

PROGRAMACIÓN DE DISPOSITIVOS MÓVILES

CÓDIGO: BEINSOF58

GRUPO: 02

MANUAL:
VENTANA EMERGENTE PERSONALIZADA ANDROID STUDIO

DOCENTE: YHON JERSON ROBLES PUENTES

PRESENTADO POR:

SEBASTIAN SANDOVAL CHAVARRO
20222208924

DANIEL FELIPE RUIZ LIZCANO
20222207514

SAMUEL ALEXANDER PERDOMO FAJARDO
20222208795

YORMAN LEANDRO TRUJILLO CHACÓN
20222209581

JUAN ESTEBAN LOZANO HERNANDEZ
20222209033

UNIVERSIDAD SURCOLOMBIANA

NEIVA, 19 DE OCTUBRE

INTRODUCCIÓN

Este manual tiene como objetivo guiar paso a paso el desarrollo de una ventana emergente personalizada para simular iniciar sesión en una aplicación Android utilizando Android Studio. La ventana emergente, también conocida como diálogo personalizado, es una interfaz crucial en muchas aplicaciones modernas para autenticar usuarios antes de acceder a ciertas funciones o contenidos.

A lo largo de este manual, se detalla cómo crear una ventana de inicio de sesión que aparezca como un cuadro de diálogo, con campos personalizados para ingresar el correo electrónico y la contraseña, así como botones de acción como "Ok" y "Cancelar". También se abordan las mejores prácticas de diseño para que la ventana sea visualmente atractiva y funcional, así como cómo manejar la validación básica de los datos ingresados.

El proyecto te permitirá:

- Comprender cómo utilizar AlertDialog para crear ventanas emergentes.
- Personalizar la apariencia y el comportamiento de los diálogos con XML.
- Implementar la lógica de validación de las credenciales de usuario.
- Mejorar la interacción de los usuarios con tu aplicación mediante interfaces personalizadas y dinámicas.

Con este manual, cualquier usuario podrá integrar fácilmente una ventana emergente de inicio de sesión en su aplicación Android, mejorando la experiencia de usuario y añadiendo una capa esencial de autenticación.

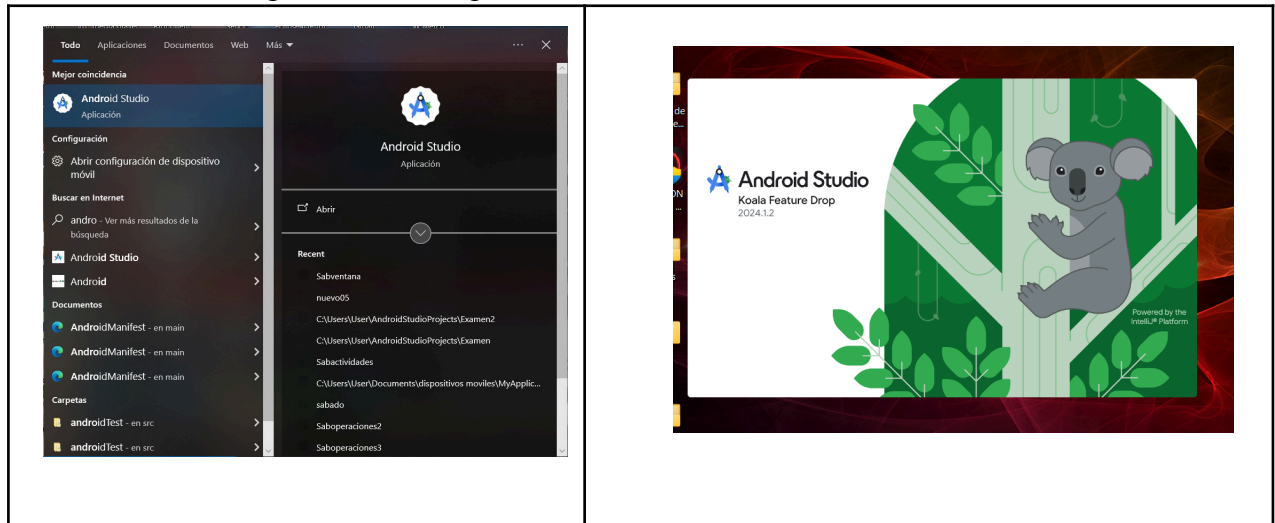
REQUISITOS

- Primero se debe verificar si se tiene instalado Android Studio, si no es el caso dirigirse al siguiente link [Android Studio](#) y proceder a instalar Android Studio. Puede ayudarse del siguiente [Tutorial Android Studio](#).
- Asegúrese que su máquina tenga los requerimientos mínimos:
 - **Sistema Operativo:**
 - **Windows:** Windows 10 (64-bit) o superior.
 - **macOS:** macOS 12.5 (Monterey) o superior.
 - **Linux:** Cualquier distribución moderna (64-bit) que soporte versiones recientes de Java y Android Studio
 - **Procesador:**
 - Procesador de 4 núcleos (CPU de 64 bits) a 2.0 GHz o más rápido.
 - **Recomendado:** Procesador Intel i5 o AMD Ryzen equivalente.
 - **Memoria RAM:**
 - **Mínimo:** 8 GB de RAM.
 - **Recomendado:** 16 GB o más para un rendimiento más fluido, especialmente al ejecutar el emulador.
 - Espacio de Almacenamiento
 - **Mínimo:** 8 GB de espacio libre en disco (se recomienda SSD).
 - **Recomendado:** 16 GB de espacio libre en disco para instalar Android Studio, SDK, emuladores, y otros archivos necesarios.
 - Java Development Kit (JDK)
 - **Java 17** instalado y configurado en el sistema. Puedes descargar la versión de Java 17 directamente desde la web oficial de Oracle o usar OpenJDK. Puede guiarse del siguiente tutorial [Tutorial Java 17](#).
 - Conexión a Internet
 - Necesaria para descargar el SDK de Android, emuladores y otras herramientas adicionales.

