



# **Actividad 2 - Pantalla de Registro**

## **Desarrollo de Aplicaciones Móviles I**

### **Ingeniería en Desarrollo de Software**

**Tutor: Humberto Jesús Ortega Vázquez**

**Alumno: Jusi Ismael Linares Gutiérrez**

**Fecha: 14/11/2023**

## Índice

Introducción .....	3
Descripción .....	3
Justificación .....	3
Desarrollo: .....	4
Interfaz .....	4
Codificación .....	6
Prueba de la aplicación .....	11
Link del video de la aplicación en ejecución: .....	11
Link del proyecto: .....	11
Conclusión .....	11
GitHub .....	11

## Introducción

Después de haber establecido la pantalla de inicio y la redirección automática a la pantalla de autenticación en la actividad anterior, ahora abordaremos la creación de la interfaz de registro de un usuario. Este componente es esencial para que los clientes puedan acceder a los servicios bancarios de manera segura y conveniente.

La pantalla de registro contendrá dos cajas de texto, una para el correo del usuario y otra para la contraseña. Sin embargo, la aplicación no solo se limita a la recolección de datos, sino que también realiza una serie de validaciones importantes. Se asegurará de que el usuario introduzca una dirección de correo electrónico válida y que la contraseña cumpla con ciertos criterios de seguridad, como la inclusión de mayúsculas, minúsculas, caracteres especiales y números.

Además, la aplicación proporcionará una respuesta en tiempo real al usuario. Si los datos introducidos son válidos, mostrará una alerta de "Bienvenido", lo que crea una experiencia amigable y segura para el usuario.

## Descripción

La pantalla de registro propuesta incluye dos cajas de texto para ingresar el correo electrónico del usuario y la contraseña. Para garantizar la seguridad de las cuentas de los usuarios, se han establecido requisitos estrictos para la contraseña, que incluyen al menos una mayúscula, una minúscula, un carácter especial y un número. Si los datos ingresados no cumplen con estos requisitos, la aplicación mostrará una pantalla modal para informar al usuario sobre la situación, lo que es fundamental para proteger las cuentas de los usuarios de posibles ataques o vulnerabilidades.

Además, la aplicación debe ser amigable y mostrar un mensaje de bienvenida apropiado según la hora del día (buenos días, buenas tardes o buenas noches), lo que mejora la experiencia del usuario. También se proporcionan botones para iniciar sesión o registrarse, lo que permite a los usuarios realizar estas acciones de manera sencilla.

## Justificación

La inclusión de una pantalla de registro de usuario en la aplicación móvil para servicios bancarios es una elección esencial. Proporciona una forma fácil para que los nuevos usuarios se unan a la aplicación. Además, la validación de la estructura del correo y la complejidad de las contraseñas mejora la seguridad al garantizar que los usuarios establezcan credenciales robustas.

La personalización con un saludo que varía según la hora del día agrega un toque amigable y cercano a la aplicación, mejorando la experiencia del usuario y fortaleciendo la relación con la marca. Además, la inclusión de funciones de registro y autenticación amplía la versatilidad de la aplicación, permitiéndole atender a un rango más amplio de usuarios y necesidades, lo que aumenta su atractivo y utilidad.

## Desarrollo:

### Interfaz

Al dar clic en registrarse aparecerá esta interfaz donde podremos ingresar nuestro correo y contraseña



The image shows a registration form titled "Registro" with a subtitle "Buenas noches". It features two input fields: "Correo Electronico" and "Contraseña". Below the fields is a purple button labeled "Registrarse". The form is set against a light pink background.

Este es el mensaje que aparecerá cuando el usuario no ingrese una contraseña valida



The image shows the same registration form as before, but with an error message displayed. The error message is titled "Error" and states: "La contraseña debe tener al menos una mayúscula, una minúscula, un carácter especial y un número." Below the message is a button labeled "Aceptar". The form is set against a dark gray background.

