Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Actividad #3

Verificación por correo electrónico

Asiqnatura: Laboratorio de Ingeniería de Dispositivos Móviles.

Grupo: 309

Hora N5 -N6 Miércoles

Docente: Héctor Hugo Flores Moreno

Equipo #2		
Matrícula	**Nombre**	**Carrera**
1924910	Ranfery Josua Peregrina Morales	ITS
1958187	Diego Pérez Tabullo	ITS

Síntesis general del código

Para esta actividad se nos pidió hacer una autenticación de la cuenta a través del correo electrónico, y no lo interpretamos como una validación de coincidencia en la cadena de texto registrada en el campo del coreo electrónico, si no como el uso del servicio de mensajería para comunicar una forma privada de confirmar el registro del usuario.

Para esta práctica nuestro objetivo era asegurar que el usuario verifique su dirección de correo electrónico después de registrarse, permitiendo el inicio de sesión solo si el correo ha sido confirmado. Esto mejora la seguridad y garantiza que el usuario tenga acceso válido al correo proporcionado.

No fue necesario crear una Activity nueva, sólo modificar la actividad de registro para que envié un correo de verificación, y la de Login, para que al hacer la validación de inicio de sesión también se valide además de la contraseña, que el correo que se esté ingresando como credencial en el campo del correo, sea el e un correo verificado.

Aunque pensamos que necesitaríamos pagar un servicio de mensajería para hacer la gestión del correo, parece ser que FireBase ya tiene una API gratuita. Aunque no es particularmente eficiente, para los fines de la práctica fue sufciente.

Esta API manda un correo desde una dirección que tuvimos que credencializar previamente con el mensaje o archivo de texto que nosotros especifiquemos. En él, un botón a un enlace PhP que almacena la actividad del usuario. En este caso, simplemente el haberse usado ya registra el correo como “Verificado”:

Al final hemos podido hacer énfasis en
no fue necesario incluir más validaciones de seguridad porque la librería de Firebase Auth ya incluye métodos para verificación de correo, pero es importante integrarlos correctamente en el flujo de la aplicación.

A continuación, mostraremos la modificación en el código de las Activity anteriores (pues no creamos ninguna nueva)

Explicación de la modificación en RegisterActivity.kt

Sólo a modo de índice, escribiremos estas líneas de código que ya estaban escritas desde la creación de esta Activity.

```
private fun guardarDatosUsuario(userId: String, nombre: String) {
    val userData = hashMapOf(
        "nombre" to nombre,
        "fechaRegistro" to System.currentTimeMillis()
    )
```

Que de los datos recibidos al escribir, además del nombre, registran la fecha de registro de ese usuario.

```
database.reference.child("usuarios").child(userId).setValue(userData)
    .addOnSuccessListener {

        auth.currentUser?.sendEmailVerification()
            ?.addOnSuccessListener {
                mostrarMensaje("Se ha enviado un correo de
verificación.")
                finish()            }
```

Después de guardar los datos del usuario en Firebase Realtime Database, enviamos un correo de verificación usando sendEmailVerification().

Si el envío es exitoso, cerramos la actividad y notificamos al usuario.

Si falla, obtendremos un error, y pasaremos ese error como parámetro a la función de “motrarMensaje”,que se encargará de mostrar en un Toast el error recibido.

Explicación de la modificación en LoginActivity.kt

```
private fun validarCredenciales() {
    val correo = etCorreoLogin.text?.toString()?.trim()
    val contrasena = etContrasenaLogin.text?.toString()?.trim()

    if (correo.isNullOrEmpty() || contrasena.isNullOrEmpty()) {
        mostrarMensaje("Todos los campos son obligatorios")
```

Estas líneas pertenecen a la función que vamos a modificar “**validarCredenciales()**”. Que se encargaba de darles formato de texto uniforme a las cadenas leídas y guardarlas

```
if (correo.isNullOrEmpty() || contrasena.isNullOrEmpty()) {
    mostrarMensaje("Todos los campos son obligatorios")
    return
}
```

Verifica si los campos de correo o contraseña están vacíos.

