

Entity Framework CORE



dicas e instruções

FRAMEWORK



É DE BEBER OU DE COMER?

WWW.GERARMEUS.COM.BR

UM CONJUNTO DE...



CÓDIGOS E PADRÕES QUE
DEVEMOS SEGUIR



FRAMEWORKS ORM



**MAPEIAM AS TABELAS COM
AS CLASSES**

WWW.GERARMEMES.COM.BR

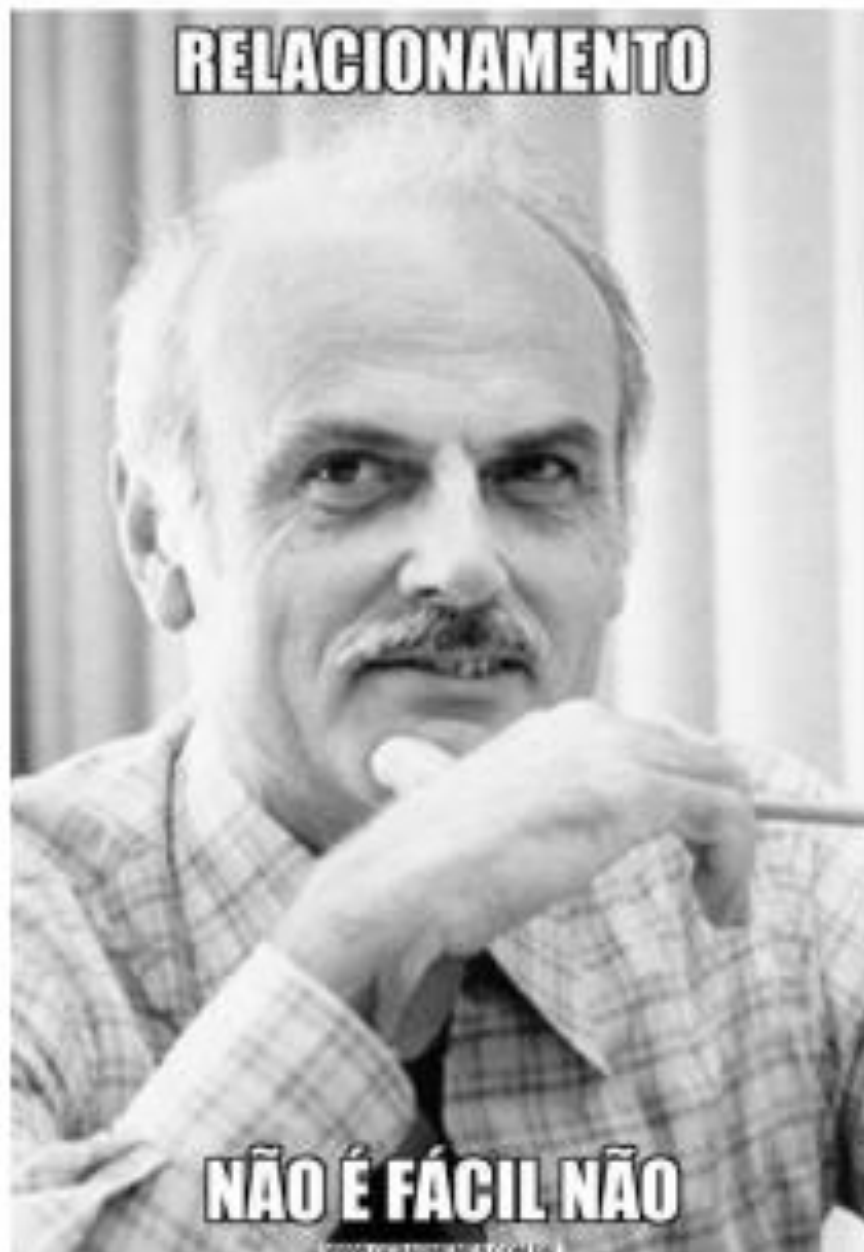
Edgar Frank Codd

modelo relacional

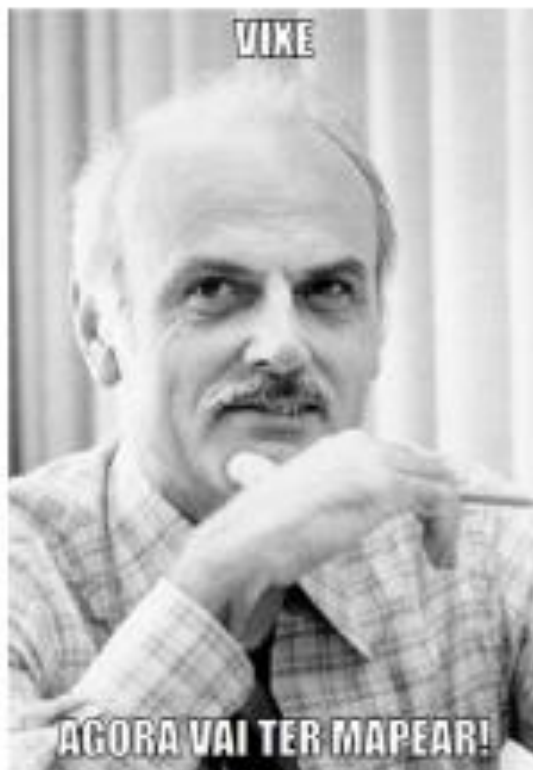
Nascimento	23 de agosto de 1923 Dorset
Morte	18 de abril de 2003 (79 anos) Flórida
Nacionalidade	britânico
Cidadania	Reino Unido, Estados Unidos, Hello World
<i>Alma mater</i>	Exeter College Universidade de Michigan Poole Grammar School
Ocupação	cientista de computação, matemático, professor(a) universitário(a)
Prêmios	Prêmio Turing (1981), Prêmio Pioneiro da Computação (1996)
Empregador	IBM
Obras destacadas	Autômato Celular de Codd, A relational model of data for large shared data banks
Causa da morte	enfarte agudo do miocárdio

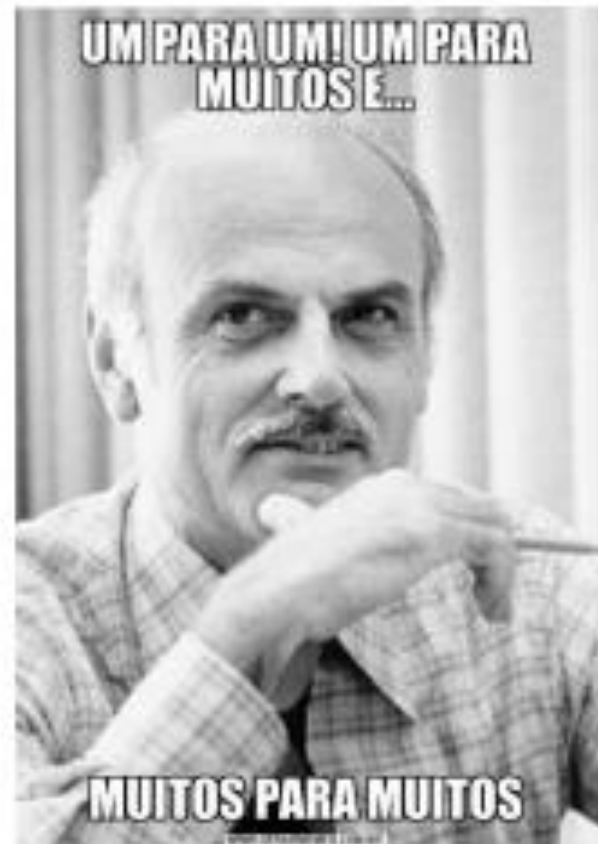
[[edite no Wikidata](#)]

RELACIONAMENTO



NÃO É FÁCIL NÃO





Sabia que o Edgar Frank Codd nasceu em 1923 e criou o modelo que vamos mapear com o professor Rômulo?

Sério! Caracas!

Que foi mano!

Tive uma visão do cara do passado vindo para o presente!

E aí?

Acho que vamos ter que mapear alguma coisa!



Ei aquela parada lá da Visão?

Cara nem me lembre disso!

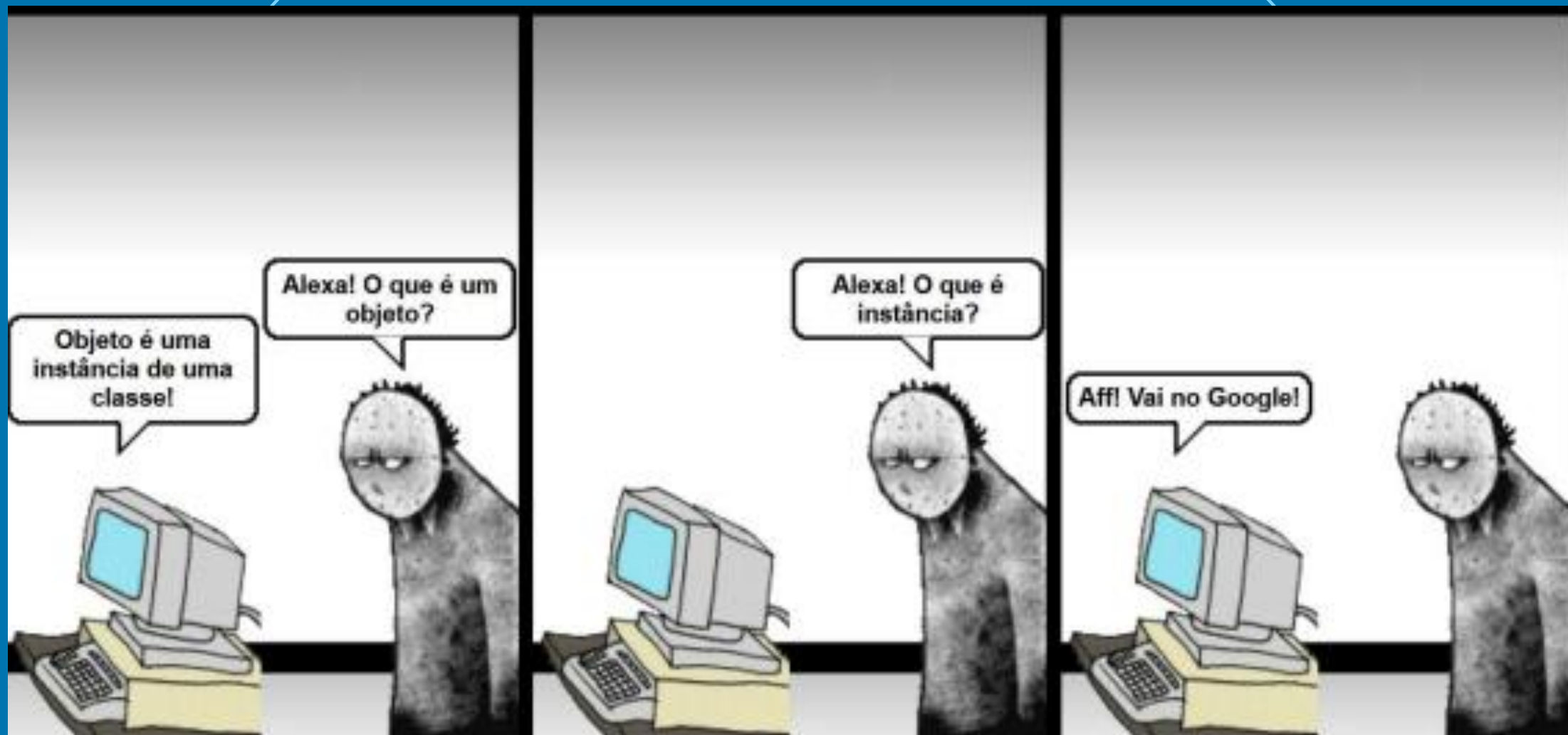
O professor Rômulo disse que vai usar um tal de ORM!

