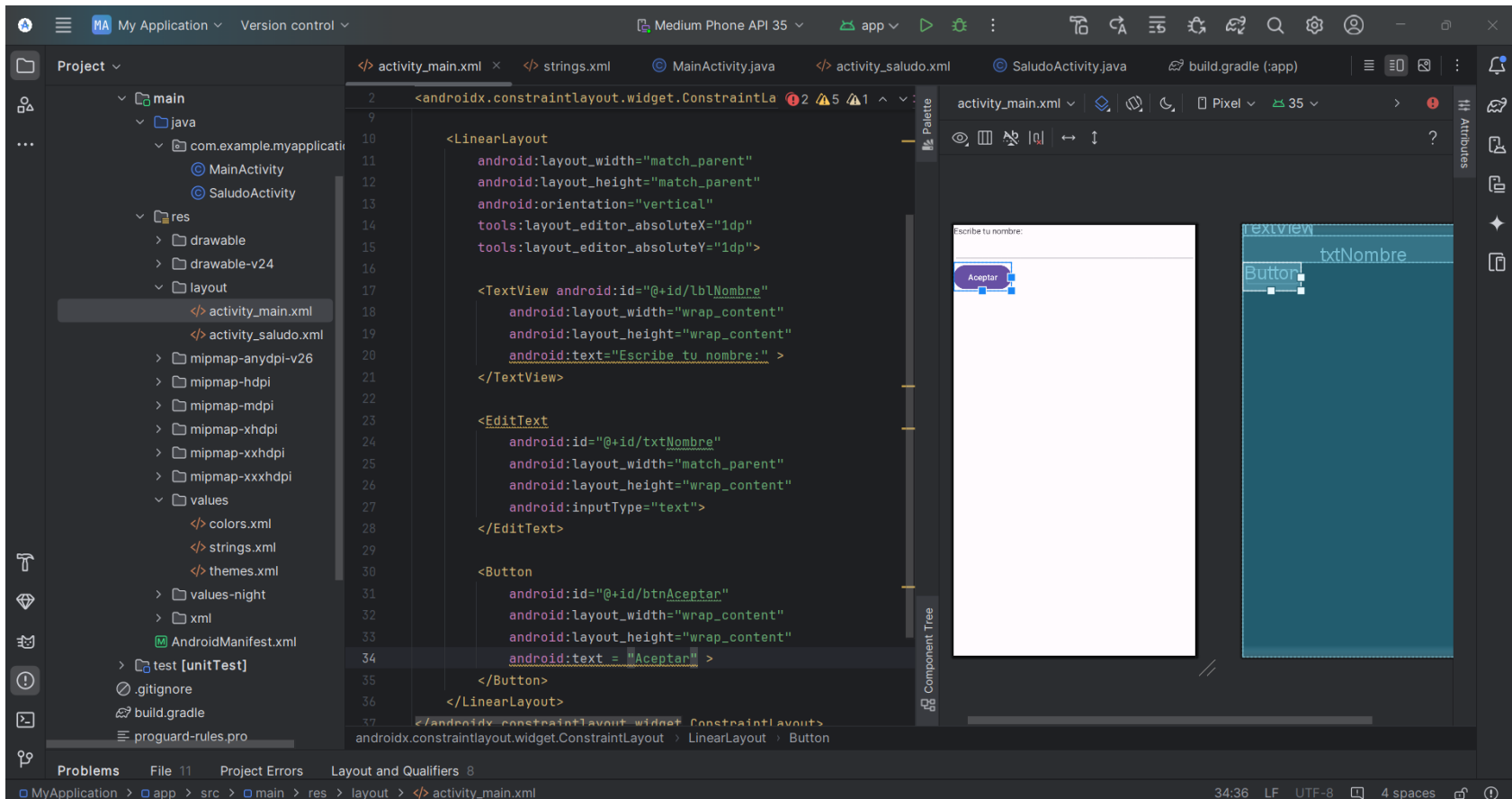
```
android:text="Escribe tu nombre:" >
```

```
<resources>
...
<string name="nombre">Escribe tu nombre:</string>
...
</resources>
```

Indicar el identificador de la cadena como valor de la propiedad `android:text`, siempre precedido del prefijo `"@string/"`, de la siguiente forma

```
android:text="@string/nombre"
```

