Aplicación móvil para identificar si un conductor se queda dormido usando modelos de Tensorflow Lite

Daniel Arce
Ingeniería mecatrónica
Universidad Nacional Colombia

Resumen – A lo largo de este proyecto se programó una aplicación móvil para dispositivos Android usando Android Studio y programando en java. La aplicación usa un modelo de tensorflow lite para determinar si el conductor tiene los ojos cerrados y lanza una alerta para despertarlo.

Índice de Términos – IA, sueño, java, Android Studio

I. INTRODUCCION

Como proyecto final de la materia: programación orientada a objetos se debía generar una aplicación que hiciera uso de la programación orientada a objetos en java y modelos de Machine Learning para resolver alguna problemática real. En este caso se pensó en una de las causas de accidentes automovilísticos más grandes, la somnolencia. La idea era generar una aplicación móvil hecha en Android Studio que pueda definir si el conductor del vehículo se está quedando dormido y en caso de que lo haga sea capaz de despertarlo o de avisar y así evitar un accidente.

II. CÓDIGO

A continuación, se desglosará paso a paso cada clase de la aplicación.

Documento recibido el 27 de febrero de 2025. Daniel Santiago Arce Sánchez. Este es un Proyecto de programación orientada a objetos en la Universidad Nacional de Colombia.

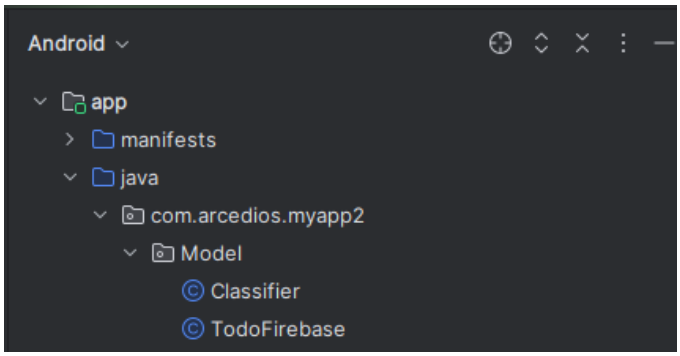
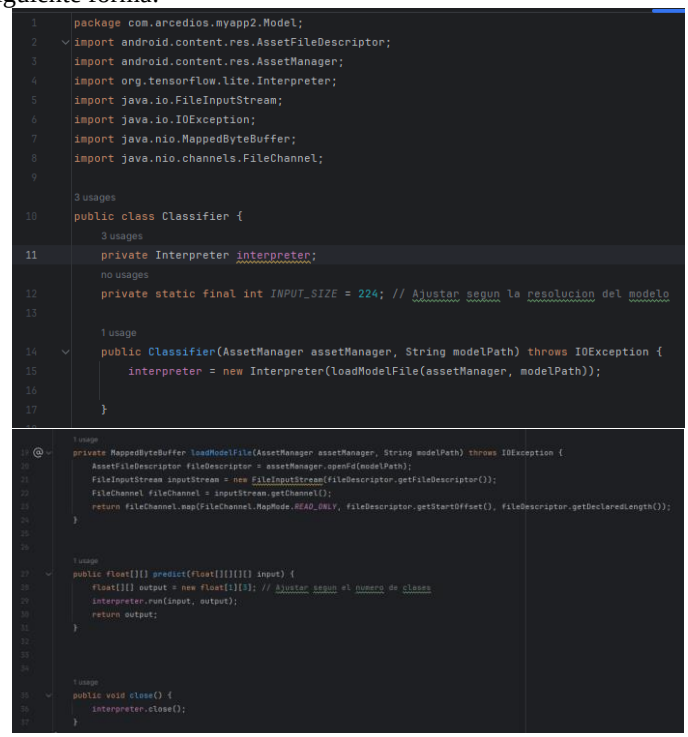


Imagen 1. Model

Dentro del paquete model se pueden encontrar las clases: Classifier.java y TodoFirebase.java, la primera se ve de la siguiente forma:



```
package com.arcedios.myapp2.Model;
import android.content.res.AssetFileDescriptor;
import android.content.res.AssetManager;
import org.tensorflow.lite.Interpreter;
import java.io.FileInputStream;
import java.io.IOException;
import java.nio.MappedByteBuffer;
import java.nio.channels.FileChannel;

3 usages
public class Classifier {
    3 usages
    private Interpreter interpreter;
    no usages
    private static final int INPUT_SIZE = 224; // Ajustar segun la resolucion del modelo

    1 usage
    public Classifier(AssetManager assetManager, String modelPath) throws IOException {
        interpreter = new Interpreter(loadModelFile(assetManager, modelPath));

    }

    1 usage
    private MappedByteBuffer loadModelFile(AssetManager assetManager, String modelPath) throws IOException {
        AssetFileDescriptor fileDescriptor = assetManager.openFd(modelPath);
        FileInputStream inputStream = new FileInputStream(fileDescriptor.getFileDescriptor());
        FileChannel fileChannel = inputStream.getChannel();
        return fileChannel.map(FileChannel.MapMode.READ_ONLY, fileDescriptor.getStartOffset(), fileDescriptor.getDeclaredLength());
    }


    1 usage
    public float[][] predict(float[][][][] input) {
        float[][] output = new float[1][3]; // Ajustar segun el numero de clases
        interpreter.run(input, output);
        return output;
    }


    1 usage
    public void close() {
        interpreter.close();
    }
```

Imagen 2. Clase Classifier.java

En esta clase se hace el proceso para cargar y ejecutar el modelo de tensorflow lite que almacené anteriormente en la carpeta assets, el modelo almacenado se llama: model.tflite

para que el modelo sea capaz de analizar las fotos estas se deben encontrar en formato de 224x224 pixels.

```
            @Override
            public void onCancelled(@NonNull DatabaseError error) {
                Log.e( tag: "TodoFirebase",  msg: "onCancelled called: " + error.getMessage());
                callback.onIdentificarResult( nombre: null); // Handle the error case
            }
        });
    }


}
```

Imagen 3. TodoFirebase.java

La función principal de esta clase es subir los datos del usuario o conductor a la base de datos para posteriormente poder registrar o hacer el login al abrir la aplicación. Estos datos se almacenan en hashmap y la base de datos se ve de la siguiente forma:

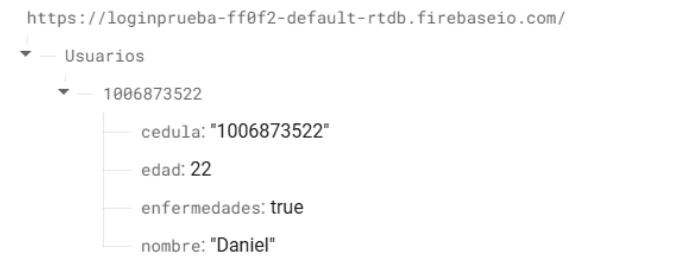


Imagen 4. Ejemplo base de datos

Después tengo la clase Todofirebse.java:

```
package com.arcedios.myapp2.Model;

import android.util.Log;

import androidx.annotation.NonNull;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.util.HashMap;
import java.util.concurrent.atomic.AtomicBoolean;

import com.arcedios.myapp2.ModelView.Usuarios;

7 usages
public class TodoFirebase {
    no usages
    public boolean aver;

    1 usage
    private static DatabaseReference databaseReference = null;

    public TodoFirebase() {
        // Initialize the database reference in the constructor
        databaseReference = FirebaseDatabase.getInstance().getReference();
    }
    public static boolean addUser(Usuarios p) {
        AtomicBoolean success = new AtomicBoolean( initialValue: false);
        TodoFirebase firebase = new TodoFirebase();
        FirebaseDatabase database = FirebaseDatabase.getInstance();
        DatabaseReference myRef = database.getReference();
        HashMap<String, Object> map = new HashMap<>();
        map.put("nombre", p.getNombre());
        map.put("cedula", p.getCedula());
        map.put("edad", p.getEdad());
        map.put("enfermedades", p.getEnfermedades());
        myRef.child( pathString: "Usuarios").child(p.getCedula()).setValue(map).addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                success.set(true);
            }


        });
        return success.get();
    }

    1 implementation
    public interface IdentificarCallback {
        1 implementation
        void onIdentificarResult(String nombre);
    }

    public static void Identificar(String cedula, IdentificarCallback callback) {
        Log.d( tag: "TodoFirebase",  msg: "Identificar called with cedula: " + cedula);
        if (cedula == null || cedula.isEmpty()) {
            Log.d( tag: "TodoFirebase",  msg: "cedula is null or empty");
            callback.onIdentificarResult( nombre: null); // Indicate an error
            return;
        }

        FirebaseDatabase database = FirebaseDatabase.getInstance();
        DatabaseReference myRef = database.getReference( path: "Usuarios");

        myRef.child(cedula).addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                Log.d( tag: "TodoFirebase",  msg: "onDataChange called");
                if (snapshot.exists()) {
                    Log.d( tag: "TodoFirebase",  msg: "snapshot exists");
                    Usuarios p = snapshot.getValue(Usuarios.class);
                    if (p != null) {
                        Log.d( tag: "TodoFirebase",  msg: "User found: " + p.getNombre());
                        callback.onIdentificarResult(p.getNombre());
                    } else {
                        Log.d( tag: "TodoFirebase",  msg: "User is null");
                        callback.onIdentificarResult( nombre: null); // Handle the case where p is null
                    }
                } else {
                    Log.d( tag: "TodoFirebase",  msg: "snapshot does not exist");
                    callback.onIdentificarResult( nombre: "No existe"); // Handle the case where the snapshot doesn't exist
                }
            }
```

Pasando al paquete ModelView:

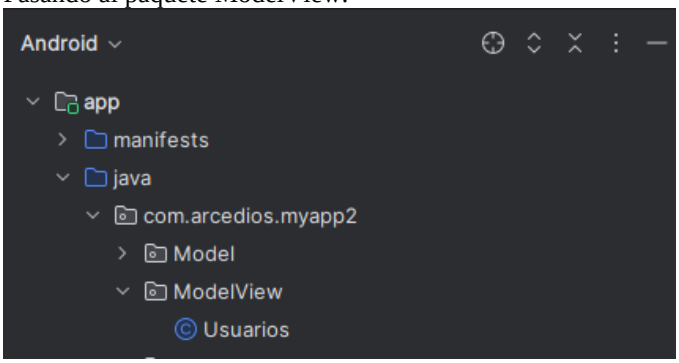


Imagen 5. Modelview

Dentro de este paquete se encuentra unicamente la clase Usuarios, en la cual se almacena la información del conductor:

```java
package com.arcedios.myapp2.ModelView;

7 usages
public class Usuarios {
    2 usages
    private   String nombre;
    2 usages
    private  String cedula;
    2 usages
    private  int edad;
    2 usages
    private  boolean enfermedades;
    1 usage
    public Usuarios() {
    }
    3 usages
    public  String getNombre() {

        return nombre;
    }

    1 usage
    public void setNombre(String nombre) {


        this.nombre = nombre;
    }

    2 usages
    public String getCedula() {

        return cedula;
    }
    2 usages
    public String getCedula() {

        return cedula;
    }

    1 usage
    public void setCedula(String cedula) {

        this.cedula = cedula;
    }

    1 usage
    public int getEdad() {

        return edad;
    }
    1 usage
    public boolean getEnfermedades() {

        return enfermedades;
    }
    2 usages
    public void setEnfermedades(boolean enfermedades) {

        this.enfermedades = enfermedades;
    }

    1 usage
    public void setEdad(int edad) {

        this.edad = edad;
    }
}
```

Imagen 6. Usuarios.java

Como se puede ver, un objeto de la clase Usuarios cuenta con los siguientes atributos: nombre, cedula, edad y enfermedades. Esta informción es útil para partes posteriores del proceso.

A continuación se verán tanto el paquete View como el paquete layout para entender el funcionamiento de cada activity.

