



# **Actividad [#2] - [Pantalla de Registro]**

## **[Desarrollo de Aplicaciones Móviles I]**

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

**Tutor: Eduardo Israel Castillo García**

**Alumno: Alan David López Rojas**

**Fecha: 17/11/2022**

# Índice

Introducción.....	pág. 3
Descripción.....	pág. 3
Justificación.....	pág. 4
Investigación.....	pág. 4
Desarrollo.....	pág. 7
Interfaz.....	pág. 7
Codificación.....	pág. 11
Prueba de la aplicación.....	pág. 17
Conclusión.....	pág. 23
Bibliografía.....	pág. 24

## Introducción

El Login es nombre dado al momento de autenticación al ingresar a un servicio o sistema.

En el momento que se inicia el Login, el usuario entra en una sesión, para ello se emplea usualmente un nombre de usuario y contraseña. Actualmente el procedimiento de Login puede ser también a través de la lectura de las huellas dactilares (con un lector de huellas) como es el caso de los celulares que integran un lector de huellas digitales, identificación morfológica del rostro (empleando una cámara), por escaneo del iris del ojo, entre otras formas. Continuando con lo que se hizo en la actividad 1, en esta ocasión se pide lo siguiente para el mismo sistema de servicios bancarios:

Requerimientos de interfaz:

- Pantalla de registro con dos cajas de texto:
  - o La primera caja servirá para ingresar el usuario.
  - o La segunda caja para ingresar la contraseña

## Descripción

En esta siguiente actividad se desarrollará la pantalla de autenticación o Login, con la cual el usuario que desee acceder a su cuenta de banco, deberá ingresar con su correo y contraseña en caso de tener una cuenta existente o registrarse para crear una nueva cuenta de banco, para ello se continua con el programa elaborado en Android studio, que es la pantalla de presentación precedida de la pantalla de autenticación, se realizara la pantalla del login con cajas de texto en donde se ingresara el correo y la contraseña, la interfaz de registro de un usuario tiene los requerimientos siguientes:

- o El sistema validará que el usuario tenga la estructura de un correo.
- o El sistema validará que la contraseña tenga al menos una mayúscula, una minúscula, un carácter especial y un número. Si esto no sucede, aparecerá una pantalla modal para indicarle al usuario esta situación.
- o Si se ingresa un usuario y contraseña válidos, el sistema mostrará una alerta con el texto “Bienvenido”.

o Un botón para ingresar y un botón para registrar.

o Una leyenda que indique “Buenos días”, “Buenas tardes” o “Buenas noches”, de acuerdo con la hora del día

## **Justificación**

Derivado de la solicitud de implementar una nueva aplicación de servicios bancarios, se tiene que implementar un control de seguridad para el cliente quien entrara a su cuenta personal, para ello se hará una pantalla de Login la cual verificara la información que el usuario proporciona y si las credenciales coinciden con las registradas permitirá el acceso, la importancia de este paso de seguridad radica en que el sistema debe asegurarse que se cumplan las condiciones de tener un correo electrónico y contraseña que pide tener ciertas condiciones para ser segura, por lo cual el que el ususuario escriba bien estos campos depende si accede o manda un mensaje de error, esto ayudara a mejorar la seguridad para el usuario y evitar suplantación de identidad.

## **Investigación**

### ● ¿Cuáles son las características de Android Studio?

Entre las características de Android Studio se puede destacar que es un sistema de compilación flexible basado en Gradle cargado de funciones con las que desarrollar soluciones para todos los dispositivos Android. Además, dispone de herramientas de Lint para identificar problemas de rendimiento, usabilidad y compatibilidad de versiones.

Principales características que incluye Android Studio:

- Soporte para programar aplicaciones para Android Wear (sistema operativo para dispositivos corporales como por ejemplo un reloj).
- Herramientas Lint (detecta código no compatible entre arquitecturas diferentes o código confuso que no es capaz de controlar el compilador) para detectar problemas de rendimiento, usabilidad y compatibilidad de versiones.
- Utiliza ProGuard para optimizar y reducir el código del proyecto al exportar a APK (muy útil para dispositivos de gama baja con limitaciones de memoria interna).
- Integración de la herramienta Gradle encargada de gestionar y automatizar la construcción de proyectos, como pueden ser las tareas de testing, compilación o empaquetado.
- Nuevo diseño del editor con soporte para la edición de temas.
- Nueva interfaz específica para el desarrollo en Android.

- Permite la importación de proyectos realizados en el entorno Eclipse, que a diferencia de Android Studio (Gradle) utiliza ANT.
- Posibilita el control de versiones accediendo a un repositorio desde el que poder descargar Mercurial, Git, Github o Subversion.
- Alertas en tiempo real de errores sintácticos, compatibilidad o rendimiento antes de compilar la aplicación.
- Vista previa en diferentes dispositivos y resoluciones.
- Integración con Google Cloud Platform, para el acceso a los diferentes servicios que proporciona Google en la nube.
- Editor de diseño que muestra una vista previa de los cambios realizados directamente en el archivo xml.

- ¿Qué es un emulador?

Un emulador es una aplicación de software que permite al sistema funcionar como si se estuviera utilizando un terminal o una impresora diferente. Un emulador de terminal se conecta a un sistema principal para acceder a los datos o las aplicaciones. Algunos de estos emuladores proporcionan un recurso para transferir archivos con el sistema principal. Otros facilitan una interfaz de programas de aplicación (API) que hace posible la comunicación entre programas y la automatización de tareas del sistema principal.

Los emuladores no solo se limitan a sistemas operativos. Estos pueden ser usados, además por una arquitectura de hardware específica. Un programa es escrito para que se ejecute en un sistema base específico. Pues el emulador sencillamente crea una simulación de ese sistema base para ejecutar el programa en un ambiente diferente.

