```
1  using Microsoft.Data.Sqlite;
2  using Microsoft.Win32.TaskScheduler;
3  using PlotterDataGH.Properties;
4  using System.Data;
5  using System.Reflection;
6  using System.Windows;
7  using System.Windows.Controls;
8
9  namespace WpfApp2
10 {
11     /// <summary>
12     /// Interaction logic for AddPlotter.xaml
13     /// </summary>
14     public partial class AddPlotter : Window
15     {
16         public bool editingMode = false;
17         int plotterId = 0;
18         string serialNumber = "";
19
20         public MainWindow ParentForm { get; set; }
21
22         public AddPlotter()
23         {
24             InitializeComponent();
25             btnDelete.Visibility = Visibility.Hidden;
26
27             DataTable dataTable = new DataTable();
28             SqliteConnection cnn;
29             SqliteCommand cmd = null;
30             cnn = new SqliteConnection("Data Source=plotterData.db;");
31             cnn.Open();
32
33             string query = string.Format("SELECT * from models");
34             cmd = new SqliteCommand(query, cnn);
35
36             SqliteDataReader reader = cmd.ExecuteReader();
37             dataTable.Load(reader);
38
39
40             foreach (DataRow dataRow in dataTable.Rows)
41             {
42                 ComboBoxItem item = new ComboBoxItem();
43
44                 item.Uid = dataRow["id"].ToString();
45                 item.Content = dataRow["plotter_type"].ToString();
46
47                 cbxPlotterType.Items.Add(item);
48             }
49         }
50
51         public void editForm(int plotterId_m)
52         {
53             plotterId = plotterId_m;
54             DataTable dataTable = new DataTable();
55             SqliteConnection cnn;
56             SqliteCommand cmd = null;
```

```
57              cnn = new SqliteConnection("Data Source=plotterData.db;");
58              cnn.Open();
59
60              string query = string.Format("SELECT printer_data.serial_number,
                  printer_data.naam, printer_data.ip, models.plotter_type from
                  `printer_data` INNER JOIN models ON printer_data.model_id =
                  models.id where printer_data.id = {0} Limit 1", plotterId);
61              cmd = new SqliteCommand(query, cnn);
62
63              SqliteDataReader reader = cmd.ExecuteReader();
64              dataTable.Load(reader);
65
66              btnDelete.Visibility = Visibility.Visible;
67              serialNumber = dataTable.Rows[0]["serial_number"].ToString();
68              tbxPlotIP.Text = dataTable.Rows[0]["IP"].ToString();
69              tbxPlotNaam.Text = dataTable.Rows[0]["Naam"].ToString();
70              cbxPlotterType.Text = dataTable.Rows[0]["plotter_type"].ToString
                  ();
71          }
72
73          private void btnCancel_Click(object sender, RoutedEventArgs e)
74          {
75              this.Close();
76          }
77
78          private void btnNext_Click(object sender, RoutedEventArgs e)
79          {
80              if (editingMode)
81              {
82                  SqliteConnection cnn;
83                  cnn = new SqliteConnection("Data Source=plotterData.db;");
84                  cnn.Open();
85
86                  string query = string.Format("UPDATE printer_data SET naam =
                      '{0}', ip = '{1}' where id = {2}", tbxPlotNaam.Text,
                      tbxPlotIP.Text, plotterId);
87                  //create command and assign the query and connection from the
                      constructor
88                  SqliteCommand cmd = new SqliteCommand(query, cnn);
89
90                  //Execute command
91                  cmd.ExecuteNonQuery();
92
93                  cnn.Close();
94
95                  ParentForm.RunScan((cbxPlotterType.SelectedItem as
                      ComboBoxItem).Uid.ToString(), tbxPlotIP.Text,
                      tbxPlotNaam.Text);
96
97                  this.Close();
98              }
99              else if (tbxPlotIP.Text != "" && tbxPlotNaam.Text != "" &&
                  cbxPlotterType.Text != "")
100             {
101                 //string connetionString;
102                 //MySqlConnection cnn;
```

```
103                 //connetionString = @"server=localhost;user
                      id=root;password=;database=printer_data_test;";
104                 //cnn = new MySqlConnection(connetionString);
105                 //cnn.Open();
106
107
108                 ///CHANGE///
109                 //string query = string.Format("INSERT INTO printer_data
                      (bedrijfs_Naam, contactpersoon, email, telefoonnummer)
                      VALUES('{0}', '{1}', '{2}', '{3}')", tbxBedrijfsnaam.Text,
                      tbxContactpersoon.Text, tbxEmail.Text,
                      tbxTelefoonnummer.Text);
110                 //create command and assign the query and connection from the
                      constructor
111                 // MySqlCommand cmd = new MySqlCommand(query, cnn);
112
113                 //Execute command
114                 //cmd.ExecuteNonQuery();
115
116                 //cnn.Close();
117                 //this.Close();
118
119                 //List<plotter> _plotter = new List<plotter>();
120                 //_plotter.Add(new plotter()
121                 //{
122                 //    ID = Settings.Default.plotterID,
123                 //    plotterNaam = tbxPlotNaam.Text,
124                 //    plotterIP = tbxPlotIP.Text,
125                 //    plotterType = tbxPlotType.Text,
126                 //    meters_printed = "0"
127                 //}) ;
128
129                 //string json = System.Text.Json.JsonSerializer.Serialize
                      (_plotter);
130
131                 //string path = @"Plotter Data\Plotter.json";
132
133                 //if (Directory.Exists(path))
134                 //{
135                 //    File.WriteAllText(path, json);
136                 //}
137                 //else
138                 //{
