

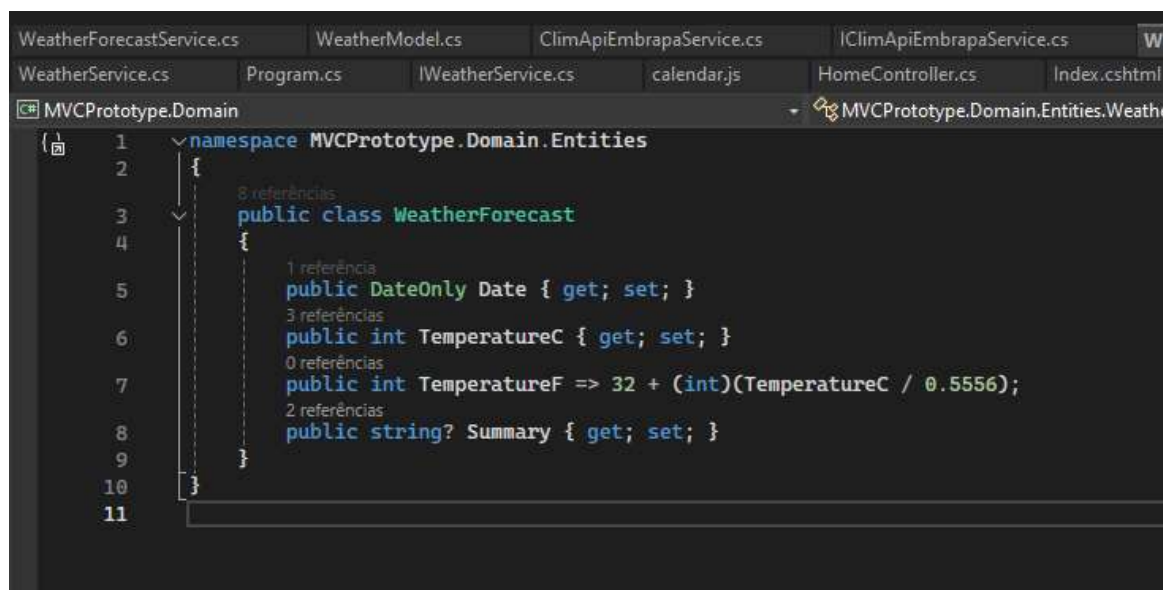
## Relatório de Atividades

O projeto é um programa em C# dotnet 8.0, em ambiente Visual Studio.

Executar as instruções abaixo:

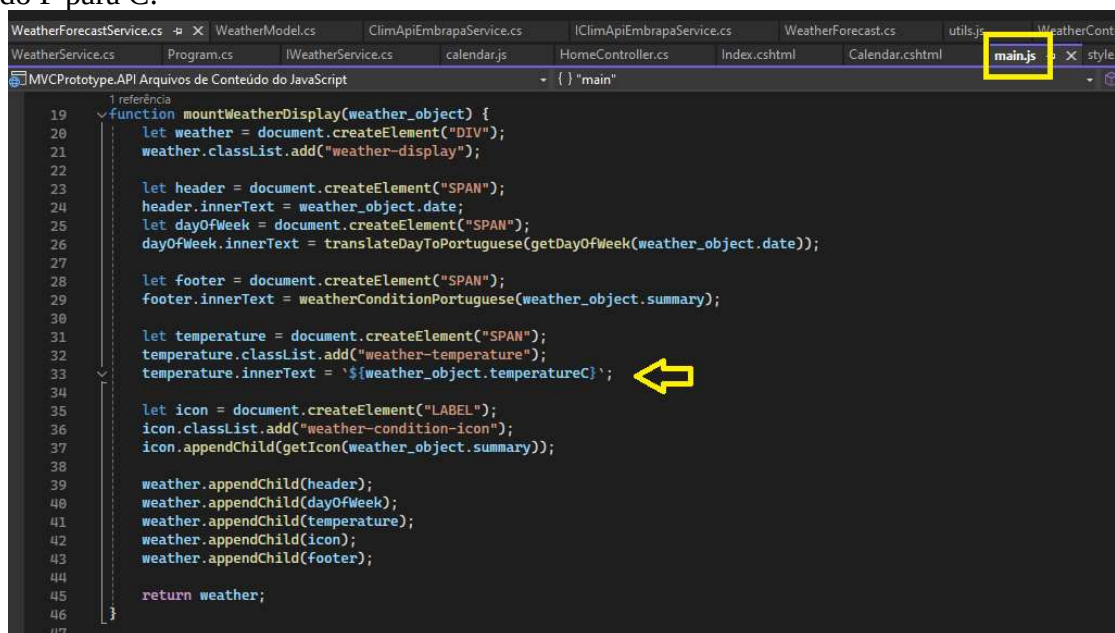
### 1. A temperatura na tela Previsão do Tempo está em Fahrenheit, alterar para Celcius;

No back-end, observa-se que a entidade já provê a temperatura em graus Celsius:




```
1 namespace MVCPrototype.Domain.Entities
2 {
3     public class WeatherForecast
4     {
5         public DateOnly Date { get; set; }
6         public int TemperatureC { get; set; }
7         public int TemperatureF => 32 + (int)(TemperatureC / 0.5556);
8         public string? Summary { get; set; }
9     }
10 }
11
```