3. Crear una nueva actividad



3. Crear una nueva actividad

Añadiremos a la clase **MainActivity** un cuadro de **texto y un botón**, obteniendo una referencia a los diferentes controles de la interfaz que necesitemos manipular, en nuestro caso sólo el cuadro de texto y el botón. Para ello definiremos ambas referencias como atributos de la clase y para obtenerlas utilizaremos el método **findViewById()** indicando el ID de cada control, definidos como siempre en la clase R. Todo esto lo haremos dentro del método **onCreate()** de la clase **MainActivity**, justo a continuación de la llamada a **setContentView()** que ya comentamos.

```
public class MainActivity extends AppCompatActivity {

    private EditText txtNombre; 1 usage
    private Button btnAceptar; 1 usage

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

        //Obtenemos una referencia a los controles de la interfaz
        txtNombre = (EditText)findViewById(R.id.txtNombre);
        btnAceptar = (Button)findViewById(R.id.btnAceptar);
    }
}
```

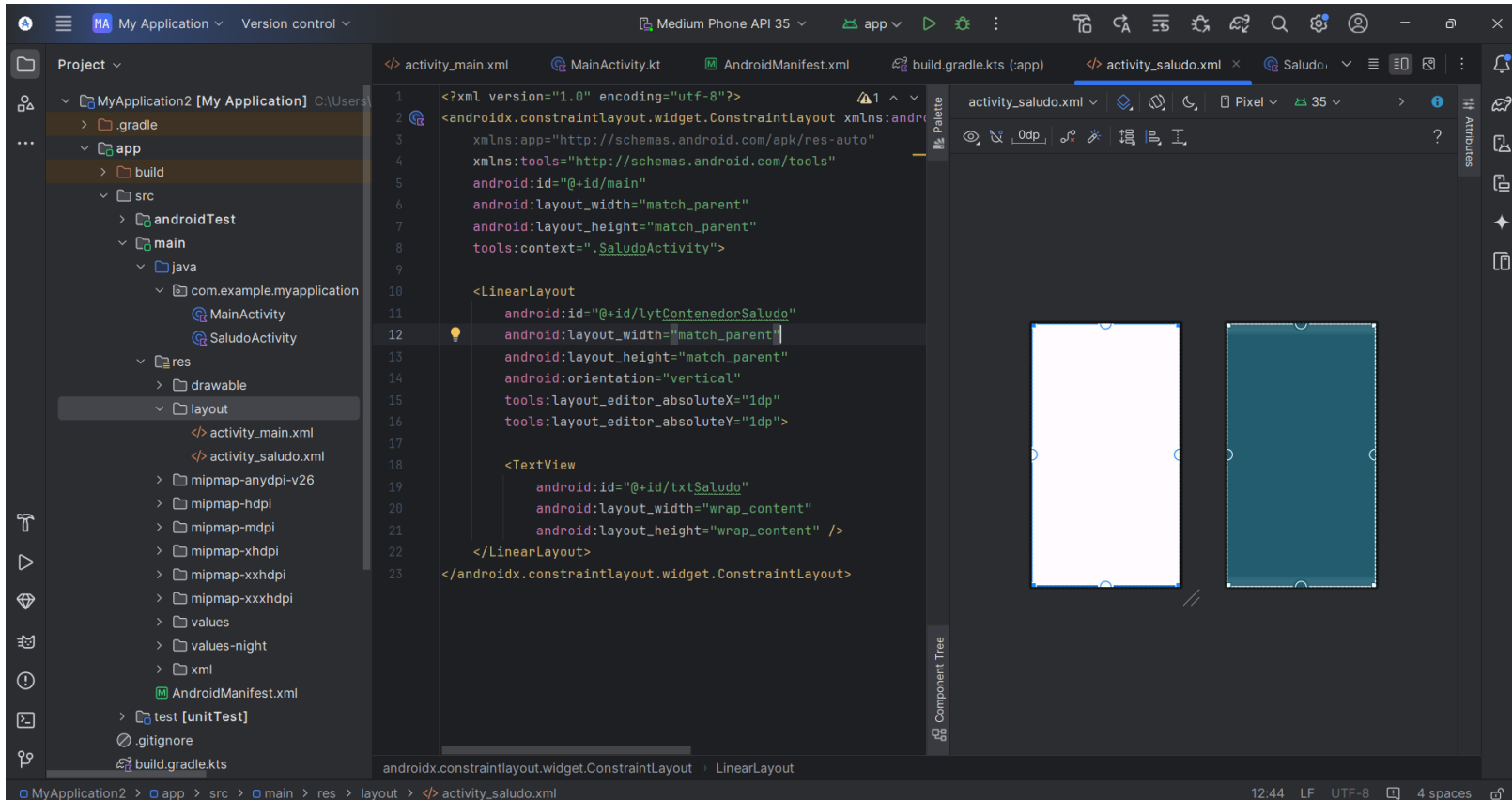
3. Crear una nueva actividad

Definiremos la interfaz de la segunda pantalla, abriendo el fichero `activity_saludo.xml`, y añadiendo esta vez tan sólo un `LinearLayout` como contenedor y una etiqueta (`TextView`) para mostrar el mensaje personalizado al usuario.

```
<LinearLayout
    android:id="@+id/lytContenedorSaludo"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:layout_editor_absoluteX="1dp"
    tools:layout_editor_absoluteY="1dp">

    <TextView
        android:id="@+id/txtSaludo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</LinearLayout>
```

3. Crear una nueva actividad



3. Crear una nueva actividad

Ya sólo nos queda implementar las acciones a tomar cuando pulsemos el botón de la pantalla.

Implementaremos el evento **onClick** de dicho botón. Este botón tendrá que ocuparse de abrir la actividad SaludoActivity pasándole toda la información necesaria.

Como ya indicamos en el apartado anterior, la comunicación entre los distintos componentes y aplicaciones en Android se realiza mediante **intents**, por lo que el primer paso es crear un objeto de este tipo.