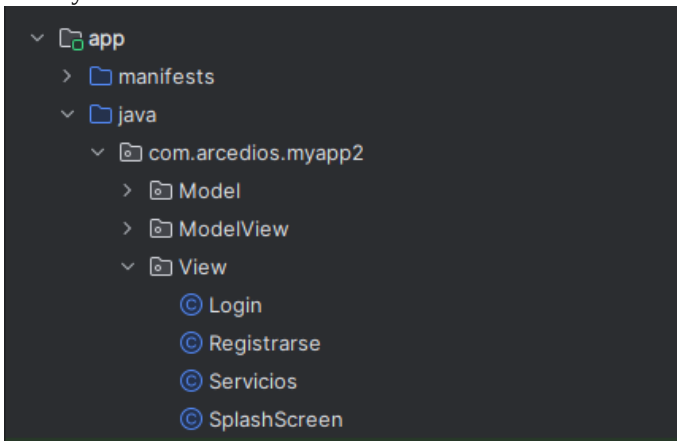


Imagen 7. Paquete View

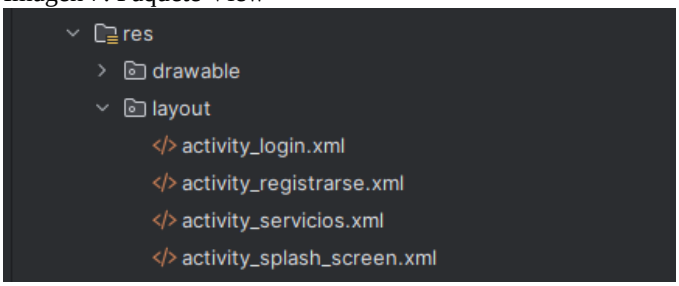


Imagen 8. Layout

En esta parte del código se genera todo lo que interactúa directamente con el usuario. Veamos primero el splashScreen:

Imagen 9. Vista del splashscreen

Aquí apenas se inicia la aplicción se muestra el logo de la aplicación junto con el slogan. Esto se genera a partir del xml:

```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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".View.SplashScreen">
    <ImageView
        android:id="@+id/logo"
        android:layout_width="308dp"
        android:layout_height="493dp"
        android:src="@drawable/logo1"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/Slogan"
        android:layout_width="300dp"
        android:layout_height="103dp"
        android:layout_marginBottom="130dp"
        android:text="No dejes que el sueño tome el volante "
        android:textAlignment="center"
        android:textSize="24sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.497"
        app:layout_constraintStart_toStartOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Imagen 10. Activity_splash_screen.xml

Finalmente pasamos al código de la activity:

```java
package com.arcedios.myapp2.View;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.util.Log;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import com.arcedios.myapp2.databinding.ActivitySplashScreenBinding;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

public class SplashScreen extends AppCompatActivity {

    2 usages
    private ActivitySplashScreenBinding binding;
    2 usages
    private DatabaseReference databaseReference;
    1 usage
    public static int tiempo = 5000;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivitySplashScreenBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());
        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {
                databaseReference = FirebaseDatabase.getInstance().getReference( path: "Usuarios");
                //Verificar si hay un usuario guardado
                checkUserLoggedIn();
            }
        }, tiempo);
    }

    private void checkUserLoggedIn() {
        databaseReference.addListenerForSingleValueEvent(new ValueEventListener() {
            2 usages
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                if (dataSnapshot.exists()) {
                    Intent intent = new Intent( packageContext: SplashScreen.this, Login.class);//aqui
                    startActivity(intent);
                    finish();
                }
                else {
                    Toast.makeText( context: SplashScreen.this, text: "No hay usuarios registrados", Toast.LENGTH_SHORT).show();
                    Intent intent = new Intent( packageContext: SplashScreen.this, Registrarse.class);
                    startActivity(intent);
                    finish();
                }
            }
            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {
                Log.e( tag: "MainActivity", msg: "Error al verificar usuario", databaseError.toException());
                Toast.makeText( context: SplashScreen.this, text: "Error al verificar usuario", Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

Imagen 11. SplashScreen

Aquí además de generar una pantalla llamativa para que el usuario, uso el metodo checkUserLogged() para determinar si el usuario ya está registrado y usando un intent lo mando a la pantalla que deba ver, ya sea la de registrarse o la de login. Pasamos a la pantalla de registro:

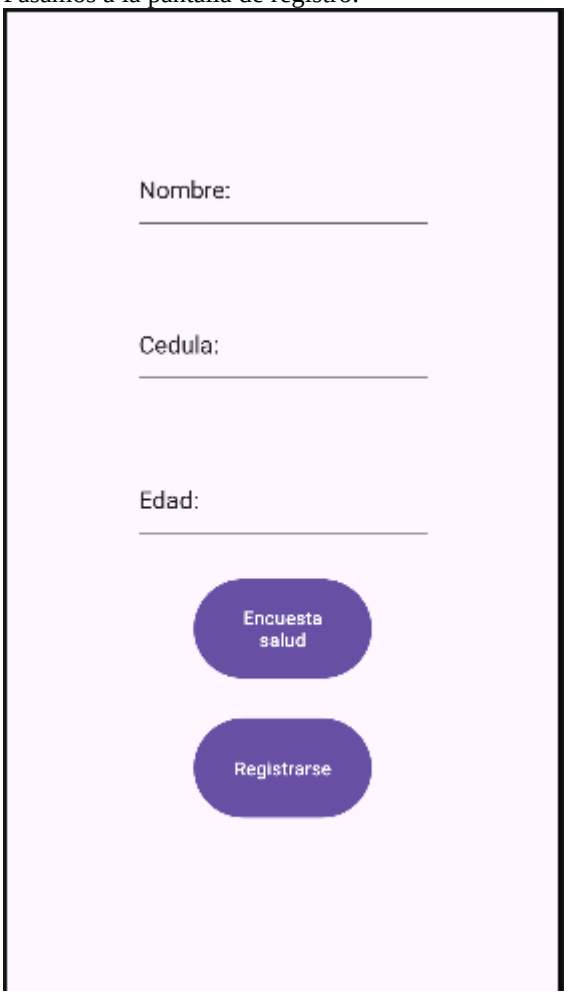


Imagen 12. Vista de registrarse

Aquí el usuario ingresa sus datos y presiona el botón de la encuesta de salud donde le aparecec la siguiente ventana:

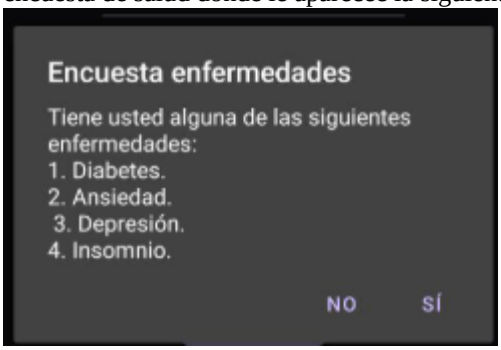


Imagen 13. Encuesta enfermedades

Finlalmente presiona el botón registrarse y pasa la siguiente pantalla.

```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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".View.Registrarse">
    <EditText
        android:id="@+id/nombre"
        android:layout_width="223dp"
        android:layout_height="66dp"
        android:layout_marginTop="100dp"
        android:ems="10"
        android:inputType="text"
        android:text="Nombre:"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <EditText
        android:id="@+id/cedula"
        android:layout_width="223dp"
        android:layout_height="66dp"
        android:layout_marginTop="50dp"
        android:ems="10"
        android:inputType="text"
        android:text="Cedula:"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/nombre" />

    <EditText
        android:id="@+id/edad"
        android:layout_width="223dp"
        android:layout_height="66dp"
        android:layout_marginTop="50dp"
        android:ems="10"
        android:inputType="text"
        android:text="Edad:"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/cedula" />

    <Button
        android:id="@+id/registro"
        android:layout_width="133dp"
        android:layout_height="82dp"
        android:text="Registrarse"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/edad" />

    <Button
        android:id="@+id/PreguntaSalud"
        android:layout_width="133dp"
        android:layout_height="82dp"
        android:text="Encuesta salud"
        app:layout_constraintBottom_toTopOf="@+id/registro"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/edad" />


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

Imagen 14. Activity_registrarse.xml

Ahora veamos el código:

```java
package com.arcedios.myapp2.View;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
import com.arcedios.myapp2.Model.TodoFirebase;
import com.arcedios.myapp2.databinding.ActivityRegistrarseBinding;
import com.arcedios.myapp2.ModelView.Usuarios;
public class Registrarse extends AppCompatActivity {
    7 usages
    private ActivityRegistrarseBinding binding;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityRegistrarseBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());
        Usuarios usuario = new Usuarios();
        binding.PreguntaSalud.setOnClickListener(view -> {
            new AlertDialog.Builder( context: this)
                    .setTitle("Encuesta enfermedades")
                    .setMessage("Tiene usted alguna de las siguientes enfermedades: \n" +
                            "1. Diabetes.\n" +
                            "2. Ansiedad.\n " +
                            "3. Depresion. \n" +
                            "4. Insomnio.")
                    .setPositiveButton( text: "Si", new DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialog, int which) {
                            Toast.makeText(getApplicationContext(), text: "aceptado", Toast.LENGTH_SHORT).show();
                            usuario.setEnFermedades(true);
                        }
                    })
                    .setNegativeButton( text: "No", new DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialog, int which) {
                            Toast.makeText(getApplicationContext(), text: "no aceptado", Toast.LENGTH_SHORT).show();

                            usuario.setEnFermedades(false);
                        }
                    })
                    .show();
        });
        binding.registro.setOnClickListener(view -> {
            String nombre = binding.nombre.getText().toString();
            String cedula = binding.cedula.getText().toString();
            int edad = Integer.parseInt(binding.edad.getText().toString());
            usuario.setNombre(nombre);
            usuario.setCedula(cedula);
            usuario.setEdad(edad);
            if (TextUtils.isEmpty(nombre) || TextUtils.isEmpty(cedula)) {
                Toast.makeText( context: this, text: "Por favor, complete todos los campos", Toast.LENGTH_SHORT).show();
                return;
            }
            boolean noC= TodoFirebase.addUser(usuario);
            if (noC) {
                Toast.makeText( context: this, text: "error al agregar el usuario", Toast.LENGTH_SHORT).show();
            } else {
                Toast.makeText( context: this, text: "Usuario agregado correctamente", Toast.LENGTH_SHORT).show();
            }
            Intent intent = new Intent( packageContext: Registrarse.this,Servicios.class);
            intent.putExtra( name: "NOMBRE", nombre); // Pass the user name
            intent.putExtra( name: "USER_ID", cedula); // Pass the user ID
            startActivity(intent);
            finish();
        });
    }
}
```

Imagen 15. Registrarse

Aquí basicamente se conectan los cuadros de texto con el boton registrarse de forma que al presionar el boton se genera el objeto de la clase usuario con los datos ingresados y se llama al metodo addUser() de la clase TodoFirebase para subir la información a la base de datos.

Por otro lado tenemos la pantalla login:

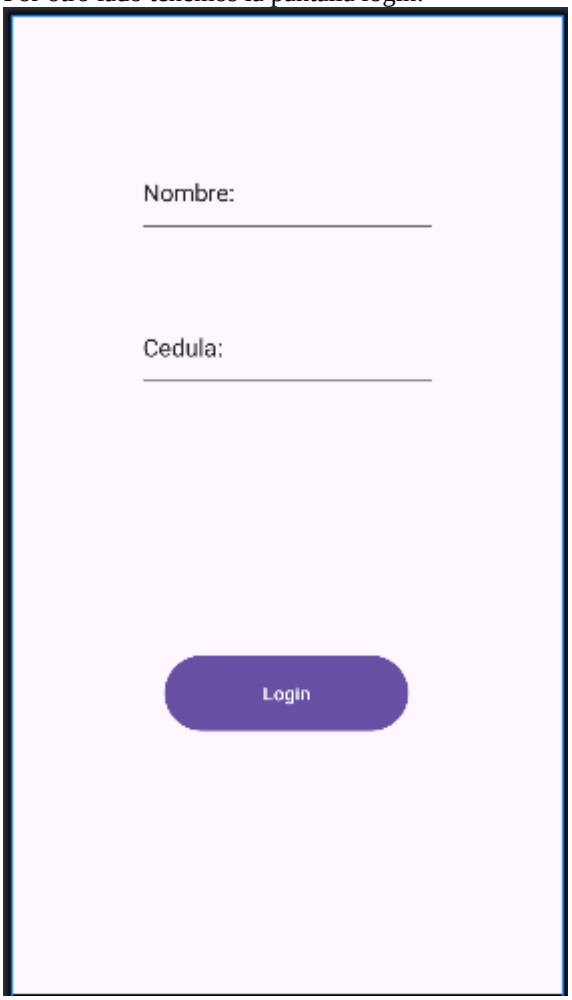


Imagen 16. Vista login

