```java
package com.example.gui_design.bluetoothActivities;

import android.graphics.Color;
import android.graphics.PorterDuff;
import android.graphics.drawable.ColorDrawable;
import android.graphics.drawable.Drawable;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import com.example.gui_design.Application;
import com.example.gui_design.R;

import java.util.List;

public class DeviceListAdapter extends RecyclerView.Adapter<DeviceListAdapter.viewHolder> {

    private List<AvailableDevice> deviceList;

    public class viewHolder extends RecyclerView.ViewHolder {
        public TextView textView_device_name;
        public ImageView imgView_device_state_icon;

        public viewHolder(View view) {
            super(view);
            textView_device_name = (TextView) view.findViewById(R.id.textView_bluetooth_device_name);
            imgView_device_state_icon = (ImageView) view.findViewById(R.id.
imageView_bluetooth_device_state_icon);
        }
    }


    public DeviceListAdapter(List<AvailableDevice> deviceList) {
        this.deviceList = deviceList;
    }

    @Override
    public viewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View itemView = LayoutInflater.from(parent.getContext())
                .inflate(R.layout.bluetooth_list_layout_style, parent, false);

        return new viewHolder(itemView);
    }

    @Override
    public void onBindViewHolder(viewHolder holder, int position) {
        AvailableDevice device = deviceList.get(position);
        holder.textView_device_name.setText(device.getDeviceName());
        holder.textView_device_name.setTextAppearance(device.getTextAppearance());

        int iconID = device.getDeviceStateIconID();

        if (iconID!=0) {
            Drawable icon = Application.getContext().getDrawable(iconID);
            icon.setColorFilter(Application.getContext().getColor(R.color.colorPrimary), PorterDuff.
Mode.SRC_ATOP);
            holder.imgView_device_state_icon.setImageDrawable(icon);
        } else {
            holder.imgView_device_state_icon.clearColorFilter();
            Drawable transparentDrawable = new ColorDrawable(Color.TRANSPARENT);
            holder.imgView_device_state_icon.setImageDrawable(transparentDrawable);
        }
    }

    @Override
    public int getItemCount() {
        return deviceList.size();
    }
}

