

Actividad [3] - [-Codificación de la Aplicación de Lector de Huella]

[Desarrollo de Aplicaciones Biométricas]

Ingeniería En Desarrollo De Software

Tutor: Marco Alonso Rodríguez Tapia

Alumno: Josué de Jesús Laveaga Valenzuela

Fecha: 22/09/2023

INDICE

Introducción.....	1
Interpretación y Argumentación Del texto solicitado.....	1.1
Justificación	1.2
Ingreso al proyecto en Android Studio.....	2
Captura del archivo MainActivity.java.....	3
Explicación del archivo MainActivity.java.....	4
Archivo Gradle.kts (Agregar las librerías).....	5
Explicación del Archivo Gradle.kts (Agregar las librerías)..	6
Captura del archivo MainActivity2.java.....	7
Explicación del archivo MainActivity2.java.....	7.1
Evidencia de la app funcionando.....	8-9
Link Video/Link App/Conclusión.....	10

Introducción

En esta fase avanzada de desarrollo de nuestra aplicación, nos adentramos en un hito crítico: la implementación de la autenticación de huellas digitales. La autenticación biométrica, concretamente el escaneo de huellas digitales, representa un pilar fundamental en la revolución tecnológica de las aplicaciones móviles. Ha transformado la manera en que los usuarios interactúan con sus dispositivos, proporcionando un equilibrio entre seguridad y comodidad que redefine las expectativas de los usuarios en la era digital.

En esta etapa, nuestra misión es programar con precisión esta característica esencial. Nos sumergimos en la programación del sistema de autenticación que hará uso del sensor de huellas digitales integrado en el dispositivo. Cuando un usuario coloca su dedo en el sensor, el sistema procesa la información biométrica única y desencadena una respuesta inmediata. Según el resultado de esta verificación biométrica, desplegamos mensajes específicos en la interfaz de usuario, como "Escaneo de huella digital exitoso" cuando la autenticación es exitosa o "Escaneo fallido, huella dactilar no registrada" cuando no se reconoce la huella digital.

Interpretación y Argumentación Del texto solicitado

La tercera actividad de nuestra aplicación se enfoca en la implementación de la autenticación de huellas digitales. Utilizaremos el sensor de huellas digitales del dispositivo y programaremos la lógica necesaria para verificar la autenticidad del usuario. Cuando un usuario escanea su huella digital, el sistema procesa esta información y proporciona una respuesta instantánea. Dependiendo del resultado de esta verificación biométrica, se mostrarán mensajes adecuados en la interfaz de usuario. Esto incluye mensajes como "Escaneo de huella digital exitoso" cuando se completa con éxito y "Escaneo fallido, huella dactilar no registrada" cuando no se reconoce la huella digital.

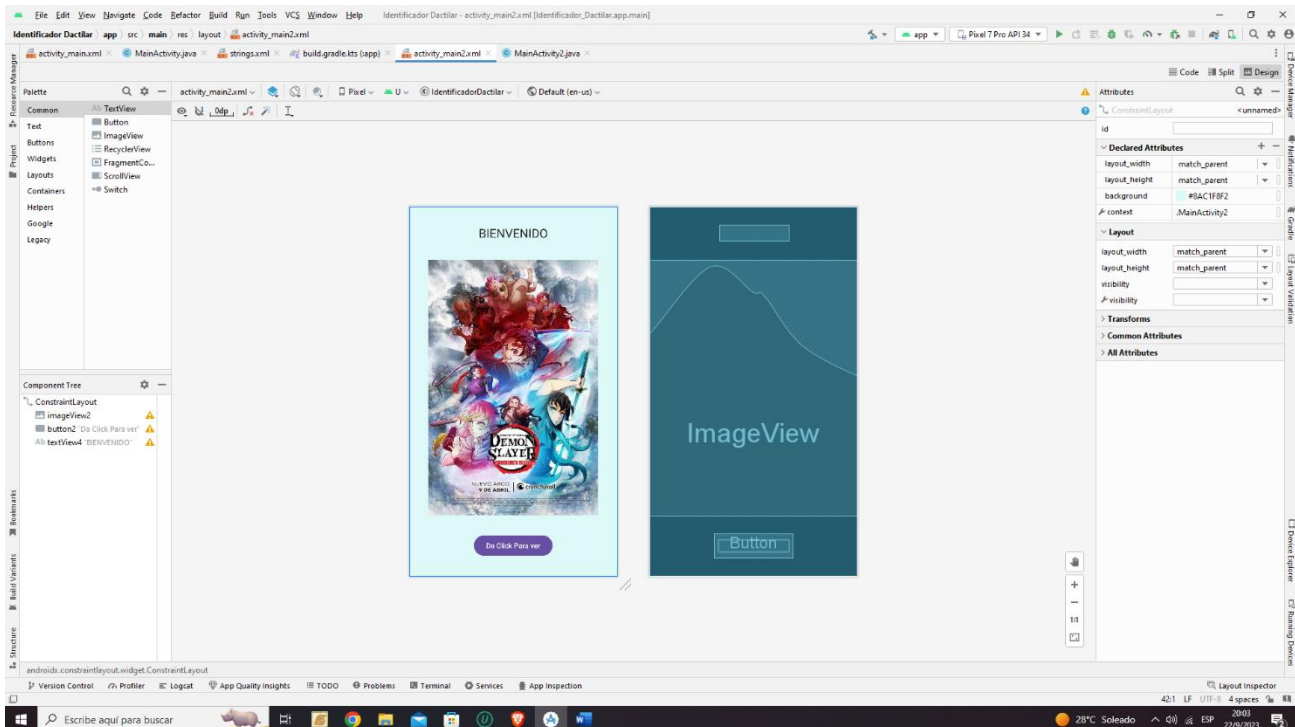
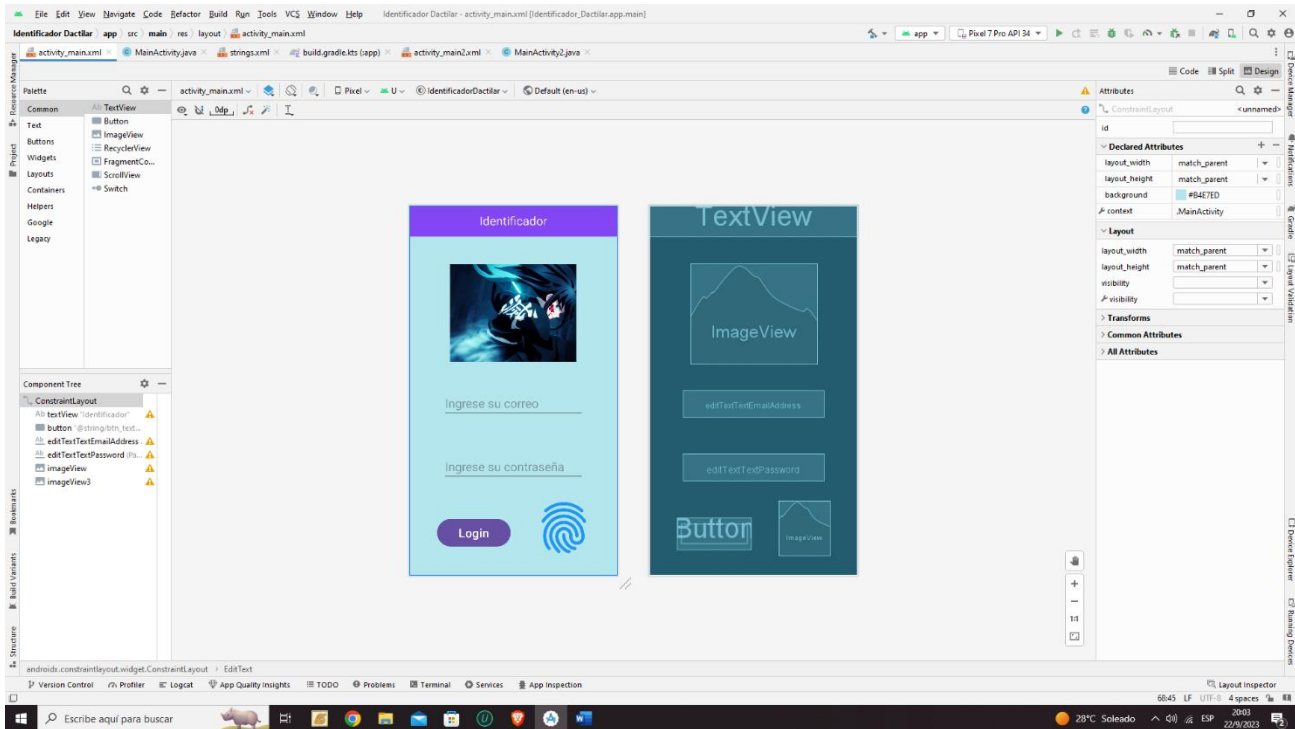
Uno de los aspectos cruciales de esta implementación es la capacidad de permitir que los usuarios accedan a la pantalla de bienvenida una vez que se haya completado una autenticación exitosa. Esto significa que la autenticación de huellas digitales no solo se trata de seguridad, sino también de brindar una experiencia de usuario fluida y agradable.

