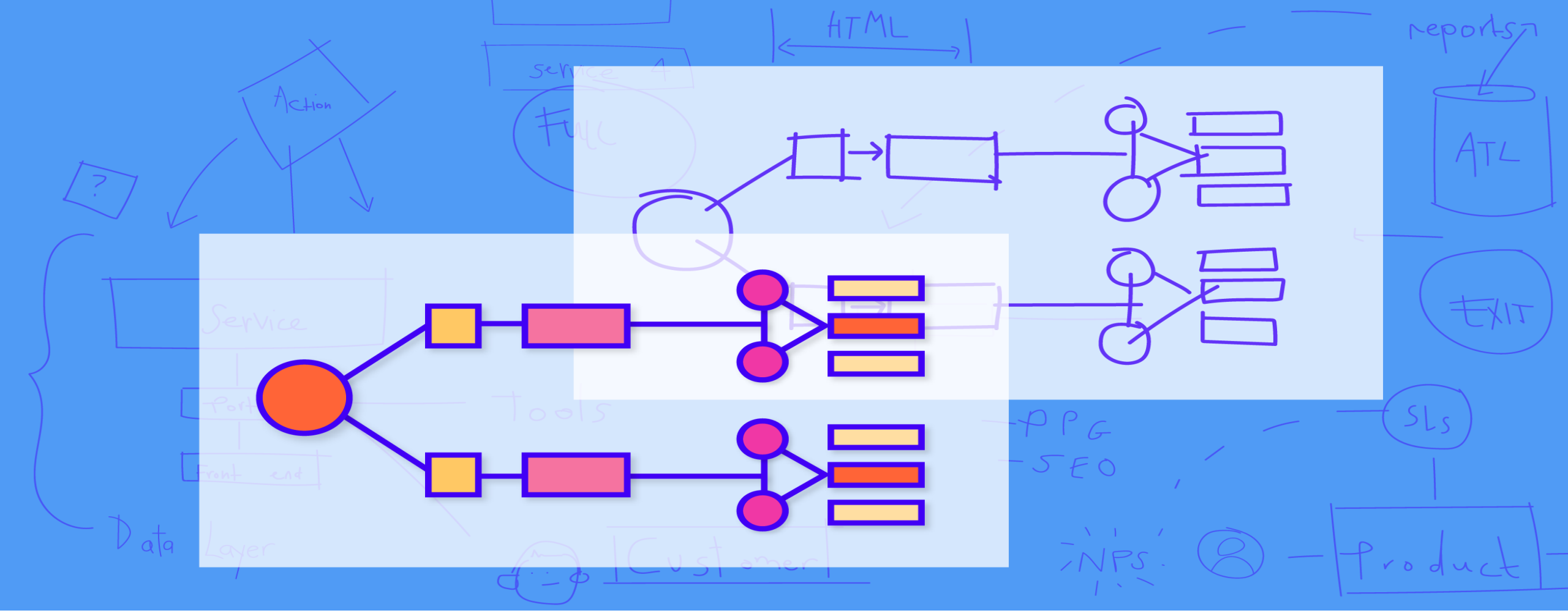


Projetos Multicamadas

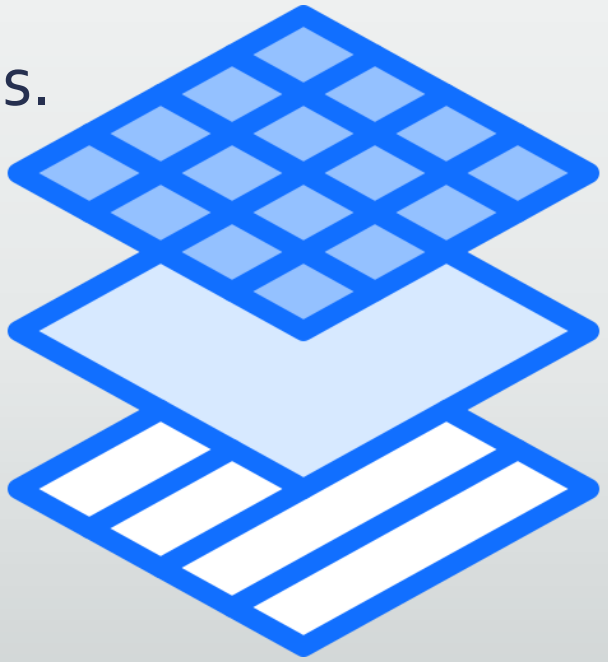
Programação III

Prof. Edson Mota, PhD, MSc, PMP

O que você entende por Programação Multicamadas?

Programação Multicamadas

- Os aplicativos multicamadas dividem a funcionalidade separando-as por agrupamentos lógicos
- Esses agrupamentos realizam operações e comunicam-se com as camadas adjacentes, gerando um fluxo de dados e processamento distribuído entre estas camadas.
- A programação orientada a objeto viabilizou o desenvolvimento multicamadas, tornando o desenvolvimento de aplicações uma verdadeira engenharia de componentes e ações.



Objetivos Principais

- Facilita a reutilização de código
- Facilita a manutenção e desenvolvimentos paralelos
- Consiste em isolar do código:
  - Acesso a Dados → Lógica de Negócio → Apresentação

    de forma a contribuir na manutenção dessas aplicações.

- Possibilita implementar uma camada contendo toda a regra de negócio da aplicação
- Expõe dados sempre no formato de objetos por meio de uma estratégia conhecida como Mapeamento Objeto ←→ Relacional

Vantagens

- Desacoplamento
- Processamento distribuído
- Componentização
- Escalabilidade
- Extensibilidade
- Manutenabilidade
- Paralelização de atividades

Desvantagens

- Aumento significativo do número de classes
- Requer mais esforço na elaboração da arquitetura
- Entendimento consistente dos fundamentos da orientação a objeto
- Maior esforço no desenvolvimento, menor esforço na manutenção

Modelo de Alto Nível

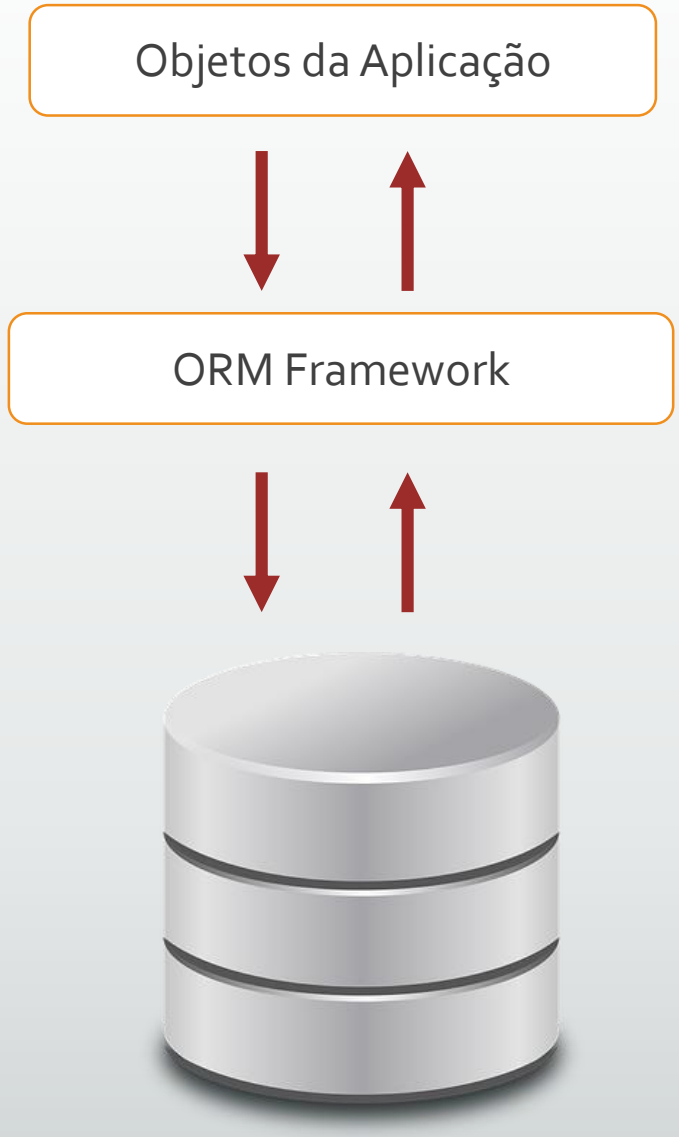
Camada de Apresentação

Camada de Negócios / Serviços

Camada de Acesso a Dados

Um Idioma Comum a Todos!