Troquei a referência no front-end, passando a usar a propriedade TemperatureC e ajustei o estilo, trocando F para C:



```
19 function mountWeatherDisplay(weather_object) {
20     let weather = document.createElement("DIV");
21     weather.classList.add("weather-display");
22
23     let header = document.createElement("SPAN");
24     header.innerText = weather_object.date;
25     let dayOfWeek = document.createElement("SPAN");
26     dayOfWeek.innerText = translateDayToPortuguese(getDayOfWeek(weather_object.date));
27
28     let footer = document.createElement("SPAN");
29     footer.innerText = weatherConditionPortuguese(weather_object.summary);
30
31     let temperature = document.createElement("SPAN");
32     temperature.classList.add("weather-temperature");
33     temperature.innerText = `${weather_object.temperatureC}`;
34
35     let icon = document.createElement("LABEL");
36     icon.classList.add("weather-condition-icon");
37     icon.appendChild(getIcon(weather_object.summary));
38
39     weather.appendChild(header);
40     weather.appendChild(dayOfWeek);
41     weather.appendChild(temperature);
42     weather.appendChild(icon);
43     weather.appendChild(footer);
44
45     return weather;
46 }
47
```

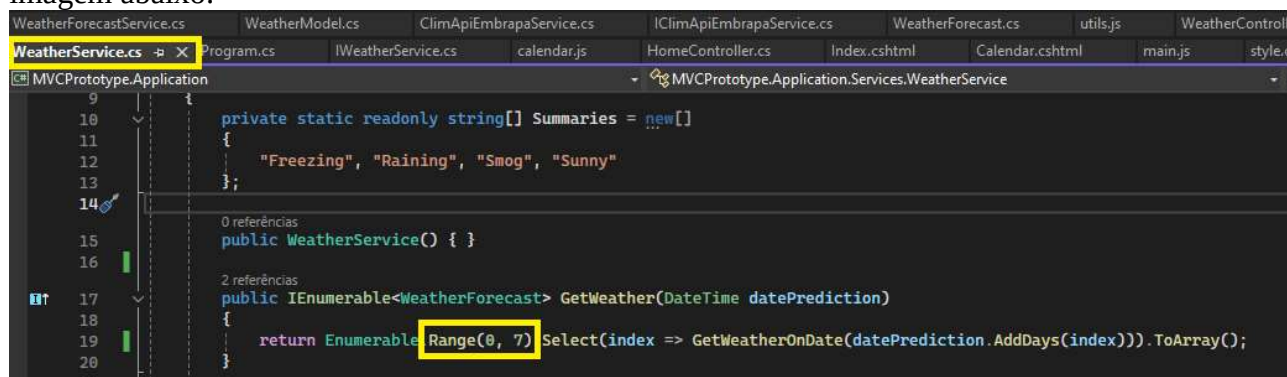
No estilo:



```
1 .weather-display .weather-temperature::after {
2     content: "C";
3     font-weight: bold;
4     font-size: 0.75em;
5 }
```

## 2. A previsão mostra 5 dias a partir do dia seguinte, ajustar para aparecer a previsão de 7 dias incluindo o dia atual;

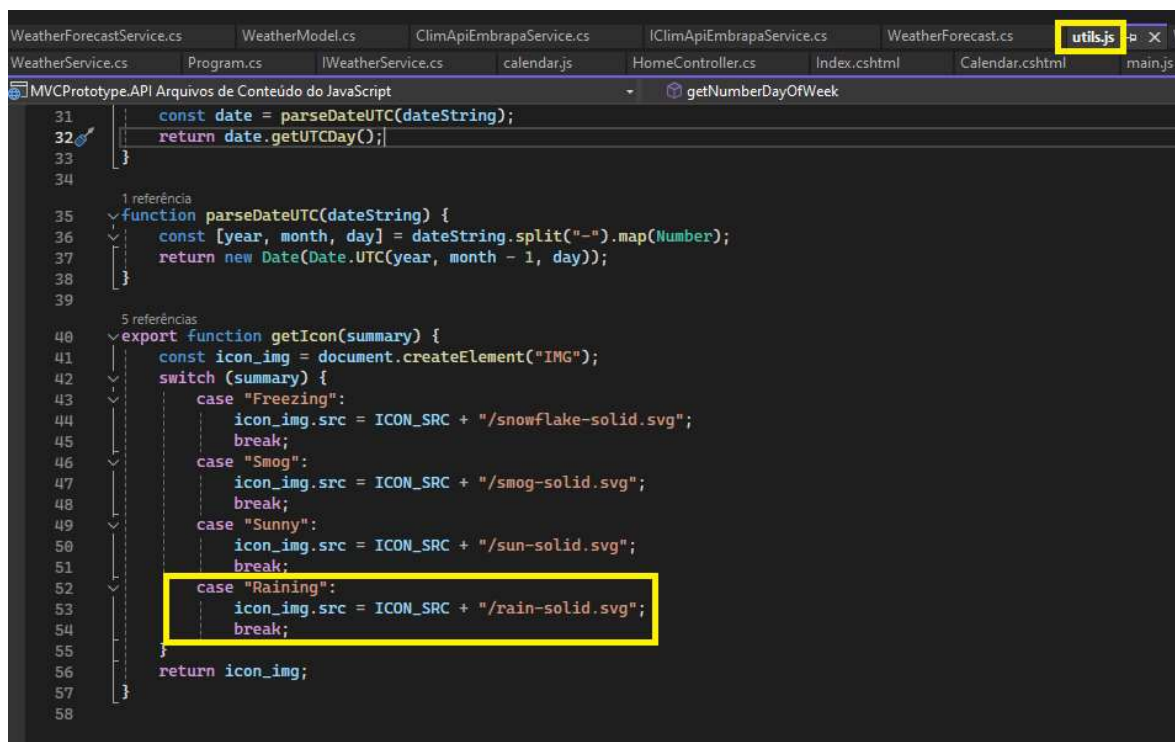
No back-end, no método GetWeather (em WeatherService), foi realizado o ajuste, conforme a imagem abaixo:



```
10 private static readonly string[] Summaries = new[]
11 {
12     "Freezing", "Raining", "Smog", "Sunny"
13 };
14
15 public WeatherService() { }
16
17 public IEnumerable<WeatherForecast> GetWeather(DateTime datePrediction)
18 {
19     return Enumerable.Range(0, 7).Select(index => GetWeatherOnDate(datePrediction.AddDays(index))).ToArray();
20 }
21
```