```

File - C:_Library256\Feria_2019\Interfaz_grafica\app\src\main\java\com\example\gui_design\GoogleSpeechRecognizer.java

```java
package com.example.gui_design;

import android.content.Intent;
import android.os.Bundle;
import android.speech.RecognitionListener;
import android.speech.RecognizerIntent;
import android.speech.SpeechRecognizer;
import android.view.View;

import com.example.gui_design.bluetoothActivities.BluetoothConnection;

import org.apache.commons.lang3.StringUtils;

import java.text.Normalizer;
import java.util.ArrayList;
import java.util.Arrays;

public class GoogleSpeechRecognizer extends MainActivity implements RecognitionListener {

    private static SpeechRecognizer speechRecognizer = null;

    public static void startSpeechListening(){
        String language =  "es-ES";
        Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
        intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,language);
        intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, language);
        intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_PREFERENCE, language);
        intent.putExtra(RecognizerIntent.EXTRA_ONLY_RETURN_LANGUAGE_PREFERENCE, language);
        intent.putExtra(RecognizerIntent.EXTRA_CALLING_PACKAGE, Application.getActivity().
getPackageName());
        intent.putExtra(RecognizerIntent.EXTRA_PARTIAL_RESULTS,true);
        speechRecognizer.startListening(intent);
    }

    public static void setupRecognizer(){
        speechRecognizer = SpeechRecognizer.createSpeechRecognizer(Application.getContext());
        speechRecognizer.setRecognitionListener(new GoogleSpeechRecognizer());
    }

    public static void shutdownRecognizer(){
        speechRecognizer.stopListening();
        speechRecognizer.cancel();
        speechRecognizer.destroy();
        speechRecognizer = null;
        System.gc();
    }

    public static void forceStopRecognizer(){
        speechRecognizer.stopListening();
        speechRecognizer.cancel();

        PocketsphinxListener.startListening();
    }


    public static void analizeListenerResults(ArrayList<String> data, float[] confidenceScore){
        outerloop:
        for (int i = 0; i < data.size(); i++) {
            String hypothesis = Normalizer.normalize(((String) data.get(i)).toLowerCase(), Normalizer.
Form.NFKD);
            String[] hypothesisSplit = hypothesis.split(" ");

            if (Arrays.asList(hypothesisSplit).contains("encender") && Arrays.asList(hypothesisSplit).
contains("velador")) {
                BluetoothConnection.composeAndSend(BluetoothConnection.commandType.TURN_RELAY_ON, 0);
                break outerloop;
            }

            if (Arrays.asList(hypothesisSplit).contains("encender") && Arrays.asList(hypothesisSplit).
contains("estufa")) {
                BluetoothConnection.composeAndSend(BluetoothConnection.commandType.TURN_RELAY_ON, 1);
                break outerloop;
            }

            if (Arrays.asList(hypothesisSplit).contains("encender") && Arrays.asList(hypothesisSplit).
contains("ventilador")) {
                BluetoothConnection.composeAndSend(BluetoothConnection.commandType.TURN_RELAY_ON, 2);
```

```
 73                 break outerloop;
 74             }
 75
 76             if (Arrays.asList(hypothesisSplit).contains("encender") && Arrays.asList(hypothesisSplit)
    .contains("luz")) {
 77                 BluetoothConnection.composeAndSend(BluetoothConnection.commandType.TURN_RELAY_ON, 3);
 78                 break outerloop;
 79             }
 80
 81             if (Arrays.asList(hypothesisSplit).contains("apagar") && Arrays.asList(hypothesisSplit).
    contains("velador")) {
 82                 BluetoothConnection.composeAndSend(BluetoothConnection.commandType.TURN_RELAY_OFF, 0)
    ;
 83                 break outerloop;
 84             }
 85
 86             if (Arrays.asList(hypothesisSplit).contains("apagar") && Arrays.asList(hypothesisSplit).
    contains("estufa")) {
 87                 BluetoothConnection.composeAndSend(BluetoothConnection.commandType.TURN_RELAY_OFF, 1)
    ;
 88                 break outerloop;
 89             }
 90
 91             if (Arrays.asList(hypothesisSplit).contains("apagar") && Arrays.asList(hypothesisSplit).
    contains("ventilador")) {
 92                 BluetoothConnection.composeAndSend(BluetoothConnection.commandType.TURN_RELAY_OFF, 2)
    ;
 93                 break outerloop;
 94             }
 95
 96             if (Arrays.asList(hypothesisSplit).contains("apagar") && Arrays.asList(hypothesisSplit).
    contains("luz")) {
 97                 BluetoothConnection.composeAndSend(BluetoothConnection.commandType.TURN_RELAY_OFF, 3)
    ;
 98                 break outerloop;
 99             }
100
101             if (Arrays.asList(hypothesisSplit).contains("encender") && Arrays.asList(hypothesisSplit)
    .contains("todo")) {
102                 BluetoothConnection.composeAndSend(BluetoothConnection.commandType.TURN_EVERYTHING_ON
    , 0);
103                 break outerloop;
104             }
105
106             if (Arrays.asList(hypothesisSplit).contains("apagar") && Arrays.asList(hypothesisSplit).
    contains("todo")) {
107                 BluetoothConnection.composeAndSend(BluetoothConnection.commandType.
    TURN_EVERYTHING_OFF, 0);
108                 break outerloop;
109             }
110
111             if (Arrays.asList(hypothesisSplit).contains("llamar") && Arrays.asList(hypothesisSplit).
    contains("enfermera")) {
112                 BluetoothConnection.composeAndSend(BluetoothConnection.commandType.CALL_NURSE, 0);
113                 break outerloop;
114             }
115
116             if (Arrays.asList(hypothesisSplit).contains("llamar") && Arrays.asList(hypothesisSplit).
    contains("enfermeria")) {
117                 BluetoothConnection.composeAndSend(BluetoothConnection.commandType.CALL_NURSE, 0);
118                 break outerloop;
119             }
120
121
122             if (Arrays.asList(hypothesisSplit).contains("encender") && Arrays.asList(hypothesisSplit)
    .contains("rele")) {
123                 for (String word : hypothesisSplit) {
124                     if (StringUtils.isNumeric(word) == true) {
125                         int number = Integer.parseInt(word);
126                         BluetoothConnection.composeAndSend(BluetoothConnection.commandType.
    TURN_RELAY_ON, number);
127                         break outerloop;
128                     }
129                 }
130             }
131
132             if (Arrays.asList(hypothesisSplit).contains("apagar") && Arrays.asList(hypothesisSplit).
```

```
132 contains("rele")) {
133                 for (String word : hypothesisSplit) {
134                     if (StringUtils.isNumeric(word) == true) {
135                         int number = Integer.parseInt(word);
136                         BluetoothConnection.composeAndSend(BluetoothConnection.commandType.
    TURN_RELAY_OFF, number);
137                         break outerloop;
138                     }
139                 }
140             }
141 
142         }
143     }
144 
145     @Override
146     public void onReadyForSpeech(Bundle params) {
147         MicrophoneHandler.setState(MicrophoneHandler.MicrophoneState.LISTENING_COMMAND);
148     }
149 
150     @Override
151     public void onBeginningOfSpeech() {
152 
153     }
154 
155     @Override
156     public void onRmsChanged(float rmsdB) {
157 
158     }
159 
160     @Override
161     public void onBufferReceived(byte[] buffer) {
162 
163     }
164 
165     @Override
166     public void onEndOfSpeech() {
167         Application.toastMessage("End of speech");
168     }
169 
170     @Override
171     public void onError(int error) {
172         Application.toastMessage("On error" + error);
173         MicrophoneHandler.setState(MicrophoneHandler.MicrophoneState.LISTENING_KEYOWRD);
174         PocketsphinxListener.startListening();
175     }
176 
177     @Override
178     public void onResults(Bundle results) {
179         String str = new String();
180         ArrayList<String> resultsArrayList = results.getStringArrayList(SpeechRecognizer.
    RESULTS_RECOGNITION);
181         float[] confidenceScore = results.getFloatArray(SpeechRecognizer.CONFIDENCE_SCORES);
182         analizeListenerResults(resultsArrayList, confidenceScore);
183 
184         StringBuilder builder = new StringBuilder();
185         for (String line : resultsArrayList) {
186             builder.append(line + "\n");
187         }
188 
189         MainActivity.textView_speechRecognitionResults.setText(builder.toString());
190         MainActivity.textView_speechRecognitionResults.setVisibility(View.VISIBLE);
191 
192 
193 
194         MicrophoneHandler.setState(MicrophoneHandler.MicrophoneState.LISTENING_KEYOWRD);
195 
196         PocketsphinxListener.startListening();
197     }
198 
199     @Override
200     public void onPartialResults(Bundle partialResults) {
201         String str = new String();
202         ArrayList<String> resultsArrayList = partialResults.getStringArrayList(SpeechRecognizer.
    RESULTS_RECOGNITION);
203         float[] confidenceScore = partialResults.getFloatArray(SpeechRecognizer.CONFIDENCE_SCORES);
204 
205         StringBuilder builder = new StringBuilder();
```

```
206         for (String line : resultsArrayList) {
207             builder.append(line + "\n");
208         }
209 
210     }
211 
212     @Override
213     public void onEvent(int eventType, Bundle params) {
214 
215     }
216 }
217 
```

```java
package com.example.gui_design.bluetoothActivities;

import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothServerSocket;
import android.bluetooth.BluetoothSocket;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.widget.Toast;

import com.example.gui_design.Application;

import java.io.InputStream;
import java.io.OutputStream;
import java.nio.charset.StandardCharsets;
import java.util.Set;
import java.util.UUID;
import java.util.concurrent.Executors;
import java.util.concurrent.ScheduledExecutorService;
import java.util.concurrent.ScheduledFuture;


public class BluetoothConnection extends BluetoothActivity {

    static ScheduledExecutorService bluetoothThreadPool= Executors.newScheduledThreadPool(1);
    static ScheduledFuture<?> bluetoothComThread;
    public static OutputStream outputStream;
    public static InputStream inputStream;
    public static BluetoothAdapter btAdapter;
    public static BluetoothDevice targetDevice;
    public static BluetoothDevice[] bluetoothDevicesList;
    public static BluetoothSocket btSocket;
    public static BluetoothServerSocket btServerSocket;
    public static UUID btUUID = UUID.fromString("00001101-0000-1000-8000-00805f9b34fb");
    public static String btString = "HC-05";
    public static Boolean bluetoothConnected = false;

    public static void listAvailableBluetoothDevices(){
        btAdapter = BluetoothAdapter.getDefaultAdapter();

        if (btAdapter != null) {
            if (btAdapter.isEnabled()) {
                Set<BluetoothDevice> bondedDevices = btAdapter.getBondedDevices();
                if(bondedDevices.size() > 0) {
                    bluetoothDevicesList = bondedDevices.toArray(new BluetoothDevice[bondedDevices.
size()]);
                    String[] devicesNames = new String[bondedDevices.size()];
                    deviceList.clear();

                    for (int x = 0; x < bondedDevices.size(); x++) {
                        devicesNames[x] = bluetoothDevicesList[x].getName();
                        deviceList.add(new AvailableDevice(bluetoothDevicesList[x].getName(),
AvailableDevice.DeviceState.DEFAULT));
                    }

                    mAdapter.notifyDataSetChanged();

                }
            } else {
                BluetoothActivity.setBluetoothStateGui(bluetoothConnectionState.BLUETOOTH_DISABLED);
            }
        }
    }


    public static void connectBluetoothDevice(int deviceIndex){
        targetDevice = bluetoothDevicesList[deviceIndex];

        try{
            btSocket = targetDevice.createRfcommSocketToServiceRecord(btUUID);
            btSocket.connect();
            inputStream = btSocket.getInputStream();
            outputStream = btSocket.getOutputStream();
            BluetoothActivity.setBluetoothStateGui(bluetoothConnectionState.SYSTEM_CONNECTED);
            deviceList.get(deviceIndex).setCurrentState(AvailableDevice.DeviceState.CONNECTED);
            BluetoothActivity.mAdapter.notifyDataSetChanged();
```

```
 76         }catch (Throwable th){
 77             Application.toastMessage(th.toString());
 78         }
 79
 80     }
 81
 82
 83
 84
 85
 86     public static void connectionLost(){
 87
 88     }
 89
 90     public static void closeAllConnections(){
 91         try {
 92             if (bluetoothComThread != null) bluetoothComThread.cancel(true);
 93             if (btSocket != null) btSocket.close();
 94             if (outputStream != null) outputStream.close();
 95             if (inputStream != null) inputStream.close();
 96             bluetoothConnected = false;
 97
 98             btSocket = null;
 99             outputStream = null;
100             inputStream = null;
101         } catch (Throwable th){
102
103         }
104     }
105
106     public enum commandType {TURN_RELAY_ON, TURN_RELAY_OFF, TURN_EVERYTHING_ON, TURN_EVERYTHING_OFF,
    CALL_NURSE}
107
108     public static boolean composeAndSend(commandType cmdType, int parameter){
109         try {
110             String commandOutput = null;
111             switch (cmdType) {
112                 case TURN_RELAY_ON:
113                     commandOutput = String.format("/TURN_RELAY_ON[%s]$", Integer.toString(parameter)
    );
114                     break;
115                 case TURN_RELAY_OFF:
116                     commandOutput = String.format("/TURN_RELAY_OFF[%s]$", Integer.toString(parameter)
     );
117                     break;
118                 case TURN_EVERYTHING_ON:
119                     commandOutput = "/TURN_EVERYTHING_ON$";
120                     break;
121                 case TURN_EVERYTHING_OFF:
122                     commandOutput = "/TURN_EVERYTHING_OFF$";
123                     break;
124                 case CALL_NURSE:
125                     commandOutput = "/CALL_NURSE";
126                     break;
127             }
128
129             outputStream.write(commandOutput.getBytes(StandardCharsets.US_ASCII));
130             Application.toastMessage(commandOutput);
131             return true;
132         }catch (Throwable th) {
133             Application.toastMessage(th.toString());
134             return false;
135         }
136     }
137
138     public static BroadcastReceiver bluetoothAvailability = new BroadcastReceiver() {
139         @Override
140         public void onReceive(Context context, Intent intent) {
141             String action = intent.getAction();
142             BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
143
144             if (BluetoothDevice.ACTION_FOUND.equals(action)) {
145
146             }
147             else if (BluetoothDevice.ACTION_ACL_CONNECTED.equals(action)) {
148                     Toast.makeText(Application.getActivity(), "Device connected!", Toast.LENGTH_SHORT
    ).show();
```

```
149             }
150             else if (BluetoothDevice.ACTION_ACL_DISCONNECTED.equals(action)) {
151                 Toast.makeText(Application.getActivity(), "Device disconnected!", Toast.LENGTH_SHORT)
    .show();
152                 BluetoothConnection.connectionLost();
153                 BluetoothConnection.closeAllConnections();
154             }
155         }
156     };
157
158     public static final BroadcastReceiver bluetoothAdapterState = new BroadcastReceiver() {
159         @Override
160         public void onReceive(Context context, Intent intent) {
161             final String action = intent.getAction();
162
163             if (action.equals(BluetoothAdapter.ACTION_STATE_CHANGED)) {
164                 final int state = intent.getIntExtra(BluetoothAdapter.EXTRA_STATE,
165                         BluetoothAdapter.ERROR);
166                 switch (state) {
167                     case BluetoothAdapter.STATE_OFF:
168
169                         break;
170                     case BluetoothAdapter.STATE_TURNING_OFF:
171
172                         break;
173                     case BluetoothAdapter.STATE_ON:
174
175                         break;
176                     case BluetoothAdapter.STATE_TURNING_ON:
177
178                         break;
179                 }
180             }
181         }
182     };
183 }
184
```

```java
package com.example.gui_design.bluetoothActivities;

import android.graphics.PorterDuff;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import android.support.v4.content.ContextCompat;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.DefaultItemAnimator;
import android.support.v7.widget.DividerItemDecoration;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.support.v7.widget.Toolbar;
import android.view.MenuItem;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;

import com.example.gui_design.Application;
import com.example.gui_design.R;
import com.example.gui_design.settingsActivities.RecyclerTouchListener;

import java.util.ArrayList;
import java.util.List;

public class BluetoothActivity extends AppCompatActivity {
    private static Toolbar toolbar;


    public static List<AvailableDevice> deviceList = new ArrayList<>();
    private RecyclerView recyclerView;
    public static DeviceListAdapter mAdapter;

    public static ImageView imgView_bluetoothStatusIcon;
    public static TextView textView_bluetoothState;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.bluetooth_layout);
        Application.setActivity(this);
        Application.setContext(this);

        toolbar = this.findViewById(R.id.toolbar);
        textView_bluetoothState = this.findViewById(R.id.textView_btStatus);
        imgView_bluetoothStatusIcon = this.findViewById(R.id.imgView_btStatusIcon);

        setSupportActionBar(toolbar);

        // add back arrow to toolbar
        if (getSupportActionBar() != null){
            getSupportActionBar().setDisplayHomeAsUpEnabled(true);
            getSupportActionBar().setDisplayShowHomeEnabled(true);

            final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
abc_ic_ab_back_material);
            upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP);
            getSupportActionBar().setHomeAsUpIndicator(upArrow);
        }

        recyclerView = (RecyclerView) findViewById(R.id.deviceListRecyclerView);

        mAdapter = new DeviceListAdapter(deviceList);

        recyclerView.setHasFixedSize(true);

        // vertical RecyclerView
        // keep movie_list_row.xml width to `match_parent`
        RecyclerView.LayoutManager mLayoutManager = new LinearLayoutManager(getApplicationContext());

        // horizontal RecyclerView
        // keep movie_list_row.xml width to `wrap_content`
        // RecyclerView.LayoutManager mLayoutManager = new LinearLayoutManager(getApplicationContext
(), LinearLayoutManager.HORIZONTAL, false);

        recyclerView.setLayoutManager(mLayoutManager);

