Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Actividad #2

Validación de contraseña

Asignatura 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 de lo que hace el código

Para la **Actividad 2** se nos pidió un programa que verificara la contraseña que el usuario eligiera para registrarse. Este registro se realizó anteriormente en la **Actividad 1**. Donde además de crear la interfaz para el registro del usuario (con su contraseña), también se hizo la base de datos en *firebase* que almacenara dicha contraseña.

En este programa lo que haremos será comprobar si en una nueva Activity llamada "LoginActivity", la contraseña que ingrese el usuario, corresponde con la registrada con el correo que también se está ingresando.

Para esta actividad fueron necesarias crear 2 Activity extra. **HomeActivity**, **LoginActivity**, además de modificar las que ya había (Y desde luego, sus documentos de estilo).

HomeActivity es la pantalla principal de la aplicación (No confundir con la pantalla inicial). Aquí es donde ocurrirá todo lo que sea que la aplicación intente hacer una vez que el usuario se haya loggeado correctamente. De momento, no se ha especificado qué pondremos una vez que el usuario pueda ingresar correctamente así que esta actividad sólo contiene *placeholder*s y textos que pueden ser remplazados más adelante. De momento sólo se demuestra cómo interactuar con Firebase Realtime Database.

LoginActivity es la actividad en la que el usuario inicia sesión con las credenciales que registró en la actividad *RegisterActivity*. Pero esta vez, toma los campos ingresados y los compara utilizando *Firebase Authentication*. Hace un par de cosas extra como validar campos vacíos y manejo de errores.

Es importante implementar una validación de contraseñas porque garantiza la seguridad de las cuentas de usuario. Sin una validación adecuada, las contraseñas podrían ser demasiado débiles, fáciles de adivinar o vulnerables a ataques como el de fuerza bruta. Una validación eficaz ayuda a asegurarte de que las contraseñas tengan un nivel mínimo de complejidad, protegiendo así la información personal de los usuarios y evitando accesos no autorizados. Además, establece buenas prácticas desde el principio, incluso si por ahora solo estamos registrando usuarios.

Explicación de código: LoginActivity.kt

Primeramente iniciamos declarando las variables que usaremos (En este caso, además de los que usamos la actividad pasada, incluiremos los de validación)

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

private var intentosFallidos = 0
private val handler = Handler(Looper.getMainLooper())
private lateinit var tvIntentosRestantes: android.widget.TextView
```

Declaramos variables para los elementos de la interfaz de usuario (etCorreoLogin, etContrasenaLogin, btnIniciarSesion, tvIntentosRestantes).
Usamos un contador (intentosFallidos) para rastrear los intentos fallidos de inicio de sesión.
Creamos un Handler para manejar operaciones en el hilo principal, como el bloqueo temporal del botón.

```
private lateinit var auth: FirebaseAuth
```

Declaramos una instancia de FirebaseAuth para interactuar con Firebase Authentication.

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

Sobrescribimos el método onCreate, que es el primer método que se ejecuta cuando se crea la actividad.
Establecemos el diseño de la interfaz de usuario usando el archivo XML activity_login.xml. (Esto ya lo habíamos hecho en la actividad anterior)

```
etCorreoLogin = findViewById(R.id.etCorreoLogin)
etContrasenaLogin = findViewById(R.id.etContrasenaLogin)
btnIniciarSesion = findViewById(R.id.btnIniciarSesion)
tvIntentosRestantes = findViewById(R.id.tvIntentosRestantes)

auth = FirebaseAuth.getInstance()
```

Vinculamos las variables declaradas anteriormente con los elementos de la interfaz de usuario definidos en el archivo XML.
E inicializamos la instancia de FirebaseAuth, que nos permitirá autenticar al usuario.

```
btnIniciarSesion.setOnClickListener {
    validarCredenciales()
}
```

Aquí configuramos un listener para el botón "Iniciar Sesión". Cuando el usuario hace clic en este botón, se ejecuta la función validarCredenciales().

```
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")
        return
    }
```

Recuperamos los valores ingresados por el usuario en los campos de correo y contraseña.
Validamos que los campos no estén vacíos. Si alguno está vacío, mostramos un mensaje de error y detenemos la ejecución de la función.

Aquí se hacen varias cosas y como todas forman parte de la misma estructura lógica, se va a malentender si separo las sentencias condicionales de sus consecuencias. Así que una vez puesto todo el bloque de código, explicaré qué hace todo este código.

