Código de la Aplicación (En Android Studio)

Proyecto: Salón de Clases Inteligente

Parte 1: Aplicación de Android

Lógica Principal (MainActivity.java)

```
package com.example.salondeclases;

import android.content.res.ColorStateList;

import android.graphics.Color;

import android.os.Bundle;

import android.os.Handler;

import android.os.Looper;

import android.view.View;

import android.widget.Button;

import android.widget.TextView;

import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import androidx.appcompat.widget.SwitchCompat;

import androidx.core.content.ContextCompat;

import com.google.gson.Gson;

import okhttp3.Call;

import okhttp3.Callback;

import okhttp3.OkHttpClient;

import okhttp3.Request;

import okhttp3.Response;

import java.io.IOException;
```

```
public class MainActivity extends AppCompatActivity {
    private final String BASE_URL = "http://192.168.24.51"; // IP del ESP32
    private final OkHttpClient client = new OkHttpClient();
    private final Gson gson = new Gson();
    private final Handler handler = new Handler(Looper.getMainLooper());

    private SwitchCompat switchAllLights;
    private TextView textGas;
    private Button buttonStopAlarm;

    // Clase para mapear el JSON de estado del ESP32
    private static class DeviceStatus {
        boolean lights;
        String gas;
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Inicialización de vistas
        switchAllLights = findViewById(R.id.switchAllLights);
        textGas = findViewById(R.id.textGas);
        buttonStopAlarm = findViewById(R.id.buttonStopAlarm);
```

```
        // Listeners
        setupSwitchListeners();
        setupButtonListeners();

        // Iniciar la actualización periódica del estado
        startPolling();
    }

    private void setupButtonListeners() {
        buttonStopAlarm.setOnClickListener(v -> {
            Toast.makeText(MainActivity.this, "Enviando comando para apagar alarma...",
Toast.LENGTH_SHORT).show();
            sendCommand("/alarm", false); // Envía "off" al endpoint /alarm
        });
    }

    private void setupSwitchListeners() {
        switchAllLights.setOnCheckedChangeListener((buttonView, isChecked) -> {
            String state = isChecked ? "ON" : "OFF";
            Toast.makeText(MainActivity.this, "Luces: " + state,
Toast.LENGTH_SHORT).show();
            sendCommand("/lights", isChecked);
        });
    }

    private void sendCommand(String endpoint, boolean state) {
        String url = BASE_URL + endpoint + "?state=" + (state ? "on" : "off");
```

```
    Request request = new Request.Builder().url(url).build();
    client.newCall(request).enqueue(new Callback() {
      @Override
      public void onFailure(Call call, IOException e) {
        runOnUiThread(() -> Toast.makeText(MainActivity.this, "Error de conexión",
Toast.LENGTH_SHORT).show());
      }
      @Override
      public void onResponse(Call call, Response response) throws IOException {
        response.close();
      }
    });
  }

  private void startPolling() {
    handler.post(new Runnable() {
      @Override
      public void run() {
        fetchStatus();
        handler.postDelayed(this, 5000); // Consulta cada 5 segundos
      }
    });
  }

  private void fetchStatus() {
    Request request = new Request.Builder().url(BASE_URL + "/status").build();
```

```
    client.newCall(request).enqueue(new Callback() {
        @Override
        public void onFailure(Call call, IOException e) {
            // No mostramos Toast aquí para no saturar al usuario
        }

        @Override
        public void onResponse(Call call, Response response) throws IOException {
            if (response.isSuccessful() && response.body() != null) {
                String json = response.body().string();
                DeviceStatus status = gson.fromJson(json, DeviceStatus.class);
                runOnUiThread(() -> updateUI(status));
            }
            response.close();
        }
    });
}

private void updateUI(DeviceStatus status) {
    if (status == null) {
        return;
    }

    // Actualizar interruptor de luces
    switchAllLights.setOnCheckedChangeListener(null);
    switchAllLights.setChecked(status.lights);
```

```
        setupSwitchListeners();

        // Actualizar sensor de gas y estado del botón de alarma
        String gasStatus = (status.gas != null) ? status.gas : "N/A";
        textGas.setText(getString(R.string.gas_sensor_status, gasStatus));

        if (gasStatus.toUpperCase().contains("ALERTA")) {
            buttonStopAlarm.setEnabled(true);
buttonStopAlarm.setBackgroundTintList(ColorStateList.valueOf(ContextCompat.getColor(this, R.color.alarm_red)));
        } else {
            buttonStopAlarm.setEnabled(false);
buttonStopAlarm.setBackgroundTintList(ColorStateList.valueOf(ContextCompat.getColor(this, R.color.colorPrimary)));
        }
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        handler.removeCallbacksAndMessages(null); // Detener el polling al cerrar la app
    }
}
```

Interfaz de Usuario (activity_main.xml)

