File - C:\Users\Grain\source\repos\gratux\PiBell\PiBell\PiBell\MainPage.xaml.cs

```
 1 using System;
 2 using System.Collections;
 3 using System.Collections.Generic;
 4 using System.Collections.ObjectModel;
 5 using System.Collections.Specialized;
 6 using System.ComponentModel;
 7 using System.Linq;
 8 using System.Text;
 9 using System.Threading.Tasks;
10 using LibVLCSharp.Shared;
11 using PiBell.Classes;
12 using Xamarin.Forms;
13 using LibVLCSharp.Forms.Shared;
14 using Microsoft.AppCenter.Push;
15 using Microsoft.AppCenter;
16
17 namespace PiBell
18 {
19     public partial class MainPage : ContentPage
20     {
21         private Player _player;
22         private Streamer _streamer;
23         private double _width, _height;
24         private bool _wasConnected;
25         private float _prevPosition;
26
27         public MainPage()
28         {
29             InitializeComponent();
30             RegenerateVideoView();
31             Core.Initialize();
32
33             _player = new Player();
34             _streamer = new Streamer();
35
36             if (!AppCenter.Configured)
37                 Push.PushNotificationReceived += async (sender, e) =>
38                 {
39 #if DEBUG
40                     Console.WriteLine("DEBUG - Push-Notification recieved");
41 #endif
42                     foreach (string key in e.CustomData.Keys)
43                     {
44 #if DEBUG
45                         Console.WriteLine($"DEBUG - Custom Data: {key}:{e.
   CustomData[key]}");
46 #endif
47                         if (key == "mrl")
48                            _player.Mrl = e.CustomData[key].ToString();
49                     }
50
51                     if (await DisplayAlert("Incoming Call", "Connect now?", "
   Yes", "No"))
52                     {
53                         _player.StartCall();
54                         _streamer.StartCall();
55                     }
56                 };
57             MessagingCenter.Subscribe<string>(this, "OnPause", app =>
58             {
```

```
 59                 _wasConnected = _player.MediaPlayer.IsPlaying;
 60                 _player.MediaPlayer.Pause();
 61                 _prevPosition = _player.MediaPlayer.Position;
 62                 _player.MediaPlayer.Stop();
 63
 64                 MainGrid.Children.Remove(VideoView);
 65             });
 66
 67             MessagingCenter.Subscribe<string>(this, "OnRestart", app =>
 68             {
 69                 RegenerateVideoView();
 70                 VideoView.MediaPlayer = _player.MediaPlayer;
 71                 if (_wasConnected)
 72                 {
 73                     _player.MediaPlayer.Play();
 74                     _player.MediaPlayer.Position = _prevPosition;
 75                 }
 76
 77                 _prevPosition = 0;
 78             });
 79
 80             _player = new Player();
 81             _streamer = new Streamer();
 82             //_player.MediaPlayer.Volume = 0;
 83
 84             BindingContext = _player;
 85         }
 86
 87         void RegenerateVideoView()
 88         {
 89             VideoView = new VideoView();
 90             MainGrid.Children.Add(VideoView, 0, 1);
 91         }
 92
 93         protected override void OnAppearing()
 94         {
 95             base.OnAppearing();
 96             _player.MediaPlayer = new MediaPlayer(_player.LibVlc);
 97         }
 98
 99         private void BtToggleSpeaker_Clicked(object sender, EventArgs e)
100         {
101             VisualStateManager.GoToState((ImageButton) sender, _player.
    ToggleSpeaker() ? "Unmute" : "Mute");
102         }
103
104         private void BtToggleMic_Clicked(object sender, EventArgs e)
