

ORM - EntityFrameWork

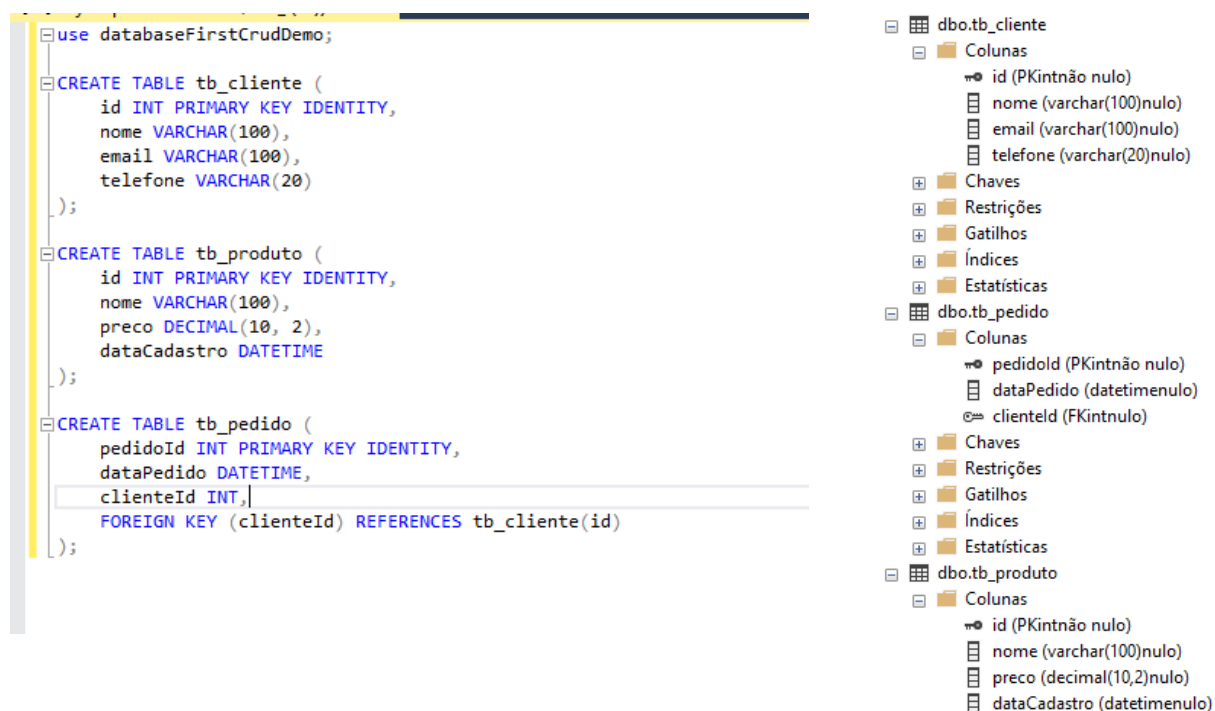
De começo há duas abordagens:

→ **Code-first:** cria-se primeiro a classe Model, para depois criar as tabelas no BD através de migration

→ **Database-first:** o banco de dados é criado e depois são criados as classes em cima do BD.

Database-first

Para os testes usaremos o seguinte script no SQLServer:



The screenshot displays a SQL Server environment. On the left, a script window shows the following T-SQL code:

```
use databaseFirstCrudDemo;  
  
CREATE TABLE tb_cliente (  
    id INT PRIMARY KEY IDENTITY,  
    nome VARCHAR(100),  
    email VARCHAR(100),  
    telefone VARCHAR(20)  
);  
  
CREATE TABLE tb_produto (  
    id INT PRIMARY KEY IDENTITY,  
    nome VARCHAR(100),  
    preco DECIMAL(10, 2),  
    dataCadastro DATETIME  
);  
  
CREATE TABLE tb_pedido (  
    pedidoId INT PRIMARY KEY IDENTITY,  
    dataPedido DATETIME,  
    clienteId INT,  
    FOREIGN KEY (clienteId) REFERENCES tb_cliente(id)  
);
```

