Macoratti.net Xamarin Forms - Previsão do Tempo (OpenWeatherMap)

Neste artigo vou mostrar como acessar um serviço REST e obter informações sobre a previsão do tempo por cidades em uma aplicação Xamarin Forms usando o Visual Studio 2015 e a linguagem C#.

Toda a teoria sobre como acessar um serviço REST pode eu já tratei nestes artigos:

- Xamarin Android - Acessando um serviço REST - Macoratti
- .NET - Apresentando a classe HttpClient - I - Macoratti
- .NET - HttpWebRequest x WebClient x HttpClient - Macoratti
- C# - Fazendo requisições na web - Macoratti

Hoje vou focar na utilização prática de como acessar um serviço web e obter resultados em uma aplicação Windows Forms usando a linguagem VB .NET.

Como exemplo eu vou acessar o serviço da OpenWeatherMap API service que permite obter a previsão de tempo *(e mapas)* para uma determinada localidade e exibir os dados retornados.

Para pode usar o serviço grátis você tem que se registrar e obter uma **API Key (AppId)** neste link: https://home.openweathermap.org/users/sign_in

A documentação de como usar a API para previsão do tempo pode ser consultada neste link: https://openweathermap.org/current#current_JSON

Basicamente, após você obter a sua **API Key** você deverá montar a URL da requisição no seguinte formato: **api.openweathermap.org/data/2.5/weather?q={nome_cidade}&appid={sua_api_key}**

Existem muitas variações e formas de enviar uma requisição para obter informações de tempo e mapas. Para detalhes consulte a documentação da API.

Isto posto, vamos o que interessa...

Recursos usados:

- **Visual Studio Community 2015 ou Xamarin Studio**
- **Xamarin**

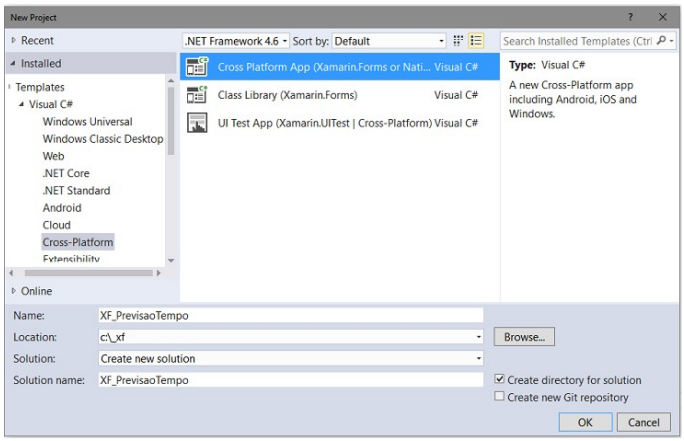
Nota: Baixe e use a versão Community 2015 do VS ela é grátis e é equivalente a versão Professional.

Criando o projeto no Visual Studio 2015 Community

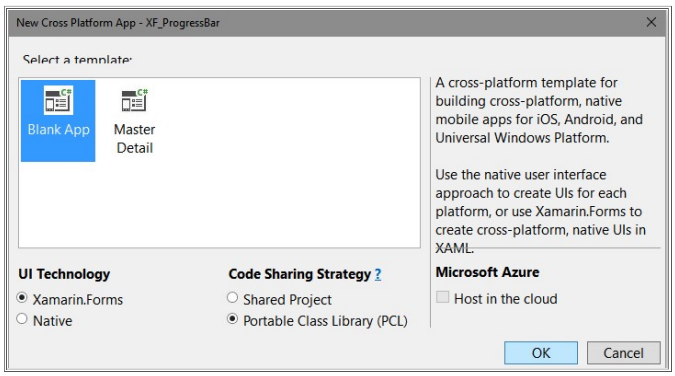
Abra o Visual Studio Community 2015 e clique em **New Project;**

Selecione Visual C#, o template **Cross Plataform** e a seguir **Cross Plataform App(Xamarin.Forms or Native);**

Informe o nome **XF_PrevisaoTempo** e clique no botão OK;



A seguir selecione **Blank App** e marque as opções - **Xamarin.Forms** e **Portable Class Library (PCL**) e clique em OK;