```

```
76          // adding inbuilt divider line
77          recyclerView.addItemDecoration(new DividerItemDecoration(this, LinearLayoutManager.VERTICAL))
    ;
78
79          // adding custom divider line with padding 16dp
80          //recyclerView.addItemDecoration(new MyDividerItemDecoration(this, LinearLayoutManager.
    HORIZONTAL, 16));
81          recyclerView.setItemAnimator(new DefaultItemAnimator());
82
83          recyclerView.setAdapter(mAdapter);
84
85          // row click listener
86          recyclerView.addOnItemTouchListener(new RecyclerTouchListener(getApplicationContext(),
    recyclerView, new RecyclerTouchListener.ClickListener() {
87              @Override
88              public void onClick(View view, int position) {
89                  AvailableDevice device = deviceList.get(position);
90
91                  BluetoothConnection.connectBluetoothDevice(position);
92              }
93
94              @Override
95              public void onLongClick(View view, int position) {
96
97              }
98          }));
99
100         BluetoothConnection.listAvailableBluetoothDevices();
101     }
102
103     @Override
104     public boolean onOptionsItemSelected(MenuItem item) {
105         if (item.getItemId() == android.R.id.home) // Press Back Icon
106         {
107             finish();
108         }
109
110         return super.onOptionsItemSelected(item);
111     }
112
113     public static enum bluetoothConnectionState { SYSTEM_DISCONNECTED, SYSTEM_CONNECTED,
    BLUETOOTH_DISABLED}
114
115     public static void setBluetoothStateGui(bluetoothConnectionState state){
116         switch (state){
117             case SYSTEM_CONNECTED:
118                 textView_bluetoothState.setText(R.string.system_connected_message);
119                 imgView_bluetoothStatusIcon.setImageResource(R.drawable.ic_done_icon);
120                 break;
121             case BLUETOOTH_DISABLED:
122                 textView_bluetoothState.setText(R.string.bluetooth_disabled_message);
123                 imgView_bluetoothStatusIcon.setImageResource(R.drawable.ic_bluetooth_disabled);
124                 break;
125             case SYSTEM_DISCONNECTED:
126                 textView_bluetoothState.setText(R.string.system_disconnected_message);
127                 imgView_bluetoothStatusIcon.setImageResource(R.drawable.ic_close_icon);
128                 break;
129         }
130     }
131
132
133 }
134
```

```java
package com.example.gui_design.bluetoothActivities;

import com.example.gui_design.R;

public class AvailableDevice {

    public enum DeviceState { DEFAULT, DISCONNECTED, CONNECTED, LOADING}
    private String deviceName;
    private DeviceState currentState;
    private int deviceStateIconID;

    public AvailableDevice() {
    }

    public AvailableDevice(String deviceName, DeviceState defaultState) {
        this.deviceName = deviceName;
        this.currentState = defaultState;
    }

    public void setCurrentState(DeviceState currentState) {
        this.currentState = currentState;
    }

    public DeviceState getCurrentState() {
        return currentState;
    }

    public void setDeviceName(String deviceName) {
        this.deviceName = deviceName;
    }

    public String getDeviceName() {
        return deviceName;
    }

    public int getDeviceStateIconID() {
        switch (currentState){
            case DEFAULT:
                return 0;
            case CONNECTED:
                return R.drawable.ic_done_icon;
            case LOADING:
                return R.drawable.loading_animation;
            case DISCONNECTED:
                return R.drawable.ic_close_icon;
        }
        return 0;
    }

    public int getTextAppearance(){
        switch (currentState){
            case DEFAULT:
                return R.style.btDeviceListAppearance;
            case CONNECTED:
                return R.style.btDeviceListSelectedAppearance;
            case LOADING:
                return R.style.btDeviceListAppearance;
            case DISCONNECTED:
                return R.style.btDeviceListAppearance;
        }
        return R.style.btDeviceListAppearance;
    }
}