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:layout_width="match_parent"
  android:layout_height="match_parent">

  <ImageView
    android:id="@+id/icon_wifi"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/wifi"
    android:layout_alignParentTop="true"
    android:layout_alignParentEnd="true"
    android:layout_marginTop="16dp"
    android:layout_marginEnd="16dp"
    android:contentDescription="@string/wifi_icon_description" />

  <ScrollView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_below="@id/icon_wifi">

    <LinearLayout
      android:orientation="vertical"
      android:padding="16dp"
      android:layout_width="match_parent"
      android:layout_height="wrap_content">
```

```
<TextView
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:gravity="center"
   android:text="@string/main_title"
   android:textSize="24sp"
   android:textStyle="bold" />

<!-- Sección Iluminación -->
<ImageView
   android:layout_width="match_parent"
   android:layout_height="122dp"
   android:layout_marginTop="16dp"
   android:contentDescription="@string/main_image_description"
   android:scaleType="centerCrop"
   android:src="@drawable/salondeclases" />

<LinearLayout
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:orientation="vertical"
   android:layout_marginTop="16dp">

   <LinearLayout
      android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:gravity="center_vertical"
        android:orientation="horizontal">

        <ImageView
            android:layout_width="36dp"
            android:layout_height="36dp"
            android:contentDescription="@string/lighting_icon_description"
            android:src="@drawable/foco" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:text="@string/lighting_section_title"
            android:textSize="18sp"
            android:textStyle="bold" />
    </LinearLayout>

    <androidx.appcompat.widget.SwitchCompat
        android:id="@+id/switchAllLights"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="start"/>

</LinearLayout>
```

```
<!-- Sección Sensores -->
<LinearLayout
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:orientation="horizontal"
   android:gravity="center_vertical"
   android:layout_marginTop="16dp">

   <ImageView
      android:layout_width="36dp"
      android:layout_height="36dp"
      android:src="@drawable/sensor"
      android:contentDescription="@string/sensors_icon_description"/>

   <TextView
      android:text="@string/sensors_section_title"
      android:textSize="18sp"
      android:textStyle="bold"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_marginStart="8dp"/>
</LinearLayout>

<TextView android:id="@+id/textGas" android:text="@string/gas_sensor_status"
   android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"/>

    <Button
        android:id="@+id/buttonStopAlarm"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:text="@string/stop_alarm_button" />

  </LinearLayout>
 </ScrollView>

</RelativeLayout>
```

Manifiesto de la Aplicación (AndroidManifest.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-permission android:name="android.permission.INTERNET" />

 <application
  android:allowBackup="true"
  android:dataExtractionRules="@xml/data_extraction_rules"
  android:fullBackupContent="@xml/backup_rules"
  android:icon="@mipmap/ic_launcher"
```

```
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Salondeclases"
        android:usesCleartextTraffic="true">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

2.1. Código Principal de la Maqueta (.ino) (ESP32)

```
#include <WiFi.h>
#include <WebServer.h>
#include <Arduino_JSON.h>
#include <ESP32Servo.h>

const char* ssid = "FIMaz Telefonos";     // Tu red Wi-Fi
const char* password = "yoapoyoalrector"; // Tu contraseña
```

```
const int trigEntrada = 5;
const int echoEntrada = 18;
const int trigSalida = 19;
const int echoSalida = 21;
const int pinServo = 13;
const int foco1 = 14;
const int foco2 = 27;
const int foco3 = 26;
const int botonFocos = 25;
const int sensorGas = 34;
const int botonGas = 33;
const int buzzerGas = 12;
const int UMBRAL_GAS = 560;
bool alarmaGasActiva = false;
unsigned long silencioGasHasta = 0;
const long TIEMPO_SILENCIO = 5000;
Servo puerta;
bool focosEnModoAuto = true;
int personasDentro = 0;

// -- Tus variables de tiempo --
unsigned long tiempoBotonGas = 0;
unsigned long tiempoUltimoBotonFocos = 0;
unsigned long tiempoDeteccion = 0;
unsigned long tiempoPuerta = 0;
```

```
bool puertaAbierta = false;
const int distanciaUmbral = 5;
const int debounce = 300;
const int tiempoPuertaAbierta = 1000;
const int tiempoBloqueoLectura = 600;
WebServer server(80);

void handleStatus() {
  bool lucesEncendidas = (digitalRead(foco1) == HIGH);
  String estadoDelGas = alarmaGasActiva ? "ALERTA DE GAS" : "Normal";

  JSONVar jsonResponse;
  jsonResponse["lights"] = lucesEncendidas;
  jsonResponse["gas"] = estadoDelGas;

  String jsonString = JSON.stringify(jsonResponse);
  server.send(200, "application/json", jsonString);
}

void encenderFocos();
void apagarFocos();

void handleLights() {
  if (server.hasArg("state")) {
    focosEnModoAuto = false;
    String estado = server.arg("state");
```

