

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.

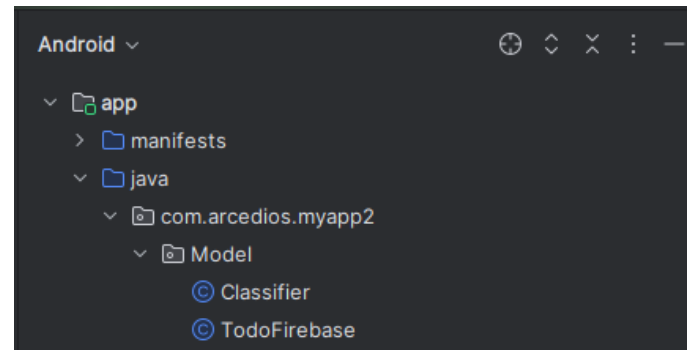


Imagen 1. Model

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

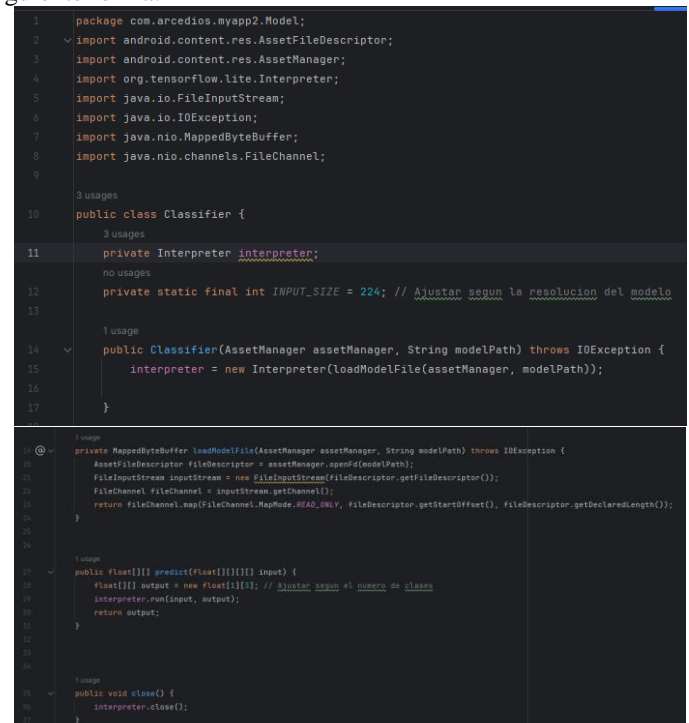


Imagen 2. Clase Classifier.java

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.

```

62
63 @Override
64 public void onCancelled(@NonNull DatabaseError error) {
65     Log.e(TAG, "TodoFirebase", msg: "onCancelled called: " + error.getMessage());
66     callback.onIdentificarResult(nombre: null); // Handle the error case
67 }
68
69
70
71
72 }

```

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

Después tengo la clase Todofirebse.java:

```

1 package com.arcedios.myapp2.Model;
2
3 import android.util.Log;
4
5 import androidx.annotation.NonNull;
6
7 import com.google.firebase.database.DataSnapshot;
8 import com.google.firebase.database.DatabaseError;
9 import com.google.firebase.database.DatabaseReference;
10 import com.google.firebase.database.FirebaseDatabase;
11 import com.google.firebase.database.ValueEventListener;
12
13 import java.util.HashMap;
14 import java.util.concurrent.atomic.AtomicBoolean;
15
16 import com.arcedios.myapp2.ModelView.Usuarios;
17
18 // usages
19 public class TodoFirebase {
20     no usages
21     public boolean aver;
22
23     1 usage
24     private static DatabaseReference databaseReference = null;
25
26
27 public TodoFirebase() {
28     // Initialize the database reference in the constructor
29     databaseReference = FirebaseDatabase.getInstance().getReference();
30 }
31
32 @
33 public static boolean addUser(Usuarios p) {
34     AtomicBoolean success = new AtomicBoolean(false);
35     TodoFirebase firebase = new TodoFirebase();
36     FirebaseDatabase database = FirebaseDatabase.getInstance();
37     DatabaseReference myRef = database.getReference();
38     HashMap<String, Object> map = new HashMap<>();
39     map.put("nombre", p.getNombre());
40     map.put("cedula", p.getCedula());
41     map.put("edad", p.getEdad());
42     map.put("enfermedades", p.getEnfermedades());
43     myRef.child("Usuarios").child(p.getCedula()).setValue(map).addOnCompleteListener(task -> {
44         if (task.isSuccessful()) {
45             success.set(true);
46         }
47     });
48     return success.get();
49 }
50
51 // Implementation
52 public interface IdentificarCallback {
53     // Implementation
54     void onIdentificarResult(String nombre);
55 }
56
57
58 public static void identificar(String cedula, IdentificarCallback callback) {
59     Log.d(TAG, "TodoFirebase", msg: "Identificar called with cedula: " + cedula);
60     if (cedula == null || cedula.isEmpty()) {
61         Log.d(TAG, "TodoFirebase", msg: "Cedula is null or empty");
62         callback.onIdentificarResult(nombre: null); // Indicate an error
63         return;
64     }
65
66     FirebaseDatabase database = FirebaseDatabase.getInstance();
67     DatabaseReference myRef = database.getReference("Usuarios");
68
69     myRef.child(cedula).addListenerForSingleValueEvent(new ValueEventListener() {
70         @Override
71         public void onDataChange(@NonNull DataSnapshot snapshot) {
72             Log.d(TAG, "TodoFirebase", msg: "onDataChange called");
73             if (snapshot.exists()) {
74                 Log.d(TAG, "TodoFirebase", msg: "snapshot exists");
75                 Usuarios p = snapshot.getValue(Usuarios.class);
76                 if (p != null) {
77                     Log.d(TAG, "TodoFirebase", msg: "User found: " + p.getNombre());
78                     callback.onIdentificarResult(p.getNombre());
79                 } else {
80                     Log.d(TAG, "TodoFirebase", msg: "User is null");
81                     callback.onIdentificarResult(nombre: null); // Handle the case where p is null
82                 }
83             } else {
84                 Log.d(TAG, "TodoFirebase", msg: "snapshot does not exist");
85                 callback.onIdentificarResult(nombre: "No existe"); // Handle the case where the snapshot doesn't exist
86             }
87         }
88     });
89 }