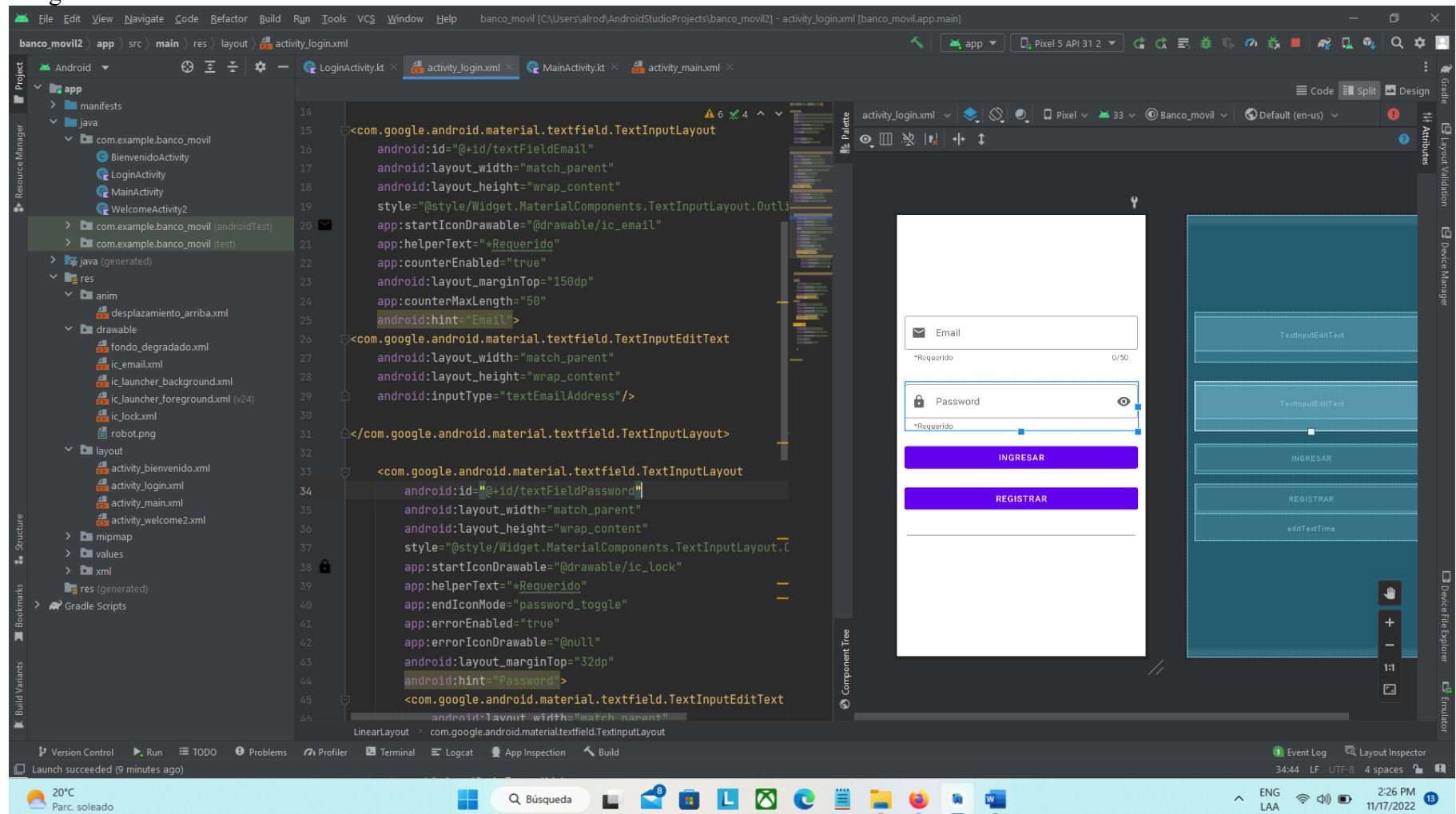
Por ejemplo, uno de los sistemas operativos más populares para teléfonos móviles es el sistema Android. Es bastante útil encontrar algún emulador que permita ejecutar este sistema en sistemas Windows. En este caso, uno de los más usados es el emulador BlueStack, el cual brinda muy buenas funciones, especialmente para ejecutar juegos.



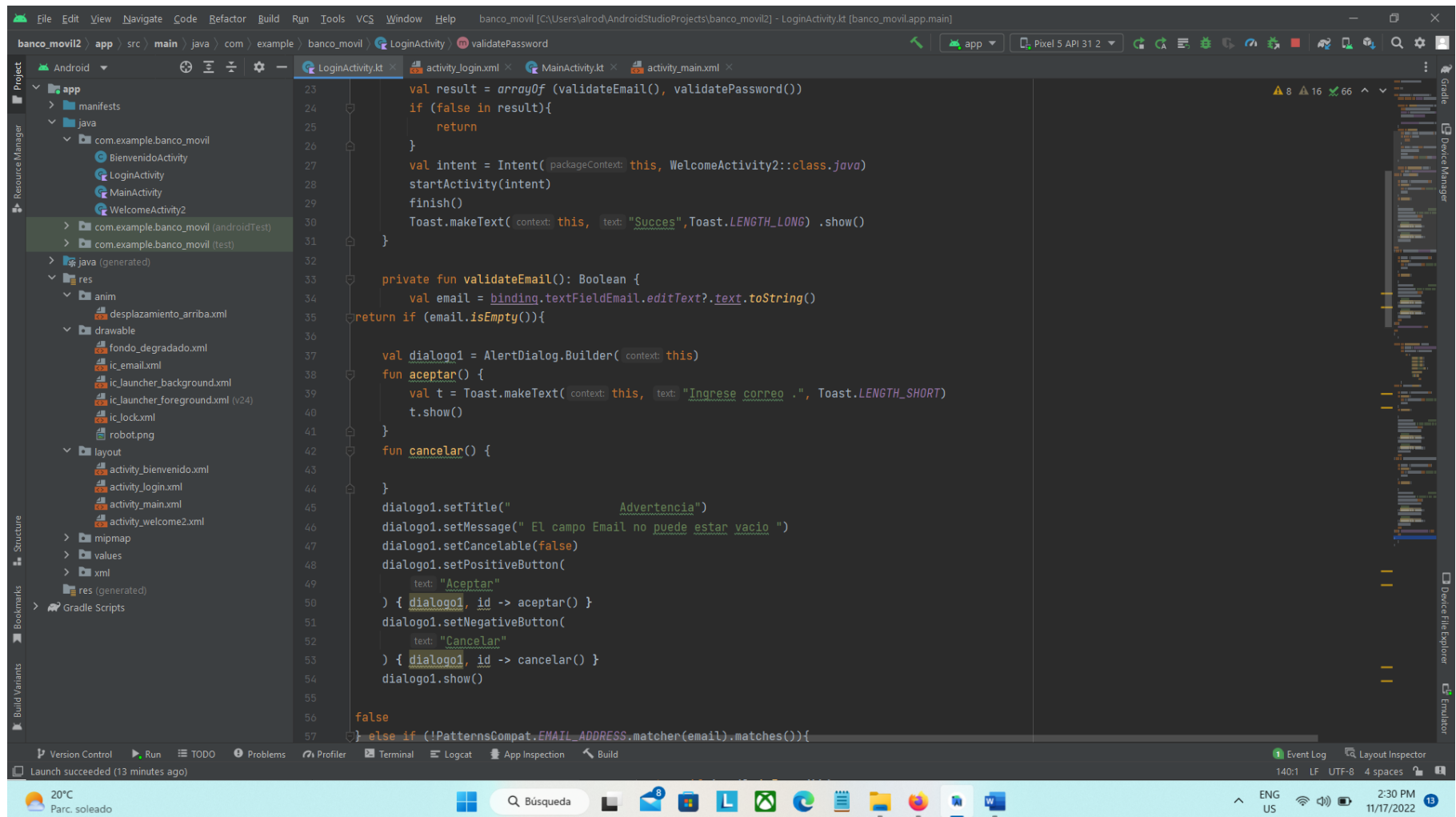
## Desarrollo

### Interfaz

Continuando desde la actividad pasada, se comienza con la elaboración del diseño de la pantalla de Login el cual lleva por nombre “activity\_login”, la cual llevara los cuadros de texto de Email y Password, que pedirá al usuario para acceder a su cuenta, agregue la opción de mostrar u ocultar el Password para que el usuario vea que está escribiendo, también se pide agregar dos botones que serán “Ingresar” y “Registrarse”.