Si alguno está vacío, muestra un mensaje al usuario y detiene la ejecución de la función con return.

```
mostrarProgreso(true)
```

Deshabilita el botón de inicio de sesión y muestra una barra de progreso (o cualquier indicador visual) para informar al usuario que se está realizando una operación en segundo plano.

```
val user = auth.currentUser


if (user?.isEmailVerified == true) {
    mostrarMensaje("Sesión iniciada correctamente")
    startActivity(Intent(this, HomeActivity::class.java))    finish()
} else {
    mostrarMensaje("Por favor, verifica tu correo electrónico.")
    auth.signOut()
```

Si el inicio de sesión es exitoso, verifica si el correo electrónico del usuario está verificado usando user?.isEmailVerified.
Si el correo está verificado, redirige al usuario a HomeActivity.
Si el correo no está verificado, muestra un mensaje y cierra la sesión automáticamente con auth.signOut().

```
    if (intentosFallidos >= 3) {
        bloquearBotonTemporalmente()
    } else {
        val intentosRestantes = 3 - intentosFallidos
        mostrarMensaje("Contraseña incorrecta. Intentos restantes:
$intentosRestantes")
        tvIntentosRestantes.text = "Intentos restantes:
$intentosRestantes"
    }
}
```

Incrementa el contador de intentos fallidos.
Si el usuario supera los 3 intentos fallidos, bloquea el botón de inicio de sesión temporalmente.
Si no, muestra un mensaje con los intentos restantes y actualiza el TextView.

```
<com.google.android.material.button.MaterialButton
    android:id="@+id/btnReenviarCorreo"
```

Usamos `<com.google.android.material.button.MaterialButton>` en lugar del tradicional `<Button>`, ya que nos proporciona un diseño más moderno y compatible con las directrices de Material Design.

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
```

Configuramos el ancho y el alto del botón como wrap_content, lo que significa que el botón se ajustará automáticamente al tamaño del texto que contiene.

```
android:text="Reenviar correo de verificación"
```

Establecemos el texto visible en el botón para indicar claramente su función al usuario.

```
app:layout constraintTop toBottomOf="@id/tvIntentosRestantes"
```

Esto significa que el botón estará **debajo** del `TextView` identificado como `tvIntentosRestantes`, el cual probablemente muestra información sobre los intentos disponibles para reenviar el correo.

```
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintEnd_toEndOf="parent"
```

Con estas líneas centramos el botón horizontalmente dentro del `ConstraintLayout`, asegurando que su posición sea equidistante de los lados de la pantalla.

```
android:layout_marginTop="16dp" />
```

Añadimos un margen superior de **16dp** para evitar que el botón quede demasiado cerca del elemento superior (tvIntentosRestantes).

Código completo de LoginActivity.kt

```
package com.example.proyecto_equipo_2

import android.content.Intent
import android.os.Bundle
import android.os.Handler
import android.os.Looper
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import com.google.android.material.textfield.TextInputEditText
import com.google.android.material.button.MaterialButton
import com.google.firebase.auth.FirebaseAuth

class LoginActivity : AppCompatActivity() {

    private lateinit var etCorreoLogin: TextInputEditText
    private lateinit var etContrasenaLogin: TextInputEditText
    private lateinit var btnIniciarSesion: MaterialButton

    private var intentosFallidos = 0 // Contador de intentos fallidos
    private val handler = Handler(Looper.getMainLooper())
    private lateinit var tvIntentosRestantes: android.widget.TextView //
TextView para mostrar intentos

    private lateinit var auth: FirebaseAuth

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_login)

        // Inicializar vistas
        etCorreoLogin = findViewById(R.id.etCorreoLogin)
        etContrasenaLogin = findViewById(R.id.etContrasenaLogin)
        btnIniciarSesion = findViewById(R.id.btnIniciarSesion)
        tvIntentosRestantes = findViewById(R.id.tvIntentosRestantes)

        // Inicializar Firebase Auth
        auth = FirebaseAuth.getInstance()

        // Configurar clic del botón "Iniciar Sesión"
        btnIniciarSesion.setOnClickListener {
            validarCredenciales()
        }
    }

    private fun validarCredenciales() {
        val correo = etCorreoLogin.text?.toString()?.trim()
        val contrasena = etContrasenaLogin.text?.toString()?.trim()

        // Validar que los campos no estén vacíos
        if (correo.isNullOrEmpty() || contrasena.isNullOrEmpty()) {
            mostrarMensaje("Todos los campos son obligatorios")
            return
        }
```