E aquelas paradas de relacionamento do Codd?

Enquanto Codd usava só a PK e FK, o ORM usa Objetos e Listas de Navegação! Mas no final tudo acaba em Code.

O que é ORM? :

OO vs MER





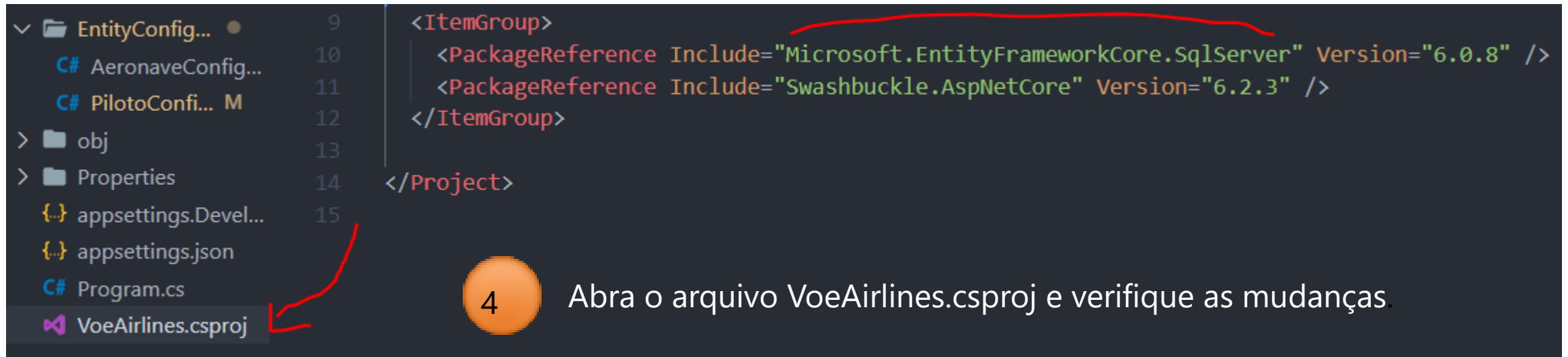
Problema – Impedância objeto-relacional

Instalação | EntityFramework

1 Abra o Visual Studio Code e utilize a tecla de atalho CTRL+ SHIFT + '

2 Verifique se o terminal integrado abriu no VS CODE.

3 Digite o seguinte comando no terminal:
dotnet add package Microsoft.EntityFrameworkCore.SqlServer



```
<ItemGroup>
  <PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer" Version="6.0.8" />
  <PackageReference Include="Swashbuckle.AspNetCore" Version="6.2.3" />
</ItemGroup>
</Project>
```

4 Abra o arquivo VoeAirlines.csproj e verifique as mudanças.

DBContext



Abstração do BD



“Você sabe o que é um banco de dados? Se a resposta for sim, então você já andou mais que a metade do caminho. O DBContext representa uma abstração do banco de dados dentro da nossa aplicação, possuindo coleções que são equivalentes as nossas já conhecidas tabelas.”

—RÔMULO C. SILVESTRE, ENG. SISTEMAS

INTRODUÇÃO

Vamos começar organizando o nosso ambiente.

Crie uma pasta Contexts:

Por mais que temos apenas um DbContext é indicado criar essa pasta para que em projetos maiores ela possa atender os diversos banco de dados.

Dentro da pasta você deve criar um arquivo.

Esse arquivo irá se chamar VoeAirlinesContext.cs que corresponde respectivamente ao nome da classe.

Não esqueça de criar também o namespace. No próximo Slide você verá a listagem de codificação.

“Você só aprender programar, programando” Dennis M. Ritchie – Criador da linguagem C.

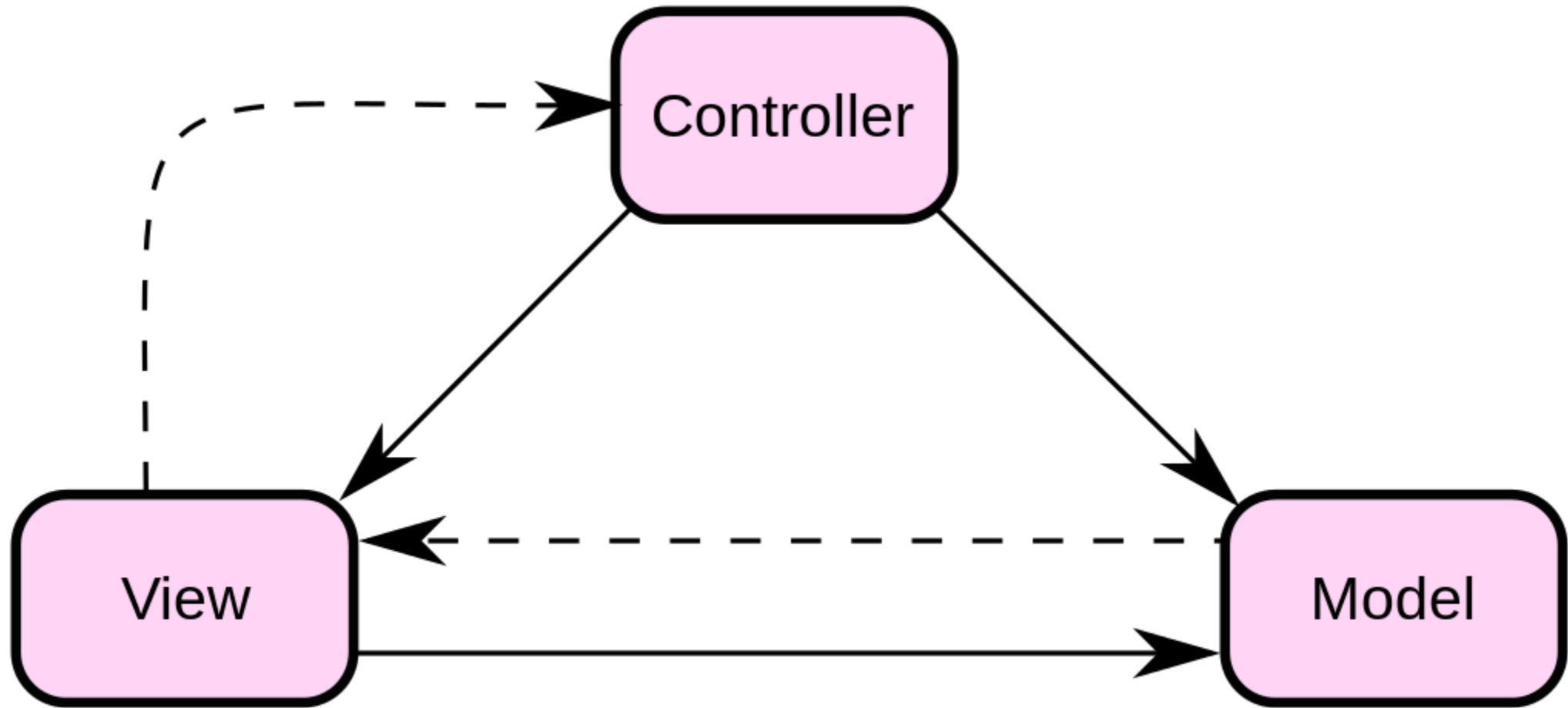
Defina e utilize namespaces

```
1  ✓ using Microsoft.EntityFrameworkCore;  
2    using VoeAirlines.Entities;  
3  
4    namespace VoeAirlines.Contexts;  
5
```


“Você só aprender programar, programando” Dennis M. Ritchie – Criador da linguagem C.

Defina a herança

```
6 public class VoeAirlinesContext:DbContext{  
7
```





O primeiro passo é criar coleções que representam as tabelas do nosso banco de dados. Para isso, utilizaremos **DbSet< >**

Rômulo C. Silvestre

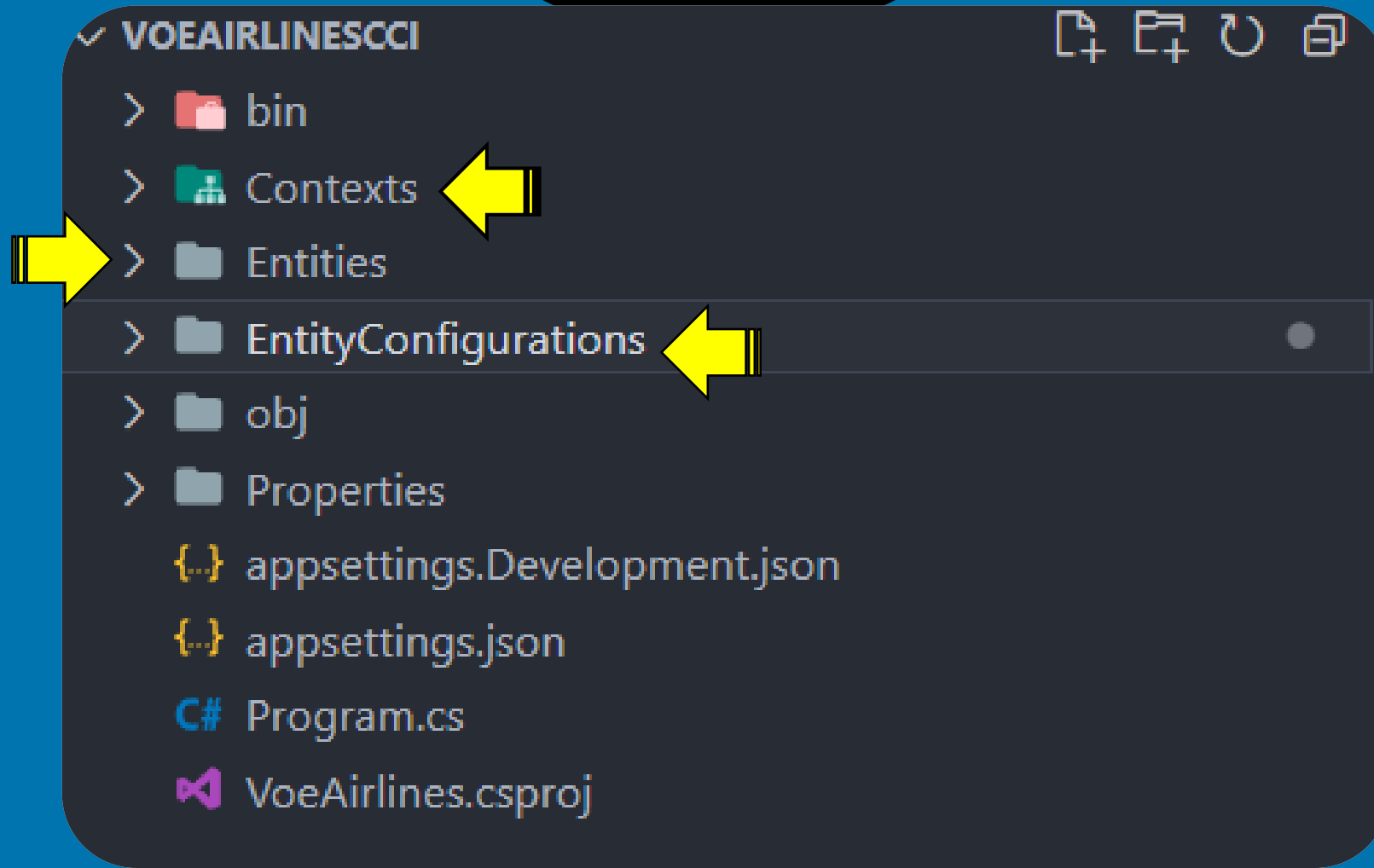
“Você só aprender programar, programando” Dennis M. Ritchie – Criador da linguagem C.