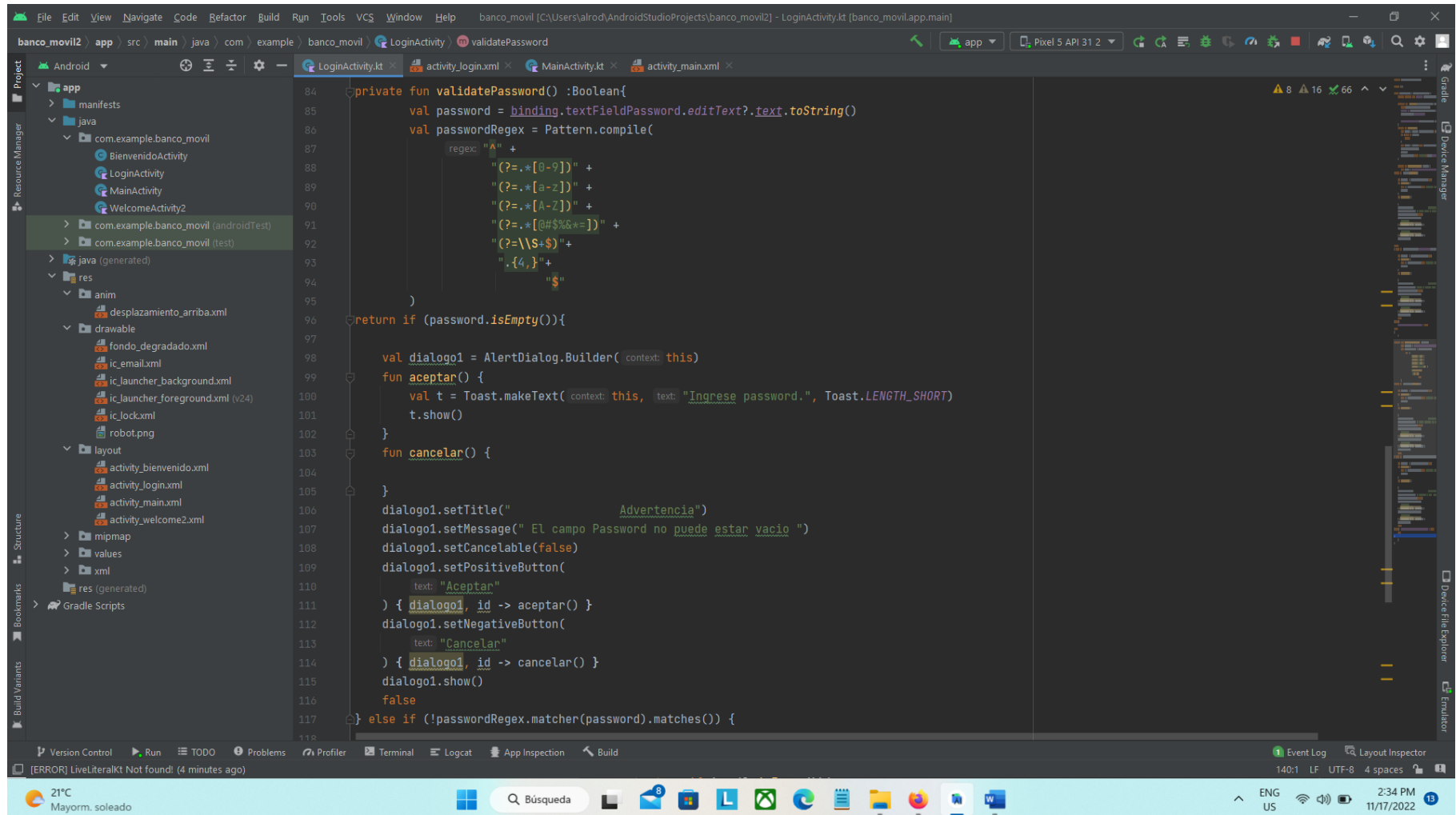
Una vez que se tiene hecho el diseño de la pantalla de Login, se hace el código para la funcionalidad en la pestaña de LoginActivity ahí se implementó el código kotlin para el funcionamiento de la aplicación, como lo es los cuadros de error que saltan en caso de no que la estructura del correo electrónico no sea la adecuada o el Password no cuenta con los caracteres.



```
23 val result = arrayOf(validateEmail(), validatePassword())
24 if (false in result){
25     return
26 }
27 val intent = Intent(packageContext: this, WelcomeActivity2::class.java)
28 startActivity(intent)
29 finish()
30 Toast.makeText(context: this, text: "Success", Toast.LENGTH_LONG).show()
31 }
32
33 private fun validateEmail(): Boolean {
34     val email = binding.textFieldEmail.editText?.text.toString()
35     return if (email.isEmpty()){
36
37         val dialogo1 = AlertDialog.Builder(context: this)
38         fun aceptar() {
39             val t = Toast.makeText(context: this, text: "Ingreso correo .", Toast.LENGTH_SHORT)
40             t.show()
41         }
42         fun cancelar() {
43
44         }
45         dialogo1.setTitle("Advertencia")
46         dialogo1.setMessage("El campo Email no puede estar vacío ")
47         dialogo1.setCancelable(false)
48         dialogo1.setPositiveButton(
49             text: "Aceptar"
50         ) { dialogo1, id -> aceptar() }
51         dialogo1.setNegativeButton(
52             text: "Cancelar"
53         ) { dialogo1, id -> cancelar() }
54         dialogo1.show()
55
56         false
57     } else if (!PatternsCompat.EMAIL_ADDRESS.matcher(email).matches()){
```