105         {
106             VisualStateManager.GoToState((ImageButton) sender, _streamer.
    ToggleMic() ? "Unmute" : "Mute");
107         }
108
109         private void BtConnect_Clicked(object sender, EventArgs e)
110         {
111             _player.StartCall();
112             _streamer.StartCall("192.168.43.78");
113         }
114
115         private void BtDisconnect_Clicked(object sender, EventArgs e)
116         {
```

```
117             _player.EndCall();
118             _streamer.EndCall();
119         }
120
121         protected override void OnSizeAllocated(double width, double height)
122         {
123             base.OnSizeAllocated(width, height);
124             if (_width != width || _height != height)
125             {
126                 _width = width;
127                 _height = height;
128                 if (width < height) //Portrait
129                 {
130                     MainGrid.ColumnDefinitions.Clear();
131                     MainGrid.ColumnDefinitions.Add(new ColumnDefinition
132                         {Width = new GridLength(1, GridUnitType.Star)});
133
134                     MainGrid.RowDefinitions.Clear();
135                     MainGrid.RowDefinitions.Add(new RowDefinition
136                         {Height = new GridLength(0.7, GridUnitType.Star)});
137                     MainGrid.RowDefinitions.Add(new RowDefinition
138                         {Height = new GridLength(3, GridUnitType.Star)});
139                     MainGrid.RowDefinitions.Add(new RowDefinition
140                         {Height = new GridLength(2, GridUnitType.Star)});
141
142                     Grid.SetColumnSpan(EditMrl, 1);
143                     Grid.SetColumn(ButtonGrid, 0);
144                     Grid.SetRow(ButtonGrid, 2);
145                 }
146                 else //Landscape
147                 {
148                     MainGrid.ColumnDefinitions.Clear();
149                     MainGrid.ColumnDefinitions.Add(new ColumnDefinition
150                         {Width = new GridLength(2, GridUnitType.Star)});
151                     MainGrid.ColumnDefinitions.Add(new ColumnDefinition
152                         {Width = new GridLength(1, GridUnitType.Star)});
153
154                     MainGrid.RowDefinitions.Clear();
155                     MainGrid.RowDefinitions.Add(new RowDefinition
156                         {Height = new GridLength(0.7, GridUnitType.Star)});
157                     MainGrid.RowDefinitions.Add(new RowDefinition
158                         {Height = new GridLength(5, GridUnitType.Star)});
159
160                     Grid.SetColumnSpan(EditMrl, 2);
161                     Grid.SetColumn(ButtonGrid, 1);
162                     Grid.SetRow(ButtonGrid, 1);
163                 }
164             }
165         }
166     }
167 }
```

File - C:\Users\Grain\source\repos\gratux\PiBell\PiBell\PiBell.iOS\AudioRecordingService.cs

```
 1 using System;
 2 using System.Collections.Generic;
 3 using System.Linq;
 4 using System.Text;
 5
 6 using UIKit;
 7 using System.Net.Sockets;
 8 using System.Net;
 9 using static PiBell.Classes.Streamer;
10 using System.Threading;
11 using Plugin.AudioRecorder;
12
13 namespace PiBell.iOS
14 {
15     class AudioRecordingService : IAudioRecordingService
16     {
17         Socket socket;
18         NetworkStream stream;
19         Thread recordingThread;
20         AudioRecorderService recorder;
21
22         public AudioRecordingService()
23         {
24             socket = new Socket(SocketType.Dgram, ProtocolType.Udp);
25             stream = new NetworkStream(socket);
26             recorder = new AudioRecorderService();
27             recordingThread = new Thread(RecordingFunction);
28         }
29
30         private void RecordingFunction()
31         {
32             stream = new NetworkStream(socket);
33             recorder.StartRecording();
34             while(true)
35             {
36                 try
37                 {
38                     recorder.GetAudioFileStream().CopyTo(stream);
39                 }
40                 catch(ThreadAbortException)
41                 {
42                     break;
43                 }
44                 catch(Exception e)
45                 {
46                     Console.WriteLine(e.Message);
47                     break;
48                 }
49             }
50             recorder.StopRecording();
51         }
52
53         public void Start(IPEndPoint target)
54         {
55             socket.Connect(target);
56             recordingThread.Start();
57         }
58
59         public void Stop()
60         {
```

```
61              recordingThread.Abort();
62              socket.Close();
63          }
64
65          public bool ToggleMic()
66          {
67              throw new NotImplementedException();
68          }
69      }
70  }
```

File - C:\Users\Grain\source\repos\gratux\PiBell\PiBell\PiBell\Classes\MediaClasses.cs

```
 1 using System;
 2 using System.Collections.Generic;
 3 using System.ComponentModel;
 4 using System.IO;
 5 using System.Net;
 6 using System.Net.Sockets;
 7 using System.Runtime.CompilerServices;
 8 using System.Runtime.InteropServices;
 9 using System.Text;
10 using System.Threading;
11 using LibVLCSharp.Shared;
12 using PiBell.Annotations;
13 using Xamarin.Forms;