```
        // Mostrar progreso (deshabilitar botón, etc.)
        mostrarProgreso(true)

        // Intentar autenticar al usuario con Firebase
        auth.signInWithEmailAndPassword(correo, contrasena)
            .addOnCompleteListener { task ->
                if (task.isSuccessful) {
                    // Obtener el usuario actual
                    val user = auth.currentUser

                    // Verificar si el correo electrónico está verificado
                    if (user?.isEmailVerified == true) {
                        mostrarMensaje("Sesión iniciada correctamente")
                        startActivity(Intent(this,
HomeActivity::class.java)) // Redirigir al Home
                        finish() // Cerrar la actividad de inicio de
sesión
                    } else {
                        mostrarMensaje("Por favor, verifica tu correo
electrónico.")
                        auth.signOut() // Cerrar la sesión
automáticamente
                    }
                } else {
                    // Manejo de intentos fallidos
                    intentosFallidos++
                    if (intentosFallidos >= 3) {
                        bloquearBotonTemporalmente()
                    } else {
                        val intentosRestantes = 3 - intentosFallidos
                        mostrarMensaje("Contraseña incorrecta. Intentos
restantes: $intentosRestantes")
                        tvIntentosRestantes.text = "Intentos restantes:
$intentosRestantes"
                    }
                }

                // Ocultar progreso (habilitar botón, etc.)
                mostrarProgreso(false)
            }
    }

    private fun bloquearBotonTemporalmente() {
        btnIniciarSesion.isEnabled = false
        mostrarMensaje("Cuenta bloqueada por 20 segundos")
        tvIntentosRestantes.text = "Cuenta bloqueada temporalmente"

        handler.postDelayed({
            btnIniciarSesion.isEnabled = true
            intentosFallidos = 0
            tvIntentosRestantes.text = "Intentos restantes: 3"
        }, 20000) // 20 segundos
    }

    private fun mostrarProgreso(mostrar: Boolean) {
        btnIniciarSesion.isEnabled = !mostrar
    }
```

```
    private fun mostrarMensaje(mensaje: String) {
        Toast.makeText(this, mensaje, Toast.LENGTH_SHORT).show()
    }
}
```

Código completo de RegsterAcivity.kt

```
package com.example.proyecto_equipo_2

import android.os.Bundle
import android.view.View
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import com.google.android.material.textfield.TextInputEditText
import com.google.android.material.button.MaterialButton
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.database.FirebaseDatabase

class RegisterActivity : AppCompatActivity() {

    private lateinit var etNombreCompleto: TextInputEditText
    private lateinit var etCorreo: TextInputEditText
    private lateinit var etContrasena: TextInputEditText
    private lateinit var btnRegistrar: MaterialButton
    private lateinit var progressBar: View

    private lateinit var auth: FirebaseAuth
    private lateinit var database: FirebaseDatabase

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_register)

        // Inicializar vistas
        etNombreCompleto = findViewById(R.id.etNombreCompleto)
        etCorreo = findViewById(R.id.etCorreo)
        etContrasena = findViewById(R.id.etContrasena)
        btnRegistrar = findViewById(R.id.btnRegistrar)
        progressBar = findViewById(R.id.progressBar)

        // Inicializar Firebase
        auth = FirebaseAuth.getInstance()
        database = FirebaseDatabase.getInstance()

        btnRegistrar.setOnClickListener {
            registrarUsuario()
        }
    }

    private fun registrarUsuario() {
        val nombre = etNombreCompleto.text?.toString()?.trim() ?: ""
        val correo = etCorreo.text?.toString()?.trim() ?: ""
        val contrasena = etContrasena.text?.toString()?.trim() ?: ""

        // Validaciones
        if (nombre.isEmpty() || correo.isEmpty() || contrasena.isEmpty())
{
            mostrarMensaje("Todos los campos son obligatorios")
            return
        }
```

```
        if (contrasena.length < 6) {
            mostrarMensaje("La contraseña debe tener al menos 6
caracteres")
            return
        }

        mostrarProgreso(true)

