Raven

H5 – Projekt opgave

Niklas W. Micheelsen & Rasmus W. Knudsen

Indholdsfortegnelse

Forord

Denne rapport skrevet ud fra vores projekt under H5, den er skrevet af Niklas W. Micheelsen & Rasmus W. Knudsen. Rapporten omhandler en prototype af vores 'Vogn monitorering og analyse' softwarepakke.

Vores projekt er tænkt som at være en prototype af en såkaldt "black box" som kan sælges som en del af en serviceaftale til større transport firmaer, såsom DHL, UPS, 3x34 osv.

Vores Development prototype er delt op i tre dele – OBD-II connector, Arduino og en Android mobil. Når vores "proof of concept" er fuldt fungerende og Raven-Desktop er færdigudviklet kan vi begynde på fase 2 af Raven.

I fase 2 vil vi arbejde på at minimere prototypen til en lille kasse som man vil kunne efterlade tilsluttet i bilens OBD-II stik. Med en indbygget GPS og et 3G GSM modem vil den kunne opdatere og sende data til Raven servers.

Navnet Raven har vi valgt som en reference til Odins to ravne, Hugin og Munin, som overvåger landskabet fra luften.

Indledning

Raven GPS er en suite af software designet for at give transportselvskaber muligheden for at tracke og analysere sine chaufførers, vogne og deres ruter.

Dette bliver udført ved hjælp af bilens OBD-II stik, der er en ny standard inden for Auto industrien. OBD står for "On-Board Diagnostics" og giver adgang til bilens CAN Bus som giver adgang til 'hjernen' af bilen dvs. data så som hastighed, RPM, Motor stress og meget mere. Disse informationer bliver sendt til chaufførens smartphone over Bluetooth, der derefter vil blive sendt til Raven GPS' database.

Et par eksempler på i hvilken situation dette vil være brugbart er f.eks. En arbejdsgiver får en klage over en medarbejders vanvidskørsel. Der vil arbejdsgiver kunne åbne Raven programmet, skrive medarbejderens nummerplade, stelnummer eller potentielt andre identificerende detaljer og se alle ruter en specifik medarbejder har foretaget. Arbejdsgiver vil så kunne trykke på en given rute og verificere at der er hold i klagen og tage aktion.

Eller at en af firmaets vogne brød sammen eller nedkøling af læsset ikke fungerer ordentligt så kan firmaet finde ud af hvor den er ved at slå den op og tilkalde den nærmeste vogn som er tilgængelig til dens position.

Projektstyring

Commented [N1]: Referencer til Trello
Billeder af Trello

Til projektstyring har vi brugt Kanbanboard og servicen "Trello", Trello giver os mulighed for hurtigt at kunne oprette nye opgraver med detaljeret beskrivelser. Trello virker på den måde at man kan organisere opgaver som digitale post-it notes og tilføje dem til forskellige kolonner.

Når en ny opgave skal oprettes starter den ved at blive tilføjet til en af vores 'To-do' kolonner for vores forskellige projekt dele. Hvis denne opgave angiver en opgave for Raven-Arduino vil den blive markeret som en Arduino opgave og tilføjes til vores 'To-Do' kolonne, når opgaven bliver begyndt på bliver opgaven flyttet til 'In-progress' og eventuelle under opgaver vil derefter blive tjekket af på listen en for en. Når alle under opgaver er fuldført vil opgaven bliver flyttet en sidste gang, til vores 'Done' kolonne der over tid vil blive længere og længere.

Problemstilling

Hvordan kan det være, at vi har valgt at lave dette produkt?

Det er hovedsageligt fordi at vi har stor interesse inden for biler, men også fordi at det ville være spændende at lave et projekt som både var afhængelig af hardware og som benyttede flere platforme, noget som ligner et realistisk produkt og som inkluderede et område vi ikke havde særligt meget kendskab til.

Hvor kom ideen fra?

Ideen kom fra at en af vores fædre som havde lavet noget lignede for mange år siden. Noget som han forslog kunne erstattes hvis nogle tog konceptet og forbedrede det. Realistisk set så er hvad vi har lavet en prototype da for at det kan erstatte den gamle version mangler der meget funktionalitet. Men det er hovedsageligt der inspirationen kom fra.

Hvad ville vi gerne lære igennem arbejdet?

Vi ville gerne lære mere om RTOS da det er et ekstremt brugbart værktøj i den virkelige verden, det har været udfordrende at arbejde med Arduino.

Problemformulering

Problemer, bekymringer eller ting der bør overvejes inden for denne type af produkt.

Hvordan vil vi overføre data fra Arduino til Android?