```

forma:

```

https://loginprueba-ff0f2-default-rtdb.firebaseio.com/
└── Usuarios
    └── 1006873522
        ├── cedula: "1006873522"
        ├── edad: 22
        ├── enfermedades: true
        └── nombre: "Daniel"

```

Imagen 4. Ejemplo base de datos
Pasando al paquete ModelView:

```

Android
└── app
    ├── manifests
    ├── java
    └── com.arcedios.myapp2
        ├── Model
        ├── ModelView
        └── Usuarios

```

Imagen 5. Modelview

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

```

1 package com.arcedios.myapp2.ModelView;
2
3 public class Usuarios {
4     private String nombre;
5     private String cedula;
6     private int edad;
7     private boolean enfermedades;
8     public Usuarios() {
9     }
10    public String getNombre() {
11    }
12    return nombre;
13    }
14
15    public void setNombre(String nombre) {
16    }
17    this.nombre = nombre;
18    }
19
20    public String getCedula() {
21    }
22    return cedula;
23    }
24
25    public void setCedula(String cedula) {
26    }
27    this.cedula = cedula;
28    }
29
30    public int getEdad() {
31    }
32    return edad;
33    }
34
35    public boolean getEnfermedades() {
36    }
37    return enfermedades;
38    }
39
40    public void setEnfermedades(boolean enfermedades) {
41    }
42    this.enfermedades = enfermedades;
43    }
44
45    public void setEdad(int edad) {
46    }
47    this.edad = edad;
48    }
49 }

```

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 información 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.

```

1 app
2 manifests
3 java
4   com.arcedios.myapp2
5     Model
6     ModelView
7     View
8       Login
9       Registrarse
10      Servicios
11      SplashScreen

```

Imagen 7. Paquete View

```

1 res
2   drawable
3   layout
4     activity_login.xml
5     activity_registrarse.xml
6     activity_servicios.xml
7     activity_splash_screen.xml

```

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

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:id="@+id/main"
6      android:layout_width="match_parent"
7      android:layout_height="match_parent"
8      tools:context=".View.SplashScreen">
9      <ImageView
10         android:id="@+id/logo"
11         android:layout_width="308dp"
12         android:layout_height="493dp"
13         android:src="@drawable/logo1"
14         app:layout_constraintBottom_toBottomOf="parent"
15         app:layout_constraintEnd_toEndOf="parent"
16         app:layout_constraintStart_toStartOf="parent"
17         app:layout_constraintTop_toTopOf="parent" />
18
19      <TextView
20         android:id="@+id/Slogan"
21         android:layout_width="308dp"
22         android:layout_height="103dp"
23         android:layout_marginBottom="130dp"
24         android:text="No dejes que el sueño tome el volante."
25         android:textAlignment="center"
26         android:textSize="24sp"
27         android:textStyle="bold"
28         app:layout_constraintBottom_toBottomOf="parent"
29         app:layout_constraintEnd_toEndOf="parent"
30         app:layout_constraintHorizontal_bias="0.497"
31         app:layout_constraintStart_toStartOf="parent" />
32  </androidx.constraintlayout.widget.ConstraintLayout>

```

Imagen 10. Activity_splash_screen.xml

Finalmente pasamos al código de la actividad:

```

1  package com.arcedios.mypapp2.View;
2  import android.content.Intent;
3  import android.os.Bundle;
4  import android.os.Handler;
5  import android.util.Log;
6  import android.widget.Toast;
7  import androidx.annotation.NonNull;
8  import androidx.appcompat.app.AppCompatActivity;
9  import com.arcedios.mypapp2.databinding.ActivitySplashScreenBinding;
10 import com.google.firebase.database.DataSnapshot;
11 import com.google.firebase.database.DatabaseError;
12 import com.google.firebase.database.DatabaseReference;
13 import com.google.firebase.database.FirebaseDatabase;
14 import com.google.firebase.database.ValueEventListener;
15
16 public class SplashScreen extends AppCompatActivity {
17     // Usages
18     private ActivitySplashScreenBinding binding;
19     // Usages
20     private DatabaseReference databaseReference;
21     // Usage
22     public static int tiempo = 5000;
23     @Override
24     protected void onCreate(Bundle savedInstanceState) {
25         super.onCreate(savedInstanceState);
26         binding = ActivitySplashScreenBinding.inflate(getLayoutInflater());
27         setContentView(binding.getRoot());
28         new Handler().postDelayed(new Runnable() {
29             @Override
30             public void run() {
31                 databaseReference = FirebaseDatabase.getInstance().getReference("users");
32                 // Verificar si hay un usuario guardado
33                 checkUserLoggedIn();
34             }
35         }, tiempo);
36
37     private void checkUserLoggedIn() {
38         databaseReference.addValueEventListener(new ValueEventListener() {
39             // Usages
40             @Override
41             public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
42                 if (dataSnapshot.exists()) {
43                     Intent intent = new Intent(SplashScreen.this, Login.class);
44                     startActivity(intent);
45                     finish();
46                 } else {
47                     Toast.makeText(SplashScreen.this, "No hay usuarios registrados", Toast.LENGTH_SHORT).show();
48                     Intent intent = new Intent(SplashScreen.this, Registrarse.class);
49                     startActivity(intent);
50                     finish();
51                 }
52             }
53             @Override
54             public void onCancelled(@NonNull DatabaseError databaseError) {
55                 Log.e("MainActivity", "Error al verificar usuario", databaseError.toException());
56                 Toast.makeText(SplashScreen.this, "Error al verificar usuario", Toast.LENGTH_SHORT).show();
57             }
58         });
59     }
60 }

```

Imagen 11. SplashScreen

Aquí además de generar una pantalla llamativa para que el usuario, uso el metodo checkUserLoggedIn() 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 aparece la siguiente ventana:

Imagen 13. Encuesta enfermedades

Finalmente presiona el botón registrarse y pasa la siguiente pantalla.

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:id="@+id/main"
6     android:layout_width="match_parent"
7     android:layout_height="match_parent"
8     tools:context=".View.Registrarse">
9     <EditText
10         android:id="@+id/nombre"
11         android:layout_width="223dp"
12         android:layout_height="44dp"
13         android:layout_marginTop="100dp"
14         android:ems="10"
15         android:inputType="text"
16         android:text="Nombre"
17         app:layout_constraintEnd_toEndOf="parent"
18         app:layout_constraintStart_toStartOf="parent"
19         app:layout_constraintTop_toTopOf="parent" />
20
21     <EditText
22         android:id="@+id/cedula"
23         android:layout_width="223dp"
24         android:layout_height="44dp"
25         android:layout_marginTop="50dp"
26         android:ems="10"
27         android:inputType="text"
28         android:text="Cedula"
29         app:layout_constraintEnd_toEndOf="parent"
30         app:layout_constraintStart_toStartOf="parent"
31         app:layout_constraintTop_toBottomOf="@+id/nombre" />
32
33     <EditText
34         android:id="@+id/edad"
35         android:layout_width="223dp"
36         android:layout_height="44dp"
37         android:layout_marginTop="50dp"
38         android:ems="10"
39         android:inputType="text"
40         android:text="Edad"
41         app:layout_constraintEnd_toEndOf="parent"
42         app:layout_constraintStart_toStartOf="parent"
43         app:layout_constraintTop_toBottomOf="@+id/cedula" />
44
45     <Button
46         android:id="@+id/registrarse"
47         android:layout_width="133dp"
48         android:layout_height="82dp"
49         android:text="Registrarse"
50         app:layout_constraintBottom_toBottomOf="parent"
51         app:layout_constraintEnd_toEndOf="parent"
52         app:layout_constraintStart_toStartOf="parent"
53         app:layout_constraintTop_toBottomOf="@+id/edad" />
54
55     <Button
56         android:id="@+id/PreguntaSalud"
57         android:layout_width="133dp"
58         android:layout_height="82dp"
59         android:text="Encuesta salud"
60         app:layout_constraintBottom_toBottomOf="parent"
61         app:layout_constraintEnd_toEndOf="parent"
62         app:layout_constraintStart_toStartOf="parent"
63         app:layout_constraintTop_toBottomOf="@+id/edad" />
64
65 </androidx.constraintlayout.widget.ConstraintLayout>

```

Imagen 14. Activity_registrarse.xml

Ahora veamos el código:

```

1 package com.arcedios.mypapp2.View;
2 import android.app.AlertDialog;
3 import android.content.DialogInterface;
4 import android.content.Intent;
5 import android.os.Bundle;
6 import android.text.TextUtils;
7 import android.widget.Toast;
8
9 import androidx.appcompat.app.AppCompatActivity;
10 import com.arcedios.mypapp2.Model.TodoFirebase;
11 import com.arcedios.mypapp2.databinding.ActivityRegistrarseBinding;
12 import com.arcedios.mypapp2.ModelView.Userarios;
13
14 public class Registrarse extends AppCompatActivity {
15     //usages
16     private ActivityRegistrarseBinding binding;
17     @Override
18     protected void onCreate(Bundle savedInstanceState) {
19         super.onCreate(savedInstanceState);
20         binding = ActivityRegistrarseBinding.inflate(getLayoutInflater());
21         setContentView(binding.getRoot());
22         Userarios usuario = new Userarios();
23         binding.PreguntaSalud.setOnClickListener(view -> {
24             new AlertDialog.Builder(this)
25                 .setTitle("Consente informaciones")
26                 .setMessage("¿Desea usted alguno de las siguientes enfermedades? \n" +
27                     "1. Diabetes \n" +
28                     "2. Asma \n" +
29                     "3. Depresión \n" +
30                     "4. Insomnio.")
31                 .setPositiveButton("SI", new DialogInterface.OnClickListener() {
32                     @Override
33                     public void onClick(DialogInterface dialog, int which) {
34                         Toast.makeText(getApplicationContext(), "No aceptado", Toast.LENGTH_SHORT).show();
35                         usuario.setEnfermedades(true);
36                     }
37                 })
38                 .setNegativeButton("NO", new DialogInterface.OnClickListener() {
39                     @Override
40                     public void onClick(DialogInterface dialog, int which) {
41                         Toast.makeText(getApplicationContext(), "No aceptado", Toast.LENGTH_SHORT).show();
42                         usuario.setEnfermedades(false);
43                     }
44                 })
45                 .show();
46         });
47         binding.registro.setOnClickListener(view -> {
48             String nombre = binding.nombre.getText().toString();
49             String cedula = binding.cedula.getText().toString();
50             int edad = Integer.parseInt(binding.edad.getText().toString());
51             usuario.setNombre(nombre);
52             usuario.setCedula(cedula);
53             usuario.setEdad(edad);
54             if (TextUtils.isEmpty(nombre) || TextUtils.isEmpty(cedula)) {
55                 Toast.makeText(this, "Por Favor, complete todos los campos", Toast.LENGTH_SHORT).show();
56                 return;
57             }
58             boolean noC = TodoFirebase.addUser(usuario);
59             if (noC) {
60                 Toast.makeText(this, "Error al agregar el usuario", Toast.LENGTH_SHORT).show();
61             } else {
62                 Toast.makeText(this, "Usuario agregado correctamente", Toast.LENGTH_SHORT).show();
63             }
64             Intent intent = new Intent(this, Servicios.class);
65             intent.putExtra("NOMBRE", nombre); // Pass the user name
66             intent.putExtra("USER_ID", cedula); // Pass the user ID
67             startActivity(intent);
68             finish();
69         });
70     }
71 }