```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"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".View.Login">
    <EditText
        android:id="@+id/nombre"
        android:layout_width="223dp"
        android:layout_height="66dp"
        android:layout_marginTop="100dp"
        android:ems="10"
        android:inputType="text"
        android:text="Nombre:"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <EditText
        android:id="@+id/cedula"
        android:layout_width="223dp"
        android:layout_height="66dp"
        android:layout_marginTop="50dp"
        android:ems="10"
        android:inputType="text"
        android:text="Cedula:"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/nombre" />
    <Button
        android:id="@+id/login"
        android:layout_width="182dp"
        android:layout_height="64dp"
        android:text="Login"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/cedula" />
```

Imagen 17. activity_login.xml

```java
package com.arcedios.myapp2.View;
import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
import com.arcedios.myapp2.Model.TodoFirebase;
import com.arcedios.myapp2.databinding.ActivityLoginBinding;
public class Login extends AppCompatActivity {
    5 usages
    private ActivityLoginBinding binding;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityLoginBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());
        binding.login.setOnClickListener(view -> {
            String nombre = binding.nombre.getText().toString();
            String cedula = binding.cedula.getText().toString();
            if (TextUtils.isEmpty(nombre) || TextUtils.isEmpty(cedula)) {
                Toast.makeText( context: this,  text: "Por favor, complete todos los campos", Toast.LENGTH_SHORT).show();
                return;
            }
            TodoFirebase.Identificar(cedula, new TodoFirebase.IdentificarCallback() {
                @Override
                public void onIdentificarResult(String nombreFirebase) {
                    if (nombreFirebase == null) {
                        Toast.makeText( context: Login.this,  text: "Error al verificar usuario", Toast.LENGTH_SHORT).show();
                    } else if (nombreFirebase.equals("No existe")) {
                        Toast.makeText( context: Login.this,  text: "El usuario no existe", Toast.LENGTH_SHORT).show();
                        Intent intent = new Intent( packageContext: Login.this, Registrarse.class);
                        startActivity(intent);
                    } else if (nombreFirebase.equals(nombre)) {
                        Intent intent = new Intent( packageContext: Login.this, Servicios.class);
                        intent.putExtra( name: "USER_ID", cedula); // Pass the user ID
                        intent.putExtra( name: "NOMBRE", nombre); // Pass the user name
                        startActivity(intent);
                        finish();
                    } else {
                        Toast.makeText( context: Login.this,  text: "Error al verificar usuario. ya hay un usuario con esa cedula", Toast.LENGTH_SHORT).show();
                    }
                }
            });
        });
    }
}
```

Imagen 18. Login

Aquí el usuario ingresa sus datos y al presionar el boton se llama al metodo TodoFirebase.Identificar(cedula) para que compruebe si la información concuerda con la que está almacenada en la base de datos si es así lo manda a la siguiente pantalla. Sino le informa cuál es el fallo o lo manda a registrarse.

Finalmente se encuentra la pantalla principal de la aplicación que es tambíen en la que se concentra la mayor parte del funcionamiento del sistema.

Imagen 19. Vista servicios apagado

En la imagen 19 podemos observar lapantalla de servicios cuando no se ha iniciado el análisis o no se ha dado el permiso de usar la cámara.

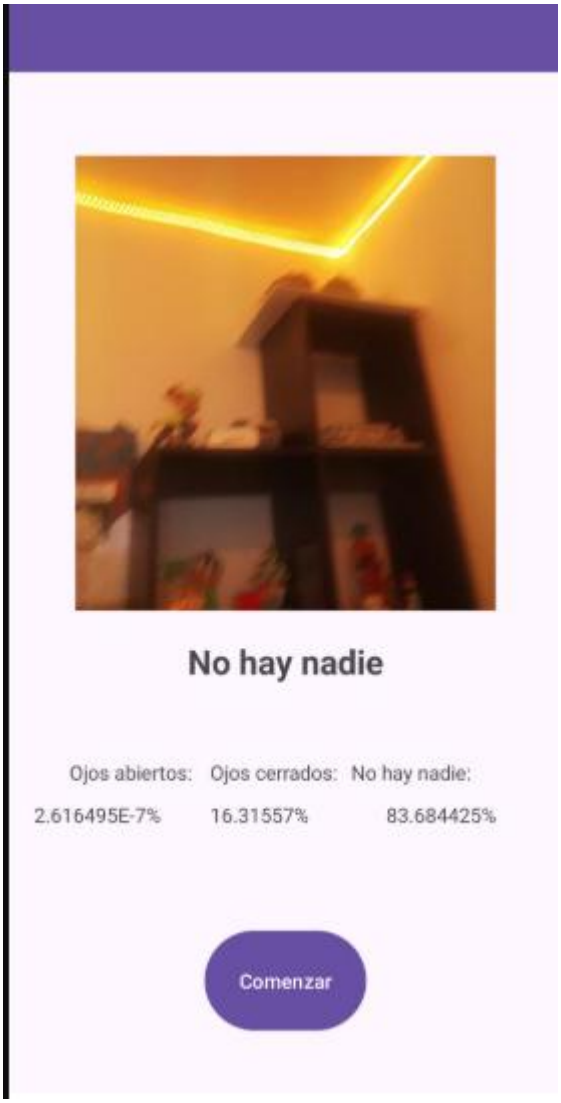


Imagen 20. Vista de servicios encendido

Una vez que el usuario da los permisos para usar la cámara y activa el análisis se abre la cámara frontal y comienza a comparar datos. En la parte inferior se muestra la probabilidad de cada uuna delas 3 opciones que tiene el mmodelo de tensorflow lite y arriba muestra la opción que es más probable.