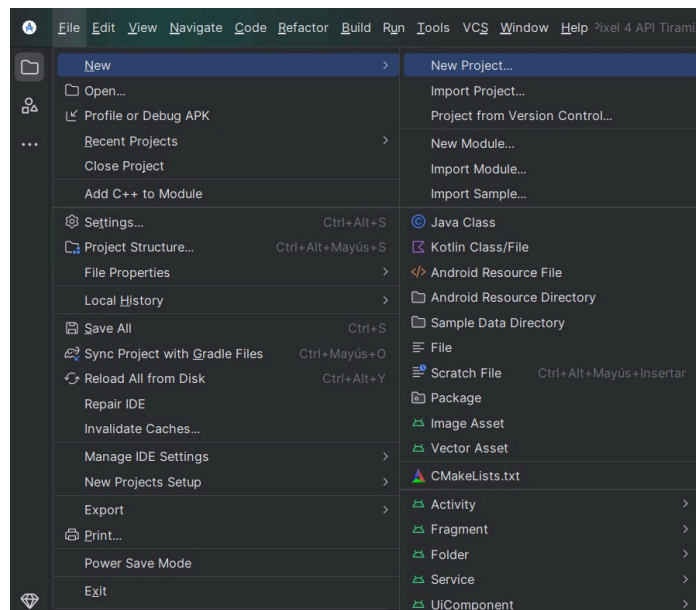
Estos requisitos garantizan que se pueda desarrollar de manera eficiente en Android Studio, utilizando Java 17 y la última versión del IDE.

MANUAL

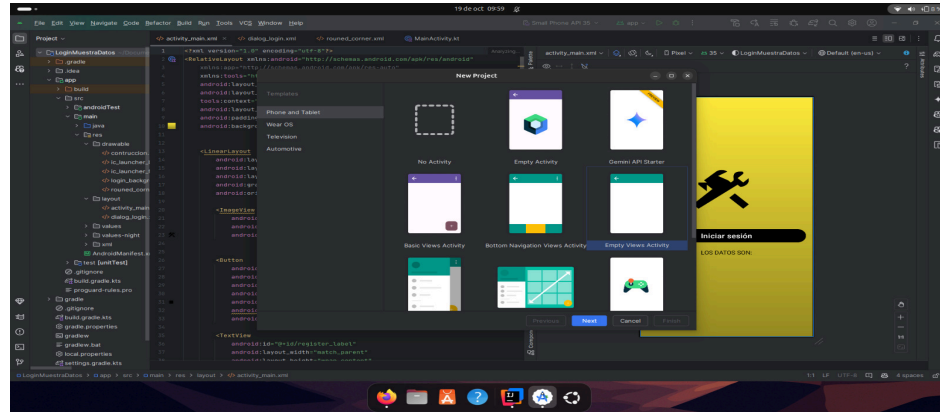
- Cuando tenga instalada la aplicación, abrirla:



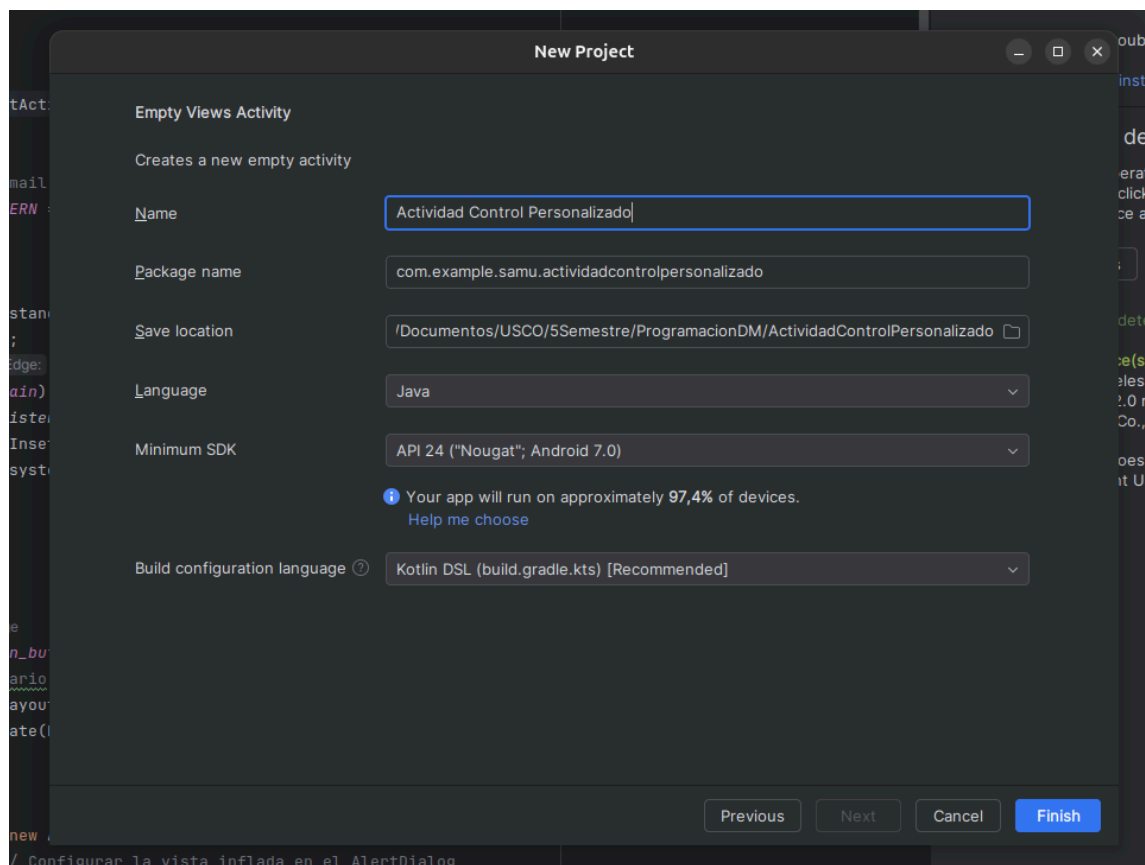
- Proceda a crear un nuevo proyecto, dirigiéndose a **New**, luego a **New project....**:



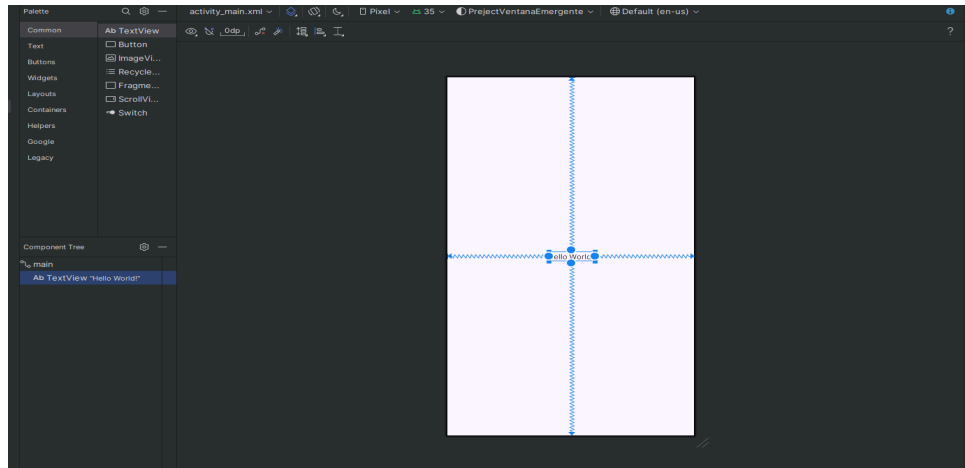
- Seguido a esto, saldrá una ventana que mostrará varias opciones para crear un nuevo proyecto, se debe escoger **Empty Views Activity** y luego oprimir **Next**:



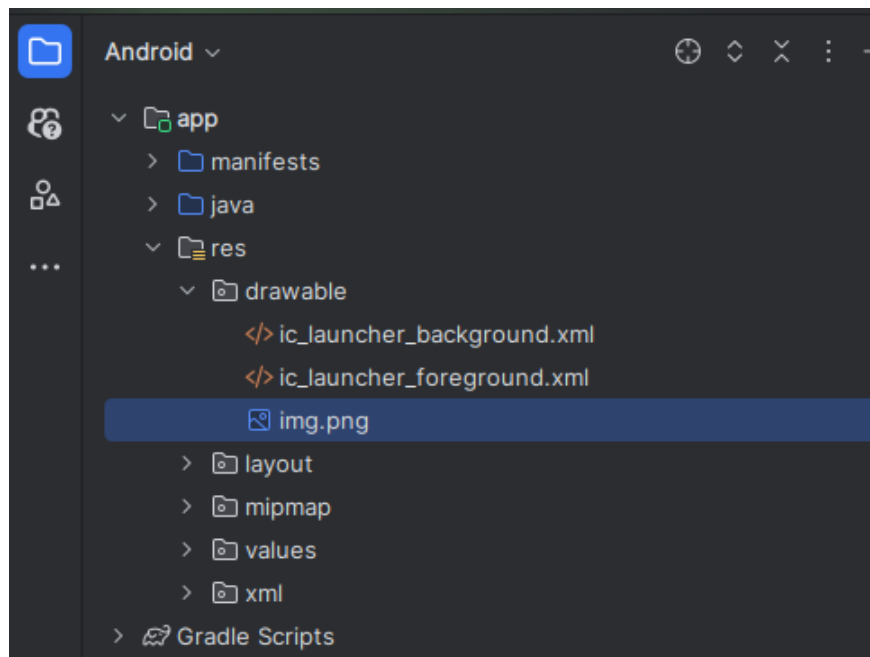
- Una ventana emergente permitirá configurar algunos aspectos del proyecto; se define como nombre **Actividad Control Personalizado**, verificar que el Package name sea **com.example.actividadcontrolpersonalizado**. Escoger **Java** como **Lenguaje** y en el apartado de **Minimum SDK** se escoge la versión de Android de su preferencia. Por temas de compatibilidad entre dispositivos se elige **Api 24 ("Nougat", Android 7.0)**. Los demás valores se dejan con la configuración que da Android Studio.