En nuestro caso particular vamos a utilizar el intent para iniciar una actividad desde otra actividad de la misma aplicación, para lo que pasaremos a su constructor una referencia a la propia actividad llamadora (**MainActivity.this**), y la clase de la actividad llamada (**SaludoActivity.class**).

Pero en nuestro ejemplo queremos también pasarle cierta información a la actividad llamada, concretamente el nombre que introduzca el usuario en el cuadro de texto de la pantalla principal. Para hacer esto creamos un objeto **Bundle**, que puede contener una lista de pares clave-valor con toda la información a pasar entre actividades. En nuestro caso sólo añadimos un dato de tipo String mediante el método **putString(clave, valor)**.

Tras esto añadiremos la información al intent mediante el método **putExtras()**.

3.1. Intent

Un **Intent(intención)** se define como un sistema de comunicación que permite interactuar entre componentes de la misma o de distintas aplicaciones Android.

Es el elemento básico de comunicación con el que podremos comenzar una aplicación, iniciar un servicio , entregar un mensaje.

Los clasificamos en **implícitos y explícitos** según sus características y construcción.

3.1. Intent explícitos

En estos Intents es necesario nombrar al componente que se necesita ejecutar, es decir, la clase Java que se necesita para realizar alguna tarea.

Su construcción bastante simple ya que sólo deberemos instanciar el Intent pasándoles como parámetros el contexto y la actividad que vamos a lanzar.

```
//Creamos el Intent
Intent intent =
    new Intent(MainActivity.this, SaludoActivity.class);
//Creamos la información a pasar entre actividades
Bundle b = new Bundle();
b.putString("NOMBRE", txtNombre.getText().toString());
//Añadimos la información al intent
intent.putExtras(b);
//Iniciamos la nueva actividad
startActivity(intent);
```


3.1. Intent implícitos

En este caso informamos al sistema de la acción que deseamos realizar y el nos responde con el componente más adecuado. Si existen varios elementos responde preguntando cual queremos seleccionar.

```
// Intent implícito para abrir una página web
Intent intent = new Intent(Intent.ACTION_VIEW);
intent.setData(Uri.parse("https://www.example.com"));
if (intent.resolveActivity(getPackageManager()) != null) {
    startActivity(intent);
}
```