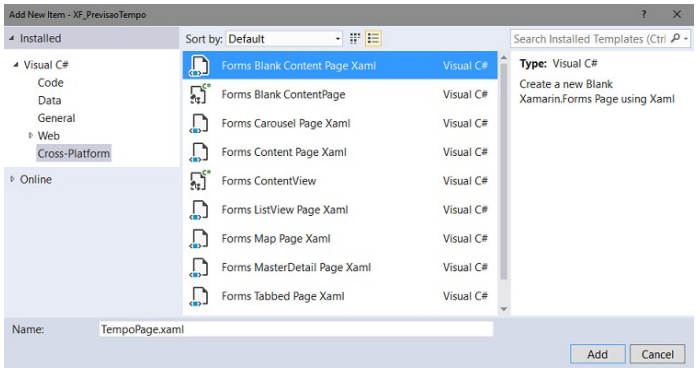


Será criado um projeto contendo no projeto **Portable** as páginas **App.xaml e MainPage.xaml.**

No code-behind do arquivo **App.xaml** temos a classe **App.cs** que irá conter o código compartilhado e que vamos usar neste artigo. Ao final teremos as referências incluídas em todos os projetos da nossa solução.

Iremos usar a página **TempoPage.xaml** como página principal da nossa aplicação. Para isso crie uma pasta chamada **Views** no seu projeto *(Project -> New Folder).*

A seguir crie pagina TempoPage.xaml nesta pasta na opção **Project -> Add New Item** e a seguir a opção abaixo:



Após criar a pagina principal vamos referenciá-la em nosso arquivo **App.cs** defininod o código que abre esta página:

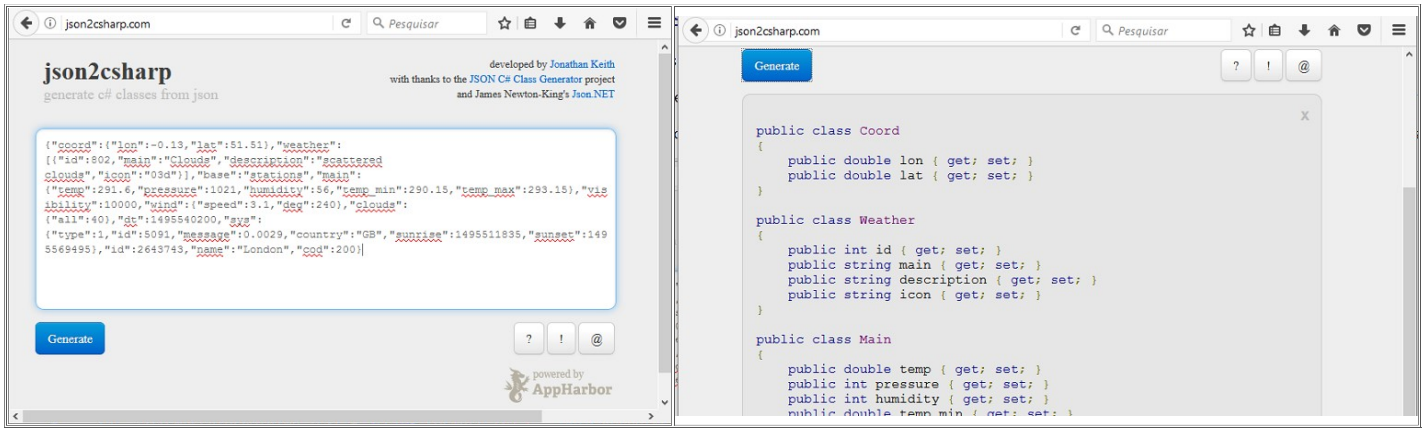
```
using Xamarin.Forms;

namespace XF_Bindable_Spinner
{
   public class App : Application
   {
      public App()
      {
         MainPage = new XF_PrevisaoTempo.Views.TempoPage();
      }

      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
      }
   }
}
```

Criando as classes de domínio para obter os dados desserializados

Precisamos criar as classes para obter os dados que serão obtidos do serviço REST e desserializados no formato JSON.