Este es el mensaje que aparecerá cuando el usuario ingrese un correo invalido



The screenshot shows a registration form titled "Registro" with the greeting "Buenas noches". It has two input fields: "Correo Electronico" and "Contraseña". The "Correo Electronico" field contains the text "jusi.lingu@gmail.com". Below the fields, there is a white error message box with the title "Error" and the text "El formato del correo electrónico no es válido." and a blue "Aceptar" button.

Registro

Buenas noches

Correo Electronico

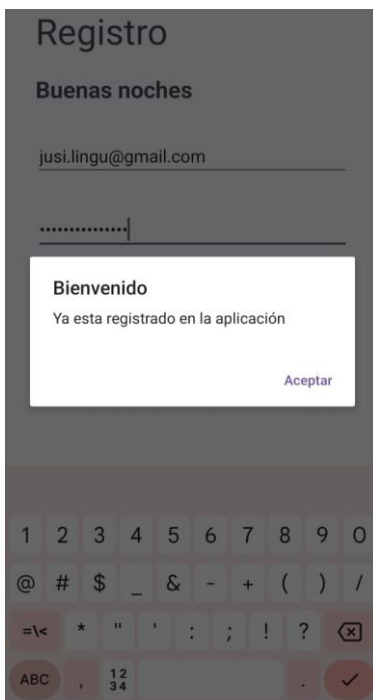
Contraseña

**Error**

El formato del correo electrónico no es válido.

Aceptar

Cuando se ingrese un correo y contraseña valido este será el mensaje de bienvenida que se mostrara al usuario



The screenshot shows the same registration form as above, but with a white welcome message box instead of an error. The "Correo Electronico" field now contains "jusi.lingu@gmail.com" and the "Contraseña" field is filled with dots. The welcome message box has the title "Bienvenido" and the text "Ya esta registrado en la aplicación" and a blue "Aceptar" button. A virtual keyboard is visible at the bottom of the screen.

Registro

Buenas noches

jusi.lingu@gmail.com

.....