```
  if (estado == "on") {
    encenderFocos();
  } else if (estado == "off") {
    apagarFocos();
  }
  server.send(200, "text/plain", "OK");
 }
}

void handleAlarm() {
 if (server.hasArg("state") && server.arg("state") == "off") {
  silencioGasHasta = millis() + TIEMPO_SILENCIO;
  noTone(buzzerGas);
 }
 server.send(200, "text/plain", "OK");
}

void handleNotFound() {
 server.send(404, "text/plain", "Ruta no encontrada.");
}
long medirDistancia(int trig, int echo) {
 digitalWrite(trig, LOW); delayMicroseconds(2);
 digitalWrite(trig, HIGH); delayMicroseconds(10);
 digitalWrite(trig, LOW);
 long duracion = pulseIn(echo, HIGH, 30000);
 long distancia = duracion * 0.034 / 2;
```

```
  if (distancia == 0) return 999;
  return distancia;
}

void encenderFocos() {
  digitalWrite(foco1, HIGH);
  digitalWrite(foco2, HIGH);
  digitalWrite(foco3, HIGH);
}

void apagarFocos() {
  digitalWrite(foco1, LOW);
  digitalWrite(foco2, LOW);
  digitalWrite(foco3, LOW);
}

void abrirPuerta() {
  puerta.write(90);
  puertaAbierta = true;
  tiempoPuerta = millis();
}

void cerrarPuertaAuto() {
  if (puertaAbierta && millis() - tiempoPuerta >= tiempoPuertaAbierta) {
    puerta.write(0);
    puertaAbierta = false;
```

```
  }
}
void setup() {
  Serial.begin(115200);
  delay(500);

  // --- Conexión WIFI ---
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
  }
  Serial.print("IP del ESP32: ");
  Serial.println(WiFi.localIP());

  // --- Configuración de Pines ---
  pinMode(trigEntrada, OUTPUT);
  pinMode(echoEntrada, INPUT);
  pinMode(trigSalida, OUTPUT);
  pinMode(echoSalida, INPUT);
  puerta.attach(pinServo);
  puerta.write(0);
  pinMode(foco1, OUTPUT);
  pinMode(foco2, OUTPUT);
  pinMode(foco3, OUTPUT);
  pinMode(botonFocos, INPUT_PULLUP);
  pinMode(botonGas, INPUT_PULLUP);
```

```
  pinMode(sensorGas, INPUT);
  pinMode(buzzerGas, OUTPUT);
  apagarFocos();
  server.on("/status", HTTP_GET, handleStatus);
  server.on("/lights", HTTP_GET, handleLights);
  server.on("/alarm", HTTP_GET, handleAlarm);
  server.onNotFound(handleNotFound);
  server.begin();
}

void loop() {
  server.handleClient();

  unsigned long ahora = millis();
  int valorGas = analogRead(sensorGas);

  if (valorGas >= UMBRAL_GAS && !alarmaGasActiva) {
    alarmaGasActiva = true;
  } else if (valorGas < UMBRAL_GAS && alarmaGasActiva) {
    alarmaGasActiva = false;
  }

  if (digitalRead(botonGas) == LOW && ahora - tiempoBotonGas > debounce) {
    if (alarmaGasActiva) {
      silencioGasHasta = ahora + TIEMPO_SILENCIO;
    }
```

```
  tiempoBotonGas = ahora;
 }

 if (alarmaGasActiva && ahora > silencioGasHasta) {
  tone(buzzerGas, 800);
 } else {
  noTone(buzzerGas);
 }

 if (digitalRead(botonFocos) == LOW && ahora - tiempoUltimoBotonFocos >
debounce) {
  focosEnModoAuto = !focosEnModoAuto;
  if (!focosEnModoAuto) apagarFocos();
  else if (personasDentro > 0) encenderFocos();
  tiempoUltimoBotonFocos = ahora;
 }

 if (ahora - tiempoDeteccion > tiempoBloqueoLectura) {
  long distEntrada = medirDistancia(trigEntrada, echoEntrada);
  long distSalida  = medirDistancia(trigSalida, echoSalida);

  if (distEntrada < distanciaUmbral && distSalida > (distanciaUmbral + 10)) {
   if (!puertaAbierta) abrirPuerta();
   personasDentro++;
   if (focosEnModoAuto && personasDentro > 0) encenderFocos();
   tiempoDeteccion = ahora;
```

```
    }

    if (distSalida < distanciaUmbral && distEntrada > (distanciaUmbral + 10)) {
      if (!puertaAbierta) abrirPuerta();
      if (personasDentro > 0) personasDentro--;
      if (personasDentro == 0 && focosEnModoAuto) apagarFocos();
      tiempoDeteccion = ahora;
    }
  }

  cerrarPuertaAuto();
}
```