# Codigo de fuente: Aplicación, todas las clases internas

## Lenguaje: JAVA 8 SE
## IDE: Android Studio

File - C:_Library256\Feria_2019\Interfaz_grafica\app\src\main\java\com\example\gui_design\ContactsActivity.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 ContactsActivity extends AppCompatActivity {
    private static Toolbar toolbar;

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

        TextToSpeechHandler.initializeTextToSpeech();
        try {
            Thread.sleep(250);
        }catch (Throwable th) { Application.toastMessage(th.toString());}

        TextToSpeechHandler.addToSpeakingQueue("hola");

    }

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

        return super.onOptionsItemSelected(item);
    }
}

```

```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 com.example.gui_design.Application;

import java.io.InputStream;
import java.io.OutputStream;
import java.nio.charset.StandardCharsets;
import java.util.ArrayList;
import java.util.List;
import java.util.Set;
import java.util.UUID;
import java.util.concurrent.Executor;
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 = BluetoothAdapter.getDefaultAdapter();
    public static BluetoothDevice targetDevice;
    public static BluetoothDevice[] bluetoothDevicesArray;
    public static List<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;

    private static Executor bluetoothConnectorExecutor = Executors.newSingleThreadExecutor();
    private static Boolean isListAvailableBluetoothDevicesRunning = false;

    public static void listAvailableBluetoothDevices(){
        if ((!isListAvailableBluetoothDevicesRunning)) {
            try {
                BluetoothActivity.setBluetoothStateGui(bluetoothConnectionState.WAITING);
                isListAvailableBluetoothDevicesRunning = true;
                bluetoothConnectorExecutor.execute(new Runnable() {
                    @Override
                    public void run() {
                        if (btAdapter != null) {
                            if (btAdapter.isEnabled()) {
                                Set<BluetoothDevice> bondedDevices = btAdapter.getBondedDevices();

                                bluetoothDevicesList = new ArrayList<>();
                                bluetoothDevicesList.addAll(bondedDevices);

                                if (bondedDevices.size() > 0) {
                                    bluetoothDevicesArray = bondedDevices.toArray(new BluetoothDevice[
bondedDevices.size()]);
                                }
                                updateBluetoothListGUI();
                            } else {
                                BluetoothActivity.setBluetoothStateGui(bluetoothConnectionState.
BLUETOOTH_DISABLED);
                            }
                        }
                        isListAvailableBluetoothDevicesRunning = false;
                    }
                });
            }catch (Throwable th)
            {
                Application.toastMessage(th.toString()); th.printStackTrace();
            }
        }
```

```
 76      }
 77
 78      public static Boolean isBluetoothAdapterReady(){
 79          return ((btAdapter != null)&&(btAdapter.isEnabled()));
 80      }
 81
 82      public static void updateBluetoothListGUI(){
 83          if (BluetoothActivity.mAdapter!=null) {
 84              BluetoothActivity.deviceList.clear();
 85              for (int x = 0; x < bluetoothDevicesArray.length; x++) {
 86                  BluetoothActivity.deviceList.add(new AvailableDevice(bluetoothDevicesArray[x].getName
    (), AvailableDevice.DeviceState.DEFAULT));
 87              }
 88
 89              if (bluetoothConnected) {
 90                  BluetoothActivity.setBluetoothStateGui(bluetoothConnectionState.SYSTEM_CONNECTED);
 91                  deviceList.get(bluetoothDevicesList.indexOf(targetDevice)).setCurrentState(
    AvailableDevice.DeviceState.CONNECTED);
 92              }
 93              else {
 94                  BluetoothActivity.setBluetoothStateGui(bluetoothConnectionState.SYSTEM_DISCONNECTED);
 95              }
 96
 97              BluetoothActivity.updateDevicesList();
 98          }
 99      }
100
101      private static Boolean attemptingBluetoothConnection = false;
102
103
104      public static void connectBluetoothDeviceSync(int deviceIndex){
105          if ((!isListAvailableBluetoothDevicesRunning)&&(!attemptingBluetoothConnection)&&
    isBluetoothAdapterReady()) {
106              try {
107                  attemptingBluetoothConnection = true;
108                  if (bluetoothConnected) closeAllConnections();
109                  BluetoothActivity.setBluetoothStateGui(bluetoothConnectionState.WAITING);
110                  targetDevice = bluetoothDevicesArray[deviceIndex];
111                  btSocket = targetDevice.createRfcommSocketToServiceRecord(btUUID);
112                  btSocket.connect();
113                  if (btSocket.isConnected()) {
114                      inputStream = btSocket.getInputStream();
115                      outputStream = btSocket.getOutputStream();
116                      bluetoothConnected = true;
117                      BluetoothActivity.setBluetoothStateGui(bluetoothConnectionState.SYSTEM_CONNECTED)
    ;
118                      BluetoothActivity.updateDevicesList();
119                  } else {
120                      bluetoothConnected = false;
121                      BluetoothActivity.setBluetoothStateGui(bluetoothConnectionState.
    SYSTEM_DISCONNECTED);
122                  }
123                  attemptingBluetoothConnection = false;
124              } catch (Throwable th) {
125                  bluetoothConnected = false;
126                  Application.toastMessage(th.toString());
127                  BluetoothActivity.setBluetoothStateGui(bluetoothConnectionState.SYSTEM_DISCONNECTED);
128                  attemptingBluetoothConnection = false;
129              }
130          }
131      }
132
133      private static int attemptingConnectionDeviceIndex;
134
135      public static void connectBluetoothDeviceAsync(int deviceIndex){
136          attemptingConnectionDeviceIndex = deviceIndex;
137          if ((!isListAvailableBluetoothDevicesRunning)&&(!attemptingBluetoothConnection)&&
    isBluetoothAdapterReady()) {
138              bluetoothConnectorExecutor.execute(new Runnable() {
139                  @Override
140                  public void run() {
141                      connectBluetoothDeviceSync(attemptingConnectionDeviceIndex);
142                  }
143              });
144          }
145      }
146
```

```
147
148
149
150
151
152     private static Executor autoConnectExecutor = Executors.newSingleThreadExecutor();
153     private static Boolean terminateAutoConnect = false;
154     private static Boolean isAutoConnectRunning = false;
155
156     public static void initBTAutoConnectService(){
157         if (isAutoConnectRunning() == false) {
158             autoConnectExecutor.execute(new Runnable() {
159                 @Override
160                 public void run() {
161                     isAutoConnectRunning = true;
162                     try { Thread.sleep(5000);} catch (Throwable th) { th.printStackTrace(); }
163                     while (!terminateAutoConnect && isBluetoothAdapterReady()) {
164                         listAvailableBluetoothDevices();
165                         while (isListAvailableBluetoothDevicesRunning) {}
166                         for (int x = 0; x < bluetoothDevicesArray.length; x++) {
167                             // MAC ADDRESS HC-05 : 98:D3:91:FD:38:05
168                             // MAC ADDRESS XT1068 : 5C:51:88:A2:38:97
169                             if (bluetoothDevicesArray[x].getAddress().compareToIgnoreCase("98:D3:91:
    FD:38:05") == 0) {
170                                 if (!terminateAutoConnect&&!bluetoothConnected)
    connectBluetoothDeviceSync(x);
171                             }
172                         }
173                         try { Thread.sleep(10000);} catch (Throwable th) { th.printStackTrace(); }
174                     }
175                     isAutoConnectRunning = false;
176                     terminateAutoConnect = false;
177                 }
178             });
179         }else{
180             terminateAutoConnect = false;
181         }
182     }
183
184     public static void terminateBTAutoConnect(){
185         terminateAutoConnect = true;
186     }
187
188     public static Boolean isAutoConnectRunning(){
189         return isAutoConnectRunning;
190     }
191
192
193
194
195     public static void connectionLost(){
196
197     }
198
199     public static void closeAllConnections(){
200         try {
201             bluetoothConnected = false;
202             if (bluetoothComThread != null) bluetoothComThread.cancel(true);
203             if (btSocket != null) btSocket.close();
204             if (outputStream != null) outputStream.close();
205             if (inputStream != null) inputStream.close();
206
207             btSocket = null;
208             outputStream = null;
209             inputStream = null;
210         } catch (Throwable th){
211             th.printStackTrace();
212         }
213     }
214
215     public enum commandType {TURN_RELAY_ON, TURN_RELAY_OFF, TURN_EVERYTHING_ON, TURN_EVERYTHING_OFF,
    CALL_NURSE}
216
217     public static boolean composeAndSend(commandType cmdType, int parameter){
218         if (bluetoothConnected) {
219             try {
220                 String commandOutput = null;
```