3. Crear una nueva actividad

```
public class MainActivity extends AppCompatActivity {

    private EditText txtNombre;  2 usages
    private Button btnAceptar;  2 usages

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

        //Obtenemos una referencia a los controles de la interfaz
        txtNombre = (EditText)findViewById(R.id.txtNombre);
        btnAceptar = (Button)findViewById(R.id.btnAceptar);

        //Implementamos el evento click del botón
        btnAceptar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //Creamos el Intent
                Intent intent =
                    new Intent( packageContext: MainActivity.this, SaludoActivity.class);
                //Creamos la información a pasar entre actividades
                Bundle b = new Bundle();
                b.putString("NOMBRE", txtNombre.getText().toString());
                //Añadimos la información al intent
                intent.putExtras(b);
                //Iniciamos la nueva actividad
                startActivity(intent);
            }
        });
    }
}
```

3. Crear una nueva actividad

Volveremos a la activity **SaludoActivity** para ampliar el método **onCreate()** para recuperar la información pasada desde la actividad principal y asignarla como texto de la etiqueta.

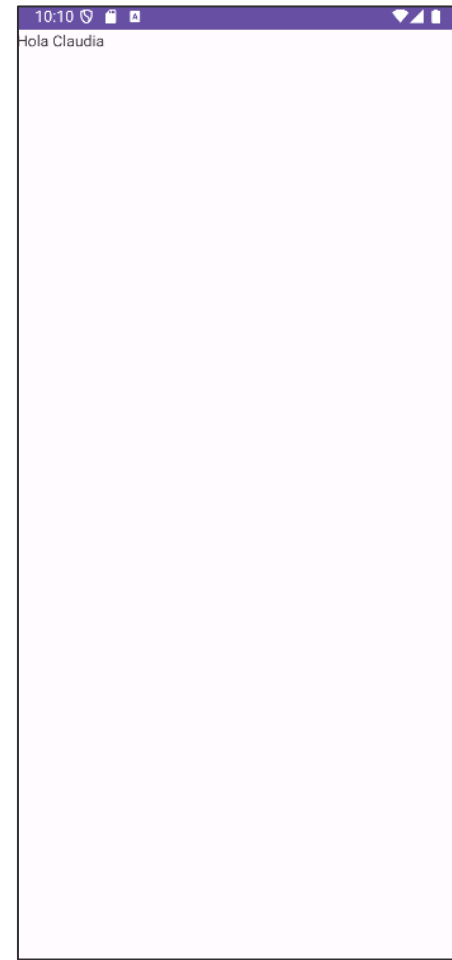
Para ello accederemos en primer lugar al intent que ha originado la actividad actual mediante el método **getIntent()** y recuperaremos su información asociada (objeto Bundle) mediante el método **getExtras()**.

Hecho esto tan sólo nos queda construir el texto de la etiqueta mediante su método **setText(texto)** y recuperando el valor de nuestra clave almacenada en el objeto **Bundle** mediante **getString(clave)**.

3. Crear una nueva actividad

```
public class SaludoActivity extends AppCompatActivity {  
  
    private TextView txtSaludo; 2 usages  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_saludo);  
  
        //Localizar los controles  
        txtSaludo = (TextView)findViewById(R.id.txtSaludo);  
        //Recuperamos la información pasada en el intent  
        Bundle bundle = this getIntent().getExtras();  
        //Construimos el mensaje a mostrar  
        txtSaludo.setText("Hola " + bundle.getString(key: "NOMBRE"));  
    }  
}
```

3. Crear una nueva actividad



ACTIVIDAD

Crea dos actividades, MainActivity y SecondActivity. Usa un intent explícito para navegar de MainActivity a SecondActivity. Crea un botón que, al hacer clic, utilice un intent explícito para abrir SecondActivity.

Usa un intent implícito para abrir una URL en el navegador web. Crea una actividad MainActivity, en la cual haya definido un botón que, al hacer clic, abra el navegador con una URL específica utilizando un intent implícito. Por ejemplo: <https://www.google.com/>

3.2. Creación de subactividades

Una actividad iniciada por medio de la función `startActivity` es independiente de su actividad padre y por lo tanto no proporcionará ninguna información cuando ésta finalice.

En Android existe la posibilidad de crear una **subactividad asociada a una actividad padre**.

Cuando finalice la subactividad se producirá la activación de un **evento** en su actividad padre, el cual podrá recoger los resultados producidos por la subactividad.

Para lanzar una subactividad usaremos el método **`startActivityForResult`** que recibe los siguientes parámetros:

- Intent con una petición explícita o implícita.
- Código de petición que estableceremos nosotros.