14
15 namespace PiBell.Classes
16 {
17     public class Player : INotifyPropertyChanged
18     {
19         public event PropertyChangedEventHandler PropertyChanged;
20
21         [NotifyPropertyChangedInvocator]
22         protected virtual void OnPropertyChanged([CallerMemberName] string
   propertyName = null)
23         {
24             PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(
   propertyName));
25         }
26
27         private bool _speakerActive;
28
29         public Player(string mrl = "https://www.freedesktop.org/software/
   gstreamer-sdk/data/media/sintel_trailer-480p.webm")
30         {
31             LibVlc = new LibVLC();
32             MediaPlayer = new MediaPlayer(LibVlc) { NetworkCaching = 0 };
33             Mrl = mrl;
34         }
35
36         public bool ToggleSpeaker()
37         {
38             MediaPlayer.Volume = _speakerActive ? 0 : 100;
39             return _speakerActive = !_speakerActive;
40         }
41
42         public void StartCall()
43         {
44             MediaPlayer.Media = new Media(LibVlc, Mrl, FromType.FromLocation
   );
45             MediaPlayer.Play();
46             MediaPlayer.Volume = _speakerActive ? 100 : 0;
47         }
48
49         public void EndCall()
50         {
51             MediaPlayer.Stop();
52         }
53
54         private string _mrl;
55         public string Mrl
56         {
```

```
 57              get => _mrl;
 58              set
 59              {
 60                  _mrl = value;
 61                  OnPropertyChanged();
 62              }
 63          }
 64
 65          private LibVLC _libVlc;
 66          public LibVLC LibVlc
 67          {
 68              get => _libVlc;
 69              set
 70              {
 71                  _libVlc = value;
 72                  OnPropertyChanged();
 73              }
 74          }
 75
 76          private MediaPlayer _mediaPlayer;
 77          public MediaPlayer MediaPlayer
 78          {
 79              get => _mediaPlayer;
 80              set
 81              {
 82                  _mediaPlayer = value;
 83                  OnPropertyChanged();
 84              }
 85          }
 86      }
 87
 88      public class Streamer
 89      {
 90          private IAudioRecordingService Service { get; set; }
 91
 92          public Streamer()
 93          {
 94              Service = DependencyService.Get<IAudioRecordingService>();
 95          }
 96
 97          public void StartCall(string targetIp = "127.0.0.1", int targetPort
     = 10000)
 98          {
 99              Service.Start(new IPEndPoint(IPAddress.Parse(targetIp),
    targetPort));
100          }
101
102          public void EndCall()
103          {
104              Service.Stop();
105          }
106
107          public bool ToggleMic()
108          {
109              return Service.ToggleMic();
110          }
111
112          public interface IAudioRecordingService
113          {
114              void Start(IPEndPoint target);
```

```
115             void Stop();
116             bool ToggleMic();
117         }
118     }
119 }
```

File - C:\Users\Grain\source\repos\gratux\PiBell\PiBell\PiBell\App.xaml.cs

```
 1 using System;
 2 using System.Diagnostics;
 3 using LibVLCSharp.Shared;
 4 using Microsoft.AppCenter;
 5 using Microsoft.AppCenter.Analytics;
 6 using Microsoft.AppCenter.Crashes;
 7 using Microsoft.AppCenter.Push;
 8 using Xamarin.Forms;
 9 using Xamarin.Forms.Xaml;
10
11 [assembly: XamlCompilation(XamlCompilationOptions.Compile)]
12
13 namespace PiBell
14 {
15     public partial class App : Application
16     {
17         public App()
18         {
19             InitializeComponent();
20             Core.Initialize();
21             MainPage = new MainPage();
22
23
24             AppCenter.LogLevel = Microsoft.AppCenter.LogLevel.Verbose;
25             AppCenter.Start("android={Your Android App secret here};" +
26                             "uwp={Your UWP App secret here};" +
27                             "ios={Your iOS App secret here}",
28                 typeof(Analytics), typeof(Crashes), typeof(Push));
29         }
30
31         protected override void OnStart()
32         {
33             // Handle when your app starts
34         }
35
36         protected override void OnSleep()
37         {
38             // Handle when your app sleeps
39         }
40
41         protected override void OnResume()
42         {
43             // Handle when your app resumes
44         }
45     }
46 }
```

File - C:\Users\Grain\source\repos\gratux\PiBell\PiBell\PiBell.Android\MainActivity.cs

```
 1 using System;
 2 using Android.App;
 3 using Android.Content.PM;
 4 using Android.Runtime;
 5 using Android.Views;
 6 using Android.Widget;
 7 using Android.OS;
 8 using LibVLCSharp.Forms.Shared;
 9 using LibVLCSharp.Shared;
10 using Xamarin.Forms;
11 using Microsoft.AppCenter.Push;
12 using Microsoft.AppCenter;
13 using Android.Util;
14 using Android;
15 using Android.Support.V4.Content;
16 using Android.Support.V4.App;
17
18 namespace PiBell.Droid
19 {
20     [Activity(Label = "PiBell", Icon = "@mipmap/icon", Theme = "@style/
   MainTheme", MainLauncher = true,
21         ConfigurationChanges = ConfigChanges.ScreenSize | ConfigChanges.
   Orientation)]
22     public class MainActivity : global::Xamarin.Forms.Platform.Android.