```

File - C:_Library256\Feria_2019\Interfaz_grafica\app\src\main\java\com\example\gui_design\Application.java

```java
package com.example.gui_design;

import android.app.Activity;
import android.content.Context;
import android.widget.Toast;

public class Application extends android.app.Application {
    private static Activity mActivity;
    private static Context mContext;

    public static Activity getActivity() { return mActivity; }
    public static void setActivity(Activity mActivitySet) { mActivity = mActivitySet; }

    public static Context getContext() { return mContext; }
    public static void setContext(Context mContextoSet) { mContext = mContextoSet; }

    public static void toastMessage(final String message){
        mActivity.runOnUiThread(new Runnable() {
            @Override
            public void run() {
                Toast.makeText(mActivity, message, Toast.LENGTH_SHORT).show();
            }
        });
    }


}

```

File - C:_Library256\Feria_2019\Interfaz_grafica\app\src\main\java\com\example\gui_design\CommandsActivity.java

```java
package com.example.gui_design;

import android.graphics.PorterDuff;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import android.support.v4.content.ContextCompat;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.MenuItem;

public class CommandsActivity extends AppCompatActivity {
    private static Toolbar toolbar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.commands_layout);
        Application.setActivity(this);
        Application.setContext(this);

        toolbar = this.findViewById(R.id.toolbar);

        setSupportActionBar(toolbar);

        // add back arrow to toolbar
        if (getSupportActionBar() != null){
            getSupportActionBar().setDisplayHomeAsUpEnabled(true);
            getSupportActionBar().setDisplayShowHomeEnabled(true);

            final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
abc_ic_ab_back_material);
            upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP);
            getSupportActionBar().setHomeAsUpIndicator(upArrow);
        }
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getItemId() == android.R.id.home) // Press Back Icon
        {
            finish();
        }

        return super.onOptionsItemSelected(item);
    }
}

