



Criando uma Minimal API para serviços de Geolocalização com SQL + C#

Trilha Engenharia de Software

Márcio Rogério Nizzola

Márcio R. Nizzola

Tech Lead na CI&T, com foco em .NET.

Formado em Análise de Sistemas e MBA em Gestão de Projetos.

Professor na Etec de Itu desde 2008, atuando nos cursos técnicos da área de tecnologia, ministrando disciplinas de programação, bancos de dados e projetos de TCC.

Desenvolvendo software desde 1992.

Membro fundador da comunidade Itu Developers.



ASSUNTO PRINCIPAL

Criar uma API de Geo Localização com .NET

Com o avanço das aplicações na atualidade, precisamos realizar buscas identificando locais ou pessoas através do seu posicionamento geográfico.

Isto intensificou-se mais ainda com o uso da aplicações Mobile, onde todos os celulares já dispõem de GPS embutido.

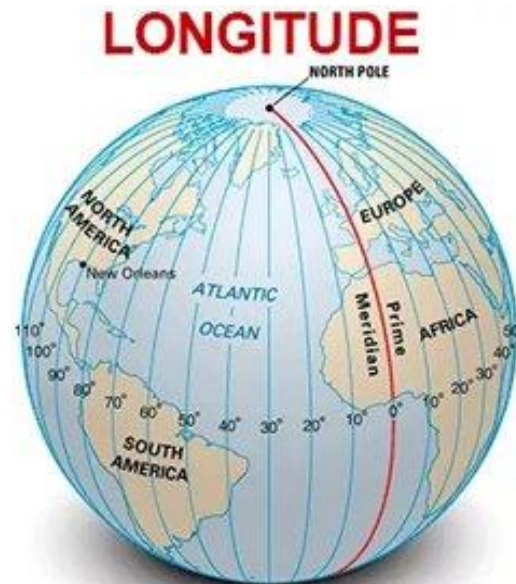
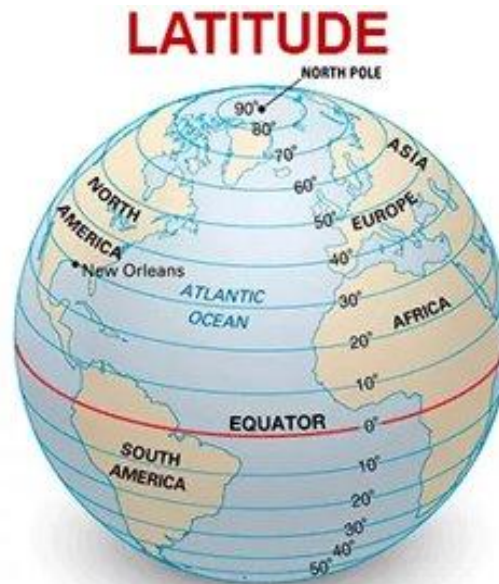
Com isto passamos a ter listas de lugares e pessoas, e precisamos identificar pontos de interesse próximos à estas posições geográficas.

Existem API'S prontas para tratar disso, porém seu acesso em grande escala incide em custos.

Exemplo: Apps de Hotéis, Comida, Entregas, Motoristas dentre outros.



Vamos
entender o
contexto da
Geo
Localização



Temos o posicionamento de qualquer ponto no mundo determinado através de medidas em graus.

Onde armazenar os dados



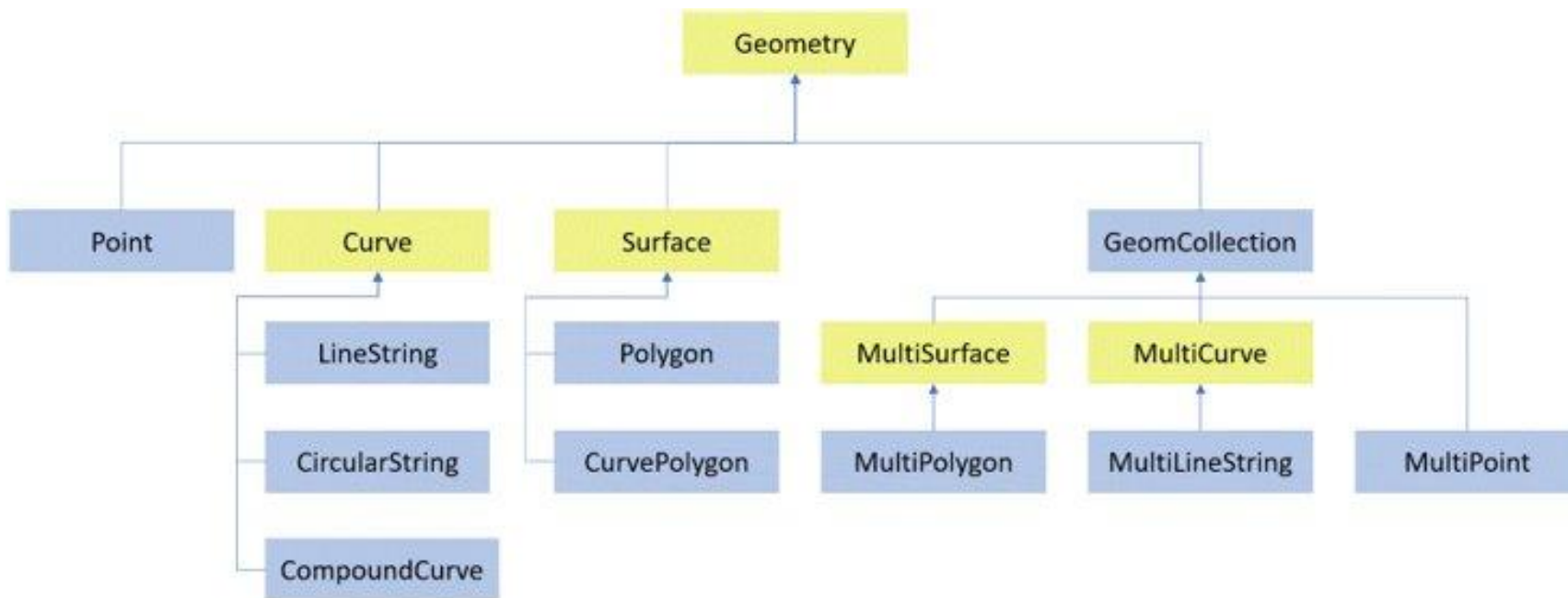
Como fazer para consultar um determinado ponto no mapa utilizando a tecnologia disponível ?