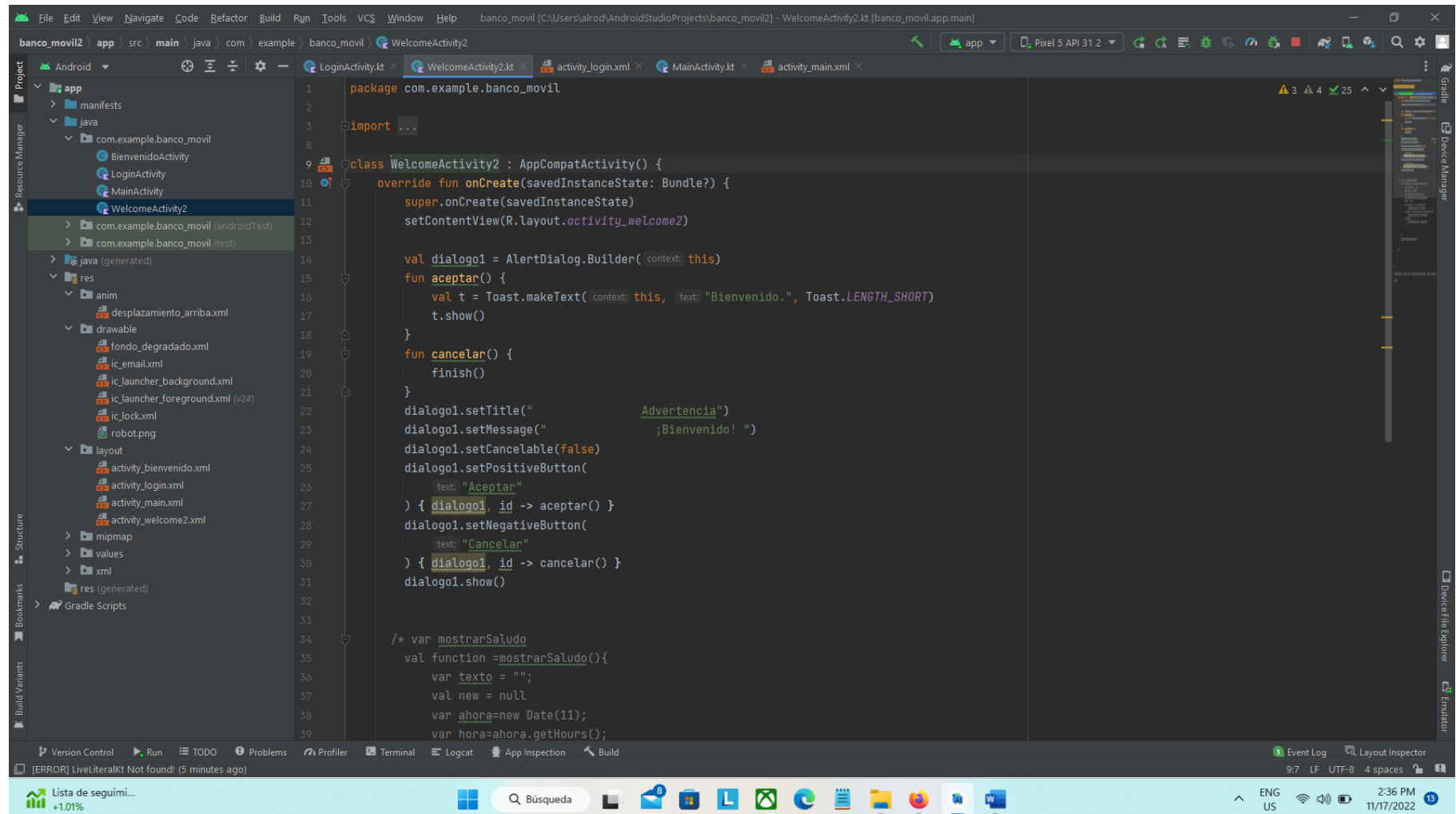
Se agrego también el código para evitar que los campos queden vacíos y mandar un mensaje de error en caso de que eso suceda, para evitar que se ingrese a la siguiente pantalla sin haber escrito el correo y contraseña.



```
84 private fun validatePassword() :Boolean{
85     val password = binding.textFieldPassword.editText?.text.toString()
86     val passwordRegex = Pattern.compile(
87         regex: "^(?=.*[0-9])" +
88             "(?=.*[a-z])" +
89             "(?=.*[A-Z])" +
90             "(?=.*[@#%&*+=])" +
91             "(?=\w{4,})" +
92             "$"
93     )
94
95     return if (password.isEmpty()){
96
97         val dialogo1 = AlertDialog.Builder(context: this)
98         fun aceptar() {
99             val t = Toast.makeText(context: this, text: "Ingrese password.", Toast.LENGTH_SHORT)
100             t.show()
101         }
102         fun cancelar() {
103
104         }
105     }
106     dialogo1.setTitle("Advertencia")
107     dialogo1.setMessage("El campo Password no puede estar vacio ")
108     dialogo1.setCancelable(false)
109     dialogo1.setPositiveButton(
110         text: "Aceptar"
111     ) { dialogo1, id -> aceptar() }
112     dialogo1.setNegativeButton(
113         text: "Cancelar"
114     ) { dialogo1, id -> cancelar() }
115     dialogo1.show()
116     false
117 } else if (!passwordRegex.matcher(password).matches()) {
118
```

The screenshot shows the Android Studio IDE with the LoginActivity.kt file open. The code implements a password validation function that checks for empty fields and specific character requirements (at least one digit, one lowercase letter, one uppercase letter, one special character, and a minimum length of 4 characters). If the password is empty, an AlertDialog is shown with the message "El campo Password no puede estar vacio" (The password field cannot be empty) and buttons for "Aceptar" (Accept) and "Cancelar" (Cancel). The IDE interface includes a Project view on the left, a toolbar at the top, and a status bar at the bottom.

Una vez ingresado correctamente el correo y la contraseña, el programa mandará a una nueva pantalla que le dará un mensaje de bienvenida confirmando que ha iniciado sesión, para ello se creó una nueva actividad la cual lleva el nombre de “WelcomeActivity”, ahí realice el código para mandar un mensaje de bienvenida, esta será llamada desde “LoginActivity” si se ingresó correctamente los datos solicitados y se presionó el botón ingresar.



```
1 package com.example.banco_movil
2
3 import ...
4
5
6
7
8
9 class WelcomeActivity2 : AppCompatActivity() {
10     override fun onCreate(savedInstanceState: Bundle?) {
11         super.onCreate(savedInstanceState)
12         setContentView(R.layout.activity_welcome2)
13
14         val dialogo1 = AlertDialog.Builder( context: this)
15         fun aceptar() {
16             val t = Toast.makeText( context: this, text: "Bienvenido.", Toast.LENGTH_SHORT)
17             t.show()
18         }
19         fun cancelar() {
20             finish()
21         }
22         dialogo1.setTitle("Advertencia")
23         dialogo1.setMessage("¡Bienvenido! ")
24         dialogo1.setCancelable(false)
25         dialogo1.setPositiveButton(
26             text: "Aceptar"
27         ) { dialogo1, id -> aceptar() }
28         dialogo1.setNegativeButton(
29             text: "Cancelar"
30         ) { dialogo1, id -> cancelar() }
31         dialogo1.show()
32
33
34         /* var mostrarSaludo
35         val function =mostrarSaludo(){
36             var texto = "";
37             val new = null
38             var ahora=new Date(11);
39             var hora=ahora.getHours();
```