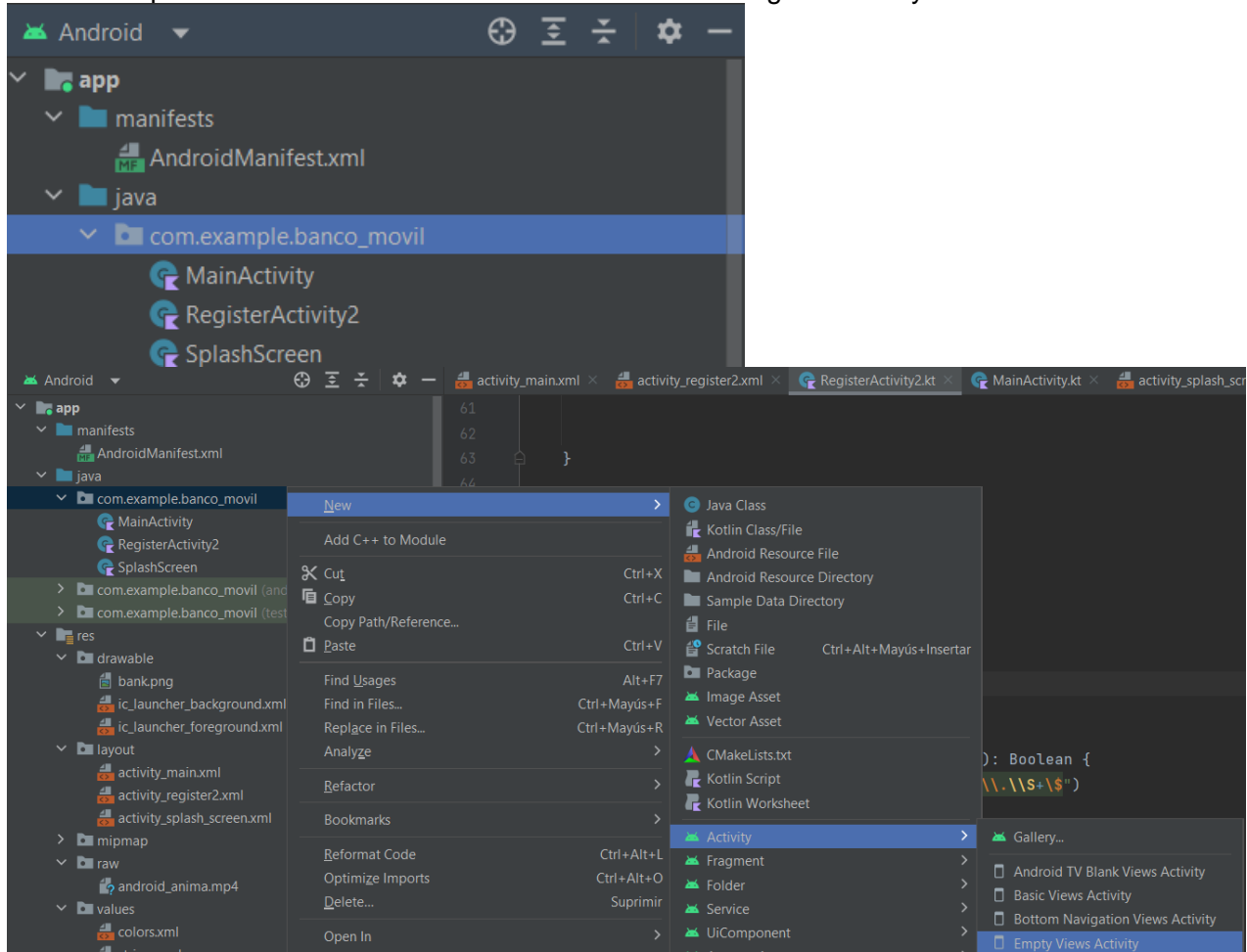
**Bienvenido**

Ya esta registrado en la aplicación

Aceptar

## Codificación

En esta carpeta crearemos una nueva actividad llamada RegisterActivity2



Una vez creado el archivo nos dirigimos primero al mainactivity.kt donde tomando en cuenta el id de nuestro botón de registro de nuestro activity\_main.xml

```
<Button
    android:id="@+id/tv_go_to_register"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginStart="32dp"
    android:layout_marginTop="32dp"
    android:layout_marginEnd="32dp"
    android:text="Registrarse"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/button" />
```

Escribiremos este código que nos servirá para que el botón de registro nos dirija a nuestro nuevo RegisterActivity2

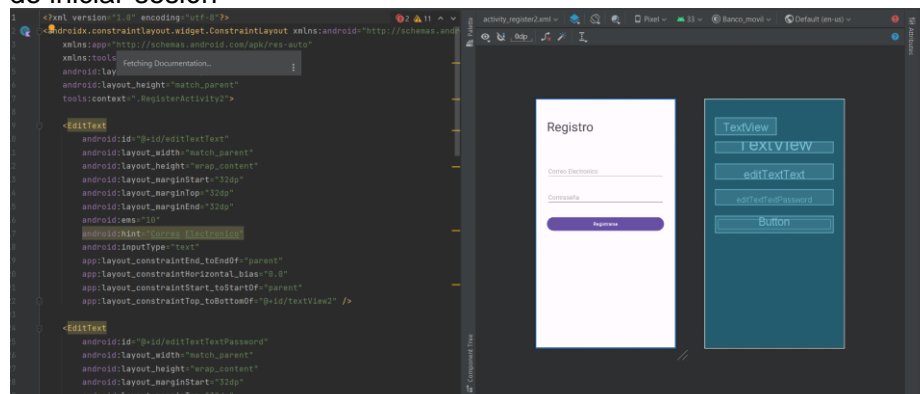
```
package com.example.banco_movil

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.View
import android.widget.Button

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        val tvGoRegister = findViewById<Button>(R.id.tv_go_to_register)
        tvGoRegister.setOnClickListener { it: View? -> {
            goToRegister()
        }}
    }

    private fun goToRegister() {
        val i = Intent(packageContext, RegisterActivity2::class.java)
        startActivity(i)
    }
}
```

Para nuestro activity\_register2.xml simplemente copiaremos y pegaremos todo el código de nuestro activity\_main.xml y borraremos lo que sea necesario (Como la imagen del banco y el botón de iniciar sesión)