Defina o DBSet

```
public DbSet<Aeronave> Aeronaves =>Set<Aeronave>();  
0 references  
public DbSet<Manutencao> Manutencoes =>Set<Manutencao>();  
0 references  
public DbSet<Piloto> Pilotos =>Set<Piloto>();  
0 references  
public DbSet<Voo> Voos =>Set<Voo>();  
0 references  
public DbSet<Cancelamento> Cancelamentos =>Set<Cancelamento>();
```

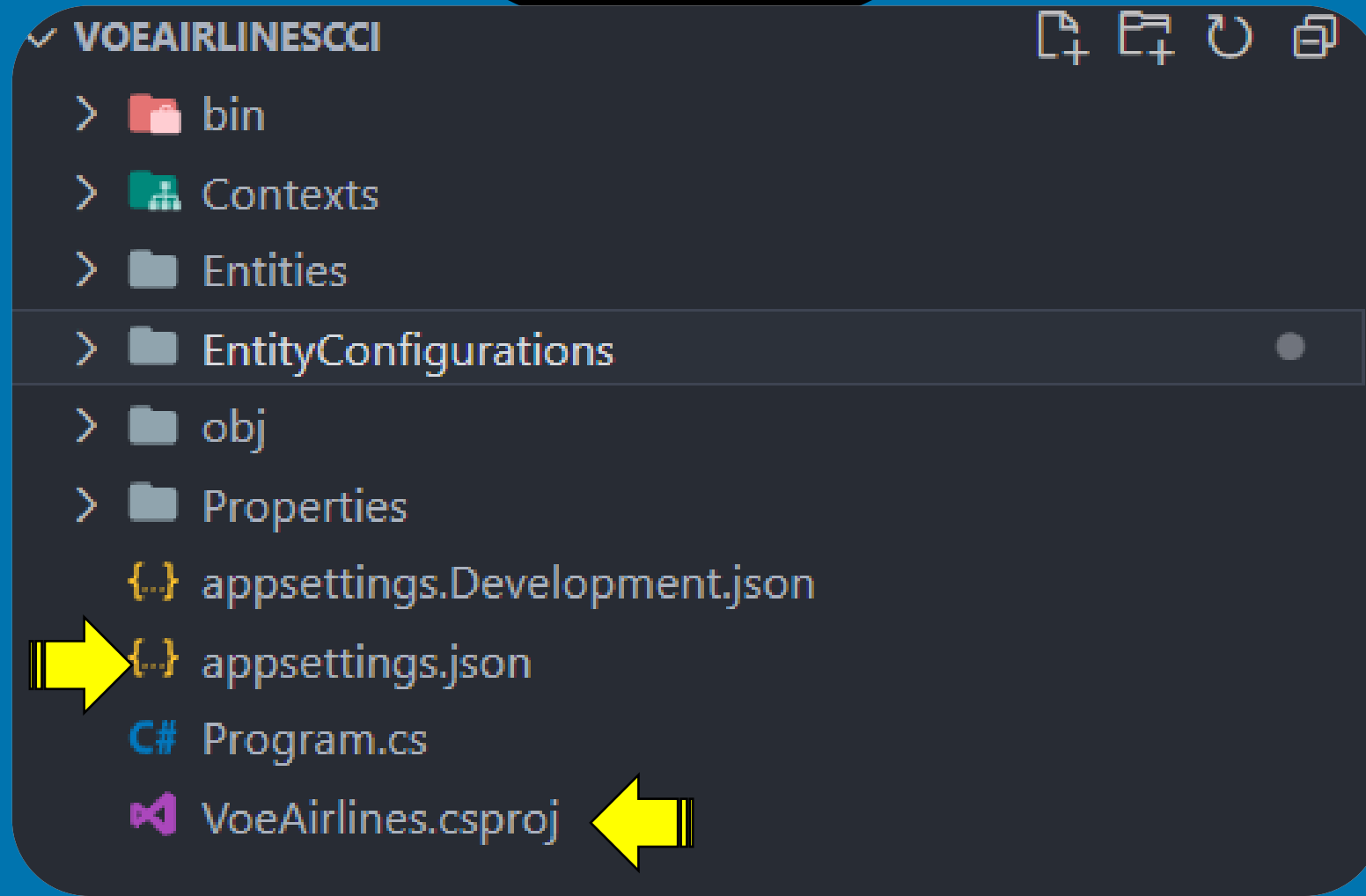
DICA

Crie as pastas seguindo os padrões do treinamento



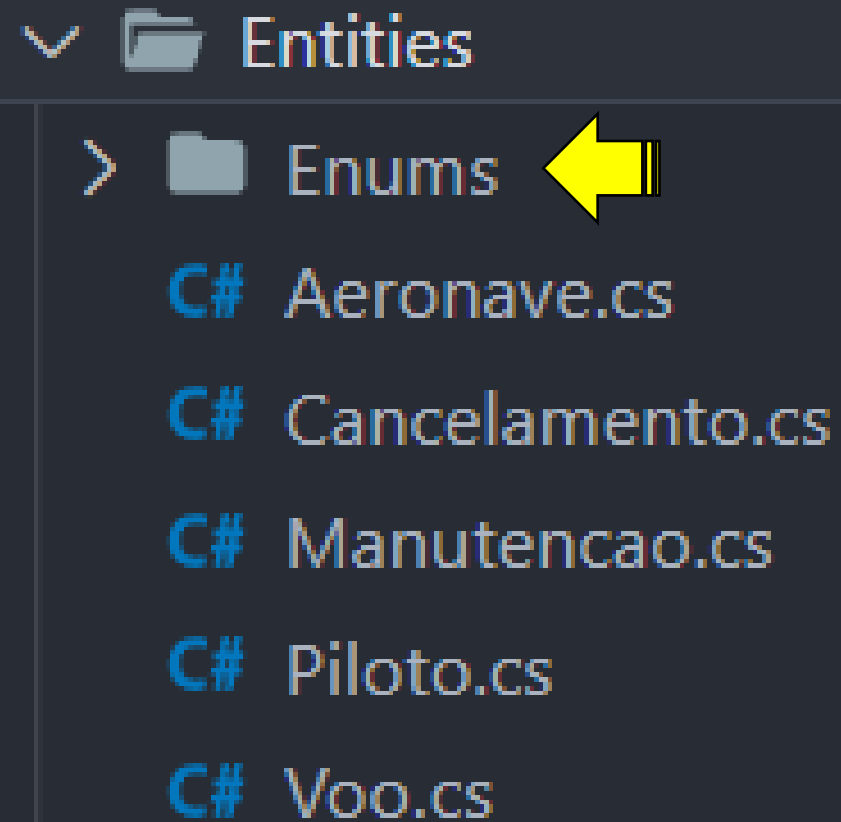
DICA

Observe os seguintes arquivos de configuração!



DICA

Entidades – Crie os seguintes arquivos



DICA

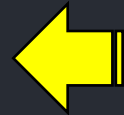
Entidades – Crie os seguintes arquivos



Entities



Enums



C# Aeronave.cs

C# Cancelamento.cs

C# Manutencao.cs

C# Piloto.cs

C# Voo.cs

DICA

Entidades – Crie os seguintes arquivos



✓  Entities

✓  Enums

C# TipoManuntecao.cs 

```
1 namespace VoeAirlines.Entities.Enums;
2
3 0 references
4 public enum TipoManutecao
5 {
6     0 references
7     Preventiva,
8     0 references
9     Corretiva
10 }
11
```

Enum

Enum agrupa constantes| PADRONIZAÇÃO.

Com ela você pode padronizar valores constantes e evitar o uso indevido de alguns valores.

O PALCO ESTÁ ARMADO
vá em frente e siga as 5 etapas básicas

1

[Aeronave.cs](#)

2

[Manutencao.cs](#)

3

[Piloto.cs](#)

4

[Voo.cs](#)

5

[Cancelamento.cs](#)

1

Aeronave.cs


DEFINA O NAMESPACE

Crie a classe, as propriedades automáticas!
Não pode esquecer do Contrutor!



```
1  ✓ //Namespace é um conjunto de classes
2    //Namespace é uma divisão lógica
3    namespace VoeAirlinesSenai.Entities;
4  ✓ //Classe: é um conjunto de objetos
5    //Objeto: é uma instância de uma classe
    3 references
```

```
6   public class Aeronave  
7   {
```

```
public int Id { get; set; }
```

1 reference

```
public string Fabricante { get; set; }
```

1 reference

```
public string Modelo { get; set; }
```

1 reference

```
public stringCodigo { get; set; }
```

0 references



```
public ICollection<Manutencao> Manutencoes { get; set; } = null!;
```

```
public Aeronave(string fabricante, string modelo, string codigo)
{
    Fabricante = fabricante;
    Modelo = modelo;
    Codigo = codigo;
}
```

CTRL+ "." - tecla de atalho para gerar o construtor.
Lembre-se de selecionar as propriedades automáticas antes.

```
1 //Namespace é um conjunto de classes
2 //Namespace é uma divisão lógica
3 namespace VoeAirlinesSenai.Entities;
4 //Classe:é um conjunto de objetos
5 //Objeto: é uma instância de uma classe
6 3 references
7 public class Aeronave
8 {
9     0 references
10     public Aeronave(string fabricante, string modelo, string codigo)
11     {
12         Fabricante = fabricante;
13         Modelo = modelo;
14        Codigo = codigo;
15     }
16     //Propriedades Automáticas
17     //Características do objeto
18     //Automático:gera o get set
19     //Métodos set -atribui
20     //Métodos get -recupera
21     //POCO-foco é o objeto
22     0 references
23     public int Id { get; set; }
24     1 reference
25     public string Fabricante { get; set; }
26     1 reference
27     public string Modelo { get; set; }
28     1 reference
29     public string Codigo { get; set; }
30     0 references
31     public ICollection<Manutencao> Manutencoes { get; set; } = null!;
32 }
```

2

Manutencao.cs

**OBSERVAÇÃO PODE SER
NULA**

DateTime | Struct | ICollection | Enum | null!