## 3. O clima "Raining" foi adicionado na API, mas não está aparecendo o ícone no front-end;

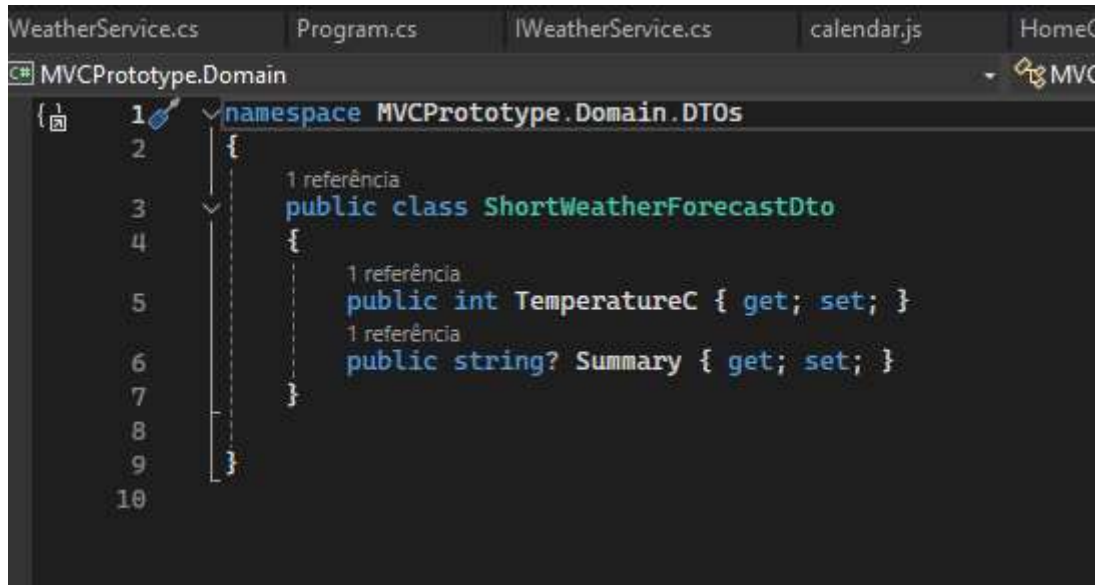
A verificação para "raining" não estava implementada no arquivo main.js, no front-end, não reconhecendo e nem carregando o ícone em questão. Como eu iria utilizar essa mesma função para o carregamento do ícone no calendário, refatorei deixando essa função e outras num arquivo à parte, o Utils.js:



```
31 const date = parseDateUTC(dateString);
32 return date.getUTCDay();
33 }
34
35 function parseDateUTC(dateString) {
36     const [year, month, day] = dateString.split("-").map(Number);
37     return new Date(Date.UTC(year, month - 1, day));
38 }
39
40 export function getIcon(summary) {
41     const icon_img = document.createElement("IMG");
42     switch (summary) {
43         case "Freezing":
44             icon_img.src = ICON_SRC + "/snowflake-solid.svg";
45             break;
46         case "Smog":
47             icon_img.src = ICON_SRC + "/smog-solid.svg";
48             break;
49         case "Sunny":
50             icon_img.src = ICON_SRC + "/sun-solid.svg";
51             break;
52         case "Raining":
53             icon_img.src = ICON_SRC + "/rain-solid.svg";
54             break;
55     }
56     return icon_img;
57 }
58
```

#### 4. Adicionar uma rota na API que recebe um range de datas e retorna uma lista do clima com somente temperatura e clima;

