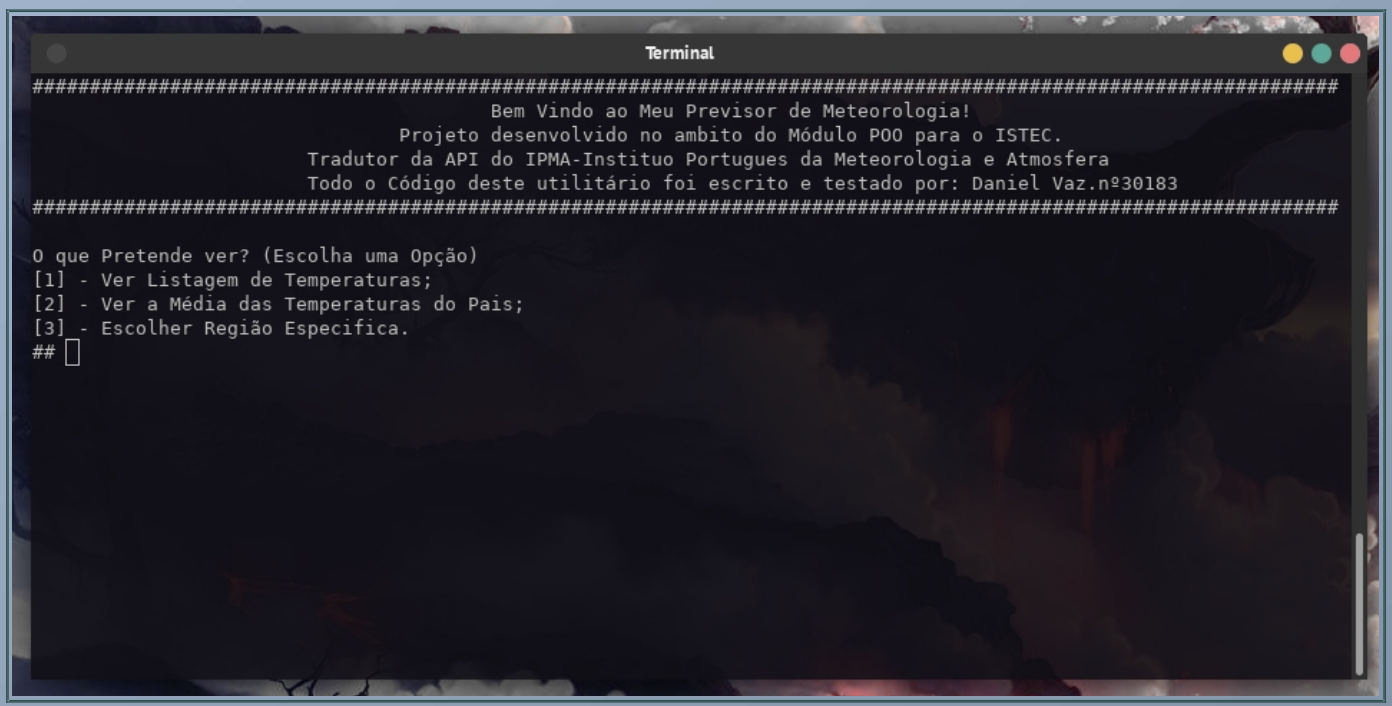
Relatório

Prognósticos Meteorológicos através da API do IPMA.

Escrito em C#



Trabalho realizado por **Daniel Vaz**
nº30183-2A-RSI

Introdução

Este trabalho surge a pedido do Professor Sandro Ferreira no módulo de **P**rogramação **O**rientada a **O**bjetos, do *Instituto Superior de Tecnologias Avançadas*.

Com este trabalho procurava-se ser documentado e criado um programa capaz de comunicar com a *API* do ***I**nstituo **P**ortuguês da **M**eteorologia e **A**tmosfera* de modo a poder retornar ao utilizador final uma previsão da meteorologia pendente o dia e o local escolhidos. Para este programa ser aprovado este teria de cumprir os seguintes requisitos:

- Apresente uma lista de temperaturas mínimas e máximas para todas as regiões disponibilizadas na API;
- Calcule a temperatura e precipitação média para todo o país;
- Permita especificar uma região e a data, obtendo todos os resultados para a data especificada.

Com estes requisitos em mente, neste trabalho será documentado o meu processo lógico em torno da aplicação construida. Note-se que decidi desde partida construir a mesma através da FrameWork "*Console Apps*", ou seja, será um programa sem qualquer interface gráfico, sendo que todo o seu manuseamento será efetuado com recurso a um terminal.

De destacar também que esta aplicação foi construida num sistema de base *Windows10*, onde todo o código foi escrito no Visual Studio Community, e o programa foi compilado e executado com ajuda dos utilitários provenientes do mesmo. Note-se ainda que foram adicionadas as FrameWorks "Newtonsoft Json.NET" para manipulação de Json's e "System.Net.Http" para efetuar os pedidos à API.