```
<?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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".View.Servicios">
    <TextView
        android:id="@+id/noHayNadie2"
        android:layout_width="107dp"
        android:layout_height="32dp"
        android:layout_marginEnd="-250dp"
        android:layout_marginBottom="40dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/imageView" />
    <TextView
        android:id="@+id/ojosCerrados2"
        android:layout_width="107dp"
        android:layout_height="32dp"
        android:layout_marginBottom="40dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/imageView" />
    <TextView
        android:id="@+id/ojosAbiertos2"
        android:layout_width="107dp"
        android:layout_height="32dp"
        android:layout_marginStart="-250dp"
        android:layout_marginBottom="40dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/imageView" />
    <androidx.appcompat.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        android:background="?attr/colorPrimary"
        android:theme="@style/ThemeOverlay.AppCompat.ActionBar"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:popupTheme="@style/ThemeOverlay.AppCompat.Light" />
    <Button
        android:id="@+id/button2"
        android:layout_width="115dp"
        android:layout_height="79dp"
        android:layout_marginBottom="40dp"
        android:text="Comenzar"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />
    <ImageView
        android:id="@+id/imageView"
        android:layout_width="298dp"
        android:layout_height="323dp"
        android:layout_marginBottom="100dp"
        app:layout_constraintBottom_toTopOf="@+id/button2"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:srcCompat="@drawable/common_google_signin_btn_icon_dark_focused" />
    <androidx.camera.view.PreviewView
        android:id="@+id/camara"
        android:layout_width="298dp"
        android:layout_height="323dp"
        android:layout_marginBottom="100dp"
        app:layout_constraintBottom_toTopOf="@+id/button2"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        />
```

```
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/imageView" />
    <TextView
        android:id="@+id/OjosAbiertos"
        android:layout_width="107dp"
        android:layout_height="32dp"
        android:layout_marginStart="-200dp"
        android:layout_marginBottom="100dp"
        android:text="Ojos abiertos"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/imageView" />
    <TextView
        android:id="@+id/NoHayNadie"
        android:layout_width="107dp"
        android:layout_height="32dp"
        android:layout_marginEnd="-200dp"
        android:layout_marginBottom="100dp"
        android:text="No hay nadie"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/imageView" />
    <TextView
        android:id="@+id/OjosCerrados"
        android:layout_width="107dp"
        android:layout_height="32dp"
        android:layout_marginBottom="100dp"
        android:text="Ojos cerrados"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/imageView" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Imagen 21. activity_servicios.xml

El funcionamiento de esta pantalla es el siguiente: inicialmente se le pide al usuario que de el permiso para usar la cámara, en este caso se usa camerax. Posteriormente, el usuario presiona el boton y este activa un handler, est hace que se tomen fotos cada 2 segundos (ese tiempo se puede ajustar según la necesidad). Después, esa foto debe ser procesada para que tenga los 224x224 pixeles para que el modelo de tensorflow lite la pueda analizar, la aplicación va almacenando los resultados en un ArrayList y una vez que tiene 10 valores almacenados revisa cuantos de ellos dicen que la persona tiene los ojos cerrados, para estas pruebas se decidió que si 5 o más de esos 10 resultados dicen que tiene los ojos cerrados se debe identificar como que la persona se está quedando dormida. Despues de finalizado ese análisis se siguen tomando fotos igualmente cada 2 segundos así que se elimina la imagen más antigua y se almacena la nueva, manteniedo los 10 datos en total, así el proceso continua hasta que la persona vuelva a presionar el botón. En caso de que el análisis detecte que la persona se está quedando dormida el fondo de la pantalla comienza a cambiar de color entre rojo y azul cada 0.5 segundos, además, usando un TextToSpeech, se hace que la aplicación le hable a la persona diciendo el siguiente mensaje constantemente “nombre despierta” en mi caso dice “Daniel despierta”. Este cambio de colores y la voz se apagan una vez que la persona abre los ojos y la mayoria de los datos dan que los ojos están abiertos o si la persona apaga el modelo. Veamos el código:

```
package com.arcedios.myapp2.View;
import android.Manifest;
import android.content.pm.PackageManager;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Matrix;
import android.media.ExifInterface;
import android.os.Bundle;
import android.os.Environment;
import android.os.Handler;
import android.util.Log;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.camera.core.CameraSelector;
import androidx.camera.core.ImageCapture;
import androidx.camera.core.ImageCaptureException;
import androidx.camera.lifecycle.ProcessCameraProvider;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.lifecycle.LifecycleOwner;
import com.arcedios.myapp2.Model.Classifier;
import com.arcedios.myapp2.R;
import com.google.common.util.concurrent.ListenableFuture;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.ArrayList;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import android.speech.tts.TextToSpeech;
import java.util.Locale;

public class Servicios extends AppCompatActivity {
    2 usages
    private androidx.camera.view.PreviewView camara;
    4 usages
    private ImageCapture imageCapture;
    2 usages
    private ExecutorService cameraExecutor;
    2 usages
    private Button recognizeButton;
    2 usages
    private static final int REQUEST_CAMERA_PERMISSION = 100;
    2 usages
    private TextView resultTextView, OjosAbiertos, OjosCerrados, NoHayNadie;
    3 usages
    private Handler handler = new Handler();
    4 usages
    private boolean isAnalyzing = false; // Para controlar el estado del análisis
    6 usages
    private ArrayList<String> historialResultados = new ArrayList<>();
    4 usages
    private boolean isColorChanging = false; // Controla si el fondo está cambiando
    3 usages
    private Handler colorHandler = new Handler();
    3 usages
    private Runnable colorRunnable;
    7 usages
    private TextToSpeech textToSpeech;

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_servicios);
    textToSpeech = new TextToSpeech( context: this, status -> {
        if (status == TextToSpeech.SUCCESS) {
            textToSpeech.setLanguage(new Locale( language: "es", country: "US")); // Para español latino
        }
    });
    String userId = getIntent().getStringExtra( name: "USER_ID");
    camara = findViewById(R.id.camara);
    recognizeButton = findViewById(R.id.button2);
    resultTextView = findViewById(R.id.Resultado);
    OjosAbiertos = findViewById(R.id.ojosAbiertos2);
    OjosCerrados = findViewById(R.id.ojosCerrados2);
    NoHayNadie = findViewById(R.id.noHayNadie2);
    cameraExecutor = Executors.newSingleThreadExecutor();
    recognizeButton.setOnClickListener(v -> toggleAnalysis());
}
@Override
protected void onResume() {
    super.onResume();
    if (ContextCompat.checkSelfPermission( context: this, Manifest.permission.CAMERA) == PackageManager.PERMISSION_GRANTED) {
        startCamera();
    } else {
        ActivityCompat.requestPermissions( activity: this, new String[]{Manifest.permission.CAMERA}, REQUEST_CAMERA_PERMISSION);
    }
}
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    if (requestCode == REQUEST_CAMERA_PERMISSION) {
        if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            startCamera();
        } else {
            Toast.makeText( context: this, text: "Permiso de cámara denegado", Toast.LENGTH_SHORT).show();
        }
    }
}
```

Imagen 22. Inicio de la actividad

Aquí se revisan los permisos, se inician las variables necesarias y cuando sea posible se inicia la actividad una vez que se presiona el boton.

