Brief application description

The code described in this document refers to a cross-platform app (in this case it only can execute on an android device because I have no way to develop iOS code) which ensures the geolocation services on device are enabled and then:

1. it retrieves the current position
2. it requests for the current weather by using OpenWeatherMap online service

The MVVM solution contains several main files useful for understanding the code. They are the following:

App.xaml.cs

It contains the application entry point and sets the first page.

```
using Xamarin.Forms;

namespace MVVM
{
    public partial class App : Application
    {
        public App()
        {
            InitializeComponent();

            MainPage = new FirstPage();
        }

        protected override void OnStart()
        {
            // Handle when your app starts
        }

        protected override void OnSleep()
        {
            // Handle when your app sleeps
        }

        protected override void OnResume()
        {
            // Handle when your app resumes
        }
    }
}
```

FirstPage.xaml

It contains the UI definition for the main page. It consists of a labels, buttons and an activity indicator (visible when an operation is in progress). The labels containing the current address and weather are directly binded with relative properties set in code behind.

```
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml" xmlns:local="clr-
namespace:MVVM" x:Class="MVVM.FirstPage" BackgroundImage="background.png">
    <StackLayout Padding="20">
        <Label x:Name="lblDescription" Text="This is a meteo utility.&#x0a;Ensure you have
           the geolocation feature enabled on your phone and click the button."
          Font="Large" HorizontalOptions="Center" VerticalOptions="CenterAndExpand"
          HorizontalTextAlignment="Center" TextColor="White"/>
        <Button x:Name="btnExecute" Text="Retrieve current location"
          HorizontalOptions="Center" VerticalOptions="CenterAndExpand"
          Clicked="OnLocationButtonClicked" BackgroundColor="Navy" TextColor="White"/>
        <ActivityIndicator x:Name="ai" HeightRequest="30" WidthRequest="30"
          HorizontalOptions="Center" VerticalOptions="Center" IsVisible="true"
          Color="White"/>
        <Label x:Name="lblAddress" Text="{Binding Path=Address,
          StringFormat='Current address is:{0}'}" Font="Large" HorizontalOptions="Start"
          VerticalOptions="CenterAndExpand" TextColor="White"/>
        <Label x:Name="lblWeather" Text="{Binding Path=WeatherDescr,
          StringFormat='Current weather is:{0}'}" Font="Large" HorizontalOptions="Start"
          VerticalOptions="CenterAndExpand" TextColor="White"/>
        <Button x:Name="btnReset" Text="Reset" HorizontalOptions="Center"
          VerticalOptions="CenterAndExpand" Clicked="OnResetButtonClicked"
          BackgroundColor="Navy" TextColor="White"/>
    </StackLayout>
    <ContentPage.BindingContext>
        <local:FirstPageViewModel/>
    </ContentPage.BindingContext>
</ContentPage>
```

FirstPage.xaml.cs

It's the code behind of the first page and it'is in charge of setting the binding context and manages the buttons' click by calling the logic written in FirstPageViewModel.cs

```
using System;
using Xamarin.Forms;

namespace MVVM
{
    public partial class FirstPage : ContentPage
    {
        FirstPageViewModel vm;
        public FirstPage()
        {
            InitializeComponent();
            Init();
        }

        void Init() {
             vm = new FirstPageViewModel();
            //Set bindings
            lblAddress.BindingContext = vm;
            lblAddress.SetBinding(Label.TextProperty, "Address", stringFormat:"Current addres
s: {0}");

            lblWeather.BindingContext = vm;
            lblWeather.SetBinding(Label.TextProperty, "WeatherDescr", stringFormat: "Current
weather: {0}");

            ai.BindingContext = vm;
            ai.SetBinding(ActivityIndicator.IsRunningProperty, "IsOperationInProgress");

            btnExecute.BindingContext = vm;
            btnExecute.SetBinding(Button.IsEnabledProperty, "UIEnabled");

            btnReset.BindingContext = vm;
            btnReset.SetBinding(Button.IsEnabledProperty, "UIEnabled");
        }


        async void OnLocationButtonClicked(object sender, EventArgs args)
        {
            vm.IsOperationInProgress = true;
            var result = await vm.UpdatePosition();
            if ((result!=null)&&(result.IsValid))
            {
                //retrieve weather
                try
                {
var weather = await vm.UpdateWeather(result.Lat, result.Lon);
                if (weather == null)
                    await DisplayAlert("Error", "An error occurred while retrieving weather i
nformation", "Ok");
                }
                catch (Exception ex)
                {
                    await DisplayAlert("Exception", string.Format("{0} --
- {1}",ex.Message,ex.InnerException.Message), "Ok");
                }

            }
            vm.IsOperationInProgress = false;
        }

        async void OnResetButtonClicked(object sender, EventArgs args)
        {
            vm.WeatherDescr = string.Empty;
            vm.Address = string.Empty;
        }
    }
}
```

FirstPageViewModel.cs

Here the main logic of the entire application is coded.