- O desenvolvimento multicamadas requer que todas as camadas "conversem", utilizando um conjunto de componentes comuns, compartilhados entre os elementos
- Estes componentes são chamados de Modelos
- Estes modelos ajudam a interpretar os dados recebidos e repassá-los a camadas adjacentes na forma de objetos
- Trata-se do mapeamento objeto-relacional

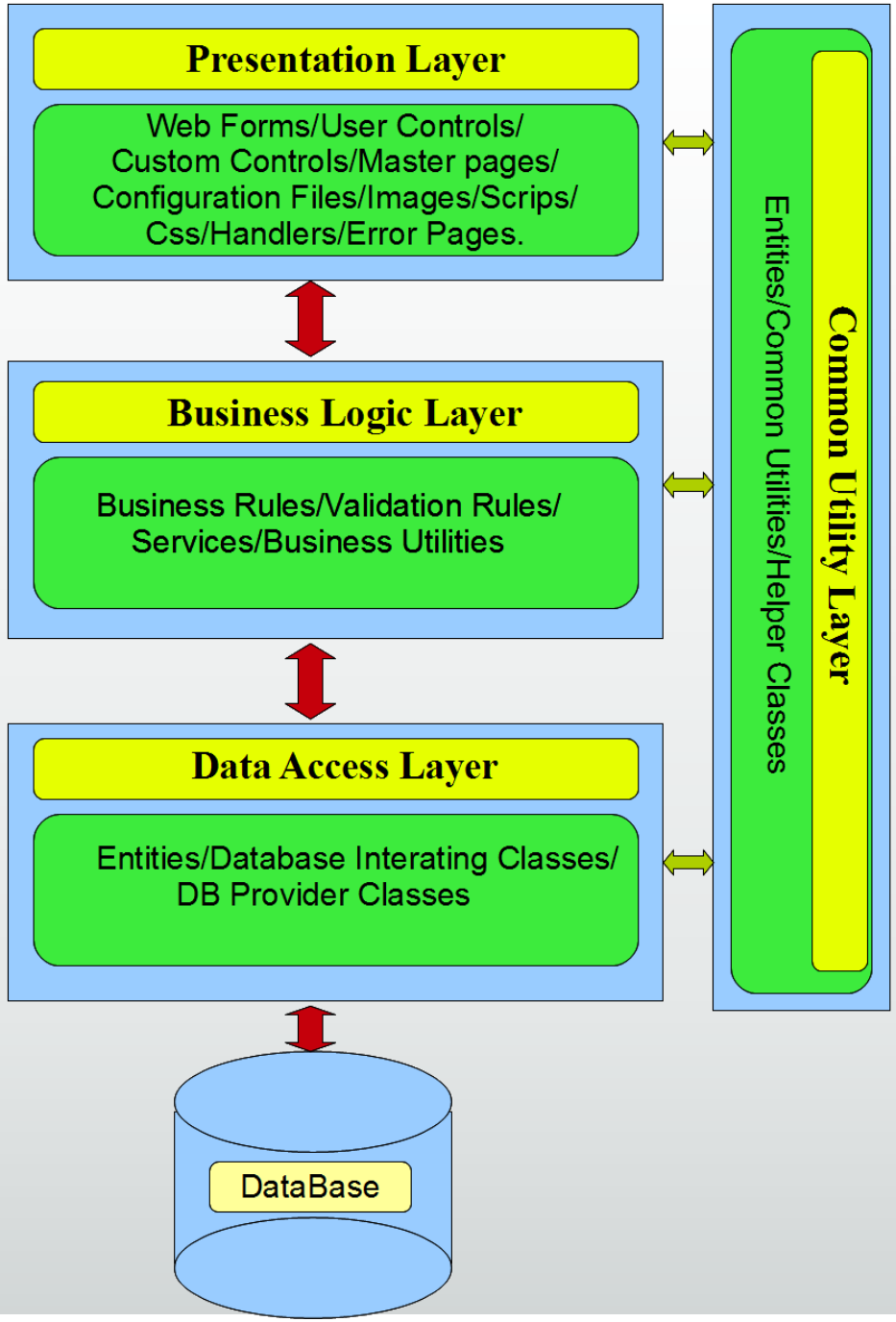


Camada de Apresentação: responsável por descrever o layout e apenas chamar as funções e regras desenvolvidas nas camadas inferiores.

Camada de Negócio, também chamada de Repository: Garante que as regras estejam disponíveis, independentemente da aplicação.

Camada de Abstração do Banco de Dados: Garante o desacoplamento com a infraestrutura de dados.

Banco de dados, armazenamento e persistência de dados



Camada de dados comum à toda a aplicação. Garante que todas as camadas conheçam entidades e atributos trafegados.

Também chamada de Model

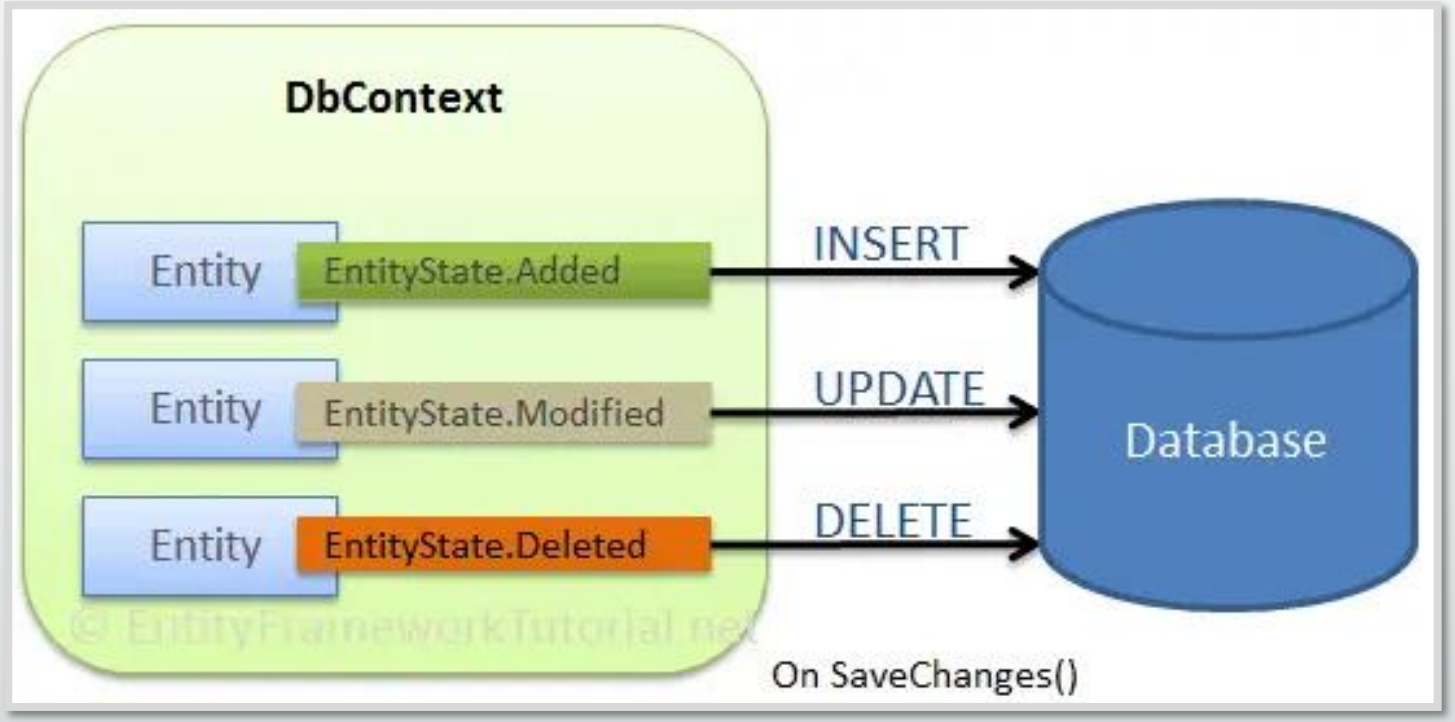
EntityFramework

- EntityFramework é um framework de acesso a dados desenvolvido pela Microsoft que contribui para pavimentar o gap que temos entre estruturas de dados convencionais e os objetos de nossas aplicações.
- Assim, EntityFramework é capaz de:
  - Gerar entidade de objetos fortemente tipados que podem ser customizados
  - Gerar o código (mapping/plumbing) responsável por estruturar a relação ORM
  - Traduzir queries para o banco de dados
  - Materializar objetos a partir de chamadas do serviço de persistência
  - Rastrear mudanças geradas a partir de manipulações “update” e “insert”

https://www.entityframeworktutorial.net/crud-operation-in-connected-scenario-entity-framework.aspx