Source Code

Todo o código existente neste projeto foi desenhado, construido e testado por mim, tomando recurso da documentação oficial tanto da linguagem C# como de todas as bibliotecas usadas.

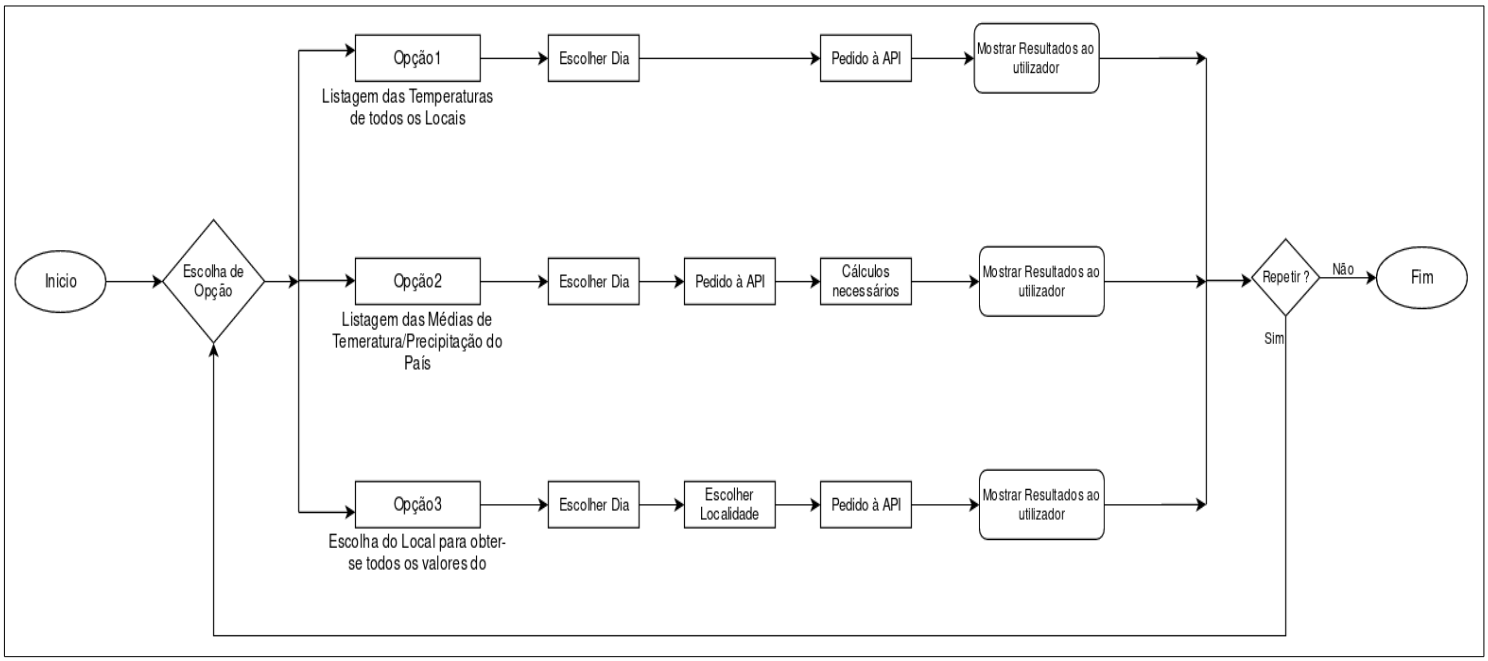
Newtonsoft Json.NET | System.Net.Http

Todo o código fonte deste projeto poderá ser consultado através do mesmo *.zip* onde este *.pdf* se encontrava, no ficheiro de nome "***Program.cs***", contudo e por questões de redundância também poderá ser consultado através do seguinte link.

Planeamento e Estrutura Lógica do Programa

Antes de sequer começarmos a desenvolver/explicar qualquer código, devemos perceber e delinear como será o funcionamento geral do programa. Para isso, podemos visualmente desenhar o mesmo num diagrama de ações, de forma a percebermos como é que devemos passar estas, para uma implementação em ***C#***.

Portanto, de uma forma global e para cumprirmos os requisitos impostos inicialmente por este exercício, podemos assentar o desenvolvimento numa esquematização deste género:



Tal como pode ser percebido pelo diagrama acima, o programa terá de iniciar um Loop que será responsável por questionar ao utilizador que tipo de ação que este pretende executar, sendo que pendente a sua escolha (Opção1/ Opção2/ Opção3), o programa irá executar uma sequencia de ações especificas à função pretendida.