0 references

```
public Manutencao(DateTime dataHora,TipoManutencao tipo, int aeronaveId,string? observacoes=null)
{
    DataHora = dataHora;
    Observacoes = observacoes;
    Tipo = tipo;
    AeronaveId = aeronaveId;
}
```

```
public int Id { get; set; }
```

1 reference

```
public DateTime DataHora { get; set; }
```

1 reference

```
public string? Observacoes { get; set; }
```

1 reference

```
public TipoManutencao Tipo { get; set; }
```

1 reference

```
public int AeronaveId { get; set; }
```

0 references

```
public Aeronave Aeronave { get; set; } = null!;
```

```
1 namespace VoeAirlinesSenai.Entities;
2 using VoeAirlinesSenai.Entities.Enums;
3 references
4 public class Manutencao
5 {
6     0 references
7     public Manutencao(DateTime dataHora, TipoManutencao tipo, int aeronaveId, string? observacoes=null)
8     {
9         DataHora = dataHora;
10        Observacoes = observacoes;
11        Tipo = tipo;
12        AeronaveId = aeronaveId;
13    }
14    0 references
15    public int Id { get; set; }
16    1 reference
17    public DateTime DataHora { get; set; }
18    1 reference
19    public string? Observacoes { get; set; }
20    1 reference
21    public TipoManutencao Tipo { get; set; }
22    1 reference
23    public int AeronaveId { get; set; }
24    0 references
25    public Aeronave Aeronave { get; set; } = null!;
26 }
```

3

Piloto

ICollection

Essa interface define métodos para manipular coleções genéricas.

```
1 namespace VoeAirlinesSenai.Entities;
  0 references
2 public class Piloto
3 {
  0 references
4 public Piloto(string nome, string matricula)
5 {
6     Nome = nome;
7     Matricula = matricula;
8 }
  0 references
9 public int Id { get; set; }
  1 reference
10 public string Nome { get; set; }
  1 reference
11 public string Matricula { get; set; }
12 }
```


4

Voo.cs

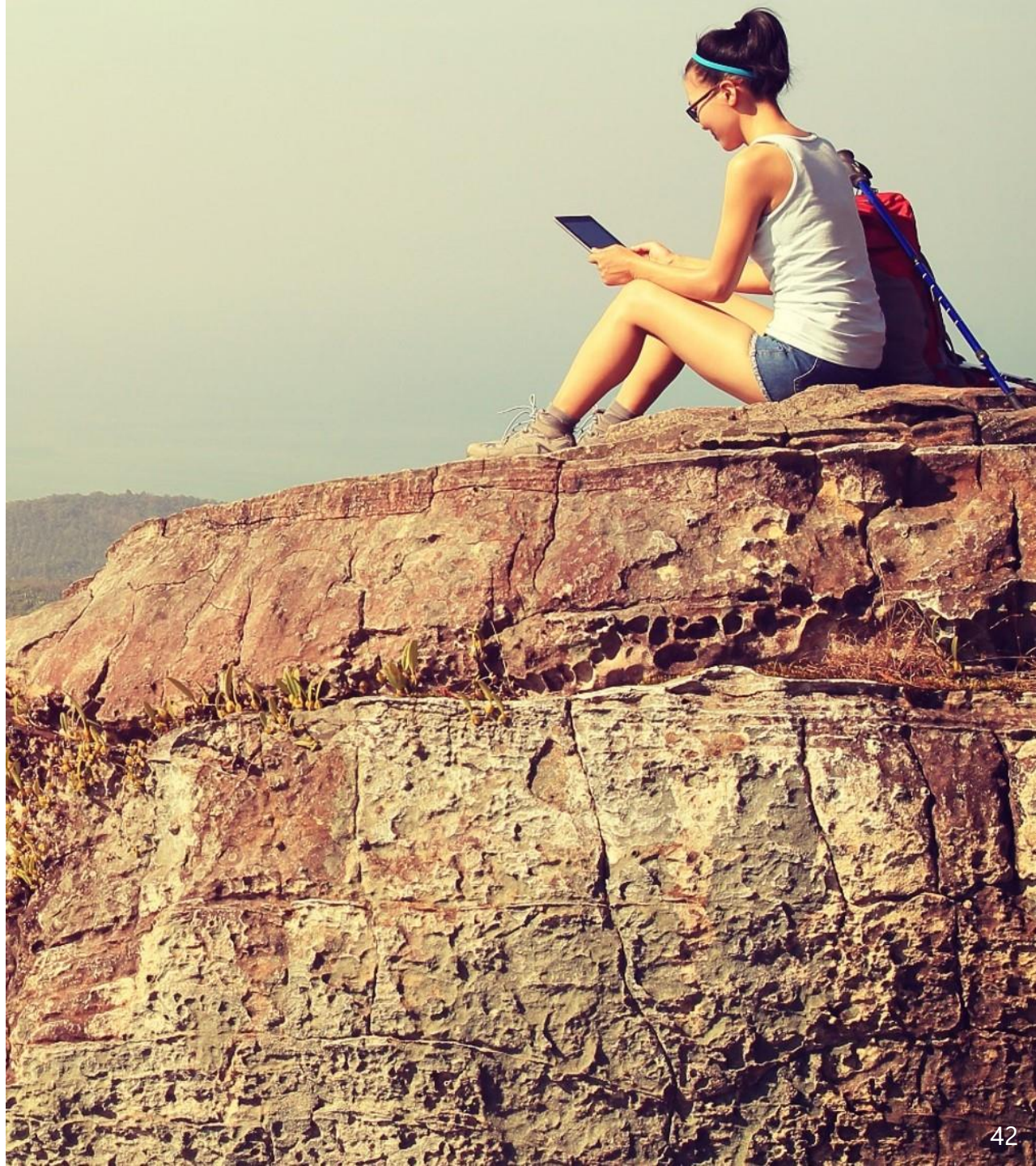
O relacionamento!

Lembra do Codd? - Relacionamento

Aqui temos um para muitos

Muitos para Muitos

Um para UM



REFERENCES

```
public Voo(string origem, string destino, DateTime dataHoraPartida, DateTime dataHoraChegada, int aeronaveId, int pilotoId)
{
    Origem = origem;
    Destino = destino;
    DataHoraPartida = dataHoraPartida;
    DataHoraChegada = dataHoraChegada;
    AeronaveId = aeronaveId;
    PilotoId = pilotoId;
}
```

```
public int Id { get; set; }  
1 reference  
public string Origem { get; set; }  
1 reference  
public string Destino { get; set; }  
1 reference  
public DateTime DataHoraPartida { get; set; }  
1 reference  
public DateTime DataHoraChegada { get; set; }  
1 reference  
public int AeronaveId { get; set; }  
1 reference  
public int PilotoId { get; set; }  
0 references  
public Aeronave Aeronave { get; set; } = null!;  
0 references  
public Piloto Piloto { get; set; } = null!;
```

```
1 namespace VoeAirlinesSenai.Entities;
  0 references
2 public class Voo
3 {
  0 references
4     public Voo(string origem, string destino, DateTime dataHoraPartida, DateTime dataHoraChegada, int aeronaveId, int pilotoId)
5     {
6         Origem = origem;
7         Destino = destino;
8         DataHoraPartida = dataHoraPartida;
9         DataHoraChegada = dataHoraChegada;
10        AeronaveId = aeronaveId;
11        PilotoId = pilotoId;
12    }
  0 references
13    public int Id { get; set; }
  1 reference
14    public string Origem { get; set; }
  1 reference
15    public string Destino { get; set; }
  1 reference
16    public DateTime DataHoraPartida { get; set; }
  1 reference
17    public DateTime DataHoraChegada { get; set; }
  1 reference
18    public int AeronaveId { get; set; }
  1 reference
19    public int PilotoId { get; set; }
  0 references
20    public Aeronave Aeronave { get; set; } = null!;
  0 references
21    public Piloto Piloto { get; set; } = null!;
22    // public Cancelamento Cancelamento { get; set; }
23 }
```



5

Cancelamento.cs

Um para Um

Nosso projeto tem pelo menos um caso de um para um. Onde um voo possui um cancelamento, e um cancelamento só pode cancelar um voo.

```
1 namespace VoeAirlinesSenai.Entities;
  1 reference
2 public class Cancelamento
3 {
  0 references
4     public Cancelamento(string motivo, DateTime dataHoraNotificacao, int vooId)
5     {
6         Motivo = motivo;
7         DataHoraNotificacao = dataHoraNotificacao;
8         VooId = vooId;
9     }
  0 references
10    public int Id { get; set; }
  1 reference
11    public string Motivo { get; set; }
  1 reference
12    public DateTime DataHoraNotificacao { get; set; }
  1 reference
13    public int VooId { get; set; }
  0 references
14    public Voo Voo { get; set; } = null!;
15 }
```



Volte na Voo.cs


```
19 | public int PilotId { get; set; }  
   | 0 references  
20 | public Aeronave Aeronave { get; set; } = null!;  
   | 0 references  
21 | public Piloto Piloto { get; set; } = null!;  
   | 0 references  
22 | 💡 public Cancelamento Cancelamento { get; set; } = null!;  
23 | }
```

Volte na classe
Voo.cs e adicione a
propriedade
Cancelamento



Volte na
Aeronave.cs

0 references

```
4 | public ICollection<Manutencao> Manutencoes { get; set; } = null!;
```

0 references

```
5 | public ICollection<Voo> Voos {get;set;}=null!;
```

```
6 | }
```