```
221                 switch (cmdType) {
222                     case TURN_RELAY_ON:
223                         commandOutput = String.format("/TURN_RELAY_ON[%s]$", Integer.toString(
    parameter));
224                         break;
225                     case TURN_RELAY_OFF:
226                         commandOutput = String.format("/TURN_RELAY_OFF[%s]$", Integer.toString(
    parameter));
227                         break;
228                     case TURN_EVERYTHING_ON:
229                         commandOutput = "/TURN_EVERYTHING_ON$";
230                         break;
231                     case TURN_EVERYTHING_OFF:
232                         commandOutput = "/TURN_EVERYTHING_OFF$";
233                         break;
234                     case CALL_NURSE:
235                         commandOutput = "/CALL_NURSE$";
236                         break;
237                 }
238 
239                 outputStream.write(commandOutput.getBytes(StandardCharsets.US_ASCII));
240                 Application.toastMessage(commandOutput);
241                 return true;
242             } catch (Throwable th) {
243                 Application.toastMessage(th.toString());
244                 return false;
245             }
246         } else { return false; }
247     }
248 
249     public static BroadcastReceiver bluetoothAdapterStatusBR = new BroadcastReceiver() {
250         @Override
251         public void onReceive(Context context, Intent intent) {
252             final String action = intent.getAction();
253             if (action.equals(BluetoothAdapter.ACTION_STATE_CHANGED)){
254                 final int stateCode = intent.getIntExtra(BluetoothAdapter.EXTRA_STATE,
    BluetoothAdapter.ERROR);
255 
256                 switch (stateCode){
257                     case BluetoothAdapter.STATE_OFF:
258                         //Bluetooth adapter turned off handler
259                         BluetoothActivity.setBluetoothStateGui(bluetoothConnectionState.
    BLUETOOTH_DISABLED);
260                         break;
261                     case BluetoothAdapter.STATE_TURNING_OFF:
262                         //Bluetooth adapter turning off handler
263                         terminateBTAutoConnect();
264                         BluetoothActivity.setBluetoothStateGui(bluetoothConnectionState.
    BT_TURNING_OFF);
265                         break;
266                     case BluetoothAdapter.STATE_ON:
267                         BluetoothActivity.setBluetoothStateGui(bluetoothConnectionState.
    SYSTEM_DISCONNECTED);
268                         updateDevicesList();
269                         initBTAutoConnectService();
270 
271                         break;
272                     case BluetoothAdapter.STATE_TURNING_ON:
273                         //Bluetooth adapter turning on handler
274                         BluetoothActivity.setBluetoothStateGui(bluetoothConnectionState.BT_TURNING_ON
    );
275                         break;
276                 }
277             }
278         }
279     };
280 
281     public static BroadcastReceiver bluetoothScanModeStatusBR = new BroadcastReceiver() {
282 
283         @Override
284         public void onReceive(Context context, Intent intent) {
285             final String action = intent.getAction();
286 
287             if (action.equals(BluetoothAdapter.ACTION_SCAN_MODE_CHANGED)) {
288 
289                 int mode = intent.getIntExtra(BluetoothAdapter.EXTRA_SCAN_MODE, BluetoothAdapter.
    ERROR);
```