```

```

18 .setNegativeButton("NO", new DialogInterface.OnClickListener() {
19     @Override
20     public void onClick(DialogInterface dialog, int which) {
21         Toast.makeText(getApplicationContext(), "No aceptado", Toast.LENGTH_SHORT).show();
22         usuario.setEnfermedades(false);
23     }
24 })
25 .show();
26 });
27
28 binding.registro.setOnClickListener(view -> {
29     String nombre = binding.nombre.getText().toString();
30     String cedula = binding.cedula.getText().toString();
31     int edad = Integer.parseInt(binding.edad.getText().toString());
32     usuario.setNombre(nombre);
33     usuario.setCedula(cedula);
34     usuario.setEdad(edad);
35     if (TextUtils.isEmpty(nombre) || TextUtils.isEmpty(cedula)) {
36         Toast.makeText(this, "Por Favor, complete todos los campos", Toast.LENGTH_SHORT).show();
37         return;
38     }
39     boolean noC = TodoFirebase.addUser(usuario);
40     if (noC) {
41         Toast.makeText(this, "Error al agregar el usuario", Toast.LENGTH_SHORT).show();
42     } else {
43         Toast.makeText(this, "Usuario agregado correctamente", Toast.LENGTH_SHORT).show();
44     }
45     Intent intent = new Intent(this, Servicios.class);
46     intent.putExtra("NOMBRE", nombre); // Pass the user name
47     intent.putExtra("USER_ID", cedula); // Pass the user ID
48     startActivity(intent);
49     finish();
50 });
51 }

```

Imagen 15. Registrarse

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

Por otro lado tenemos la pantalla login:

Nombre:

Cedula:

Login

Imagen 16. Vista login

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout
3      xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:app="http://schemas.android.com/apk/res-auto"
5      xmlns:tools="http://schemas.android.com/tools"
6      android:id="@+id/main"
7      android:layout_width="match_parent"
8      android:layout_height="match_parent"
9      tools:context=".View.Login">
10     <EditText
11         android:id="@+id/nombre"
12         android:layout_width="223dp"
13         android:layout_height="66dp"
14         android:layout_marginTop="100dp"
15         android:ems="10"
16         android:inputType="text"
17         android:text="Nombre"
18         app:layout_constraintEnd_toEndOf="parent"
19         app:layout_constraintStart_toStartOf="parent"
20         app:layout_constraintTop_toTopOf="parent" />
21     <EditText
22         android:id="@+id/cedula"
23         android:layout_width="223dp"
24         android:layout_height="66dp"
25         android:layout_marginTop="50dp"
26         android:ems="10"
27         android:inputType="text"
28         android:text="Cedula"
29         app:layout_constraintEnd_toEndOf="parent"
30         app:layout_constraintStart_toStartOf="parent"
31         app:layout_constraintTop_toBottomOf="@+id/nombre" />
32     <Button
33         android:id="@+id/login"
34         android:layout_width="182dp"
35         android:layout_height="64dp"
36         android:text="Login"
37         app:layout_constraintBottom_toBottomOf="parent"
38         app:layout_constraintEnd_toEndOf="parent"
39         app:layout_constraintStart_toStartOf="parent"
40         app:layout_constraintTop_toBottomOf="@+id/cedula" />

```

Imagen 17. activity_login.xml