De uma forma geral essas ações irão na sua essência assentar na seguinte logística:

- Questionar ao utilizador sobre que dias ou localidades é que este pretende saber os valores meteorológicos;
- Pendente a escolha do utilizador efetuar os pedidos dos ficheiros necessários à API;
- Trabalhar de alguma forma os Dados para serem de seguida apresentados de forma formatada ao utilizador.

Ao fim de serem mostrados os dados ao utilizador, este deverá ter a escolha de poder sair do programa, apenas continuando caso pretenda ver outros dados ou chamar outras opções.

Análise do Código

Tal como já foi referenciado, todo o código fonte poderá ser consultado no link facultado acima. Sendo que serão colocados excertos do mesmo para se explicar em detalhe os seus funcionamentos.

Constantes declaradas a nível do NameSpace:

```
// URLs dos diferentes ficheiros a pedir à API.
//    -> Os primeiros 3 servem para obter os dados gerais dos diferentes locais do Pais.
//    -> Os últimos 3 são para "traduzir" os ID's presentes nos dados principais, para strings escritos em Português.

        const string url1 = "http://api.ipma.pt/open-data/forecast/meteorology/cities/daily/hp-daily-forecast-day0.json";
        const string url2 = "http://api.ipma.pt/open-data/forecast/meteorology/cities/daily/hp-daily-forecast-day1.json";
        const string url3 = "http://api.ipma.pt/open-data/forecast/meteorology/cities/daily/hp-daily-forecast-day2.json";
```

```
        const string urlLocalidades = "http://api.ipma.pt/open-data/distrits-islands.json";
        const string urlTempo = "http://api.ipma.pt/open-data/weather-type-classe.json";
const string urlVento = "http://api.ipma.pt/open-data/wind-speed-daily-classe.json";
```

Neste excerto de código estamos a declarar de forma constante quais o URLs que serão usados ao longo da execução do programa, note-se que estas são as únicas variáveis estáticas no nosso código pois na perspetiva desta construção, estes links são na realidade a única certeza que temos, sobre onde vamos obter os nossos dados. Por mais que sejam manipuladas ou alteradas diferentes variáveis ao longo da execução do programa, sabemos que estas serão sempre iguais.

Classes Modelo:

Quando comunicarmos com a API do IPMA, vamos obter uma string de dados organizados num formato mundialmente reconhecido como Json, deste modo temos de ter uma maneira de modelar estes dados obtidos num Objeto que possa ser manipulado pela linguagem que está a ser utilizada. Para conseguirmos efetuar isto mesmo, vamos necessitar de especificar através de classes no nosso programa como é que estes Objetos serão compostos, ou seja quais os seus atributos.

Uma questão importante a ter em conta é que na realidade estas classes têm de respeitar na totalidade a estrutura dos Json's que iram armazenar, isto para mais tarde não surgirem dificuldades a nível da atribuição dos dados obtidos pela API nos respetivos Objetos. Outra questão a ter em conta no nosso programa em particular é que neste, vamos comunicar com a API pedindo 4 ficheiros diferentes, os Dados, as Localidades, os Tipos de Tempo e os Tipos de Vento, deste modo temos de ter no mínimo 4 Classes Modelo que necessitam de ser criadas.

Contudo, vamos analisar o resultado obtido pela API quando pedimos o ficheiros dos dados referentes ao dia de amanhã:


```
{
 "owner": "IPMA",
 "country": "PT",
 "forecastDate": "2019-02-11",
 "data": [
```

```
  {
   "precipitaProb": "0.0",
   "tMin": 5,
   "tMax": 14,
   "predWindDir": "N",
   "idWeatherType": 2,
   "classWindSpeed": 1,
   "longitude": "-8.6535",
   "globalIdLocal": 1010500,
   "latitude": "40.6413"
  },
(...)
 ],
 "dataUpdate": "2019-02-11T17:31:05"
}
```

Como podemos ver através do excerto de código acima, na realidade temos uma lista de valores primária composta por : "*owner*" , "*country*" , "*forecastDate*" , "*data*" e "*dataUpdate*" . No entanto, dentro do nosso campo "*data*" temos um outro sub-conjunto de atributos referentes a cada uma das localidades que o IPMA devolve resultados.

Deste modo podemos encarar esta estrutura como sendo uma Classe *Modelo* primária que armazena uma Lista de tipo "*data*" de dados secundários.

Uma vez conhecida esta estrutura, basicamente duplicamos as nossas classes Modelos a criar, sendo que para cada um dos ficheiros a serem pedidos à API temos de ter 2 classes modelo para os armazenar, a classe primária e a classe secundária onde realmente vão surgir os dados.