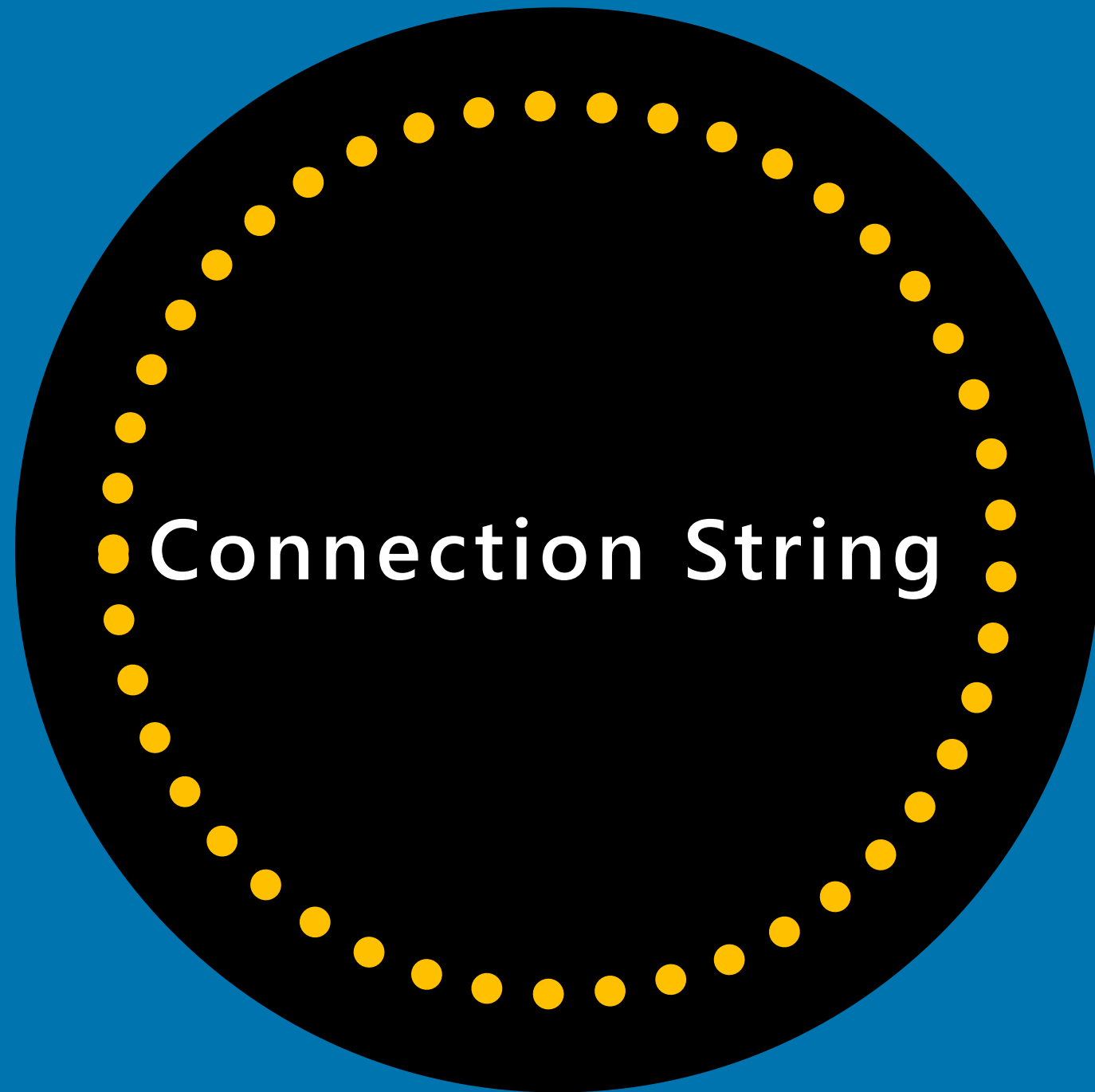


Volte na Piloto.cs

```
10 | 1 reference  
    | public string Nome { get; set; }  
11 | 1 reference  
    | public string Matricula { get; set; }  
12 | 0 references  
    | ⚡ public ICollection<Voo> Voos { get; set; }=null!;  
13 | }
```