```

1  package com.arcedios.mypapp;
2  import android.content.Intent;
3  import android.os.Bundle;
4  import android.text.TextUtils;
5  import android.widget.Toast;
6  import androidx.appcompat.app.AppCompatActivity;
7  import com.arcedios.mypapp.Model.TodoFirebase;
8  import com.arcedios.mypapp.databinding.ActivityLoginBinding;
9  public class Login extends AppCompatActivity {
10     @Override
11     protected void onCreate(Bundle savedInstanceState) {
12         super.onCreate(savedInstanceState);
13         binding = ActivityLoginBinding.inflate(getLayoutInflater());
14         setContentView(binding.getRoot());
15         binding.login.setOnClickListener(view -> {
16             String nombre = binding.nombre.getText().toString();
17             String cedula = binding.cedula.getText().toString();
18             if (TextUtils.isEmpty(nombre) || TextUtils.isEmpty(cedula)) {
19                 Toast.makeText(this, "Por favor, complete todos los campos", Toast.LENGTH_SHORT).show();
20                 return;
21             }
22             TodoFirebase.Identificar(cedula, new TodoFirebase.IdentificarCallback() {
23                 @Override
24                 public void onIdentificarResult(String response) {
25                     if (response == null) {
26                         Toast.makeText(this, "Error al verificar usuario", Toast.LENGTH_SHORT).show();
27                     } else if (response.equals("no existe")) {
28                         Toast.makeText(this, "El usuario no existe", Toast.LENGTH_SHORT).show();
29                         Intent intent = new Intent(Login.this, Registrar.class);
30                         startActivity(intent);
31                     } else if (response.equals("existe")) {
32                         Intent intent = new Intent(Login.this, Services.class);
33                         intent.putExtra("user_id", response); // Pass the user id
34                         startActivity(intent);
35                         finish();
36                     } else {
37                         Toast.makeText(this, "Error al verificar usuario, ya fue verificado con esa cedula", Toast.LENGTH_SHORT).show();
38                     }
39                 }
40             });
41         });
42     }
43 }

```

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 también en la que se concentra la mayor parte del funcionamiento del sistema.



Imagen 19. Vista servicios apagado

En la imagen 19 podemos observar la pantalla 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 una de las 3 opciones que tiene el modelo de tensorflow lite y arriba muestra la opción que es más probable.

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:id="@+id/main"
6      android:layout_width="match_parent"
7      android:layout_height="match_parent"
8      tools:context=".View.Servicios">
9
10     <TextView
11         android:id="@+id/noHayNadie2"
12         android:layout_width="107dp"
13         android:layout_height="32dp"
14         android:layout_marginEnd="-250dp"
15         android:layout_marginBottom="40dp"
16         app:layout_constraintBottom_toBottomOf="parent"
17         app:layout_constraintEnd_toEndOf="parent"
18         app:layout_constraintStart_toStartOf="parent"
19         app:layout_constraintTop_toBottomOf="@+id/imageView" />
20
21     <TextView
22         android:id="@+id/glasCerrado2"
23         android:layout_width="107dp"
24         android:layout_height="32dp"
25         android:layout_marginBottom="40dp"
26         app:layout_constraintBottom_toBottomOf="parent"
27         app:layout_constraintEnd_toEndOf="parent"
28         app:layout_constraintStart_toStartOf="parent"
29         app:layout_constraintTop_toBottomOf="@+id/imageView" />
30
31     <TextView
32         android:id="@+id/glasAbierto2"
33         android:layout_width="107dp"
34         android:layout_height="32dp"
35         android:layout_marginStart="-250dp"
36         android:layout_marginBottom="40dp"
37         app:layout_constraintBottom_toBottomOf="parent"
38         app:layout_constraintEnd_toEndOf="parent"
39         app:layout_constraintStart_toStartOf="parent"
40         app:layout_constraintTop_toBottomOf="@+id/imageView" />
41
42     <androidx.appcompat.widget.Toolbar
43         android:id="@+id/toolbar"
44         android:layout_width="match_parent"
45         android:layout_height="?attr/actionBarSize"
46         android:background="?attr/colorPrimary"
47         android:theme="@style/ThemeOverlay.AppCompat.ActionBar"
48         app:layout_constraintEnd_toEndOf="parent"
49         app:layout_constraintStart_toStartOf="parent"
50         app:layout_constraintTop_toTopOf="parent"
51         app:popupTheme="@style/ThemeOverlay.AppCompat.Light" />
52
53     <Button
54         android:id="@+id/button2"
55         android:layout_width="115dp"
56         android:layout_height="79dp"
57         android:layout_marginBottom="40dp"
58         android:text="Comenzar"
59         app:layout_constraintBottom_toBottomOf="parent"
60         app:layout_constraintEnd_toEndOf="parent"
61         app:layout_constraintStart_toStartOf="parent"
62         app:layout_constraintTop_toTopOf="parent"
63         tools:srcCompat="@drawable/common_google_signin_btn_icon_dark_focused" />
64
65     <androidx.camera.view.PreviewView
66         android:id="@+id/camera"
67         android:layout_width="298dp"
68         android:layout_height="323dp"
69         android:layout_marginBottom="100dp"
70         app:layout_constraintBottom_toBottomOf="@+id/button2"
71         app:layout_constraintEnd_toEndOf="parent"
72         app:layout_constraintStart_toStartOf="parent"
73         app:layout_constraintTop_toTopOf="parent"
74         />

```



```

86 app:layout_constraintEnd_toEndOf="parent"
87 app:layout_constraintStart_toStartOf="parent"
88 app:layout_constraintTop_toBottomOf="@+id/imageView" />
89
90 <TextView
91     android:id="@+id/OjosAbiertos"
92     android:layout_width="107dp"
93     android:layout_height="32dp"
94     android:layout_marginStart="-200dp"
95     android:layout_marginBottom="100dp"
96     android:text="Ojos abiertos"
97     app:layout_constraintBottom_toBottomOf="parent"
98     app:layout_constraintEnd_toEndOf="parent"
99     app:layout_constraintStart_toStartOf="parent"
100     app:layout_constraintTop_toBottomOf="@+id/imageView" />
101
102 <TextView
103     android:id="@+id/NoHayNadie"
104     android:layout_width="107dp"
105     android:layout_height="32dp"
106     android:layout_marginEnd="-200dp"
107     android:layout_marginBottom="100dp"
108     android:text="No hay nadie"
109     app:layout_constraintBottom_toBottomOf="parent"
110     app:layout_constraintEnd_toEndOf="parent"
111     app:layout_constraintStart_toStartOf="parent"
112     app:layout_constraintTop_toBottomOf="@+id/imageView" />
113
114 <TextView
115     android:id="@+id/OjosCerrados"
116     android:layout_width="107dp"
117     android:layout_height="32dp"
118     android:layout_marginBottom="100dp"
119     android:text="Ojos cerrados"
120     app:layout_constraintBottom_toBottomOf="parent"
121     app:layout_constraintEnd_toEndOf="parent"
122     app:layout_constraintStart_toStartOf="parent"
123     app:layout_constraintTop_toBottomOf="@+id/imageView" />
124
125 </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. Después 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, manteniendo 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 mayoría de los datos dan que los ojos están abiertos o si la persona apaga el modelo. Veamos el código:

```

1 package com.arcedios.myapplication;
2 import android.Manifest;
3 import android.content.pm.PackageManager;
4 import android.graphics.Bitmap;
5 import android.graphics.BitmapFactory;
6 import android.graphics.Matrix;
7 import android.media.ExifInterface;
8 import android.os.Bundle;
9 import android.os.Environment;
10 import android.os.Handler;
11 import android.util.Log;
12 import android.widget.Button;
13 import android.widget.TextView;
14 import android.widget.Toast;
15 import androidx.annotation.NonNull;
16 import androidx.appcompat.app.AppCompatActivity;
17 import androidx.camera.core.CameraSelector;
18 import androidx.camera.core.ImageCapture;
19 import androidx.camera.core.ImageCaptureException;
20 import androidx.camera.lifecycle.ProcessCameraProvider;
21 import androidx.core.app.ActivityCompat;
22 import androidx.core.content.ContextCompat;
23 import androidx.lifecycle.LifecycleOwner;
24 import com.arcedios.myapplication.Model.Classifier;
25 import com.arcedios.myapplication.R;
26 import com.google.common.util.concurrent.ListenableFuture;
27 import java.io.File;
28 import java.io.FileOutputStream;
29 import java.io.IOException;
30 import java.util.ArrayList;
31 import java.util.concurrent.ExecutorService;
32 import java.util.concurrent.Executors;
33 import android.speech.tts.TextToSpeech;
34 import java.util.Locale;
35
36 public class Servicios extends AppCompatActivity {
37     2 usages
38     private android.camera.view.PreviewView camera;
39     4 usages
40     private ImageCapture imageCapture;
41     2 usages
42     private ExecutorService cameraExecutor;
43     2 usages
44     private Button recognizeButton;
45     2 usages
46     private static final int REQUEST_CAMERA_PERMISSION = 100;
47     2 usages
48     private TextView resultTextView, OjosAbiertos, OjosCerrados, NoHayNadie;
49     3 usages
50     private Handler handler = new Handler();
51     4 usages
52     private boolean isAnalyzing = false; // Para controlar el estado del análisis
53     6 usages
54     private ArrayList<String> historialResultados = new ArrayList<>();
55     4 usages
56     private boolean isColorChanging = false; // Controla si el fondo está cambiando
57     3 usages
58     private Handler colorHandler = new Handler();
59     3 usages
60     private Runnable colorRunnable;
61     7 usages
62     private TextToSpeech textToSpeech;
63
64     @Override
65     protected void onCreate(Bundle savedInstanceState) {
66         super.onCreate(savedInstanceState);
67         setContentView(R.layout.activity_servicios);
68         textToSpeech = new TextToSpeech(this, status -> {
69             if (status == TextToSpeech.SUCCESS) {
70                 textToSpeech.setLanguage(new Locale("es", "US")); // Para español latino
71             }
72         });
73         String userId = getIntent().getStringExtra("USER_ID");
74         camera = findViewById(R.id.camera);
75         recognizeButton = findViewById(R.id.button2);
76         resultTextView = findViewById(R.id.resultado);
77         OjosAbiertos = findViewById(R.id.ojosAbiertos2);
78         OjosCerrados = findViewById(R.id.ojosCerrados2);
79         NoHayNadie = findViewById(R.id.NoHayNadie2);
80         cameraExecutor = Executors.newSingleThreadExecutor();
81         recognizeButton.setOnClickListener() -> toggleAnalysis();
82     }
83
84     @Override
85     protected void onResume() {
86         super.onResume();
87         if (ContextCompat.checkSelfPermission(this, Manifest.permission.CAMERA) == PackageManager.PERMISSION_GRANTED) {
88             startCamera();
89         } else {
90             ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.CAMERA}, REQUEST_CAMERA_PERMISSION);
91         }
92     }
93
94     @Override
95     public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
96         super.onRequestPermissionsResult(requestCode, permissions, grantResults);
97         if (requestCode == REQUEST_CAMERA_PERMISSION) {
98             if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
99                 startCamera();
100             } else {
101                 Toast.makeText(this, "Permisos de cámara denegados", Toast.LENGTH_SHORT).show();
102             }
103         }
104     }

```

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 botón.

```

100 private void startCamera() {
101     ListenableFuture<ProcessCameraProvider> cameraProviderFuture = ProcessCameraProvider.getInstance(this);
102     cameraProviderFuture.addListener(() -> {
103         try {
104             ProcessCameraProvider cameraProvider = cameraProviderFuture.get();
105             android.camera.core.Preview preview = new android.camera.core.Preview.Builder().build();
106             preview.setSurfaceProvider(camera.getSurfaceProvider());
107             imageCapture = new ImageCapture.Builder().build();
108             cameraProvider.unbindAll();
109             cameraProvider.bindToLifecycle(this, CameraSelector.DEFAULT_FRONT_CAMERA, preview, imageCapture);
110         } catch (Exception e) {
111             Log.d(this, "Camera", "Error al iniciar la cámara", e);
112         }
113     }, ContextCompat.getMainExecutor(this));
114 }
115
116 private void toggleAnalysis() {
117     if (isAnalyzing) {
118         isAnalyzing = false;
119         handler.removeCallbacksAndMessages(null); // Detener análisis
120         Toast.makeText(this, "Análisis detenido", Toast.LENGTH_SHORT).show();
121     } else {
122         isAnalyzing = true;
123         startAnalysisLoop();
124         Toast.makeText(this, "Análisis iniciado", Toast.LENGTH_SHORT).show();
125     }
126 }
127
128 private void startAnalysisLoop() {
129     if (!isAnalyzing) return;
130     takePhoto();
131     handler.postDelayed(this::startAnalysisLoop, (long) 2000);
132 }
133 }

```

Imagen 23. Cámara y handler

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