Justificación

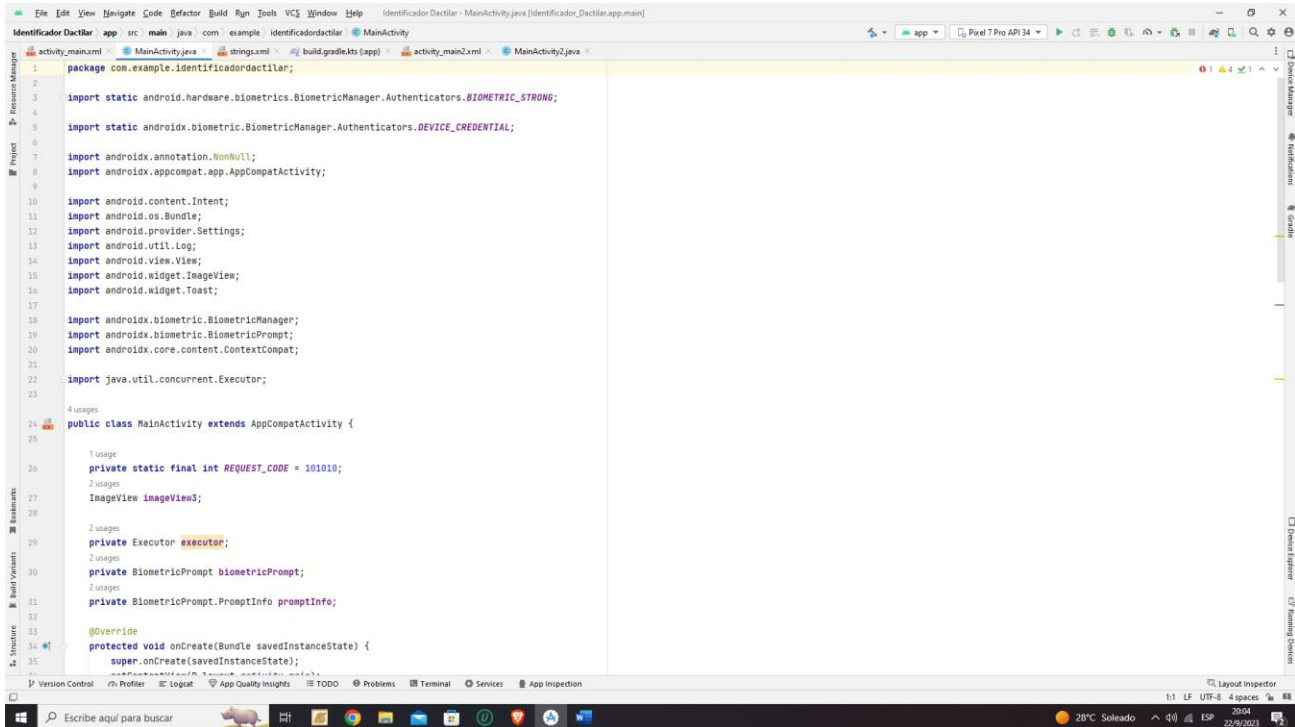
La inclusión de la autenticación de huellas digitales en nuestra aplicación es esencial por múltiples razones. En primer lugar, representa una medida de seguridad avanzada. La huella digital es una característica biométrica única y altamente segura. Al aprovechar esta tecnología, estamos fortaleciendo la protección de los datos del usuario, lo que se traduce en una mayor confianza y satisfacción de los usuarios.

En segundo lugar, la autenticación biométrica, incluido el escaneo de huellas digitales, se ha convertido en una expectativa estándar para los usuarios de aplicaciones móviles. Los usuarios buscan comodidad y seguridad, y esta característica cumple con ambas. Al incorporarla, mantenemos nuestra aplicación a la vanguardia de las tendencias tecnológicas y aseguramos que nuestra base de usuarios se sienta respaldada y protegida.