```
290
291                 switch (mode) {
292                     case BluetoothAdapter.SCAN_MODE_CONNECTABLE_DISCOVERABLE:
293
294                         break;
295                     case BluetoothAdapter.SCAN_MODE_CONNECTABLE:
296
297                         break;
298                     case BluetoothAdapter.SCAN_MODE_NONE:
299
300                         break;
301                 }
302             }
303         }
304     };
305
306     public static BroadcastReceiver bluetoothDeviceStatusBR = new BroadcastReceiver() {
307         @Override
308         public void onReceive(Context context, Intent intent) {
309             String action = intent.getAction();
310
311             switch (action){
312                 case BluetoothDevice.ACTION_ACL_CONNECTED:
313                     terminateBTAutoConnect();
314                     updateDevicesList();
315                     break;
316                 case BluetoothDevice.ACTION_ACL_DISCONNECTED:
317                     closeAllConnections();
318                     BluetoothActivity.setBluetoothStateGui(bluetoothConnectionState.
    SYSTEM_DISCONNECTED);
319                     initBTAutoConnectService();
320                     updateBluetoothListGUI();
321                     //Application.toastMessage("Dispositivo desconectado!");
322                     break;
323                 case BluetoothDevice.ACTION_ACL_DISCONNECT_REQUESTED:
324
325                     break;
326                 case BluetoothDevice.ACTION_BOND_STATE_CHANGED:
327
328                     break;
329             }
330         }
331     };
332
333 }
334
```

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

import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.graphics.PorterDuff;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import android.os.Looper;
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.ProgressBar;
import android.widget.TextView;

import com.example.gui_design.Application;
import com.example.gui_design.R;
import com.example.gui_design.settingsActivities.BluetoothSettingsActivity;
import com.example.gui_design.settingsActivities.RecyclerTouchListener;
import com.github.ybq.android.spinkit.sprite.Sprite;
import com.github.ybq.android.spinkit.style.Circle;

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;
    public static ProgressBar loading_progressBar;

    public static Boolean guiInitialized = false;


    @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);
        loading_progressBar = this.findViewById(R.id.loading_progressBar);

        Sprite fadingCircle = new Circle();
        fadingCircle.setColor(Color.WHITE);
        loading_progressBar.setIndeterminateDrawable(fadingCircle);

        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);
```

```
 77
 78          mAdapter = new DeviceListAdapter(deviceList);
 79
 80          recyclerView.setHasFixedSize(true);
 81
 82          // vertical RecyclerView
 83          // keep movie_list_row.xml width to `match_parent`
 84          RecyclerView.LayoutManager mLayoutManager = new LinearLayoutManager(getApplicationContext());
 85
 86          // horizontal RecyclerView
 87          // keep movie_list_row.xml width to `wrap_content`
 88          // RecyclerView.LayoutManager mLayoutManager = new LinearLayoutManager(getApplicationContext
    (), LinearLayoutManager.HORIZONTAL, false);
 89
 90          recyclerView.setLayoutManager(mLayoutManager);
 91
 92          // adding inbuilt divider line
 93          recyclerView.addItemDecoration(new DividerItemDecoration(this, LinearLayoutManager.VERTICAL))
    ;
 94
 95          // adding custom divider line with padding 16dp
 96          //recyclerView.addItemDecoration(new MyDividerItemDecoration(this, LinearLayoutManager.
    HORIZONTAL, 16));
 97          recyclerView.setItemAnimator(new DefaultItemAnimator());
 98
 99          recyclerView.setAdapter(mAdapter);
100
101          // row click listener
102          recyclerView.addOnItemTouchListener(new RecyclerTouchListener(getApplicationContext(),
    recyclerView, new RecyclerTouchListener.ClickListener() {
103              @Override
104              public void onClick(View view, int position) {
105                  AvailableDevice device = deviceList.get(position);
106
107                  BluetoothConnection.connectBluetoothDeviceAsync(position);
108              }
109
110              @Override
111              public void onLongClick(View view, int position) {
112
113              }
114          }));
115
116          guiInitialized = true;
117
118          BluetoothConnection.listAvailableBluetoothDevices();
119
120      }
121
122      @Override
123      public boolean onOptionsItemSelected(MenuItem item) {
124          if (item.getItemId() == android.R.id.home) // Press Back Icon
125          {
126              finish();
127          }
128
129          return super.onOptionsItemSelected(item);
130      }
131
132      public static void updateDevicesList(){
133          if (guiInitialized) {
134              Application.getActivity().runOnUiThread(new Runnable() {
135                  @Override
136                  public void run() {
137                      BluetoothActivity.mAdapter.notifyDataSetChanged();
138                  }
139              });
140          }
141      }
142
143      public void configBtn_onClick(View view){
144          Context currentContext = Application.getContext();
145          Intent i = new Intent(currentContext, BluetoothSettingsActivity.class);
146          currentContext.startActivity(i);
147      }
148
149
```

```
150     public enum bluetoothConnectionState { SYSTEM_DISCONNECTED, SYSTEM_CONNECTED, BLUETOOTH_DISABLED,
     WAITING, BT_TURNING_OFF, BT_TURNING_ON, UNDEFINED}
151     public static bluetoothConnectionState currentBluetoothState = bluetoothConnectionState.UNDEFINED
    ;
152
153     public static void setBluetoothStateGui(bluetoothConnectionState state){
154         if (guiInitialized) {
155             currentBluetoothState = state;
156
157             if (Looper.myLooper() == Looper.getMainLooper()) {
158                 switch (currentBluetoothState) {
159                     case SYSTEM_CONNECTED:
160                         textView_bluetoothState.setText(R.string.system_connected_message);
161                         imgView_bluetoothStatusIcon.setImageResource(R.drawable.ic_done_icon);
162                         imgView_bluetoothStatusIcon.setVisibility(View.VISIBLE);
163                         loading_progressBar.setVisibility(View.GONE);
164                         break;
165                     case BLUETOOTH_DISABLED:
166                         textView_bluetoothState.setText(R.string.bluetooth_disabled_message);
167                         imgView_bluetoothStatusIcon.setImageResource(R.drawable.ic_bluetooth_disabled
    );
168                         imgView_bluetoothStatusIcon.setVisibility(View.VISIBLE);
169                         loading_progressBar.setVisibility(View.GONE);
170                         break;
171                     case BT_TURNING_ON:
172                         textView_bluetoothState.setText(R.string.bluetooth_initializing_adapater);
173                         imgView_bluetoothStatusIcon.setVisibility(View.GONE);
174                         loading_progressBar.setVisibility(View.VISIBLE);
175                         break;
176                     case BT_TURNING_OFF:
177                         textView_bluetoothState.setText(R.string.bluetooth_closing_adapter);
178                         imgView_bluetoothStatusIcon.setVisibility(View.GONE);
179                         loading_progressBar.setVisibility(View.VISIBLE);
180                         break;
181                     case SYSTEM_DISCONNECTED:
182                         textView_bluetoothState.setText(R.string.system_disconnected_message);
183                         imgView_bluetoothStatusIcon.setImageResource(R.drawable.ic_close_icon);
184                         imgView_bluetoothStatusIcon.setVisibility(View.VISIBLE);
185                         loading_progressBar.setVisibility(View.GONE);
186                         break;
187                     case WAITING:
188                         textView_bluetoothState.setText(R.string.
    bluetooth_attempting_connection_message);
189                         imgView_bluetoothStatusIcon.setVisibility(View.GONE);
190                         loading_progressBar.setVisibility(View.VISIBLE);
191                         break;
192                 }
193             } else {
194                 Application.getActivity().runOnUiThread(new Runnable() {
195                     @Override
196                     public void run() {
197                         switch (currentBluetoothState) {
198                             case SYSTEM_CONNECTED:
199                                 textView_bluetoothState.setText(R.string.system_connected_message);
200                                 imgView_bluetoothStatusIcon.setImageResource(R.drawable.ic_done_icon)
    ;
201                                 imgView_bluetoothStatusIcon.setVisibility(View.VISIBLE);
202                                 loading_progressBar.setVisibility(View.GONE);
203                                 break;
204                             case BLUETOOTH_DISABLED:
205                                 textView_bluetoothState.setText(R.string.bluetooth_disabled_message);
206                                 imgView_bluetoothStatusIcon.setImageResource(R.drawable.
    ic_bluetooth_disabled);
207                                 imgView_bluetoothStatusIcon.setVisibility(View.VISIBLE);
208                                 loading_progressBar.setVisibility(View.GONE);
209                                 break;
210                             case BT_TURNING_ON:
211                                 textView_bluetoothState.setText(R.string.
    bluetooth_initializing_adapater);
212                                 imgView_bluetoothStatusIcon.setVisibility(View.GONE);
213                                 loading_progressBar.setVisibility(View.VISIBLE);
214                                 break;
215                             case BT_TURNING_OFF:
216                                 textView_bluetoothState.setText(R.string.bluetooth_closing_adapter);
217                                 imgView_bluetoothStatusIcon.setVisibility(View.GONE);
218                                 loading_progressBar.setVisibility(View.GONE);
219                                 break;
```

```
220                             case SYSTEM_DISCONNECTED:
221                                 textView_bluetoothState.setText(R.string.system_disconnected_message)
    ;
222                                 imgView_bluetoothStatusIcon.setImageResource(R.drawable.ic_close_icon
    );
223                                 imgView_bluetoothStatusIcon.setVisibility(View.VISIBLE);
224                                 loading_progressBar.setVisibility(View.GONE);
225                                 break;
226                             case WAITING:
227                                 textView_bluetoothState.setText(R.string.
    bluetooth_attempting_connection_message);
228                                 imgView_bluetoothStatusIcon.setVisibility(View.GONE);
229                                 loading_progressBar.setVisibility(View.VISIBLE);
230                                 break;
231                         }
232                     }
233                 });
234             }
235         }
236     }
237
238
239 }
240
```

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.arch.lifecycle.LifecycleObserver;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.Context;
import android.content.IntentFilter;

import com.example.gui_design.bluetoothActivities.BluetoothConnection;

import java.io.IOException;

public class Application extends android.app.Application implements LifecycleObserver {

    @Override
    public void onCreate() {
        super.onCreate();

        // ProcessLifecycleOwner.get().getLifecycle().addObserver(this);

        IntentFilter bluetoothAdapterActionChangedFilter = new IntentFilter(BluetoothAdapter.ACTION_STATE_CHANGED);
        registerReceiver(BluetoothConnection.bluetoothAdapterStatusBR, bluetoothAdapterActionChangedFilter);

        IntentFilter bluetoothScanModeStatusChangedFilter = new IntentFilter();
        bluetoothScanModeStatusChangedFilter.addAction(BluetoothAdapter.ACTION_DISCOVERY_STARTED);
        bluetoothScanModeStatusChangedFilter.addAction(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
        bluetoothScanModeStatusChangedFilter.addAction(BluetoothAdapter.ACTION_SCAN_MODE_CHANGED);
        registerReceiver(BluetoothConnection.bluetoothScanModeStatusBR, bluetoothScanModeStatusChangedFilter);