DbContext

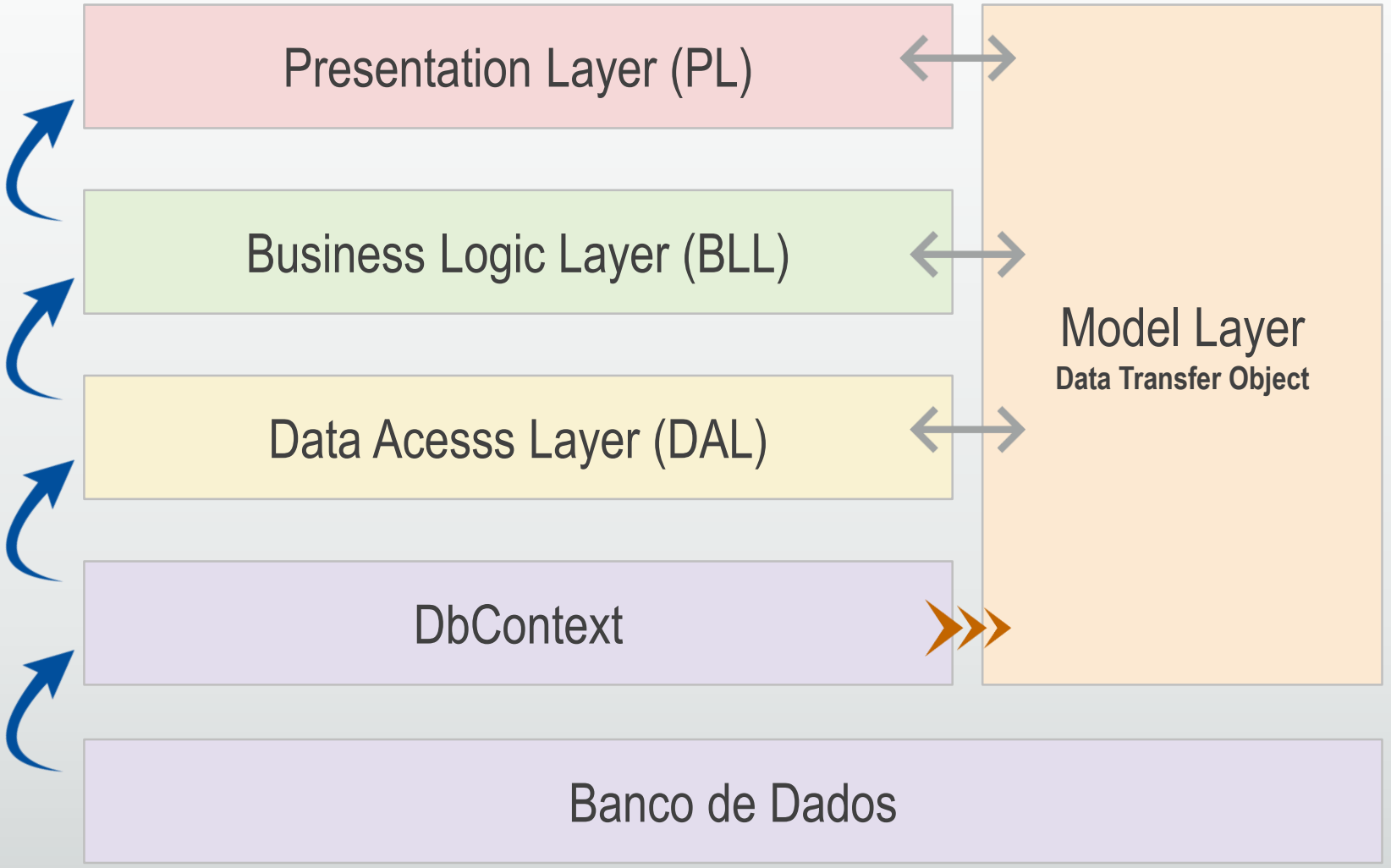
- Em EntityFramework, o acesso a dados é operacionalizado pela classe DbContext



- DbContext se utiliza de objetos DbSet para realizar o mapeamento
- Uma vez mapeados as entidades, a infraestrutura do EntityFramework é capaz de realizar a manutenção dos dados por meio do framework .

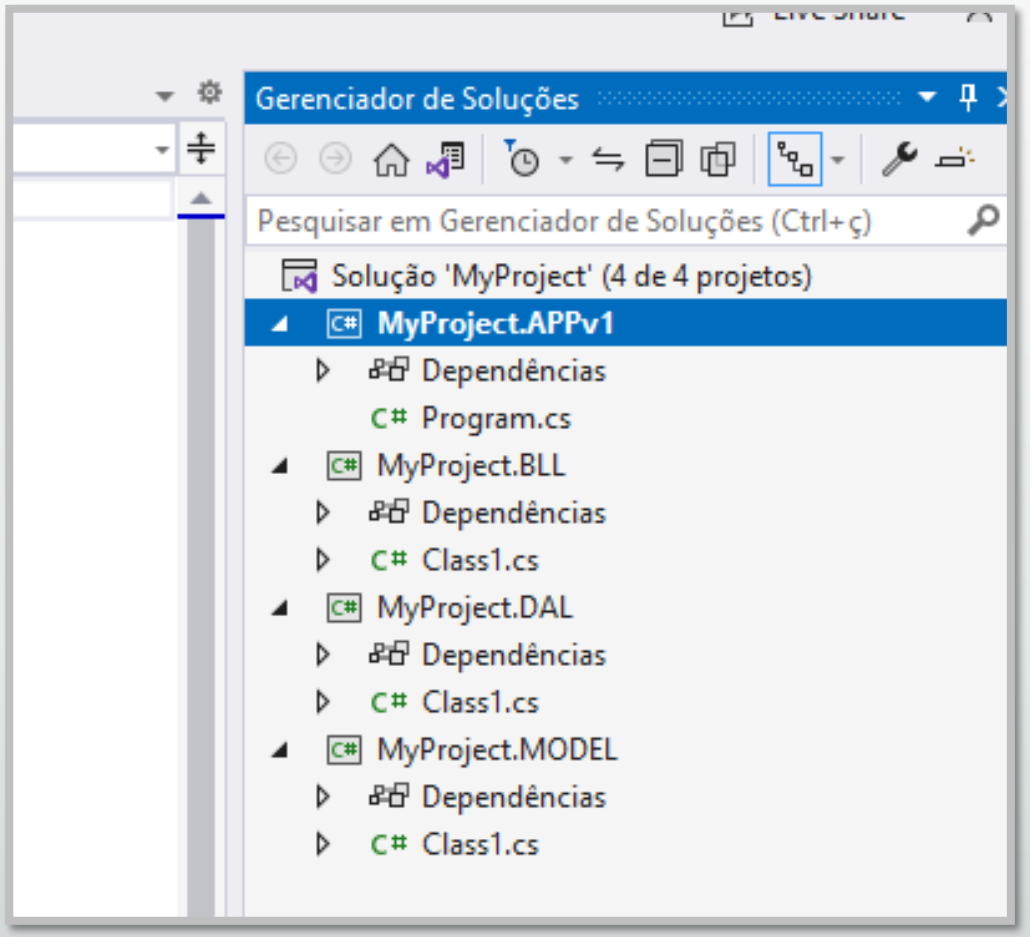
Vamos começar?