Con estas líneas de código logramos que un TextView en nuestra interfaz de registros pueda mostrar el mensaje Buenos Días/Tardes/Noches dependiendo de la hora

```
// Obtener la hora actual
val horaActual = Calendar.getInstance().get(Calendar.HOUR_OF_DAY)
val saludo = obtenerSaludo(horaActual)

//Aquí se define el saludo dependiendo la hora
val textView2 = findViewById<TextView>(R.id.textView2)
textView2.text = saludo
```

Con este código se toma una hora como entrada y devuelve un saludo personalizado según la hora del día

```
//Saludo "Buen@s Días/Tardes/Noches"
private fun obtenerSaludo(hora: Int): String {
    return when (hora) {
        in 0 ≤ .. ≤ 11 -> "Buenos días"
        in 12 ≤ .. ≤ 17 -> "Buenas tardes"
        else -> "Buenas noches"
    }
}
```

estas líneas de código asignan referencias a elementos de la interfaz de usuario (dos campos de texto EditText y un botón Button) a variables en el código de Kotlin.

```
val editTextEmail = findViewById<EditText>(R.id.editTextText)
val editTextPassword = findViewById<EditText>(R.id.editTextTextPassword)
val buttonRegister = findViewById<Button>(R.id.button)
```

```
//Metodo para la estructura del correo
private fun formatoCorreoValido(correo: String): Boolean {
    val regexCorreo = Regex(pattern: "^\\S+@\\S+\\.\\S+$")
    return regexCorreo.matches(correo)
}
```



Esta parte del código comprueba si la cadena de correo electrónico proporcionada sigue el formato especificado por la expresión regular y devuelve un valor booleano indicando si es válido o no.

- `^`: Indica que la coincidencia debe comenzar desde el inicio de la cadena.
- `\S+`: Coincide con uno o más caracteres que no son espacios en blanco.
- `@`: Coincide con el símbolo '@'.
- `\\S+`: Otra vez, coincide con uno o más caracteres que no son espacios en blanco.
- `\\.`: Coincide con el símbolo de punto '.'.
- `\\S+`: Nuevamente, coincide con uno o más caracteres que no son espacios en blanco.
- `$`: Indica que la coincidencia debe llegar hasta el final de la cadena.

```
//Metodo para la estructura del correo
private fun formatoCorreoValido(correo: String): Boolean {
    val regexCorreo = Regex( pattern: "^\\S+@\\S+\\.\\S+$")
    return regexCorreo.matches(correo)
}
```

Se verifica la validez de una contraseña que cumpla con los requisitos que pide el PDF de la actividad, al final devuelve un valor booleano indicando si la contraseña cumple con los requisitos.

- `val mayuscula = Regex("[A-Z]").containsMatchIn(contrasena)`: Declara una variable `mayuscula` y utiliza una expresión regular para verificar si la contraseña (`contrasena`) contiene al menos una letra mayúscula. La función `containsMatchIn` devuelve `true` si encuentra una coincidencia en la cadena y `false` si no.
- `val minuscula = Regex("[a-z]").containsMatchIn(contrasena)`: Similar al caso anterior, verifica si la contraseña contiene al menos una letra minúscula.
- `val caracterEspecial = Regex("[!@#$%^&*(),.?\\":{}|<>]").containsMatchIn(contrasena)`: Comprueba si la contraseña contiene al menos un carácter especial, como un símbolo de exclamación, arroba, almohadilla, etc.
- `val numero = Regex("[0-9]").containsMatchIn(contrasena)`: Verifica si la contraseña contiene al menos un número.
- `return mayuscula && minuscula && caracterEspecial && numero`: Devuelve `true` si la contraseña cumple con todos los requisitos (mayúsculas, minúsculas, carácter especial y número) y `false` si no cumple con alguno de ellos.

```
//Metodo para la estructura de la contraseña
private fun contrasenaValida(contrasena: String): Boolean {
    val mayuscula = Regex( pattern: "[A-Z]").containsMatchIn(contrasena)
    val minuscula = Regex( pattern: "[a-z]").containsMatchIn(contrasena)
    val caracterEspecial = Regex( pattern: "[!@#$%^&*(),.?\\":{}|<>]").containsMatchIn(contrasena)
    val numero = Regex( pattern: "[0-9]").containsMatchIn(contrasena)

    // Comprobar si la contraseña cumple con todos los requisitos
    return mayuscula && minuscula && caracterEspecial && numero
}
```