        IntentFilter bluetoothDeviceStatusChangedFilter = new IntentFilter();
        bluetoothDeviceStatusChangedFilter.addAction(BluetoothDevice.ACTION_ACL_CONNECTED);
        bluetoothDeviceStatusChangedFilter.addAction(BluetoothDevice.ACTION_ACL_DISCONNECTED);
        bluetoothDeviceStatusChangedFilter.addAction(BluetoothDevice.ACTION_ACL_DISCONNECT_REQUESTED);
        bluetoothDeviceStatusChangedFilter.addAction(BluetoothDevice.ACTION_BOND_STATE_CHANGED);
        registerReceiver(BluetoothConnection.bluetoothDeviceStatusBR, bluetoothDeviceStatusChangedFilter);

        Thread internetDetector = new Thread(){
            @Override
            public void run() {
                super.run();
                while(true){
                    try {
                        internetConnectionStatus = isConnected();
                        Thread.sleep(10000);
                        if (internetConnectionStatus) {Application.toastMessage("Internet connection detected");}
                        else {Application.toastMessage("Proper internet connection has not been detected");}
                    } catch (Throwable th) { th.printStackTrace(); Application.toastMessage(th.toString());}
                }
            }
        };

        internetDetector.start();

    }

    @Override
    public void onTerminate() {
        super.onTerminate();
    }

    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; }

```

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

```
71      public static void toastMessage(final String message){
72          mActivity.runOnUiThread(new Runnable() {
73              @Override
74              public void run() {
75                  if (mActivity != null) {
76                      //Toast.makeText(mActivity, message, Toast.LENGTH_SHORT).show();
77                  }
78              }
79          });
80      }
81
82      private static boolean internetConnectionStatus;
83
84      public static boolean isSystemOnline(){
85          return internetConnectionStatus;
86      }
87
88      public boolean isConnected() throws InterruptedException, IOException {
89          final String command = "ping -c 1 google.com";
90          return Runtime.getRuntime().exec(command).waitFor() == 0;
91      }
92
93
94
95
96
97
98  }
99
```

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

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;

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

public class InternetConectivityActivity extends AppCompatActivity {
    private static Toolbar toolbar;

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

```

```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;
    }
}

```

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

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;

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

public class CommandsSettingsActivity extends AppCompatActivity {
    private static Toolbar toolbar;

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

```

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

import android.graphics.PorterDuff;
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 SettingsLayoutAdapter extends RecyclerView.Adapter<SettingsLayoutAdapter.viewHolder> {

    private List<AvailableSetting> settingsList;

    public class viewHolder extends RecyclerView.ViewHolder {
        public TextView setting_title, setting_subtitle;
        public ImageView setting_icon;

        public viewHolder(View view) {
            super(view);
            setting_title = (TextView) view.findViewById(R.id.setting_title);
            setting_subtitle = (TextView) view.findViewById(R.id.setting_subtitle);
            setting_icon = (ImageView) view.findViewById(R.id.setting_icon);
        }
    }


    public SettingsLayoutAdapter(List<AvailableSetting> settingsList) {
        this.settingsList = settingsList;
    }

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

        return new viewHolder(itemView);
    }

    @Override
    public void onBindViewHolder(viewHolder holder, int position) {
        AvailableSetting setting = settingsList.get(position);
        holder.setting_title.setText(setting.getTitle());
        holder.setting_subtitle.setText(setting.getSubtitle());

        Drawable icon = Application.getContext().getDrawable(setting.getIconResourceID());
        icon.setColorFilter(Application.getContext().getColor(R.color.colorPrimary), PorterDuff.Mode.
SRC_ATOP);
        holder.setting_icon.setImageDrawable(icon);
    }

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

```

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

import android.content.DialogInterface;
import android.graphics.PorterDuff;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import android.speech.tts.Voice;
import android.support.v4.content.ContextCompat;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ProgressBar;

import com.example.gui_design.Application;
import com.example.gui_design.R;
import com.example.gui_design.TextToSpeechHandler;
import com.github.ybq.android.spinkit.sprite.Sprite;
import com.github.ybq.android.spinkit.style.FadingCircle;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.Locale;

public class VoiceRecognitionSettingsActivity extends AppCompatActivity {
    private static Toolbar toolbar;

    private Button speech_test_btn;
    private Button set_voice_btn;
    private EditText phrase_editText;
    private static ProgressBar progressBar;

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

        toolbar = this.findViewById(R.id.toolbar);
        speech_test_btn = this.findViewById(R.id.speech_test_btn);
        set_voice_btn = this.findViewById(R.id.set_voice_btn);
        phrase_editText = this.findViewById(R.id.phrase_editText);

        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);
        }

        TextToSpeechHandler.initializeTextToSpeech();


        progressBar = (ProgressBar)findViewById(R.id.progressBar);
        Sprite fadingCircle = new FadingCircle();
        fadingCircle.setColor(R.color.colorPrimary);
        progressBar.setIndeterminateDrawable(fadingCircle);


    }

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

```
 77              finish();
 78          }
 79
 80          return super.onOptionsItemSelected(item);
 81      }
 82
 83      public void speechTestOnClick(View view){
 84          if (TextToSpeechHandler.isInitialized){
 85              //TextToSpeechHandler.addToSpeakingQueue(phrase_editText.getText().toString());
 86
 87              Date currentTime = Calendar.getInstance().getTime();
 88              SimpleDateFormat df = new SimpleDateFormat("d-MMMM-yyyy", new Locale("spa"));
 89              String todayAsString = df.format(currentTime);
 90              SimpleDateFormat onlyTimeFormar = new SimpleDateFormat("HH:mm:ss", new Locale("spa"));
 91              String onlyTimeString = onlyTimeFormar.format(currentTime);
 92
 93              TextToSpeechHandler.addToSpeakingQueue("Hoy es " + todayAsString + "y son las" +
    onlyTimeString);
 94
 95          }else{
 96              Application.toastMessage("Not initialized");
 97          }
 98      }
 99
100      public static ArrayList<String> voiceName= new ArrayList<>();
101      public static Voice[] voiceArray;
102
103      public void setVoiceOnClick(View view){
104          AlertDialog.Builder builder = new AlertDialog.Builder(Application.getContext());
105
106          voiceArray = TextToSpeechHandler.getVoicesList().toArray(new Voice[TextToSpeechHandler.
    getVoicesList().size()]);
107
108          for(Voice voice : voiceArray){
109              voiceName.add(voice.getName());
110          }
111
112          builder.setTitle("Seleccionar voz");
113          builder.setItems(voiceName.toArray(new String[voiceName.size()]), new DialogInterface.
    OnClickListener() {
114              @Override
115              public void onClick(DialogInterface dialog, int which) {
116                  TextToSpeechHandler.setVoice(voiceArray[which]);
117              }
118          });
119          builder.show();
120      }
121
122      @Override
123      protected void onPause() {
124          if (TextToSpeechHandler.isInitialized){
125              //TextToSpeechHandler.shutdownTextSpeech();
126          }
127
128          super.onPause();
129      }
130  }
131
```

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

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;

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

public class ContactsCommunicationActivity extends AppCompatActivity {
    private static Toolbar toolbar;

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

```

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

public class AvailableSetting {
    private String title, subtitle;
    private int iconResourceID;
    private Class targetActivity;

    public AvailableSetting() {
    }

    public AvailableSetting(String title, String subtitle, int iconResourceID, Class targetActivity) {
        this.title = title;
        this.subtitle = subtitle;
        this.iconResourceID = iconResourceID;
        this.targetActivity = targetActivity;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String name) {
        this.title = name;
    }

    public String getSubtitle() {
        return subtitle;
    }

    public void setSubtitle(String subtitle) {
        this.subtitle = subtitle;
    }

    public int getIconResourceID() {
        return iconResourceID;
    }

    public void setIconResourceID(int iconResourceID) {
        this.iconResourceID = iconResourceID;
    }

    public Class getTargetActivity() { return targetActivity; }

    public void setTargetActivity(Class targetActivity) { this.targetActivity = targetActivity; }
}