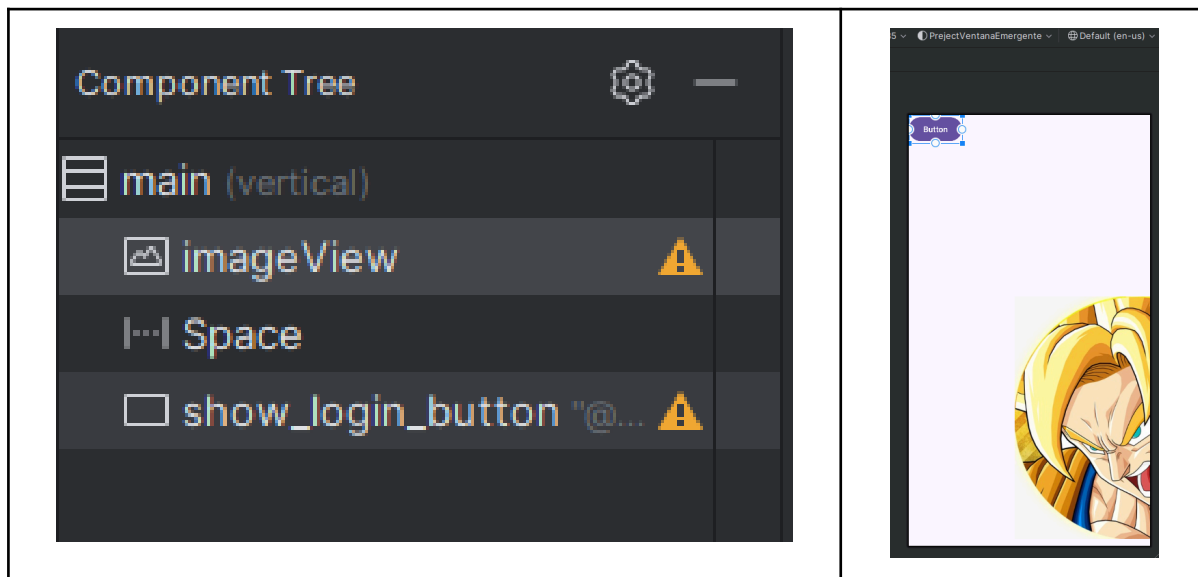
- Se borra el mensaje “*Hello World*” para obtener la plantilla en blanco.



- Terminada la configuración, se abrirá el proyecto. Al dirigirse a la parte superior izquierda se mostrarán todas las carpetas que lo componen. Proceda a abrir la carpeta *res* luego la carpeta *drawable* y pegue la imagen deseada para cargarla y es la que acompañará la vista.



- Teniendo definida la imagen, se procede a colocar un **Space** que se encuentra en el apartado de **Layout**, y luego se pone un **Button** el cual se encuentra en el apartado **Common**. Esto para darle un espacio en la ventana y colocar un botón. Se procede a definir los elementos en el Component Tree de la siguiente manera:



Se dirige al archivo **activity_main.xml**, en el cual se encuentra la configuración que debe llevar la pantalla, para definir los atributos de los elementos. Puede copiar la siguiente tabla para agregarla al documento.

```
<?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"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center_horizontal|top|center_vertical"
    android:orientation="vertical"
    tools:context=".MainActivity">

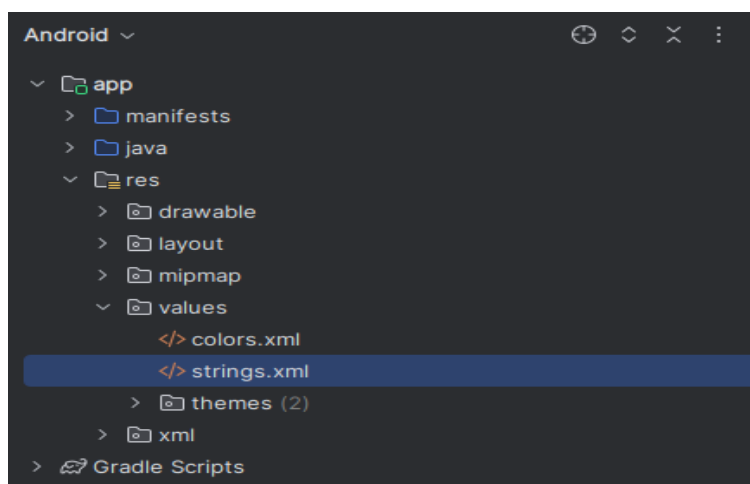
    <ImageView
        android:id="@+id/imageView"
        android:layout_width="200dp"
        android:layout_height="200dp"
        android:layout_marginTop="30dp"
        app:srcCompat="@drawable/imagen_usuario" />

    <Space
        android:layout_width="match_parent"
        android:layout_height="30dp" />

    <Button
        android:id="@+id/show_login_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:layout_centerInParent="true"
        android:onClick="onClick"
        android:text="@string/main_button"
        android:textSize="20sp" />
    </LinearLayout>
```