O Sql possui dois tipos de dados que poucos conhecem:

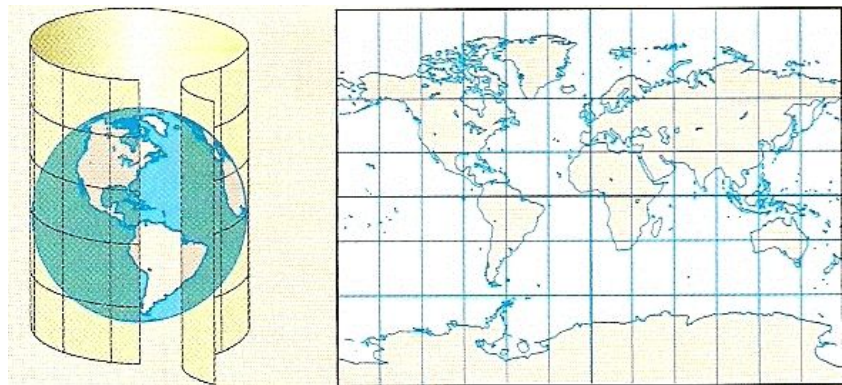
Geometry = utilizada para dados Geométricos

Geography = utilizada para dados Geográficos, é derivada de Geometry, porém é adaptada para a projeção cartográfica, pois a curvatura da terra é considerada em suas projeções.

Os tipos de dados **geometry** e **geography** dão suporte a 16 tipos de objetos de dados espaciais



O que é o SRID



Cada forma geométrica possui um sistema de referência espacial associado a ela, e cada um desses sistemas de referência possui um ID do Sistema de Referência Espacial para informar qual sistema será usado para interpretar cada objeto espacial.

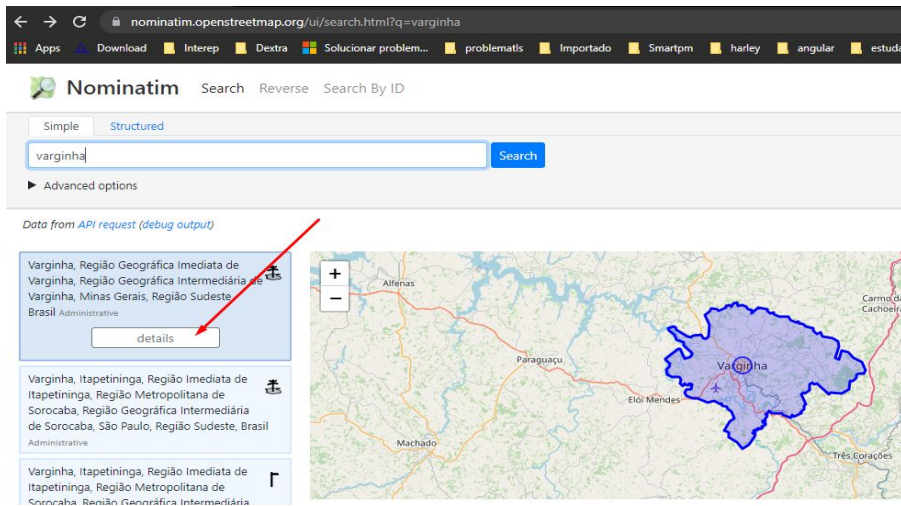
Um SRID comum em uso é 4326, que representa dados espaciais usando coordenadas de longitude e latitude na superfície da Terra conforme definido no padrão WGS84 , que também é usado para o Sistema de Posicionamento Global (GPS)

Principais métodos de Geography Sql usados

Métodos	Funcionalidade
Point()	Cria um ponto geo baseado nas coordenadas Ex: POINT(-23.17996298 ,-47.302378099)
STPointFromText()	Cria um ponto geo através de string Ex: STPointFromText('POINT(-47.302378099 -23.17996298)', 4326)
STGeomFromText()	Cria um objeto geo através de string Ex: STGeomFromText('POINT(-23.2159214 -47.26859020000001)',4326)
STIntersects()	Verifica se um ponto está contido em um objeto geo
MakeValid()	Converte uma instância de geometry inválida num tipo válido de OGC
STIsValid()	Retorna se o objeto do tipo geography é válido
STAsText ()	Retorna em string um objeto Geo do banco de dados
STBuffer()	Cria uma circunferência baseada na distância em metros de um ponto
.STDistance ()	Retorna a distância linear entre dois pontos Geography

Onde podemos obter os polígonos ?

Existe a base de dados do OpenStreet Map que possui polígonos de localização de todo o planeta e pode ser baixada, porém para o nosso exemplo vamos fazer de forma manual buscando apenas demonstrar o conceito de Geo Localização.



Nominatim Search Reverse Search By ID

Simple Structured

varginha Search

Advanced options

Data from API request (debug output)

Varginha, Região Geográfica Imediata de Varginha, Região Geográfica Intermediária de Varginha, Minas Gerais, Região Sudeste, Brasil

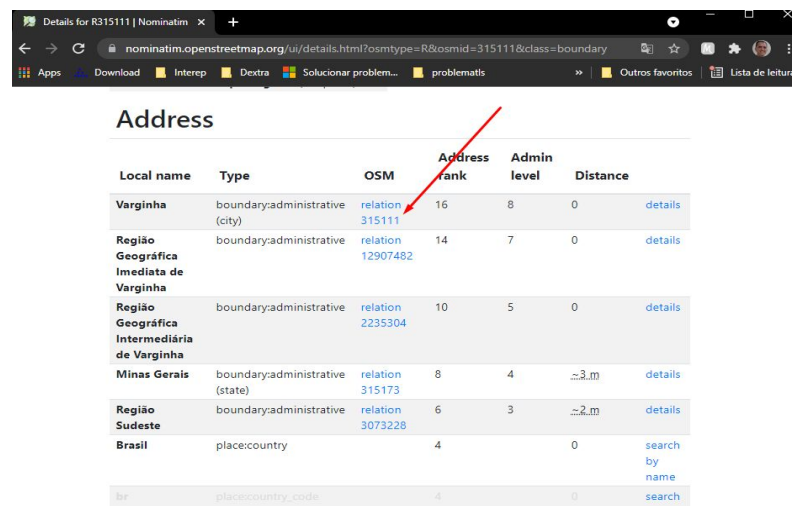
details

Varginha, Itapetininga, Região Imediata de Itapetininga, Região Metropolitana de Sorocaba, Região Geográfica Intermediária de Sorocaba, São Paulo, Região Sudeste, Brasil

Administrative

Varginha, Itapetininga, Região Imediata de Itapetininga, Região Metropolitana de Sorocaba, Região Geográfica Intermediária de Sorocaba, São Paulo, Região Sudeste, Brasil

**Acessando o
site Nominatim buscando pelo nome**



Details for R315111 | Nominatim

nominatim.openstreetmap.org/ui/details.html?osmtype=R&osmid=315111&class=boundary

Address

Local name	Type	OSM	Address rank	Admin level	Distance	
Varginha	boundary:administrative (city)	relation 315111	16	8	0	details
Região Geográfica Imediata de Varginha	boundary:administrative	relation 12907482	14	7	0	details
Região Geográfica Intermediária de Varginha	boundary:administrative	relation 2235304	10	5	0	details
Minas Gerais	boundary:administrative (state)	relation 315173	8	4	~3.0m	details
Região Sudeste	boundary:administrative	relation 3073228	6	3	~2.0m	details
Brasil	place:country		4		0	search by name
br	place:country_code		4		0	search

Depois obtendo o número da localidade

Onde podemos obter os polígonos ?