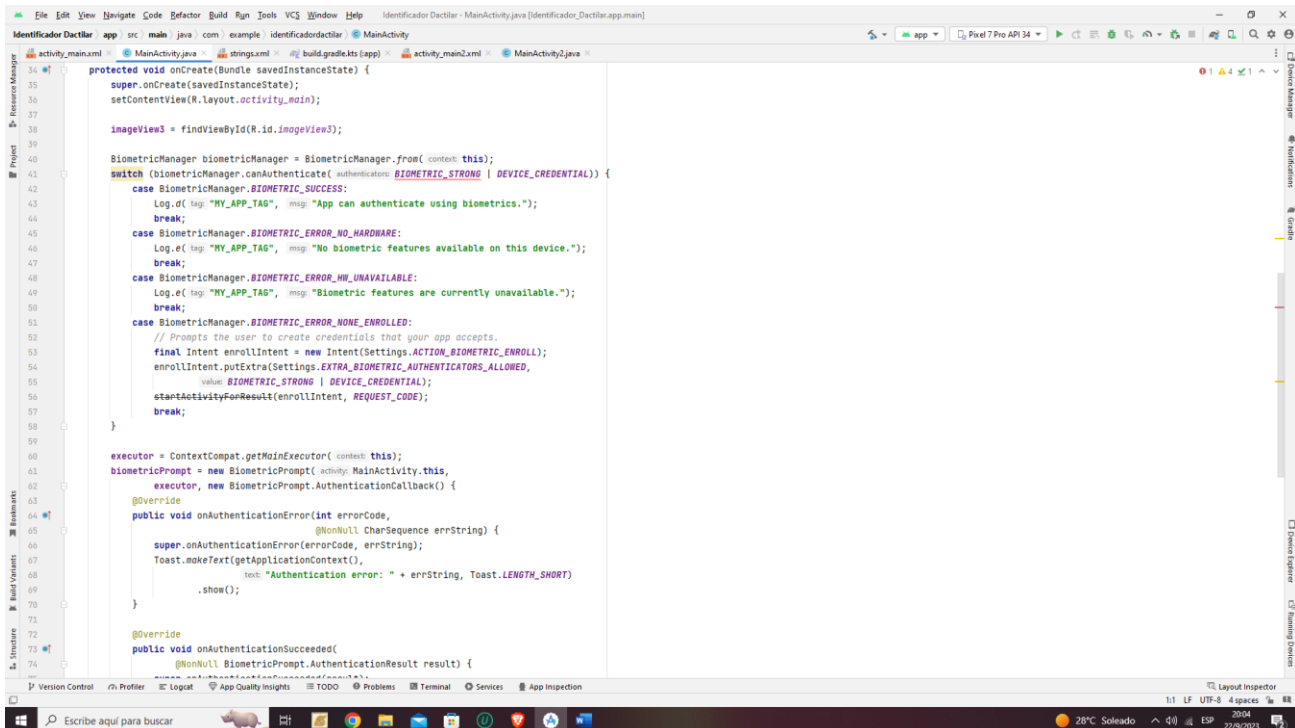
Ingreso al proyecto en Android Studio



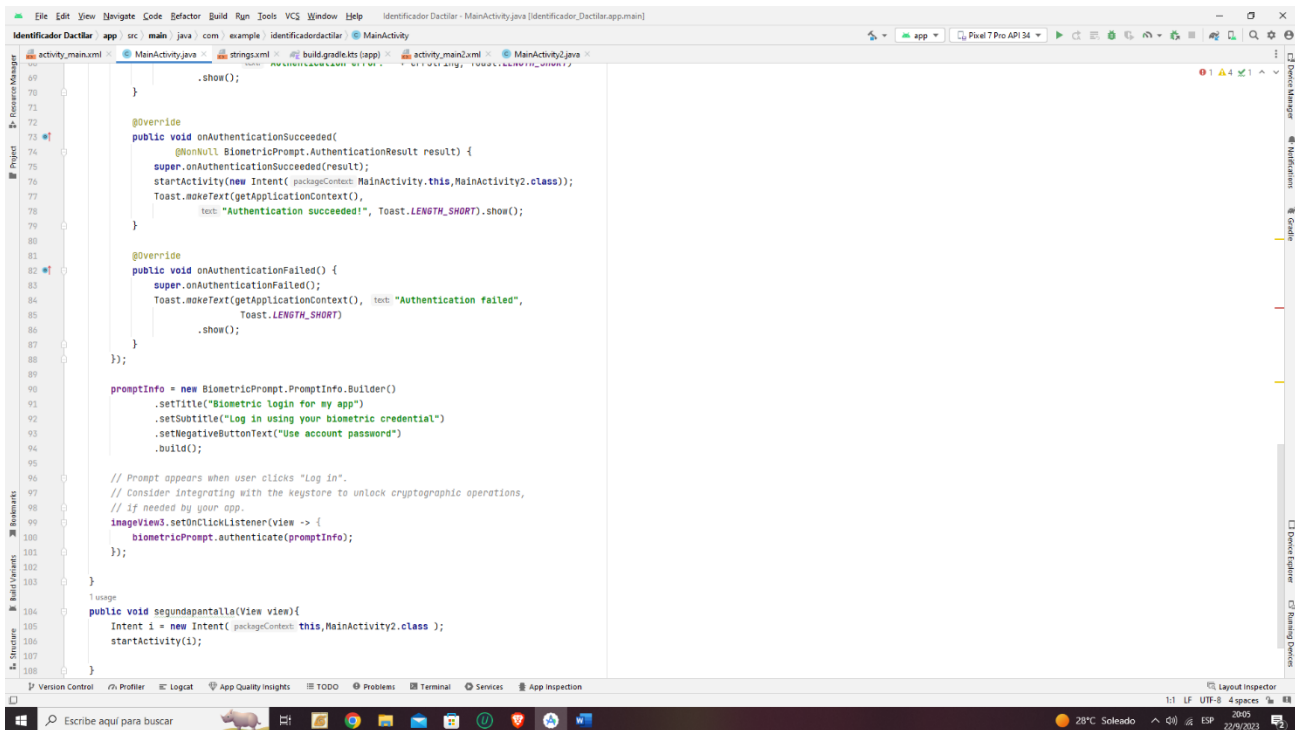
Captura del archivo MainActivity.java



```
1 package com.example.identificadordactilar;
2
3 import static android.hardware.biometrics.BiometricManager.Authenticators.BIOMETRIC_STRONG;
4
5 import static androidx.biometric.BiometricManager.Authenticators.DEVICE_CREDENTIAL;
6
7 import androidx.annotation.NonNull;
8 import androidx.appcompat.app.AppCompatActivity;
9
10 import android.content.Intent;
11 import android.os.Bundle;
12 import android.provider.Settings;
13 import android.util.Log;
14 import android.view.View;
15 import android.widget.ImageView;
16 import android.widget.Toast;
17
18 import androidx.biometric.BiometricManager;
19 import androidx.biometric.BiometricPrompt;
20 import androidx.core.content.ContextCompat;
21
22 import java.util.concurrent.Executor;
23
24 public class MainActivity extends AppCompatActivity {
25
26     private static final int REQUEST_CODE = 101010;
27     ImageView imageView3;
28
29     private Executor executor;
30     private BiometricPrompt biometricPrompt;
31     private BiometricPrompt.PromptInfo promptInfo;
32
33     @Override
34     protected void onCreate(Bundle savedInstanceState) {
35         super.onCreate(savedInstanceState);
36     }
37 }
```



```
34     protected void onCreate(Bundle savedInstanceState) {
35         super.onCreate(savedInstanceState);
36         setContentView(R.layout.activity_main);
37
38         imageView3 = findViewById(R.id.imageView3);
39
40         BiometricManager biometricManager = BiometricManager.from(this);
41         switch (biometricManager.canAuthenticate(authenticators: BIOMETRIC_STRONG | DEVICE_CREDENTIAL)) {
42             case BiometricManager.BIOMETRIC_SUCCESS:
43                 Log.d(tag: "MY_APP_TAG", msg: "App can authenticate using biometrics.");
44                 break;
45             case BiometricManager.BIOMETRIC_ERROR_NO_HARDWARE:
46                 Log.e(tag: "MY_APP_TAG", msg: "No biometric features available on this device.");
47                 break;
48             case BiometricManager.BIOMETRIC_ERROR_HW_UNAVAILABLE:
49                 Log.e(tag: "MY_APP_TAG", msg: "Biometric features are currently unavailable.");
50                 break;
51             case BiometricManager.BIOMETRIC_ERROR_NONE_ENROLLED:
52                 // Prompts the user to create credentials that your app accepts.
53                 final Intent enrollIntent = new Intent(Settings.ACTION_BIOMETRIC_ENROLL);
54                 enrollIntent.putExtra(Settings.EXTRA_BIOMETRIC_AUTHENTICATORS_ALLOWED,
55                     BiometricManager.BIOMETRIC_STRONG | DEVICE_CREDENTIAL);
56                 startActivityForResult(enrollIntent, REQUEST_CODE);
57                 break;
58         }
59
60         executor = ContextCompat.getMainExecutor(this);
61         biometricPrompt = new BiometricPrompt(activity: MainActivity.this,
62             executor, new BiometricPrompt.AuthenticationCallback() {
63                 @Override
64                 public void onAuthenticationError(int errorCode, @NonNull CharSequence errString) {
65                     super.onAuthenticationError(errorCode, errString);
66                     Toast.makeText(getApplicationContext(),
67                         text: "Authentication error: " + errString, Toast.LENGTH_SHORT)
68                         .show();
69                 }
70             }
71         );
72
73         @Override
74         public void onAuthenticationSucceeded(@NonNull BiometricPrompt.AuthenticationResult result) {
75             // Success
76         }
77     }
78 }
```