## Codificación

### Login

```
package com.example.banco_movil

import android.content.Intent
import android.os.Bundle
import android.widget.Toast
import androidx.appcompat.app.AlertDialog
import androidx.appcompat.app.AppCompatActivity
import androidx.core.util.PatternsCompat
import com.example.banco_movil.databinding.ActivityLoginBinding
import java.util.regex.Pattern

class LoginActivity : AppCompatActivity() {
    private lateinit var binding: ActivityLoginBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityLoginBinding.inflate(layoutInflater)
        setContentView(binding.root)

        binding.btnIngresar.setOnClickListener {validate() }
    }
    private fun validate(){
        val result = arrayOf (validateEmail(), validatePassword())
        if (false in result){
            return
        }
        val intent = Intent(this, WelcomeActivity2::class.java)
        startActivity(intent)
        finish()
        Toast.makeText(this, "Succes",Toast.LENGTH_LONG) .show()
    }

    private fun validateEmail(): Boolean {
        val email = binding.textFieldEmail.editText?.text.toString()
        return if (email.isEmpty()){

            val dialogo1 = AlertDialog.Builder(this)
            fun aceptar() {
                val t = Toast.makeText(this, "Ingrese correo .", Toast.LENGTH_SHORT)
```

```

        t.show()
    }
    fun cancelar() {

    }
    dialog1.setTitle("                    Advertencia")
    dialog1.setMessage(" El campo Email no puede estar vacio ")
    dialog1.setCancelable(false)
    dialog1.setPositiveButton(
        "Aceptar"
    ) { dialog1, id -> aceptar() }
    dialog1.setNegativeButton(
        "Cancelar"
    ) { dialog1, id -> cancelar() }
    dialog1.show()

false
} else if (!PatternsCompat.EMAIL_ADDRESS.matcher(email).matches()) {

    val dialog1 = AlertDialog.Builder(this)
    fun aceptar() {
        val t = Toast.makeText(this, "Ingrese correo valido.", Toast.LENGTH_SHORT)
        t.show()
    }
    fun cancelar() {

    }
    dialog1.setTitle("                    Advertencia")
    dialog1.setMessage(" Correo no valido, debe tener estructura de correo electronico ")
    dialog1.setCancelable(false)
    dialog1.setPositiveButton(
        "Aceptar"
    ) { dialog1, id -> aceptar() }
    dialog1.setNegativeButton(
        "Cancelar"
    ) { dialog1, id -> cancelar() }
    dialog1.show()
    false
} else {
    binding.textFieldEmail.error = null
true
}
}
}

```

```

private fun validatePassword() :Boolean{
    val password = binding.textFieldPassword.editText?.text.toString()
    val passwordRegex = Pattern.compile(
        "^" +
            "(?=.*[0-9])" +
            "(?=.*[a-z])" +
            "(?=.*[A-Z])" +
            "(?=.*[!@#$%^&*])" +
            "(?=.*\\S+$)" +
            ".{4,}" +
            "$"
    )
}
return if (password.isEmpty()){

    val dialogo1 = AlertDialog.Builder(this)
    fun aceptar() {
        val t = Toast.makeText(this, "Ingresa password.", Toast.LENGTH_SHORT)
        t.show()
    }
    fun cancelar() {

    }
    dialogo1.setTitle("Advertencia")
    dialogo1.setMessage(" El campo Password no puede estar vacio ")
    dialogo1.setCancelable(false)
    dialogo1.setPositiveButton(
        "Aceptar"
    ) { dialogo1, id -> aceptar() }
    dialogo1.setNegativeButton(
        "Cancelar"
    ) { dialogo1, id -> cancelar() }
    dialogo1.show()
    false
} else if (!passwordRegex.matcher(password).matches()) {

    val dialogo1 = AlertDialog.Builder(this)
    fun aceptar() {
        val t = Toast.makeText(this, "Ingresa correo valido.", Toast.LENGTH_SHORT)
        t.show()
    }
    fun cancelar() {

```

```

    }
    dialogo1.setTitle("                    Advertencia")
    dialogo1.setMessage(" La contraseña debe contener al menos una mayuscula, una minuscula, un numero, un
caracter especial y tener al menos 4 caracteres ")
    dialogo1.setCancelable(false)
    dialogo1.setPositiveButton(
        "Aceptar"
    ) { dialogo1, id -> aceptar() }
    dialogo1.setNegativeButton(
        "Cancelar"
    ) { dialogo1, id -> cancelar() }
    dialogo1.show()
    false
} else { binding.textFieldPassword.error = null
    true
}
}
}
}

```

### Pantalla de bienvenida

```

package com.example.banco_movil

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Toast
import androidx.appcompat.app.AlertDialog
import com.google.type.Date

class WelcomeActivity2 : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_welcome2)

        val dialogo1 = AlertDialog.Builder(this)
        fun aceptar() {
            val t = Toast.makeText(this, "Bienvenido.", Toast.LENGTH_SHORT)
            t.show()
        }
        fun cancelar() {
            finish()
        }
        dialogo1.setTitle("                    Ingresó")
        dialogo1.setMessage("                    ¡Bienvenido! ")
    }
}

```

```
dialogo1.setCancelable(false)
dialogo1.setPositiveButton(
    "Aceptar"
) { dialogo1, id -> aceptar() }
dialogo1.setNegativeButton(
    "Cancelar"
) { dialogo1, id -> cancelar() }
dialogo1.show()
```

## Diseño de pantalla de Login

```
<?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:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".LoginActivity"

    android:padding="12dp"
    android:orientation="vertical">

    <com.google.android.material.textfield.TextInputLayout
        android:id="@+id/textFieldEmail"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"
        app:startIconDrawable="@drawable/ic_email"
        app:helperText="*Requerido"
        app:counterEnabled="true"
        android:layout_marginTop="150dp"
        app:counterMaxLength="50"
        android:hint="Email">
        <com.google.android.material.textfield.TextInputEditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:inputType="textEmailAddress"/>
    </com.google.android.material.textfield.TextInputLayout>

    <com.google.android.material.textfield.TextInputLayout
```

```
    android:id="@+id/textFieldPassword"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"
    app:startIconDrawable="@drawable/ic_lock"
    app:helperText="*Requerido"
    app:endIconMode="password_toggle"
    app:errorEnabled="true"
    app:errorIconDrawable="@null"
    android:layout_marginTop="32dp"
    android:hint="Password">
    <com.google.android.material.textfield.TextInputEditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="textPassword"/>
```

```
</com.google.android.material.textfield.TextInputLayout>
```

```
<Button
```

```
    android:id="@+id/btnIngresar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"
    android:text="INGRESAR"/>
```

```
<Button
```

```
    android:id="@+id/btnRegistrar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"
    android:text="Registrar"/>
```

```
<EditText
```

```
    android:id="@+id/editTextTime"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:ems="10"
    android:inputType="time"
```