Foi implementado o método e, para trazer somente as propriedades solicitadas, criei uma entidade em DTO.



```
1 namespace MVCPrototype.Domain.DTOS
2 {
3     1 referência
4     public class ShortWeatherForecastDto
5     {
6         1 referência
7         public int TemperatureC { get; set; }
8         1 referência
9         public string? Summary { get; set; }
10    }
```

O respectivo endpoint na controller:



```
[Route("GetWeatherRangeDate")]
[HttpGet]
0 referências
public IActionResult getWeatherRangeDate(DateTime fromDate, DateTime toDate)
{
    try
    {
        var response = _weatherService.GetWeatherRangeDate(fromDate, toDate)
            .Select(weather => new ShortWeatherForecastDto
            {
                TemperatureC = weather.TemperatureC,
                Summary = weather.Summary
            });

        return StatusCode(200, response);
    }
    catch (ArgumentException ex)
    {
        return BadRequest(new { message = "Requisição inválida.", details = ex.Message });
    }
    catch (Exception ex)
    {
        return StatusCode(500, new { message = "Erro interno no servidor.", details = ex.Message });
    }
}
```

Criei uma tela para comprovar o funcionamento:

localhost:44391/Home/DadosCurtos

th Flight Tracker | Fligh... Find your QTH locat... AMSAT - AMSAT On... (10) Space Station C... Radio DXFUN Clust... EASP - ENTIDADES... YouTube

Home Calendário Dados da Embrapa para os próximos dias Temperatura e clima por período

# Temperatura e clima por período

Data Inicial: 06/02/2025 Data Final: 13/02/2025

Buscar

Temperatura	Clima
-16 °C	Ensolarado
-12 °C	Nublado
53 °C	Nublado
49 °C	Ensolarado
1 °C	Ensolarado
5 °C	Ensolarado
44 °C	Congelante
-13 °C	Congelante

Como há a possibilidade de digitação de um intervalo muito longo, considerei até 30 dias de possibilidade de retorno de dados:

Home Page - MVCPrototype

localhost:44391/Home/DadosCurtos

Google Earth Flight Tracker | Fligh... Find your QTH locat... AMSAT - AMSAT On... (10) Space Station C... Radio DXFUN Clust... EASP - ENTIDADES... YouTube

MVCPrototype Home Calendário Dados da Embrapa para os próximos dias Temperatura e clima por período

# Temperatura e clima por período

Data Inicial: 01/02/2025 Data Final: 04/03/2025

Buscar

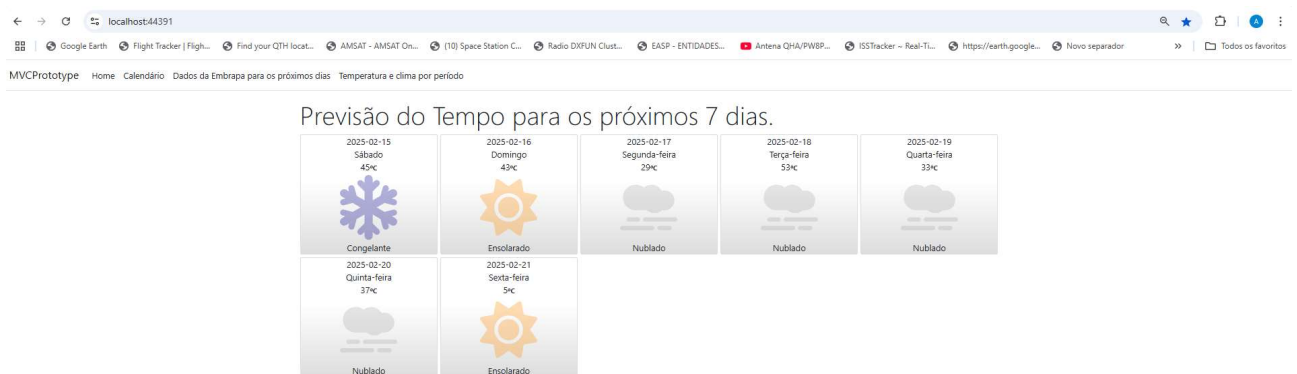
A previsão do tempo está restrita a um intervalo de até 30 dias.