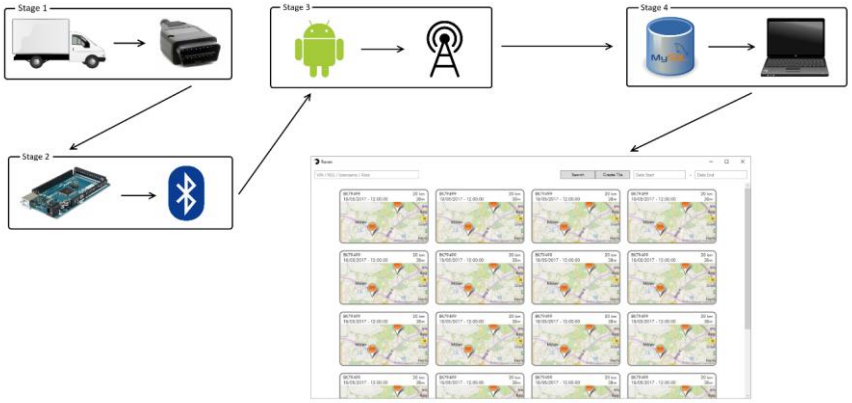
Til vores prototype version valgte vi at bruge Bluetooth imellem Arduino og Android da et Bluetooth modul til Arduino ikke koster mange penge.

Vi havde overvejet et GSM modul (data forbindelse via SIM kort), så var der ikke behov for Android, men vi syntes at et GSM modul ville være for dyrt i første omgang.

Hvordan vil vi lagre data?

Vi valgte at lagre vores data i Json format på en SQL server, det endte med at blive en LONGTEXT da det største antal tegn tilladt. Der er sikkert hunderede vis af bedre muligheder, men da det er relativt hurtigt at lave indsæt eller udtræk og at vi indtil videre ikke har ramt grænsen for vores logs.

Rigt billede



Krav

Desktop:

- Søgefunktion hvor brugere kan filtrere ruter efter registrerings numre, førerens brugernavn og andre parametre.

- Vise en boks for hver tur kørt, som indeholder generel information og et kort over ruten. Information som, registrerings nummer, distance, varighed, dato og tid på starten af turen.
- Detaljeret tur visning som viser den fulde rute, punkter som er blevet registreret i løbet af turen som kan vise dato, tid, hastighed, omdrejninger i minuttet, breddegrad og længdegrad på det tidspunkt hvis brugeren klikker på et af punkterne.

Android:

- Modtag data fra Bluetooth modulet på Arduino'en.
- Filtrere modtaget Json linjer og formatere til gyldig og læseligt Json logs.
- Send filtreret data til SQL via GSM/Data forbindelse på Android.

Arduino:

- Læs fra OBD-II.
- Efter en læse cyklus start sending.
- Send data til Android i Json format over Bluetooth til Android.

Database:

- Skal have et login table med registreringsnummer, brugernavn og krypteret password.
- Skal have et trip table hvor vi kan gemme ture med start/slut tidspunkt, førerens brugernavn, førerens registrerings nummer og selve loggen.

Udviklingsmiljøer

Raven-Desktop

Miljø: Visual Studio 2017
Raven-Desktop er skrevet i WPF (C#, XAML) ved brug af Bing Maps SDK

Raven-Arduino

Miljø: Arduino IDE
Raven-Arduino er skrevet i C.

Raven-Android

Miljø: Android Studio
Raven-Android er skrevet i java.

Værktøjer

Balsamiq Mockups 3

Visual Studio

Android Studio

Arduino IDE

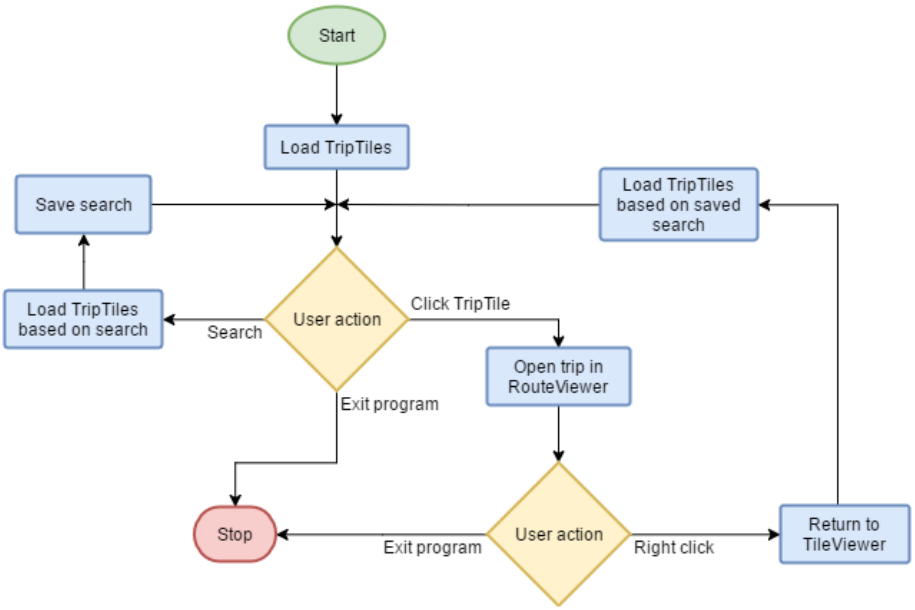
MYSQL

GitHub

Trello

Flowcharts

Commented [N2]: Mangler Raven-Desktop LoginWindow diagram
Mangler Raven-Android diagram
Mangler Raven-Arduino diagram



- TripTiles er bokse der viser general information omkring en tur, sammen med et kort af den kørte rute.