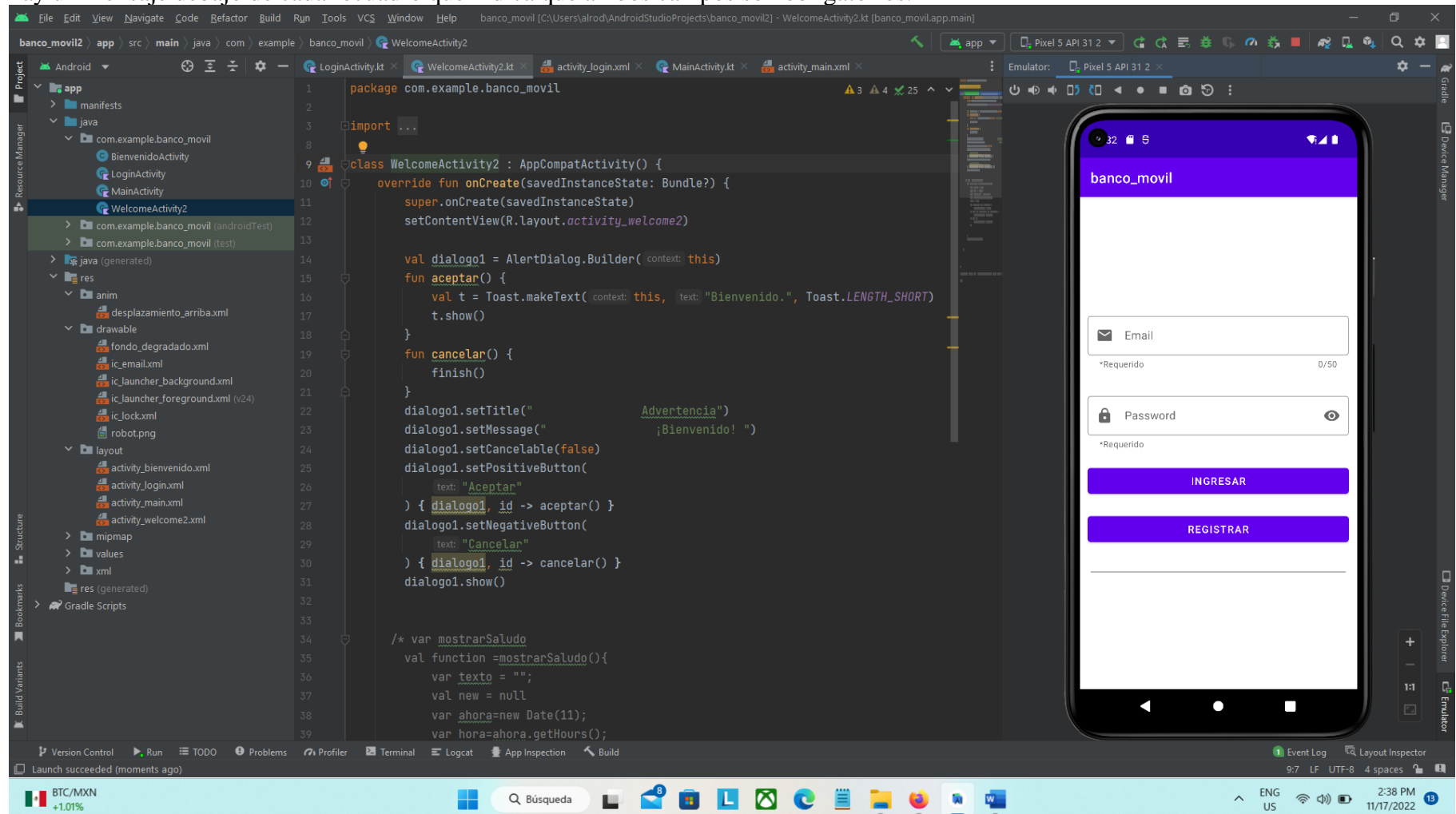
```
/>
```

```
</LinearLayout>
```

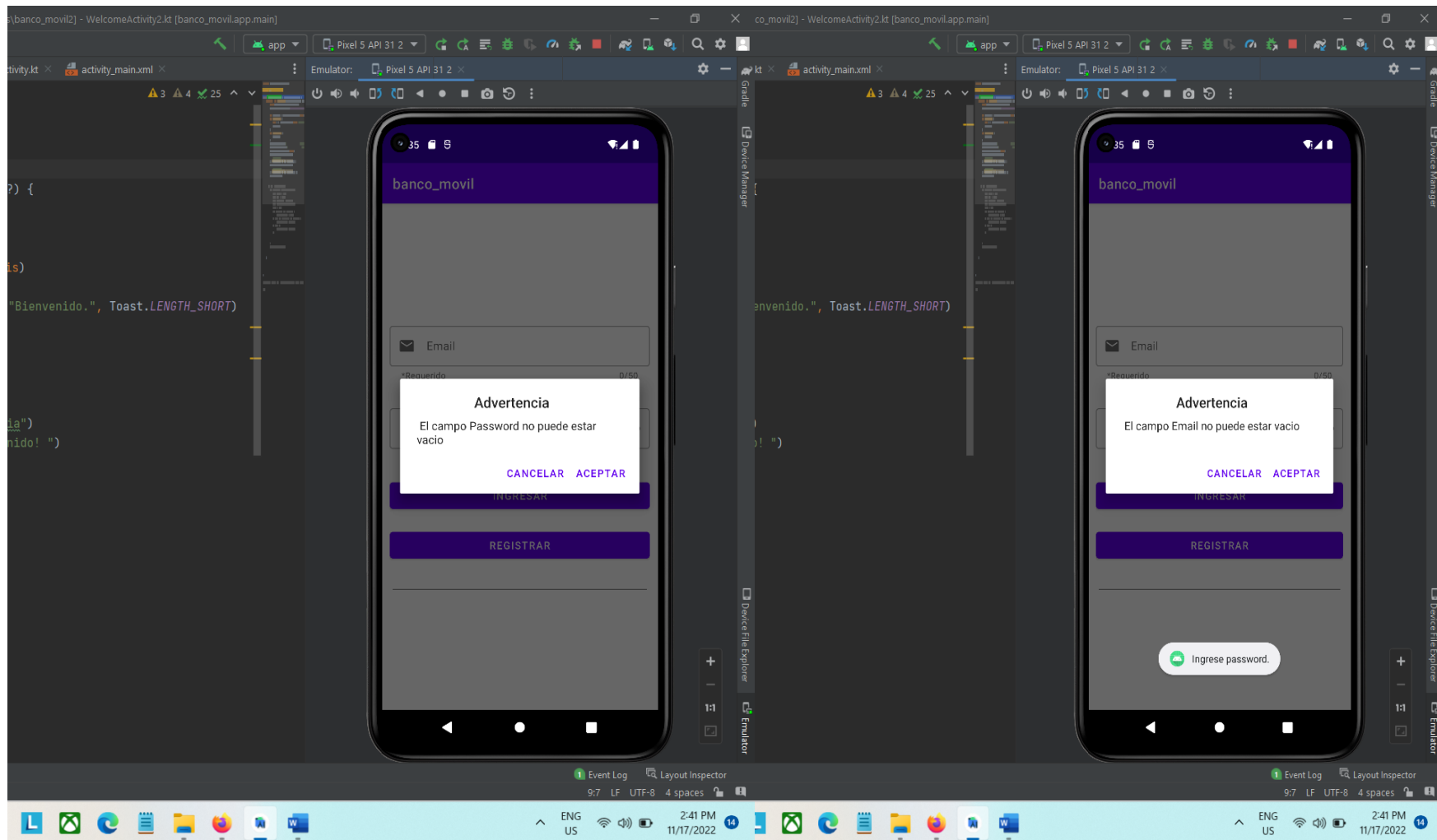


## Prueba de aplicación

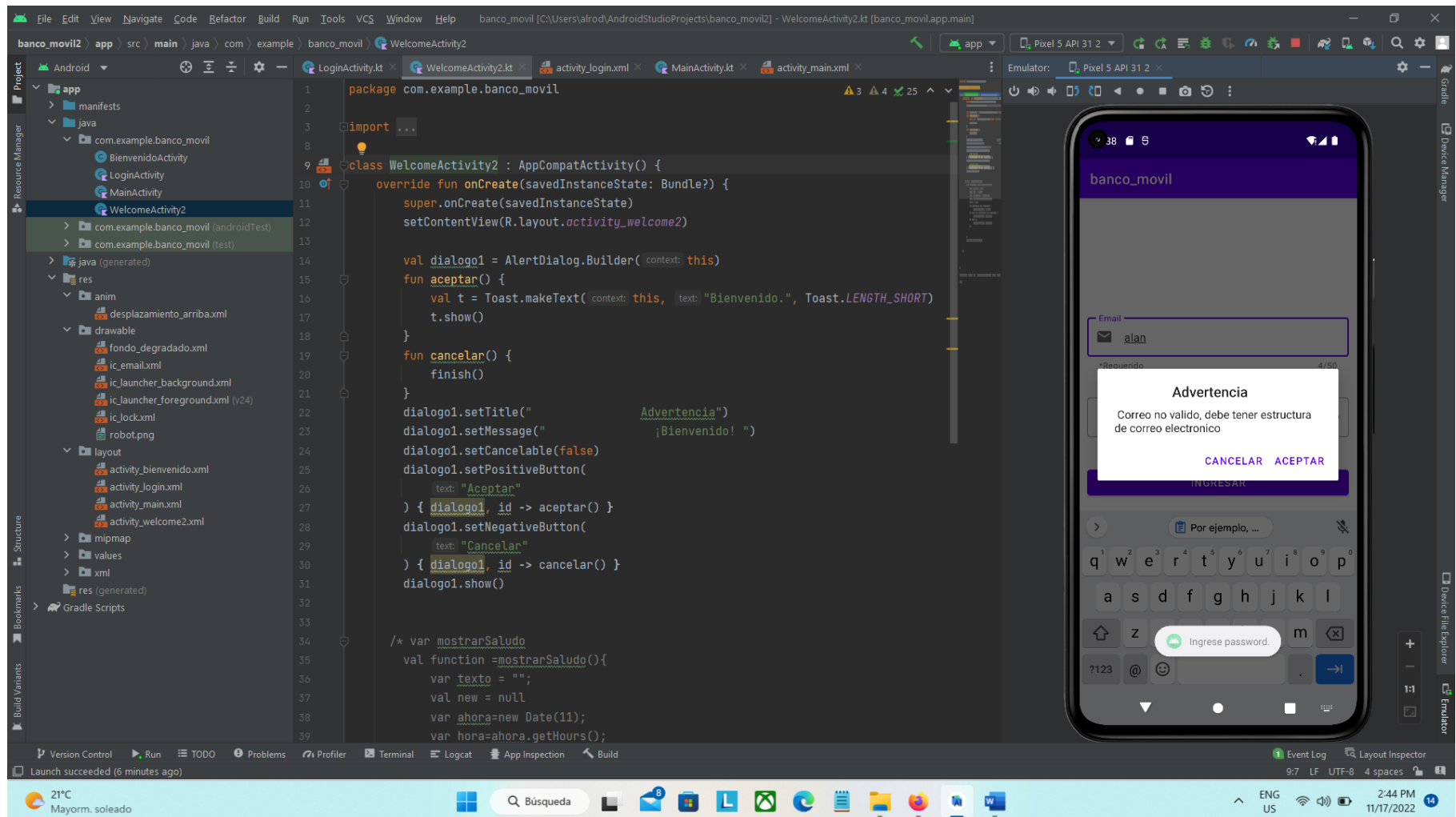
Una vez lanzado el programa, pasa el Splash Screen, y nos abre la pantalla de Login donde hay que