```

File - C:_Library256\Feria_2019\Interfaz_grafica\app\src\main\java\com\example\gui_design\MainActivity.java

```
 1 package com.example.gui_design;
 2
 3 import android.Manifest;
 4 import android.content.pm.PackageManager;
 5 import android.os.Bundle;
 6 import android.support.v4.app.ActivityCompat;
 7 import android.support.v4.content.ContextCompat;
 8 import android.support.v7.app.AppCompatActivity;
 9 import android.util.DisplayMetrics;
10 import android.view.View;
11 import android.widget.ImageButton;
12 import android.widget.TextView;
13
14 import com.example.gui_design.bluetoothActivities.BluetoothConnection;
15 import com.example.gui_design.settingsActivities.settings_layout;
16
17 public class MainActivity extends AppCompatActivity {
18
19     static ImageButton imgBtn_microphone;
20     static ImageButton imgBtn_microphoneBackground;
21     static ImageButton imgBtn_commands;
22     static ImageButton imgBtn_contacts;
23     static ImageButton imgBtn_bluetooth;
24     static ImageButton imgBtn_settings;
25     static ImageButton imgBtn_menu;
26     static MainMenuHandler mainMenuHandler;
27     static TextView textView_speechRecognitionResults;
28
29
30     @Override
31     protected void onCreate(Bundle savedInstanceState) {
32         super.onCreate(savedInstanceState);
33         setContentView(R.layout.activity_main);
34
35         if(savedInstanceState == null) {
36
37             mainMenuHandler = new MainMenuHandler(this, this);
38             Application.setActivity(this);
39             Application.setContext(this);
40             MainMenuHandler.dpiConstant = ((float) this.getResources().getDisplayMetrics().densityDpi
   / DisplayMetrics.DENSITY_DEFAULT);
41
42             imgBtn_microphone = findViewById(R.id.imgBtn_microphone);
43             imgBtn_microphoneBackground = findViewById(R.id.imgBtn_microphone_background);
44             imgBtn_commands = findViewById(R.id.imgBtn_commands);
45             imgBtn_contacts = findViewById(R.id.imgBtn_contacts);
46             imgBtn_bluetooth = findViewById(R.id.imgBtn_bluetooth);
47             imgBtn_settings = findViewById(R.id.imgBtn_settings);
48             imgBtn_menu = findViewById(R.id.imgBtn_menu);
49             textView_speechRecognitionResults = findViewById(R.id.textView_speechRecognitionResults);
50
51             if (ContextCompat.checkSelfPermission(this,
52                     Manifest.permission.RECORD_AUDIO)
53                     != PackageManager.PERMISSION_GRANTED) {
54                 if (ActivityCompat.shouldShowRequestPermissionRationale(this,
55                         Manifest.permission.RECORD_AUDIO)) {
56                 } else {
57                     ActivityCompat.requestPermissions(this,
58                             new String[]{Manifest.permission.RECORD_AUDIO},
59                             527);
60                 }
61             } else {
62                 GoogleSpeechRecognizer.setupRecognizer();
63                 PocketsphinxListener.setupRecognizer();
64                 PocketsphinxListener.startListening();
65
66             }
67
68             settings_layout.setupSettingsList();
69
70
71         }
72     }
73
74     @Override
75     public void onRequestPermissionsResult(int requestCode,
76                                            String permissions[], int[] grantResults) {
```

```
 77          switch (requestCode) {
 78              case 527: {
 79                  // If request is cancelled, the result arrays are empty.
 80                  if (grantResults.length > 0
 81                          && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
 82
 83                      GoogleSpeechRecognizer.setupRecognizer();
 84                      PocketsphinxListener.setupRecognizer();
 85
 86
 87                  } else { }
 88              }
 89          }
 90      }
 91
 92      public void onClickListener_menu(View view){ mainMenuHandler.menuButtonOnClick(); }
 93
 94      public void onClickListener_commands(View view){ mainMenuHandler.commandsButtonOnClick(); }
 95
 96      public void onClickListener_settings(View view){ mainMenuHandler.settingsButtonOnClick(); }
 97
 98      public void onClickListener_contacts(View view){ mainMenuHandler.contactsButtonOnClick(); }
 99
100      public void onClickListener_bluetooth(View view){ mainMenuHandler.bluetoothButtonOnClick(); }
101
102      public void onClickListener_microphone(View view){ MicrophoneHandler.microphoneOnClick(); }
103
104      @Override
105      public void onDestroy() {
106          super.onDestroy();
107          PocketsphinxListener.shutdownRecognizer();
108          GoogleSpeechRecognizer.shutdownRecognizer();
109          BluetoothConnection.closeAllConnections();
110      }
111
112  }
113
```