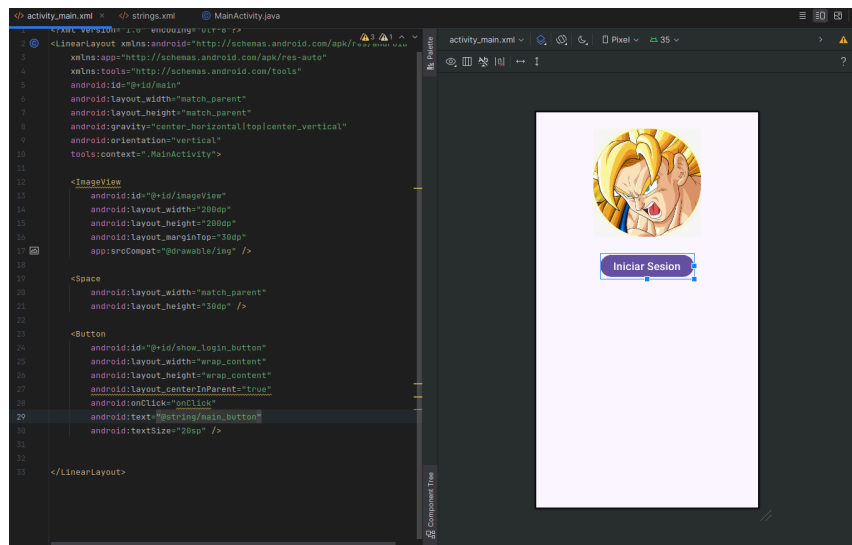
- Después, dirigirse nuevamente a la parte superior izquierda de la aplicación, donde se encuentran las carpetas del proyecto, y abrir la carpeta **values** que se encuentra en la carpeta **res**, ahí se encuentra el archivo **strings.xml**, proceda a abrirlo:



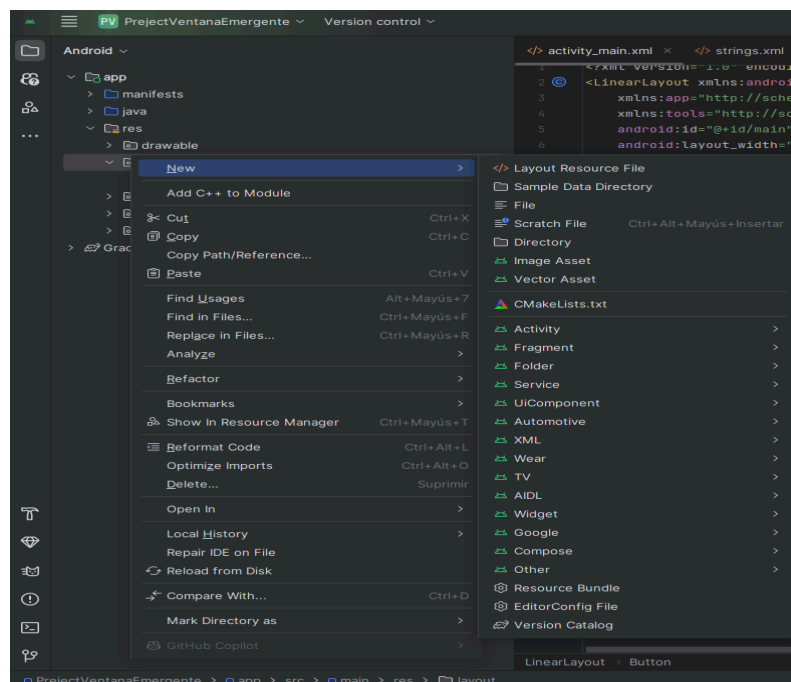
- Abierto el archivo **strings.xml**, se usará para cargar de forma dinámica los textos que aparecerán en la ventana. En este caso se usa para nombrar el botón **Iniciar sesión**:

```
<resources>
    <string name="app_name">actividadcontrolpersonalizado</string>
    <string name="main_button">Iniciar sesión</string>
</resources>
```

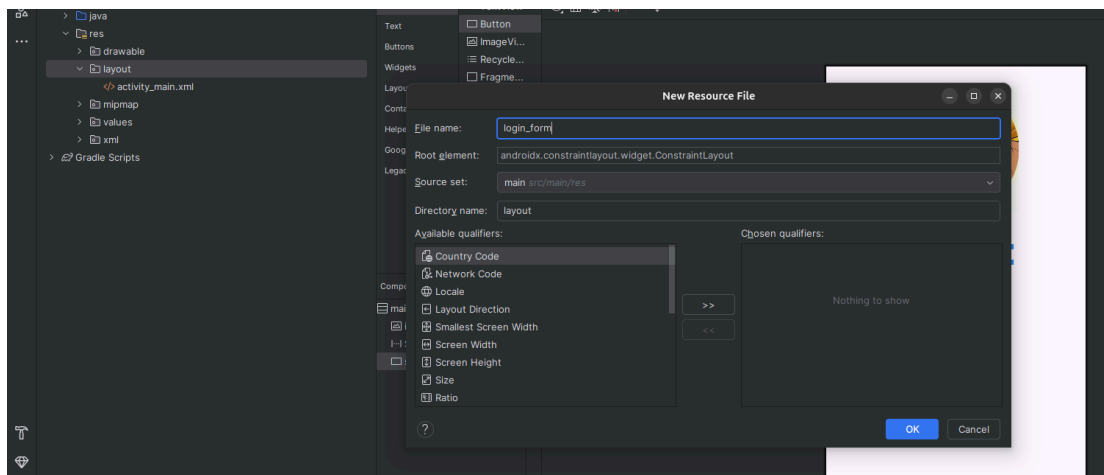

- Siguiendo las indicaciones anteriores, la vista debe lucir de la siguiente manera



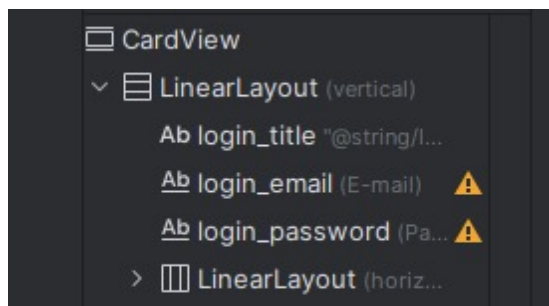
- Terminado esto, dirigirse nuevamente donde se encuentran todas las carpetas que componen el proyecto (Icono de carpeta de la barra lateral izquierda, seleccionar la carpeta **layout**, oprimir click derecho, **New** y **Layout Resource File**. Este documento contendrá la vista del formulario de inicio de sesión.