Explicación del archivo MainActivity.java

inicialización de la interfaz de usuario: La actividad principal (MainActivity) se encarga de la inicialización de la interfaz de usuario mediante el archivo XML correspondiente. Esto incluye la configuración de la vista y la inicialización de un ImageView llamado imageView3.

Gestión de la autenticación biométrica: Se utiliza la clase BiometricManager para evaluar la capacidad del dispositivo de manejar la autenticación biométrica. Se verifica si la autenticación biométrica fuerte y la autenticación mediante credenciales del dispositivo están disponibles.

Manejo de casos de BiometricManager: Dependiendo del resultado de la verificación anterior, se toman diferentes acciones:

Si la autenticación biométrica es exitosa, se muestra un mensaje de registro exitoso en los registros.

Si no hay hardware biométrico disponible en el dispositivo, se registra un mensaje de error indicando que no hay características biométricas disponibles.

Si las características biométricas no están disponibles en ese momento (por ejemplo, si no se ha configurado ninguna huella digital), se inicia una actividad para permitir al usuario registrar nuevas credenciales biométricas.

Configuración del prompt biométrico: Se configura un objeto BiometricPrompt con una devolución de llamada personalizada para manejar los resultados de autenticación biométrica. También se configura la información del prompt, incluyendo el título y el texto del botón negativo.

Inicio de la autenticación biométrica: Cuando el usuario hace clic en imageView3, se inicia el proceso de autenticación biométrica mediante el objeto biometricPrompt configurado previamente.

Manejo de eventos de autenticación: Se definen tres métodos de devolución de llamada para gestionar los eventos de autenticación biométrica:

onAuthenticationError: Maneja errores durante el proceso de autenticación.

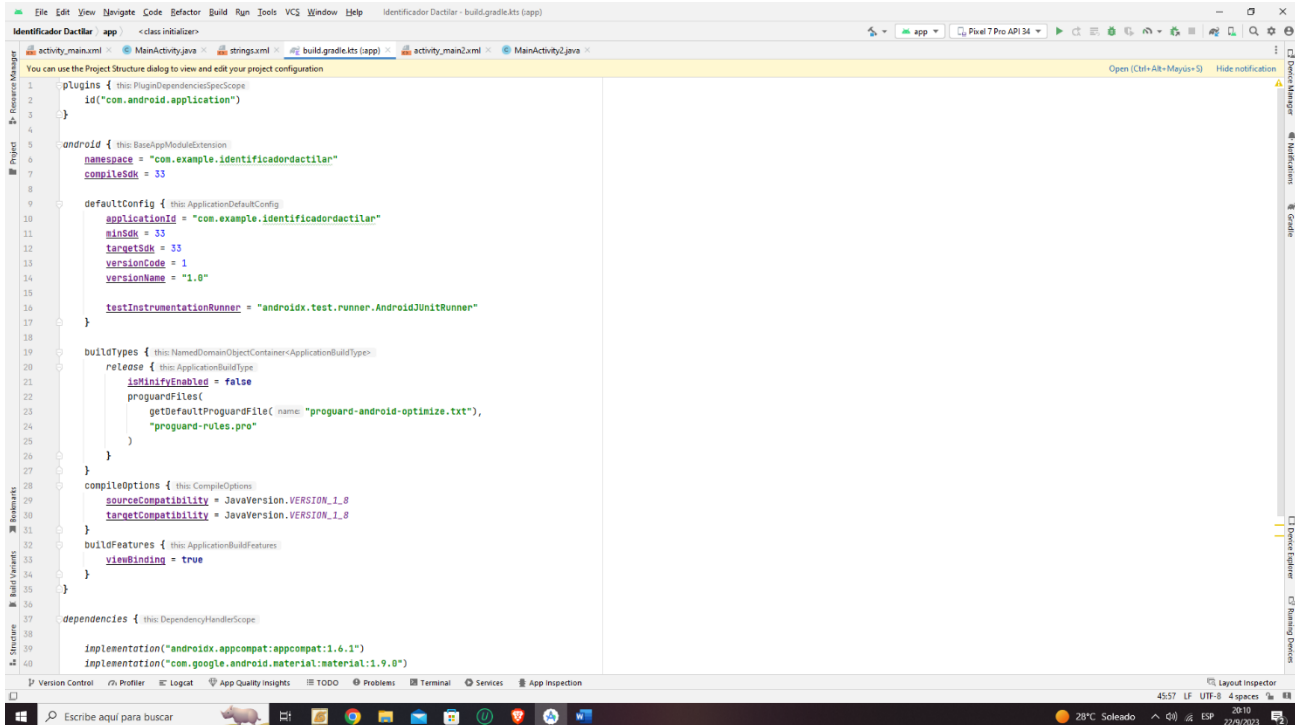
onAuthenticationSucceeded: Maneja la autenticación exitosa y muestra un mensaje de registro exitoso.

onAuthenticationFailed: Maneja el caso en que la autenticación biométrica falla.