Nossa Implementação



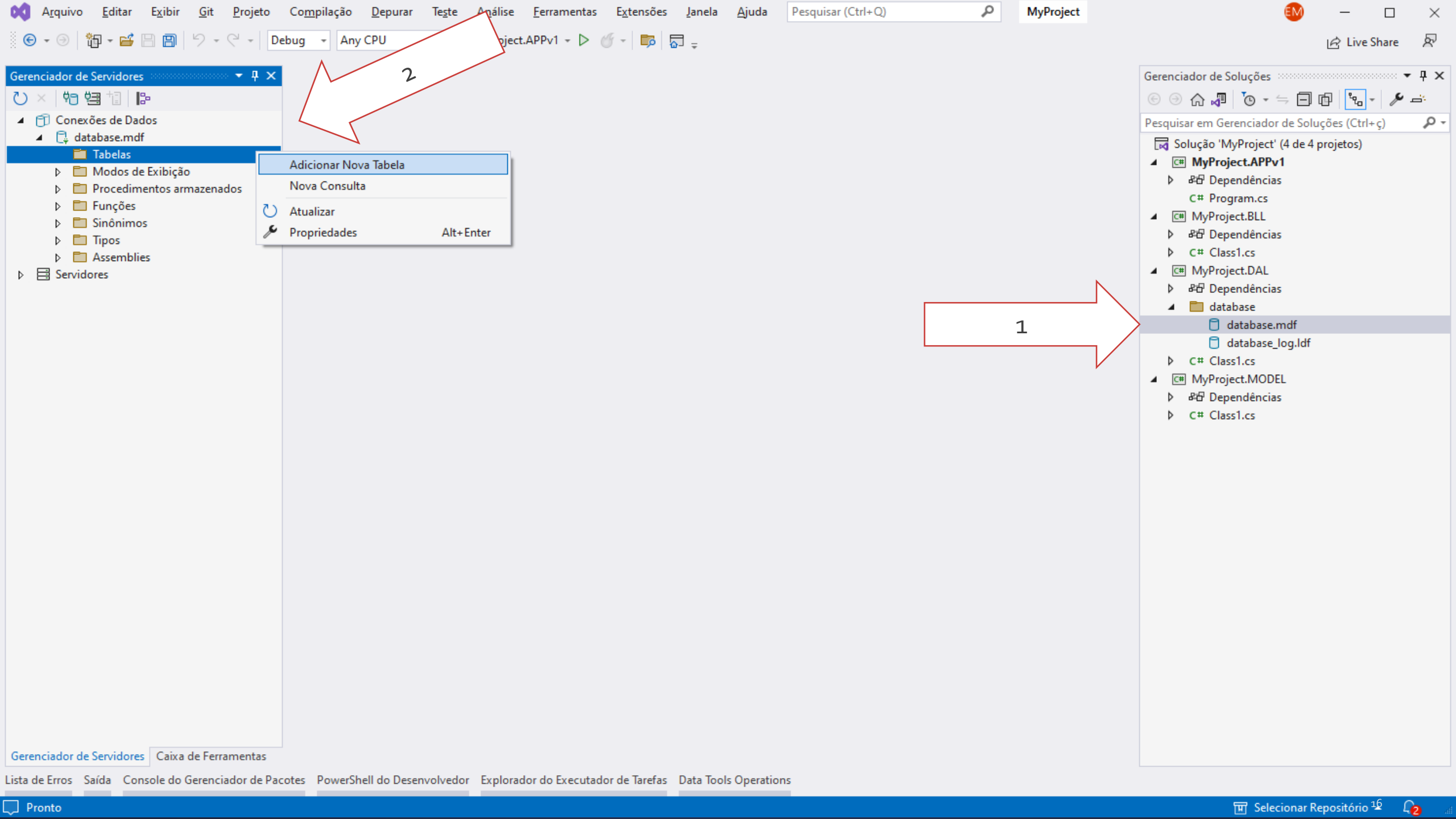
Criando o Projeto

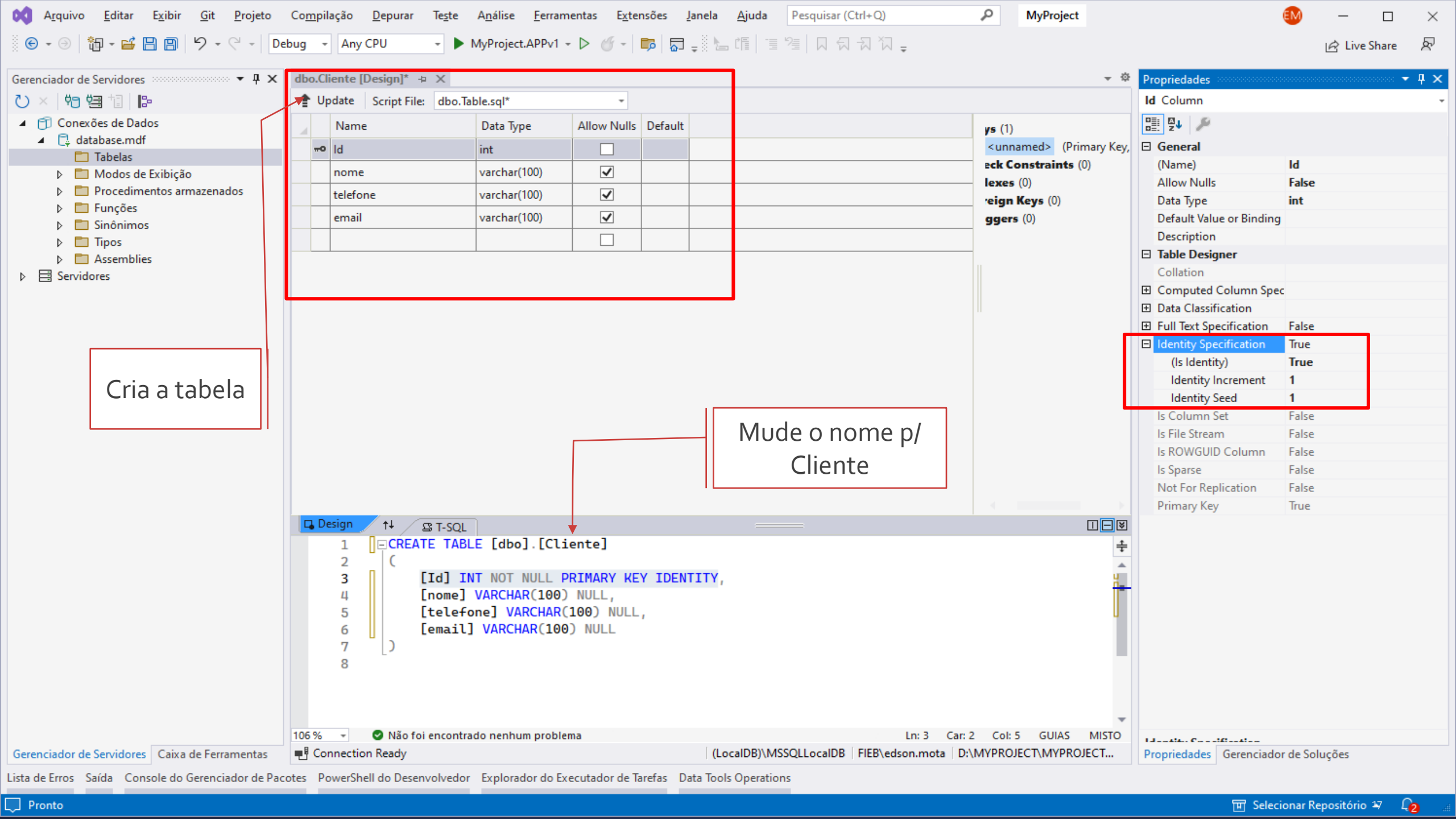
- Crie um projeto com o template "Solução em Branco"
- Com esse template criado, vamos criar 4 projetos
  - **MyProject.APPv1**
    - Camada de Apresentação
    - Projeto Console ou Windows Form
  - **MyProject.BLL**
    - Camada de Negócio
    - Projeto Biblioteca de Classes
  - **MyProject.DAL**
    - Camada de Acessos a Dados
    - Projeto Biblioteca de Classes
  - **MyProject.MODEL**
    - Camada para Modelo de Dados
    - Projeto Biblioteca de Classes



Configurando o Projeto DAL

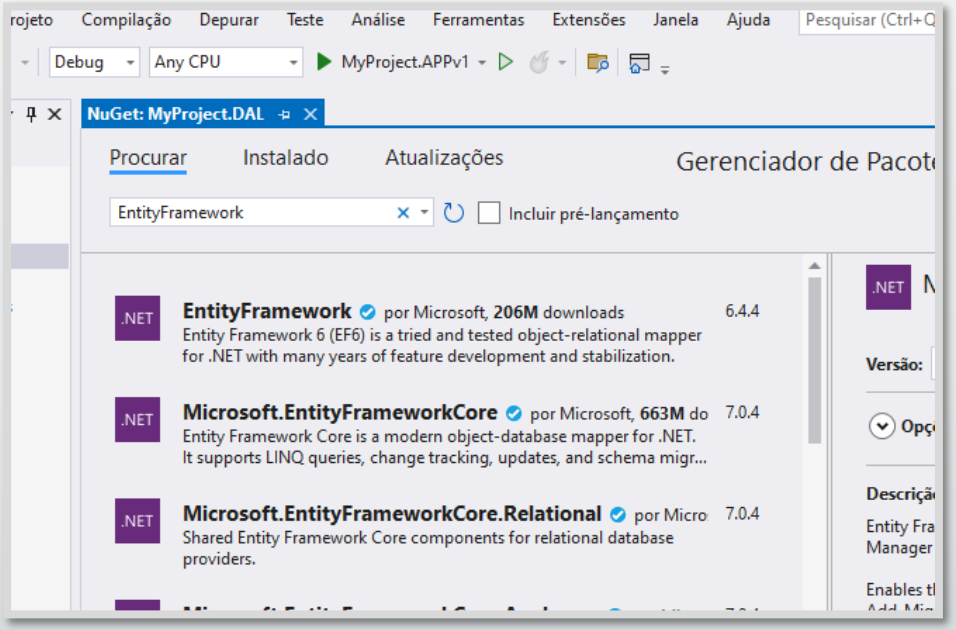
- Primeiro, vamos criar o banco de dados
  - Crie uma pasta “database” em MyProject.DAL
  - Botão direito na pasta e adicione um objeto do tipo: “Banco de Dados Baseado em Serviço”, chame o banco de dados de “database.mdf”
  - Duplo clique no Banco levará ao Gerenciador de Servidores, onde vamos configurar o banco


Arquivo Editar Exibir Git Projeto Compilação Depurar Teste Análise Ferramentas Extensões Janela Ajuda
Pesquisar (Ctrl+Q)
MyProject
Debug
Any CPU
Live Share
Gerenciador de Servidores
Conexões de Dados
database.mdf
Tabelas
Modos de Exibição
Procedimentos armazenados
Funções
Sinônimos
Tipos
Assemblies
Servidores
2
Adicionar Nova Tabela
Nova Consulta
Atualizar
Propriedades
Alt+Enter
1
Gerenciador de Soluções
Pesquisar em Gerenciador de Soluções (Ctrl+ç)
Solução 'MyProject' (4 de 4 projetos)
MyProject.APPv1
Dependências
Program.cs
MyProject.BLL
Dependências
Class1.cs
MyProject.DAL
Dependências
database
database.mdf
database_log.ldf
Class1.cs
MyProject.MODEL
Dependências
Class1.cs
Gerenciador de Servidores
Caixa de Ferramentas
Lista de Erros
Saída
Console do Gerenciador de Pacotes
PowerShell do Desenvolvedor
Explorador do Executador de Tarefas
Data Tools Operations
Pronto
Selecionar Repositório