AULA 2

Conexão com o Banco de Dados

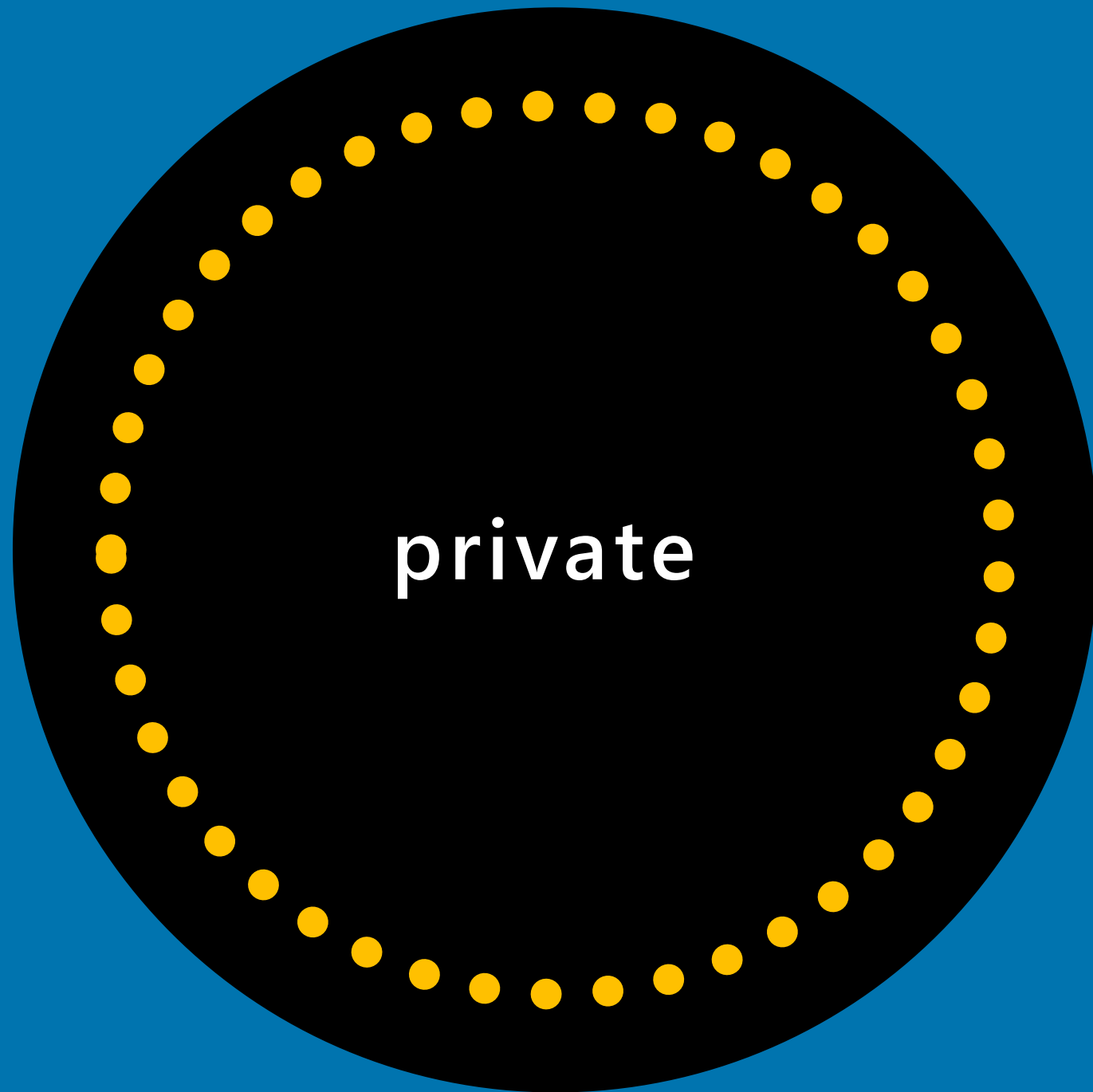




Autenticação com Sql Server

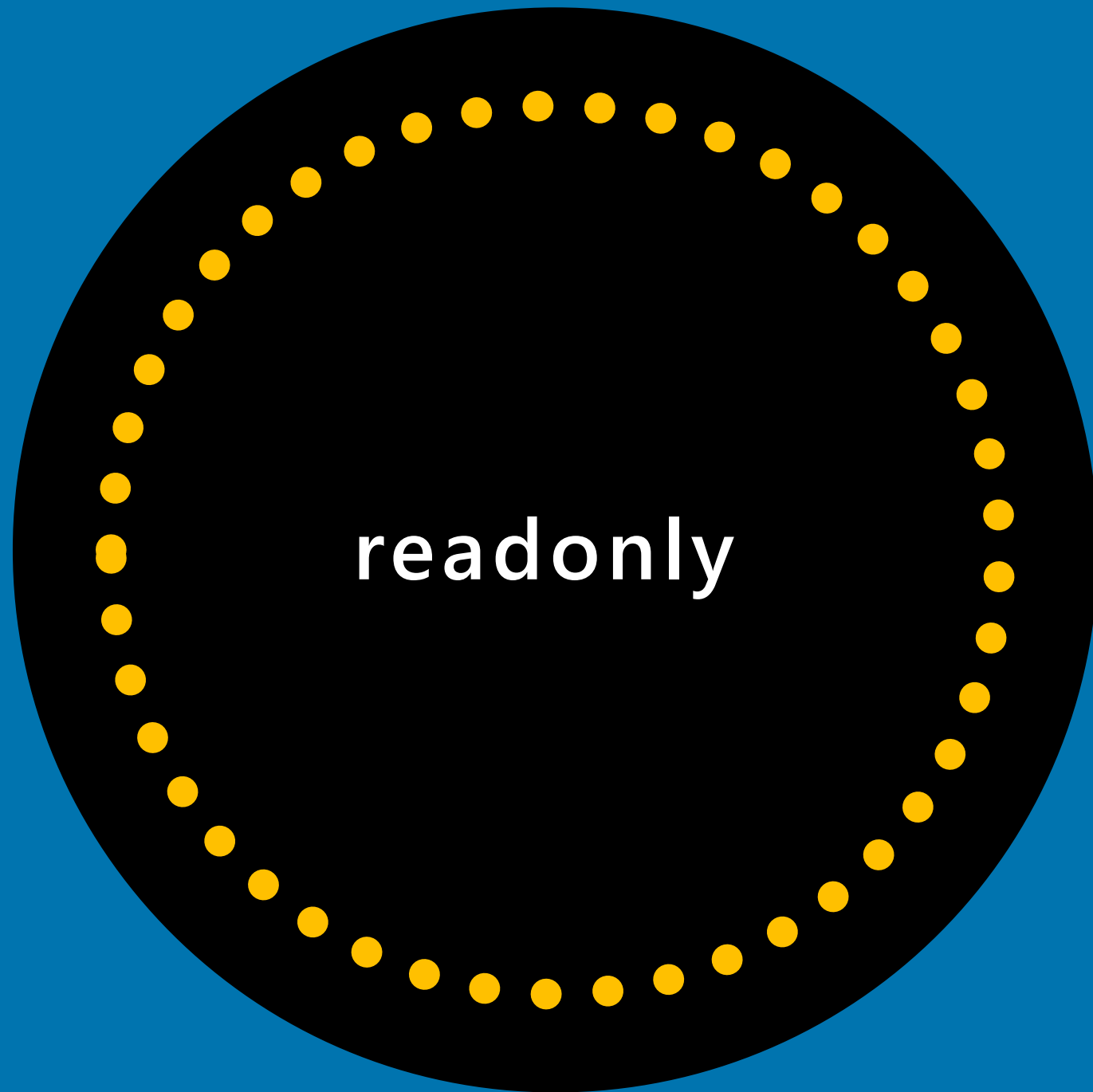


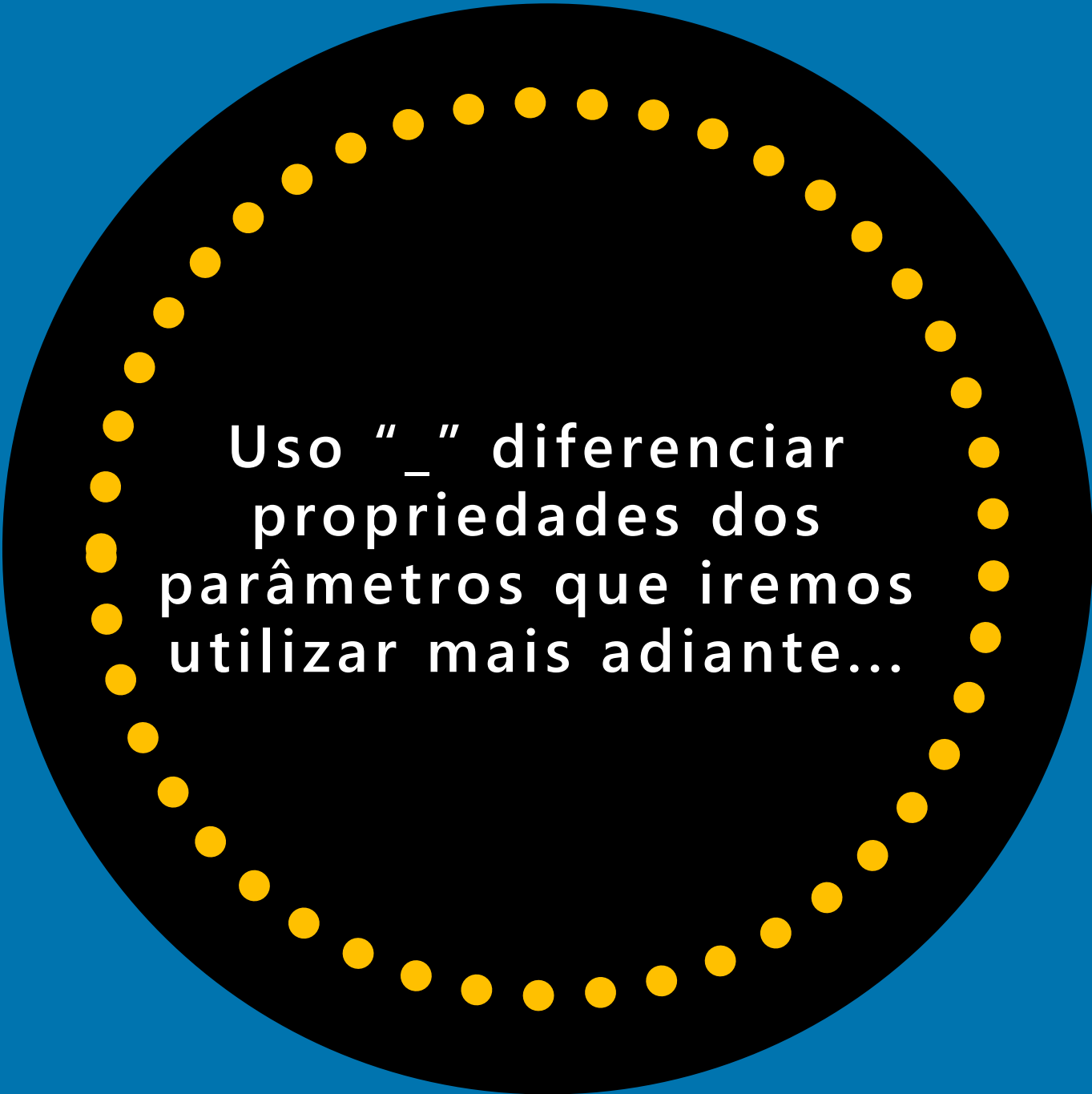
Propriedade
Privada



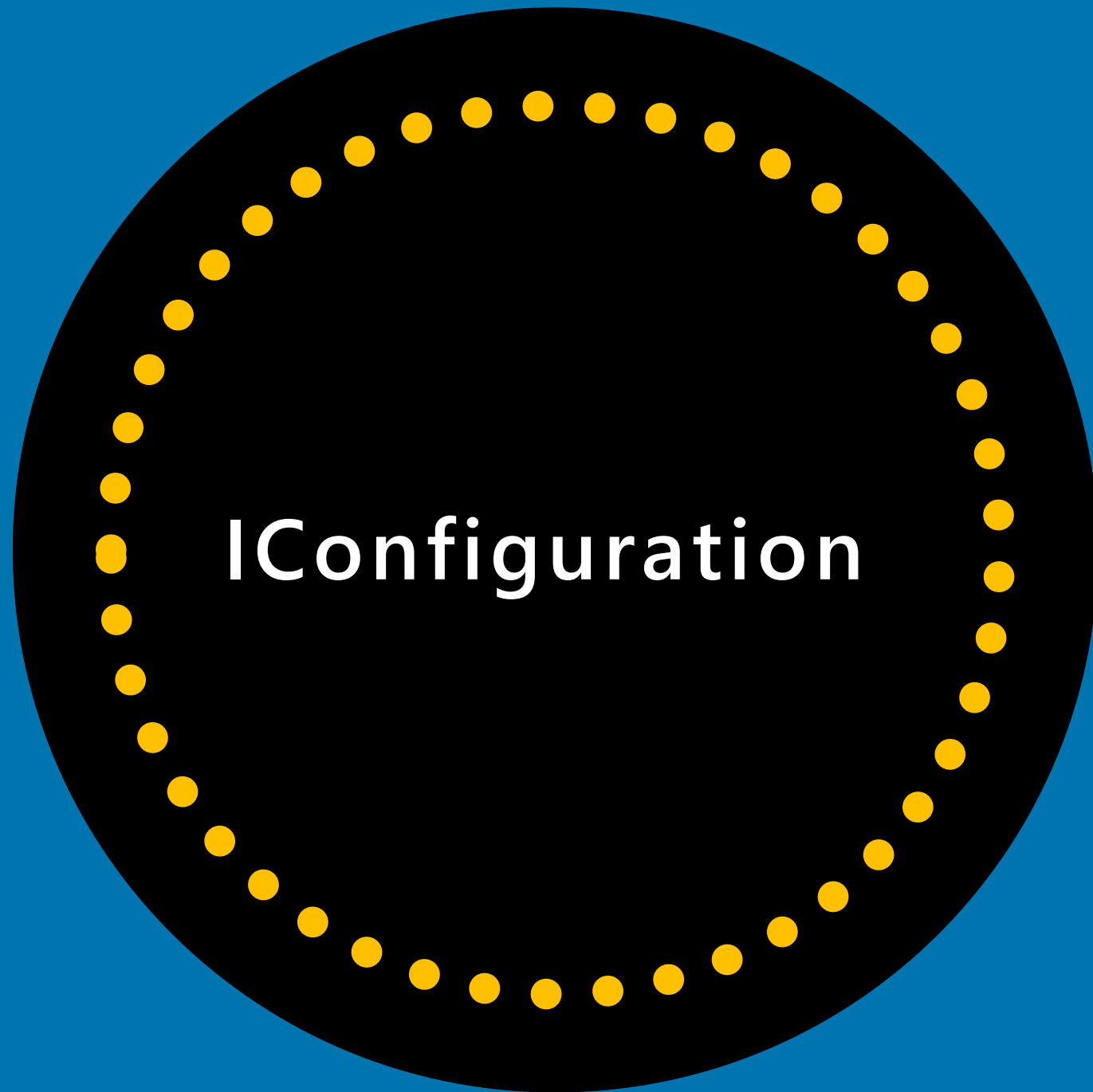


Somente de
leitura





Uso "_" diferenciar
propriedades dos
parâmetros que iremos
utilizar mais adiante...



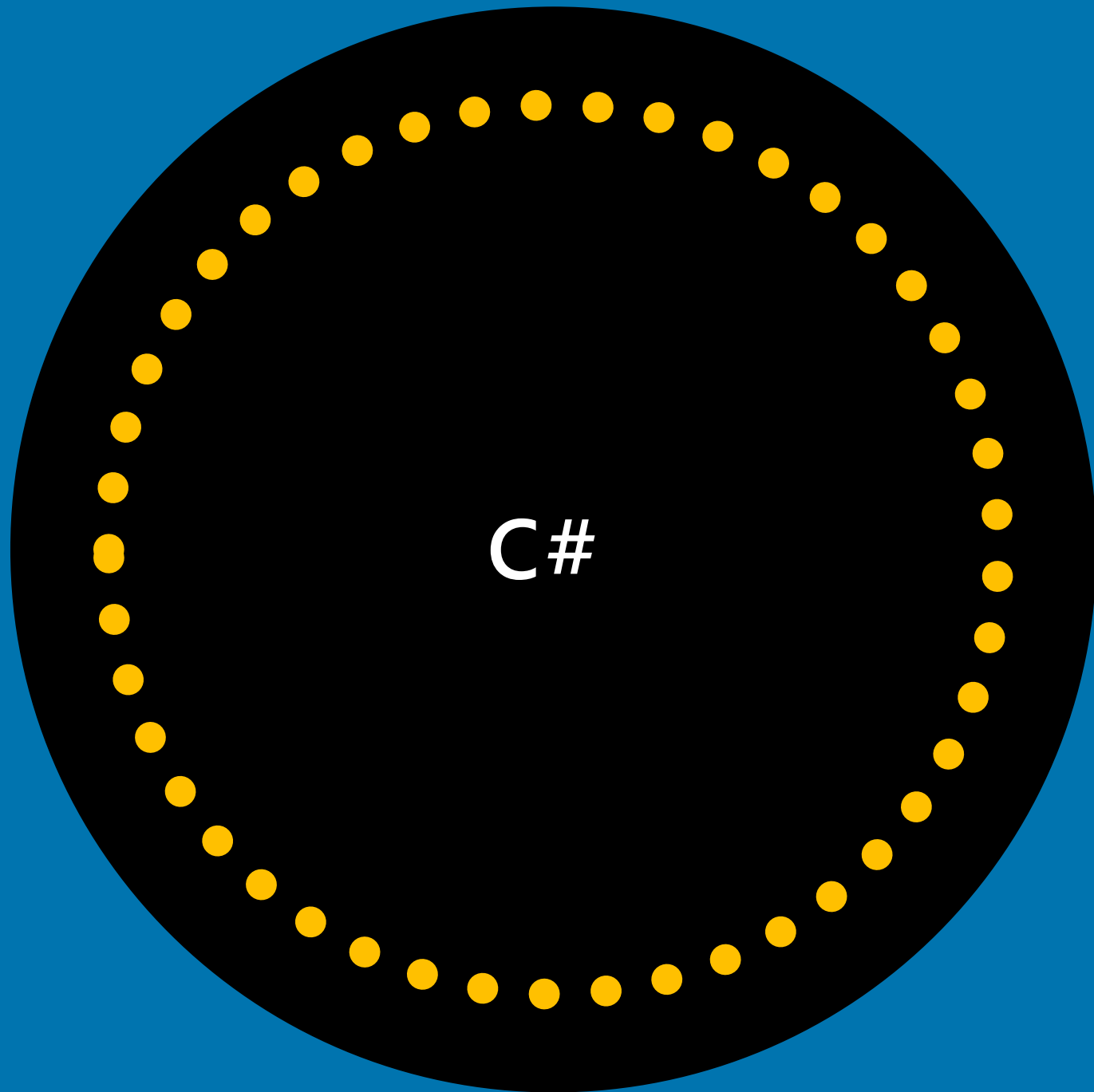
IConfiguration

The logo consists of a large black circle centered on a blue background. Inside the black circle, there is a ring of 30 small yellow dots arranged in a circular pattern. The text "DbContext" is written in white, sans-serif font in the center of the black circle.