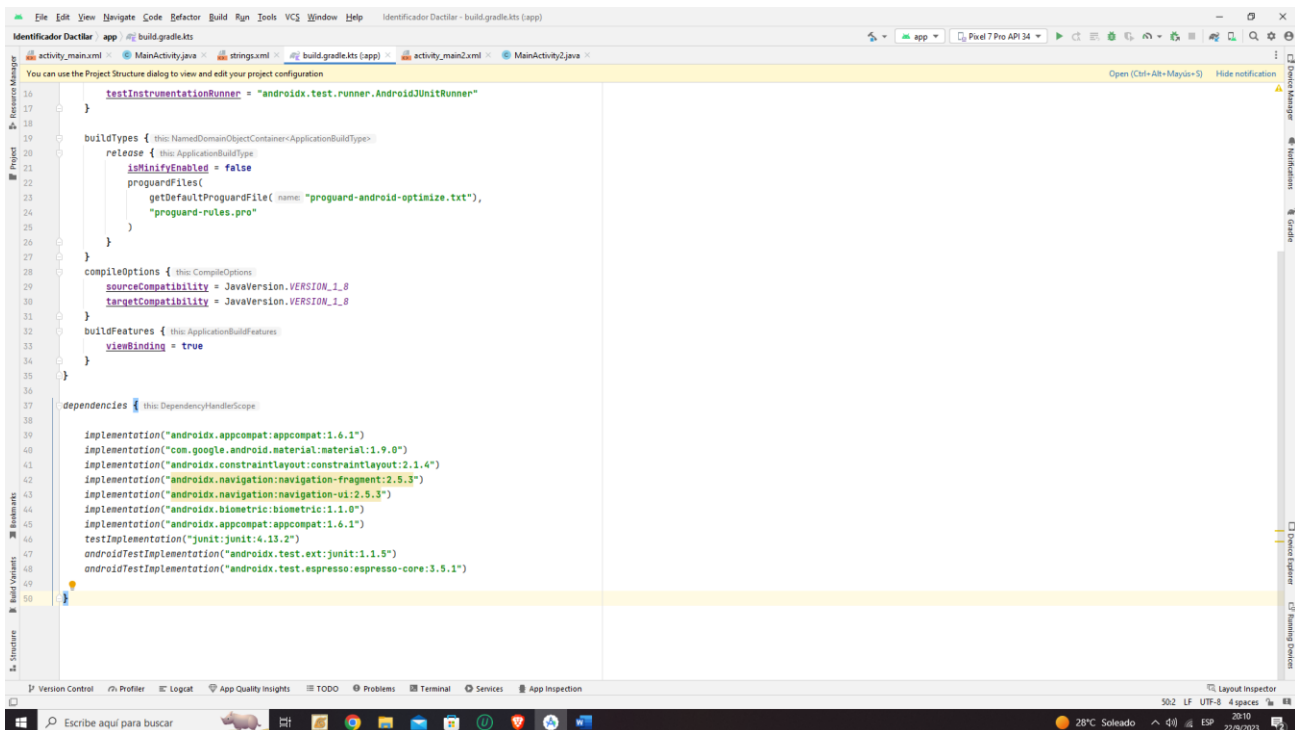
Redirección a la segunda pantalla: Cuando la autenticación biométrica es exitosa, se inicia una nueva actividad (MainActivity2) que probablemente representa la pantalla de bienvenida de la aplicación.

Este código se encarga de gestionar la autenticación biométrica en una aplicación Android, verificando la disponibilidad de hardware biométrico, configurando un prompt de autenticación, manejando eventos de autenticación y redirigiendo al usuario a la pantalla principal después de una autenticación exitosa.

Archivo Gradle.kts (Agregar las librerías)



```
1 plugins { this PluginDependenciesSpecScope
2     id("com.android.application")
3 }
4
5 android { this BaseAppModuleExtension
6     namespace = "com.example.identificadordactilar"
7     compileSdk = 33
8
9     defaultConfig { this ApplicationDefaultConfig
10         applicationId = "com.example.identificadordactilar"
11         minSdk = 33
12         targetSdk = 33
13         versionCode = 1
14         versionName = "1.0"
15
16         testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
17     }
18
19     buildTypes { this NamedDomainObjectContainer<ApplicationBuildType>
20         release { this ApplicationBuildType
21             isMinifyEnabled = false
22             proguardFiles(
23                 getDefaultProguardFile( name: "proguard-android-optimize.txt",
24                     "proguard-rules.pro"
25             )
26         }
27     }
28
29     compileOptions { this CompileOptions
30         sourceCompatibility = JavaVersion.VERSION_1_8
31         targetCompatibility = JavaVersion.VERSION_1_8
32     }
33
34     buildFeatures { this ApplicationBuildFeatures
35         viewBinding = true
36     }
37
38     dependencies { this DependencyHandlerScope
39         implementation("androidx.appcompat:appcompat:1.6.1")
40         implementation("com.google.android.material:material:1.9.0")
41     }
42 }
```



```
16     testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
17 }
18
19 buildTypes { this NamedDomainObjectContainer<ApplicationBuildType>
20     release { this ApplicationBuildType
21         isMinifyEnabled = false
22         proguardFiles(
23             getDefaultProguardFile( name: "proguard-android-optimize.txt",
24                 "proguard-rules.pro"
25             )
26         )
27     }
28
29     compileOptions { this CompileOptions
30         sourceCompatibility = JavaVersion.VERSION_1_8
31         targetCompatibility = JavaVersion.VERSION_1_8
32     }
33
34     buildFeatures { this ApplicationBuildFeatures
35         viewBinding = true
36     }
37
38     dependencies { this DependencyHandlerScope
39         implementation("androidx.appcompat:appcompat:1.6.1")
40         implementation("com.google.android.material:material:1.9.0")
41         implementation("androidx.constraintlayout:constraintlayout:2.1.4")
42         implementation("androidx.navigation:navigation-fragment:2.5.3")
43         implementation("androidx.navigation:navigation-ui:2.5.3")
44         implementation("androidx.biometric:biometric:1.1.0")
45         implementation("androidx.appcompat:appcompat:1.6.1")
46         testImplementation("junit:junit:4.13.2")
47         androidTestImplementation("androidx.test.ext:junit:1.1.5")
48         androidTestImplementation("androidx.test.espresso:espresso-core:3.5.1")
49     }
50 }
```

Explicación del Archivo Gradle.kts (Agregar las librerías)

Dependencias mencionadas en el archivo de construcción del proyecto (build.gradle):

`implementation("androidx.biometric:biometric:1.1.0"):`

Esta dependencia pertenece a la biblioteca `androidx.biometric` y se utiliza para integrar funcionalidades relacionadas con la autenticación biométrica en tu aplicación Android. La versión específica que estás utilizando es la "1.1.0".

La biblioteca `androidx.biometric` proporciona las clases y métodos necesarios para interactuar con el sistema de autenticación biométrica del dispositivo, como el escaneo de huellas digitales, el reconocimiento facial o la autenticación de iris.

Al incluir esta dependencia en tu proyecto, puedes aprovechar las características biométricas del dispositivo de manera sencilla y segura para autenticar a los usuarios en tu aplicación.