        // Registrar en Firebase Auth
        auth.createUserWithEmailAndPassword(correo, contrasena)
            .addOnCompleteListener { task ->
                if (task.isSuccessful) {
                    val userId = auth.currentUser?.uid
                    if (userId != null) {
                        guardarDatosUsuario(userId, nombre)
                    } else {
                        mostrarProgreso(false)
                        mostrarMensaje("Error al obtener el ID de
usuario")
                    }
                } else {
                    mostrarProgreso(false)
                    mostrarMensaje(obtenerMensajeError(task.exception))
                }
            }
    }

    private fun guardarDatosUsuario(userId: String, nombre: String) {
        val userData = hashMapOf(
            "nombre" to nombre,
            "fechaRegistro" to System.currentTimeMillis()
        )


database.reference.child("usuarios").child(userId).setValue(userData)
                .addOnSuccessListener {
                    // Enviar correo de verificación
                    auth.currentUser?.sendEmailVerification()
                        ?.addOnSuccessListener {
                            mostrarMensaje("Se ha enviado un correo de
verificación.")
                            finish() // Cierra la actividad de registro
                        }
                    ?.addOnFailureListener { e ->
                        mostrarMensaje("Error al enviar correo:
${e.message}")
                    }
            }
            .addOnFailureListener { e ->
                mostrarMensaje("Error al guardar datos: ${e.message}")
            }
    }


database.reference.child("usuarios").child(userId).setValue(userData)
            .addOnSuccessListener {
                mostrarProgreso(false)
```

```
                mostrarMensaje("Usuario registrado exitosamente")
                finish()
            }
            .addOnFailureListener { e ->
                mostrarProgreso(false)
                mostrarMensaje("Error al guardar datos: ${e.message}")
            }
    }

    private fun obtenerMensajeError(exception: Exception?): String {
        return when (exception?.message) {
            "The email address is badly formatted." -> "El formato del
correo electrónico no es válido"
            "The email address is already in use by another account." ->
"Este correo ya está registrado"
            "A network error (such as timeout, interrupted connection or
unreachable host) has occurred." ->
                "Error de conexión. Verifica tu internet"
            else -> exception?.message ?: "Error desconocido"
        }
    }

    private fun mostrarProgreso(mostrar: Boolean) {
        progressBar.visibility = if (mostrar) View.VISIBLE else View.GONE
        btnRegistrar.isEnabled = !mostrar
    }

    private fun mostrarMensaje(mensaje: String) {
        Toast.makeText(this, mensaje, Toast.LENGTH_SHORT).show()
    }
}
```

Código completo de activity_login.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    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"
    android:padding="16dp">

    <!-- Campo de entrada para el correo -->
    <com.google.android.material.textfield.TextInputLayout
        android:id="@+id/tilCorreoLogin"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Correo electrónico"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent">

        <com.google.android.material.textfield.TextInputEditText
            android:id="@+id/etCorreoLogin"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:inputType="textEmailAddress" />
    </com.google.android.material.textfield.TextInputLayout>

    <!-- Campo de entrada para la contraseña -->
    <com.google.android.material.textfield.TextInputLayout
        android:id="@+id/tilContrasenaLogin"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Contraseña"
        app:passwordToggleEnabled="true"
        app:layout_constraintTop_toBottomOf="@+id/tilCorreoLogin"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        android:layout_marginTop="16dp">

        <com.google.android.material.textfield.TextInputEditText
            android:id="@+id/etContrasenaLogin"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:inputType="textPassword" />
    </com.google.android.material.textfield.TextInputLayout>

    <!-- Botón para iniciar sesión -->
    <com.google.android.material.button.MaterialButton
        android:id="@+id/btnIniciarSesion"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Iniciar Sesión"
        app:layout_constraintTop_toBottomOf="@+id/tilContrasenaLogin"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        android:layout_marginTop="24dp" />
```

```
    <!-- Botón para reenviar el correo -->
    <com.google.android.material.button.MaterialButton
        android:id="@+id/btnReenviarCorreo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Reenviar correo de verificación"
        app:layout_constraintTop_toBottomOf="@id/tvIntentosRestantes"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        android:layout_marginTop="16dp" />

    <!-- Texto para mostrar intentos restantes -->
    <TextView
        android:id="@+id/tvIntentosRestantes"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Intentos restantes: 3"
        android:layout_marginTop="16dp"
        app:layout_constraintTop_toBottomOf="@+id/btnIniciarSesion"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```