ingresar los datos que se piden, incluso hay un mensaje debajo de cada recuadro que indica que ambos campos son obligatorios.



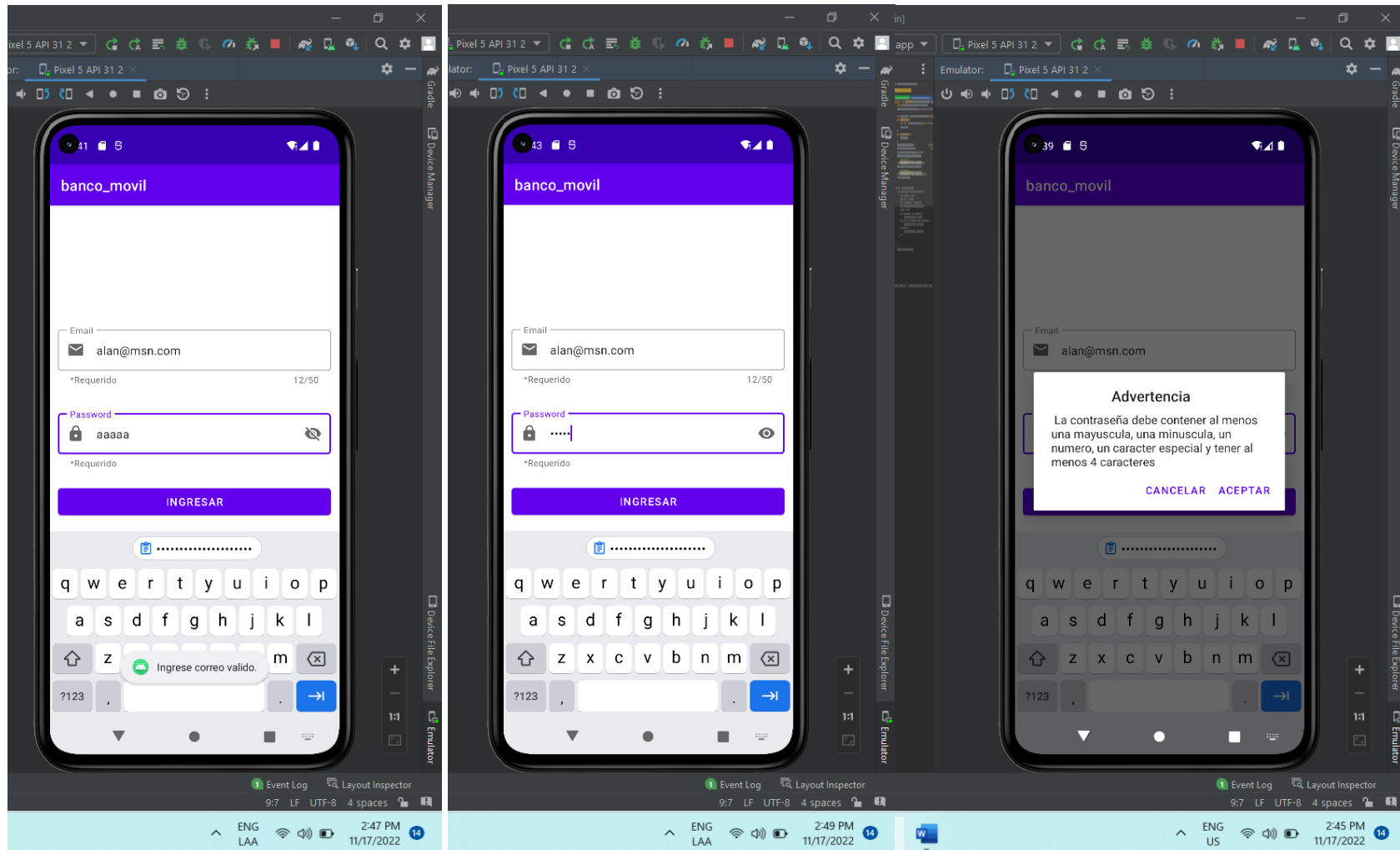
Cuando se hace clic en el botón ingresar sin haber escrito en los campos arroja el mensaje que no deben estar vacíos.