```

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

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
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 com.example.gui_design.Application;
import com.example.gui_design.R;

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

public class settings_layout extends AppCompatActivity {

    private static Toolbar toolbar;

    public static List<AvailableSetting> settingsList = new ArrayList<>();
    private RecyclerView recyclerView;
    public static SettingsLayoutAdapter mAdapter;


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

        toolbar = 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);
        }

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

        mAdapter = new SettingsLayoutAdapter(settingsList);

        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);

        // adding inbuilt divider line
        recyclerView.addItemDecoration(new DividerItemDecoration(this, LinearLayoutManager.VERTICAL));

        // adding custom divider line with padding 16dp
```

```
76          //recyclerView.addItemDecoration(new MyDividerItemDecoration(this, LinearLayoutManager.
    HORIZONTAL, 16));
77          recyclerView.setItemAnimator(new DefaultItemAnimator());
78
79          recyclerView.setAdapter(mAdapter);
80
81          // row click listener
82          recyclerView.addOnItemTouchListener(new RecyclerTouchListener(getApplicationContext(),
    recyclerView, new RecyclerTouchListener.ClickListener() {
83              @Override
84              public void onClick(View view, int position) {
85                  AvailableSetting setting = settingsList.get(position);
86
87                  Context currentContext = Application.getContext();
88                  Activity currentActivity = Application.getActivity();
89                  Intent i = new Intent(currentContext, setting.getTargetActivity());
90                  currentContext.startActivity(i);
91                  //currentActivity.overridePendingTransition(R.anim.fade_in, R.anim.fade_out);
92              }
93
94              @Override
95              public void onLongClick(View view, int position) {
96
97              }
98          }));
99
100         mAdapter.notifyDataSetChanged();
101
102     }
103
104     @Override
105     public boolean onOptionsItemSelected(MenuItem item) {
106         if (item.getItemId() == android.R.id.home) // Press Back Icon
107         {
108             finish();
109         }
110
111         return super.onOptionsItemSelected(item);
112     }
113
114     public static void setupSettingsList() {
115
116         settingsList.add(new AvailableSetting(
117             Application.getContext().getString(R.string.internet_conectivity_title),
118                 Application.getContext().getString(R.string.internet_conectivity_subtitle),
119             R.drawable.ic_setting_internet_connectivity,
120             InternetConectivityActivity.class
121         ));
122
123         settingsList.add(new AvailableSetting(
124                 Application.getContext().getString(R.string.bluetooth_settings_title),
125                 Application.getContext().getString(R.string.bluetooth_settings_subtitle),
126                 R.drawable.ic_setting_bluetooth,
127                 BluetoothSettingsActivity.class
128         ));
129
130         settingsList.add(new AvailableSetting(
131                 Application.getContext().getString(R.string.voice_recognition_settings_title),
132                 Application.getContext().getString(R.string.voice_recognition_settings_subtitle),
133                 R.drawable.ic_setting_voice_recognition,
134                 VoiceRecognitionSettingsActivity.class
135         ));
136
137         settingsList.add(new AvailableSetting(
138                 Application.getContext().getString(R.string.commands_settings_title),
139                 Application.getContext().getString(R.string.commands_settings_subtitle),
140                 R.drawable.ic_setting_commands_3,
141                 CommandsSettingsActivity.class
142         ));
143
144         settingsList.add(new AvailableSetting(
145                 Application.getContext().getString(R.string.contacts_communication_title),
146                 Application.getContext().getString(R.string.contacts_communication_subtitle),
147                 R.drawable.ic_setting_contacts_and_communication,
148                 ContactsCommunicationActivity.class
149         ));
150
```

```
151         settingsList.add(new AvailableSetting(
152                 Application.getContext().getString(R.string.appearance_miscellaneous_title),
153                 Application.getContext().getString(R.string.appearance_miscellaneous_subtitle),
154                 R.drawable.ic_setting_appearance_and_miscellaneous,
155                 AppearanceMiscellaneousActivity.class
156         ));
157 
158         settingsList.add(new AvailableSetting(
159                 Application.getContext().getString(R.string.developer_console_activity_title),
160                 Application.getContext().getString(R.string.developer_console_subtitle),
161                 R.drawable.ic_setting_developer_console,
162                 DeveloperConsoleActivity.class
163         ));
164 
165         settingsList.add(new AvailableSetting(
166                 Application.getContext().getString(R.string.information_help_title),
167                 Application.getContext().getString(R.string.information_help_subtitle),
168                 R.drawable.ic_setting_information_and_help,
169                 InformationHelpActivity.class
170         ));
171 
172         // notify adapter about data set changes
173         // so that it will render the list with new data
174         //mAdapter.notifyDataSetChanged();
175     }
176 
177 
178 }
179 
```

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

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;

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

public class AppearanceMiscellaneousActivity extends AppCompatActivity {

    private static Toolbar toolbar;

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

```

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

import android.content.Context;
import android.content.res.Resources;
import android.content.res.TypedArray;
import android.graphics.Canvas;
import android.graphics.Rect;
import android.graphics.drawable.Drawable;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.util.TypedValue;
import android.view.View;

/**
 * Created by Ravi on 30/10/15.
 * updated by Ravi on 14/11/17
 */
public class MyDividerItemDecoration extends RecyclerView.ItemDecoration {

    private static final int[] ATTRS = new int[]{
            android.R.attr.listDivider
    };

    public static final int HORIZONTAL_LIST = LinearLayoutManager.HORIZONTAL;

    public static final int VERTICAL_LIST = LinearLayoutManager.VERTICAL;

    private Drawable mDivider;
    private int mOrientation;
    private Context context;
    private int margin;

    public MyDividerItemDecoration(Context context, int orientation, int margin) {
        this.context = context;
        this.margin = margin;
        final TypedArray a = context.obtainStyledAttributes(ATTRS);
        mDivider = a.getDrawable(0);
        a.recycle();
        setOrientation(orientation);
    }

    public void setOrientation(int orientation) {
        if (orientation != HORIZONTAL_LIST && orientation != VERTICAL_LIST) {
            throw new IllegalArgumentException("invalid orientation");
        }
        mOrientation = orientation;
    }

    @Override
    public void onDrawOver(Canvas c, RecyclerView parent, RecyclerView.State state) {
        if (mOrientation == VERTICAL_LIST) {
            drawVertical(c, parent);
        } else {
            drawHorizontal(c, parent);
        }
    }

    public void drawVertical(Canvas c, RecyclerView parent) {
        final int left = parent.getPaddingLeft();
        final int right = parent.getWidth() - parent.getPaddingRight();

        final int childCount = parent.getChildCount();
        for (int i = 0; i < childCount; i++) {
            final View child = parent.getChildAt(i);
            final RecyclerView.LayoutParams params = (RecyclerView.LayoutParams) child
                    .getLayoutParams();
            final int top = child.getBottom() + params.bottomMargin;
            final int bottom = top + mDivider.getIntrinsicHeight();
            mDivider.setBounds(left + dpToPx(margin), top, right - dpToPx(margin), bottom);
            mDivider.draw(c);
        }
    }

    public void drawHorizontal(Canvas c, RecyclerView parent) {
        final int top = parent.getPaddingTop();
        final int bottom = parent.getHeight() - parent.getPaddingBottom();

```