Cuando una subactividad esté preparada para terminar llamaremos a **`setResult` antes de la llamada `finish`** para devolver un resultado a la actividad padre.

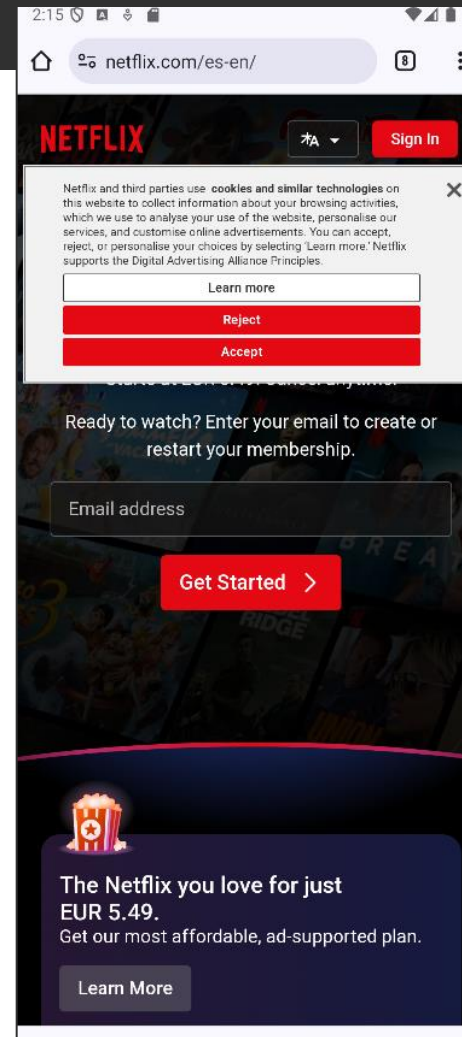
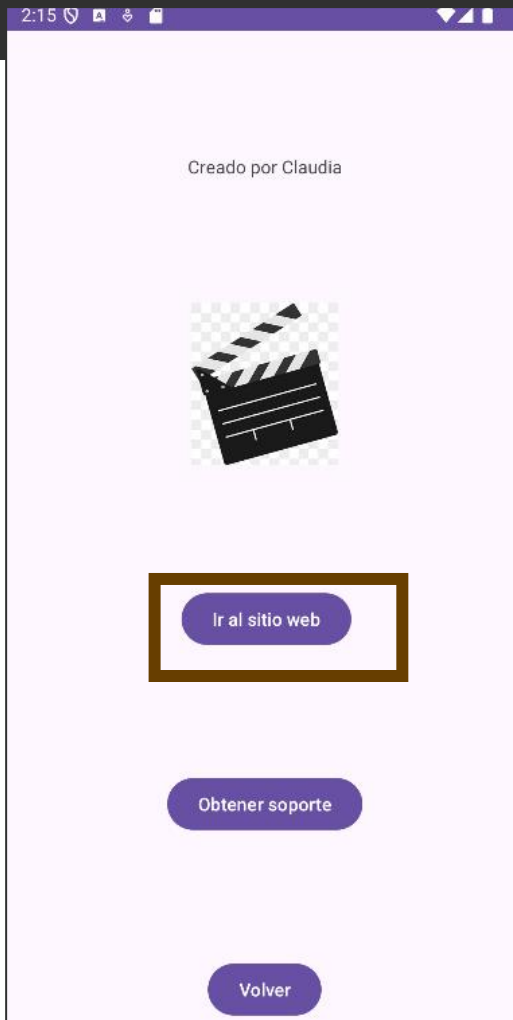
- El código de resultado será `Activity.RESULT_OK` o `Activity.RESULT_CANCELED`.