Esta función toma un título y un mensaje como parámetros, crea un cuadro de diálogo con ese título y mensaje, y muestra un botón "Aceptar" que, cuando se presiona, cierra el cuadro de diálogo.

```
//Para mostrar las alertas de correos/contraseñas erróneas
private fun mostrarAlerta(titulo: String, mensaje: String) {
    val builder = AlertDialog.Builder(context: this)
    builder.setTitle(titulo)
    builder.setMessage(mensaje)
    builder.setPositiveButton(text: "Aceptar") { dialog, _ ->
        dialog.dismiss()
    }
    val dialog = builder.create()
    dialog.show()
}
```

Esto mas adelante lo utilizare para un IF, solo son variables booleanas con el valor de false

```
//Variables booleanas para validar que el correo y contraseña tienen el formato correcto
var buenCorreo: Boolean = false
var buenaContraseña: Boolean = false
```

Con este código al dar clic en el botón de registrar tras correr los métodos estos regresan un valor true/false y dependiendo el valor se decidirá si se mostrara el mensaje de Error o a su respectiva variable se le asignara un valor true

```
buttonRegister.setOnClickListener { it.view()
    //Validar formato del correo
    if (formatoCorreoValido(editTextEmail.text.toString())) {
        // El formato del correo electrónico es válido
        buenCorreo = true
    } else {
        // El formato del correo electrónico no es válido
        mostrarAlerta(titulo: "Error", mensaje: "El formato del correo electrónico no es válido.")
    }

    // Validar formato de la contraseña
    if (contraseñaValida(editTextPassword.text.toString())) {
        // La contraseña es válida
        buenaContraseña = true
    } else {
        // La contraseña no cumple con los requisitos
        mostrarAlerta(
            titulo: "Error",
            mensaje: "La contraseña debe tener al menos una mayúscula, una minúscula, un carácter especial y un número."
        )
    }
}
```

Con este IF validamos que el correo y contraseña sean correctos, entonces se mostrara el mensaje de bienvenida

```
//El formato del correo y la contraseña son correctos por lo que se muestra el mensaje de Bienvenida
if(buenCorreo == true && buenaContrasena == true) {
    mostrarAlerta(
        titulo: "Bienvenido",
        mensaje: "Ya esta registrado en la aplicación"
    )
}
```

## Prueba de la aplicación

Link del video de la aplicación en ejecución:

<https://drive.google.com/file/d/1DkCD9oXXZkXELrrMf-GF4vib-yJF1K6N/view?usp=sharing>

Link del proyecto:

<https://drive.google.com/file/d/1SKJJmN-ltkVrvLZGY0BwFaMAKWxUJnUD/view?usp=sharing>

## Conclusión

En el proceso de crear la aplicación para la unidad de servicios bancarios, hemos logrado diseñar una pantalla de registro que facilita a los usuarios ingresar sus datos de manera segura. La aplicación verifica que el correo tenga un formato adecuado y que la contraseña sea lo suficientemente complicada para proteger la cuenta del usuario.

Una ventaja es que, si algo no está bien al registrarse, la aplicación le avisa al usuario de inmediato mediante una ventana emergente. Esto es muy útil porque ayuda a corregir errores de manera rápida. Después de un registro exitoso, la aplicación da la bienvenida al usuario con un mensaje amigable. Además, se ha agregado un toque especial al mostrar saludos personalizados de acuerdo con la hora del día. En el trabajo diario, este proyecto destaca la importancia de cuidar los detalles y pensar en la comodidad del usuario al desarrollar software. La aplicación no solo cumple con las reglas técnicas, sino que también intenta hacer la experiencia del usuario más agradable.

## GitHub

[https://github.com/JusiLinGu/Practicas\\_UMI](https://github.com/JusiLinGu/Practicas_UMI)