Arquivo Editar Exibir Git Projeto Compilação Depurar Teste Análise Ferramentas Extensões Janela Ajuda
Pesquisar (Ctrl+Q)
MyProject
Debug
Any CPU
MyProject.APPv1
Live Share
Gerenciador de Servidores
Conexões de Dados
database.mdf
Tabelas
Modos de Exibição
Procedimentos armazenados
Funções
Sinônimos
Tipos
Assemblies
Servidores
dbo.Cliente [Design]*
Update
Script File: dbo.Table.sql*
Name
Data Type
Allow Nulls
Default
Id
int
nome
varchar(100)
telefone
varchar(100)
email
varchar(100)
Cria a tabela
Mude o nome p/ Cliente
Design
T-SQL
CREATE TABLE [dbo].[Cliente]
(
[Id] INT NOT NULL PRIMARY KEY IDENTITY,
[nome] VARCHAR(100) NULL,
[telefone] VARCHAR(100) NULL,
[email] VARCHAR(100) NULL
)
Não foi encontrado nenhum problema
Ln: 3 Car: 2 Col: 5 GUIAS MISTO
Connection Ready
(LocalDB)\MSSQLLocalDB
FIEB\edson.mota
D:\MYPROJECT\MYPROJECT...
Propriedades
Id Column
General
(Name) Id
Allow Nulls False
Data Type int
Default Value or Binding
Description
Table Designer
Collation
Computed Column Spec
Data Classification
Full Text Specification False
Identity Specification True
(Is Identity) True
Identity Increment 1
Identity Seed 1
Is Column Set False
Is File Stream False
Is ROWGUID Column False
Is Sparse False
Not For Replication False
Primary Key True
Gerenciador de Servidores
Caixa de Ferramentas
Propriedades
Gerenciador de Soluções
Lista de Erros
Saída
Console do Gerenciador de Pacotes
PowerShell do Desenvolvedor
Explorador do Executador de Tarefas
Data Tools Operations
Pronto
Selecionar Repositório

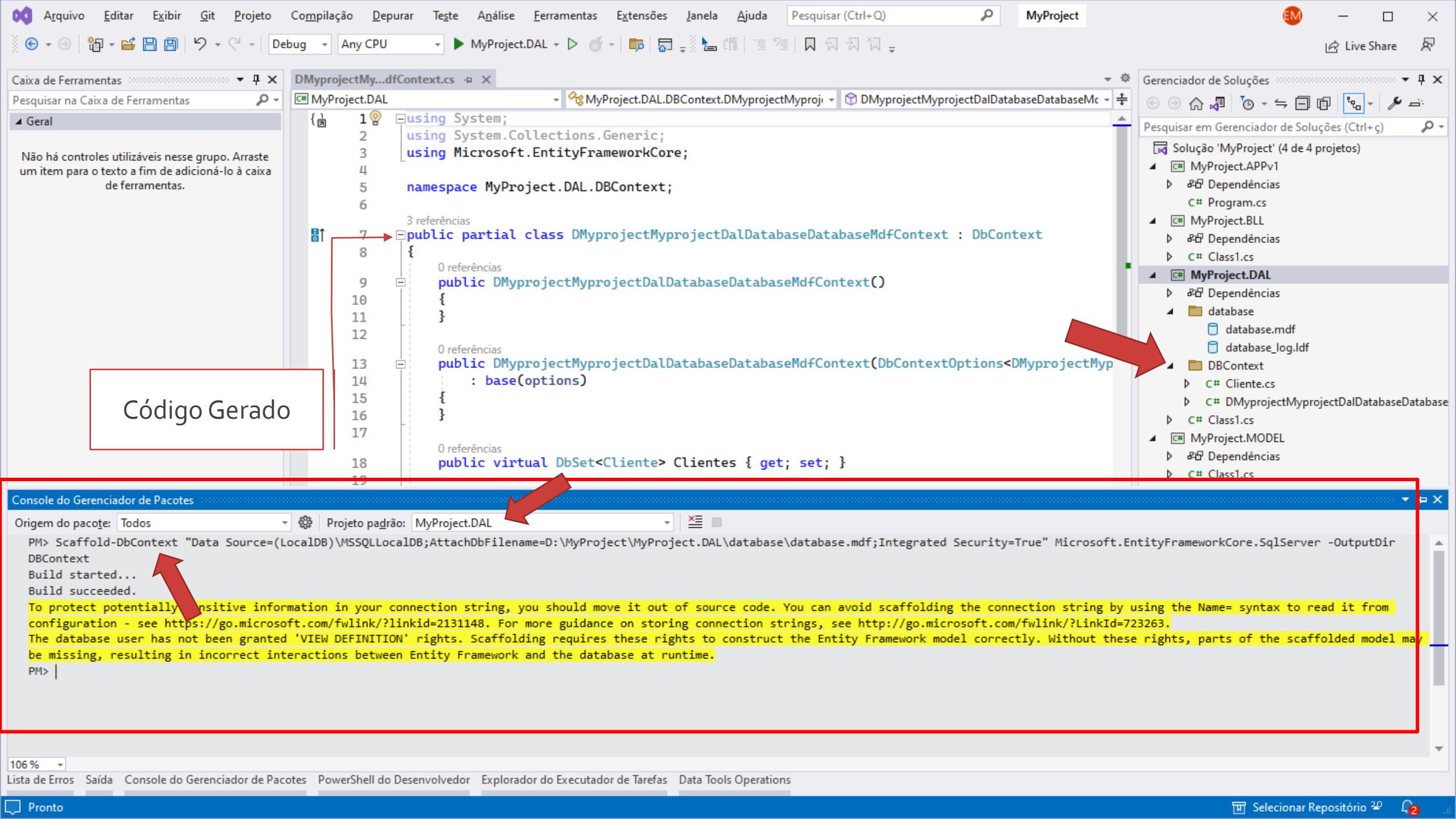
Configurando o Projeto DAL

- Importantes dependências
  - Botão direito em MyProject.DAL → Gerenciar Pacotes do NuGet
  - Na aba Procurar, **localize** e **instale** as seguintes extensões
    - **EntityFramework**
    - **Microsoft. EntityFrameworkCore**
    - **Microsoft. EntityFrameworkCore.SQLServer**
    - **Microsoft. EntityFrameworkCore.Tools**



Configurando o Projeto DAL

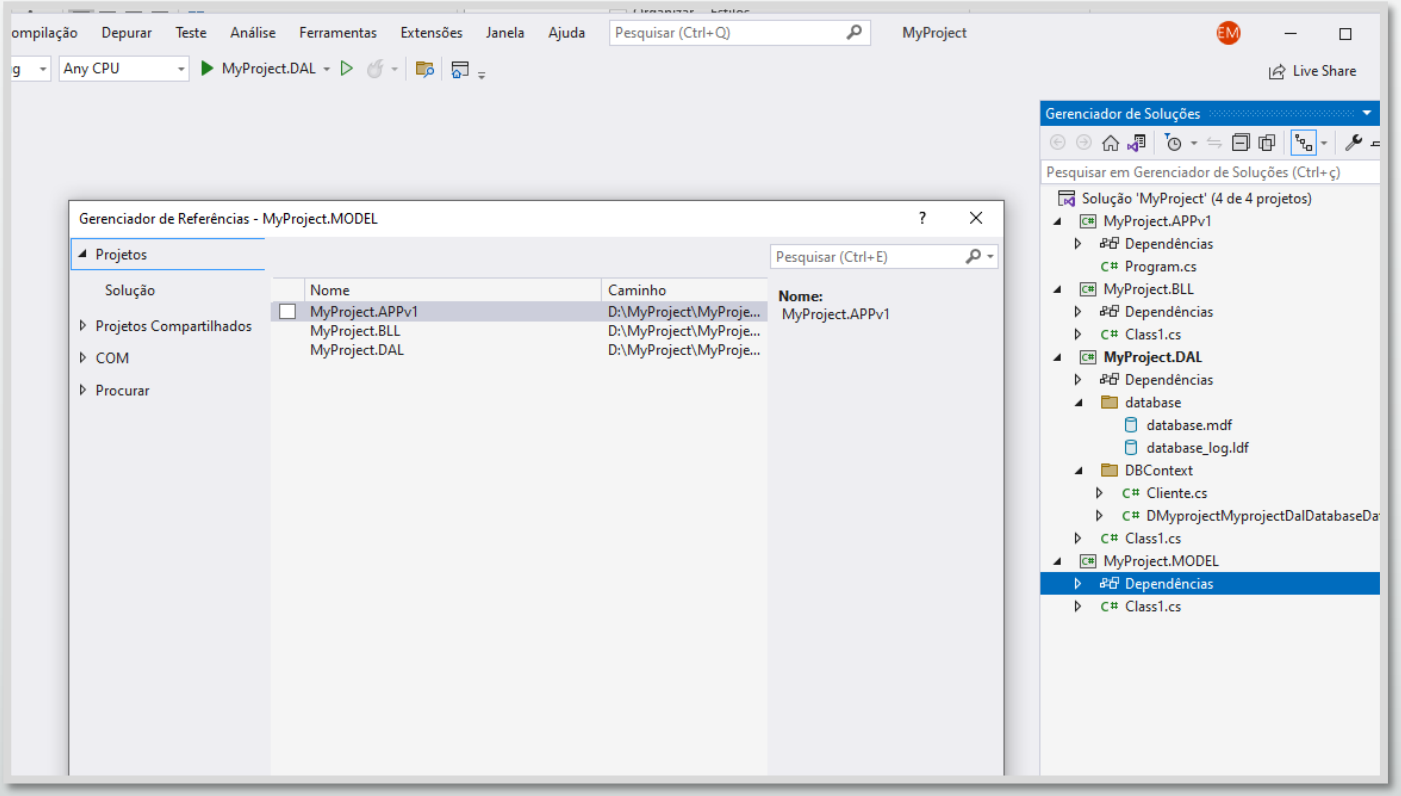
- Gerando códigos da camada modelo com: **Scaffold-DbContext**
- Para fazer isso, precisamos rodar um comando no prompt do Visual Studio.NET
  - Localize a janela **Console do Gerenciador de Pacotes**
  - Se não estiver aberta, acesse exibir → Outras Janelas → Console do Gerenciador de Pacotes
  - Na janela , selecione MyProject.DAL como projeto Padrão
  - Defina MyProject.DAL como projeto de inicialização (botão direito, definir como projeto de inicialização)
  - Depois disso, execute o seguinte comando:
    - Scaffold-DbContext "Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:**caminho**\database.mdf;Integrated Security=True" Microsoft.EntityFrameworkCore.SqlServer -OutputDir DBContext