- Se abrirá una ventana emergente que mostrará la configuración para la vista que se acaba de crear. Darle el nombre **login_form** de y dejar la demás configuración como la de Android Studio:



- Dirigirse al archivo recién creado y agregar los siguientes elementos al **Component Tree**, los cuales se encuentran en la vista de diseño de Android Studio, en la paleta bajo los elementos **Text** y en **Layout**.



- Al abrir la configuración del **login_form.xml**, se encontrará los atributos definidos del Component Tree, a continuación tomar el código de la siguiente tabla y pegarlo en el archivo **login_form.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
    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="wrap_content"
    android:layout_margin="30dp"
    android:gravity="center"
    app:cardBackgroundColor="@color/white"
    app:cardCornerRadius="30dp"
    app:cardElevation="20dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:orientation="vertical">
```

```
android:padding="24dp">

<TextView
    android:id="@+id/login_title"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/login_title"
    android:textAlignment="center"
    android:textColor="@color/lavender"
    android:textSize="36sp"
    android:textStyle="bold" />

<EditText
    android:id="@+id/login_email"
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:layout_marginTop="40dp"
    android:background="@color/white"
    android:drawablePadding="8dp"
    android:hint="@string/email_hint"
    android:inputType="textEmailAddress"
    android:padding="8dp"
    android:textColor="@color/black" />

<EditText
    android:id="@+id/login_password"
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:layout_marginTop="20dp"
    android:background="@color/white"
    android:drawablePadding="8dp"
    android:hint="@string/password_hint"
    android:inputType="textPassword"
    android:padding="8dp"
    android:textColor="@color/black" />

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginTop="30dp"
    android:orientation="horizontal">

    <Button
        android:id="@+id/login_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:backgroundTint="@color/lavender"
        android:text="@string/login_button"
        android:textSize="18sp"
        app:cornerRadius="20dp" />

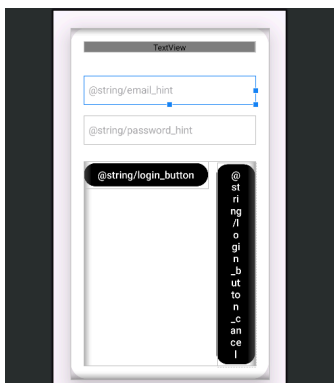
    <Button
        android:id="@+id/cancel_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="15dp"
        android:backgroundTint="@color/lavender"
        android:text="@string/login_button_cancel"
        android:textSize="18sp"
        app:cornerRadius="20dp" />
```

```
</LinearLayout>

</LinearLayout>

</androidx.cardview.widget.CardView>
```

- Después de colocar los atributos dados en la anterior tabla en el archivo **login_form.xml**, se debe visualizar lo siguiente en la pestaña de diseño (Icono de paisaje a la derecha de la barra superior)



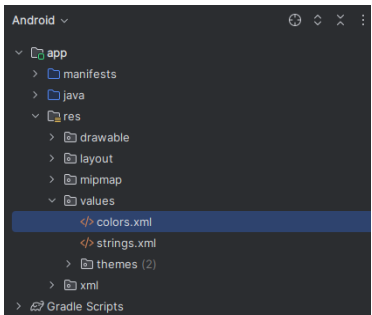
- Nuevamente el archivo **strings.xml**, se usará para cargar de forma dinámica los textos que aparecerán en la ventana. En este caso se usa para nombrar el string, **Ingreso**, **Email**, **Contraseña** y el botón **Ok** y **Cancelar** :

```
<resources>
  <string name="app_name">actividadcontrolpersonalizado</string>

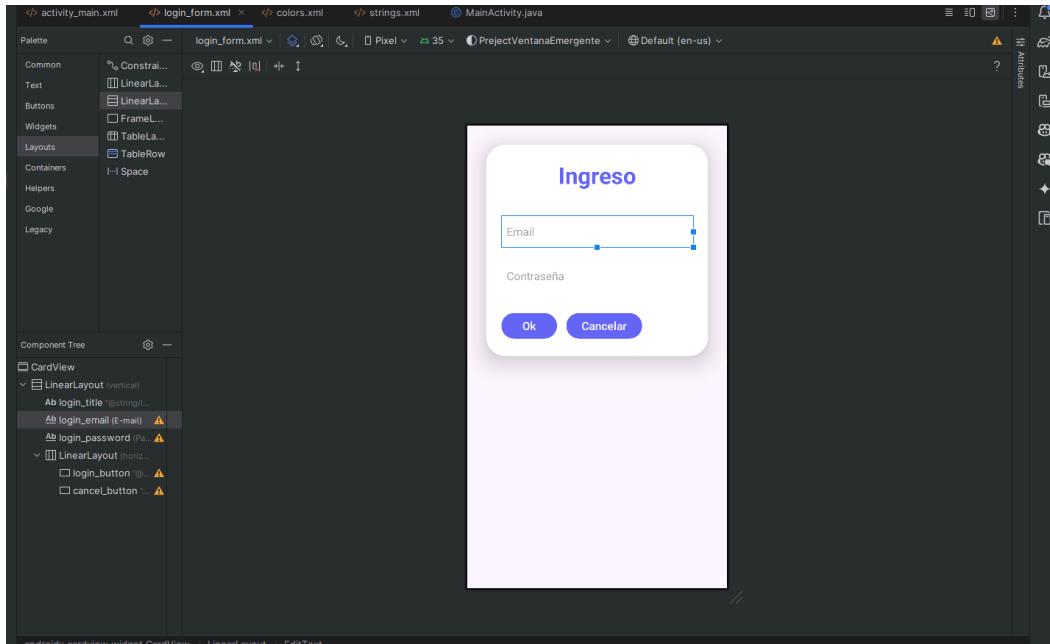
  <string name="login_title">Ingreso</string>
  <string name="email_hint">Email</string>
  <string name="password_hint">Contraseña</string>
  <string name="login_button">Ok</string>
  <string name="login_button_cancel">Cancelar</string>