Kodebeskrivelse

Raven-Desktop

1.1.0 – MainWindow.xaml.cs

Er en klasse der indeholder fire globale variabler, TripTileCollection der indeholder ture af vores type 'Tile'(4.1.0), ConnectionString som indeholder addressen, database navnet og login til vores server, Username som vi bruger til første søgning af TripTiles og visning af hvem man er logget ind som i vindue titlen, Username bliver sat af LoginWindow. LastSearch som bliver brugt til at gemme den seneste søgning som brugeren har fortaget sig, grunden til at vi gemmer søgningen er fordi, når brugeren har klikket på et 'TripTile' viser det et stort kort omkring ruten, men når brugeren trykker højreklik så sender vi brugeren tilbage til hvor man kan se 'TripTiles', der bruger vi 'LastSearch' til at finde de samme 'TripTiles' som brugeren kiggede på sidst uden at brugeren behøver at søge igen.

1.1.1 - MainWindow

Er en constructor som laver et nyt LoginWindow(2.1.0) element, og så skjuler den sig selv så MainWindow hverken vises på skærmen eller i værktøjslinjen. Derefter hiver den LoginWindow frem og afventer at LoginWindow bliver lukket, afhængelig af 'LoginSuccess' boolean fra LoginWindow lukker den enten alle vinduer eller viser sig selv igen og opdaterer nogle variabler inden den begynder at indlæse TripTiles.

1.1.2 – LoadTripTiles

Er en metode som starter en forbindelse via en 'connection string' til vores Mysql server med udtaget "SELECT * FROM trips" hvilket betyder, hent alle linjer i 'trips' tabellen og læg det i et DataTable. Efter den har hentet informationerne kalder den GenerateTiles metoden og medsender DataTable'et.

1.1.3 – LoadTripTiles(string search)

Er en metode som næsten er helt ens med LoadTripTiles(1.1.2), denne metode laver bare udtrækket "SELECT * FROM trips WHERE CONCAT_WS('', time_started, time_ended, driver_username, driver_reg) LIKE '%{search}%'" hvis søgefeltet ikke er tomt. Derefter kalder den GenerateTiles(1.1.4) med DataTable'et.

1.1.4 – GenerateTiles

Er en metode som først tømmer 'TripTileSelection' og derefter begynder at løbe igennem alle rækker som den fik fra LoadTripTiles(1.1.2/1.1.3). For hver række fisker den informationer ud af rækken, såsom 'rowId', 'title', 'date', 'dateStart', 'dateEnd' og 'logs'.

Så tager den 'logs' og deserialiserer indholdet til en liste ved navn 'results' af 'RootObject' typen. RootObject er en klasse som benytter JSON attributter for at sortere data til det korrekte felt.

Efter den har deserialiseret loggen begynder den at tjekke alle resultaterne igennem for at finde fire koordinater. Det mest nordlige, sydlige, vestlige og østlige punkt af alle koordinaterne i 'results'

Efter den har fundet de fire koordinater laver den en 'LocationRect' ud fra de koordinater men med en forskydning for at synliggøre vores "knappenåle" som vi vil sætte på korterne. LocationRect kan bruges til at finde et midtpunkt og grænser for hvor meget kortene må zoomes ind.

Så danner den start og slut lokationer ud fra de første koordinater i listen og de sidste. De koordinater bruges til at lave vores grønne start knappenål og vores røde slut knappenål.

Herefter danner den et 'MapLayer' kaldet 'polyLineLayer' hvilket er et grafisk lag du kan smide hen over kortet, det bruger vi til at vise den kørte rute, men først skal det tegnes. Det gør den ved at løbe alle indekser igennem 'results' og lave en 'MapPolyLine' imellem hvert indeks ved hjælp af indeksernes koordinater og tilføjer det til 'polyLineLayer' som et barn, til sidst har vi et 'MapLayer' som indeholder vores rute.

Derefter formaterer den datoen til at se pænere ud.

Så laver den en 'distance' variabel og udregner den totale distance for ruten ved at løbe igennem 'results' og udregne distancen imellem to koordinater ved at kalde CalculateDistance(1.1.6) sammen med to 'Locations' og lægger den returnerede meter værdi til 'distance' for hvert loop. Efter den er færdig med for løkken dividerer den 'distance' med 1000 for at konvertere til km.