Essa tarefa fica bem simples se usarmos o recurso disponíbilizado http://json2csharp.com/. Aqui podemos gerar as classes C# a partir do JSON.

Então basta enviar uma requisição, obter o JSON retornar e a seguir copiar e colar este JSON no site indicado para gerar as classes C#.



Para simplificar vamos reunir as informações básicas que vamos exibir em uma classe chamada **Tempo.** Crie uma pasta **Models** no projeto e a seguir crie o arquivo **Tempo.cs** nesta pasta com o código abaixo:

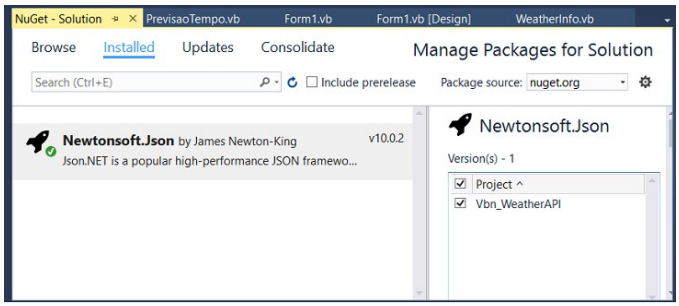
```
namespace XF_PrevisaoTempo.Model
{
   public class Tempo
   {
      public string Title { get; set; }
      public string Temperature { get; set; }
      public string Wind { get; set; }
      public string Humidity { get; set; }
      public string Visibility { get; set; }
      public string Sunrise { get; set; }
      public string Sunset { get; set; }
```

```
        public Tempo()
        {
            // Como as Labels se vinculam a estes valores vamos defini-los
            // como uma string vazia no construtor
            this.Title = " ";
            this.Temperature = " ";
            this.Wind = " ";
            this.Humidity = " ";
            this.Visibility = " ";
            this.Sunrise = " ";
            this.Sunset = " ";
        }
    }
}
```

Essa será a nossa classe de domínio que iremos usar para obter as informações e exibí-las via databinding na página **TempoPage.xaml.**

Criando a classe para acessar o serviço REST

Para poder definir o código da classe que acessa o serviço temos que incluir uma referência a API **Newtonsoft.Json** no projeto via Nuget:



Crie uma pasta chamada **Service** no projeto e a seguir crie o arquivo **DataService.cs** definindo o código abaixo:

```
using Newtonsoft.Json;
using System.Net.Http;
using System.Threading.Tasks;

namespace XF_PrevisaoTempo.Service
{
    public class DataService
    {

        public static async Task<Tempo> GetPrevisaoDoTempo(string cidade)
        {
            string appId = "Numero_da_sua_API_KEY";
            string queryString = "http://api.openweathermap.org/data/2.5/weather?q=" + cidade + "&units=metric" + "&appid=" + appId;

            dynamic resultado = await DataService.getDataFromService(queryString).ConfigureAwait(false);

            if (resultado["weather"] != null)
            {
                Tempo previsao = new Tempo();
                previsao.Title = (string)resultado["name"];
                previsao.Temperature = (string)resultado["main"]["temp"] + " C";
                previsao.Wind = (string)resultado["wind"]["speed"] + " mph";
                previsao.Humidity = (string)resultado["main"]["humidity"] + " %";
                previsao.Visibility = (string)resultado["weather"][0]["main"];

                DateTime time = new DateTime(1970, 1, 1, 0, 0, 0, 0);
                DateTime sunrise = time.AddSeconds((double)resultado["sys"]["sunrise"]);
                DateTime sunset = time.AddSeconds((double)resultado["sys"]["sunset"]);
                previsao.Sunrise = String.Format("{0:d/MM/yyyy HH:mm:ss}", sunrise);
                previsao.Sunset = String.Format("{0:d/MM/yyyy HH:mm:ss}", sunset);
                return previsao;
            }
            else
            {
                return null;
            }
        }

        public static async Task<dynamic> getDataFromService(string queryString)
        {
            HttpClient client = new HttpClient();
            var response = await client.GetAsync(queryString);

            dynamic data = null;
            if (response != null)
            {
                string json = response.Content.ReadAsStringAsync().Result;
                data = JsonConvert.DeserializeObject(json);
            }

            return data;
        }

        public static async Task<dynamic> getDataFromServiceByCity(string city)
        {
            string appId = "numero_da_sua_API_KEY";

            string url = string.Format("http://api.openweathermap.org/data/2.5/forecast/daily?q={0}&units=metric&cnt=1&APPID={1}", city.Trim(), appId);

            HttpClient client = new HttpClient();
```

```
            var response = await client.GetAsync(url);

            dynamic data = null;
            if (response != null)
            {
                string json = response.Content.ReadAsStringAsync().Result;
                data = JsonConvert.DeserializeObject(json);
            }

            return data;
        }
    }
}
```