Visto ser desnecessário a repetição do código por cada uma das Classes necessárias, aqui abaixo deixo apenas a estrutura de exemplo necessária para comunicar com a API de forma a obter os valores gerais das temperaturas em todas as localidades disponíveis:

```
// Classes Modelo Para Valores.

        public class Dados
        {
            public float precipitaProb { get; set; }
            public int tMin { get; set; }
            public int tMax { get; set; }
            public string predWindDir { get; set; }
            public int idWeatherType { get; set; }
            public int classWindSpeed { get; set; }
            public string longitude { get; set; }
            public int globalIdLocal { get; set; }
```

```
            public string latitude { get; set; }
            public int? classPrecInt { get; set; }
        }
        public class Modelo
        {
            public string owner { get; set; }
            public string country { get; set; }
            public string forecastDate { get; set; }
            public List<Dados> data { get; set; }
            public string dataUpdate { get; set; }
}
```

Classe de armazenamento dos Métodos a serem evocados:

De forma a segmentar mais o nosso código e visto que pretendia que o mesmo fosse todo visível apenas em um único documento, decidi que todas os métodos que fosse necessitar ao longo do programa seriam colocados dentro de uma única classe de forma a mais tarde sempre que precisa-se, seria apenas necessário o instanciamento de uma única classe para aceder a todos os métodos disponíveis.

Abaixo vão ficar todos os métodos que foram criados para o funcionamento do programa. Note-se porem que apenas no próximo capitulo é que será explicado a estrutura da nossa classe "*Main*", sendo que é a partir desta que os nossos métodos serão chamados, tornando assim mais fácil a interpretação dos mesmos dado o contexto em que estes são evocados.

Banner:

```
// O Método "Banner()" serve apenas propósitos estéticos, sendo evocado para por
o terminal mais apelativo.

            public void Banner()
            {
                Console.Clear(); //Limpar Ecrã.

Console.WriteLine("##############################################################
#####################################################");
                Console.WriteLine("\t\t\t\tBem Vindo ao Meu Previsor de
Meteorologia!");
                Console.WriteLine("\t\t\tProjeto desenvolvido no ambito do
Módulo POO para o ISTEC.");
                Console.WriteLine("\t\tTradutor da API do IPMA-Instituo
Portugues da Meteorologia e Atmosfera");
                Console.WriteLine("\t\tTodo o Código deste utilitário foi
escrito e testado por: Daniel Vaz.nº30183");

Console.WriteLine("##############################################################
#####################################################");
                Console.WriteLine("");
            }
```

Este método não tem qualquer utilidade prática no programa, apenas existe para esteticamente o programa se tornar mais apresentável , sendo que este método é responsável por limpar o ecrã e colocar algumas informações pertinentes no topo do terminal.

Confirmador de variáveis numéricas:

```
// O Método "OpcoesCheck()" serve para retornar um valor booleano informando o
resto do programa,
// se o string que lhe foi passado pode ser de forma segura convertido para
inteiro, ou não.
```

```
public bool OpcoesCheck(string temp_opc)
{
    return int.TryParse(temp_opc, out int opc);
}
```

Este método apesar de simples é de grande importância isto pois este é o responsável por receber um variável e verificar se esta pode ser convertida sem problemas para um numero inteiro. Caso não seja despoletado nenhum erro este método retorna "***true***" , caso não seja possível esta conversão, o método retorna "***false***" .

Opção 1:

```
// O Método "Opcao1()" é chamado caso o utilizador queira obter as Temperaturas
Min\Max de todas as Localidades.
//  - Após Comunicar com o respetivo URL, irá obter o resultado da API das
Localidades e Dados Gerais.
//  - Vai organizar os dados e apresentar os mesmos ao utilizador.
```

```
public async Task Opcao1(string url)
{
    Funcoes f = new Funcoes();
    Console.Clear();
    f.Banner();
    HttpClient cliente = new HttpClient();
    var conteudo = await cliente.GetStringAsync(url); // Obter Dados
Gerais.
    Modelo valores =
JsonConvert.DeserializeObject<Modelo>(conteudo);
    var localidades = await cliente.GetStringAsync(urlLocalidades);
// Obter Localidades.
    Localidades registos =
JsonConvert.DeserializeObject<Localidades>(localidades);

    // Abaixo segue-se toda apresentação dos dados ao utilizador.
    Console.WriteLine("======================");
    foreach (var i in valores.data)
```

```
            {
                foreach (var i2 in registos.data)
                {
                    if (i2.globalIdLocal == i.globalIdLocal)
                    {
                        Console.WriteLine("ID_Localidade: {0}", i2.local);
                    }
                }
                Console.WriteLine("Temperatura Min.: {0}º", i.tMin);
                Console.WriteLine("Temperatura Max.: {0}º", i.tMax);
                Console.WriteLine("======================");
            }
            Console.ReadLine(); // Reter a consola para dar tempo para
Leitura.
        }
```

Este método é um dos 3 principais do meu programa, sendo que corresponde a uma das 3 ações que este é possível de executar. Este método é chamado sempre que o utilizador quiser verificar os resultados das temperaturas Máximas e Mínimas de todas as localidades disponíveis na API.