PROYECTO

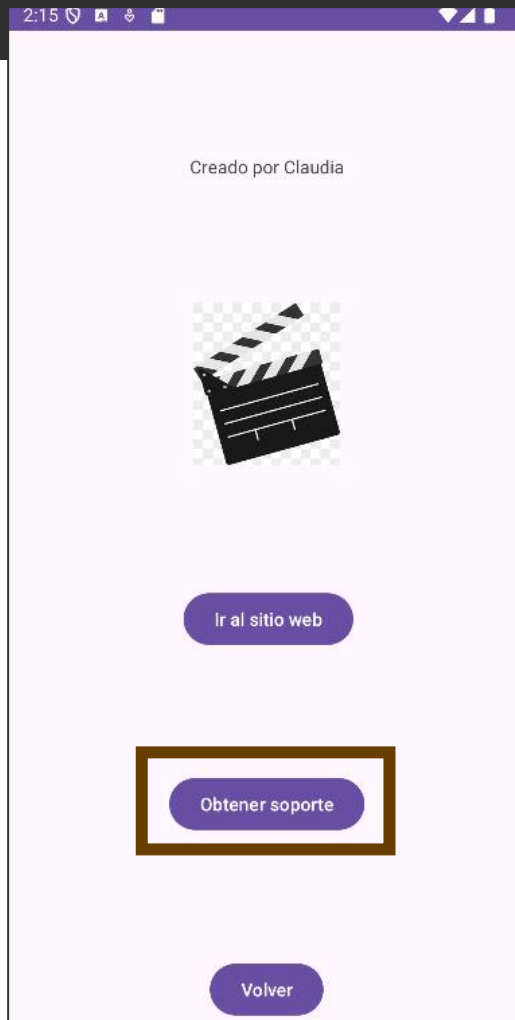
Continuaremos con el proyecto Fimoteca iniciando en el tema anterior. Añadiremos algo de funcionalidad a los botones creados:

- En el botón “**Ir al sitio web**” haremos que se lance un intent implícito para abrir una página web. Puedes usar un intent de tipo **Intent.ACTION_VIEW**
- En el botón “**Obtener soporte**” lanzaremos un intent implícito para enviar un email a nuestra dirección de correo. Para ello puedes utilizar **Intent.ACTION_SENDTO** con una URI de tipo “mailto:midireccion@dominio.com”
- En el botón “**Volver**” cerraremos la actividad llamando al método **finish()**

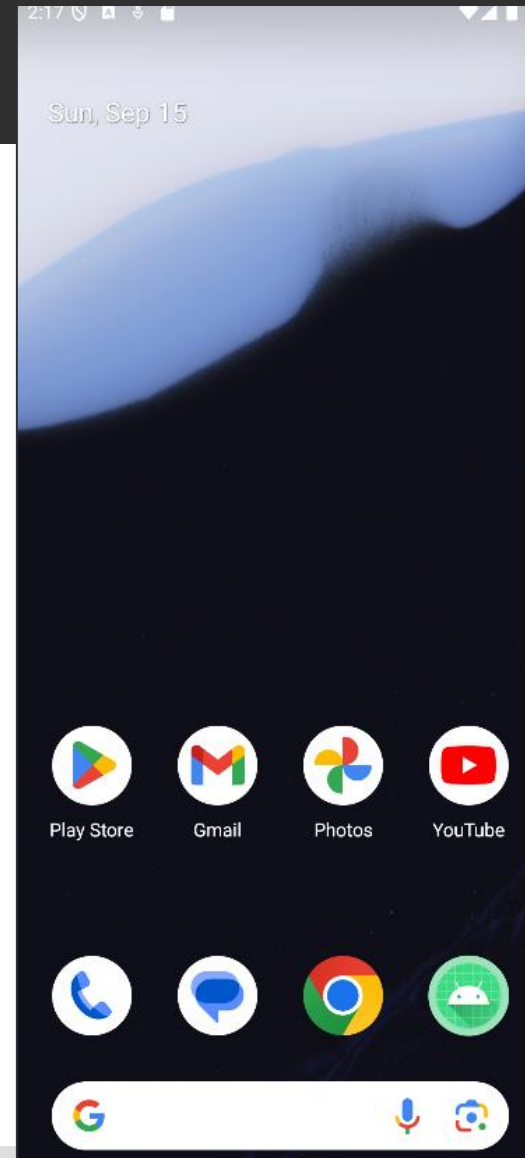
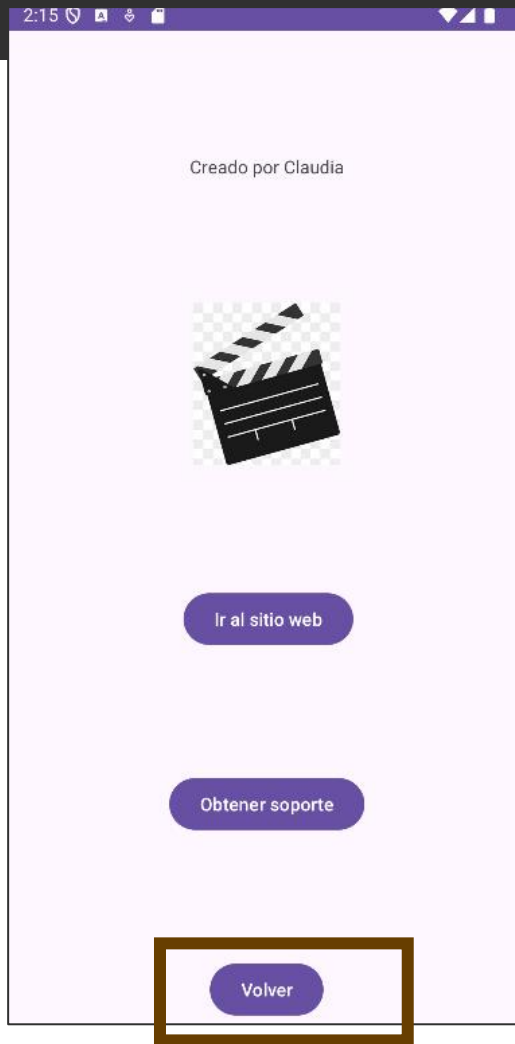
PROYECTO