Aproveitei para refatorar algumas dependências em WeatherService para oferecer mais reaproveitamento de código.

```
WeatherForecastService.cs | WeatherModel.cs | ClimApiEmbrapaService.cs | IClimApiEmbrapaService.cs | WeatherForecast.cs | utils.js | WeatherController.cs | NcepGfsData.cs | IWeatherForec...
WeatherService.cs | Program.cs | IWeatherService.cs | calendar.js | HomeController.cs | index.cshtml | Calendar.cshtml | main.js | style.css | Sho...
MVCPrototype.Application | MVCPrototype.Application.Services.WeatherService | WeatherService()

17 public IEnumerable<WeatherForecast> GetWeather(DateTime datePrediction)
18 {
19     return Enumerable.Range(0, 7).Select(index => GetWeatherOnDate(datePrediction.AddDays(index))).ToArray();
20 }
21
22 3 referências
23 public IEnumerable<WeatherForecast> GetWeatherRangeDate(DateTime fromDate, DateTime toDate)
24 {
25     return Enumerable.Range(0, GetDaysDifference(fromDate, toDate)+1).Select(index => GetWeatherOnDate(fromDate.AddDays(index))).ToArray();
26 }
27
28 2 referências
29 public IEnumerable<WeatherForecast> GetWeatherCurrentMonthYear(short month, short year)
30 {
31     if (month < 1 || month > 12)
32         throw new ArgumentOutOfRangeException(nameof(month), "O número do mês deve estar entre 1 e 12.");
33
34     if (year < (DateTime.Now).Year)
35         throw new ArgumentOutOfRangeException(nameof(month), "O ano deve ser maior ou igual ao ano corrente.");
36
37     var monthYear = string.Concat(month.ToString("D2"), "-", year.ToString("D4"));
38     var fromDate = ConvertToDate(string.Concat("01-", monthYear));
39     var toDate = ConvertToDate(string.Concat(DateTime.Date.ToString("D2"), "-", monthYear));
40     return GetWeatherRangeDate(fromDate, toDate);
41 }
42
43 2 referências
44 private WeatherForecast GetWeatherOnDate(DateTime date)
45 {
46     return (new WeatherForecast
47     {
48         Date = DateOnly.FromDateTime(date),
49         TemperatureC = Random.Shared.Next(-20, 55),
50         Summary = Summaries[Random.Shared.Next(Summaries.Length)]
51     });
52 }
```

No final, a tela com a previsão para os 7 dias (incluindo o dia atual) ficou assim:



Tomei a liberdade de alterar o título e foram contemplados também os seguintes itens do tópico “bônus”:

1. Os climas estão em inglês, apresentar no frontend em português;
2. Apresentar os dias da semana na Previsão do Tempo; (também realizei a tradução)

Sobre o outro item de “bônus”, para alterar a cor da imagem SVG

4. Alterar a cor dos ícones para #B4EEFA.

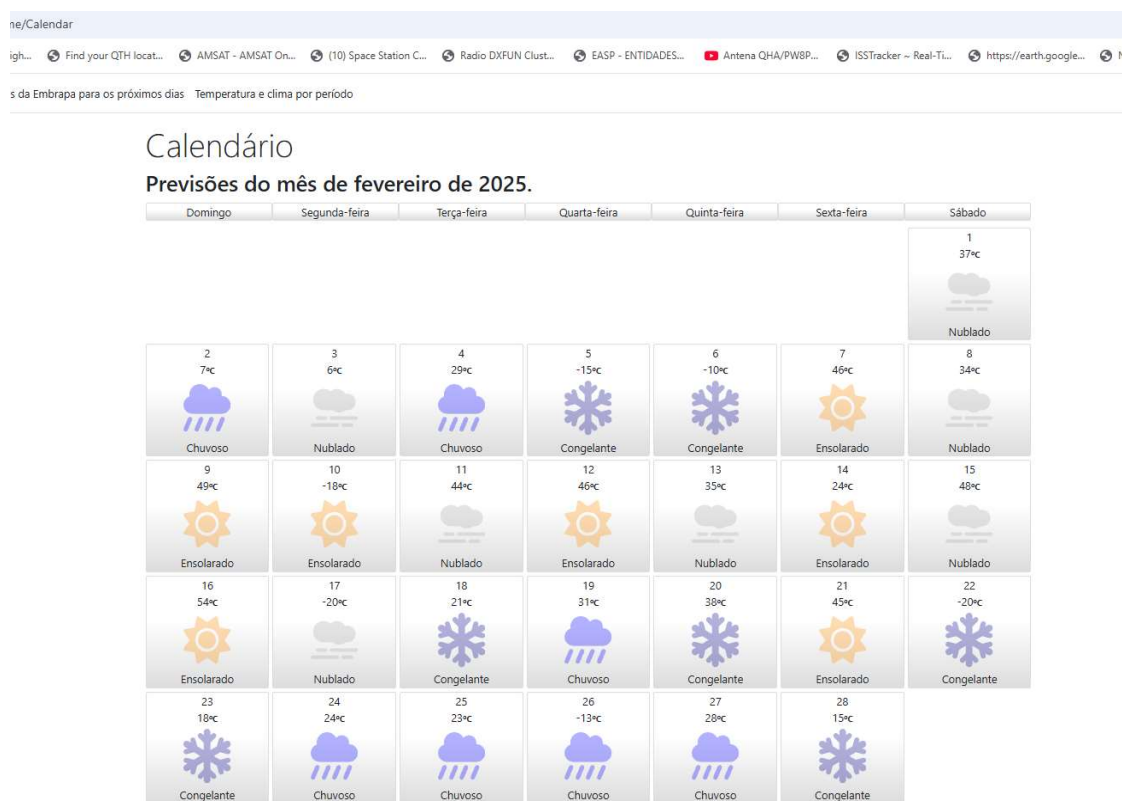


Aproveitei e adotei cores variadas conforme o clima, sendo o amarelo para dias ensolarados, um azul mais escuro para o clima congelante, um azul mais claro para o clima chuvoso e o cinza para dia nublado.