Acessando o endereço:

<https://polygons.openstreetmap.fr/index.py?id=315111>

<https://polygons.openstreetmap.fr/index.py?id=315111>

List of available polygons for id = 315111

params	timestamp	NPoints	Length	WKT	GeoJSON	poly	Image
0	2021-06-08 15:54:06 492803	1541	0.302630299359633	WKT	GeoJSON	poly	Image

[Refresh original geometry](#)

Generate a simplified polygon

X, Y, Z are parameters for the following postgis equation. The default values are chosen according to the size of the original geometry to give a slightly bigger geometry, without too many nodes.

Note that:

- $X > 0$ will give a polygon bigger than the original geometry, and guaranteed to contain it.
- $X = 0$ will give a polygon similar to the original geometry.
- $X < 0$ will give a polygon smaller than the original geometry, and guaranteed to be smaller.

X 0.004000
Y 0.001000
Z 0.001000

SQL requests:

- $X > 0$: `ST_Union(ST_MakeValid(ST_SimplifyPreserveTopology(geom, 0.00001)), ST_Buffer(ST_SimplifyPreserveTopology(ST_SnapToGrid(ST_Buffer(geom, X), Y), Z)))`
- $X = 0$: `ST_Buffer(ST_SimplifyPreserveTopology(ST_SnapToGrid(ST_Buffer(geom, X), Y), Z))`
- $X < 0$: `ST_Intersection(geom, ST_Buffer(ST_SimplifyPreserveTopology(ST_SnapToGrid(ST_Buffer(geom, X), Y), Z))`

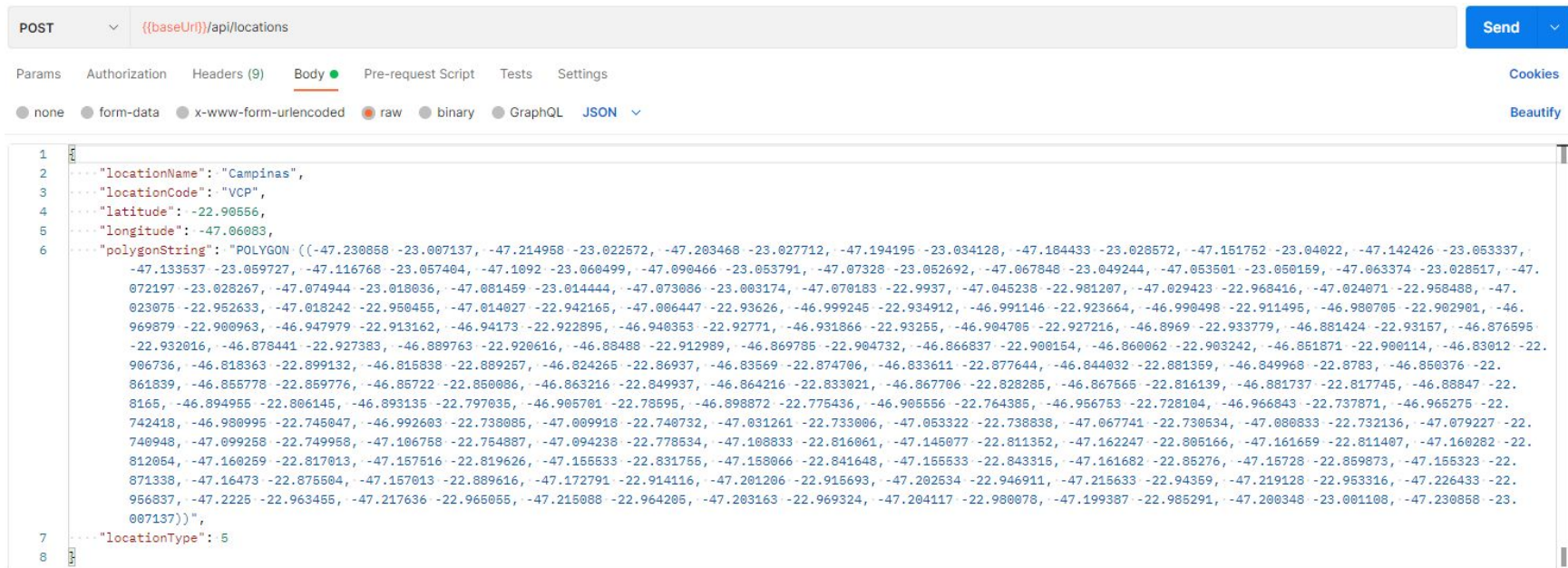
Agora temos o polígono aqui para ser inserido no Sql Server