This class contains the properties binded to the UI, it implements the geolocation logic (by calling an external object which is CrossGeolocator) and the HTTP client which retrieves the weather information from OpenWeatherMap service.

```
using System;
using System.ComponentModel;
using Xamarin.Forms;
using Plugin.Geolocator;
using Plugin.Geolocator.Abstractions;
using System.Linq;
using System.Threading.Tasks;

namespace MVVM
{
    public class FirstPageViewModel : INotifyPropertyChanged
    {
        string _className = "FirstPageViewModel";
        public event PropertyChangedEventHandler PropertyChanged;

        CustomPosition _position = new CustomPosition();

        private string _address = "";
        public string Address {
            get { return _address;}
            set {
                if (_address == value)
                    return;
                _address = value;
                OnPropertyChanged("Address");
            }
        }

        private string _weatherdescr = "";
        public string WeatherDescr {
            get { return _weatherdescr;}
            set {
                if (_weatherdescr.Equals(value))
                    return;
                _weatherdescr = value;
                OnPropertyChanged("WeatherDescr");
            }
        }


        private int _count = 0;
        public int Count {
            get { return _count;}
            set {
                if (_count == value)
                    return;
                _count = value;
                OnPropertyChanged("Count");
            }
        }

        public bool UIEnabled {
            get { return !IsOperationInProgress;}
        }

        private bool _isOperationInProgress = false;
        public bool IsOperationInProgress {
            get { return _isOperationInProgress;}
            set {
                _isOperationInProgress = value;
                OnPropertyChanged("IsOperationInProgress");
                OnPropertyChanged("UIEnabled");
            }
        }
```

```
        public FirstPageViewModel()
        {
        }

        public async Task<CustomPosition> UpdatePosition()
        {
            try
            {
                //IsOperationInProgress = true;
                if (!Plugin.Geolocator.CrossGeolocator.Current.IsGeolocationAvailable)
                {
                    await Application.Current.MainPage.DisplayAlert("Geolocation is unavailable
!", "It seems that geolocation is not available on this device!", "Close");
                    //IsOperationInProgress = false;
                    return null;
                }
                if (!Plugin.Geolocator.CrossGeolocator.Current.IsGeolocationEnabled)
                {
                    await Application.Current.MainPage.DisplayAlert("Geolocation is unavailable
!", "You must enable phone geolocation first!", "Close");
                    //IsOperationInProgress = false;
                    return null;
                }
                Position pos = await CrossGeolocator.Current.GetPositionAsync(timeout: new Time
Span(0, 0, 25));
                if (PositionIsValid(pos))
                {
                    var addrs = await CrossGeolocator.Current.GetAddressesForPositionAsync(pos)
;
                    if (addrs.Count() > 0)
                    {
                        _position.Alt = pos.Altitude;
                        _position.Lon = pos.Longitude;
                        _position.Lat = pos.Latitude;
                        _position.Street = addrs.First().Thoroughfare;
                        _position.Number = addrs.First().SubThoroughfare;
                        _position.City = addrs.First().Locality;
                        _position.Country = addrs.First().CountryName;
                        _position.IsValid = true;
                        Address = String.Format("{0} {1}, {2} {3}", _position.Street, _position
.Number, _position.City, _position.Country);
                    }
                }
                else
                {
                    //cant 'retrieve position
                    _position.IsValid = false;
                }
                return _position;
            }
            catch (Exception ex)
            {
                var dudu = ex;
                return null;
            }
            finally
            {
                //IsOperationInProgress = false;
            }
        }

        private bool PositionIsValid(Position p_pos)
        {
            if (p_pos.Timestamp.Year==DateTime.Now.Year)
                return true;
            return false;
        }
```

```
        public async Task<RESTClient.WeatherResult> UpdateWeather(double lat, double lon)
        {
            RESTClient.WeatherResult weatherResult = null;
            try
            {
                RESTClient.Client client = new RESTClient.Client();
                weatherResult = await client.GetCurrentWeather(lat, lon);
                WeatherDescr = string.Format("{0} ({1})", weatherResult.Main, weatherResult.Des
cription);
            }
            catch (Exception ex)
            {
                System.Diagnostics.Debug.WriteLine("Exception occurred in UpdateWeather method!
");
                throw (ex);
                return null;
            }
            return weatherResult;
        }

        protected virtual void OnPropertyChanged(string propertyName)
        {
            if (PropertyChanged != null)
            {
                PropertyChanged(this,
                   new PropertyChangedEventArgs(propertyName));
            }
        }
    }
}
```

Client.cs

This class uses OpenWeatherMap.org APIs in order to retrieve weather information from a position described by latitude and longitude coordinates. After that, it parses the result in order to return back only some weather descriptions.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Http;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;

namespace RESTClient
{
    public class Client
    {
        //weather example: http://api.openweathermap.org/data/2.5/weather?lat=45.7982741&lon=9.0
954442&APPID=f9c37aafc23e975a2625e2119a9219d9
        //const string URL_weather = "http://api.openweathermap.org/data/2.5/";
        const string URL_weather = "http://146.185.181.89/data/2.5/";
        const string API_KEY_weather = "f9c37aafc23e975a2625e2119a9219d9";

        HttpClient _httpClient;

        public Client()
        {
            _httpClient = new HttpClient();
        }

        public async Task<WeatherResult> GetCurrentWeather(double latitude, double longitude)
        {
            try
            {
                WeatherResult weather = new WeatherResult();
                string lat = latitude.ToString().Replace(",", ".");
                string lon = longitude.ToString().Replace(",", ".");
                string completeUrl = string.Format("{0}weather?lat={1}&lon={2}&APPID={3}", URL_w
eather, lat, lon, API_KEY_weather);
                _httpClient.Timeout = new TimeSpan(0, 0, 30);
                string jsonResult = await _httpClient.GetStringAsync(completeUrl);
                var jsonParsed = Newtonsoft.Json.Linq.JObject.Parse(jsonResult);
                weather.Main = jsonParsed.SelectToken("weather[0].main").ToString();
                weather.Description = jsonParsed.SelectToken("weather[0].description").ToString(
);
                return weather;
            }
            catch (Exception ex)
            {
                throw (ex);
            }

        }
    }
}
```