De forma a ser dissecado o seu funcionamento podemos visualizar fundamentalmente as seguintes etapas do seu processamento:

- Existe um contacto com a API de modo a ser solicitado um determinado URL;
- O Conteúdo obtido é "Deserialized" ou seja "Descodificado" sendo este associado ao objeto do seu respetivo tipo;
- São iterados todos os diferentes valores de forma a serem apresentados apenas aqueles que são os pretendidos.

Pode também ser constatado que no meu caso acima de forma a os dados apresentados estarem devidamente associados ao nome completo das respetivas localidades, é feita também uma correlação dos dados obtidos pelo ficheiro das Localidades, deste modo invés de serem apresentados apenas com um ID_Local, são apresentados com o nome completo.

Opção 2:

```
// O Método "Opcao2()", é usado caso o utilizador pretenda ver os resultados
Médios
// de Probabilidade de chuva e Temperatura.
//  - Após Comunicar com o respetivo URL, irá obter o resultado da API dos Dados
Gerais.
//  - Vai iterar por todos os valores que pretende e efetuar os cálculos
necessários(Somas e Médias).
```

```
//  - Vai organizar os dados e apresentar os mesmos ao utilizador.

            public async Task Opcao2(string url)
            {
                Funcoes f = new Funcoes();
                Console.Clear();
                f.Banner();
                HttpClient cliente = new HttpClient();
                var conteudo = await cliente.GetStringAsync(url); // Obter Dados
Gerais.
                Modelo valores =
JsonConvert.DeserializeObject<Modelo>(conteudo);

                int contador = 0; // Iterador
                float Tprecipit = 0; // Total de Probailidade de Chuva em Todo o
Pais.
                float Ttemp = 0; // Total da Média das temperatura de Chuva em
Todo o Pais.

                foreach (var i in valores.data)
                {
                    var precipit = i.precipitaProb;
                    var minima = i.tMin;
                    var maxima = i.tMax;

                    contador++;
                    Tprecipit += precipit;
                    Ttemp += ((minima + maxima) / 2);
                }

                Tprecipit = Tprecipit / (contador - 1);
                Ttemp = Ttemp / (contador - 1);

                // Apresentar dados Finais ao utilizador.

Console.WriteLine("=========================================");
                Console.WriteLine("Temperatura Média de todo o Pais: {0}°",
String.Format("{0:0.00}", Tprecipit));
                Console.WriteLine("Probabilidade de Chuva em todo o Pais: {0}%",
String.Format("{0:0.00}", Ttemp));

Console.WriteLine("=========================================");
                Console.ReadLine(); // Reter a consola para dar tempo para
Leitura.
            }
```

O método "*Opcao2()*" é evocado sempre que o utilizador quiser saber a média dos valores referentes ás temperaturas e probabilidades de precipitação de todo o País.

Este método têm uma lógica semelhante ao anterior, sendo que a Azul, podemos ver todo o código que é facilmente reutilizado entre funções, seguindo a mesma lógica. A principal diferença deste método em relação ao anterior é que aquando a iteração cíclica de valores estes não devem ser de imediato apresentados em terminal, mas sim serem incrementados em variáveis de forma a

após a obtenção de todos os dados, serem efetuados os cálculos necessários para as médias e apenas após estes cálculos terminarem, serem apresentados os dois valores finais ao utilizador.