139                 //    Directory.CreateDirectory("Plotter Data");
140                 //    File.WriteAllText(path, json);
141                 //}
142
143                 //Settings.Default.plotterID++;
144
145                 ParentForm.RunScan((cbxPlotterType.SelectedItem as
                      ComboBoxItem).Uid.ToString(), tbxPlotIP.Text,
                      tbxPlotNaam.Text);
146
147
148                 //Get the service on the local machine
149                 using (TaskService ts = new TaskService())
```

```
150                 {
151
152                     var debugField = System.IO.Path.GetDirectoryName(
153  Assembly.GetExecutingAssembly().GetName().CodeBase);
154
155                     debugField = debugField.Substring(6);
156
157                     var filename = debugField + @"/ghWebscraper.exe";
158
159                     //var scheduler = debugField + @"/Test";
160
161                     //Start the Converted python file and pass the paramater
162                     string arguments = string.Format(@"{0} {1} {2} {3}",
                        (cbxPlotterType.SelectedItem as ComboBoxItem).Uid.ToString
                        (), tbxPlotIP.Text, Settings.Default.sendData,
                        tbxPlotNaam.Text);
163
164                     TaskDefinition td = ts.NewTask();
165
166                     if (ts.GetTask("Plotter Scanner") != null)
167                     {
168                         td = ts.GetTask("Plotter Scanner").Definition;
169                     }
170
171                     // Create a new task definition and assign properties
172
173                     td.RegistrationInfo.Description = "Scans plotter";
174                     td.RegistrationInfo.Author = "Goedhart Groep";
175
176                     if (td.Triggers == null)
177                     {
178                         // Create a trigger that will fire the task at this
                        time every day
179                         td.Triggers.Add(new DailyTrigger { DaysInterval =
                        1 });
180                     }
181
182                     // Create an action that will launch Notepad whenever the
                        trigger fires
183                     td.Actions.Add(new ExecAction(filename, arguments,
                        debugField));
184
185
186
187                     // Register the task in the root folder
188                     ts.RootFolder.RegisterTaskDefinition(@"Plotter Scanner",
                        td);
189
190
191                 }
192
193                 using (TaskService ts = new TaskService())
194                 {
195                     var debugField = System.IO.Path.GetDirectoryName(
196          Assembly.GetExecutingAssembly().GetName().CodeBase);
197
```

```
198                     debugField = debugField.Substring(6);
199
200                     var filename = debugField + @"/NewWay.exe";
201
202                     TaskDefinition td = ts.NewTask();
203
204                     if (ts.GetTask("Plotter Scanner") != null)
205                     {
206                         td = ts.GetTask("Plotter Scanner").Definition;
207                     }
208
209                     // Create a new task definition and assign properties
210
211                     td.RegistrationInfo.Description = "Scans plotter";
212                     td.RegistrationInfo.Author = "Goedhart Groep";
213
214                     if (td.Triggers.Count == 0)
215                     {
216                         // Create a trigger that will fire the task at this
                        time every day
217                         td.Triggers.Add(new DailyTrigger { DaysInterval =
                        1 });
218                     }
219
220                     // Create an action that will launch Notepad whenever the
                        trigger fires
221                     td.Actions.Add(new ExecAction(filename, null,
                        debugField));
222
223                     // Register the task in the root folder
224                     ts.RootFolder.RegisterTaskDefinition(@"Plotter Scanner",
                        td);
225                 }
226                 this.Close();
227             }
228             else
229             {
230                 MessageBox.Show("Vul alstjeblieft alle velden in");
231             }
232         }
233
234         private void btnDelete_Click(object sender, RoutedEventArgs e)
235         {
236             MessageBoxResult messageBoxResult = System.Windows.MessageBox.Show
                  ("Weet je het zeker?", "Plotter Verwijderen",
                  System.Windows.MessageBoxButton.YesNo);
237             if (messageBoxResult == MessageBoxResult.Yes)
238             {
239                 SqliteConnection cnn1;
240                 cnn1 = new SqliteConnection("Data Source=plotterData.db;");
241                 cnn1.Open();
242
243                 string query = string.Format("Delete from printer_data where
                    serial_number = '{0}'", serialNumber);
244                 //create command and assign the query and connection from the
                    constructor
```

```
245                 SqliteCommand cmd1 = new SqliteCommand(query, cnn1);
246
247                 //Execute command
248                 cmd1.ExecuteNonQuery();
249
250                 cnn1.Close();
251
252                 // Get the service on the local machine
253                 using (TaskService ts = new TaskService())
254                 {
255                     if (ts.GetTask("Plotter Scanner") != null)
256                     {
257                         ts.RootFolder.DeleteTask("Plotter Scanner");
258                     }
259                 }
260
261                 ParentForm.fillerGrid.RowDefinitions.Clear();
262                 ParentForm.fillerGrid.Children.Clear();
263                 ParentForm.LoadData();
264                 ParentForm.TaskCreater();
265
266                 this.Close();
267
268             }
269         }
270     }
271 }
272
```