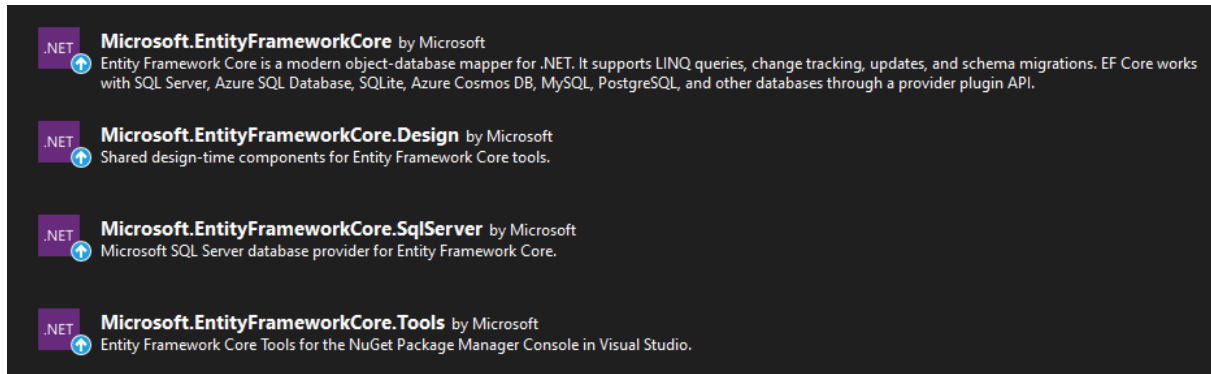
On the right, the 'Server Enterprise' tree shows the database 'dbo' with three tables: 'tb_cliente', 'tb_pedido', and 'tb_produto'. The 'tb_cliente' table has columns: id (PK, not null), nome (varchar(100), null), email (varchar(100), null), and telefone (varchar(20), null). The 'tb_pedido' table has columns: pedidoId (PK, not null), dataPedido (datetime, null), and clienteId (FK, null). The 'tb_produto' table has columns: id (PK, not null), nome (varchar(100), null), preco (decimal(10,2), null), and dataCadastro (datetime, null).

Foram criadas as tabelas `tb_cliente`, `tb_produto` & `tb_pedido` no sqlserver.

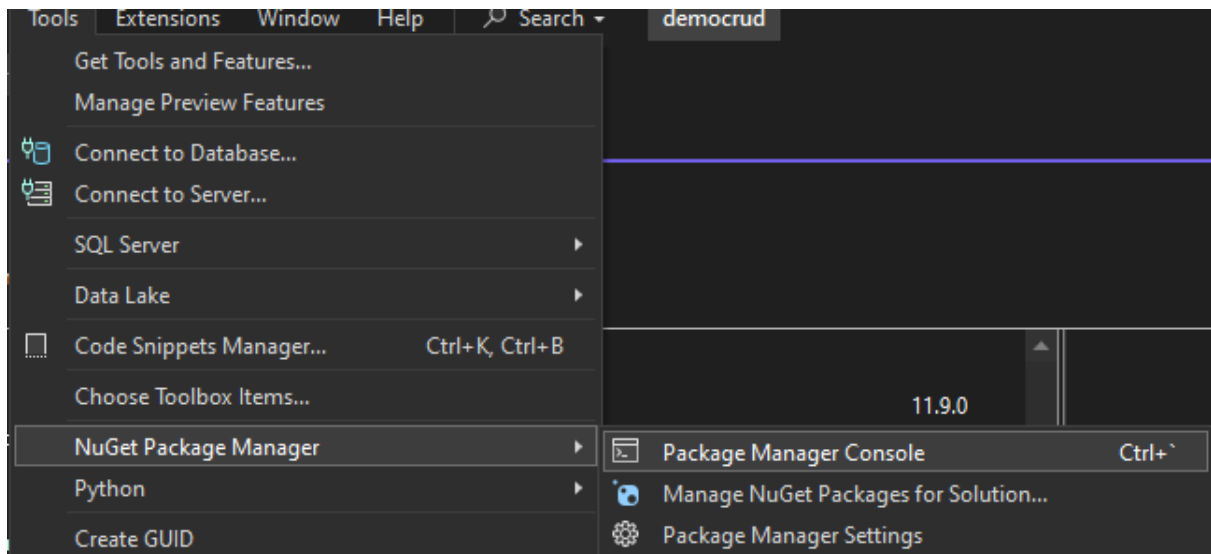
Agora indo para o código, será necessário os plugins:

- Microsoft.EntityFrameworkCore
- Microsoft.EntityFrameworkCore.Design
- Microsoft.EntityFrameworkCore.Tools

E possivelmente o `Microsoft.EntityFrameworkCore.SqlServer` (mas vai depender do BD utilizado)



Dentro do Visual Studio, vamos para `Tools > Nuget Package manager > Package Manager Console`



Dentro do terminal basta usar o comando `Scaffold-DbContext "SuaStringDeConexao" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models`

- "suastringdeconexao" - adicionar a string de conexão adequada
- "Microsoft.EntityFrameworkCore.SqlServer" - o provedor do BD, poderia ser mysql, postgres (`Pomelo.EntityFrameworkCore.MySql` ou

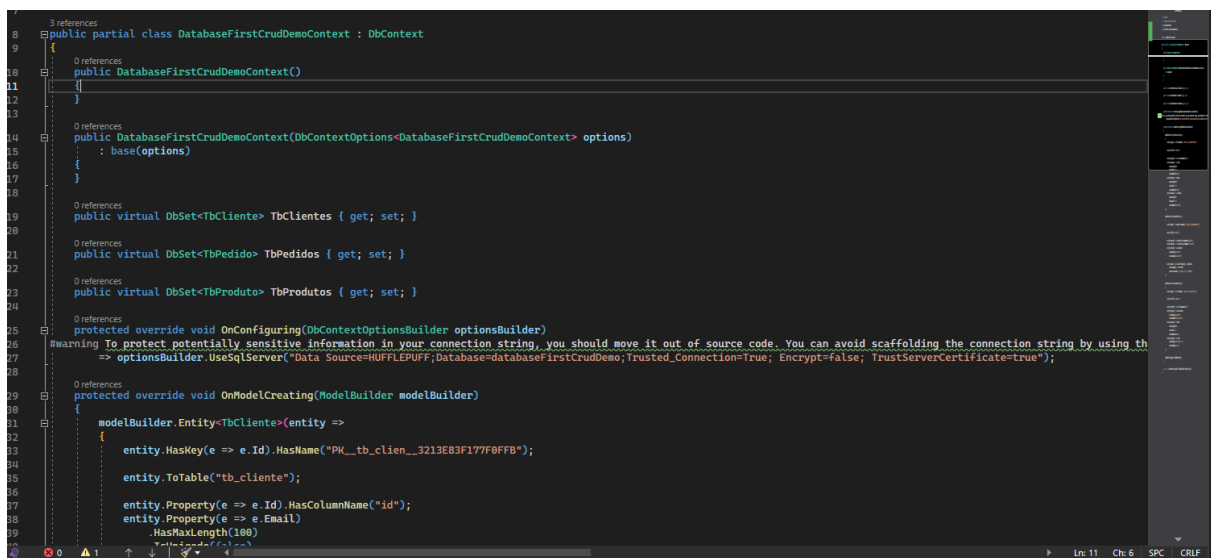
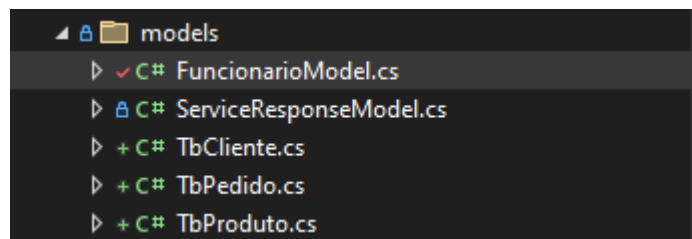
`Npgsql.EntityFrameworkCore.PostgreSQL`) ou outro.

