```
1 using MySql.Data.MySqlClient;
 2 using PlotterDataGH.Properties;
 3 using System.Data;
 4 using System.Diagnostics;
 5 using System.IO;
 6 using System.Reflection;
 7 using System.Windows;
 8 using System.Windows.Controls;
 9 using Newtonsoft.Json;
10 using PlotterDataGH;
11 using System.Threading;
12 using System.Net;
using System.Collections.Generic;
14 using Microsoft.Data.Sqlite;
15 using System.Text;
16 using System.Net.NetworkInformation;
17 using System.Windows.Media;
18 using System.Threading.Tasks;
19 using System;
20 using Microsoft.Win32.TaskScheduler;
21 using System.Net.Http;
22
23 namespace WpfApp2
24 {
25
       /// <summary>
       /// Interaction logic for UserControl1.xaml
26
27
       /// </summary>
28
       ///
29
       public partial class UserControl1 : UserControl
30
31
           bool cartridgeHidden = true;
32
           public int plotterId;
33
           public string plotterIp;
34
           public string serialnm;
           AutoResetEvent waiter = new AutoResetEvent(false);
35
36
           private static readonly HttpClient client = new HttpClient();
37
38
           public MainWindow ParentForm { get; set; }
39
40
           public UserControl1()
41
42
            {
                InitializeComponent();
43
               btnScan.IsEnabled = false;
44
45
            }
46
47
           private void SetTimer()
48
49
                //Set a timer between pings
50
                System.Windows.Threading.DispatcherTimer dispatcherTimer = new
                 System.Windows.Threading.DispatcherTimer();
51
                dispatcherTimer.Tick += dispatcherTimer Tick;
52
                dispatcherTimer.Interval = new TimeSpan(0, 10, 0);
53
                dispatcherTimer.Start();
54
           }
55
```

```
...ter Data\WpfApp2\WpfApp2\plotterControl.xaml.cs
56
            public void loadData()
57
58
                 //Load cartridge data and put it in a class called
                   cartridgeControl
59
                 DataTable dataTable = new DataTable();
60
                 SqliteConnection cnn;
                 SqliteCommand cmd = null;
61
                 cnn = new SqliteConnection("Data Source=plotterData.db;");
62
63
                 cnn.Open();
64
                 string query = string.Format("SELECT * FROM `cartridge reading`
65
                   where `parent id` = {0}", plotterId);
                 cmd = new SqliteCommand(query, cnn);
66
67
                 SqliteDataReader reader = cmd.ExecuteReader();
 68
 69
                 dataTable.Load(reader);
 70
71
                 foreach (DataRow row in dataTable.Rows)
72
73
                     RowDefinition rd = new RowDefinition();
74
                     rd.Height = GridLength.Auto;
75
                     plotterControl.RowDefinitions.Add(rd);
 76
                     cartridgeControl cartridgeControls = new cartridgeControl();
77
                     cartridgeControls.lblCartridgeNaam.Content = row
                       ["cartridge_model"].ToString();
                     cartridgeControls.lblCatridgeVolume.Content = row
78
                       ["volume"].ToString();
                     plotterControl.Children.Add(cartridgeControls);
79
80
                     Grid.SetRow(cartridgeControls,
                       plotterControl.RowDefinitions.Count);
81
                 }
82