```

134 private void takePhoto() {
135     if (imageCapture == null) return;
136     File photoFile = new File(getExternalFilesDir(Environment.DIRECTORY_PICTURES), "captured_image.jpg");
137     ImageCapture.OutputFileOptions outputFileOptions = new ImageCapture.OutputFileOptions.Builder(photoFile).build();
138     imageCapture.takePicture(outputFileOptions, ContextCompat.getMainExecutor(this), new ImageCapture.OnImageSavedCallback() {
139         @Override
140         public void onImageSaved(@NonNull ImageCapture.OutputFileResults outputFileResults) {
141             processImage(photoFile);
142         }
143     });
144     @Override
145     public void onError(@NonNull ImageCaptureException exception) {
146         Toast.makeText(this, "Error al tomar la foto", Toast.LENGTH_SHORT).show();
147     }
148 });
149
150 private void processImage(File imageFile) {
151     File processedImageFile = new File(getExternalFilesDir(Environment.DIRECTORY_PICTURES), "processed_image.jpg");
152     try (FileOutputStream out = new FileOutputStream(processedImageFile)) {
153         bitmap.compress(Bitmap.CompressFormat.JPEG, 100, out);
154     } catch (IOException e) {
155         Log.d(this, "SaveImage", "Error al guardar la imagen procesada", e);
156     }
157 }
158
159 private float[][][] preprocessImage(Bitmap bitmap) {
160     int width = 224, height = 224;
161     float[][][] input = new float[3][width][height];
162     for (int i = 0; i < height; i++) {
163         for (int j = 0; j < width; j++) {
164             int pixel = bitmap.getPixel(j, i);
165             input[0][i][j] = ((pixel >> 16) & 0xFF) / 255f - 1f;
166             input[1][i][j] = ((pixel >> 8) & 0xFF) / 255f - 1f;
167             input[2][i][j] = ((pixel & 0xFF) / 255f) - 1f;
168         }
169     }
170     return input;
171 }

```

Imagen 24. Tomar foto

Aquí se toma la foto y se almace, después comienza el procesamiento de la imagen.

```

172 private Bitmap fixImageRotation(String imagePath) {
173     try {
174         ExifInterface exif = new ExifInterface(imagePath);
175         int orientation = exif.getAttributeInt(ExifInterface.TAG_ORIENTATION, ExifInterface.ORIENTATION_UNDEFINED);
176         Matrix matrix = new Matrix();
177         switch (orientation) {
178             case ExifInterface.ORIENTATION_ROTATE_90:
179                 matrix.postRotate(90f);
180                 break;
181             case ExifInterface.ORIENTATION_ROTATE_180:
182                 matrix.postRotate(180f);
183                 break;
184             case ExifInterface.ORIENTATION_ROTATE_270:
185                 matrix.postRotate(270f);
186                 break;
187             default:
188                 return BitmapFactory.decodeFile(imagePath);
189         }
190         Bitmap originalBitmap = BitmapFactory.decodeFile(imagePath);
191         return Bitmap.createScaledBitmap(originalBitmap, 0, 0, 0, originalBitmap.getWidth(), originalBitmap.getHeight(), true);
192     } catch (IOException e) {
193         e.printStackTrace();
194         return null;
195     }
196 }

```

```

197 private void processImage(File imageFile) {
198     try {
199         Bitmap correctedBitmap = fixImageRotation(imageFile.getAbsolutePath());
200         if (correctedBitmap == null) return;
201         Bitmap resizedBitmap = Bitmap.createScaledBitmap(correctedBitmap, 224, 224, true);
202         saveProcessedImage(resizedBitmap);
203         float[][][] input = preprocessImage(resizedBitmap);
204         Classifier classifier = new Classifier(getResources(), "model_tfLite");
205         float[] results = classifier.predict(input);
206         classifier.close();
207         Object[] labels = new Object[results.length];
208         Object[] probabilities = new Object[results.length];
209         for (int i = 0; i < results.length; i++) {
210             float proba = results[i] * 100;
211             String label = "Objeto detectado: " + results[i] * 100 + "%";
212             String[] labels = {"Objeto detectado", "Objeto detectado", "Objeto detectado"};
213             int bestMatch = -1;
214             float bestConfidence = 0;
215             for (int i = 0; i < results.length; i++) {
216                 if (results[i][0] > bestConfidence) {
217                     bestConfidence = results[i][0];
218                     bestMatch = i;
219                 }
220             }
221             if (historialResultados.size() >= 10) {
222                 historialResultados.remove(0); // Eliminar el más antiguo
223             }
224             historialResultados.add(labels[bestMatch]); // Agregar el nuevo resultado
225             resultTextView.setText(labels[bestMatch]);
226             for (String resultado : historialResultados) {
227                 Log.d(this, "Historial", resultado);
228             }
229             if (historialResultados.size() >= 10) {
230                 analizarDatos();
231             }
232         } catch (IOException e) {
233             e.printStackTrace();
234         }
235     }
236 }
237
238 private void analizarDatos() {
239     String idUsuario = getIntent().getStringExtra("USER_ID");
240     int countObjetosCerrados = 0;
241     for (String resultado : historialResultados) {
242         if (resultado.equals("Objeto cerrado")) {
243             countObjetosCerrados++;
244         }
245     }
246     if (countObjetosCerrados >= 5) {
247         String nombre = getIntent().getStringExtra("NOMBRE");
248         String mensaje = nombre + " detectado";
249         Toast.makeText(this, mensaje, Toast.LENGTH_LONG).show();
250         startColorAnimation(); // Iniciar cambio de color
251         speakWarning(mensaje); // Dar la advertencia
252     } else {
253         stopColorAnimation(); // Detener si ya no hay peligro
254     }
255 }

```

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.