Opção 3:

```
// O Método "Opcao3()" é usado caso o utilizador pretenda escolher um local
especifico,
// de modo a obter todos os dados disponíveis sobre o mesmo.
//  Este método segue uma lógica semelhante aos anteriores sendo que são usados
todos os diferentes URLs,
//  de modo a podermos apresentar os dados todos personalizados, e em Português
invés de ID's.

            public async Task Opcao3(string url, int localId)
            {
                Funcoes f = new Funcoes();
                Console.Clear();
                f.Banner();
                HttpClient cliente = new HttpClient();
                var conteudo = await cliente.GetStringAsync(url); // Obter Dados
Gerais.
                Modelo valores =
JsonConvert.DeserializeObject<Modelo>(conteudo);
                var conteudoVento = await cliente.GetStringAsync(urlVento); //
Obter Dados Sobre os Ventos.
                ModeloVentos valoresVentos =
JsonConvert.DeserializeObject<ModeloVentos>(conteudoVento);
                var conteudoTempo = await cliente.GetStringAsync(urlTempo); //
Obter Dados sobre o Tipo de Tempo.
                ModeloTempo valoresTempo =
JsonConvert.DeserializeObject<ModeloTempo>(conteudoTempo);
                var conteudoLocalidade = await
cliente.GetStringAsync(urlLocalidades); // Obter Localidades.
                Localidades localidade =
JsonConvert.DeserializeObject<Localidades>(conteudoLocalidade);

// Alguns registos nem sempre devolvem informações.
// De modo a reduzir problemas inesperados que comprometeriam o programa,
/ é usado este valor Booleano para confirmar se existem ou não resultados
disponíveis.
                bool result = false;

                string zona = ""; // Guardar nome da Localidade que foi
escolhida.
                foreach (var i in localidade.data)
                {
                    if (localId == i.globalIdLocal)
                    {
                        zona = i.local;
                    }
                }

                foreach (var i in valores.data)
                {
                    if (i.globalIdLocal == localId)
```

```
                    {
                        result = true;
// Apresentação dos Resultados ao utilizador.

Console.WriteLine("====================={0}============================", zona);
                        Console.WriteLine("Probabilidade de Percipitação: {0}%",
i.precipitaProb);
                        Console.WriteLine("Temperatura Minima: {0}°", i.tMin);
                        Console.WriteLine("Temperatura Máxima: {0}°", i.tMax);
                        Console.WriteLine("Direção do Vento: {0}",
i.predWindDir);
                        foreach (var x in valoresTempo.data)
                        {
                            if (x.idWeatherType == i.idWeatherType)
                            {
                                Console.WriteLine("Tipo de Tempo: {0}",
x.descIdWeatherTypePT);
                            }
                        }
                        foreach (var y in valoresVentos.data)
                        {
                            if (y.classWindSpeed == i.classWindSpeed)
                            {
                                Console.WriteLine("Velocidade de vento: {0}",
y.descClassWindSpeedDailyPT);
                            }
                        }
                        Console.WriteLine("Longitude: {0}", i.longitude);
                        Console.WriteLine("Latitude: {0}", i.latitude);

Console.WriteLine("============================================================"
);
                    }
                }

                if (!result)
                {

Console.WriteLine("============================================================
==========");
                    Console.WriteLine("!! De momento não temos dados
relativamente á localidade selecionada. !!");

Console.WriteLine("============================================================
==========");
                    Console.ReadLine(); // Reter a consola para dar tempo para
Leitura.
                }

                Console.ReadLine(); // Reter a consola para dar tempo para
Leitura.
            }
```

Este é o ultimo dos 3 principais métodos dos programa, sendo que é apenas mais vasto pois comunica com a API solicitando todos os ficheiros disponibilizados pela mesma e necessita de mais de um parâmetro em comparação com as outras, visto ser este o método que é chamada sempre que pretendemos saber os valores referentes a uma localização especifica.

Por questões de simplificação, mais uma vez as secções que são "reutilizáveis" entre métodos serão marcadas em Azul.

Fora a lógica que já foi explicada em métodos anteriores, neste método também temos um sistema de forma a prevenir erros - caso o utilizador tenha escolhida uma localidade onde de momento o IPMA não possua dados sobre a mesma é devidamente notificado sobre a inexistência dos mesmos, fazendo assim "bypass" a um possível erro a quando execução do programa.

Escolha de Data:

```
// O Método "EscolhaDia()" é usado sempre que pretendemos questionar ao
utilizador, qual o dia que pretende saber os dados.

            public string EscolhaDia()
            {
                Funcoes f = new Funcoes();
                int resposta = 0;
                while (resposta == 0)
                {
                    Console.Clear();
                    f.Banner();
                    Console.WriteLine("Quer ver os resultados de que dia?");
                    Console.WriteLine("[1] - Hoje;");
                    Console.WriteLine("[2] - Amanhã;");
                    Console.WriteLine("[3] - Depois-de-Amanhã.");
                    Console.Write("## ");
                    string temp_resposta = Console.ReadLine();
                    if (f.OpcoesCheck(temp_resposta))
                    {
                        resposta = Convert.ToInt32(temp_resposta);
                        switch (resposta)
                        {
                            case 1:
                                return url1;
                            case 2:
                                return url2;
                            case 3:
                                return url3;
                            default:
                                Console.WriteLine("Não escolheu uma opção
Válida!");
                                Console.ReadLine(); // Reter a consola para dar
tempo para Leitura.
                                resposta = 0;
                                Environment.Exit(0);
                                break;
                        }
                    }
                    else
                    {
                        Console.WriteLine("Não escolheu uma opção válida!");
```

```
                    Console.ReadLine(); // Reter a consola para dar tempo
para Leitura.
                    resposta = 0;
                }
            }
            return "";
        }
```

Este é um dos 2 métodos auxiliares aos outros 3 métodos principais. Isto pois, é através deste método que o utilizador tem a capacidade de escolher o dia sobre qual pretende obter os valores.