https://polygons.openstreetmap.fr/get_wkt.py?id=315111¶ms=0

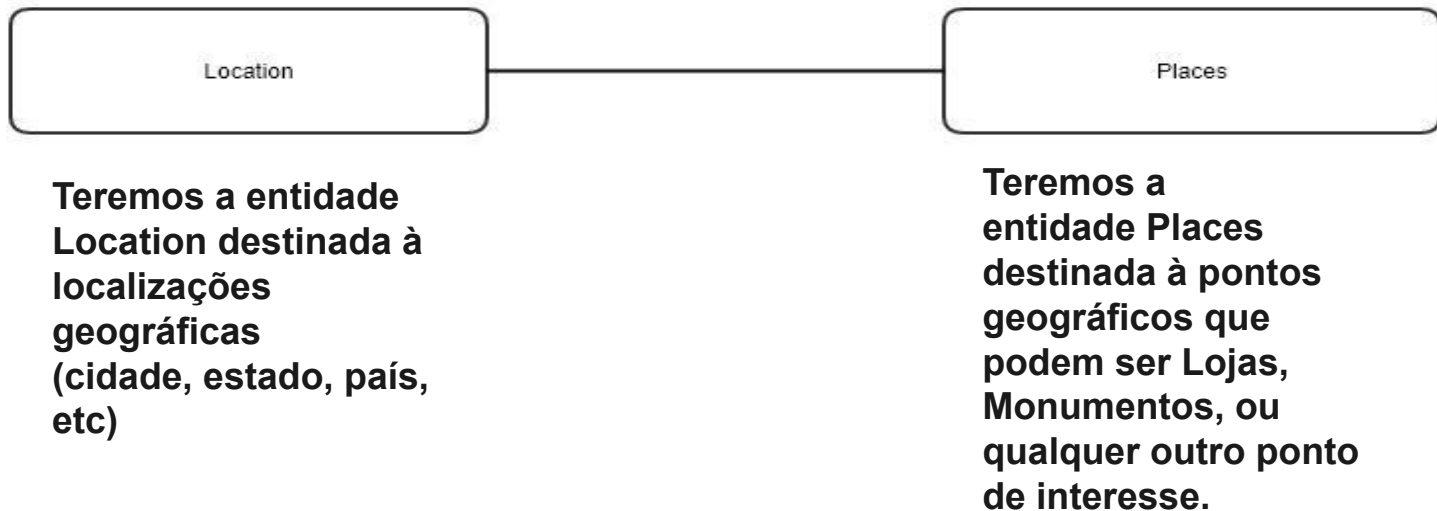
```
SRID=4326;MULTIPOLYGON((-45.5455844 -21.4985279,-45.5429562 -21.4976699,-45.5405328 -21.4969107,-45.5373697
-21.4971147,-45.5345592 -21.4980421,-45.5339611 -21.4991579,-45.5336341 -21.5000753,-45.5333688 -21.5011646,-45.5318217
-21.5022769,-45.5306747 -21.5035152,-45.5287054 -21.5047249,-45.5273141 -21.5052388,-45.5254051 -21.505453,-45.5221481
-21.5086119,-45.5202678 -21.509608,-45.5183382 -21.5101454,-45.5161375 -21.5113135,-45.5128275 -21.5127911,-45.5110019
-21.5126772,-45.507957 -21.5107055,-45.5060155 -21.5100063,-45.5016214 -21.5091351,-45.4979085 -21.5088305,-45.495983
-21.5097063,-45.4943779 -21.5117976,-45.4937771 -21.5136137,-45.492346 -21.5147969,-45.4906723 -21.5152968,-45.4885054
-21.5153495,-45.4869447 -21.5146862,-45.486 -21.514,-45.485064 -21.5116217,-45.4844484 -21.5086177,-45.4842709
-21.5068118,-45.4837482 -21.505304,-45.4838119 -21.5056414,-45.4843507 -21.5028186,-45.4851581 -21.5019295,-45.4853577
-21.5015692,-45.4853753 -21.4999377,-45.4845123 -21.4990949,-45.4840993 -21.4970935,-45.4839057 -21.4940207,-45.4848644
-21.4914229,-45.4874376 -21.4879519,-45.489 -21.4851629,-45.4892107 -21.4829865,-45.4891748 -21.4814064,-45.4896956
-21.4785049,-45.4885973 -21.4773409,-45.4860763 -21.4764951,-45.4844426 -21.4751217,-45.4838746 -21.4734554,-45.483869
-21.4716973,-45.4830096 -21.4703904,-45.4825761 -21.4695045,-45.4824762 -21.4688225,-45.4818033 -21.4687827,-45.4811865
-21.4684647,-45.481022 -21.4683007,-45.4808436 -21.4681826,-45.4802525 -21.4682496,-45.4793527 -21.4680515,-45.4786749
-21.4680119,-45.4782222 -21.4685369,-45.4776721 -21.4686683,-45.4770264 -21.4691621,-45.4760395 -21.4694362,-45.4751166
-21.4693866,-45.4740627 -21.4697544,-45.4727814 -21.4702734,-45.4724188 -21.4702704,-45.4721371 -21.4704512,-45.4719855
-21.4709044,-45.4717064 -21.470904,-45.4713785 -21.4707372,-45.4709343 -21.4707666,-45.4706614 -21.4718145,-45.4705675
-21.471565,-45.4702075 -21.4718705,-45.4698475 -21.4719186,-45.4698081 -21.4716568,-45.4688522 -21.4712837,-45.4687622
-21.4709004,-45.4678266 -21.4707592,-45.466662 -21.4710263,-45.4656504 -21.4716075,-45.4651433 -21.4715258,-45.4647411
-21.4711564,-45.4647088 -21.4706722,-45.4644886 -21.4704415,-45.4639218 -21.4701913,-45.463551 -21.4698006,-45.4622845
-21.4678257,-45.4621246 -21.4667023,-45.4615684 -21.4666328,-45.4603993 -21.4669154,-45.459222 -21.4666902,-45.4587667
-21.4662015,-45.4588222 -21.4648241,-45.458 -21.464,-45.4568232 -21.4645847,-45.4562419 -21.4646805,-45.455618
-21.4646159,-45.4535192 -21.4637826,-45.4524927 -21.4639866,-45.4516004 -21.4645081,-45.4505666 -21.4652112,-45.449816
-21.4651054,-45.4490436 -21.4646355,-45.4483318 -21.463241,-45.4477899 -21.4623473,-45.4481005 -21.4613107,-45.4475622
-21.4606211,-45.4467409 -21.4606069,-45.4459621 -21.4606046,-45.4451551 -21.4603033,-45.4434353 -21.4593162,-45.4443613
-21.4598359,-45.4420624 -21.4612251,-45.4420592 -21.4625541,-45.4409282 -21.4635521,-45.4404108 -21.4645999,-45.4396322
-21.4647403,-45.4392996 -21.4643762,-45.4392607 -21.463733,-45.438944 -21.4636618,-45.4385281 -21.4641175,-45.4375375
-21.4634444,-45.4367456 -21.463689,-45.4356139 -21.4634265,-45.4342977 -21.4632474,-45.4335565 -21.463764,-45.4329991
-21.4640881,-45.4317446 -21.4643292,-45.4312827 -21.4651501,-45.4304618 -21.4656686,-45.4301006 -21.4659623,-45.4297115
-21.465921,-45.4289279 -21.4655127,-45.4285194 -21.4655448,-45.4281638 -21.4656524,-45.4279384 -21.4657925,-45.4278135
-21.46461031,-45.4275284 -21.4668955,-45.4270337 -21.4673552,-45.4265867 -21.467471,-45.4262215 -21.4679194,-45.4255969
-21.4678738,-45.4253218 -21.4680922,-45.4251409 -21.4685854,-45.4251087 -21.4690946,-45.4254949 -21.4698933,-45.4252995
```

Como importar os polígonos ?

Agora é só executar a chamada da API pelo Postman (quando estiver pronta a API)



Como vai ficar nosso projeto ?



Como será feita a API !

Os tipos de dados geográficos do Sql Server faziam parte do Entity Framework no passado, mas com o lançamento do Entity Framework Core, deixaram de serem suportados.