File - C:_Library256\Feria_2019\Interfaz_grafica\app\src\main\java\com\example\gui_design\MainMenuHandler.java

```java
package com.example.gui_design;

import android.animation.ValueAnimator;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.view.View;
import android.view.animation.AccelerateDecelerateInterpolator;

import com.example.gui_design.bluetoothActivities.BluetoothActivity;
import com.example.gui_design.settingsActivities.settings_layout;

public class MainMenuHandler extends MainActivity{

    public static float dpiConstant;
    public static float radialMenuRadius = 263.0f;
    public static boolean menuUncoiled = false;
    public static boolean isAnimationRunning = false;
    public static Context mainMenuContext;
    public static Activity mainMenuActivity;

    public MainMenuHandler(Context context, Activity activity){
        this.mainMenuActivity = activity;
        this.mainMenuContext = context;
    }

    public static void menuButtonOnClick(){
        long animationDuration = 850;

        if(isAnimationRunning==false) {
            isAnimationRunning = true;

            if (menuUncoiled == false) {
                ValueAnimator animatorCommands = ValueAnimator.ofFloat(360, 72);
                animatorCommands.setDuration(animationDuration);
                animatorCommands.setInterpolator(new AccelerateDecelerateInterpolator());
                animatorCommands.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
                    @Override
                    public void onAnimationUpdate(ValueAnimator animation) {
                        float value = ((Float) (animation.getAnimatedValue()))
                                .floatValue();

                        float translationX = (float) (Math.cos(value * (Math.PI / 180.0f)) * (
radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
                        float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.0d) -
radialMenuRadius * translationX));
                        if (value > 180.0f) {
                            translationY = -translationY;
                        }

                        imgBtn_commands.setTranslationX(convertDpToPixel(translationX));
                        imgBtn_commands.setTranslationY(convertDpToPixel(translationY));

                    }
                });
                imgBtn_commands.setVisibility(View.VISIBLE);
                animatorCommands.start();

                ValueAnimator animatorContacts = ValueAnimator.ofFloat(360, 144);
                animatorContacts.setDuration(animationDuration);
                animatorContacts.setInterpolator(new AccelerateDecelerateInterpolator());
                animatorContacts.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
                    @Override
                    public void onAnimationUpdate(ValueAnimator animation) {
                        float value = ((Float) (animation.getAnimatedValue()))
                                .floatValue();

                        float translationX = (float) (Math.cos(value * (Math.PI / 180.0f)) * (
radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
                        float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.0d) -
radialMenuRadius * translationX));
                        if (value > 180.0f) {
                            translationY = -translationY;
                        }

                        imgBtn_contacts.setTranslationX(convertDpToPixel(translationX));
                        imgBtn_contacts.setTranslationY(convertDpToPixel(translationY));
```

```
 74
 75                      }
 76                  });
 77                  imgBtn_contacts.setVisibility(View.VISIBLE);
 78                  animatorContacts.start();
 79
 80                  ValueAnimator animatorSettings = ValueAnimator.ofFloat(360, 216);
 81                  animatorSettings.setDuration(animationDuration);
 82                  animatorSettings.setInterpolator(new AccelerateDecelerateInterpolator());
 83                  animatorSettings.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
 84                      @Override
 85                      public void onAnimationUpdate(ValueAnimator animation) {
 86                          float value = ((Float) (animation.getAnimatedValue()))
 87                                  .floatValue();
 88
 89                          float translationX = (float) (Math.cos(value * (Math.PI / 180.0f)) * (
    radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
 90                          float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.0d) -
    radialMenuRadius * translationX));
 91                          if (value > 180.0f) {
 92                              translationY = -translationY;
 93                          }
 94
 95                          imgBtn_settings.setTranslationX(convertDpToPixel(translationX));
 96                          imgBtn_settings.setTranslationY(convertDpToPixel(translationY));
 97
 98                      }
 99                  });
100                  imgBtn_settings.setVisibility(View.VISIBLE);
101                  animatorSettings.start();
102
103                  ValueAnimator animatorBluetooth = ValueAnimator.ofFloat(360, 288);
104                  animatorBluetooth.setDuration(animationDuration);
105                  animatorBluetooth.setInterpolator(new AccelerateDecelerateInterpolator());
106                  animatorBluetooth.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
107                      @Override
108                      public void onAnimationUpdate(ValueAnimator animation) {
109                          float value = ((Float) (animation.getAnimatedValue()))
110                                  .floatValue();
111
112                          float translationX = (float) (Math.cos(value * (Math.PI / 180.0f)) * (
    radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
113                          float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.0d) -
    radialMenuRadius * translationX));
114                          if (value > 180.0f) {
115                              translationY = -translationY;
116                          }
117
118                          imgBtn_bluetooth.setTranslationX(convertDpToPixel(translationX));
119                          imgBtn_bluetooth.setTranslationY(convertDpToPixel(translationY));
120
121                      }
122                  });
123                  imgBtn_bluetooth.setVisibility(View.VISIBLE);
124                  animatorBluetooth.start();
125
126                  imgBtn_commands.animate().alpha(1.0f).setDuration(animationDuration).setInterpolator(
    new AccelerateDecelerateInterpolator());
127                  imgBtn_contacts.animate().alpha(1.0f).setDuration(animationDuration).setInterpolator(
    new AccelerateDecelerateInterpolator());
128                  imgBtn_settings.animate().alpha(1.0f).setDuration(animationDuration).setInterpolator(
    new AccelerateDecelerateInterpolator());
129                  imgBtn_bluetooth.animate().alpha(1.0f).setDuration(animationDuration).setInterpolator
    (new AccelerateDecelerateInterpolator()).withEndAction(new Runnable() {
130                      @Override
131                      public void run() {
132                          isAnimationRunning = false;
133                      }
134                  });
135
136                  menuUncoiled = true;
137
138
139              } else {
140
141
142                  ValueAnimator animatorCommands = ValueAnimator.ofFloat(72, 360);
```

```
143                 animatorCommands.setDuration(animationDuration);
144                 animatorCommands.setInterpolator(new AccelerateDecelerateInterpolator());
145                 animatorCommands.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
146                     @Override
147                     public void onAnimationUpdate(ValueAnimator animation) {
148                         float value = ((Float) (animation.getAnimatedValue()))
149                                 .floatValue();
150
151                         float translationX = (float) (Math.cos(value * (Math.PI / 180.0f)) * (
    radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
152                         float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.0d) -
    radialMenuRadius * translationX));
153                         if (value > 180.0f) {
154                             translationY = -translationY;
155                         }
156
157                         imgBtn_commands.setTranslationX(convertDpToPixel(translationX));
158                         imgBtn_commands.setTranslationY(convertDpToPixel(translationY));
159
160                     }
161                 });
162                 imgBtn_commands.setVisibility(View.VISIBLE);
163                 animatorCommands.start();
164
165                 ValueAnimator animatorContacts = ValueAnimator.ofFloat(144, 360);
166                 animatorContacts.setDuration(animationDuration);
167                 animatorContacts.setInterpolator(new AccelerateDecelerateInterpolator());
168                 animatorContacts.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
169                     @Override
170                     public void onAnimationUpdate(ValueAnimator animation) {
171                         float value = ((Float) (animation.getAnimatedValue()))
172                                 .floatValue();
173
174                         float translationX = (float) (Math.cos(value * (Math.PI / 180.0f)) * (
    radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
175                         float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.0d) -
    radialMenuRadius * translationX));
176                         if (value > 180.0f) {
177                             translationY = -translationY;
178                         }
179
180                         imgBtn_contacts.setTranslationX(convertDpToPixel(translationX));
181                         imgBtn_contacts.setTranslationY(convertDpToPixel(translationY));
182
183                     }
184                 });
185                 imgBtn_contacts.setVisibility(View.VISIBLE);
186                 animatorContacts.start();
187
188                 ValueAnimator animatorSettings = ValueAnimator.ofFloat(216, 360);
189                 animatorSettings.setDuration(animationDuration);
190                 animatorSettings.setInterpolator(new AccelerateDecelerateInterpolator());
191                 animatorSettings.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
192                     @Override
193                     public void onAnimationUpdate(ValueAnimator animation) {
194                         float value = ((Float) (animation.getAnimatedValue()))
195                                 .floatValue();
196
197                         float translationX = (float) (Math.cos(value * (Math.PI / 180.0f)) * (
    radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
198                         float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.0d) -
    radialMenuRadius * translationX));
199                         if (value > 180.0f) {
200                             translationY = -translationY;
201                         }
202
203                         imgBtn_settings.setTranslationX(convertDpToPixel(translationX));
204                         imgBtn_settings.setTranslationY(convertDpToPixel(translationY));
205
206                     }
207                 });
208                 imgBtn_settings.setVisibility(View.VISIBLE);
209                 animatorSettings.start();
210
211                 ValueAnimator animatorBluetooth = ValueAnimator.ofFloat(288, 360);
212                 animatorBluetooth.setDuration(animationDuration);
213                 animatorBluetooth.setInterpolator(new AccelerateDecelerateInterpolator());
```

```
214                 animatorBluetooth.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
215                     @Override
216                     public void onAnimationUpdate(ValueAnimator animation) {
217                         float value = ((Float) (animation.getAnimatedValue()))
218                                 .floatValue();
219
220                         float translationX = (float) (Math.cos(value * (Math.PI / 180.0f)) * (
    radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
221                         float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.0d) -
    radialMenuRadius * translationX));
222                         if (value > 180.0f) {
223                             translationY = -translationY;
224                         }
225
226                         imgBtn_bluetooth.setTranslationX(convertDpToPixel(translationX));
227                         imgBtn_bluetooth.setTranslationY(convertDpToPixel(translationY));
228
229                     }
230                 });
231                 imgBtn_bluetooth.setVisibility(View.VISIBLE);
232                 animatorBluetooth.start();
233
234                 imgBtn_commands.animate().alpha(0.0f).setDuration(animationDuration).setInterpolator(
    new AccelerateDecelerateInterpolator());
235                 imgBtn_contacts.animate().alpha(0.0f).setDuration(animationDuration).setInterpolator(
    new AccelerateDecelerateInterpolator());
236                 imgBtn_settings.animate().alpha(0.0f).setDuration(animationDuration).setInterpolator(
    new AccelerateDecelerateInterpolator());
237                 imgBtn_bluetooth.animate().alpha(0.0f).setDuration(animationDuration).setInterpolator
    (new AccelerateDecelerateInterpolator()).withEndAction(new Runnable() {
238                     @Override
239                     public void run() {
240                         isAnimationRunning = false;
241                     }
242                 });
243
244                 menuUncoiled = false;
245             }
246         }
247     }
248
249     public static void commandsButtonOnClick(){
250         Context currentContext = Application.getContext();
251         Activity currentActivity = Application.getActivity();
252         Intent i = new Intent(currentContext, CommandsActivity.class);
253         currentContext.startActivity(i);
254     }
255
256     public static void contactsButtonOnClick(){
257         Context currentContext = Application.getContext();
258         Activity currentActivity = Application.getActivity();
259         Intent i = new Intent(currentContext, ContactsActivity.class);
260         currentContext.startActivity(i);
261     }
262
263     public static void bluetoothButtonOnClick(){
264         Context currentContext = Application.getContext();
265         Activity currentActivity = Application.getActivity();
266         Intent i = new Intent(currentContext, BluetoothActivity.class);
267         currentContext.startActivity(i);
268     }
269
270     public static void settingsButtonOnClick(){
271         Intent i = new Intent(Application.getContext(), settings_layout.class);
272         mainMenuContext.startActivity(i);
273         //mainMenuActivity.overridePendingTransition(R.anim.fade_in, R.anim.fade_out);
274     }
275
276
277
278
279
280
281
282
283     /**
284      * This method converts dp unit to equivalent pixels, depending on device density.
```

```
285      *
286      * @param dp A value in dp (density independent pixels) unit. Which we need to convert into
    pixels
287      * @return A float value to represent px equivalent to dp depending on device density
288      */
289     public static float convertDpToPixel(float dp){ return dp * dpiConstant; }
290
291     /**
292      * This method converts device specific pixels to density independent pixels.
293      *
294      * @param px A value in px (pixels) unit. Which we need to convert into db
295      * @return A float value to represent dp equivalent to px value
296      */
297     public static float convertPixelsToDp(float px){
298         return px / dpiConstant;
299     }
300 }
301
```