PROYECTO



PROYECTO



PROYECTO

Vamos a añadir nuevas actividades al proyecto anterior y a crear transiciones entre ellas:

- **FilmListActivity:** será la actividad principal de nuestra aplicación. Edita el fichero AndroidManifest.xml para que ésta pase a ser la actividad principal, en lugar de MainActivity. En ella añadiremos tres **botones: Ver película A, Ver película B y Acerca de.**
- **FilmDataActivity:** mostraremos los datos de la película. Añadiremos en ella una etiqueta con el **texto Datos de la película** y tres **botones: Ver película relacionada, Editar película y Volver a la principal.**
- **FilmEditActivity:** mostraremos únicamente la etiqueta de **texto Editando película** y dos **botones: Guardar y Cancelar.**

PROYECTO

A continuación, añadiremos transiciones entre ellas mediante intents explícitos:

- Los botones **Ver película A**, **Ver película B** de FilmListActivity abrirán la actividad FilmDataActivity.
- El botón **Acerca de** de FilmListActivity abrirá la actividad MainActivity.
- El botón **Ver película relacionada** de FilmDataActivity abrirá la actividad FilmDataActivity.
- El botón **Editar película** de FilmDataActivity abrirá la actividad FilmEditActivity.
- El botón **Volver a la principal** de FilmDataActivity abrirá la actividad FilmListActivity.
- Los botones **Guardar** y **Cancelar** de FilmEditActivity cerrarán la actividad llamando al método finish()

PROYECTO

A continuación, vamos a pasar un parámetro al lanzar nuestras actividades para indicar la película que se ha seleccionado.

En la actividad **FilmListActivity** añadiremos un **EditText** para añadir el nombre de la película. Este valor se enviará como **extra** en el intent al pulsar el botón **Ver película A** o **Ver película B**

En la actividad **FilmDataActivity** se creará un elemento de tipo **TextView** en el que se mostrará el nombre de la película recibido.

Haremos lo mismo con el botón **Ver película relacionada** de **FilmDataActivity** que deberá pasar un **extra** con otro nombre de película.

PROYECTO

La actividad `FilmEditActivity` es una actividad que producirá un resultado, ya que puede modificar los datos de la película que estamos consultando. Vamos a hacer que como resultado **nos indique si el usuario ha hecho cambios en la película o si ha descartado los cambios, aprovechando los result codes definidos:**

(`Activity.RESULT_OK` o `Activity.RESULT_CANCELED`).

En primer lugar, vamos a llamar a dicha actividad desde `FilmDataActivity` con el método **`startActivityForResult`**. Implementaremos también de forma adecuada el método **`onActivityResult`** y en caso de haber sido editada lo indicaremos en la etiqueta de texto.

Por otro lado, en `FilmEditActivity` haremos que al pulsar el botón Guardar devuelva el resultado de confirmación (`RESULT_OK`) y que cuando se pulse **Cancelar devuelva el de cancelación (`RESULT_CANCELED`)**.