Arquivo Editar Exibir Git Projeto Compilação Depurar Teste Análise Ferramentas Extensões Janela Ajuda
Pesquisar (Ctrl+Q)
MyProject
Debug
Any CPU
MyProject.DAL
Live Share
Caixa de Ferramentas
Pesquisar na Caixa de Ferramentas
Geral
Não há controles utilizáveis nesse grupo. Arraste um item para o texto a fim de adicioná-lo à caixa de ferramentas.
DMyprojectMy...dfContext.cs
MyProject.DAL
MyProject.DAL.DBContext.DMyprojectMyproj
DMyprojectMyprojectDalDatabaseDatabaseMc
1 using System;
2 using System.Collections.Generic;
3 using Microsoft.EntityFrameworkCore;
4
5 namespace MyProject.DAL.DBContext;
6
3 referências
7 public partial class DMyprojectMyprojectDalDatabaseDatabaseMdfContext : DbContext
8 {
0 referências
9 public DMyprojectMyprojectDalDatabaseDatabaseMdfContext()
10 {
11 }
12
0 referências
13 public DMyprojectMyprojectDalDatabaseDatabaseMdfContext(DbContextOptions<DMyprojectMyp
14 : base(options)
15 {
16 }
17
0 referências
18 public virtual DbSet<Cliente> Clientes { get; set; }
Código Gerado
Gerenciador de Soluções
Pesquisar em Gerenciador de Soluções (Ctrl+ç)
Solução 'MyProject' (4 de 4 projetos)
MyProject.APPv1
Dependências
Program.cs
MyProject.BLL
Dependências
Class1.cs
MyProject.DAL
Dependências
database
database.mdf
database_log.ldf
DBContext
Cliente.cs
DMyprojectMyprojectDalDatabaseDatabase
Class1.cs
MyProject.MODEL
Dependências
Class1.cs
Console do Gerenciador de Pacotes
Origem do pacote: Todos
Projeto padrão: MyProject.DAL
PM> Scaffold-DbContext "Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=D:\MyProject\MyProject.DAL\database\database.mdf;Integrated Security=True" Microsoft.EntityFrameworkCore.SqlServer -OutputDir DBContext
Build started...
Build succeeded.
To protect potentially sensitive information in your connection string, you should move it out of source code. You can avoid scaffolding the connection string by using the Name= syntax to read it from configuration - see https://go.microsoft.com/fwlink/?linkid=2131148. For more guidance on storing connection strings, see http://go.microsoft.com/fwlink/?LinkId=723263.
The database user has not been granted 'VIEW DEFINITION' rights. Scaffolding requires these rights to construct the Entity Framework model correctly. Without these rights, parts of the scaffolded model may be missing, resulting in incorrect interactions between Entity Framework and the database at runtime.
PM>
106 %
Lista de Erros Saída Console do Gerenciador de Pacotes PowerShell do Desenvolvedor Explorador do Executador de Tarefas Data Tools Operations
Pronto
Selecionar Repositório

Configurando as Referências

- Para que todos estes projetos estejam conectados é preciso adicionar as suas referências

- A adição de referência funciona da seguinte forma:
  - Com o projeto aberto, no item dependências, clique com o botão direito em Adicionar Referências

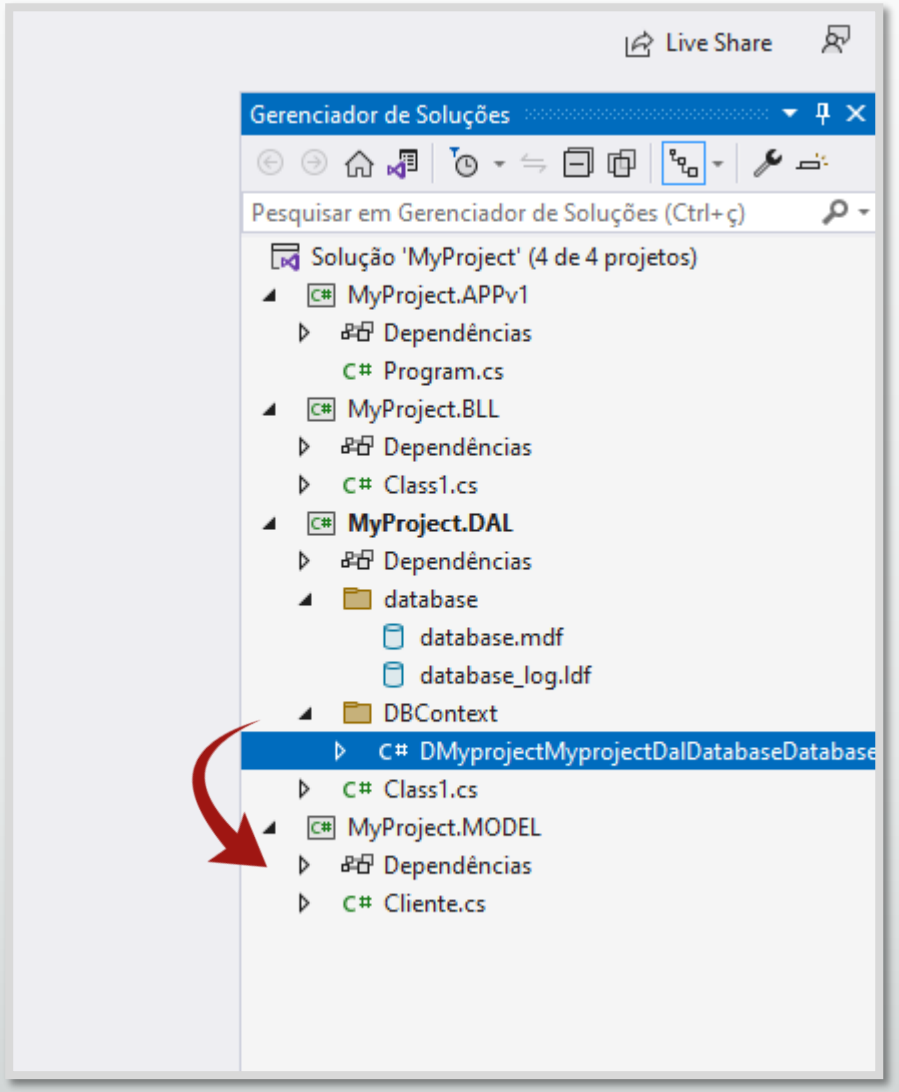


Configurando as Referências

- As referência seguem a seguinte lógica:
  - **MyProject.APP**
    - MyProject.BLL
    - MyProject.MODEL
  - **MyProject.BLL**
    - MyProject.MODEL
    - MyProject.DAO
  - **MyProject.DAL**
    - MyProject.Model