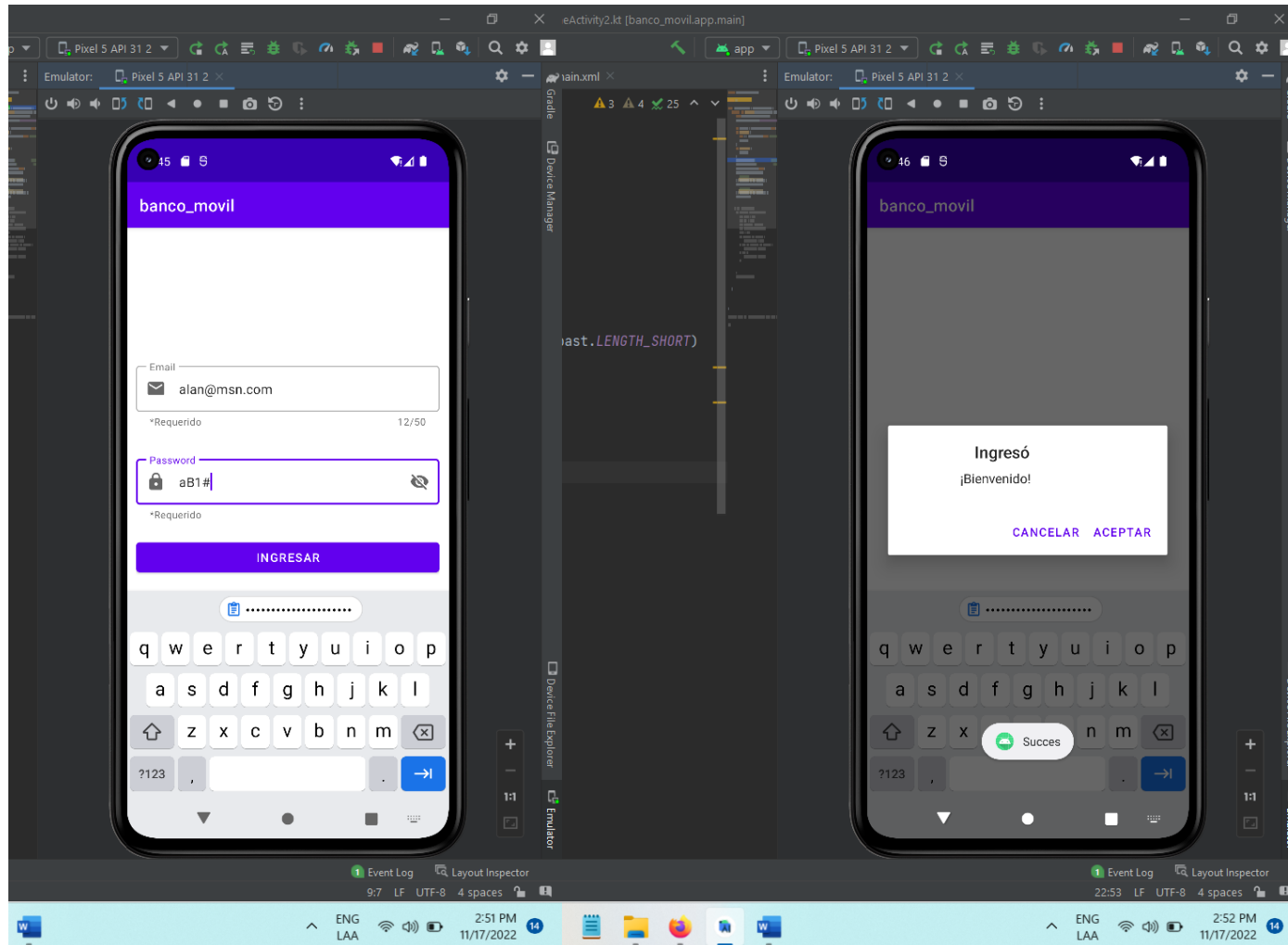
Si se escribe incorrectamente en el campo de correo, por ejemplo, solo letras, envía el mensaje que se requiere escribir un correo que tenga la estructura solicitada de correo electrónico.



De igual manera si se escribe una contraseña muy débil, nos mandara el mensaje que nos indica que debe contener la contraseña, en este caso pide una letra minúscula, una mayúscula, un número, un carácter especial y yo agregue que debe tener al menos cuatro caracteres para poder aceptar la contraseña.



Una vez que se cumple con los requisitos solicitados, el programa nos manda a la nueva pantalla que nos dará el mensaje de bienvenida y nos dirá que ingresamos a nuestra sesión.





## Conclusión

El formulario de Login da acceso a un sitio o aplicación y por ende a los datos personales, como es el caso de una aplicación, sobre todo en temas como instituciones bancarias. Esta pantalla cumple con una tarea fundamental de seguridad; pero muchas veces se omite evaluar si los procedimientos de nombre de usuario (email), claves (contraseñas) y autenticación cumplen con las recomendaciones de seguridad. Para mejorar la seguridad se recomienda en el password, una combinación entre letras mayúsculas, minúsculas, números y algunos caracteres especiales como (\$.#\* %) con una longitud entre 6 y 8 caracteres y para el email debe utilizar la dirección electrónica valida del usuario ya que es única y lleva una estructura. Esa es la función de seguridad que solicita esta actividad , aprendí a desarrollar una pantalla de autenticación que para otorgar acceso al siguiente nivel del sitio, el programa verifica que el usuario tenga un correo permitido, y la contraseña cumpla con los requisitos mencionados de lo contrario marca error, el programa de Android Studio me facilito mucho la creación de esta pantalla, ya que con la ayuda de las características del programa, como el emulador pude comprobar en tiempo real el funcionamiento de mi aplicación y revisar errores en la programación, esta actividad me dejo mucho aprendizaje ya que tuve que buscar como hacer la programación en kotlin y extrapolar eso aprendido a la actividad que me solicitan.

## ENLACE DEL PROYECTO

[https://drive.google.com/file/d/1RCyqJdnGRDg-9ZO\\_vQf0ZnESZvxmlQtJ/view?usp=share\\_link](https://drive.google.com/file/d/1RCyqJdnGRDg-9ZO_vQf0ZnESZvxmlQtJ/view?usp=share_link)

## **Bibliografía**

Developers, Introducción a Android Studio, sitio web  
<https://developer.android.com/studio/intro?hl=es-419>

Marlon Mederos Corrales, Primer acercamiento al Android Studio, Universidad tecnológica de la Habana, Monografía CUJAE 2021,  
[https://www.researchgate.net/publication/354474559\\_Primer\\_acercamiento\\_al\\_Android\\_Studio](https://www.researchgate.net/publication/354474559_Primer_acercamiento_al_Android_Studio)

Salvador Gómez Oliver, Manual Programación Android, Versión online 3.0, junio 2013,  
<https://aluzardo.github.io/trabajo-fin-de-grado/Tutoriales/Manual%20Programacion%20Android.pdf>

Yendry Lopez Gómez, ¿Qué es un Emulador?, <https://lovtechnology.com/que-es-un-emulador/>