Derefter laver den en 'duration' streng og udregner turens varighed ved at finde differencen imellem 'dateStart' og 'dateEnd', og formaterer 'duration' til en bedre format for brugeren at læse.

Til sidst skaber den en ny 'Tile'(4.1.0) i vores 'TripTileCollection' med de relevante informationer som den har forberedt.

Eksempel:

```
TripTileCollection.Add(new Tile(rowId, startLocation, endLocation, bounds, polyLineLayer, title, date, distance, duration));
```

1.1.5 – RouteViewerLoadTrip

Er en metode som modtager et 'tripId' som integer og laver et "SELECT * FROM trips WHERE id={tripId}" udtræk fra vores SQL server ved hjælp af 'tripId'.

Den tager loggen fra den række der matcher id'et og deserialiserer indholdet til en liste som den derefter retunerer til det sted 'RouteViewerLoadTrip' blev kaldt fra.

1.1.6 – CalculateDistance

Er en metode som modtager to 'Locations' (previousPin og currentPin), den tager deres højdegrad og længdegrad, og laver to 'GeoCoordinate'. 'GeoCoordinate' klassen har en metode som hedder 'GetDistanceTo' som udregner distancen for os, det retunerer vi til det sted 'CalculateDistance' blev kaldt fra.

1.1.7 – SearchBtn_OnClick

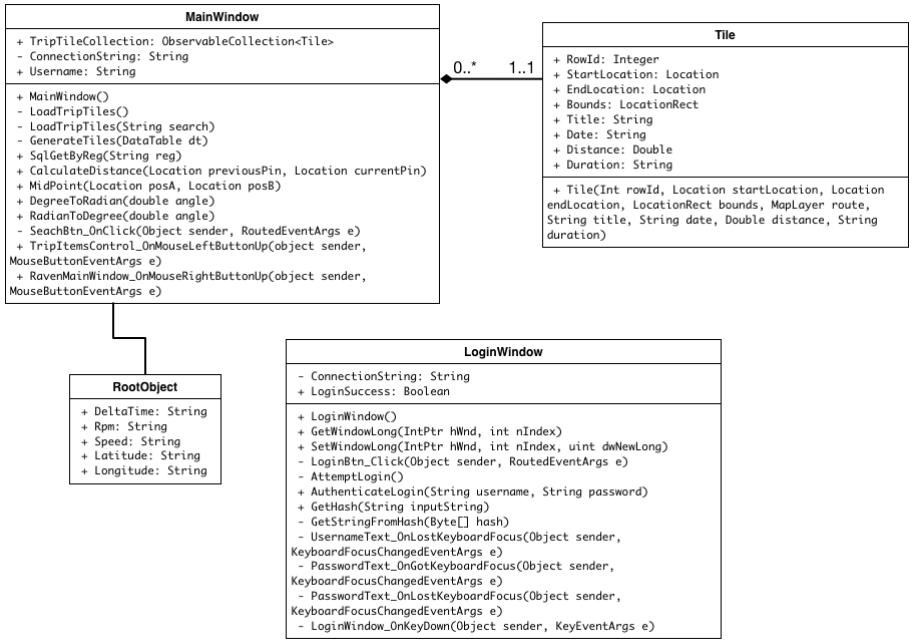
1.1.8 – TripItemsControl_OnMouseLeftButtonUp

1.1.9 – RavenMainWindow_OnMouseRightButtonUp

1.2.1 – RootObject

Klasse Diagram

Raven-Desktop



Commented [N3]: Mangler Raven-Android diagram

Kildekode

Desktop

1.1.0 - MainWindow.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Data;
using System.Windows;
using System.Windows.Controls;
using System.Device.Location;
using System.Globalization;
using System.Windows.Input;
using System.Windows.Media;
using MySql.Data.MySqlClient;
using Microsoft.Maps.MapControl.WPF;
using Newtonsoft.Json;
using Raven.Windows;

namespace Raven {
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow {
        public static ObservableCollection<Tile> TripTileCollection { get; set; } =
            new ObservableCollection<Tile>(); // Collection of TripTiles, using custom Tile type

        private const string ConnectionString = "Server=raven-
gps.com;Database=raven;Uid=root;Pwd=#### ";
        public static string Username = "";
        public static string LastSearch = "";
```

1.1.1 – MainWindow

```
        public MainWindow() {
            // Initiate LoginWindow element
            var loginWindow = new LoginWindow();
            Hide();
            loginWindow.Show();
            InitializeComponent();

            loginWindow.Closed += delegate {
                if (loginWindow.LoginSuccess) {
                    Title = $"Raven - Logged in as {Username}";
                    Show();
                    if (Username == "DebugUser") {
                        LoadTripTiles();
                    }
                    else {
                        LoadTripTiles(Username);
                        LastSearch = Username;
                    }
                }
                else {
                    Close();
                }
            };
        }
```

1.1.2 – LoadTripTiles