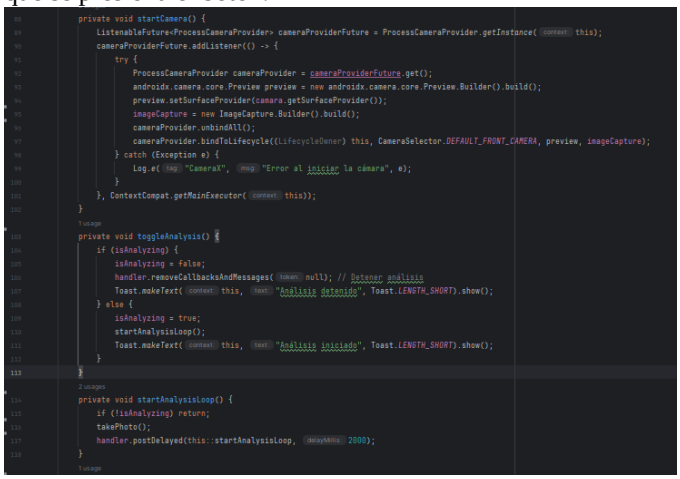


Imagen 23. Cámara y handler

Estos métodos se encargan de iniciar la cámara y de mantener el procesos iniciado hasta que el usuario vuelva a presionar el botón.

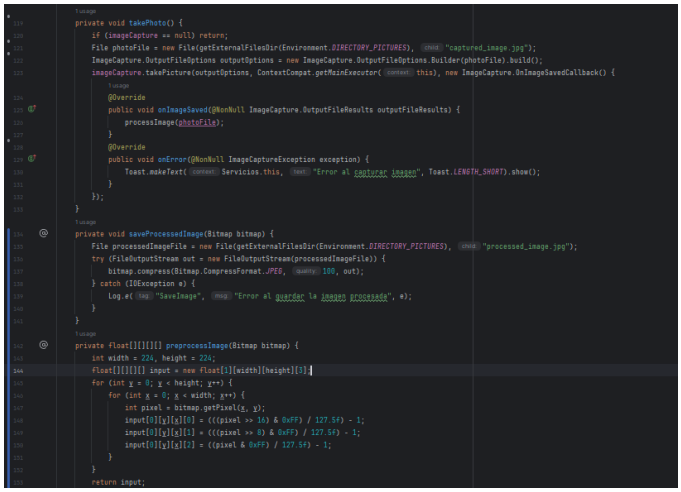


Imagen 24. Tomar foto

Aquí se toma la foto y se almace, depués comienza el procesado de la imagen.

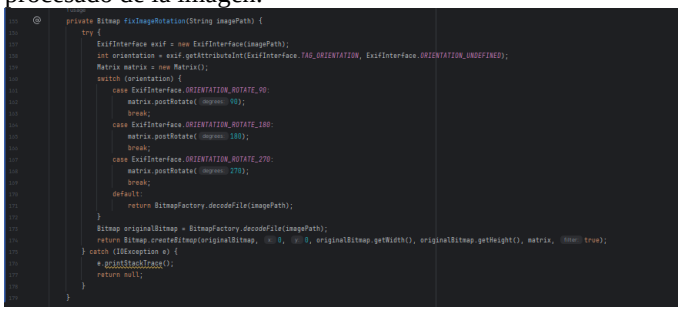


Imagen 25. Bitmap y ArrayList

El método FixImageRotation revisa que la imagen se encuentre en la posición correcta para que la vea el modelo si no lo está la rota para que quede correctamente, procesimage termina de preparar la imagen poniendola como un Bitmap de 224x224 pixeles y llama al modelo para que revise los resultados, además va almacenando los resultados en la lista para que analizarDatos detecte si la persona se está quedando dormida.

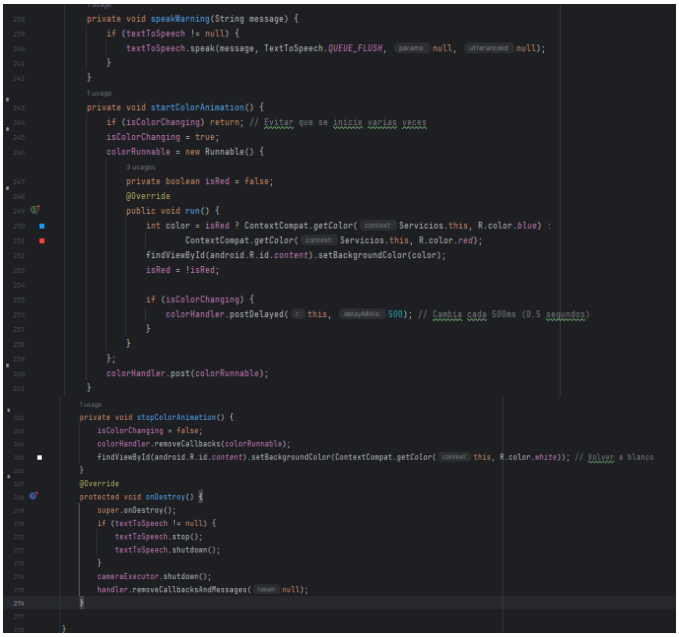


Imagen 26. Voz y colores

En caso de que analizarDatos determine que la persona se

queda dormida llama a speakWarning y startColorAnimation, los cuales controlan el aviso para tratar de despertar al conductor.