DbContext

The image features a solid blue background. Two white circles are present: a large one on the left and a smaller one on the right, partially overlapping the larger one. The text "Vamos começar?" is centered horizontally across the middle of the image, overlapping both circles.

Vamos começar?



Tipos por valor do C#

1

int

2

float

3

double

4

struct

5

enums

Tipos por referência do C#

1

interface

2

classes

3

delegates

4

arrays

5

strings



A linguagem C# por padrão considera os tipos por valor como “não anuláveis” ou “non nullable”.

Prof. Rômulo C. Silvestre



Para criar uma variável de um tipo anulável, basta adicionar um ponto de **interrogação(?)** no final do nome do tipo correspondente, na declaração da variável.

Ex: **int? valor =0;** (*valor é um nullable type*) **ou Nullable<int> valor=0;**



O código a seguir declara uma variável do tipo **int** anulável.
O sufixo(?) é anexado ao nome do tipo, não ao nome da variável.

```
int? meuInteiro = 28;
```

Com essa instrução de declaração, o compilador cria o tipo anulável que agora aceita **null**;


Connection String



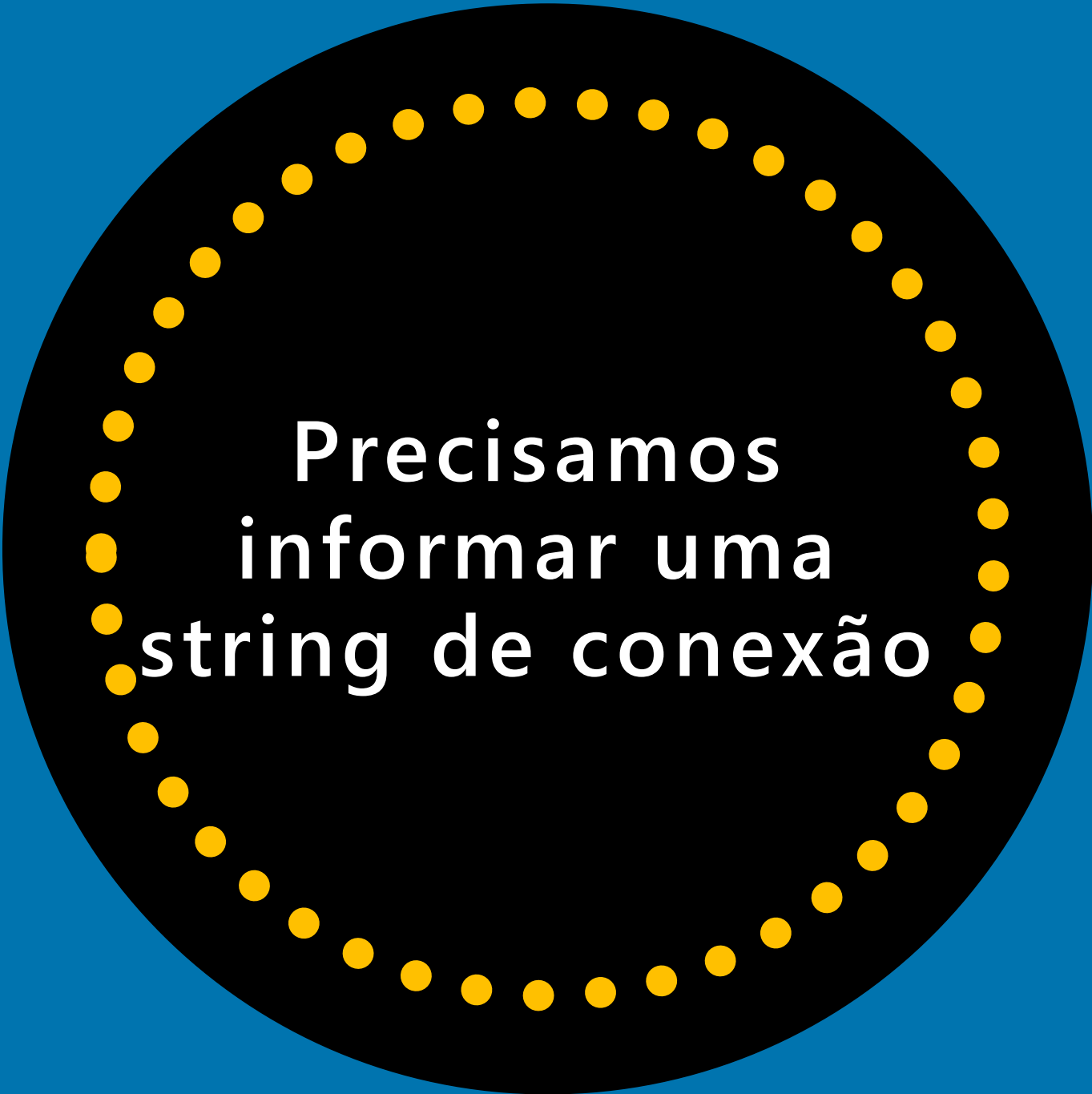
Site Connection
Strings [aqui](#)

The logo consists of a large black circle centered on a blue background. Inside the black circle is a ring of small yellow dots. The text "DbContext Estruturado" is written in white, sans-serif font in the center of the black circle.

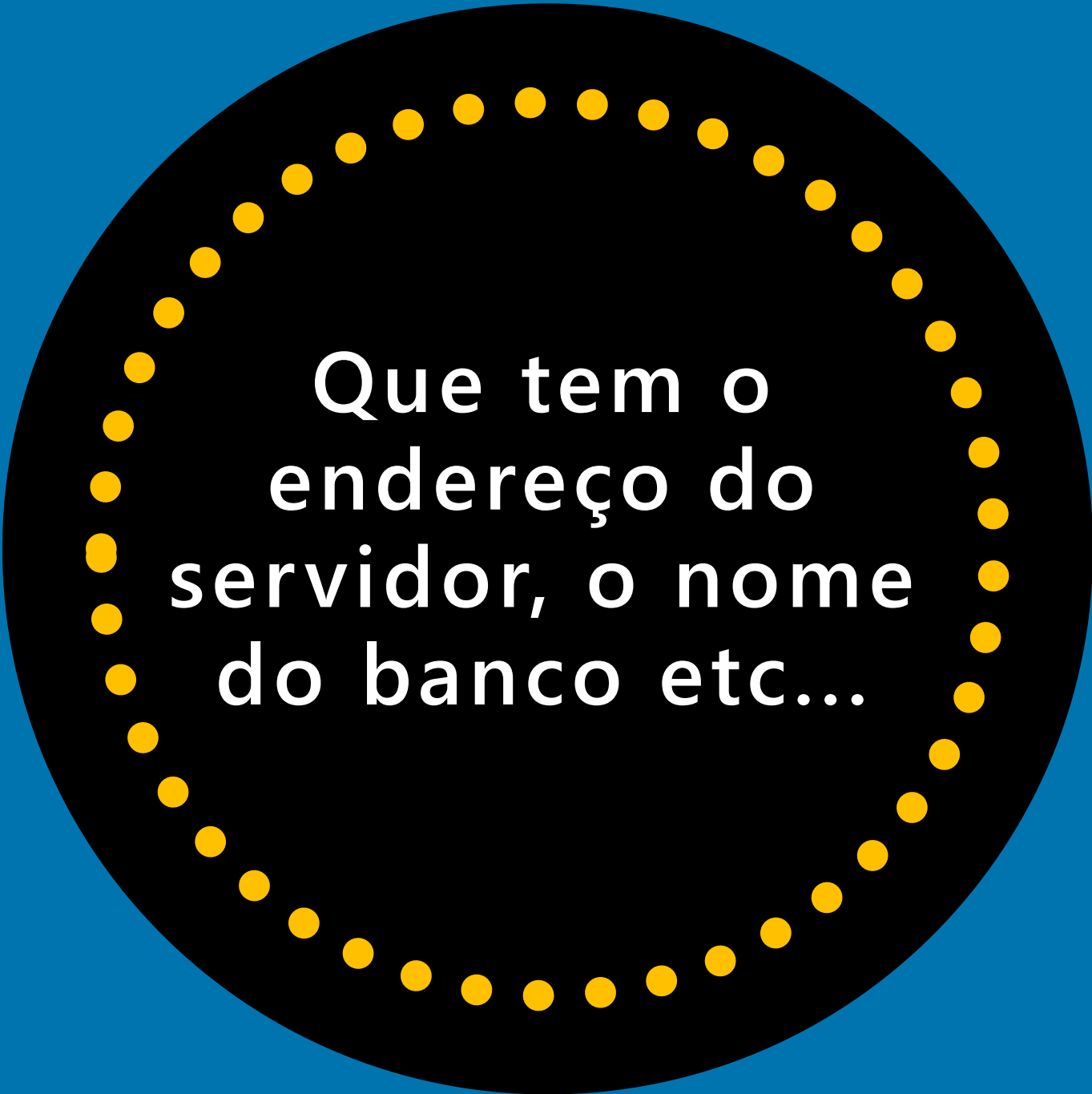
DbContext Estruturado




Configurar
DbContext para
acessar o banco
de dados



Precisamos
informar uma
string de conexão



Que tem o
endereço do
servidor, o nome
do banco etc...