   FormsAppCompatActivity
23     {
24         protected override void OnCreate(Bundle savedInstanceState)
25         {
26             TabLayoutResource = Resource.Layout.Tabbar;
27             ToolbarResource = Resource.Layout.Toolbar;
28
29             base.OnCreate(savedInstanceState);
30             global::Xamarin.Forms.Forms.Init(this, savedInstanceState);
31             LoadApplication(new App());
32
33             if (CheckSelfPermission(Manifest.Permission.RecordAudio) !=
   Permission.Granted)
34             {
35                 RequestPermissions(new[] { Manifest.Permission.RecordAudio
    }, 1);
36             }
37         }
38
39         protected override void OnPause()
40         {
41             base.OnPause();
42             MessagingCenter.Send("app", "OnPause");
43         }
44
45         protected override void OnRestart()
46         {
47             base.OnRestart();
48             MessagingCenter.Send("app", "OnRestart");
49         }
50     }
51 }
```

File - C:\Users\Grain\source\repos\gratux\PiBell\PiBell\PiBell.Android\AudioRecordingService.cs

```
 1 using System;
 2 using System.Collections.Generic;
 3 using System.IO;
 4 using System.Linq;
 5 using System.Text;
 6 using System.Threading;
 7 using Android.App;
 8 using Android.Content;
 9 using Android.Media;
10 using Android.OS;
11 using Android.Runtime;
12 using Android.Views;
13 using Android.Widget;
14 using Java.Nio;
15 using Xamarin.Forms;
16 using PiBell.Classes;
17 using System.Net.Sockets;
18 using System.Net;
19
20 [assembly: Dependency(typeof(PiBell.Droid.AudioRecordingService))]
21
22 namespace PiBell.Droid
23 {
24     class AudioRecordingService : Streamer.IAudioRecordingService
25     {
26         private Thread _recordingThread;
27         private volatile byte[] _audioBuffer;
28         private UdpClient _udp;
29         private IPEndPoint _target;
30         private bool _micActive;
31
32         public AudioRecordingService()
33         {
34             _audioBuffer = new byte[AudioRecord.GetMinBufferSize(48000,
   ChannelIn.Mono, Android.Media.Encoding.Pcm16bit) * 3];
35             //_audioRecord = new AudioRecord(AudioSource.Mic, 48000,
   ChannelIn.Mono, Android.Media.Encoding.Pcm16bit, _audioBuffer.Length);
36             _udp = new UdpClient();
37         }
38
39         public void Start(IPEndPoint target)
40         {
41             this._target = target;
42             _recordingThread = new Thread(RecordAudio) { IsBackground = true
    };
43             _recordingThread.Start();
44 #if DEBUG
45             Toast.MakeText(Android.App.Application.Context, "Started
   Recording", ToastLength.Short).Show();
46 #endif
47         }
48
49         private void RecordAudio()
50         {
51             var audioRecord = new AudioRecord(AudioSource.Mic, 48000,
   ChannelIn.Mono, Android.Media.Encoding.Pcm16bit, _audioBuffer.Length);
52             audioRecord.StartRecording();
53             while (true)
54             {
55                 try
```

```
56                  {
57                      audioRecord.Read(_audioBuffer, 0, _audioBuffer.Length);
58                      _udp.Send(_micActive ? _audioBuffer : new byte[
   _audioBuffer.Length], _audioBuffer.Length, _target);
59                  }
60                  catch (ThreadAbortException)
61                  {
62                      //before the thread stops
63                      audioRecord.Stop();
64                      audioRecord.Release();
65                      break;
66                  }
67                  catch (Exception e)
68                  {
69                      Console.WriteLine($"AudioRecord: {e.Message}");
70                      break;
71                  }
72              }
73          }
74
75          public void Stop()
76          {
77              _recordingThread.Abort();
78  #if DEBUG
79              Toast.MakeText(Android.App.Application.Context, "Stopped
   Recording", ToastLength.Short).Show();
80  #endif
81          }
82
83          public bool ToggleMic()
84          {
85              return _micActive = !_micActive;
86          }
87      }
88  }
```

File - C:\Users\Grain\source\repos\gratux\PiBell\PiBell\PiBell\MainPage.xaml

```
 1 <?xml version="1.0" encoding="utf-8"?>
 2
 3 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
 4              xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
 5              xmlns:local="clr-namespace:PiBell"
 6              xmlns:piBell="clr-namespace:PiBell;assembly=PiBell"
 7              xmlns:shared="clr-namespace:LibVLCSharp.Forms.Shared;assembly=
   LibVLCSharp.Forms"
 8              x:Class="PiBell.MainPage"
 9              Title="Main Page"
10              BackgroundColor="#ff1b1b1b">
11
12     <ContentPage.Content>
13         <Grid Margin="10" x:Name="MainGrid">
14             <Grid.RowDefinitions>
15                 <RowDefinition Height=".7*" />
16                 <RowDefinition Height="3*" />
17                 <RowDefinition Height="2*" />
18             </Grid.RowDefinitions>
19             <Grid.ColumnDefinitions>
20                 <ColumnDefinition Width="*" />
21             </Grid.ColumnDefinitions>
22
23             <Entry x:Name="EditMrl" Text="{Binding Mrl, Mode=TwoWay}" Grid.