```
private void LoadTripTiles() {
    var connection = new MySqlConnection(ConnectionString);
    var dt = new DataTable();
    connection.Open();

    try {
        var command = connection.CreateCommand();
        command.CommandText = "SELECT * FROM trips";

        using (var dr = command.ExecuteReader()) {
            dt.Load(dr);
            GenerateTiles(dt);
        }
    }
    catch (MySqlException exception) {
        MessageBox.Show(exception.ToString());
    }
}
```

1.1.3 – LoadTripTiles (med søge parameter)

```
private void LoadTripTiles(string search) {
    var connection = new MySqlConnection(ConnectionString);
    var dt = new DataTable();
    connection.Open();

    try {
        var command = connection.CreateCommand();

        if (search == String.Empty) {
            command.CommandText = "SELECT * FROM trips";
        }
        else {
            command.CommandText = $"SELECT * FROM trips WHERE CONCAT_WS('', time_started,
time_ended, driver_username, driver_reg) LIKE '%{search}%'";
        }

        using (var dr = command.ExecuteReader()) {
            dt.Load(dr);
            GenerateTiles(dt);
        }
    }
    catch (MySqlException exception) {
        MessageBox.Show(exception.ToString());
    }
}
```

1.1.4 – GenerateTiles

```
public void GenerateTiles(DataTable dt) {
    TripTileCollection.Clear();
    foreach (DataRow row in dt.Rows) {
        try {
            var rowId = int.Parse(row["id"].ToString());
            var logs = row["log_file"].ToString();
            var title = row["driver_reg"].ToString();
            var date = row["time_started"].ToString();
            var dateStart = DateTime.Parse(row["time_started"].ToString());
            var dateEnd = DateTime.Parse(row["time_ended"].ToString());
```

```
            var results = JsonConvert.DeserializeObject<List<RootObject>>(logs);

            // Set Bounds to fit all pins
            var mostNorth = double.Parse(results[0].Latitude, CultureInfo.InvariantCulture);
            var mostSouth = double.Parse(results[0].Latitude, CultureInfo.InvariantCulture);
            var mostEast = double.Parse(results[0].Longitude, CultureInfo.InvariantCulture);
            var mostWest = double.Parse(results[0].Longitude, CultureInfo.InvariantCulture);

            for (var i = 1; i < results.Count; i++) {
                var lat = double.Parse(results[i].Latitude, CultureInfo.InvariantCulture);
                var lng = double.Parse(results[i].Longitude, CultureInfo.InvariantCulture);

                if (lat > mostNorth) {
                    mostNorth = lat;
                }
                else if (lat < mostSouth) {
                    mostSouth = lat;
                }
                if (lng > mostEast) {
                    mostEast = lng;
                }
                else if (lat < mostWest) {
                    mostWest = lng;
                }
            }
            var bounds = new LocationRect(new Location(mostNorth + 0.015, mostWest + 0.0075),
new Location(mostSouth - 0.002, mostEast - 0.0075));

            // Set start and end locations
            var startLocation = new Location(double.Parse(results[0].Latitude,
CultureInfo.InvariantCulture), double.Parse(results[0].Longitude, CultureInfo.InvariantCulture));
            var endLocation = new Location(double.Parse(results[results.Count - 1].Latitude,
CultureInfo.InvariantCulture), double.Parse(results[results.Count - 1].Longitude,
CultureInfo.InvariantCulture));

            // Create MapLayer of driven route
            var polyLineLayer = new MapLayer(); // Layer used only for MapPolyLines for easier cleaning
            for (var i = 1; i < results.Count; i++) {
                var polyLine = new MapPolyline();
                var colourBrush = new SolidColorBrush {Color = Color.FromRgb(232, 123, 45)};
                polyLine.Stroke = colourBrush;
                polyLine.StrokeThickness = 2;
                polyLine.Opacity = 1.0;

                polyLine.Locations = new LocationCollection {
                    new Location(double.Parse(results[i - 1].Latitude, CultureInfo.InvariantCulture),
double.Parse(results[i - 1].Longitude, CultureInfo.InvariantCulture)),
                    new Location(double.Parse(results[i].Latitude, CultureInfo.InvariantCulture),
double.Parse(results[i].Longitude, CultureInfo.InvariantCulture))
                };
                polyLineLayer.Children.Add(polyLine); // Adds a new line to the layer
            }

            // Format date
            date = date.Replace('-', '/');
            date = date.Replace(" ", " - ");

            // Calculate distance
            var distance = 0.0;
            for (var i = 1; i < results.Count; i++) {
```

```
                var oldIndex = new Location(double.Parse(results[i - 1].Latitude,
CultureInfo.InvariantCulture), double.Parse(results[i - 1].Longitude, CultureInfo.InvariantCulture));
                var newIndex = new Location(double.Parse(results[i].Latitude,
CultureInfo.InvariantCulture), double.Parse(results[i].Longitude, CultureInfo.InvariantCulture));

                distance = distance + CalculateDistance(oldIndex, newIndex);
            }
            distance = distance / 1000; // Convert from meters to kilometers