## 5. Implementar a tela Calendário, com a previsão do mês atual.

Para o calendário, vi que a aparência dos dados da primeira tela ficou com um bom visual, bem ilustrativo, pensei em manter esse visual, mas, respeitando o aspecto de um calendário, que apresenta todos os dias do mês corrente, iniciando a data do primeiro dia com deslocamento, conforme o dia da semana, de forma dinâmica. Vou tentar refatorar e ver se consigo uma abordagem melhor no trecho que “gera os quadros em branco”.

Procurei distinguir os estilos e criar outros, conforme a necessidade. Vi também que não tinha sentido em mostrar a data inteira, e, nesse caso passei a exibir somente o número do dia do mês, já que acima temos o nome do mês e o ano correntes, além de um título para os dias da semana, conforme a imagem abaixo:



Em ambas as telas (home e calendário), os dados continuam “mockados”.

Item bônus também contemplado:

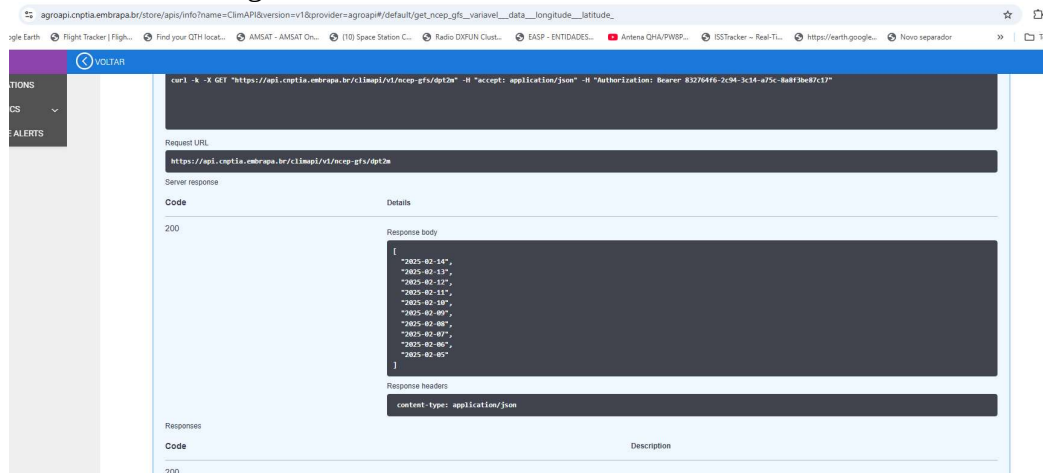
3. Apresentar os dias da semana no Calendário; (na forma de título no topo, para evitar repetição)  
Foi respeitado o deslocamento do primeiro dia do mês na semana.

## Bônus:

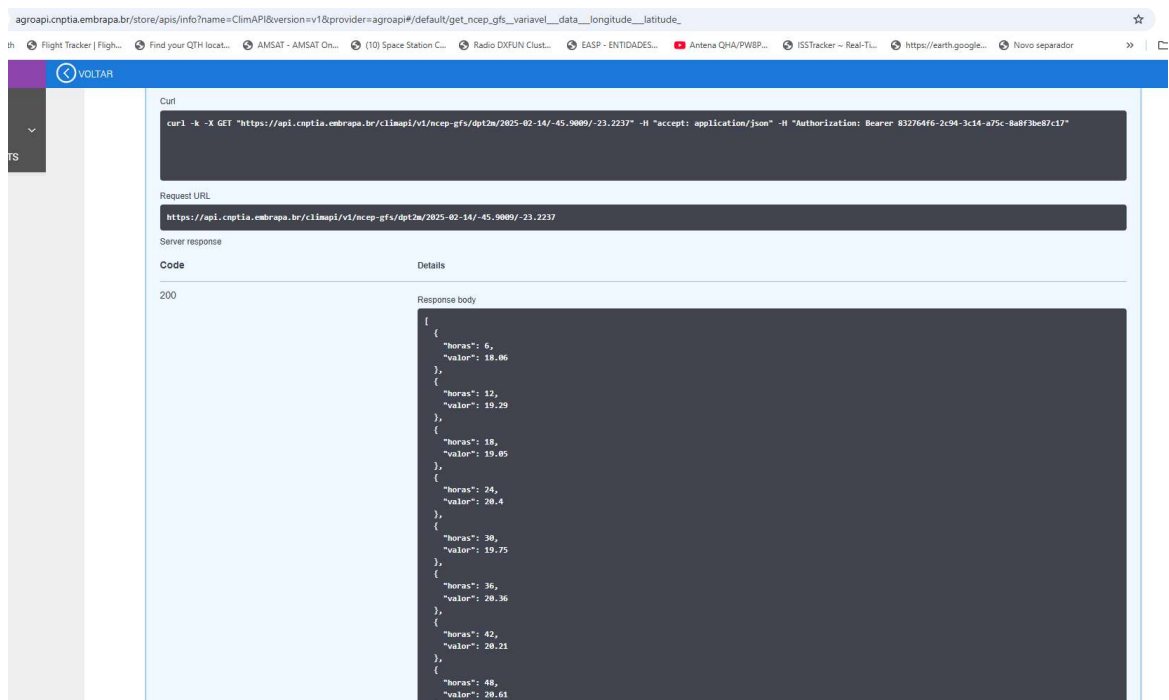
### 5. Integrar o backend com alguma API gratuita. (Ex. EMBRAPA)