                 RowDefinition last = new RowDefinition();
                 plotterControl.RowDefinitions.Add(last);
83
84
                 for (int x = plotterControl.RowDefinitions.Count - 1; <math>x > 0; x--)
85
86
                 {
                     plotterControl.RowDefinitions[x].Height = new GridLength(0);
87
88
                 }
89
90
                 SetTimer();
91
                 SendPing();
            }
92
93
94
            #region Check status
95
            private void dispatcherTimer Tick(object sender, EventArgs e)
96
             {
                 //Send a ping to the plotter and if it responds turn the light
97
                   green
98
                 SendPing();
99
             }
100
101
            private void SendPing()
102
103
                 Ping pingSender = new Ping();
104
105
                 // When the PingCompleted event is raised,
```

```
...ter Data\WpfApp2\WpfApp2\WpfApp2\plotterControl.xaml.cs
106
                 // the PingCompletedCallback method is called.
107
                 pingSender.PingCompleted += new PingCompletedEventHandler
                                                                                     P
                   (PingCompletedCallback);
108
109
                 // Create a buffer of 32 bytes of data to be transmitted.
110
                 string data = "aaaaaaaaaaaaaaaaaaaaaaaaaaaaa";
                 byte[] buffer = Encoding.ASCII.GetBytes(data);
111
112
113
                 // Wait 12 seconds for a reply.
114
                 int timeout = 12000;
115
116
                 // Set options for transmission:
                 // The data can go through 64 gateways or routers
117
118
                 // before it is destroyed, and the data packet
                 // cannot be fragmented.
119
120
                 PingOptions options = new PingOptions(64, true);
121
                 pingSender.SendAsync(plotterIp, timeout, buffer, options, waiter);
122
123
             }
124
125
             private void PingCompletedCallback(object sender,
                                                                                     P
               PingCompletedEventArgs e)
126
127
                 // If the operation was canceled, display a message to the user.
                 if (e.Cancelled)
128
129
130
                     ellStatus.Fill = new SolidColorBrush(Colors.Red);
131
                     btnScan.IsEnabled = false;
132
                     // Let the main thread resume.
133
134
                     // UserToken is the AutoResetEvent object that the main thread
135
                     // is waiting for.
136
                     ((AutoResetEvent)e.UserState).Set();
137
                 }
138
139
                 // If an error occurred, display the exception to the user.
                 if (e.Error != null)
140
141
                 {
142
                     ellStatus.Fill = new SolidColorBrush(Colors.Red);
                     btnScan.IsEnabled = false;
143
144
                     // Let the main thread resume.
145
146
                     ((AutoResetEvent)e.UserState).Set();
147
                 }
148
                 PingReply reply = e.Reply;
149
150
151
                 DisplayReply(reply);
152
153
                 // Let the main thread resume.
                 ((AutoResetEvent)e.UserState).Set();
154
             }
155
156
```