File - C:_Library256\Feria_2019\Interfaz_grafica\app\src\main\java\com\example\gui_design\MicrophoneHandler.java

```
1  package com.example.gui_design;
2
3  import android.graphics.PorterDuff;
4  import android.view.animation.AccelerateDecelerateInterpolator;
5
6  import java.util.concurrent.Callable;
7  import java.util.concurrent.ExecutorService;
8  import java.util.concurrent.Executors;
9  import java.util.concurrent.Future;
10
11 public class MicrophoneHandler extends MainActivity {
12
13     public enum MicrophoneState { INHIBITED, LISTENING_KEYOWRD, LISTENING_COMMAND}
14
15     static private MicrophoneState currentState = MicrophoneState.LISTENING_KEYOWRD;
16
17     public static void microphoneOnClick(){
18         switch (currentState){
19             case LISTENING_KEYOWRD:
20                 setState(MicrophoneState.INHIBITED);
21                 break;
22             case LISTENING_COMMAND:
23                 GoogleSpeechRecognizer.forceStopRecognizer();
24                 setState(MicrophoneState.LISTENING_KEYOWRD);
25                 break;
26             case INHIBITED:
27                 setState(MicrophoneState.LISTENING_KEYOWRD);
28                 break;
29         }
30
31     }
32
33     public static MicrophoneState getState(){
34         return currentState;
35     }
36
37     public static void setState(MicrophoneState desiredState){
38         switch (desiredState){
39             case INHIBITED:
40                 imgBtn_microphoneBackground.clearColorFilter();
41                 imgBtn_microphoneBackground.setColorFilter(Application.getContext().getColor(R.color.
   ColorBackMicInhibited), PorterDuff.Mode.SRC_ATOP);
42                 imgBtn_microphone.setImageResource(R.drawable.ic_main_mic_off);
43                 endAnimation();
44                 currentState = MicrophoneState.INHIBITED;
45                 break;
46             case LISTENING_KEYOWRD:
47                 imgBtn_microphoneBackground.clearColorFilter();
48                 imgBtn_microphoneBackground.setColorFilter(Application.getContext().getColor(R.color.
   ColorBackMicKeyword), PorterDuff.Mode.SRC_ATOP);
49                 imgBtn_microphone.setImageResource(R.drawable.ic_main_mic_listening);
50                 endAnimation();
51                 currentState = MicrophoneState.LISTENING_KEYOWRD;
52                 break;
53             case LISTENING_COMMAND:
54                 imgBtn_microphoneBackground.clearColorFilter();
55                 imgBtn_microphoneBackground.setColorFilter(Application.getContext().getColor(R.color.
   ColorBackMicCommand), PorterDuff.Mode.SRC_ATOP);
56                 imgBtn_microphone.setImageResource(R.drawable.ic_main_mic_listening);
57                 startAnimation();
58                 currentState = MicrophoneState.LISTENING_COMMAND;
59                 break;
60         }
61     }
62
63     private static ExecutorService executor = Executors.newSingleThreadExecutor();
64     private static boolean terminateAnimation = false;
65     private static boolean animationHasFinished = false;
66
67     public static void startAnimation(){
68         terminateAnimation = false;
69         Future<Void> future = executor.submit(new Callable<Void>() {
70             public Void call() throws Exception {
71                 try {
72                     while (terminateAnimation == false) {
73                         imgBtn_microphoneBackground.animate().scaleX(0.90f).setDuration(350).
   setInterpolator(new AccelerateDecelerateInterpolator());
```