            // Calculate duration
            // MAYBE Change to get difference between first and last index of 'results' collection
            string duration;
            var difference = dateEnd - dateStart;
            if (difference.TotalDays >= 1) {
                duration = difference.Days + "d " + difference.Hours + "h " + difference.Minutes + "m";
            }
            else if (difference.TotalHours >= 1) {
                duration = difference.Hours + "h " + difference.Minutes + "m";
            }
            else {
                duration = difference.Minutes + "m";
            }

            // Create TripTile
            TripTileCollection.Add(new Tile(rowId, startLocation, endLocation, bounds, polyLineLayer,
title, date, distance, duration));
        }
        catch (Exception) {
            //ignore
        }
    }
}
```

1.1.5 – RouteViewerLoadTrip

```
private List<RootObject> RouteViewerLoadTrip(int tripId) {
    var connection = new MySqlConnection(ConnectionString);
    var dt = new DataTable();
    connection.Open();

    try {
        var command = connection.CreateCommand();
        command.CommandText = $"SELECT * FROM trips WHERE id={tripId}";

        using (var dr = command.ExecuteReader()) {
            dt.Load(dr);
            foreach (DataRow row in dt.Rows) {
                var logs = row["log_file"].ToString();
                return JsonConvert.DeserializeObject<List<RootObject>>(logs);
            }
        }
    }
    catch (MySqlException exception) {
        MessageBox.Show(exception.ToString());
    }

    return null;
}
```

1.1.6 – CalculateDistance

```
// Returns the distance (in metric meters) between two locations
```

```
public static double CalculateDistance(Location previousPin, Location currentPin) {
    try {
        var firstCoordinate = new GeoCoordinate(previousPin.Longitude, previousPin.Latitude);
        var secondCoordinate = new GeoCoordinate(currentPin.Longitude, currentPin.Latitude);

        return firstCoordinate.GetDistanceTo(secondCoordinate);
    }
    catch (Exception) {
        return 0;
    }
}
```

1.1.7 – SearchBtn_OnClick

```
private void SearchBtn_OnClick(object sender, RoutedEventArgs e) {
    if (SearchDetailsBox.Text != string.Empty) {
        LoadTripTiles(SearchDetailsBox.Text);
    }
    else {
        LoadTripTiles();
    }

    LastSearch = SearchDetailsBox.Text;
}
```

1.1.8 - TripItemsControl_OnMouseLeftButtonUp

```
// TODO Generate pins with custom orange ellipse style
// TODO Click event for pins that display all details of that pin on the right hand side
private void TripItemsControl_OnMouseLeftButtonUp(object sender, MouseButtonEventArgs e) {
    // Visibility control
    TripTileGrid.Visibility = Visibility.Collapsed;
    RouteViewerGrid.Visibility = Visibility.Visible;
    RavenMainWindow.WindowState = WindowState.Maximized;

    // Load info
    var clickedItem = (FrameworkElement) e.OriginalSource;
    var item = (Tile) clickedItem.DataContext;
    var locations = RouteViewerLoadTrip(item.RowId);

    // Create Start/Stop pins
    var startLocation = new Pushpin {Background = Brushes.Green, Location = item.StartLocation,
Cursor = Cursors.Hand};
    var endLocation = new Pushpin {Background = Brushes.Red, Location = item.EndLocation,
Cursor = Cursors.Hand};

    // Create MapLayer of 'locations'
    var polyLineLayer = new MapLayer(); // Layer used only for MapPolyLines for easier cleaning
    for (var i = 1; i < locations.Count; i++) {
        var polyLine = new MapPolyline();
        var colourBrush = new SolidColorBrush {Color = Color.FromRgb(232, 123, 45)};
        polyLine.Stroke = colourBrush;
        polyLine.StrokeThickness = 3;
        polyLine.Opacity = 1.0;

        polyLine.Locations = new LocationCollection {
            new Location(double.Parse(locations[i - 1].Latitude, CultureInfo.InvariantCulture),
double.Parse(locations[i - 1].Longitude, CultureInfo.InvariantCulture)),
            new Location(double.Parse(locations[i].Latitude, CultureInfo.InvariantCulture),
double.Parse(locations[i].Longitude, CultureInfo.InvariantCulture))
        };
        polyLineLayer.Children.Add(polyLine); // Adds a new line to the layer
```

```
        }

        // Create new bounds
        var bounds = new LocationRect(new Location(item.Bounds.Center.Latitude - 0.0065,
item.Bounds.Center.Longitude), item.Bounds.Width + 0.05, item.Bounds.Height + 0.05);

        // Clear TripTileCollection for smoother RouteViewerMap
        TripTileCollection.Clear();