Estive pesquisando algumas APIs com acesso liberado e que pudessem trazer os dados de modo a formar uma base para constituir as previsões. Talvez por eu ainda ser novo nesse seguimento, observei uma grande variedade de modelos de dados entre os objetos oferecidos. Como basicamente precisamos mostrar a temperatura e uma breve legenda que descreverá o “comportamento” do tempo para o dia, pensei em formar um modelo na memória que pudesse atender a essa necessidade. Observei também que, logicamente, os requests informam a localização, seja longitude e latitude ou identificação de município e a data de referência. Como se trata de previsão, observei também que as APIs normalmente retornam um range curto de dias, já que quanto mais longe vai, mais está sujeita a alterações, por conta do clima.

Pesquisei mais sobre a API da Embrapa, mais precisamente a ClimAPI e vi que ela tem dois endpoints que poderiam me ajudar. Um desses endpoints retorna uma lista de datas com dados disponíveis, conforme a imagem abaixo:



O outro endpoint é mais especializado e oferece previsões de valores de 6 em 6 horas, segregando os dados por tipo de variável climática.



Nesse caso, irei considerar as “horas” 6, 24, 48, 72... até 240, que representam 10 dias. Então, com isso podemos minimizar a quantidade de requests contra a API da Embrapa (que limita a 1000 para uso grátis) e podemos formar um objeto consolidando as variáveis climáticas por dia.

Como no final precisamos saber se o dia na data estará ensolarado, chuvoso, nublado ou “congelante”, procurei entender (ainda que num modelo muito simples) quais seriam as variáveis básicas para determinar esses status acima.

Encontrei um modelo (altamente simplista) que adotei:

```
private string WeatherForecastSummary()
{
    double SunsdsfcHours = Sunsdsfc / 3600.0;

    double TotalCloudCover = Lcdclcll + Mcdcmcll + Hcdchcll;

    if (Tmin2m < 3 || Tmpsfc < 5)
        return "Freezing";

    if (Apcpsfc > 3.5 || TotalCloudCover > 80)
        return "Raining";

    if (SunsdsfcHours > 6 && TotalCloudCover < 40)
        return "Sunny";

    return "Smog";
}
```

Separei então as variáveis climáticas de interesse e, na segunda parte do programa, parti para fazer os requests para as variáveis Tmin2m, Tmpsfc, Apcpsfc, Lcdclcll, Hcdchcll, Mcdcmcll e Sunsdsfc.

Criei também uma entidade para armazenar esse modelo de dados:

```
namespace MVCPrototype.Domain.Entities
{
    5 referências
    public class WeatherModel
    {
        2 referências
        public string? Date { get; set; }

        2 referências
        public double Tmin2m { get; set; }
        2 referências
        public double Tmpsfc { get; set; }
        2 referências
        public double Apcpsfc { get; set; }
        2 referências
        public double Lcdclcll { get; set; }
        2 referências
        public double Hcdchcll { get; set; }
        2 referências
        public double Mcdcmcll { get; set; }
        2 referências
        public double Sunsdsfc { get; set; }

        0 referências
        public string Summary => WeatherForecastSummary();
    }
}
```



A propriedade Summary resolverá dinamicamente a legenda do clima para o dia. Nesse ponto confesso que eu poderia ter utilizado Enums.

Em linhas gerais, o método abaixo é capaz de montar o modelo de até 20 dias, considerando também algumas datas passadas. Há casos em que a previsão ficou como undefined, porque o método que criei não sabe como resolver valores fora do que foram previstos. Optei também por manter a data no formato string aaaa-mm-dd.

Abaixo temos uma amostra de dados processados pelo método que consome a API ClimAPI da Embrapa.

Index	Date	Tmin2m	Tmpsfc	Apopsfc	Lodolcll	Hodchcll	Modomcll	Sunsdscfc	Summary
0	"2025-02-05"	20.04	19.67	0	4.1	100	99.5	0	"Smog"
1	"2025-02-06"	19.54	19.03	2.63	0.1	100	2.6	0	"Smog"
2	"2025-02-07"	19.44	18.68	0.06	2.8	44.1	0	0	"Undefined"
3	"2025-02-08"	19.19	18.75	0	0	0.8	2.8	0	"Undefined"
4	"2025-02-09"	18.11	17.54	0	6.2	54	0	0	"Undefined"
5	"2025-02-10"	18.41	18.02	0	0	94.7	0	0	"Smog"
6	"2025-02-11"	19.74	19.45	0	0	100	0	0	"Smog"
7	"2025-02-12"	20.3	19.69	0	0	99.6	0	0	"Smog"
8	"2025-02-13"	21.6	21.52	0.94	0	99.9	39.5	0	"Smog"
9	"2025-02-14"	19.16	18.63	0	0	48.4	25.4	0	"Undefined"
10	"2025-02-15"	20.7	20.9	28	73.8	88.9	96.8	7980	"Raining"
11	"2025-02-16"	22.08	21.35	4.25	24.9	12.1	5	13200	"Sunny"
12	"2025-02-17"	25.01	23.95	0	94.6	38.1	0.4	7141	"Undefined"
13	"2025-02-18"	24.15	22.75	0	29.8	0	0	13058	"Sunny"
14	"2025-02-19"	23.46	22.35	0	20.7	0.4	8.6	13050	"Sunny"
15	"2025-02-20"	23.7	23.36	0.75	25.7	36.8	67.9	8314	"Sunny"
16	"2025-02-21"	23.21	22.05	0.81	63.9	0.4	0.6	6638	"Undefined"
17	"2025-02-22"	23.61	23.22	0.25	7.9	2.9	0	12900	"Sunny"
18	"2025-02-23"	24.97	24.45	0.13	48.2	1.7	2	12644	"Undefined"
19	"2025-02-24"	23.15	22.32	0.56	35	25.4	34.1	6997	"Undefined"