```
 78          final int childCount = parent.getChildCount();
 79          for (int i = 0; i < childCount; i++) {
 80              final View child = parent.getChildAt(i);
 81              final RecyclerView.LayoutParams params = (RecyclerView.LayoutParams) child
 82                      .getLayoutParams();
 83              final int left = child.getRight() + params.rightMargin;
 84              final int right = left + mDivider.getIntrinsicHeight();
 85              mDivider.setBounds(left, top + dpToPx(margin), right, bottom - dpToPx(margin));
 86              mDivider.draw(c);
 87          }
 88      }
 89
 90      @Override
 91      public void getItemOffsets(Rect outRect, View view, RecyclerView parent, RecyclerView.State state
    ) {
 92          if (mOrientation == VERTICAL_LIST) {
 93              outRect.set(0, 0, 0, mDivider.getIntrinsicHeight());
 94          } else {
 95              outRect.set(0, 0, mDivider.getIntrinsicWidth(), 0);
 96          }
 97      }
 98
 99      private int dpToPx(int dp) {
100          Resources r = context.getResources();
101          return Math.round(TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_DIP, dp, r.
    getDisplayMetrics()));
102      }
103  }
```

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.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Date;
import java.util.Locale;

public class GoogleSpeechRecognizer extends MainActivity implements RecognitionListener {

    private static SpeechRecognizer speechRecognizer = null;

    private boolean forceCancel = false;

    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);
        if (Application.isSystemOnline()==false) { intent.putExtra(RecognizerIntent.
EXTRA_PREFER_OFFLINE,true); }
        speechRecognizer.startListening(intent);


        if (Application.isSystemOnline()) {
            Thread autoStop = new Thread() {
                @Override
                public void run() {
                    super.run();
                    try {
                        Thread.sleep(4000);
                        forceStopRecognizer();
                    } catch (Throwable th) {
                        Application.toastMessage(th.toString());
                    }
                }
            };
            autoStop.start();
        }

    }

    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(){
        Application.getActivity().runOnUiThread(
```

```
 76                 new Runnable() {
 77                     @Override
 78                     public void run() {
 79                         speechRecognizer.stopListening();
 80                         MicrophoneHandler.setState(MicrophoneHandler.MicrophoneState.
    LISTENING_KEYOWRD);
 81                     }
 82                 }
 83         );
 84 
 85     }
 86 
 87 
 88     public static void analizeListenerResults(ArrayList<String> data, float[] confidenceScore){
 89         outerloop:
 90         for (int i = 0; i < data.size(); i++) {
 91             String hypothesis = Normalizer.normalize(((String) data.get(i)).toLowerCase(), Normalizer
    .Form.NFKD);
 92             String[] hypothesisSplit = hypothesis.split(" ");
 93 
 94             if (Arrays.asList(hypothesisSplit).contains("encender") && Arrays.asList(hypothesisSplit)
    .contains("velador")) {
 95                 BluetoothConnection.composeAndSend(BluetoothConnection.commandType.TURN_RELAY_ON, 0);
 96                 break outerloop;
 97             }
 98 
 99             if (Arrays.asList(hypothesisSplit).contains("encender") && Arrays.asList(hypothesisSplit)
    .contains("estufa")) {
100                 BluetoothConnection.composeAndSend(BluetoothConnection.commandType.TURN_RELAY_ON, 1);
101                 break outerloop;
102             }
103 
104             if (Arrays.asList(hypothesisSplit).contains("encender") && Arrays.asList(hypothesisSplit)
    .contains("ventilador")) {
105                 BluetoothConnection.composeAndSend(BluetoothConnection.commandType.TURN_RELAY_ON, 2);
106                 break outerloop;
107             }
108 
109             if (Arrays.asList(hypothesisSplit).contains("encender") && Arrays.asList(hypothesisSplit)
    .contains("luz")) {
110                 BluetoothConnection.composeAndSend(BluetoothConnection.commandType.TURN_RELAY_ON, 3);
111                 break outerloop;
112             }
113 
114             if (Arrays.asList(hypothesisSplit).contains("apagar") && Arrays.asList(hypothesisSplit).
    contains("velador")) {
115                 BluetoothConnection.composeAndSend(BluetoothConnection.commandType.TURN_RELAY_OFF, 0)
    ;
116                 break outerloop;
117             }
118 
119             if (Arrays.asList(hypothesisSplit).contains("apagar") && Arrays.asList(hypothesisSplit).
    contains("estufa")) {
120                 BluetoothConnection.composeAndSend(BluetoothConnection.commandType.TURN_RELAY_OFF, 1)
    ;
121                 break outerloop;
122             }
123 
124             if (Arrays.asList(hypothesisSplit).contains("apagar") && Arrays.asList(hypothesisSplit).
    contains("ventilador")) {
125                 BluetoothConnection.composeAndSend(BluetoothConnection.commandType.TURN_RELAY_OFF, 2)
    ;
126                 break outerloop;
127             }
128 
129             if (Arrays.asList(hypothesisSplit).contains("apagar") && Arrays.asList(hypothesisSplit).
    contains("luz")) {
130                 BluetoothConnection.composeAndSend(BluetoothConnection.commandType.TURN_RELAY_OFF, 3)
    ;
131                 break outerloop;
132             }
133 
134             if (Arrays.asList(hypothesisSplit).contains("encender") && Arrays.asList(hypothesisSplit)
    .contains("todo")) {
135                 BluetoothConnection.composeAndSend(BluetoothConnection.commandType.TURN_EVERYTHING_ON
    , 0);
136                 break outerloop;
```

```
137             }
138
139             if (Arrays.asList(hypothesisSplit).contains("apagar") && Arrays.asList(hypothesisSplit).
    contains("todo")) {
140                 BluetoothConnection.composeAndSend(BluetoothConnection.commandType.
    TURN_EVERYTHING_OFF, 0);
141                 break outerloop;
142             }
143
144             if (Arrays.asList(hypothesisSplit).contains("llamar") && Arrays.asList(hypothesisSplit).
    contains("enfermera")) {
145                 BluetoothConnection.composeAndSend(BluetoothConnection.commandType.CALL_NURSE, 0);
146                 break outerloop;
147             }
148
149             if (Arrays.asList(hypothesisSplit).contains("llamar") && Arrays.asList(hypothesisSplit).
    contains("enfermeria")) {
150                 BluetoothConnection.composeAndSend(BluetoothConnection.commandType.CALL_NURSE, 0);
151                 break outerloop;
152             }
153
154
155             if (Arrays.asList(hypothesisSplit).contains("encender") && Arrays.asList(hypothesisSplit)
    .contains("rele")) {
156                 for (String word : hypothesisSplit) {
157                     if (StringUtils.isNumeric(word) == true) {
158                         int number = Integer.parseInt(word);
159                         BluetoothConnection.composeAndSend(BluetoothConnection.commandType.
    TURN_RELAY_ON, number);
160                         break outerloop;
161                     }
162                 }
163             }
164
165             if (Arrays.asList(hypothesisSplit).contains("apagar") && Arrays.asList(hypothesisSplit).
    contains("rele")) {
166                 for (String word : hypothesisSplit) {
167                     if (StringUtils.isNumeric(word) == true) {
168                         int number = Integer.parseInt(word);
169                         BluetoothConnection.composeAndSend(BluetoothConnection.commandType.
    TURN_RELAY_OFF, number);
170                         break outerloop;
171                     }
172                 }
173             }
174
175             if (Arrays.asList(hypothesisSplit).contains("decir") && Arrays.asList(hypothesisSplit).
    contains("hora")){
176                 Date currentTime = Calendar.getInstance().getTime();
177                 SimpleDateFormat df = new SimpleDateFormat("d-MMMM-yyyy", new Locale("spa"));
178                 String todayAsString = df.format(currentTime);
179                 SimpleDateFormat onlyTimeFormar = new SimpleDateFormat("HH:mm:ss", new Locale("spa"))
    ;
180                 String onlyTimeString = onlyTimeFormar.format(currentTime);
181
182                 //TextToSpeechHandler.initializeTextToSpeech();
183                 TextToSpeechHandler.addToSpeakingQueue("Hoy es " + todayAsString + "y son las" +
    onlyTimeString);
184                 Application.toastMessage("Decir hora");
185             }
186
187         }
188     }
189
190     @Override
191     public void onReadyForSpeech(Bundle params) {
192         MicrophoneHandler.setState(MicrophoneHandler.MicrophoneState.LISTENING_COMMAND);
193     }
194
195     @Override
196     public void onBeginningOfSpeech() {
197
198     }
199
200     @Override
201     public void onRmsChanged(float rmsdB) {
202
```

```
203     }
204
205     @Override
206     public void onBufferReceived(byte[] buffer) {
207
208     }
209
210     @Override
211     public void onEndOfSpeech() {
212
213         //Application.toastMessage("End of speech");
214     }
215
216     @Override
217     public void onError(int error) {
218         //Application.toastMessage("On error" + error);
219
220         if (forceCancel==false) {
221             PocketsphinxListener.startListening();
222             MicrophoneHandler.setState(MicrophoneHandler.MicrophoneState.LISTENING_KEYOWRD);
223         }
224     }
225
226     @Override
227     public void onResults(Bundle results) {
228         String str = new String();
229         ArrayList<String> resultsArrayList = results.getStringArrayList(SpeechRecognizer.
    RESULTS_RECOGNITION);
230         float[] confidenceScore = results.getFloatArray(SpeechRecognizer.CONFIDENCE_SCORES);
231         analizeListenerResults(resultsArrayList, confidenceScore);
232
233         StringBuilder builder = new StringBuilder();
234         for (String line : resultsArrayList) {
235             builder.append(line + "\n");
236         }
237
238         MainActivity.textView_speechRecognitionResults.setText(builder.toString());
239         MainActivity.textView_speechRecognitionResults.setVisibility(View.VISIBLE);
240
241
242
243         MicrophoneHandler.setState(MicrophoneHandler.MicrophoneState.LISTENING_KEYOWRD);
244
245         PocketsphinxListener.startListening();
246     }
247
248     @Override
249     public void onPartialResults(Bundle partialResults) {
250         String str = new String();
251         ArrayList<String> resultsArrayList = partialResults.getStringArrayList(SpeechRecognizer.
    RESULTS_RECOGNITION);
252         float[] confidenceScore = partialResults.getFloatArray(SpeechRecognizer.CONFIDENCE_SCORES);
253
254         StringBuilder builder = new StringBuilder();
255         for (String line : resultsArrayList) {
256             builder.append(line + "\n");
257         }
258
259     }
260
261     @Override
262     public void onEvent(int eventType, Bundle params) {
263
264     }
265 }
266
```