Por conta disto, usaremos o Dapper para a realização do acesso ao banco de dados.

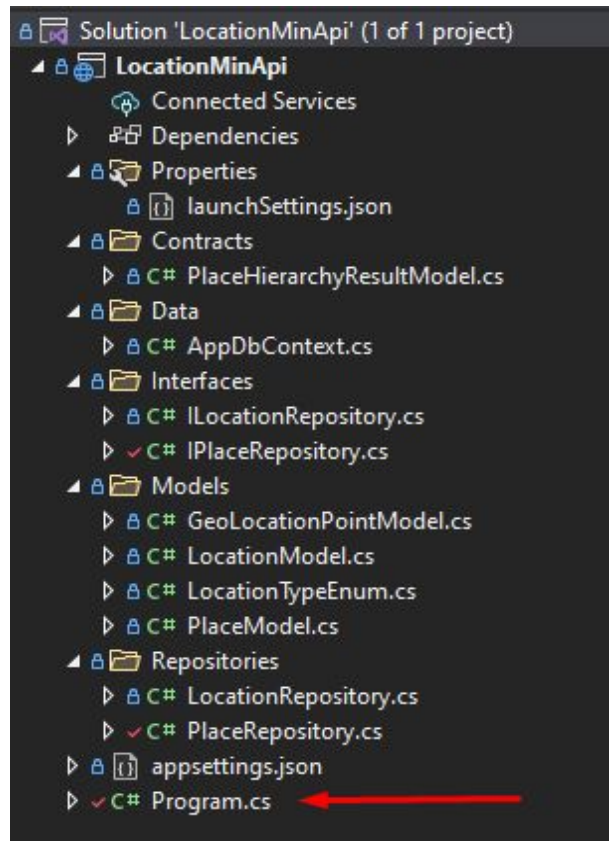


Dapper

a simple object mapper for .Net

Como será feita a API !

Na versão do .NET 6 foi disponibilizada a funcionalidade chamada de "Minimal Api" que permite a criação de APIs com o mínimo de dependência do framework WebAPI e a criação de código muito mais simples e compacto.



**Ao invés de controllers,
escrevemos nossos endpoints
no program.cs**

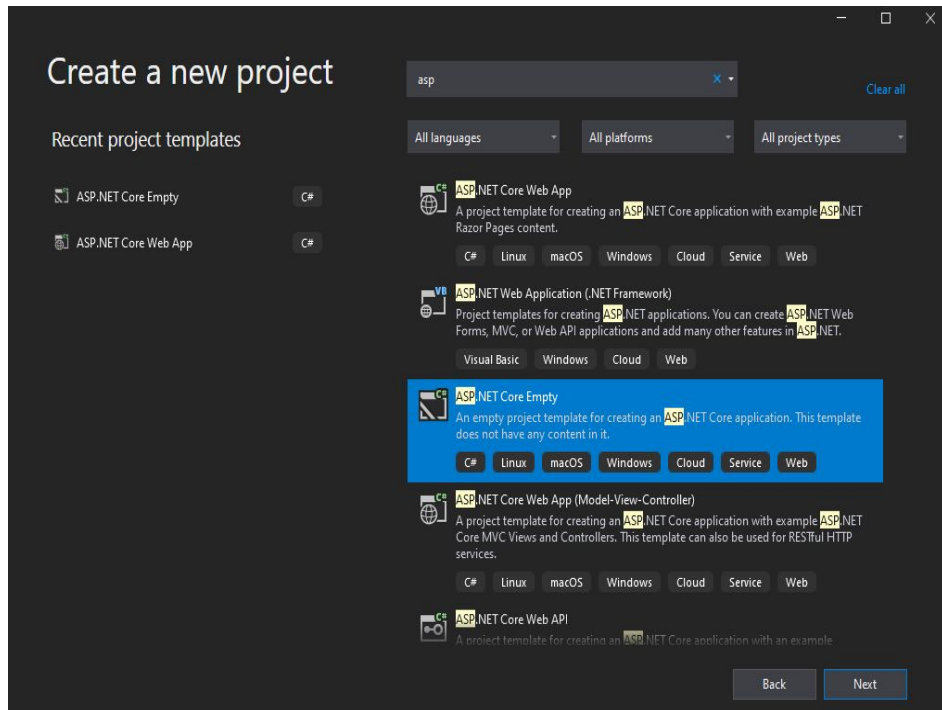
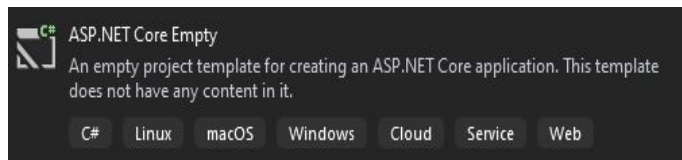
Como será feita a API !

Nesta modalidade de aplicação, nós utilizamos o template "AspNet Core Empty"

Através do comando:

dotnet new web -o MeuProjeto

Ou diretamente pelo visual studio



Observação: este tipo de projeto somente está disponível no Visual Studio 2022.

Como será feita a API !

Mas o que há de diferente então ?

Dentro do Program.cs colocamos os métodos que iremos utilizar.

Com os métodos:

MapGet

MapPost

MapPut

MapDelete

```
1  using System;
2  using Microsoft.AspNetCore.Builder;
3  using Microsoft.Extensions.Hosting;
4
5  var builder = WebApplication.CreateBuilder(args);
6  await using var app = builder.Build();
7
8  if (app.Environment.IsDevelopment())
9  {
10     app.UseDeveloperExceptionPage();
11 }
12
13 app.MapGet("/", (Func<string>)(() => "Hello World!"));
14 await app.RunAsync();
15
```


Como vai ficar nosso projeto ?

Teremos métodos GET seguindo este formato da Minimal Api

```
app.MapGet("api/locations", async (ILocationRepository _locationRepository) =>
{
    var result = await _locationRepository.GetAll();
    if (result == null)
        return Results.NotFound();

    return Results.Ok(result);
});
```

Teremos métodos POST seguindo este formato

```
app.MapPost("api/locations", async (ILocationRepository _locationRepository, LocationModel location) =>
{
    if (location == null)
        return Results.BadRequest();

    var result = await _locationRepository.Add(location);

    return Results.Ok(result);
});
```

Como importar os polígonos ?