`implementation("androidx.appcompat:appcompat:1.6.1"):`

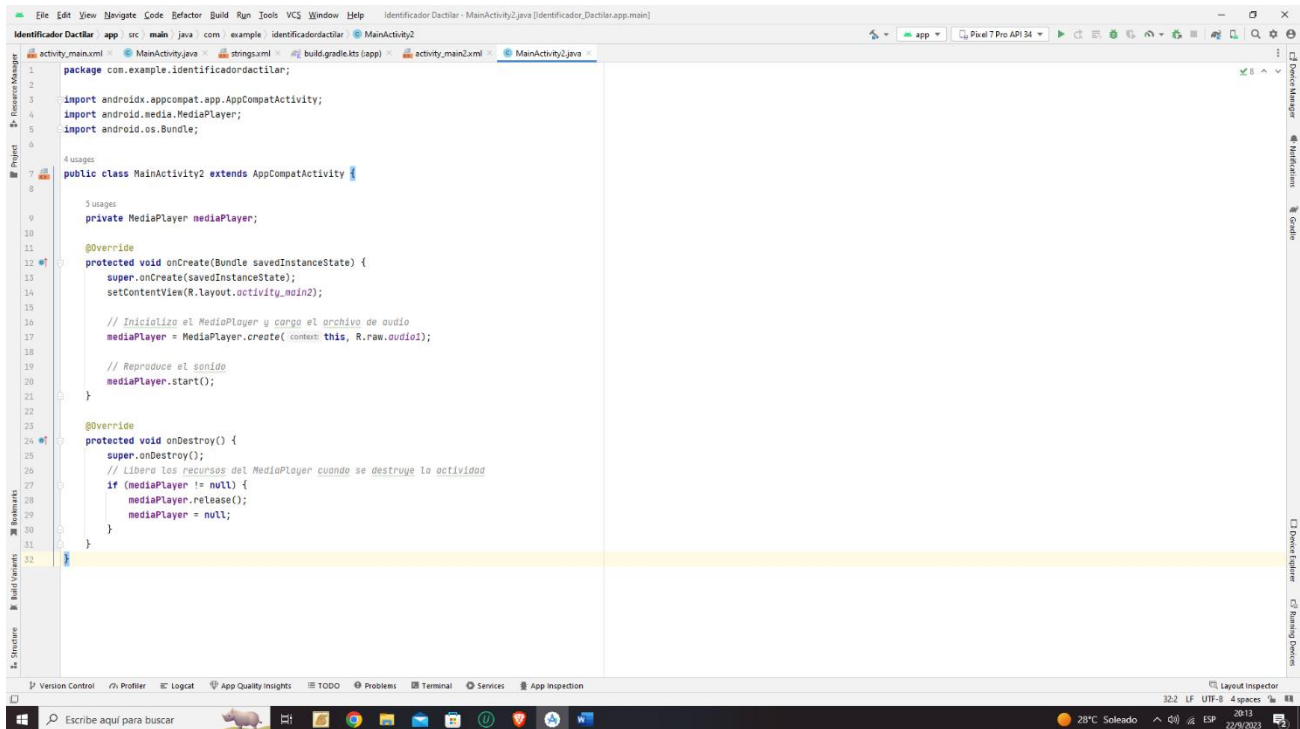
Esta dependencia pertenece a la biblioteca `androidx.appcompat`, que proporciona un conjunto de herramientas y utilidades que permiten crear interfaces de usuario modernas y compatibles con versiones anteriores de Android.

La versión específica que estás utilizando es la "1.6.1".

`androidx.appcompat` es esencial en la mayoría de las aplicaciones de Android modernas, ya que ayuda a garantizar la consistencia de la apariencia y el comportamiento de la aplicación en diferentes versiones de Android.

También proporciona funcionalidades de compatibilidad con las últimas características de diseño de Material Design y facilita la creación de aplicaciones con una apariencia coherente en dispositivos con diferentes versiones de Android.

Captura del archivo MainActivity2.java



Explicación del archivo MainActivity2.java

MainActivity2 de una aplicación Android. Aquí está la explicación de lo que hace este código:

Extensión de AppCompatActivity: La clase MainActivity2 extiende AppCompatActivity, lo que significa que esta actividad está diseñada para funcionar en versiones anteriores de Android y es compatible con características modernas de Android.

Declaración de un objeto MediaPlayer: Se declara una instancia de la clase MediaPlayer llamada mediaPlayer. Esta clase se utiliza para reproducir archivos de audio en la aplicación.

Método onCreate(): Este método se llama cuando la actividad se crea. Aquí es donde se inicializan y configuran muchos de los componentes y recursos utilizados por la actividad.

setContentView(R.layout.activity_main2): Establece el diseño de la interfaz de usuario para esta actividad. El diseño se carga desde el archivo XML activity_main2.xml.

mediaPlayer = MediaPlayer.create(this, R.raw.audio1): Inicializa el objeto mediaPlayer y carga un archivo de audio para su reproducción. El archivo de audio se encuentra en la carpeta res/raw de tu proyecto y se llama audio1.