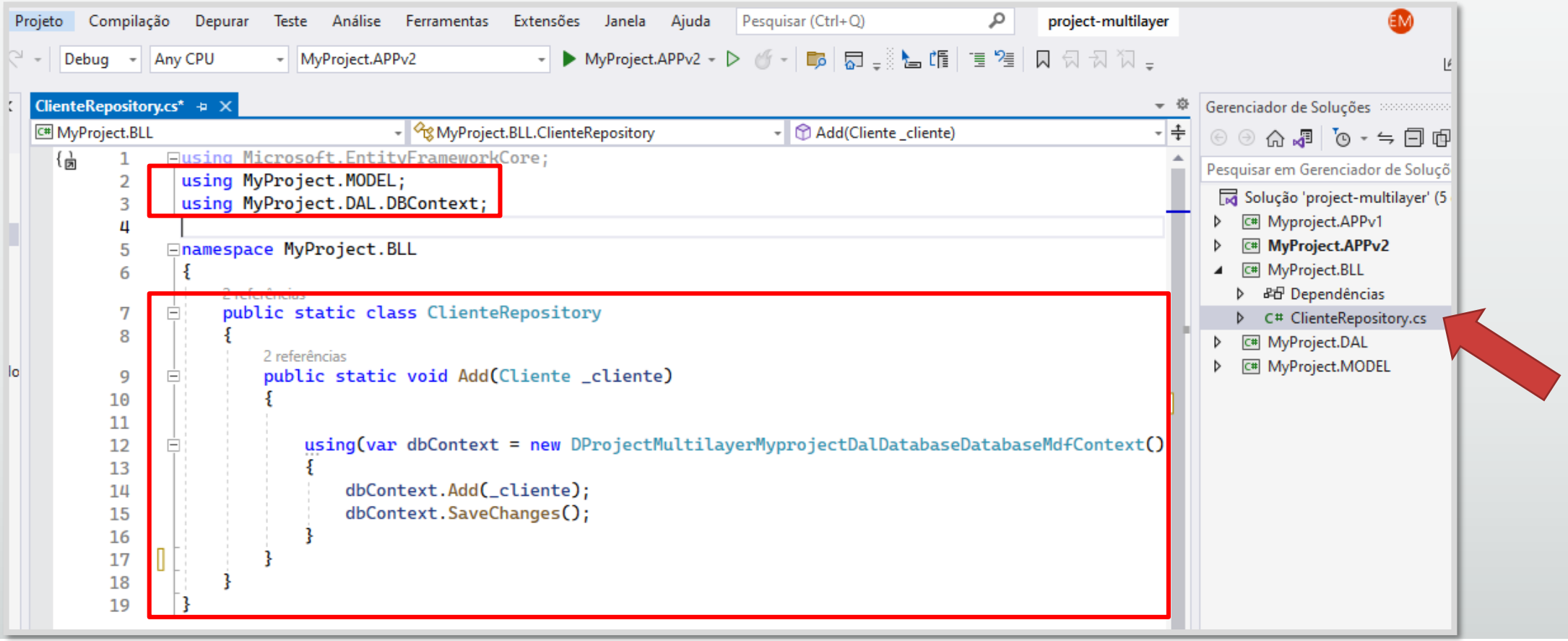
Configurando o Projeto Model

- O projeto Model é responsável por descrever as entidades e atributos utilizados na aplicação.
- Entretanto, esse código já foi gerado pelo Scafold, então podemos apenas copiar para a classe model
- Recorte a classe Cliente da pasta DBContext do Projecto MyProject.DAL e cole na raiz do projeto **MyProject.MODEL**
- Depois disso, abra a classe Cliente.cs e modifique o namespace para **MyProject.MODEL**
- Por fim, no projeto MyProject.DAL abra o arquivo de contexto "DMyprojectMyprojectDalDatabaseDatabaseMdfContext.cs" e INCLUA o namespace **de** MyProject.DAL.DBContext **para** namespace **MyProject.MODEL**
- Apague o Class1.cs gerado com o projeto.



Configurando o Projeto BLL – Business Logic

- Altere o Nome da classe para "ClienteRepository.cs"
- Dentro dessa classe, vamos acessar o projeto DAL e realizar a adição de um registro



GetById(int Id)

- Permite recuperar os dados por ID

```
0 referências
public static Cliente GetById(int Id)
{

    using (var dbContext = new DProjectMultilayerMyprojectDalDatabaseDatabaseMdfContext())
    {
        var cliente = dbContext.Clientes.Single(p => p.Id == Id);
        return cliente;

    }
}
```

GetAll()

- Recupera todos os dados da Tabela Cliente

```
2 referências
public static List<Cliente> GetAll()
{

    using (var dbContext = new DProjectMultilayerMyprojectDalDatabaseDatabaseMdfContext())
    {
        var cliente = dbContext.Clientes.ToList();
        return cliente;

    }
}
```

Update(Cliente _cliente)

- Permite a edição dos dados de um Cliente

```
2 referências
public static void Update(Cliente _cliente)
{
    using (var dbContext = new DProjectMultilayerMyprojectDalDatabaseDatabaseMdfContext())
    {
        var cliente = dbContext.Clientes.Single(p => p.Id == _cliente.Id);
        cliente.Nome = _cliente.Nome;
        cliente.Email = _cliente.Email;
        cliente.Telefone = _cliente.Telefone;
        dbContext.SaveChanges();
    }
}
```

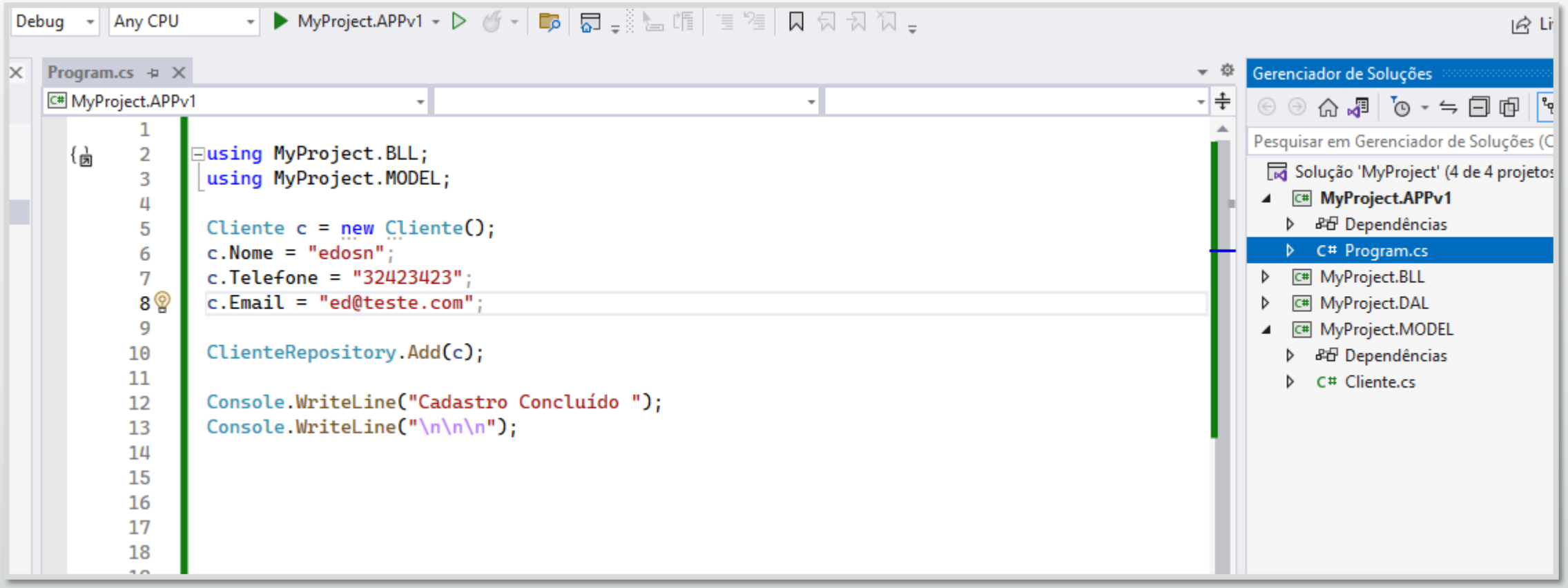
Excluir(Cliente _cliente)

- Recupera todos os dados da Tabela Cliente

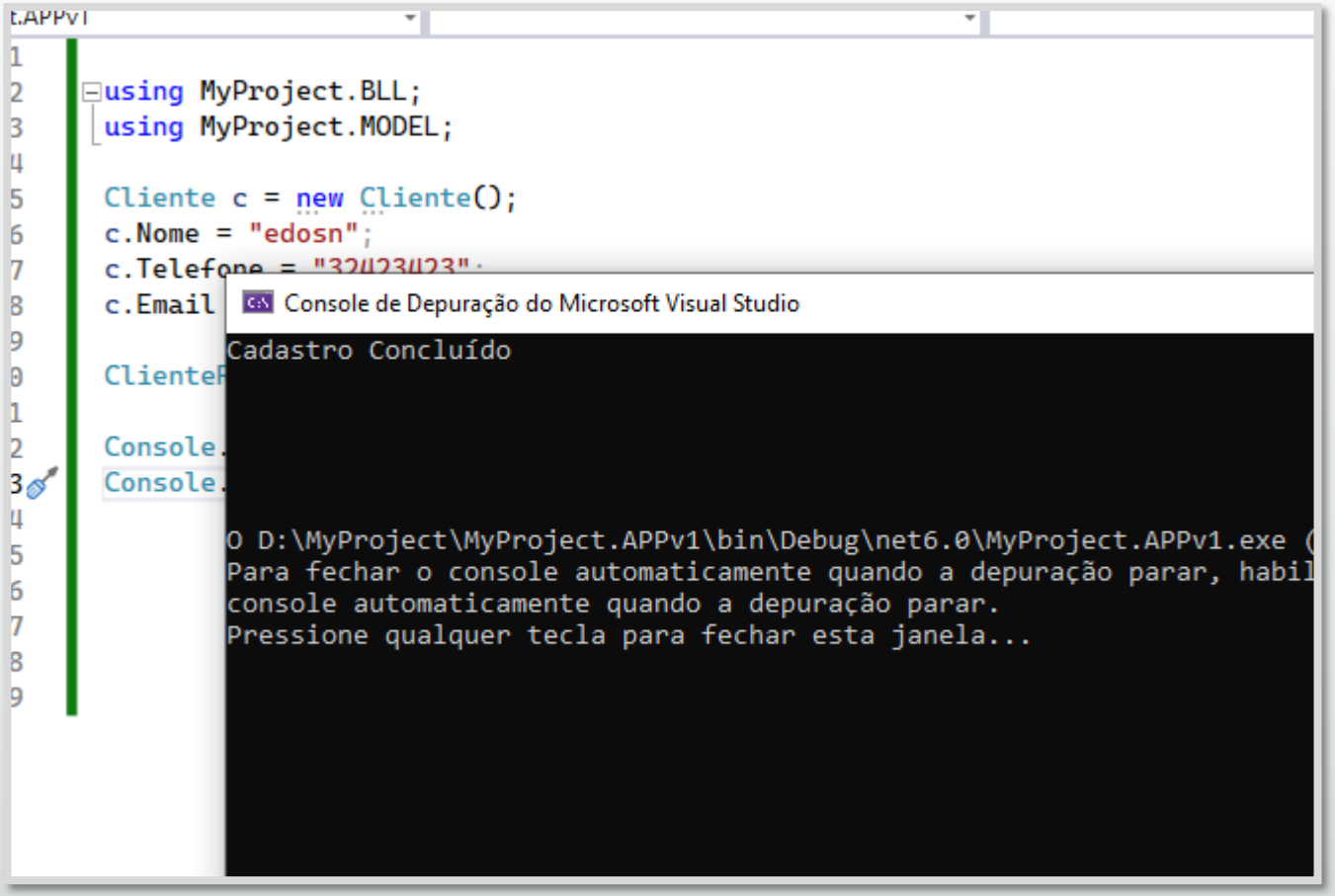
```
2 referências
public static void Excluir(Cliente _cliente)
{
    using (var dbContext = new DProjectMultilayerMyprojectDalDatabaseDatabaseMdfContext())
    {
        var cliente = dbContext.Clientes.Single(p => p.Id == _cliente.Id);
        dbContext.Remove(cliente);
        dbContext.SaveChanges();
    }
}
```