Temos nesta classe os métodos :

- **GetPrevisaoDoTempo**(string cidade) - Retorna a previsão do tempo a partir do nome de uma cidade;
- **getDataFromServiceByCity**(string city) - retorna a previsão do tempo para o nome de uma cidade usando outro formato de requisição;
- **getDataFromService**(string **queryString**) - retorna as informações com base em uma string que representa a URL de requisição para o serviço;

No exemplo vamos usar o método **GetPrevisaoDoTempo**() passando o nome da cidade onde iremos montar a **querystring** e chamar o método **GetDataFromService**.

Definindo a interface com o usuário na página TempoPage.xaml

Abra o arquivo **TempoPage.xaml** e altere o seu código conforme mostrado a seguir:

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="XF_PrevisaoTempo.Views.TempoPage">

            <StackLayout>
                <StackLayout Orientation="Horizontal" HorizontalOptions="FillAndExpand" BackgroundColor="#545454">
                    <StackLayout Padding="10,10,10,10" HorizontalOptions="Start">
                        <Label Text="Procurar Cidade" TextColor="White" FontAttributes="Bold" FontSize="Medium" />
                        <Label x:Name="cidadeLabel" Text="Digite o nome da Cidade"  TextColor="Yellow" FontSize="Small"/>
                        <Entry x:Name="cidadeEntry"/>
                    </StackLayout>
                    <StackLayout Padding="0,0,0,10" VerticalOptions="End">
                        <Button x:Name="btnPrevisao" Text="Previsão do Tempo" WidthRequest="185"
                                BorderWidth="1" Clicked="btnPrevisao_Clicked"/>
                    </StackLayout>
                </StackLayout>
                <StackLayout Padding="10,10,10,10" HorizontalOptions="Start">
                    <Image x:Name="imgTempo" />
                    <Label Text="Local" />
                    <Label Text="{Binding Title}" TextColor="Black" />
                    <Label Text="Temperatura"  />
                    <Label x:Name="tempLabel" TextColor="Black"  Text="{Binding Temperature}"  />
                    <Label Text="Veloc. Vento"  />
                    <Label x:Name="windLabel" Text="{Binding Wind}" TextColor="Black"  />
                    <Label Text="Humidade" />
                    <Label x:Name="humidityLabel" Text="{Binding Humidity}" TextColor="Black"/>
                    <Label Text="Visibilidade" />
                    <Label x:Name="visibilitylabel" Text="{Binding Visibility}" TextColor="Black"/>
                    <Label Text="Hora Nascimento Sol"  />
                    <Label x:Name="sunriseLabel" Text="{Binding Sunrise}" TextColor="Black" />
                    <Label Text="Hora do pôr do Sol" />
                    <Label x:Name="sunsetLabel" Text="{Binding Sunset}" TextColor="Black"/>
                </StackLayout>
            </StackLayout>

</ContentPage>
```

Neste código estamos definindo uma view Entry onde o usuário vai informar o nome da cidade para compor a URL a partir da qual iremos obter a previsão do tempo.

A seguir temos diversas Labels onde iremos exibir o resultado via databinding.

Definindo o código do evento Click do botão de comando e do construtor da página

Para poder animar essa progressbar usamos no arquivo code-behind **TempoPage.xaml.cs** o código abaixo:

```
using System;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
using XF_PrevisaoTempo.Logic;
using XF_PrevisaoTempo.Model;

namespace XF_PrevisaoTempo.Views
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class TempoPage : ContentPage
    {
        public TempoPage()
        {
            InitializeComponent();
            this.Title = "Previsão do Tempo";
            //define o databinding para os objetos padrão iniciais
            this.BindingContext = new Tempo();
        }

        private async void btnPrevisao_Clicked(object sender, EventArgs e)
        {
            try
            {
                if (!String.IsNullOrEmpty(cidadeEntry.Text))
                {
                    Tempo previsaoDoTempo = await DataService.GetPrevisaoDoTempo(cidadeEntry.Text);
                    this.BindingContext = previsaoDoTempo ;
```

```
                btnPrevisao.Text = "Nova Previsão";
            }
        }
        catch (Exception ex)
        {
            await DisplayAlert("Erro ", ex.Message, "OK");
        }
    }
  }
}
```

Vamos entender o código :

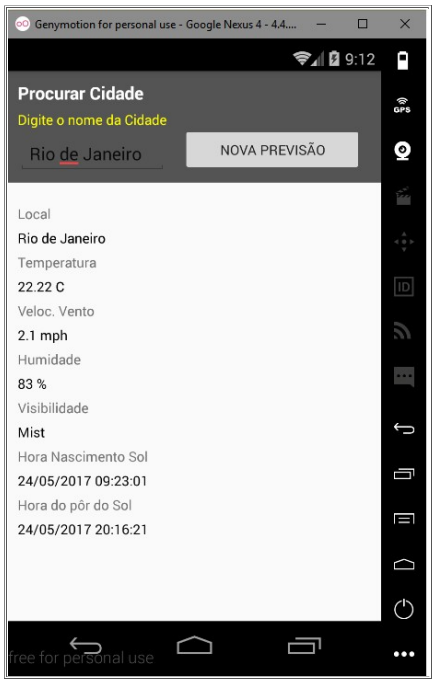
1- No evento **Click** do botão de comando estamos chamando o método **GetPrevisaoDoTempo**() passando o nome da cidade informada.

```
Tempo previsaoDoTempo = await DataService.GetPrevisaoDoTempo(cidadeEntry.Text);
```

2- Obtemos as informações e preenchemo o objeto **previsaoDoTempo** que é do tipo da nossa classe **Tempo** e fazemos a vinculação ao **BindingContext** para poder exibir os dados na página:

```
this.BindingContext = previsaoDoTempo ;
```

Após isso , excutando o projeto e informando o nome de uma ciade iremos obter o seguinte resultado:



Na figura vemos a execução no emulador **Genymotion** para Android.

Pegue o projeto aqui : **XF_PrevisaoTempo.zip** (sem as referências)

"Porque a lei foi dada por Moisés; a graça e a verdade vieram por Jesus Cristo." João 1:17

Referências:

- Seção VB .NET do Site Macoratti.net
- Super DVD .NET - A sua porta de entrada na plataforma .NET

- Super DVD Vídeo Aulas - Vídeo Aula sobre VB .NET, ASP .NET e C#
- Super DVD C# - Recursos de aprendizagens e vídeo aulas para C#
- Seção C# do site Macoratti.net
- Seção ASP .NET do site Macoratti .net
- Curso Básico VB .NET - Vídeo Aulas
- Curso C# Básico - Vídeo Aulas
- Curso Fundamentos da Programação Orientada a Objetos com VB .NET NEW
- **Macoratti .net | Facebook**
- **macoratti - YouTube**
- **Jose C Macoratti (@macorati) | Twitter**
- VB.NET 2005 - Controles - Macoratti.net
- Seção de Jogos do site Macoratti .net
- Xamarim - Desenvolvimento Multiplataforma com C# ... - Macoratti.net
- Xamarin - Apresentando Xamarin.Forms - Macoratti.net
- Xamarin.Forms - Olá Mundo - Criando sua primeira ... - Macoratti.net
- Xamarin.Forms - Olá Mundo - Anatomia da aplicação - Macoratti.net
- https://developer.xamarin.com/api/type/Android.App.AlertDialog/
- Xamarin Android - Tratando eventos de forma declarativa
- Seção Mobile/Xamarin do site Macoratti .net
- https://developer.android.com/reference/android/app/Activity.html
- https://developer.xamarin.com/api/type/Android.Widget.ProgressBar/

José Carlos Macoratti