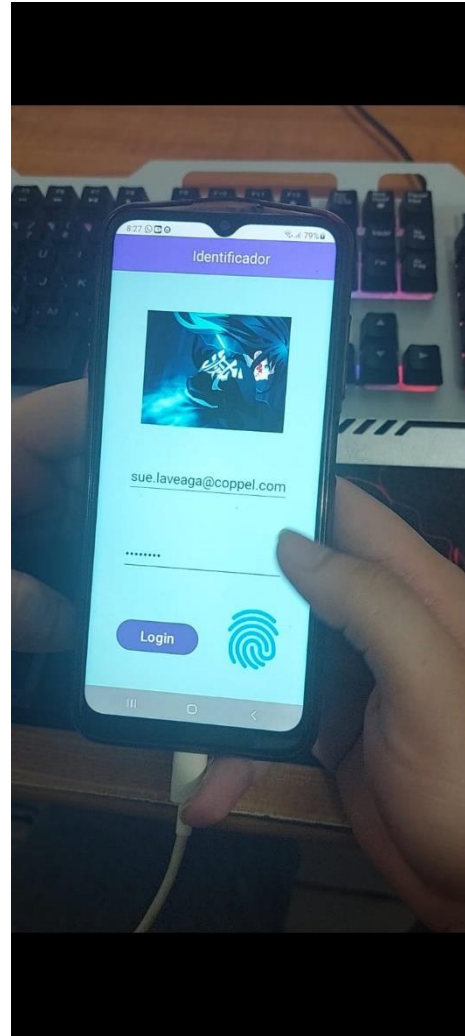
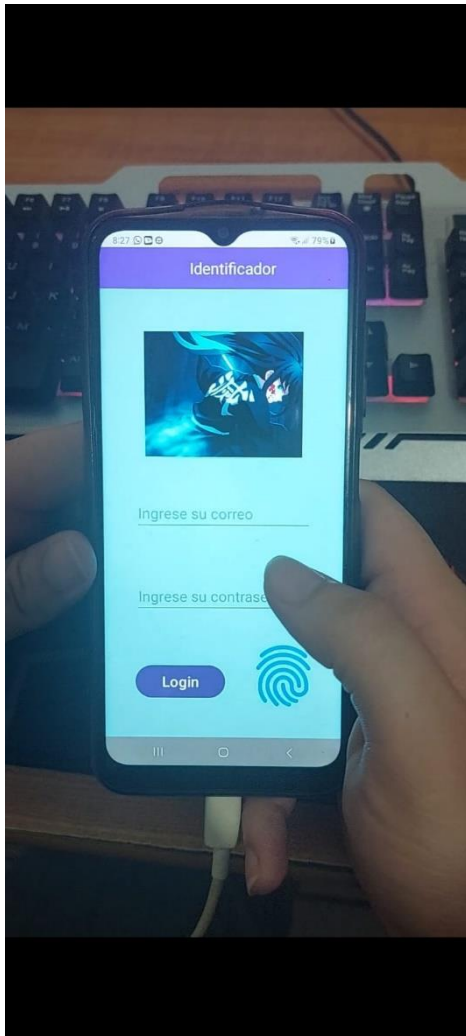
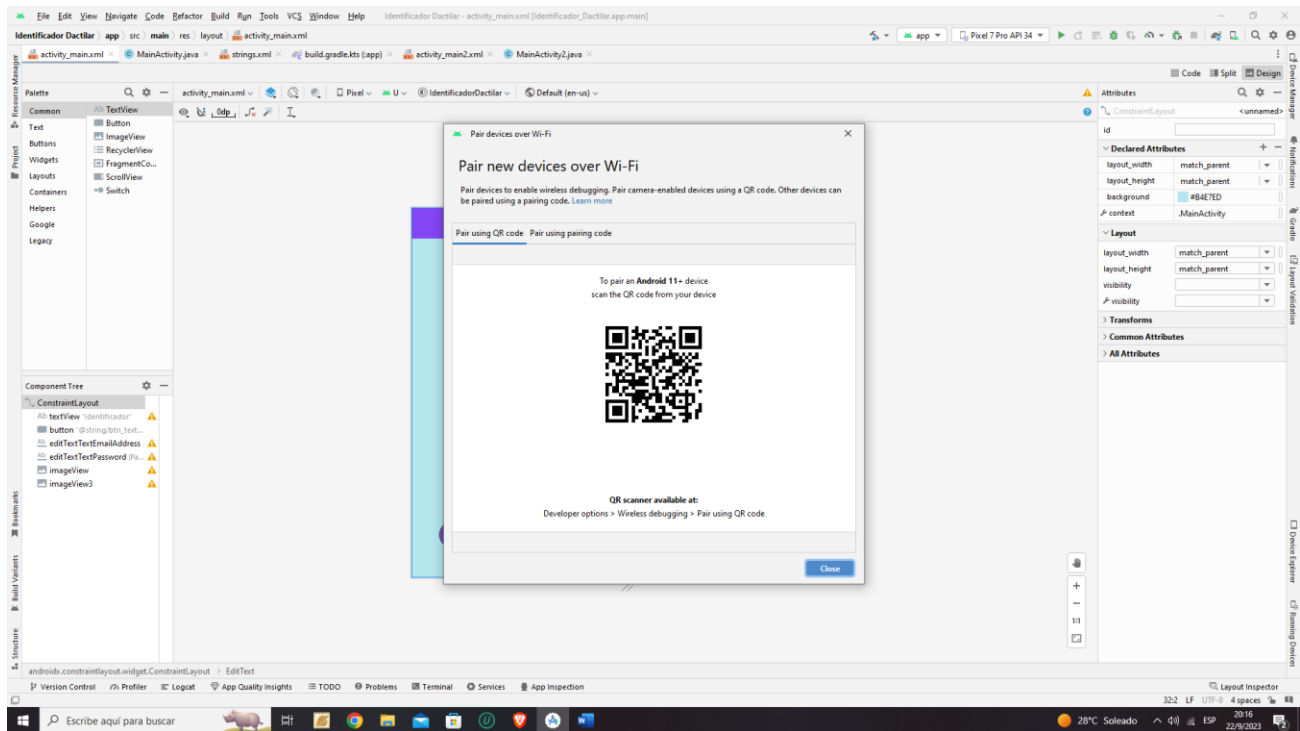
mediaPlayer.start(): Inicia la reproducción del archivo de audio cargado en el mediaPlayer. Esto reproduce el sonido en segundo plano cuando la actividad se crea.