```
    mostrarProgreso(true)

    auth.signInWithEmailAndPassword(correo, contrasena)
        .addOnCompleteListener { task ->
            if (task.isSuccessful) {
                mostrarMensaje("Sesión iniciada correctamente")
                startActivity(Intent(this, HomeActivity::class.java))
                finish()
            } else {
                intentosFallidos++
                if (intentosFallidos >= 3) {
                    bloquearBotonTemporalmente()
                } else {
                    val intentosRestantes = 3 - intentosFallidos
                    mostrarMensaje("Contraseña incorrecta. Intentos
restantes: $intentosRestantes")
                    tvIntentosRestantes.text = "Intentos restantes:
$intentosRestantes"
                }
            }
            mostrarProgreso(false)
        }
}
```

Mostrar progreso: Deshabilitamos el botón "Iniciar Sesión" para evitar múltiples clics mientras se realiza la autenticación.

Autenticación con Firebase: Usamos `signInWithEmailAndPassword` para validar las credenciales del usuario.

Manejo del resultado:
Si el inicio de sesión es exitoso, mostramos un mensaje de éxito, redirigimos al usuario a `HomeActivity` y cerramos `LoginActivity`.

Si el inicio de sesión falla, incrementamos el contador de intentos fallidos.

> Si hay 3 o más intentos fallidos, bloqueamos el botón temporalmente.
>
> Si hay menos de 3 intentos fallidos, mostramos un mensaje con los intentos restantes y actualizamos el `TextView`.

Ocultar progreso: Habilitamos el botón "Iniciar Sesión" nuevamente.

```
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
}
```

Deshabilitamos el botón "Iniciar Sesión" y mostramos un mensaje de bloqueo.
Usamos un Handler para habilitar el botón después de 20 segundos.
Reiniciamos el contador de intentos fallidos y actualizamos el TextView.

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

private fun mostrarMensaje(mensaje: String) {
    Toast.makeText(this, mensaje, Toast.LENGTH_SHORT).show()
}
 = "Intentos restantes: 3"
    }, 20000) }
```

mostrarProgreso: Controla la habilitación/deshabilitación del botón "Iniciar Sesión".
mostrarMensaje: Muestra un mensaje temporal (Toast) en la pantalla.

Explicación del código: HomeActivity.kt

```
private lateinit var database: FirebaseDatabase
private lateinit var messageRef: DatabaseReference
```

Declaramos dos variables:

- database: Una instancia de FirebaseDatabase, que nos permitirá interactuar con la base de datos en tiempo real de Firebase.
- messageRef: Una referencia a un nodo específico en la base de datos. En este caso, usaremos un nodo llamado "message".

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

Sobrescribimos el método onCreate, que es el primer método que se ejecuta cuando se crea la actividad. También establecemos el diseño de la interfaz de usuario usando el archivo XML activity_home.xml.

Inicializamos la instancia de FirebaseDatabase. Para que nos podamos conectar a la

```
// Inicializamos la instancia de la base de datos
database = FirebaseDatabase.getInstance()
```

```
messageRef = database.getReference("message")
```

base de datos en tiempo real de Firebase.

Creamos una referencia a un nodo específico en la base de datos llamado "message". Este nodo puede contener cualquier tipo de dato (cadena, número, objeto, etc.).

Si el nodo no existe, Firebase lo creará automáticamente.

```
messageRef.setValue("Hello, Firebase!") { error, _ ->
    if (error != null) {
        Toast.makeText(this, "Error al escribir en Firebase:
${error.message}", Toast.LENGTH_SHORT).show()
    } else {
        Toast.makeText(this, "Mensaje guardado correctamente",
Toast.LENGTH_SHORT).show()
    }
}
```

Escribimos el valor "Hello, Firebase!" en el nodo "message" de la base de datos.
Usamos un callback ({ error, _ -> ... }) para manejar el resultado de la operación:
Si hay un error, mostramos un mensaje de error usando Toast.
Si la operación es exitosa, mostramos un mensaje de éxito.

```
messageRef.setValue("Hello, Firebase!") { error, _ ->
    if (error != null) {
        Toast.makeText(this, "Error al escribir en Firebase:
${error.message}", Toast.LENGTH_SHORT).show()
    } else {
        Toast.makeText(this, "Mensaje guardado correctamente",
Toast.LENGTH_SHORT).show()
    }
}
```

Leemos el valor almacenado en el nodo "message" usando messageRef.get().
Usamos addOnSuccessListener para manejar el éxito de la operación:

- Si el nodo existe (snapshot.exists()), obtenemos su valor y lo mostramos en un Toast.
- Si el nodo no existe, mostramos un mensaje indicando que no hay datos.