Poderia armazenar esses dados em um cache no Redis, mas, como se trata de previsão e ela pode mudar, não sei se seria uma boa abordagem. Por outro lado, o consumo sem critério desse recurso pode trazer degradação. O método também dá suporte às coordenadas geográficas, que pode mudar todo o contexto da previsão, então, se fosse consumida pelo front-end, o ideal seria considerar também a passagem desses dados.

De qualquer maneira, considerei as coordenadas de São José dos Campos, SP:

Longitude -45.9009

Latitude -23.2237

Para fins de comprovação da implementação do acesso à API da Embrapa, acrescentei uma tela que permite ver os dados, com base nas datas oferecidas pela plataforma (20 dias):

←→🔍localhost:44391/Home/Embrapa

📄Google Earth📍Flight Tracker | Fligh...📍Find your QTH locat...📡AMSAT - AMSAT On...📡(10) Space Station C...📡Radio DIXFUN Clust...📡EASP - ENTIDADES...📡Antena QHA/PWBP...📡ISSTracker - Real-TL...🌐https://earth.google...🔌Novo separador»📁Todos os favoritos

MVCPrototypeHomeCalendárioDados da Embrapa para os próximos dias

Dados da Embrapa para os próximos dias

Data	Temperatura Mínima	Temperatura na Superfície	Precipitação	Cobertura de Nuvens (Baixa)	Cobertura de Nuvens (Média)	Cobertura de Nuvens (Alta)	Duração do Sol	Condição do Tempo
2025-02-06	19.54 °C	19.09 °C	2.63 kg/m^2	0.1%	100%	2.6%	0 s	Chuvoso
2025-02-07	19.44 °C	18.68 °C	0.06 kg/m^2	2.8%	44.1%	0%	0 s	Nublado
2025-02-08	19.19 °C	18.75 °C	0 kg/m^2	0%	0.8%	2.8%	0 s	Nublado
2025-02-09	18.11 °C	17.54 °C	0 kg/m^2	6.2%	54%	0%	0 s	Nublado
2025-02-10	18.41 °C	18.02 °C	0 kg/m^2	0%	94.7%	0%	0 s	Chuvoso
2025-02-11	19.74 °C	19.45 °C	0 kg/m^2	0%	100%	0%	0 s	Chuvoso
2025-02-12	20.3 °C	19.69 °C	0 kg/m^2	0%	99.6%	0%	0 s	Chuvoso
2025-02-13	21.6 °C	21.52 °C	0.94 kg/m^2	100%	99.9%	39.5%	0 s	Chuvoso
2025-02-14	19.16 °C	18.63 °C	0 kg/m^2	0%	48.4%	25.4%	0 s	Nublado
2025-02-15	18.62 °C	18.25 °C	0.19 kg/m^2	7.4%	35.6%	12%	0 s	Nublado
2025-02-16	22.7 °C	22.35 °C	1.75 kg/m^2	22.6%	2%	0%	13200 s	Nublado
2025-02-17	25.37 °C	24.85 °C	0.06 kg/m^2	99.7%	38.2%	32.8%	5346 s	Chuvoso
2025-02-18	24.21 °C	23.35 °C	0 kg/m^2	53.9%	0%	8.4%	12160 s	Nublado
2025-02-19	23.06 °C	23.16 °C	0.81 kg/m^2	58.6%	0%	99.4%	920 s	Chuvoso
2025-02-20	23.17 °C	22.91 °C	0.75 kg/m^2	41%	0.8%	32.7%	8943 s	Nublado
2025-02-21	21.91 °C	21.82 °C	4.31 kg/m^2	35.4%	100%	25.2%	7261 s	Chuvoso
2025-02-22	22.08 °C	21.65 °C	1 kg/m^2	13.7%	85.7%	100%	1273 s	Chuvoso
2025-02-23	20.65 °C	19.97 °C	0.06 kg/m^2	6.8%	70.2%	91.9%	12896 s	Chuvoso
2025-02-24	22.04 °C	21.99 °C	3.06 kg/m^2	50.3%	83.2%	5.1%	12783 s	Chuvoso
2025-02-25	21.83 °C	21.07 °C	2.25 kg/m^2	0.4%	91.6%	18.7%	12750 s	Chuvoso