Este método inicia-se questionando o utilizador para escolher qual o dia que pretende usar como base para obter os valores referentes ao mesmo. De seguida, e após confirmação da resposta obtida ser válida, é efetuado um "Switch\ Case" para ser retornado qual o URL que deve ser usado como parâmetro para um dos métodos principais (***Opcao1()/ Opcao2() / Opcao3()***). Note-se que o URL que é retornado por este método será um dos definidos no inicio do programa como constantes.

Escolha da Localidade:

```
// O Método "EscolhaLocal()" é usado para auxiliar o método "Opcao3()", de modo
a poder se saber, o local que o utilizador quer usar para se saber os
resultados.
        public int EscolhaLocal()
        {
            Funcoes f = new Funcoes();
            int resposta = 0;
            while (resposta == 0)
            {
                Console.Clear();
                f.Banner();
                Console.WriteLine("Quer ver os resultados de que
Localidade?");
                Console.WriteLine("[1] - Aveiro \t\t\t\t [16] - Setúbal");
                Console.WriteLine("[2] - Beja \t\t\t\t [17] - Viana do
Castelo");
                Console.WriteLine("[3] - Braga \t\t\t\t [18] - Vila Real");
                Console.WriteLine("[4] - Bragança \t\t\t\t [19] - Viseu");
                Console.WriteLine("[5] - Castelo Branco \t\t\t [20] -
Funchal");
                Console.WriteLine("[6] - Coimbra \t\t\t\t [21] - Porto
Santo");
                Console.WriteLine("[7] - Évora \t\t\t\t [22] - Vila do
Porto");
                Console.WriteLine("[8] - Faro \t\t\t\t [23] - Ponta
Delgada");
                Console.WriteLine("[9] - Guarda \t\t\t\t [24] - Agra do
Heroísmo");
```

```
                Console.WriteLine("[10] - Leiria \t\t\t\t [25] - Santa Cruz
da Graciosa");
                Console.WriteLine("[11] - Lisboa \t\t\t\t [26] - Velas");
                Console.WriteLine("[12] - Lisboa - Jardim Botânico \t [27] -
Madalena ");
                Console.WriteLine("[13] - Portalegre \t\t\t [28] - Horta");
                Console.WriteLine("[14] - Porto \t\t\t\t [29] - Santa Cruz
das Flores");
                Console.WriteLine("[15] - Santarém \t\t\t [30] - Vila do
Corvo");
                Console.Write("## ");
                string temp_resposta = Console.ReadLine();
                if (f.OpcoesCheck(temp_resposta))
                {
                    resposta = Convert.ToInt32(temp_resposta);
                    switch (resposta)
                    {
                        case 1:
                            return 1010500;
                        case 2:
                            return 1010500;
                        case 3:
                            return 1030300;
                        case 4:
                            return 1040200;
                        case 5:
                            return 1050200;
                        case 6:
                            return 1060300;
                        case 7:
                            return 1070500;
                        case 8:
                            return 1080500;
                        case 9:
                            return 1090700;
                        case 10:
                            return 1090700;
                        case 11:
                            return 1110600;
                        case 12:
                            return 1110622;
                        case 13:
                            return 1121400;
                        case 14:
                            return 1131200;
                        case 15:
                            return 1141600;
                        case 16:
                            return 1151200;
                        case 17:
                            return 1160900;
                        case 18:
                            return 1171400;
                        case 19:
                            return 1182300;
                        case 20:
                            return 2310300;
                        case 21:
                            return 2320100;
                        case 22:
                            return 3410100;
```

```
                        case 23:
                            return 3420300;
                        case 24:
                            return 3430100;
                        case 25:
                            return 3440100;
                        case 26:
                            return 3450200;
                        case 27:
                            return 3460200;
                        case 28:
                            return 3470100;
                        case 29:
                            return 3480200;
                        case 30:
                            return 3490100;
                        default:
                            Console.WriteLine("Não escolheu uma opção
Válida!");
                            Console.ReadLine(); // Reter a consola para dar
tempo para Leitura.
                            resposta = 0;
                            Environment.Exit(0);
                            break;
                    }
                }
                else
                {
                    Console.WriteLine("Não escolheu uma opção válida!");
                    Console.ReadLine(); // Reter a consola para dar tempo
para Leitura.
                    resposta = 0;
                }
            }
            return 0;
        }

    }
```

Neste ultimo método semelhante-mente ao anterior é questionado ao utilizador qual a localidade que este pretende usar para saber os respetivos valores. Sendo que mais uma vez após a confirmação da resposta, é efetuado outro "Switch/Case" de modo a ser retornado um ID que está por sua vez diretamente associado a uma localidade na API do IPMA.

Esta função é apenas utilizada quando o utilizador chama pelo método "***Opcao3()***" isto pois é neste método que precisamos saber de antemão, qual a localidade que o utilizador quer usar para serem obtidos todos os dados disponíveis na API.