```

256 private void speakWarning(String message) {
257     if (textToSpeech != null) {
258         textToSpeech.speak(message, TextToSpeech.QUEUE_FLUSH, null, null);
259     }
260 }
261
262 private void startColorAnimation() {
263     if (isColorChanging) return; // Evitar que se inicie varias veces
264     isColorChanging = true;
265     colorRunnable = new Runnable() {
266         @Override
267         public void run() {
268             int color = isRed ? ContextCompat.getColor(this, R.color.blue) :
269                 ContextCompat.getColor(this, R.color.red);
270             findViewById(R.id.content).setBackgroundColor(color);
271             isRed = !isRed;
272             if (isColorChanging) {
273                 colorHandler.postDelayed(this, (long) 500); // Cambia cada 500ms (0.5 segundos)
274             }
275         }
276     };
277     colorHandler.post(colorRunnable);
278 }
279
280 private void stopColorAnimation() {
281     isColorChanging = false;
282     colorHandler.removeCallbacks(colorRunnable);
283     findViewById(R.id.content).setBackgroundColor(ContextCompat.getColor(this, R.color.white)); // Volver a blanco
284 }
285
286 @Override
287 protected void onDestroy() {
288     super.onDestroy();
289     if (textToSpeech != null) {
290         textToSpeech.stop();
291         textToSpeech.shutdown();
292     }
293     cameraExecutor.shutdown();
294     handler.removeCallbacksAndMessages(null);
295 }

```

Imagen 26. Voz y colores

En caso de que analizarDatos determine que la persona se

Para que toda la aplicación 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:

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:tools="http://schemas.android.com/tools">
4
5     <uses-feature
6         android:name="android.hardware.camera"
7         android:required="false" />
8
9     <uses-permission android:name="android.permission.CAMERA" />
10
11     <application
12         android:allowBackup="true"
13         android:dataExtractionRules="@xml/data_extraction_rules"
14         android:fullBackupContent="@xml/backup_rules"
15         android:icon="@mipmap/ic_launcher"
16         android:label="@string/app_name"
17         android:roundIcon="@mipmap/ic_launcher_round"
18         android:supportRtl="true"
19         android:theme="@style/Theme.MyApp2"
20         tools:targetApi="31">
21         <activity
22             android:name=".View.Registrarse"
23             android:exported="false" />
24
25         <activity
26             android:name=".View.Login"
27             android:exported="false" />
28
29         <activity
30             android:name=".View.Servicios"
31             android:exported="true"
32             android:theme="@style/Theme.MyApp2" />
33
34         <activity
35             android:name=".View.SplashScreen"
36             android:exported="true">
37             <intent-filter>
38                 <action android:name="android.intent.action.MAIN" />
39
40                 <category android:name="android.intent.category.LAUNCHER" />
41             </intent-filter>
42         </activity>
43
44         <provider
45             android:name="androidx.core.content.FileProvider"
46             android:authorities="com.ancedios.myapplication2.fileprovider"
47             android:grantUriPermissions="true"
48             android:exported="false">
49             <meta-data
50                 android:name="android.support.FILE_PROVIDER_PATHS"
51                 android:resource="@xml/file_paths" />
52         </provider>
53     </application>
54 </manifest>

```

Imagen 27. AndroidManifest

```

1  plugins {
2      alias(libs.plugins.android.application)
3      alias(libs.plugins.google.gms.google.services)
4  }
5
6  android {
7      namespace = "com.arcedios.myapplication"
8      compileSdk = 35
9
10     defaultConfig {
11         applicationId = "com.arcedios.myapplication"
12         minSdk = 30
13         targetSdk = 35
14         versionCode = 1
15         versionName = "1.0"
16         renderscriptTargetApi = 21
17         renderscriptSupportModeEnabled = true
18
19         testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
20     }
21
22     buildTypes {
23         release {
24             isMinifyEnabled = false
25             proguardFiles(
26                 getDefaultProguardFile("proguard-android-optimize.txt"),
27                 "proguard-rules.pro"
28             )
29         }
30     }
31
32     compileOptions {
33         sourceCompatibility = JavaVersion.VERSION_11
34         targetCompatibility = JavaVersion.VERSION_11
35     }
36 }
37
38 buildFeatures {
39     viewBinding = true
40     dataBinding = true
41     modelBinding = true
42 }
43
44 dependencies {
45     implementation ("org.jetbrains.kotlin:kotlin-stdlib:1.9.0")
46
47     implementation(libs.appcompat)
48     implementation(libs.material)
49     implementation(libs.activity)
50     implementation(libs.constraintlayout)
51     implementation(libs.firebase.database)
52     implementation("org.tensorflow:tensorflow-lite:2.9.0")
53     implementation ("org.tensorflow:tensorflow-lite:2.9.0")
54     implementation ("org.tensorflow:tensorflow-lite-metadata:0.4.3")
55     implementation ("org.tensorflow:tensorflow-lite-support:0.4.3")
56     testImplementation(libs.junit)
57     androidTestImplementation(libs.ext.junit)
58     androidTestImplementation(libs.espresso.core)
59     implementation("androidx.camera:camera-core:1.3.0")
60     implementation("androidx.camera:camera-camera2:1.3.0")
61     implementation("androidx.camera:camera-lifecycle:1.3.0")
62     implementation("androidx.camera:camera-view:1.3.0")
63 }

```

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.

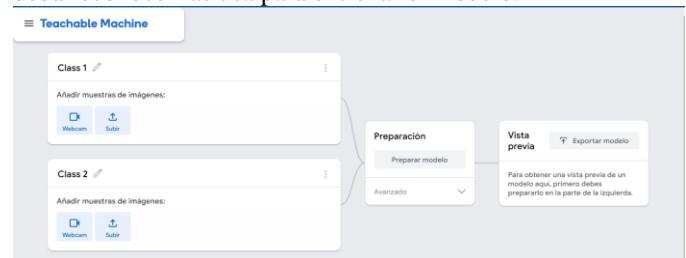


Imagen 29. Teacheble Machine

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.

personal considero que el trabajo fue muy divertido y aprendí mucho.

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 Teachable 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