Usamos addOnFailureListener para manejar errores durante la lectura, mostrando un mensaje de error si ocurre algún problema.

Explicación de activity_home.xml

```
<androidx.constraintlayout.widget.ConstraintLayout
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"
```

Utilizamos un ConstraintLayout como contenedor principal de la actividad. Este tipo de layout nos permite posicionar los elementos de la interfaz de manera flexible y eficiente.

Definimos el ancho y alto del layout como match_parent, lo que significa que ocupará todo el espacio disponible en la pantalla.

```
<androidx.constraintlayout.widget.ConstraintLayout
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"
```

Usamos el atributo tools:context para asociar este archivo XML con la clase HomeActivity. Esto ayuda a Android Studio a proporcionar sugerencias y herramientas de diseño específicas para esta actividad

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="¡Bienvenido a Home!"
```

Creamos un TextView para mostrar un mensaje de bienvenida en la pantalla.

Configuramos el ancho y alto como wrap_content, lo que significa que el tamaño del texto determinará el tamaño del TextView.

Establecemos el texto como "¡Bienvenido a Home!".

```
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintTop_toTopOf="parent" />
```

- Usamos restricciones (constraints) para posicionar el TextView:
    - Lo centramos horizontalmente (layout_constraintLeft_toLeftOf y layout_constraintRight_toRightOf).
    - Lo centramos verticalmente (layout_constraintTop_toTopOf y layout_constraintBottom_toBottomOf).
- Estas restricciones aseguran que el TextView esté perfectamente centrado en la pantalla, independientemente del tamaño del dispositivo.

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

Esta línea sólo cierra el contraint.layout.

Explicación de activity_login.xml

```
<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">
```

- Creamos un campo de texto para que el usuario ingrese su contraseña.
- Le asignamos un ID (tilContrasenaLogin) para poder referenciarlo desde el código Kotlin.
- Configuramos el ancho como match_parent y el alto como wrap_content.
- Establecemos un texto de sugerencia (hint) que dice "Contraseña".
- Habilitamos un botón de alternancia (passwordToggleEnabled) que permite al usuario mostrar u ocultar la contraseña.
- Usamos restricciones (constraints) para posicionar el campo debajo del campo de correo electrónico (layout_constraintTop_toBottomOf) y centrarlo horizontalmente (layout_constraintStart_toStartOf y layout_constraintEnd_toEndOf).
- Añadimos un margen superior de 16dp para separar este campo del campo anterior.

- Dentro del TextInputLayout, colocamos un TextInputEditText con el tipo de entrada (inputType) como textPassword, lo que oculta el texto ingresado.

```
<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">
```

- Creamos un campo de texto para que el usuario ingrese su correo electrónico. Utilizamos TextInputLayout y TextInputEditText, componentes de Material Design que mejoran la apariencia y funcionalidad de los campos de texto.
- Le asignamos un ID (tilCorreoLogin) para poder referenciarlo desde el código Kotlin.
- Configuramos el ancho como match_parent (ocupa todo el ancho disponible) y el alto como wrap_content (se ajusta al contenido).
- Establecemos un texto de sugerencia (hint) que dice "Correo electrónico".
- Usamos restricciones (constraints) para posicionar el campo en la parte superior de la pantalla (layout_constraintTop_toTopOf) y centrarlo horizontalmente (layout_constraintStart_toStartOf y layout_constraintEnd_toEndOf).

```
    <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>
```

- Dentro del TextInputLayout, colocamos un TextInputEditText para que el usuario ingrese el texto. Configuramos el tipo de entrada (inputType) como textEmailAddress, lo que optimiza el teclado para correos electrónicos.

```
<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/tilContrasenaLogin"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Contraseña"
```

- Creamos un campo de texto para que el usuario ingrese su contraseña.
- Le asignamos un ID (tilContrasenaLogin) para poder referenciarlo desde el código Kotlin.
- Configuramos el ancho como match_parent y el alto como wrap_content.

- Establecemos un texto de sugerencia (hint) que dice "Contraseña".

```
app:passwordToggleEnabled="true"
app:layout_constraintTop_toBottomOf="@+id/tilCorreoLogin"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintEnd_toEndOf="parent"
android:layout_marginTop="16dp">
```

- Habilitamos un botón de alternancia (passwordToggleEnabled) que permite al usuario mostrar u ocultar la contraseña.
- Usamos restricciones (constraints) para posicionar el campo debajo del campo de correo electrónico (layout_constraintTop_toBottomOf) y centrarlo horizontalmente (layout_constraintStart_toStartOf y layout_constraintEnd_toEndOf).
- Añadimos un margen superior de 16dp para separar este campo del campo anterior.
- Dentro del TextInputLayout, colocamos un TextInputEditText con el tipo de entrada (inputType) como textPassword, lo que oculta el texto ingresado.