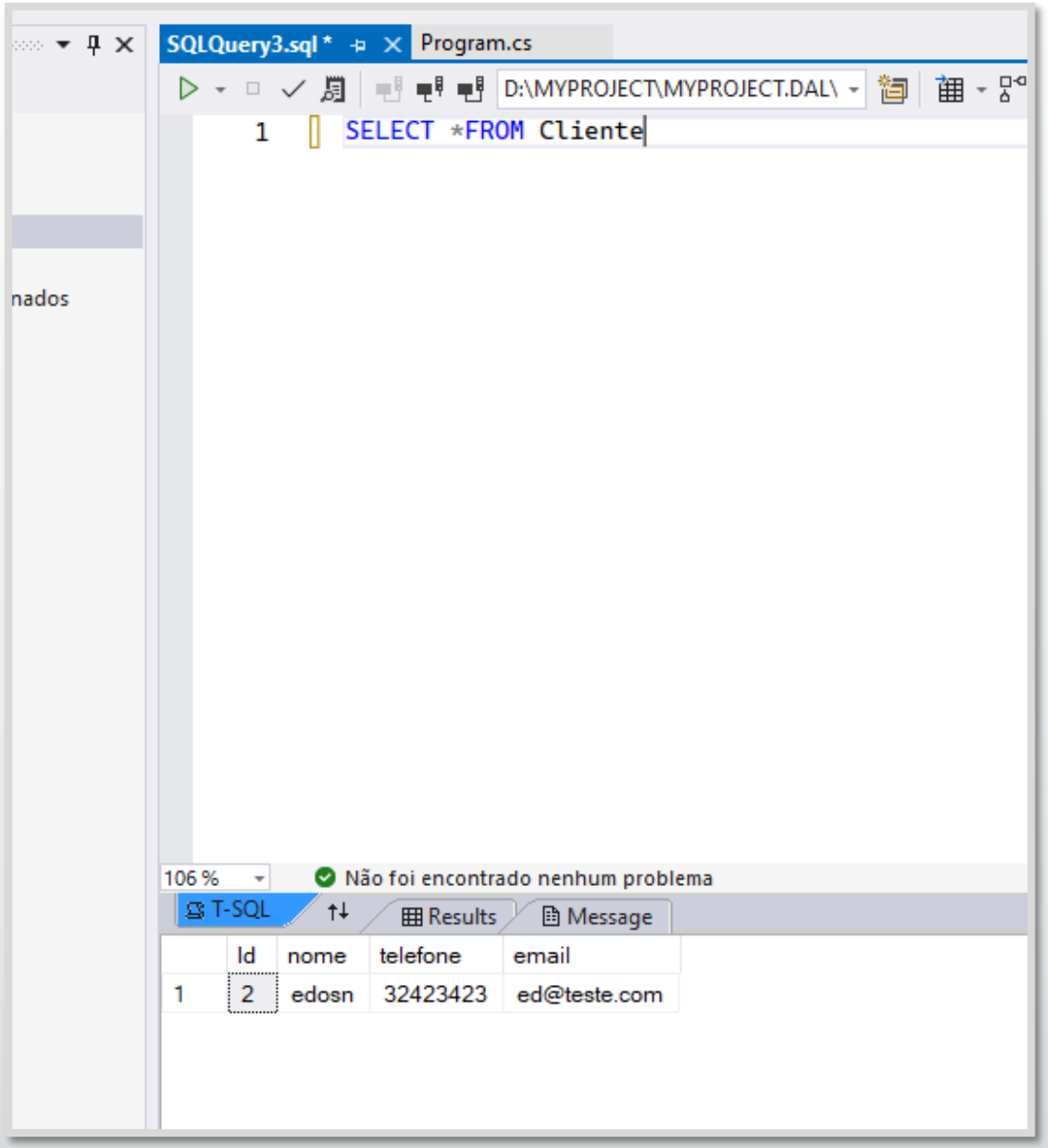
Configurando o Projeto APL – Camada de Aplicação

- Aplicação de Console, precisamos apenas acessar a camada de negócio e passar o objeto que desejamos persistir ou recuperar do banco de dados

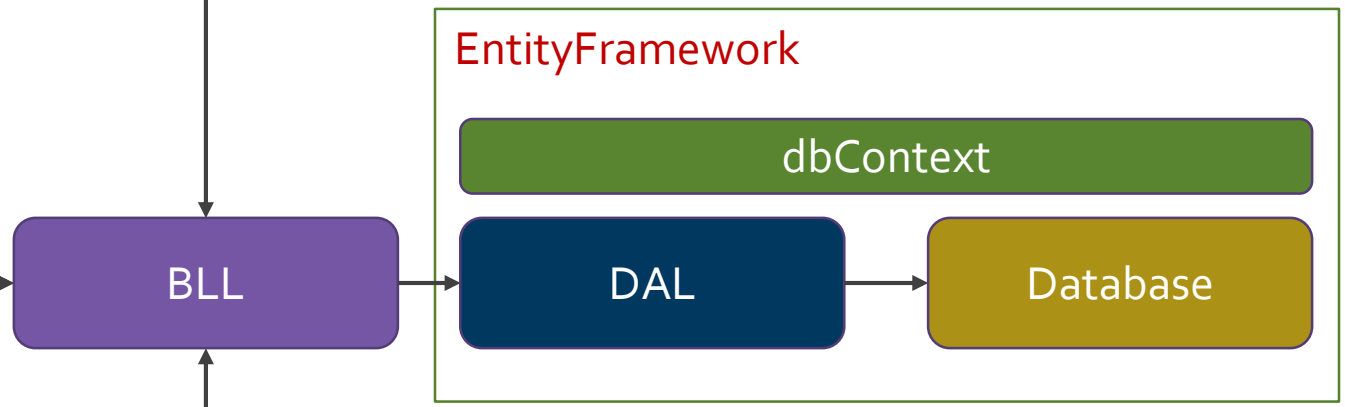


Executando


using MyProject.BLL;
using MyProject.MODEL;
Cliente c = new Cliente();
c.Nome = "edosn";
Console de Depuração do Microsoft Visual Studio
Cadastro Concluído
O D:\MyProject\MyProject.APPv1\bin\Debug\net6.0\MyProject.APPv1.exe
Para fechar o console automaticamente quando a depuração parar, habil
console automaticamente quando a depuração parar.
Pressione qualquer tecla para fechar esta janela...


SQLQuery3.sql *
Program.cs
D:\MYPROJECT\MYPROJECT.DAL\
SELECT *FROM Cliente
106 %
Não foi encontrado nenhum problema
T-SQL
Results
Message
Id nome telefone email
1 2 edosn 32423423 ed@teste.com

```
using MyProject.BLL;
using MyProject.MODEL;

Cliente c = new Cliente();
c.Nome = "Marcos";
c.Telefone = "r3245434";
c.Email = "marcos@teste.com";

c = ClienteRepository.Add(c);

Console.WriteLine("\nRegistro Inserido com Sucesso\n");

Console.ReadLine();

//-------------------------------------------------------

c.Nome = "Antonio Jose";
c.Telefone = "234234";
c.Email = "ant@teste.com";

ClienteRepository.Update(c);
Console.WriteLine("\nRegistro Editado com Sucesso\n");

Console.ReadLine();

//-------------------------------------------------------

List<Cliente> list = ClienteRepository.GetAll();

foreach(Cliente cli in list)
{
    Console.WriteLine(cli.Nome);
}

Console.WriteLine("\nLista de Registros Cadastrados \n");

Console.ReadLine();

//-------------------------------------------------------


ClienteRepository.Excluir(c);

Console.WriteLine("\nRegistro Excluído com Sucesso\n");

Console.ReadLine();
```



Bons Estudos!