   Row="0" Grid.Column="0" TextColor="White" />
24
25             <shared:VideoView x:Name="VideoView" Grid.Column="0" Grid.Row="1
   " MediaPlayer="{Binding MediaPlayer}"/>
26
27             <Grid x:Name="ButtonGrid" Grid.Row="2" Grid.Column="0">
28                 <Grid.ColumnDefinitions>
29                     <ColumnDefinition Width="*" />
30                     <ColumnDefinition Width="*" />
31                 </Grid.ColumnDefinitions>
32                 <Grid.RowDefinitions>
33                     <RowDefinition Height="*" />
34                     <RowDefinition Height="*" />
35                 </Grid.RowDefinitions>
36
37                 <ImageButton x:Name="BtConnect" Source="call.png"
   BackgroundColor="LimeGreen" Grid.Row="0"
38                              Grid.Column="0" Margin="10"
39                              Clicked="BtConnect_Clicked"/>
40                 <ImageButton x:Name="BtDisconnect" Source="Hangup.png"
   BackgroundColor="Red" Grid.Row="0"
41                              Grid.Column="1" Margin="10"
42                              Clicked="BtDisconnect_Clicked"/>
43
44                 <ImageButton x:Name="BtToggleSpeaker" Source="SpeakerMute.png
   " BackgroundColor="Transparent"
45                              Grid.Row="1" Grid.Column="0" Margin="10"
46                              Clicked="BtToggleSpeaker_Clicked">
47                     <VisualStateManager.VisualStateGroups>
48                         <VisualStateGroup x:Name="SpeakerStates">
49                             <VisualState Name="Mute">
50                                 <VisualState.Setters>
51                                     <Setter Property="Source"
52                                             Value="SpeakerMute.png" />
53                                 </VisualState.Setters>
54                             </VisualState>
```

File - C:\Users\Grain\source\repos\gratux\PiBell\PiBell\PiBell\MainPage.xaml

```
55                              <VisualState Name="Unmute">
56                                  <VisualState.Setters>
57                                      <Setter Property="Source"
58                                              Value="SpeakerUnmute.png" />
59                                  </VisualState.Setters>
60                              </VisualState>
61                          </VisualStateGroup>
62                      </VisualStateManager.VisualStateGroups>
63                  </ImageButton>
64
65                  <ImageButton x:Name="BtToggleMic" Source="MicMute.png"
   BackgroundColor="Transparent"
66                               Grid.Row="1" Grid.Column="1" Margin="10"
67                               Clicked="BtToggleMic_Clicked">
68                      <VisualStateManager.VisualStateGroups>
69                          <VisualStateGroup x:Name="MicStates">
70                              <VisualState Name="Mute">
71                                  <VisualState.Setters>
72                                      <Setter Property="Source"
73                                              Value="MicMute.png" />
74                                  </VisualState.Setters>
75                              </VisualState>
76                              <VisualState Name="Unmute">
77                                  <VisualState.Setters>
78                                      <Setter Property="Source"
79                                              Value="MicUnmute.png" />
80                                  </VisualState.Setters>
81                              </VisualState>
82                          </VisualStateGroup>
83                      </VisualStateManager.VisualStateGroups>
84                  </ImageButton>
85              </Grid>
86          </Grid>
87      </ContentPage.Content>
88  </ContentPage>
```