```
    <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>
```


- Usamos restricciones (constraints) para posicionar el campo debajo del campo de correo electrónico (layout_constraintTop_toBottomOf) y centrarlo horizontalmente (layout_constraintStart_toStartOf y layout_constraintEnd_toEndOf).
- Añadimos un margen superior de 16dp para separar este campo del campo anterior.
- Dentro del TextInputLayout, colocamos un TextInputEditText con el tipo de entrada (inputType) como textPassword, lo que oculta el texto ingresado.


```
<com.google.android.material.button.MaterialButton
    android:id="@+id/btnIniciarSesion"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Iniciar Sesión"
```

- Creamos un botón utilizando MaterialButton, que es un componente de la biblioteca de Material Design para Android.

- Le asignamos un ID (btnIniciarSesion) para poder referenciarlo desde el código Kotlin.

```
app:layout_constraintTop_toBottomOf="@+id/tilContrasenaLogin"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintEnd_toEndOf="parent"
android:layout_marginTop="24dp" />
```

- Configuramos el ancho como match_parent (ocupa todo el ancho disponible) y el alto como wrap_content (se ajusta al contenido).
- Establecemos el texto del botón como "Iniciar Sesión".
- Usamos restricciones (constraints) para posicionar el botón debajo del campo de contraseña (layout_constraintTop_toBottomOf) y centrarlo horizontalmente (layout_constraintStart_toStartOf y layout_constraintEnd_toEndOf).
- Añadimos un margen superior de 24dp para separar el botón del campo anterior.

```
<TextView
    android:id="@+id/tvIntentosRestantes"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Intentos restantes: 3"
```

- Creamos un TextView para mostrar el número de intentos restantes de inicio de sesión.
- Le asignamos un ID (tvIntentosRestantes) para poder referenciarlo desde el código Kotlin.
- Configuramos el ancho y alto como wrap_content, lo que significa que el tamaño del texto determinará el tamaño del TextView.
- Establecemos el texto inicial como "Intentos restantes: 3".

```
android:layout_marginTop="16dp"
app:layout_constraintTop_toBottomOf="@+id/btnIniciarSesion"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintEnd_toEndOf="parent" />
```

- Usamos restricciones (constraints) para posicionar el TextView debajo del botón "Iniciar Sesión" (layout_constraintTop_toBottomOf) y centrarlo horizontalmente (layout_constraintStart_toStartOf y layout_constraintEnd_toEndOf).

- Añadimos un margen superior de 16dp para separar este texto del botón.

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

Cierra toda la definición de interfaz de pantalla del inicio de sesión.

Modificaciones en AndroidManifest.xml

Sólo se añadió: `<activity android:name=".HomeActivity" />` para especificar que ahora habrá una actividad dinámica que se llama “HomeActivity”.

Modificaciones en MainActivity.kt

Para esta actividad consideramos que no es necesario volver a escribir el código explicado en la práctica anterior, pero sí es relevante que empecemos por explicar los cambios en este documento y cómo se incorporan con las necesidades de la práctica actual.

Además del botón de: `startActivity(Intent(this, RegisterActivity::class.java))`
Añadimos un botón de: `startActivity(Intent(this, LoginActivity::class.java))`

Que es un botón en la actividad principal para que el usuario que ya tenga sesión registrada pueda iniciar sesión además de registrarse en caso de no tenerla.

Inicializa la vista de un botón registrado en los recursos visuales de la aplicación. Como en la práctica anterior para el botón de registro. Pero para este, se lanza la Activity del inicio de sesión.