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

import android.content.Context;
import android.support.v7.widget.RecyclerView;
import android.view.GestureDetector;
import android.view.MotionEvent;
import android.view.View;

/**
 * Created by Ravi Tamada on 03/09/16.
 * updated by Ravi on 14/11/17
 * www.androidhive.info
 */
public class RecyclerTouchListener implements RecyclerView.OnItemTouchListener {

    private GestureDetector gestureDetector;
    private ClickListener clickListener;

    public RecyclerTouchListener(Context context, final RecyclerView recyclerView, final ClickListener clickListener) {
        this.clickListener = clickListener;
        gestureDetector = new GestureDetector(context, new GestureDetector.SimpleOnGestureListener() {
            @Override
            public boolean onSingleTapUp(MotionEvent e) {
                return true;
            }

            @Override
            public void onLongPress(MotionEvent e) {
                View child = recyclerView.findChildViewUnder(e.getX(), e.getY());
                if (child != null && clickListener != null) {
                    clickListener.onLongClick(child, recyclerView.getChildAdapterPosition(child));
                }
            }
        });
    }

    @Override
    public boolean onInterceptTouchEvent(RecyclerView rv, MotionEvent e) {

        View child = rv.findChildViewUnder(e.getX(), e.getY());
        if (child != null && clickListener != null && gestureDetector.onTouchEvent(e)) {
            clickListener.onClick(child, rv.getChildAdapterPosition(child));
        }
        return false;
    }

    @Override
    public void onTouchEvent(RecyclerView rv, MotionEvent e) {
    }

    @Override
    public void onRequestDisallowInterceptTouchEvent(boolean disallowIntercept) {

    }

    public interface ClickListener {
        void onClick(View view, int position);

        void onLongClick(View view, int position);
    }
}
```

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

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;

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

public class BluetoothSettingsActivity extends AppCompatActivity {
    private static Toolbar toolbar;

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

```

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

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;

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

public class InformationHelpActivity extends AppCompatActivity {
    private static Toolbar toolbar;

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

```

```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();
    }
}

```

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

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;

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

public class DeveloperConsoleActivity extends AppCompatActivity {
    private static Toolbar toolbar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.developer_console_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\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\MainActivity.java

```java
package com.example.gui_design;

import android.Manifest;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
import android.support.v7.app.AppCompatActivity;
import android.util.DisplayMetrics;
import android.view.View;
import android.widget.ImageButton;
import android.widget.TextView;

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

public class MainActivity extends AppCompatActivity {

    static ImageButton imgBtn_microphone;
    static ImageButton imgBtn_microphoneBackground;
    static ImageButton imgBtn_commands;
    static ImageButton imgBtn_contacts;
    static ImageButton imgBtn_bluetooth;
    static ImageButton imgBtn_settings;
    static ImageButton imgBtn_menu;
    static MainMenuHandler mainMenuHandler;
    static TextView textView_speechRecognitionResults;


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

        if(savedInstanceState == null) {

            mainMenuHandler = new MainMenuHandler(this, this);
            Application.setActivity(this);
            Application.setContext(this);
            MainMenuHandler.dpiConstant = ((float) this.getResources().getDisplayMetrics().densityDpi / DisplayMetrics.DENSITY_DEFAULT);

            imgBtn_microphone = findViewById(R.id.imgBtn_microphone);
            imgBtn_microphoneBackground = findViewById(R.id.imgBtn_microphone_background);
            imgBtn_commands = findViewById(R.id.imgBtn_commands);
            imgBtn_contacts = findViewById(R.id.imgBtn_contacts);
            imgBtn_bluetooth = findViewById(R.id.imgBtn_bluetooth);
            imgBtn_settings = findViewById(R.id.imgBtn_settings);
            imgBtn_menu = findViewById(R.id.imgBtn_menu);
            textView_speechRecognitionResults = findViewById(R.id.textView_speechRecognitionResults);

            if (ContextCompat.checkSelfPermission(this,
                    Manifest.permission.RECORD_AUDIO)
                    != PackageManager.PERMISSION_GRANTED) {
                if (ActivityCompat.shouldShowRequestPermissionRationale(this,
                        Manifest.permission.RECORD_AUDIO)) {
                } else {
                    ActivityCompat.requestPermissions(this,
                            new String[]{Manifest.permission.RECORD_AUDIO},
                            527);
                }
            } else {
                GoogleSpeechRecognizer.setupRecognizer();
                PocketsphinxListener.setupRecognizer();
                PocketsphinxListener.startListening();

            }

            settings_layout.setupSettingsList();

            BluetoothConnection.initBTAutoConnectService();