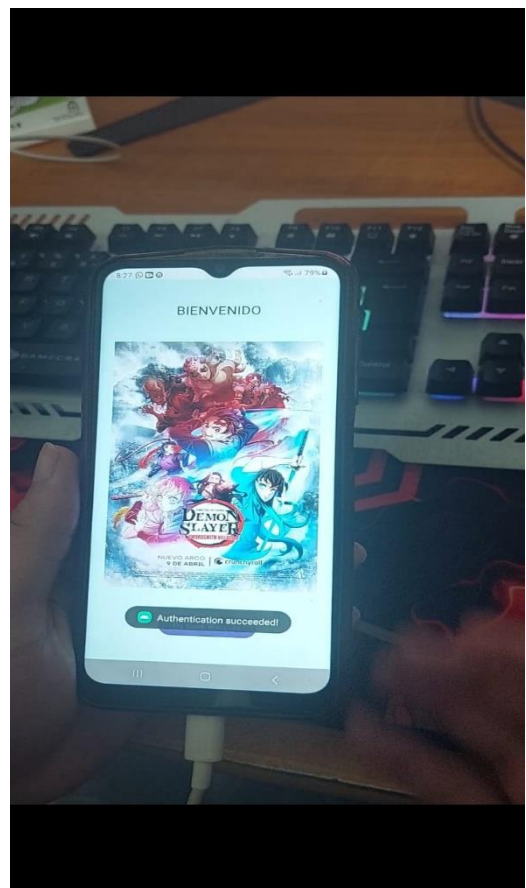
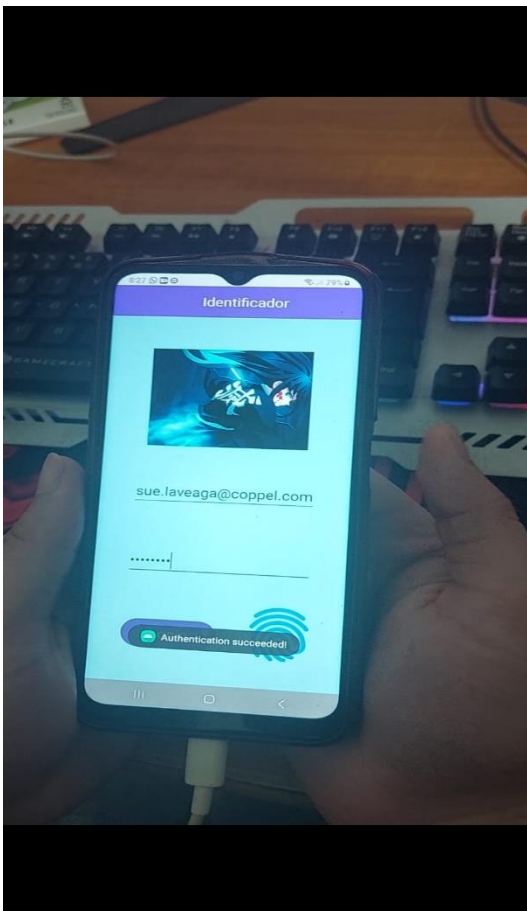
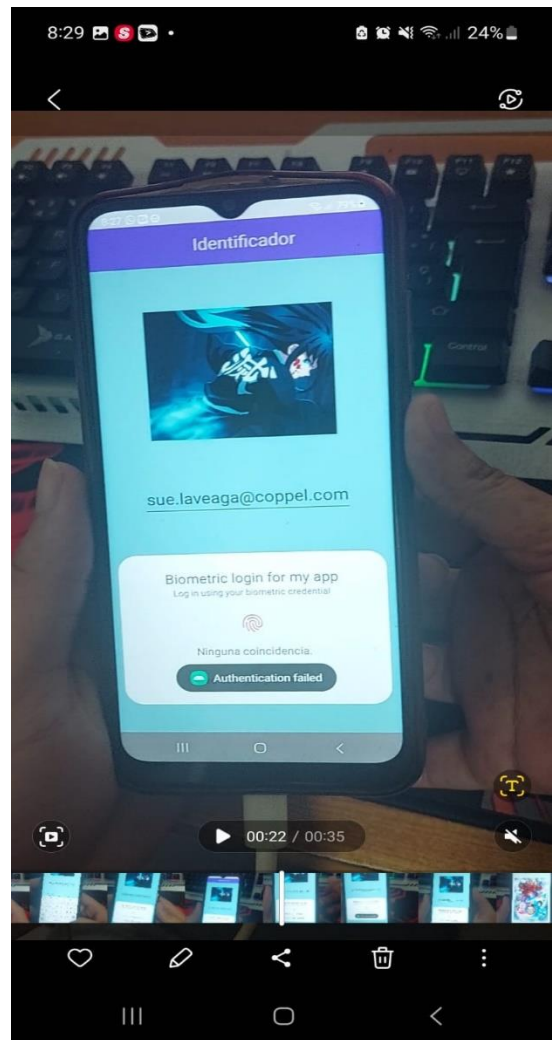
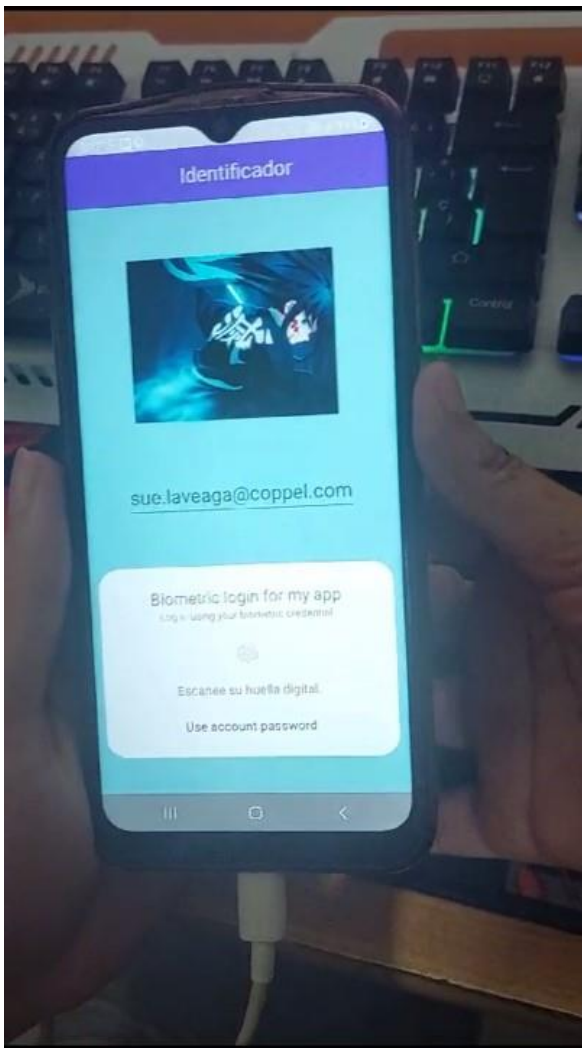
Método onDestroy(): Este método se llama cuando la actividad se destruye. Aquí se realiza la limpieza y liberación de recursos, en este caso, el objeto mediaPlayer.

mediaPlayer.release(): Libera los recursos utilizados por el mediaPlayer. Esto es importante para evitar fugas de memoria y asegurarse de que el recurso se libere adecuadamente cuando la actividad se destruye.

mediaPlayer = null; Asigna null al objeto mediaPlayer para indicar que ya no se hace referencia a él y está listo para ser recolectado por el recolector de basura de Android.

Evidencia de la app funcionando





Link de video para su descarga y visualización correcta

<https://drive.google.com/file/d/1-ibQU-3yiLdUogwAkCgmGiLeUNVgwUa-/view?usp=sharing>

Link de la App de Android Studio

https://drive.google.com/file/d/1xzUWpa4jfcXZyxoe_ljSrtht4wi45cVU/view?usp=sharing

Conclusión

En la tercera actividad de desarrollo de la aplicación, hemos logrado un avance significativo al implementar la funcionalidad de autenticación biométrica, específicamente el escaneo de huellas digitales. Este paso es crucial tanto desde el punto de vista de la seguridad como de la experiencia del usuario.

Hemos logrado los siguientes hitos clave:

Integración de la Autenticación Biométrica: Hemos utilizado la biblioteca `androidx.biometric` para integrar de manera efectiva la autenticación biométrica en nuestra aplicación. Esto permite a los usuarios autenticarse de manera segura utilizando sus huellas digitales.

Gestión de Casos Biométricos: Hemos implementado una lógica que verifica la disponibilidad de la autenticación biométrica en el dispositivo y toma las medidas adecuadas en función de ese estado. Si el hardware biométrico no está disponible, se proporciona al usuario una opción para configurarlo.

Interacción con el Usuario: Hemos configurado un prompt biométrico personalizado para interactuar con los usuarios durante el proceso de autenticación. Esto incluye la presentación de mensajes de éxito, errores o fallos en la autenticación biométrica.

Transición a la Pantalla de Bienvenida: Una vez que la autenticación biométrica tiene éxito, hemos implementado una transición suave a la pantalla de bienvenida (`MainActivity2`) de nuestra aplicación.

Además, en esta actividad, también hemos incorporado la reproducción de un archivo de audio cuando se accede a la pantalla de bienvenida, brindando una experiencia multisensorial al usuario.

En conjunto, estos logros representan un paso esencial hacia la creación de una aplicación segura, moderna y atractiva para nuestros usuarios. La autenticación biométrica no solo mejora la seguridad al proporcionar un método de autenticación altamente confiable, sino que también mejora la experiencia del usuario al eliminar la necesidad de contraseñas complicadas. La reproducción de audio agrega un toque adicional de inmersión, haciendo que la aplicación sea más atractiva y cautivadora.

Continuaremos avanzando en el desarrollo de la aplicación, construyendo sobre esta base sólida, y trabajando en nuevas características y mejoras que beneficien a nuestros usuarios y fortalezcan la posición de nuestra aplicación en el mercado móvil.