public void DisplayReply(PingReply reply)

if (reply == null)

157

158159

```
...ter Data\WpfApp2\WpfApp2\WpfApp2\plotterControl.xaml.cs
160
                 {
161
                     return;
162
                 }
163
164
                 if (reply.Status == IPStatus.Success)
165
166
                 {
167
                     ellStatus.Fill = new SolidColorBrush(Colors.Green);
168
                     btnScan.IsEnabled = true;
169
                 }
170
                 else
171
                 {
                     ellStatus.Fill = new SolidColorBrush(Colors.Red);
172
173
                     btnScan.IsEnabled = false;
174
                 }
             }
175
176
             #endregion
177
178
179
             private void btnExpand_Click(object sender, RoutedEventArgs e)
180
                 //Expand the control which will show the cartridges
181
182
                 if(cartridgeHidden)
183
184
                     for(int x = 0; x <= plotterControl.RowDefinitions.Count - 1; x →</pre>
185
                     {
                         plotterControl.RowDefinitions[x].Height = GridLength.Auto;
186
187
                     }
188
189
                     cartridgeHidden = false;
190
191
                     Button button = sender as Button;
192
                     MahApps.Metro.IconPacks.PackIconFontAwesome fe =
                       button.Content as
                       MahApps.Metro.IconPacks.PackIconFontAwesome;
193
                     fe.Kind =
                       MahApps.Metro.IconPacks.PackIconFontAwesomeKind.AngleDoubleU →
                       pSolid;
194
                 }
                 else
195
196
                 {
                     for(int x = plotterControl.RowDefinitions.Count - 1; x > 0;
197
                       x--)
198
                         plotterControl.RowDefinitions[x].Height = new GridLength
199
                         (0);
200
                     }
201
                     cartridgeHidden = true;
202
203
                     Button button = sender as Button;
204
                     MahApps.Metro.IconPacks.PackIconFontAwesome fe =
                       button.Content as
                                                                                      P
                       MahApps.Metro.IconPacks.PackIconFontAwesome;
205
                     fe.Kind =
                       MahApps.Metro.IconPacks.PackIconFontAwesomeKind.AngleDoubleD →
```

```
...ter Data\WpfApp2\WpfApp2\plotterControl.xaml.cs
                       ownSolid;
206
                 }
207
             }
208
209
            private void Button Click(object sender, RoutedEventArgs e)
210
                 AddPlotter addPlotter = new AddPlotter();
211
212
                 addPlotter.ParentForm = ParentForm;
213
                 addPlotter.editForm(plotterId);
214
                 addPlotter.editingMode = true;
215
                 addPlotter.Show();
216
            }
217
218
            #region Scanning
219
220
            private void Button Click 1(object sender, RoutedEventArgs e)
221
222
                 //Run a scan of this specific plotter
223
                RunScan(lblMerk.Uid.ToString(), plotterIp,
                                                                                     P
                   lblNaam.Content.ToString());
224
                MahApps.Metro.IconPacks.PackIconFontAwesome fe = btnScan.Content
                   as MahApps.Metro.IconPacks.PackIconFontAwesome;
225
                 fe.Kind =
                   MahApps.Metro.IconPacks.PackIconFontAwesomeKind.SyncSolid;
226
                btnScan.IsEnabled = false;
            }
227
228
229
            public void RunScan(string Merk, string IP, string Naam)
230
                 var debugField = Path.GetDirectoryName(
231
232
        Assembly.GetExecutingAssembly().GetName().CodeBase);
233
                 debugField = debugField.Substring(6);
234
235
                 var filename = debugField + @"/ghWebscraper.exe";
236
237
                 //Start the Converted python file and pass the paramater
238
239
                 string arguments = string.Format(@"{0} {1} {2} {3}", Merk, IP,
                   Settings.Default.sendData, Naam);
240
                 ellStatus.Fill = new SolidColorBrush(Colors.Orange);
241
242
243
                 //Process myProcess = new Process();
244
                 //myProcess.Exited += new EventHandler(myProcess Exited);
245
                 //myProcess.StartInfo.FileName = filename;
246
                 //myProcess.StartInfo.Arguments = arguments;
247
                 //myProcess.Start();
248
249
                 doStuff(filename, arguments);
250
            }
251
            async System.Threading.Tasks.Task doStuff(string fileName, string
252
               args)
253
             {
                 ParentForm.DisableWhileScanning();
254
```

await RunProcessAsync(fileName, args);