File - C:_Library256\Feria_2019\Interfaz_grafica\app\src\main\java\com\example\gui_design\MicrophoneHandler.java

```
 74                         imgBtn_microphone.animate().scaleX(0.90f).setDuration(350).setInterpolator(
    new AccelerateDecelerateInterpolator());
 75                         imgBtn_microphoneBackground.animate().scaleY(0.90f).setDuration(350).
    setInterpolator(new AccelerateDecelerateInterpolator());
 76                         imgBtn_microphone.animate().scaleY(0.90f).setDuration(350).setInterpolator(
    new AccelerateDecelerateInterpolator()).withEndAction(new Runnable() {
 77                             @Override
 78                             public void run() {
 79                                 animationHasFinished = true;
 80                             }
 81                         });
 82                         while(animationHasFinished==false);
 83                         animationHasFinished= false;
 84                         imgBtn_microphoneBackground.animate().scaleX(1.00f).setDuration(350).
    setInterpolator(new AccelerateDecelerateInterpolator());
 85                         imgBtn_microphone.animate().scaleX(1.00f).setDuration(350).setInterpolator(
    new AccelerateDecelerateInterpolator());
 86                         imgBtn_microphoneBackground.animate().scaleY(1.00f).setDuration(350).
    setInterpolator(new AccelerateDecelerateInterpolator());
 87                         imgBtn_microphone.animate().scaleY(1.00f).setDuration(350).setInterpolator(
    new AccelerateDecelerateInterpolator()).withEndAction(new Runnable() {
 88                             @Override
 89                             public void run() {
 90                                 animationHasFinished = true;
 91                             }
 92                         });;
 93                         while(animationHasFinished==false);
 94                         animationHasFinished = false;
 95                     }
 96                     terminateAnimation = false;
 97                 }catch (Throwable th){ ;
 98                     th.printStackTrace();
 99                 }
100                 return null;
101             }
102         });
103     }
104
105     public static void endAnimation(){
106         terminateAnimation = true;
107     }
108
109 }
110
```

File - C:_Library256\Feria_2019\Interfaz_grafica\app\src\main\java\com\example\gui_design\PocketsphinxListener.java

```java
package com.example.gui_design;

import java.io.File;

import edu.cmu.pocketsphinx.Assets;
import edu.cmu.pocketsphinx.Hypothesis;
import edu.cmu.pocketsphinx.RecognitionListener;
import edu.cmu.pocketsphinx.SpeechRecognizer;
import edu.cmu.pocketsphinx.SpeechRecognizerSetup;

public final class PocketsphinxListener implements RecognitionListener {
    private static SpeechRecognizer recognizer;
    private static String KEYPHRASE = "asistente";
    private static final String KWS_SEARCH = "wakeup";


    @Override
    public void onPartialResult(Hypothesis hypothesis) {
        if (hypothesis == null)
            return;

        String text = hypothesis.getHypstr();

        if (text.contains(KEYPHRASE)) {

            Application.toastMessage(KEYPHRASE);
            recognizer.stop();
            //recognizer.cancel();
            //recognizer.shutdown();
            GoogleSpeechRecognizer.startSpeechListening();
            //recognizer.startListening(KWS_SEARCH);
        }

    }


    @Override
    public void onResult(Hypothesis hypothesis) {
        if (hypothesis != null) {
            String text = hypothesis.getHypstr();

        }
    }

    @Override
    public void onBeginningOfSpeech() {
    }

    @Override
    public void onEndOfSpeech() {

    }

    public static void setupRecognizer(){
        try {
            Assets assets = new Assets(Application.getContext());
            File assetDir = assets.syncAssets();
            recognizer = SpeechRecognizerSetup.defaultSetup()
                    .setAcousticModel(new File(assetDir, "es-es-ptm"))
                    .setDictionary(new File(assetDir, "cmudict-es-es.dict"))
                    .setKeywordThreshold(Float.MIN_VALUE
                    )
                    .setRawLogDir(assetDir) // To disable logging of raw audio comment out this call (takes a lot of space on the device)

                    .getRecognizer();
            recognizer.addListener(new PocketsphinxListener());
            recognizer.addKeyphraseSearch(KWS_SEARCH, KEYPHRASE);
        } catch (Throwable th){
            Application.toastMessage(th.toString());
        }
    }

    public static void startListening(){
        recognizer.startListening(KWS_SEARCH);
    }

```

File - C:_Library256\Feria_2019\Interfaz_grafica\app\src\main\java\com\example\gui_design\PocketsphinxListener.java

```
 77      public static void shutdownRecognizer(){
 78          if (recognizer != null) {
 79              recognizer.cancel();
 80              recognizer.shutdown();
 81          }
 82      }
 83
 84      public static void stopRecognizer(){
 85          recognizer.stop();
 86      }
 87
 88      @Override
 89      public void onError(Exception error) {
 90          Application.toastMessage("Error: " + error.toString());
 91      }
 92
 93      @Override
 94      public void onTimeout() {
 95          Application.toastMessage("Timeout!");
 96      }
 97
 98
 99
100
101 }
102
```