  <string name="main_button">Iniciar sesión</string>
</resources>
```

- Dirigirse nuevamente donde se encuentran las carpetas que componen el proyecto, abrir la carpeta **res**, después dirigirse a **values** y luego a **colors**, para abrir el archivo **colors.xml** y editarlo:

	<pre><?xml version="1.0" encoding="utf-8"?> <resources> <color name="lavender">#6767F6</color> <color name="black">#000000</color> <color name="white">#FFFFFF</color> </resources></pre>
---	---

- Si se han seguido todos los pasos correctamente, en el archivo *login_form.xml* se debe visualizar lo siguiente:



- Se procede con la lógica del programa, se dirige al archivo **MainActivity.java** y se copia y pega el siguiente script en java. Sí el nombre del proyecto es diferente debe tener en cuenta la dirección del archivo, es decir el package y el nombre de la clase. Dentro de éste, señalado bajo comentarios (Doble slash) se encuentra explicado cada una de los bloques de código que se ocupan de una función en específico. Para referencias al respecto, consultar la documentación oficial de [Android Studio](https://developer.android.com/studio) sobre el tema:

```

package com.example.actividadcontrolpersonalizado;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class MainActivity extends AppCompatActivity {

    // Expresión regular para validar un email
    private static final String EMAIL_PATTERN = "[a-zA-Z0-9._-]+@[a-z]+\\.+[a-z]+";