Precisamos criar uma tabela no banco de dados e uma procedure que fará a inserção da localização no banco de dados.

dbo.Location
Colunas
LocationId (PK, int, não nulo)
LocationName (nvarchar(60), nulo)
LocationCode (nvarchar(5), nulo)
LocationPoint (geography, nulo)
Polygon (geography, nulo)
LocationType (int, não nulo)
Latitude (decimal(15,9), nulo)
Longitude (decimal(15,9), nulo)

Os dois campos destacados
são criados para armazenar
o tipo de dado "geography"

```
CREATE PROCEDURE [dbo].[LocationInsert]

    @Lat decimal(15,9),
    @Long decimal(15,9),
    @GeoMultiPoly varchar(max),
    @PlaceName varchar(max),
    @PlaceCode varchar(3) = '',
    @PlaceType integer

AS BEGIN

    SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

    DECLARE @g geography = geography::Point( @Lat, @Long, 4326);
    DECLARE @pol geography = geography::STGeomFromText(@GeoMultiPoly, 4326);

    INSERT INTO LOCATION ( LocationName, LocationCode, LocationPoint, Polygon,
        LocationType , Latitude, Longitude)
        values ( @PlaceName, @PlaceCode, @g, @pol , @PlaceType , @Lat, @Long );

    SELECT TOP 1 * FROM LOCATION WHERE LOCATIONID = ( SELECT MAX(LocationId) FROM LOCATION );

END;
GO
```

Como importar os locais ?

Precisamos criar uma tabela no banco de dados e uma procedure que fará a inserção dos lugares.

dbo.Places
Colunas
PlaceId (PK, int, não nulo)
Name (nvarchar(max), nulo)
Address (nvarchar(max), nulo)
PhoneNumber (nvarchar(max), nulo)
Snippet (nvarchar(max), nulo)
Description (nvarchar(max), nulo)
StyleUrl (nvarchar(max), nulo)
Latitude (float, não nulo)
Longitude (float, não nulo)
LocationPoint (geography, nulo)
Id (nvarchar(max), nulo)

O campo marcado será utilizado para armazenar o tipo de dado "geography"

```
CREATE PROCEDURE [dbo].[PlaceInsert]

    @Lat decimal(15,9),
    @Long decimal(15,9),
    @Name varchar(max),
    @Address varchar(max),
    @Description varchar(max) = '',
    @PhoneNumber varchar(15),
    @Snippet varchar(max) = '',
    @styleUrl varchar(max) = '',
    @Id varchar(15) = ''

AS BEGIN

    SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

    DECLARE @point geography =
        geography::STPointFromText('POINT(' + CAST(@Lat AS VARCHAR(20))
            + ' ' + CAST(@Long AS VARCHAR(20)) + ')', 4326);

    INSERT INTO [dbo].[Places]
        ([Name] ,[Address] ,[PhoneNumber] ,[Snippet] ,[Description]
        ,[StyleUrl] ,[Latitude] ,[Longitude] ,[LocationPoint]
        ,[Id])
    VALUES
        ( @Name, @Address, @PhoneNumber, @Snippet, @Description,
        @styleUrl, @Lat, @Long, @point, @Id )

    SELECT TOP 1 * FROM [DBO].[Places] where PlaceId =(SELECT MAX(PlaceId) FROM [dbo].[Places]));

END;
GO
```

Como importar os polígonos ?

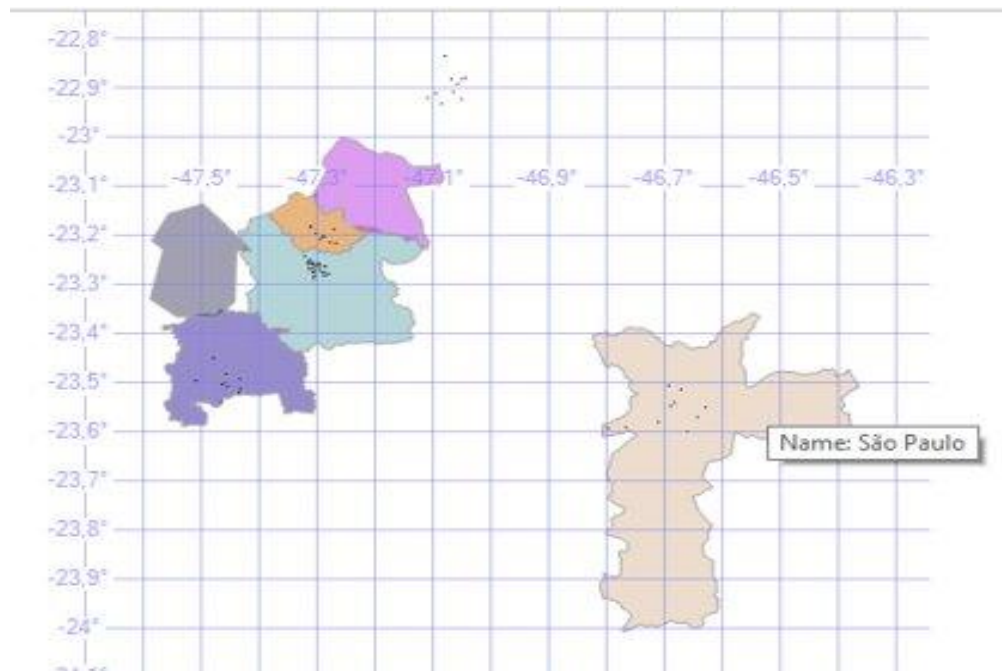
Depois foi só criar o método para realizar a inclusão utilizando-se do Dapper

```
public async Task<LocationModel> Add(LocationModel model)
{
    using (IDbConnection db = new SqlConnection(_connectionString))
    {
        var procedure = "LocationInsert";
        var values = new
        {
            @Lat = model.LocationPointValues.Latitude,
            @Long = model.LocationPointValues.Longitude,
            @GeoMultiPoly = model.MultiPolygonString,
            @PlaceName = model.LocationName,
            @PlaceCode = model.LocationCode,
            @PlaceType = (int)model.LocationType
        };

        var results = await db.QueryAsync<LocationModel>(procedure, values, commandType: CommandType.StoredProcedure);
        return results.FirstOrDefault();
    }
}
```

Mesclando Pontos do Mapa x Polígonos

```
SELECT  
  geography::Point( Latitude, Longitude, 4326) LocationPoint,  
  Name  
FROM PLACES  
union all  
select Polygon,  
  locationName  
from location where LocationType=5
```



Podemos selecionar através de um raio a região desejada usando STBuffer

```
DECLARE @g geography, @circle geography
```

```
SET @g = geography::Point(-23.2159214 ,-47.26859020000001,  
4326);
```