        }
    }

    @Override
    public void onRequestPermissionsResult(int requestCode,
```

```
77                                          String permissions[], int[] grantResults) {
78          switch (requestCode) {
79              case 527: {
80                  // If request is cancelled, the result arrays are empty.
81                  if (grantResults.length > 0
82                          && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
83
84                      GoogleSpeechRecognizer.setupRecognizer();
85                      PocketsphinxListener.setupRecognizer();
86
87
88                  } else { }
89              }
90          }
91      }
92
93      public void onClickListener_menu(View view){ mainMenuHandler.menuButtonOnClick(); }
94
95      public void onClickListener_commands(View view){ mainMenuHandler.commandsButtonOnClick(); }
96
97      public void onClickListener_settings(View view){ mainMenuHandler.settingsButtonOnClick(); }
98
99      public void onClickListener_contacts(View view){ mainMenuHandler.contactsButtonOnClick(); }
100
101     public void onClickListener_bluetooth(View view){ mainMenuHandler.bluetoothButtonOnClick(); }
102
103     public void onClickListener_microphone(View view){ MicrophoneHandler.microphoneOnClick(); }
104
105     @Override
106     public void onDestroy() {
107
108         PocketsphinxListener.shutdownRecognizer();
109         GoogleSpeechRecognizer.shutdownRecognizer();
110         BluetoothConnection.closeAllConnections();
111         TextToSpeechHandler.shutdownTextSpeech();
112
113         super.onDestroy();
114     }
115
116 }
117
```

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

```java
package com.example.gui_design;

import android.graphics.PorterDuff;
import android.view.animation.AccelerateDecelerateInterpolator;

import java.util.concurrent.Callable;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.Future;

public class MicrophoneHandler extends MainActivity {

    public enum MicrophoneState { INHIBITED, LISTENING_KEYOWRD, LISTENING_COMMAND}

    static private MicrophoneState currentState = MicrophoneState.LISTENING_KEYOWRD;

    public static void microphoneOnClick(){
        switch (currentState){
            case LISTENING_KEYOWRD:
                PocketsphinxListener.stopRecognizer();
                setState(MicrophoneState.INHIBITED);
                break;
            case LISTENING_COMMAND:
                GoogleSpeechRecognizer.forceStopRecognizer();
                //PocketsphinxListener.startListening();
                setState(MicrophoneState.LISTENING_KEYOWRD);
                break;
            case INHIBITED:
                setState(MicrophoneState.LISTENING_KEYOWRD);
                PocketsphinxListener.startListening();
                break;
        }

    }

    public static MicrophoneState getState(){
        return currentState;
    }

    public static void setState(MicrophoneState desiredState){
        switch (desiredState){
            case INHIBITED:
                imgBtn_microphoneBackground.clearColorFilter();
                imgBtn_microphoneBackground.setColorFilter(Application.getContext().getColor(R.color.ColorBackMicInhibited), PorterDuff.Mode.SRC_ATOP);
                imgBtn_microphone.setImageResource(R.drawable.ic_main_mic_off);
                endAnimation();
                currentState = MicrophoneState.INHIBITED;
                break;
            case LISTENING_KEYOWRD:
                imgBtn_microphoneBackground.clearColorFilter();
                imgBtn_microphoneBackground.setColorFilter(Application.getContext().getColor(R.color.ColorBackMicKeyword), PorterDuff.Mode.SRC_ATOP);
                imgBtn_microphone.setImageResource(R.drawable.ic_main_mic_listening);
                endAnimation();
                currentState = MicrophoneState.LISTENING_KEYOWRD;
                break;
            case LISTENING_COMMAND:
                imgBtn_microphoneBackground.clearColorFilter();
                imgBtn_microphoneBackground.setColorFilter(Application.getContext().getColor(R.color.ColorBackMicCommand), PorterDuff.Mode.SRC_ATOP);
                imgBtn_microphone.setImageResource(R.drawable.ic_main_mic_listening);
                startAnimation();
                currentState = MicrophoneState.LISTENING_COMMAND;
                break;
        }
    }

    private static ExecutorService executor = Executors.newSingleThreadExecutor();
    private static boolean terminateAnimation = false;
    private static boolean animationHasFinished = false;

    public static void startAnimation(){
        terminateAnimation = false;
        Future<Void> future = executor.submit(new Callable<Void>() {
            public Void call() throws Exception {
                try {
```

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

```
 75                     while (terminateAnimation == false) {
 76                         imgBtn_microphoneBackground.animate().scaleX(0.90f).setDuration(350).
    setInterpolator(new AccelerateDecelerateInterpolator());
 77                         imgBtn_microphone.animate().scaleX(0.90f).setDuration(350).setInterpolator(
    new AccelerateDecelerateInterpolator());
 78                         imgBtn_microphoneBackground.animate().scaleY(0.90f).setDuration(350).
    setInterpolator(new AccelerateDecelerateInterpolator());
 79                         imgBtn_microphone.animate().scaleY(0.90f).setDuration(350).setInterpolator(
    new AccelerateDecelerateInterpolator()).withEndAction(new Runnable() {
 80                             @Override
 81                             public void run() {
 82                                 animationHasFinished = true;
 83                             }
 84                         });
 85                         while(animationHasFinished==false);
 86                         animationHasFinished= false;
 87                         imgBtn_microphoneBackground.animate().scaleX(1.00f).setDuration(350).
    setInterpolator(new AccelerateDecelerateInterpolator());
 88                         imgBtn_microphone.animate().scaleX(1.00f).setDuration(350).setInterpolator(
    new AccelerateDecelerateInterpolator());
 89                         imgBtn_microphoneBackground.animate().scaleY(1.00f).setDuration(350).
    setInterpolator(new AccelerateDecelerateInterpolator());
 90                         imgBtn_microphone.animate().scaleY(1.00f).setDuration(350).setInterpolator(
    new AccelerateDecelerateInterpolator()).withEndAction(new Runnable() {
 91                             @Override
 92                             public void run() {
 93                                 animationHasFinished = true;
 94                             }
 95                         });;
 96                         while(animationHasFinished==false);
 97                         animationHasFinished = false;
 98                     }
 99                     terminateAnimation = false;
100                 }catch (Throwable th){ ;
101                     th.printStackTrace();
102                 }
103                 return null;
104             }
105         });
106     }
107
108     public static void endAnimation(){
109         terminateAnimation = true;
110     }
111
112 }
113
```

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
```

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

```java
package com.example.gui_design;

import android.speech.tts.TextToSpeech;
import android.speech.tts.Voice;

import java.util.Locale;
import java.util.Set;

public class TextToSpeechHandler {

    private static TextToSpeech textToSpeech;
    public static boolean isInitialized = false;

    public static void initializeTextToSpeech(){
        if (isInitialized == false) {
            textToSpeech = new TextToSpeech(Application.getContext(), new TextToSpeech.OnInitListener(
) {
                @Override
                public void onInit(int status) {
                    try {
                        if (status != TextToSpeech.ERROR) {
                            textToSpeech.setLanguage(new Locale("spa"));
                            //textToSpeech.setLanguage(Locale.UK);
                            //textToSpeech.setOnUtteranceProgressListener(utteranceListener);
                            textToSpeech.setPitch(1.0f);
                            textToSpeech.setSpeechRate(1.0f);
                            textToSpeech.getVoices();
                            isInitialized = true;
                        }
                    } catch (Throwable th) {
                        Application.toastMessage(th.toString());
                    }
                }
            });
        }
    }

    public static void addToSpeakingQueue(String text){
        try {
            if (isInitialized) {
                textToSpeech.speak(text, TextToSpeech.QUEUE_ADD, null, "232");
            }
        } catch (Throwable th) {
            th.printStackTrace();
            Application.toastMessage(th.toString());
        }
    }

    public static void shutdownTextSpeech(){
        if(textToSpeech != null) {

            textToSpeech.stop();
            textToSpeech.shutdown();
        }
    }

    public static Set<Voice> getVoicesList(){
        if (isInitialized){
            return textToSpeech.getVoices();
        } else {
            return null;
        }
    }

    public static void setVoice(Voice targetVoice){
        if(isInitialized) {
            textToSpeech.setVoice(targetVoice);
        }
    }



}
```