Classe Main:

```
public static void Main(string[] args)
        {

//Instanciamento de classe Funcoes.
//  Nesta classe vamos ter acesso a todos os métodos que foram criados para executar certas ações.
            Funcoes f = new Funcoes();

            int opc = 0;
            string url;
            int localId;

// Loop While "Infinito"
//  Aqui forçamos a consola num ciclo constante. Deste modo o utilizador pode executar os comandos,
//  as vezes que quiser, dando assim uma continuidade á utilização do programa.

            while (opc == 0)
            {
                Console.Clear();
                f.Banner();
                Console.WriteLine("O que Pretende ver? (Escolha uma Opção)");
                Console.WriteLine("[1] - Ver Listagem de Temperaturas;");
                Console.WriteLine("[2] - Ver a Média das Temperaturas do Pais;");
                Console.WriteLine("[3] - Escolher Região Especifica.");
                Console.Write("## ");
                string temp_opc = Console.ReadLine();
                if (f.OpcoesCheck(temp_opc))
                {
                    opc = Convert.ToInt32(temp_opc);
                    switch (opc)
                    {
                        case 1:
                            url = f.EscolhaDia();
                            f.Opcao1(url).Wait();
                            break;
                        case 2:
                            url = f.EscolhaDia();
                            f.Opcao2(url).Wait();
                            break;
                        case 3:
                            url = f.EscolhaDia();
                            localId = f.EscolhaLocal();
                            f.Opcao3(url, localId).Wait();
                            break;
                        default:
                            Console.WriteLine("Não escolheu uma opção Válida!");
                            opc = 0;
                            Console.ReadLine();
                            break;
                    }
                }
                else
                {
                    Console.WriteLine("Não escolheu uma opção Válida!");
                    Console.ReadLine();
                }
```

```
// Saída do While Loop:
//  É aqui questionado ao utilizador se este pretende sair ou não do programa.
//  Terminando o Loop e consequentemente terminando o programa.

                Console.Clear();
                f.Banner();
                Console.WriteLine("\t\t\t\t\tDeseja continuar? (S)im ou (N)ao");
                Console.Write("## ");
                string exit = Console.ReadLine();
                if (exit == "s" || exit == "S" || exit == "sim" || exit ==
"Sim")
                {
                    opc = 0;
                }
                else
                {
                    Console.Clear();
                    f.Banner();
                    Console.WriteLine("\t\t\t\t\tObrigado pela sua utilização.
Adeus!");
                    Console.ReadLine();
                    Environment.Exit(0);
                }
            }
 Console.ReadLine(); // Reter a consola para dar tempo para Leitura.
}
```

Por esta altura, deve ser já compreendido que a classe Main é a classe por onde o programa começa a ser executado, sendo que devem ser feitas as necessárias evocações dos restantes métodos auxiliares nesta secção do programa.

Analisando a nossa classe *Main, v*emos que fundamentalmente tomamos as medidas necessárias para inicializarmos um "*While Loop 'Infinito'*", isto pois este loop é o que vai permitir dar continuidade ao funcionamento do nosso programa sendo que caso o utilizador termine de obter um conjunto de resultados, se pretender obter outros diferentes o possa fazer seguidamente não necessitando de abrir o programa novamente. De notar, que sempre após a conclusão de uma determinada ação no programa, é questionado ao utilizador se este pretende continuar ou não, permitindo assim este ter o controlo sobre quando parar a utilização do programa.

Dentro do nosso "*While Loop*":

- Questionamos o utilizador sobre qual ação é que este pretende executar com o programa;
- Confirmamos se a resposta fornecida é válida;
- Pendente a escolha do user, será executado o respetivo método.

Com uma análise individualizada de cada um dos campos deste "Switch/Case", vemos que sempre antes de serem executados os métodos associados à escolha do utilizador, são chamados os métodos auxiliares "***EscolhaDia()***" e/ou "***EscolhaLocal()***" visto que são estes que vão retornar ao *Main* quais as informações especificas que o utilizador pretende obter, sendo assim passados como parâmetros para os métodos "***OpcoesX()***".

Conclusão

Apesar de não ser perfeito, o programa aqui demonstrado cumpre os requisitos impostos inicialmente e eleva a fasquia do desafio estando preparado para corrigir alguns possíveis erros por parte do utilizador, tornando-o assim numa solução mais robusta e eficiente. Para mais, graças a estrutura criada com os diferentes métodos e classes, damos liberdade ao programa para este ser facilmente expansível, podendo ser adicionado sem receio novas operações e ou mais mecanismos de controlo sobre o utilizador.

Espera-se que associado com esta documentação se tenha uma fácil abordagem ao condigo fonte do programa, permitindo assim a sua total interpretação e análise.

Obrigado pela sua leitura.