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

```
ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
    Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
    v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
    return insets;
});
}

public void onClick(View view) {
    if (view.getId() == R.id.show_login_button) {
        // Inflar el layout del formulario de login
        LayoutInflater inflater = getLayoutInflater();
        View loginView = inflater.inflate(R.layout.login_form, null);

        // Crear el AlertDialog
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setView(loginView); // Configurar la vista inflada en el AlertDialog
        AlertDialog dialog = builder.create();

        dialog.setCancelable(false); // Configurar la vista para que no se cierre

        // Obtener los botones del formulario
        Button loginButton = loginView.findViewById(R.id.login_button);
        Button cancelButton = loginView.findViewById(R.id.cancel_button);

        // Manejar el evento del botón Login
        loginButton.setOnClickListener(v -> {
            // Obtener los campos de texto (email y contraseña)
            EditText emailInput = loginView.findViewById(R.id.login_email);
            EditText passwordInput = loginView.findViewById(R.id.login_password);

            // Obtener los valores ingresados
            String email = emailInput.getText().toString();
            String password = passwordInput.getText().toString();

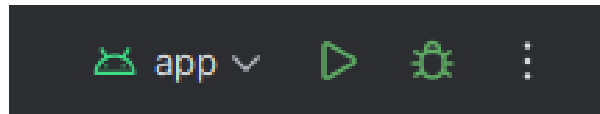
            // Lógica para el login
            if (!email.isEmpty() && !password.isEmpty()) {
                if (isValidEmail(email)) {
                    if (email.equals("usuario@gmail.com") && password.equals("1234")) {
                        Toast.makeText(MainActivity.this, "Bienvenido", Toast.LENGTH_SHORT).show();
                        dialog.setCancelable(true);
                        dialog.dismiss(); // Cerrar el diálogo después del login
                    } else {
                        Toast.makeText(this, "Datos erroneos", Toast.LENGTH_SHORT).show();
                    }
                } else {
                    Toast.makeText(this, "Email no valido", Toast.LENGTH_SHORT).show();
                }
            } else if (email.isEmpty()) {
                Toast.makeText(MainActivity.this, "Por favor ingrese el email",
Toast.LENGTH_SHORT).show();
                emailInput.requestFocus();
            } else {
                Toast.makeText(this, "Porfavor ingrese la contraseña", Toast.LENGTH_SHORT).show();
                passwordInput.requestFocus();
            }
        });

        // Manejar el evento del botón Cancel
        cancelButton.setOnClickListener(v -> {
            dialog.dismiss(); // Cerrar el diálogo si se cancela
        });

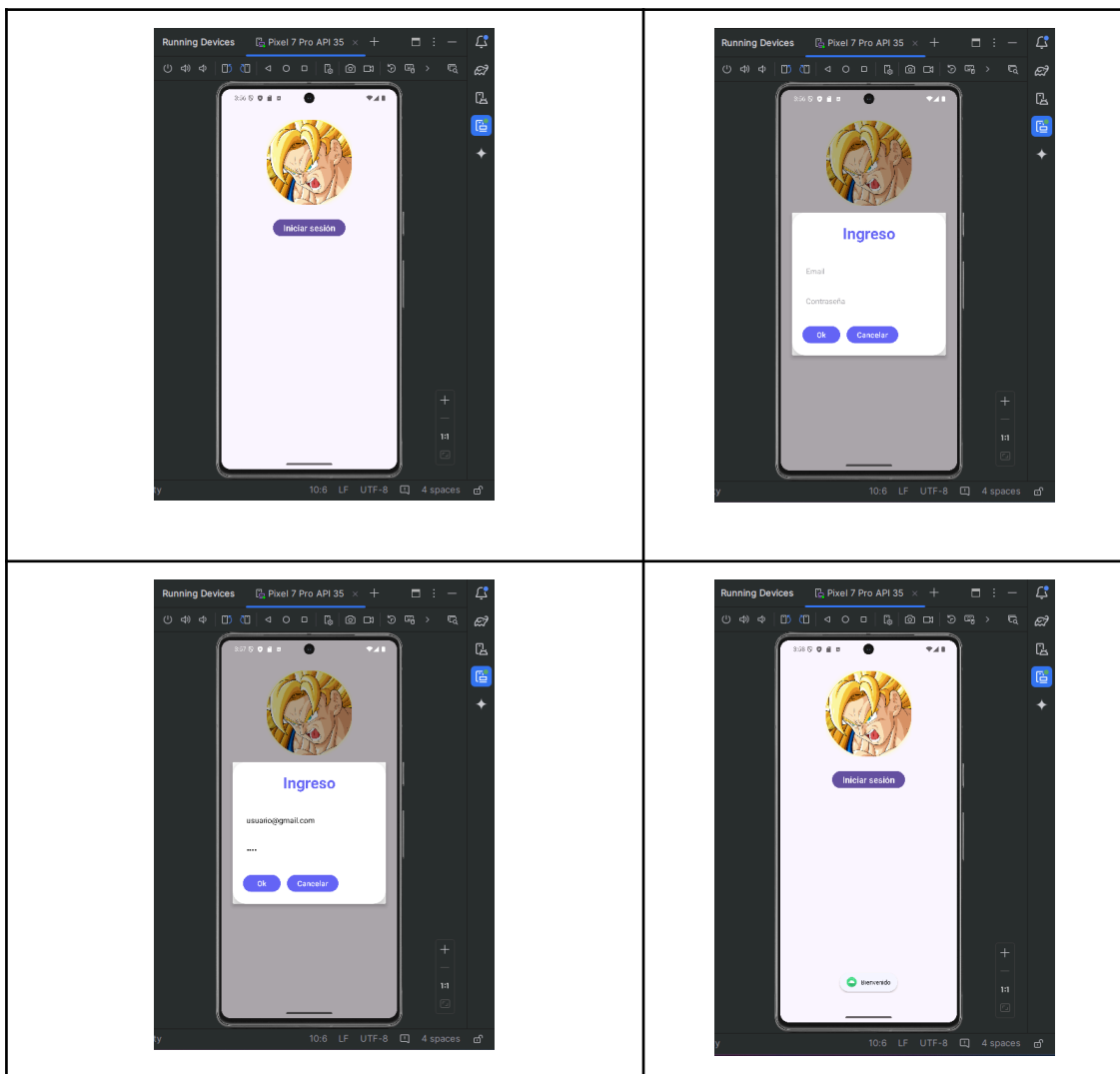
        dialog.show(); // Mostrar el AlertDialog
    }
}

// Método para verificar si el email tiene un formato válido
private boolean isValidEmail(String email) {
    Pattern pattern = Pattern.compile(EMAIL_PATTERN);
    Matcher matcher = pattern.matcher(email);
    return matcher.matches();
}
}
```

- Terminada la lógica para que el programa funcione, este se encuentra totalmente finalizado, por lo cual se procede a probar el programa. Para esto hay que dirigirse a la parte superior de Android Studio y dar click al botón de play.:



- El programa correrá en la máquina virtual que se tenga instalada en Android Studio. En el caso de no tenerla, dirigirse al siguiente link [Tutorial Máquina Virtual](#), en este encontrará un tutorial muy simple para realizar una máquina virtual en Android Studio; en caso de tener problemas en el momento de crear la máquina virtual o al intentar correrla por limitaciones del computador en el que se está trabajando, dirigirse al siguiente link [Tutorial Conectar Celular](#), en el cual se muestra un tutorial de como conectar un celular android y poder correr el programa en este, con las opciones de desarrollador (Depurador USB y/o Depurador Wi-Fi).
- Se debe visualizar lo siguiente al ejecutar la aplicación:



REFERENCIAS

Android Developers. (n.d.). Crear y manejar dispositivos virtuales..

<https://developer.android.com/studio/run/managing-avds>

Android Developers. (n.d.). Instalar Android Studio.

<https://developer.android.com/studio/install>

Android Developers. (n.d.). Correr aplicaciones en dispositivos de hardware..

<https://developer.android.com/studio/run/device>

Google. (s.f.). *Diálogos personalizados*. Android Developers.

<https://developer.android.com/develop/ui/views/components/dialogs?hl=es-419#CustomLayout>

Oracle. (n.d.). Documentación JDK 17. <https://docs.oracle.com/en/java/javase/17>

ANEXO

Este proyecto se encuentra en el siguiente repositorio de GitHub:

<https://github.com/Danitron775/Actividad-Android-Studio-Control-Personalizado>