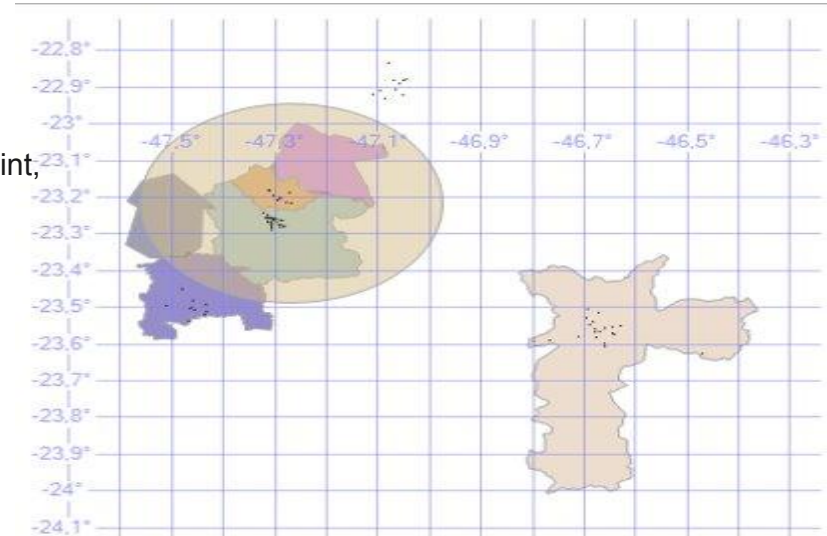
```
SET @circle=@g.STBuffer(30000);
```

```
SELECT geography::Point( Latitude, Longitude, 4326) LocationPoint,  
Name FROM PLACES  
union all
```

```
SELECT Polygon, locationName  
FROM location WHERE LocationType=5
```

```
union all
```

```
SELECT  
    @circle,  
    'regiao' name
```



Como fazer estas consultas então ?

Criaremos procedures no SQL

Então agora podemos montar na API um método que busque os pontos por coordenadas dentro do mapa.

Utilizando:

STIntersects

```
CREATE PROCEDURE [dbo].[WhatLocationIs]
    @Lat decimal(15,9),
    @Long decimal(15,9),
    @LocationType integer = 0
AS BEGIN
    SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

    declare @point geography = geography::Point(@Lat,@Long, 4326);

    SELECT *
    FROM (
        select
            lo.LocationId,
            lo.LocationName,
            lo.Polygon.MakeValid() Polygon,
            lo.LocationType,
            lo.latitude,
            lo.longitude

        from Location lo
        where ( @LocationType = 0 or lo.LocationType = @LocationType )
    ) A
    where A.Polygon.STIntersects(@point) = 1

END;
GO
```


Como fazer estas consultas então ?

Dentro da aplicação faremos a chamada à Procedure garantindo assim que se obtenha a hierarquia de localização de um ponto no mapa.

Utilizando as consultas com Dapper.

```
public async Task<ICollection<LocationModel>> GetFromGeo(double latitude, double longitude)
{
    using (IDbConnection connection = new SqlConnection(_connectionString))
    {
        var procedure = "WhatLocationIs";
        var values = new { @Lat = latitude, @Long = longitude };
        var results = await connection.QueryAsync<LocationModel>(procedure, values, commandType: CommandType.StoredProcedure);
        return results.ToList();
    }
}
```


Agora será só testar no Postman

Ao executar o endpoint passando latitude e longitude, é possível identificar onde este ponto está inserido dentro de uma hierarquia geográfica.

The screenshot shows the Postman interface for a GET request to the endpoint `/api/locations/geo/latitude/longitude`. The request is configured with path variables for `latitude` and `longitude`. The response is a JSON array of location objects.

Query Params

KEY	VALUE
Key	Value

Path Variables

KEY	VALUE
latitude	-23.30886648
longitude	-47.285233955

Body | Cookies | Headers (4) | Test Results

Pretty | Raw | Preview | Visualize | JSON |

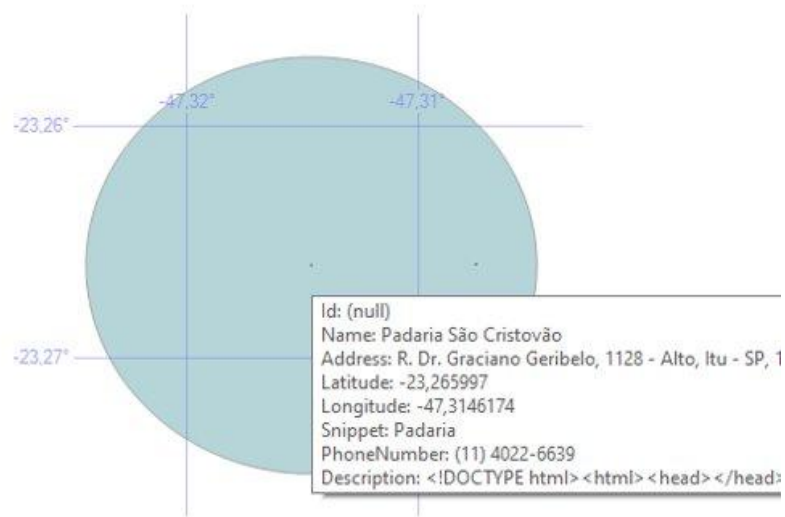
```
1  [
2    {
3      "locationId": 1,
4      "locationName": "Itu",
5      "locationCode": null,
6      "locationType": 6,
7      "polygonString": null,
8      "latitude": -23.308866487,
9      "longitude": -47.285233955
10   },
11   {
12     "locationId": 6,
13     "locationName": "São Paulo",
14     "locationCode": null,
15     "locationType": 3,
16     "polygonString": null,
17     "latitude": -23.5506507,
18     "longitude": -46.6333824
19   },
20   {
21     "locationId": 7,
22     "locationName": "Brasil",
23     "locationCode": null,
24     "locationType": 2,
```

E se quisermos procurar locais num raio ?

É possível !!

```
DECLARE @point geography =  
geography::Point(@Lat,@Long, 4326);  
DECLARE @Circle geography = @point.STBuffer(@Ray);
```

```
SELECT *  
FROM (  
SELECT  
    PL.*  
FROM Places pl  
WHERE ( @PlaceType = " or @PlaceType = pl.Snippet )  
A  
WHERE @circle.STIntersects(a.LocationPoint) = 1
```