        // Setup Map
        RouteViewerMap.SetView(bounds);
        RouteViewerMap.Children.Add(polyLineLayer);

        RouteViewerMap.Children.Add(startLocation);
        RouteViewerMap.Children.Add(endLocation);
    }
```

1.1.9 - RavenMainWindow_OnMouseRightButtonUp

```
    private void RavenMainWindow_OnMouseRightButtonUp(object sender, MouseButtonEventArgs e)
{
        // Visibility control
        TripTileGrid.Visibility = Visibility.Visible;
        RouteViewerGrid.Visibility = Visibility.Collapsed;
        RavenMainWindow.WindowState = WindowState.Normal;

        // Cleanup Map
        RouteViewerMap.Children.Clear();

        // Reload TripTiles from previous search
        LoadTripTiles(LastSearch);
    }
```

1.2.1 - RootObject

```
    public class RootObject {
        [JsonProperty(PropertyName = "DeltaTime")]
        public string DeltaTime { get; set; }

        [JsonProperty(PropertyName = "Rpm")]
        public string Rpm { get; set; }

        [JsonProperty(PropertyName = "Speed")]
        public string Speed { get; set; }

        [JsonProperty(PropertyName = "Lat")]
        public string Latitude { get; set; }

        [JsonProperty(PropertyName = "Lng")]
        public string Longitude { get; set; }
    }

  }
}
```

2.1.0 - LoginWindow.xaml.cs

```
using System;
using System.Data;
using System.Linq;
using System.Runtime.InteropServices;
using System.Security.Cryptography;
using System.Text;
using System.Windows;
using System.Windows.Input;
using MySql.Data.MySqlClient;

namespace Raven.Windows {
    /// <summary>
    /// Interaction logic for LoginWindow.xaml
    /// </summary>
    public partial class LoginWindow {
        private const string ConnectionString = "Server=raven-
gps.com;Database=raven;Uid=root;Pwd=####";
        public bool LoginSuccess;
```

2.1.1 - LoginWindow

```
        public LoginWindow() {
            InitializeComponent();
        }
```

2.1.2 – LoginBtn_Click

```
        private void LoginBtn_Click(object sender, RoutedEventArgs e) {
            AttemptLogin();
        }
```

2.1.3 - AttemptLogin

```
        private void AttemptLogin() {
            // If login is corrrect changes LoginSuccess to true and closes LoginWindow
            if (AuthenticateLogin(UsernameText.Text.ToLower(), PasswordText.Password)) {
                LoginSuccess = true;
                MainWindow.Username = UsernameText.Text.ToLower();
                Close();
            }
            // If username or password is incorrect show error
            else {
                MessageBox.Show("Username or password is incorrect");
            }
        }
```

2.1.4 - AuthenticateLogin

```
        // Returns true or false depending on a SQL check for a login that matches username and password
        public bool AuthenticateLogin(string username, string password) {
            // Stores hash value of password in hash
            var hash = GetHash(password);

            var connection = new MySqlConnection(ConnectionString);
            var dt = new DataTable();
            connection.Open();

            try {
                var command = connection.CreateCommand();
                command.CommandText = "SELECT * FROM logins";
```

```
            using (var dr = command.ExecuteReader()) {
                dt.Load(dr);

                // Returns true using LINQ expression
                if (dt.Rows.Cast<DataRow>().Where(variable => variable.Field<string>("username")
== username).Any(variable => variable.Field<string>("password") == hash)) {
                    return true;
                }
            }
        }
        catch (MySqlException exception) {
            MessageBox.Show(exception.ToString());
            return false;
        }
        return false;
    }
```

2.1.5 - GetHash

```
    // Returns plain password converted to SHA512 encrypted as string
    public static string GetHash(string inputString) {
        var sha512 = SHA512.Create();
        var bytes = Encoding.UTF8.GetBytes(inputString);
        var hash = sha512.ComputeHash(bytes);
        return GetStringFromHash(hash);
    }
```

2.1.6 - GetStringFromHash

```
    // Returns a hash string built from a byte array
    private static string GetStringFromHash(byte[] hash) {
        var result = new StringBuilder();
        foreach (var t in hash) {
            result.Append(t.ToString("X2"));
        }
        return result.ToString();
    }
```

2.1.7 - UsernameText_LostKeyboardFocus

```
    // Empties UsernameText if there are no characters when losing focus
    private void UsernameText_OnLostKeyboardFocus(object sender,
KeyboardFocusChangedEventArgs e) {
        if (string.IsNullOrEmpty(UsernameText.Text)) {
            UsernameText.Text = null;
        }
    }
```

2.1.8 - PasswordText_OnGotKeyboardFocus

```
    // Empties PasswordText if it contains dummy characters when getting focus
    private void PasswordText_OnGotKeyboardFocus(object sender,
KeyboardFocusChangedEventArgs e) {
        if (PasswordText.Password == "PPPPPPPP") {
            PasswordText.Password = null;
        }
    }
```

2.1.9 - PasswordText_LostGotKeyboardFocus