255

```
...ter Data\WpfApp2\WpfApp2\WpfApp2\plotterControl.xaml.cs
```

```
256
257
                 //MahApps.Metro.IconPacks.PackIconFontAwesome fe = btnScan.Content →
                    as MahApps.Metro.IconPacks.PackIconFontAwesome;
258
                 //fe.Kind =
                   MahApps.Metro.IconPacks.PackIconFontAwesomeKind.BinocularsSolid;
259
                 //btnScan.IsEnabled = true;
260
261
                 ParentForm.fillerGrid.RowDefinitions.Clear();
262
                 ParentForm.fillerGrid.Children.Clear();
263
                 ParentForm.LoadData();
264
            }
265
            public static async Task<int> RunProcessAsync(string fileName, string →
266
               args)
267
268
                 using (var process = new Process
269
                     StartInfo =
270
271
            {
272
                 FileName = fileName, Arguments = args,
273
                 UseShellExecute = false, CreateNoWindow = true,
                 RedirectStandardOutput = true, RedirectStandardError = true
274
275
             },
276
                     EnableRaisingEvents = true
277
                 })
278
                 {
279
                     return await RunProcessAsync(process).ConfigureAwait(false);
280
                 }
281
            private static Task<int> RunProcessAsync(Process process)
282
283
284
                 var tcs = new TaskCompletionSource<int>();
285
286
                 process.Exited += (s, ea) => tcs.SetResult(process.ExitCode);
                 process.OutputDataReceived += (s, ea) => Console.WriteLine
287
                   (ea.Data);
                 process.ErrorDataReceived += (s, ea) ⇒ Console.WriteLine("ERR: " →
288
                   + ea.Data);
289
290
                 bool started = process.Start();
291
                 if (!started)
292
                 {
293
                     //you may allow for the process to be re-used (started =
                       false)
                     //but I'm not sure about the guarantees of the Exited event in ₹
294
                        such a case
295
                     throw new InvalidOperationException("Could not start process: →
                       " + process);
296
                 }
297
298
                 process.BeginOutputReadLine();
299
                 process.BeginErrorReadLine();
300
301
                 return tcs.Task;
302
303
            }
```

```
...ter Data\WpfApp2\WpfApp2\WpfApp2\plotterControl.xaml.cs
```

```
7
```

```
304
305
             #endregion
306
307
             #region Send Data
308
             private void btnSend Click(object sender, RoutedEventArgs e)
309
                 //Send data to Goedhart Group
310
311
                 Await();
312
             }
313
314
             async System.Threading.Tasks.Task Await()
315
                 var task = SendPlotterAsync();
316
317
                 int timeout = 1000;
                 if (await System.Threading.Tasks.Task.WhenAny(task,
318
                   System.Threading.Tasks.Task.Delay(timeout)) == task)
319
320
                     // task completed within timeout
321
                 }
                 else
322
323
                 {
                     MessageBox.Show("Verbinding met de Goedhart Servers kon niet
324
                       gemaakt worden");
325
                 }
326
             }
327
328
             async System.Threading.Tasks.Task SendPlotterAsync()
329
330
                 var values = new Dictionary<string, string>
331
332
                     { "postType", "Plotter" },
333
                     { "serial_number", serialnm },
                     { "model_id", lblMerk.Uid.ToString() },
334
335
                     { "meters printed", lblMeterstand.Content.ToString() },
                     { "naam", lblNaam.Content.ToString() },
336
                     { "IP", plotterIp },
337
                     { "bedrijfs_Naam", Settings.Default.bedrijfsNaam }
338
339
                 };
340
341
                 var content = new FormUrlEncodedContent(values);
342
                 var response = await client.PostAsync("http://10.0.0.125/",
343
                   content);
344
345
                 var responseString = await response.Content.ReadAsStringAsync();
346
347
                 DataTable dataTable = new DataTable();
348
                 SqliteConnection cnn;
349
                 SqliteCommand cmd = null;
350
                 cnn = new SqliteConnection("Data Source=plotterData.db;");
351
                 cnn.Open();
352
                 string query = string.Format("SELECT * FROM `cartridge reading`
353
                   where `parent_id` = {0}", plotterId);
354
                 cmd = new SqliteCommand(query, cnn);
355
```

```
...ter Data\WpfApp2\WpfApp2\plotterControl.xaml.cs
```

386

```
356
                 SqliteDataReader reader = cmd.ExecuteReader();
357
                 dataTable.Load(reader);
358
359
                 foreach (DataRow row in dataTable.Rows)
360
                 {
                     SendCartridgeAsync(responseString, row
361
                       ["cartridge_model"].ToString(), row["volume"].ToString(),
                       row["max_volume"].ToString());
362
                 }
363
             }
364
             async System.Threading.Tasks.Task SendCartridgeAsync(string parent id, →
365
                string cartridge_model, string volume, string max_volume = null)
366
                 var values = new Dictionary<string, string>
367
368
                 {
369
                     { "postType", "Cartridge" },
                     { "parent_id", parent_id },
370
371
                     { "cartridge_model", cartridge_model },
                     { "volume", volume },
372
                     { "max_volume", max_volume }
373
                 };
374
375
376
                 var content = new FormUrlEncodedContent(values);
377
                 var response = await client.PostAsync("http://10.0.0.125/",
378
                   content);
379
380
                 var responseString = await response.Content.ReadAsStringAsync();
             }
381
382
383
             #endregion
384
         }
385 }
```