Credenciais
necessárias para
se conectar ao
SQL Server



Crie uma
propriedade
privada

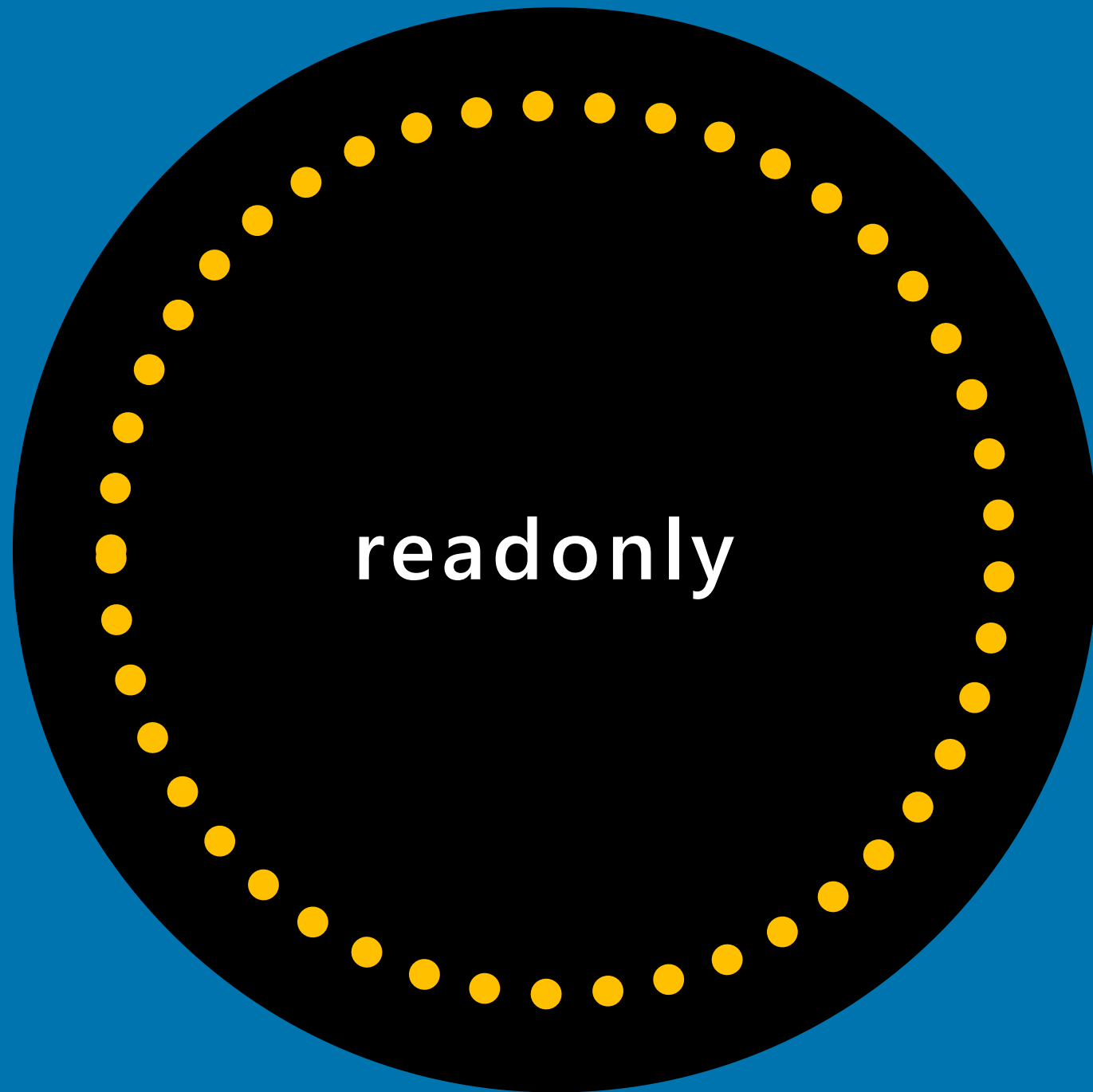


private

Humm.. Isso deve ser
encapsulamento...

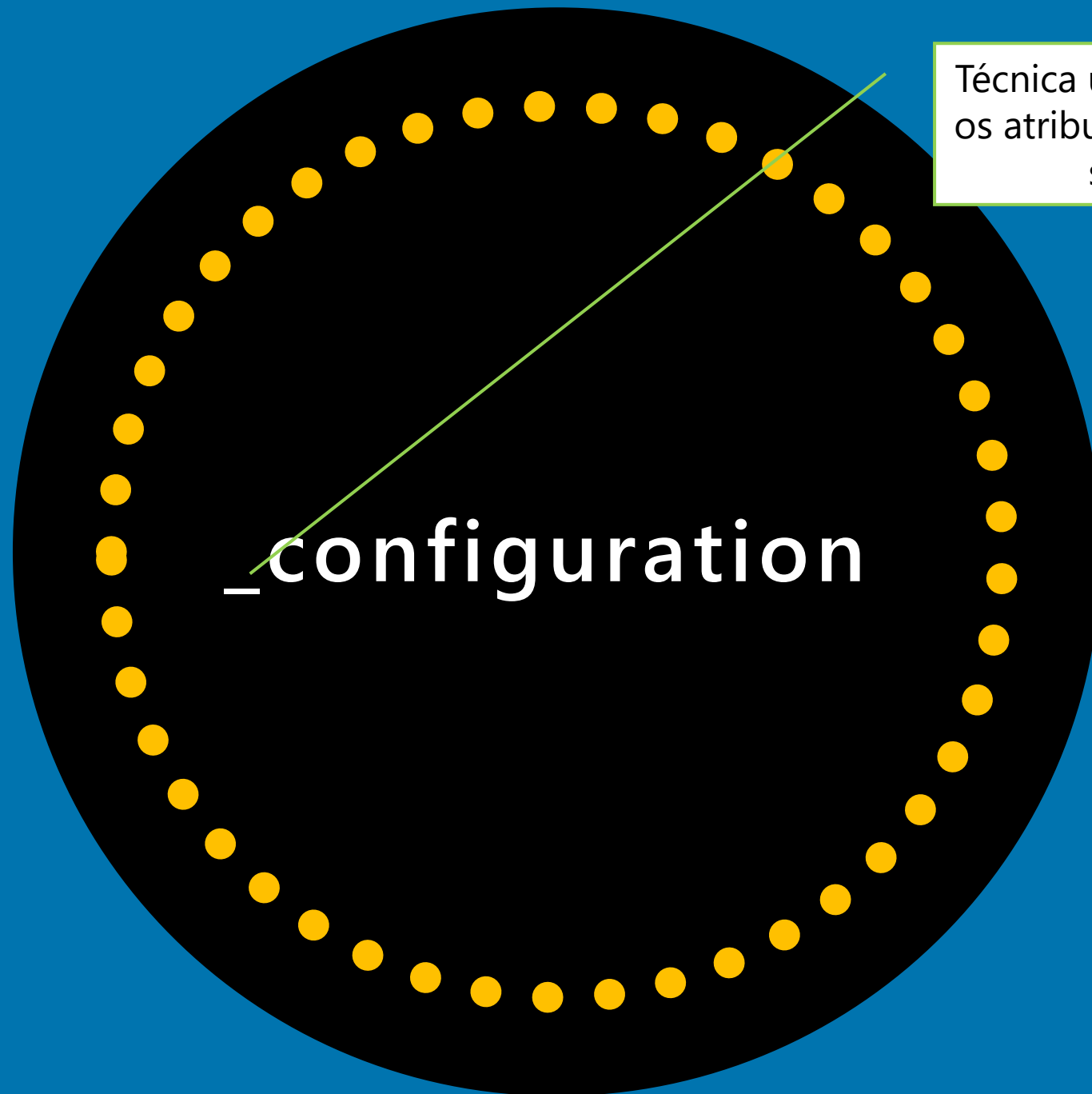


Apenas leitura





Tipo:
IConfiguration



Técnica utilizada para diferenciar os atributos dos parâmetros que serão atribuídos.

IConfiguration

Uma Classe do próprio
ASP.NET


Permite carregar uma estrutura de
configurações.

Configurações dinâmicas

Vai ler do arquivo *appsettings.json*

```
1 {  
2  
3   "ConnectionStrings": {  
4     "VoeAirlines": "Server=localhost\\SQLExpress;Database=VoeAirlines;Integrated Security=True;"  
5   },  
6   "Logging": {  
7     "LogLevel": {  
8       "Default": "Information",  
9       "Microsoft.AspNetCore": "Warning"  
10    }  
11  },  
12  "AllowedHosts": "*"   
13 }  
14
```

Humm. É equivalente
ao webconfig em
versões anteriores



● **IConfiguration** vai nos permitir
● ler a string de conexão do
● arquivo .json



Crie o construtor da
propriedade `_configuration`

```
1  using Microsoft.EntityFrameworkCore;
2  using VoeAirlines.Entities;
3  namespace VoeAirlines.Contexts;
4  public class VoeAirlinesContext:DbContext{
5      private readonly IConfiguration _configuration;
6
7      public VoeAirlinesContext(IConfiguration configuration)
8      {
9          _configuration = configuration;
10     }
11     public DbSet<Aeronave> Aeronaves =>Set<Aeronave>();
12     public DbSet<Manutencao> Manutencoes =>Set<Manutencao>();
13     public DbSet<Piloto> Pilotos =>Set<Piloto>();
14     public DbSet<Voo> Voos =>Set<Voo>();
15     public DbSet<Cancelamento> Cancelamentos =>Set<Cancelamento>();
16 }
```

Eita! **Injeção** de Dependência.

O próprio EF vai **injetar** a instância para mim.

“Você só aprender programar, programando” Dennis M. Ritchie – Criador da linguagem C.

IConfiguration definido

```
private readonly IConfiguration _configuration;

0 references
public VoeAirlinesContext(IConfiguration configuration)
{
    _configuration = configuration;
}
```



Não!

Não é preciso...

Instanciar

Manualmente

O IConfiguration



Já posso utilizá-lo que ele vai carregar para mim.

Sobre o IConfiguration



Vamos utilizar o objeto que criamos.

Sobre o IConfiguration..



protected **override**

Sobre Polimorfismo!

A graphic featuring a large black circle centered on a blue background. Inside the black circle, the word "Polimorfismo" is written in white. Surrounding the text is a ring of small yellow dots.

Polimorfismo



Poli = Muito Morfismo = Formas

Sobre o Polimorfismo!



Nadar() é um método polimórfico, pois cada um pode nadar de um jeito diferente usando o mesmo método.

Rômulo C. Silvestre – Sobre Polimorfismo



Sobrescrita ou Sobrepor um método é que dizemos,
override é o que programamos.

Rômulo C. Silvestre – Sobre polimorfismo



Acredito que o polimorfismo verdadeiro encontra-se principalmente nas interfaces e classes abstratas.

Rômulo C. Silvestre – Sobre Polimorfismo

Vamos definir um Polimorfismo



```
protected override void OnConfiguring(DbContextOptionBuilder optionsBuilder){  
    optionsBuilder.UseSqlServer(_configuration.GetConnectionString("VoeAirlines"));  
}
```


The image features a solid blue background. Two thin white circles are present: a large one on the left and a smaller one on the right, partially overlapping the larger one. Centered horizontally across the middle of the image is the text "META CONCLUÍDA!" in a bold, white, sans-serif font.

META CONCLUÍDA!



CONCLUSÃO

Tudo pronto para a próxima etapa.

INSPIRE

Não pare de avançar.

AÇÃO

Parabéns a toda nossa equipe.



VENCER É UMA QUESTÃO DE ESCOLHA. VAMOS
VENCER ESSE SPRINT?

—RÔMULO C. SILVESTRE

ROMULO C. SILVESTRE!

OBRIGADO