- “-OutputDir Models” - esse parâmetro é OPCIONAL, porém especifica o diretório de saída na qual as classes serão criadas, nesse caso o Models

O comando final ficou:

```
Scaffold-DbContext "Data
Source=HUFFLEPUFF;Database=databaseFirstCrudDemo;Trusted_Connection=True; Encrypt=false;
TrustServerCertificate=true" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models
```

O resultado foi a criação das classes juntamente com a classe de Context configurando o banco de dados:



Agora é possível criar o CRUD em cima dessas classes.

Database-first SEM SCAFFOLD

→ Criar classe de contexto

- Criar conexão no Program.cs
- Fazer mapeamento das tabelas dentro do contexto
- Criar crud

1) Criar classe de contexto

```
3 references
5 public class ComercioContext : DbContext
6 {
7     0 references
8     public ComercioContext(DbContextOptions<ComercioContext> options) : base(options)
9     {
10         // criar tabelas
11     }
12 }
```

2) Criar conexão no Program.cs

```
31 // configurando o dbContext para utilizar mysql
32
33 builder.Services.AddDbContext<ComercioContext>(options =>
34 {
35     options.UseMySQL(builder.Configuration.GetConnectionString("MySQLConnectionString"), ServerVersion.AutoDetect("MySQLConnectionString"));
36 });
37
38
```

O trecho "MySQLConnectionString" está configurado dentro do appsettings

```
7 },
8 "ConnectionStrings": {
9     "DefaultConnection": "Data Source=HUFFLEPUFF;Database=FuncionariosWebApi;Trusted_Connection=True; Encrypt=false; TrustServerCertificate=true",
10     "MySQLConnectionString": "server=127.0.0.1;uid=root;database=databasefirstdemo;password=''"
11 },
12 }
```

3) Mapear as tabelas

```

1  using System.ComponentModel.DataAnnotations.Schema;
2
3  namespace democrud.models.ComercioModels
4  {
5      [Table("tb_cliente")]
6      public class Cliente
7      {
8          public int id { get; set; }
9          // [Column("name")]
10         public string name { get; set; }
11         public string email { get; set; }
12         public string telefone { get; set; }
13     }
14 }
15

```

```

1  using System.ComponentModel.DataAnnotations.Schema;
2
3  namespace democrud.models.ComercioModels
4  {
5      [Table("tb_produto")]
6      public class Produto
7      {
8          public int id { get; set; }
9          public string nome { get; set; }
10         public decimal preco { get; set; }
11     }
12 }
13

```

```

1  namespace democrud.models.ComercioModels
2  {
3      public class Venda
4      {
5          public int id { get; set; }
6          public int cliente_id { get; set; }
7          public int produto_id { get; set; }
8          public DateTime dataVenda { get; set; }
9      }
10 }
11

```

4) Agora referenciamos as classes dentro do Contexto

```
1 using democrud.models.ComercioModels;
2 using Microsoft.EntityFrameworkCore;
3
4 namespace democrud.DataContext
5 {
6     3 references
7     public class ComercioContext : DbContext
8     {
9         0 references
10        public ComercioContext(DbContextOptions<ComercioContext> options) : base(options)
11        {
12
13            //criar tabelas
14
15            0 references
16            public DbSet<Cliente> Cliente { get; set; }
17            0 references
18            public DbSet<Produto> Produto { get; set; }
19            0 references
20            public DbSet<Venda> Venda { get; set; }
21        }
22    }
```

Nota do chatgepeto:

A sobrescrita do método **OnModelCreating** é comumente usada na abordagem code-first para definir o mapeamento entre as entidades do aplicativo e as tabelas do banco de dados. No entanto, na abordagem database-first, o mapeamento já está definido no próprio banco de dados.

O método **OnConfiguring** geralmente é usado para configurar o provedor de banco de dados e a string de conexão. No entanto, na configuração do Startup, você já está configurando o contexto do banco de dados e a string de conexão usando a injeção de dependência, então não há necessidade de sobrescrever esse método na classe de contexto.