```
    // Fills PasswordText with dummy characters when losing focus
    private void PasswordText_LostGotKeyboardFocus(object sender,
KeyboardFocusChangedEventArgs e) {
```

```
            if (PasswordText.Password == "") {
                PasswordText.Password = "PPPPPPPP";
            }
        }
```

2.1.10 - LoginWindow_OnKeyDown

```
        // Key event that registers Enter or RShift + Enter
        private void LoginWindow_OnKeyDown(object sender, KeyEventArgs e) {
            switch (e.Key) {
                case Key.Enter:
                    if (Keyboard.IsKeyDown(Key.RightShift)) {
                        LoginSuccess = true;
                        MainWindow.Username = "DebugUser";
                        Close();
                    }
                    else {
                        AttemptLogin();
                    }
                    break;
            }
        }
    }
}
```

3.1.0 - TileMap.xaml.cs

```
using System.Windows;
using System.Windows.Media;
using Microsoft.Maps.MapControl.WPF;

namespace Raven.Controls {
    /// <summary>
    /// Interaction logic for TileMap.xaml
    /// </summary>
    public partial class TileMap {
```

3.1.1 - TileMap

```
        public TileMap() {
            InitializeComponent();
        }
```

3.1.2 – TileMap_OnLoaded

```
        private void TileMap_OnLoaded(object sender, RoutedEventArgs e) {
            var startLocation = new Pushpin { Background = Brushes.Green, Location =
((Tile)DataContext).StartLocation };
            var endLocation = new Pushpin { Background = Brushes.Red, Location =
((Tile)DataContext).EndLocation };

            Children.Add(((Tile)DataContext).Route);
            Children.Add(startLocation);
            Children.Add(endLocation);
            SetView(((Tile)DataContext).Bounds);
        }
    }
}
```

4.1.0 – Tile.cs

```
using Microsoft.Maps.MapControl.WPF;
```

```
namespace Raven {
    public class Tile {
        public int RowId { get; set; }
        public Location StartLocation { get; set; }
        public Location EndLocation { get; set; }
        public LocationRect Bounds { get; set; }
        public MapLayer Route { get; set; }
        public string Title { get; set; }
        public string Date { get; set; }
        public double Distance { get; set; }
        public string Duration { get; set; }
```

4.1.1 - Tile

```
        public Tile(int rowId, Location startLocation, Location endLocation, LocationRect
bounds, MapLayer route, string title, string date, double distance, string duration) {
            RowId = rowId;
            StartLocation = startLocation;
            EndLocation = endLocation;
            Bounds = bounds;
            Route = route;
            Title = title;
            Date = date;
            Distance = distance;
            Duration = duration;
        }
    }
}
```

Android

Commented [N4]: Mangler Android kode, husk kommentarer i kilde kode

Arduino

Commented [N5]: Mangler Arduino kode, husk kommentarer i kilde kode

SQL Script

```
USE raven;

DROP TABLE if exists raven.trips;
DROP TABLE if exists raven.logins;

CREATE TABLE IF NOT EXISTS `raven`.`logins` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `reg` VARCHAR(10) NOT NULL,
  `username` VARCHAR(32) NOT NULL,
  `password` mediumtext NOT NULL,
  PRIMARY KEY (`id`, `username`),
  UNIQUE INDEX `username_UNIQUE` (`username` ASC))
COMMENT = 'Login table for Raven-GPS authentication';

INSERT INTO logins (`reg`,`username`,`password`) VALUES (
   "BK79499",
   "rwejlgaard",
"B109F3BBBC244EB82441917ED06D618B9008DD09B3BEFD1B5E07394C706A8BB980B1D7785E5976EC049B46DF5F1326AF
5A2EA6D103FD07C95385FFAB0CACBC86"
);

INSERT INTO logins (`reg`,`username`,`password`) VALUES (
```

```
  "BE70846",
  "nwmicheelsen",
"B109F3BBBC244EB82441917ED06D618B9008DD09B3BEFD1B5E07394C706A8BB980B1D7785E5976EC049B46DF5F1326AF
5A2EA6D103FD07C95385FFAB0CACBC86"
);

CREATE TABLE `raven`.`trips` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `time_started` DATETIME NOT NULL,
  `time_ended` DATETIME NOT NULL,
  `driver_username` VARCHAR(32) NOT NULL,
  `driver_reg` VARCHAR(20) NOT NULL,
  `log_file` LONGTEXT NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC),
  INDEX `reg_idx` (`driver_reg` ASC),
  INDEX `driver_username_idx` (`driver_username` ASC),
  CONSTRAINT `driver_username_FK`
    FOREIGN KEY (`driver_username`)
    REFERENCES `raven`.`logins` (`username`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
COMMENT = 'Trip table for Raven-GPS';
```