Para que toda la alpicación funcione correctamente se debe revisar que tanto el build.gradle a nivel de aplicación como el AndroidManifest tengan toda la información necesaria pues de estos depende el uso de firebase, el modelo de tensorflow y el acceso a la cámara entre otras cosas:

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <uses-feature
        android:name="android.hardware.camera"
        android:required="false" />

    <uses-permission android:name="android.permission.CAMERA" />

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="MyApp2"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.MyApp2"
        tools:targetApi="31">
        <activity
            android:name=".View.Registrarse"
            android:exported="false" />

        <activity
            android:name=".View.Login"
            android:exported="false" />
        <activity
            android:name=".View.Servicios"
            android:exported="true"
            android:theme="@style/Theme.MyApp2" />
        <activity
            android:name=".View.SplashScreen"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <provider
            android:name="androidx.core.content.FileProvider"
            android:authorities="com.arcedios.myapp2.fileprovider"
            android:grantUriPermissions="true"
            android:exported="false">
            <meta-data
                android:name="android.support.FILE_PROVIDER_PATHS"
                android:resource="@xml/file_paths" />
        </provider>
    </application>

</manifest>
```

Imagen 27. AndroidManifest

```
plugins {
    alias(libs.plugins.android.application)
    alias(libs.plugins.google.gms.google.services)
}

android {
    namespace = "com.arcedios.myapp2"
    compileSdk = 35

    defaultConfig {
        applicationId = "com.arcedios.myapp2"
        minSdk = 30
        targetSdk = 35
        versionCode = 1
        versionName = "1.0"
        renderscriptTargetApi = 21
        renderscriptSupportModeEnabled = true

        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            isMinifyEnabled = false
            proguardFiles(
                getDefaultProguardFile("proguard-android-optimize.txt"),
                "proguard-rules.pro"
            )
        }
    }
    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_11
        targetCompatibility = JavaVersion.VERSION_11
    }
    buildFeatures {
        viewBinding = true
        dataBinding = true
        mlModelBinding = true

    }
}

dependencies {
    implementation ("org.bytedeco:javacv:1.5.8")

    implementation(libs.appcompat)
    implementation(libs.material)
    implementation(libs.activity)
    implementation(libs.constraintlayout)
    implementation(libs.firebase.database)
    implementation("org.tensorflow:tensorflow-lite:2.9.0")
    implementation ("org.tensorflow:tensorflow-lite:2.9.0")
    implementation ("org.tensorflow:tensorflow-lite-metadata:0.4.3")
    implementation ("org.tensorflow:tensorflow-lite-support:0.4.3")
    testImplementation(libs.junit)
    androidTestImplementation(libs.ext.junit)
    androidTestImplementation(libs.espresso.core)
    implementation("androidx.camera:camera-core:1.3.0")
    implementation("androidx.camera:camera-camera2:1.3.0")
    implementation("androidx.camera:camera-lifecycle:1.3.0")
    implementation("androidx.camera:camera-view:1.3.0")
}
```

Imagen 28. Build.gradle app

III. EL MODELO TENSORFLOW LITE

Para generar el modelo entrenado usé la página: https://teachablemachine.withgoogle.com/train/image

Esta permite subir las imágenes para cada clase que el modelo deba reconocer las usa para entrenar el modelo.

Imagen 29. Teacheble Machine

Una vez que el modelo se encuentra listo se debe escoger la opción de tensorflow lite que es la que se puede usar para android studio. Automáticamente se generan 2 archivos, uno llamado model.tflite y otro en .txt, el segundo solo sirve para recordar el orden de las clases. Finalmente, solo se debe pasar el archivo model.tflite a la carpeta assets y usar el código que

tengo en Classifier.java.

IV. DIFICULTADES

Metadata: Al buscar información en internet sobre cómo integrar el modelo de tensorflow lite en android studio se hablaba mucho de la necesidad de tener metadata en el archivo model.tflite, viendo que el mío no lo tenía pasé varios días intentando agregar esa opción al archivo pero finalmente no pude hacerlo, sin embargo, con el método descrito a lo largo de este documento el proceso fue mucho más sencillo.

Procesado de imagen: Al tener mi modelo cargado y funcional la siguiente dificultad fue preparar la imagen para que se pudiera analizar por el modelo. Una vez que tenía la imagen como el bitmap con los pixeles correctos el modelo era muy aleatorio, daba resultados incorrectos y no lograba encontrar el error. Para solucionar este error comencé a guardar la imagen que había tomado la cámara y la imagen después del procesado y ahí pude ver que la imagen después de procesada estaba girada 90 grados en todos los casos, por eso hice el método FixImageRotation con el cuál la imagen ya queda con la orientación original y el modelo comenzó a leerla correctamente.

V. ASPECTOS PARA MEJORAR

1. Aunque Teacheble Machine es una herramienta muy práctica y fácil de entender, me parece que mi modelo no terminó de ser tan preciso como me hubiera gustado, aunque agregué más de mil fotos diferentes para cada tipo de caso. La efectividad del modelo sigue dependiendo mucho de la cercanía de la persona a la cámara o del tamaño de los ojos por lo que si quisiera lanzar esta aplicación realmente debería buscar otra forma de entrenar el modelo.
2. En mis planes iniciales estaba la idea de implementar que si el usuario cuenta con alguna de las enfermedades listadas el filtro para detectar si se queda dormido sea más sensible, sin embargo, por cuestiones de tiempo no lo alcancé a hacer.
3. Pensando en una aplicación realmente útil para conductores alrededor del mundo sería interesante hacer que la aplicación sea completamente funcional sin acceso a internet.

VI. CONCLUSIÓN

Haciendo un análisis global de la aplicación, considero que el trabajo realizado fue satisfactorio teniendo en cuenta que la aplicación es capaz de identificar si la persona se está quedando dormida además de que tiene 2 formas de avisar a la persona y tratar de despertarla, una que cambia de colores y otra que directamente le habla a la persona y lo llama por su nombre lo cual sería muy útil para advertirle del peligro. Como proyecto considero que la aplicación tiene muchas más cosas que se podrían llevar a cabo a futuro en las que espero continuar trabajando más adelante. Como pensamiento personal considero que el trabajo fue muy divertido y aprendí mucho.