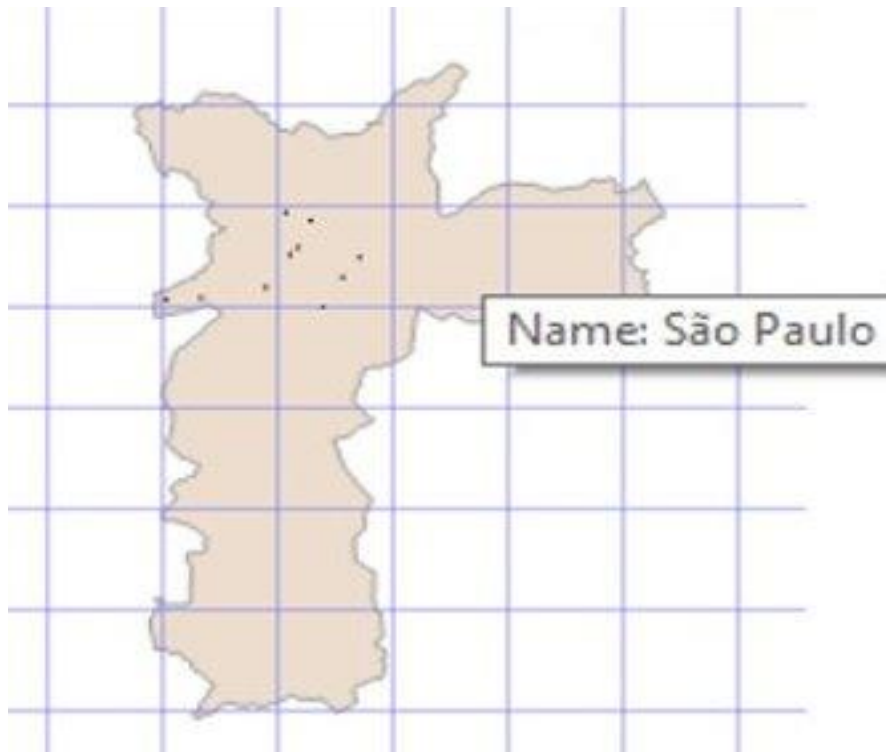
E se quisermos dentro da cidade específica?

Só isso ??

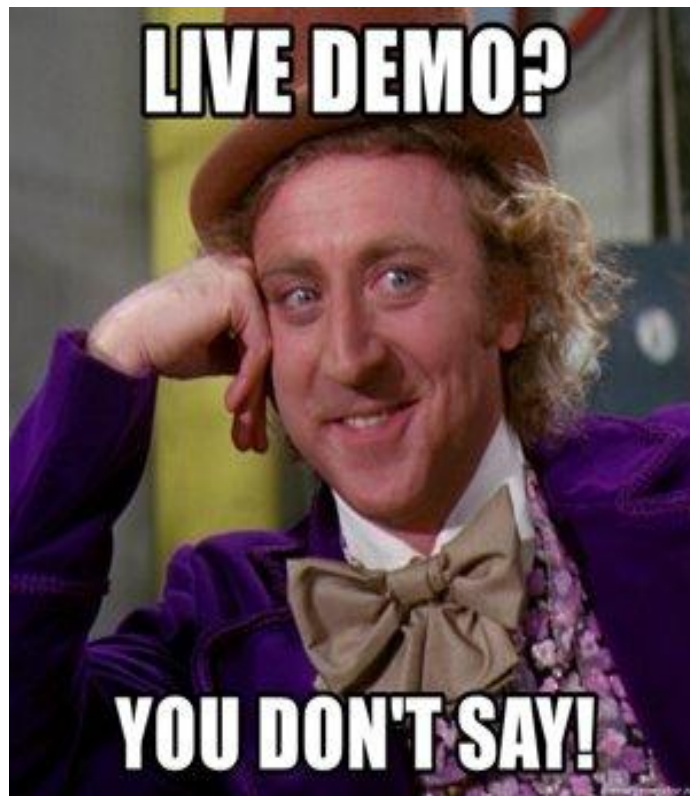
```
DECLARE @poly geography;
```

```
SELECT @poly = Polygon  
FROM Location  
WHERE LocationId=@LocationId;
```

```
SELECT *  
FROM (  
  SELECT  
    PL.*  
    FROM Places pl  
    WHERE ( @PlaceType = "  
            or @PlaceType = pl.Snippet )  
  ) A  
WHERE  
@poly.STIntersects(a.LocationPoint) = 1
```



Chegou a esperada hora !!!



Referências minimal API

Site Oficial Microsoft

[Tutorial: Create a minimal web API with ASP.NET Core | Microsoft Docs](#)

Publicação Balta.IO

<https://balta.io/blog/aspnet-minimal-apis>

Publicação Renato Groffe

<https://renatogroffe.medium.com/novidades-do-net-6-suporte-a-swagger-em-minimal-apis-8ace47739028>

Publicação Macoratti

http://www.macoratti.net/21/09/aspn6_minapi1.htm

Sites diversos

<https://benfoster.io/blog/mvc-to-minimal-apis-aspnet-6/>

Low Ceremony, High Value: A Tour of Minimal APIs in .NET 6 (telerik.com)

<https://anthonygiretti.com/2021/08/12/asp-net-core-6-working-with-minimal-apis/>

Para obter locais geográficos

<https://nominatim.openstreetmap.org/ui/details.html?osmtype=R&osmid=298285&class=boundary>

Para obter o polígono

<https://polygons.openstreetmap.fr/index.py?id=298285>

Quer saber sobre Dapper ? (minha palestra)

https://www.youtube.com/watch?v=N6CH_v7YukA

Referências Sql Server

Tipo : Point

<https://docs.microsoft.com/pt-br/sql/t-sql/spatial-geography/point-geography-data-type?view=sql-server-ver15>

Tipo : Polygon

<https://docs.microsoft.com/pt-br/sql/relational-databases/spatial/polygon?view=sql-server-ver15>

Multipolygon

<https://docs.microsoft.com/pt-br/sql/relational-databases/spatial/multipolygon?view=sql-server-ver15>

STIsValid

<https://docs.microsoft.com/pt-br/sql/t-sql/spatial-geography/stisvalid-geography-data-type?view=sql-server-ver15>

MakeValid

<https://docs.microsoft.com/pt-br/sql/t-sql/spatial-geography/makevalid-geography-data-type?view=sql-server-ver15>

STIntersects

<https://docs.microsoft.com/pt-br/sql/t-sql/spatial-geography/stintersects-geography-data-type?view=sql-server-ver15>

STDistance

<https://docs.microsoft.com/pt-br/sql/t-sql/spatial-geography/stdistance-geography-data-type?view=sql-server-ver15>

STBuffer

<https://docs.microsoft.com/pt-br/sql/t-sql/spatial-geography/stbuffer-geography-data-type?view=sql-server-ver15>

StUnion

<https://docs.microsoft.com/pt-br/sql/t-sql/spatial-geometry/stunion-geometry-data-type?view=sql-server-ver15>

CONTATOS

Marcio R Nizzola

Repositório da palestra:

[Github.com/nizzola/TDCFutur2021](https://github.com/nizzola/TDCFutur2021)

Meus contatos: linktr.ee/NIZZOLA



DÚVIDAS ?