```
findViewById<MaterialButton>(R.id.btnIniciarSesion_Main).setOnClickLis
tener {
    startActivity(Intent(this, LoginActivity::class.java))
}
```

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.util.Log
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import com.google.android.material.button.MaterialButton
import com.google.android.material.textfield.TextInputEditText
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.FirebaseAuthException
class LoginActivity : AppCompatActivity() {

    private lateinit var etCorreoLogin: TextInputEditText
    private lateinit var etContrasenaLogin: TextInputEditText
    private lateinit var btnIniciarSesion: MaterialButton
    private lateinit var tvIntentosRestantes: TextView
    private lateinit var auth: FirebaseAuth // Inicializa Firebase Auth
    private var intentosFallidos = 0
    private val handler = Handler(Looper.getMainLooper())

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

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

        // Vincula vistas
        etCorreoLogin = findViewById(R.id.etCorreoLogin)
        etContrasenaLogin = findViewById(R.id.etContrasenaLogin)
        btnIniciarSesion = findViewById(R.id.btnIniciarSesion)
        tvIntentosRestantes = findViewById(R.id.tvIntentosRestantes)
        tvIntentosRestantes.text = "Intentos restantes: 3"
        btnIniciarSesion.setOnClickListener { validarCredenciales() }
    }

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

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

        mostrarProgreso(true)

        auth.signInWithEmailAndPassword(correo, contrasena)
            .addOnCompleteListener(this) { task ->
                if (task.isSuccessful) {
                    mostrarMensaje("Sesión iniciada correctamente")
```

```
                    startActivity(Intent(this, HomeActivity::class.java))
                    finish()
                } else {
                    val exception = task.exception as?
FirebaseAuthException
                    handleFirebaseAuthError(exception)
                }
                mostrarProgreso(false)
            }
    }

    private fun handleFirebaseAuthError(exception:
FirebaseAuthException?) {
        intentosFallidos++
        when (exception?.errorCode) {
            "ERROR_WRONG_PASSWORD" -> {
                if (intentosFallidos >= 3) {
                    bloquearBotonTemporalmente()
                } else {
                    val restantes = 3 - intentosFallidos
                    mostrarMensaje("Contraseña incorrecta. Intentos
restantes: $restantes")
                    tvIntentosRestantes.text = "Intentos restantes:
$restantes"
                }
            }
            "ERROR_USER_NOT_FOUND" -> mostrarMensaje("Usuario no
encontrado")
            "ERROR_INVALID_EMAIL" -> mostrarMensaje("Correo electrónico
inválido")
            else -> {
                Log.e("FirebaseAuth", "Error de autenticación:
${exception?.message}")
                mostrarMensaje("Error de autenticación. Intenta de
nuevo.")
            }
        }
    }
    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)
    }

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

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

Código completo de HomeActivity.kt

```
package com.example.proyecto_equipo_2

import android.os.Bundle
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import com.google.firebase.database.FirebaseDatabase

class HomeActivity : AppCompatActivity() {

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

        // Obtén una referencia a Firebase Realtime Database
        val database = FirebaseDatabase.getInstance()
        val testRef = database.getReference("pruebaConexion")

        // Escribir datos de prueba en Firebase
        testRef.setValue("Conexión Exitosa")
            .addOnSuccessListener {
                // Mostrar un mensaje si se escribió correctamente
                Toast.makeText(this, "Conexión Exitosa: Dato guardado",
Toast.LENGTH_SHORT).show()
            }
            .addOnFailureListener { exception ->
                // Mostrar el error si falló
                Toast.makeText(this, "Error al conectar:
${exception.message}", Toast.LENGTH_SHORT).show()
            }

        // Leer el valor almacenado en Firebase
        testRef.get()
            .addOnSuccessListener { snapshot ->
                if (snapshot.exists()) {
                    val value = snapshot.value.toString()
                    Toast.makeText(this, "Valor leído: $value",
Toast.LENGTH_SHORT).show()
                } else {
                    Toast.makeText(this, "No existe el nodo en Firebase",
Toast.LENGTH_SHORT).show()
                }
            }
            .addOnFailureListener { exception ->
                Toast.makeText(this, "Error al leer datos:
${exception.message}", Toast.LENGTH_SHORT).show()
            }
    }
}
```

Código completo de RegisterActivity.kt

```
package com.example.proyecto_equipo_2

import android.content.Intent
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import com.google.android.material.button.MaterialButton

class MainActivity : AppCompatActivity() {

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

        // Configurar botones

findViewById<MaterialButton>(R.id.btnIrARegistro).setOnClickListener {
            startActivity(Intent(this, RegisterActivity::class.java)) //
Inicia RegisterActivity
        }
    }
}
```

Código completo de activity_home.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"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".HomeActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="¡Bienvenido a Home!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

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

Código completo de activity_main.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">

    <!-- Texto de bienvenida -->
    <TextView
        android:id="@+id/tvBienvenida"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Bienvenido a la App"
        android:textSize="24sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toTopOf="@+id/btnIrARegistro"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_chainStyle="packed" />

    <!-- Botón para redirigir al registro -->
    <com.google.android.material.button.MaterialButton
        android:id="@+id/btnIrARegistro"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="24dp"
        android:padding="12dp"
        android:text="Ir a Registro"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/tvBienvenida"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />

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

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" />

    <!-- 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>
```