

# SISTEMAS DISTRIBUÍDOS

C# - Acesso a Banco de Dados  
Visual Studio

PROFESSOR HILDEBERTO MELO



# Apresentação

- Técnico em Desenvolvimento de Sistemas - Ibratec, Recife-PE
- Bacharel em Sistemas de Informação – FIR, Recife-PE
- Especialista em Docência no Ensino Superior – Faculdade Maurício de Nassau, Recife-PE
- Mestre em Ciência da Computação – UFPE/CIN, Recife-PE
- *Curriculum Lattes* <http://lattes.cnpq.br/0759508594425296>)
- *Homepage* <https://sites.google.com/site/hildebertomelo/>



# Disciplinas Lecionadas

---

- Desenvolvimento de Aplicações Desktop
- Programação Orientada a Objetos
- Estrutura de Dados
- Tecnologia da Informação & Sociedade
- Sistemas Operacionais
- Sistemas Distribuídos
- Introdução a Informática
- Lógica de Programação
- Informática Aplicada a Saúde
- Banco de Dados
- Projeto de Banco de Dados
- Análise de Projetos Orientado a Objetos
- Programação Cliente Servidor
- Linguagens de Programação: C, C#, Pascal, PHP, ASP, Delphi, Java, JavaScript
- Programação WEB



# Roteiro

---

- Criação da Base de Dados
- Utilizando classes
  - Criação da Classe Aluno
  - Criação da Classe de Conexão
  - Inserindo registros na base de dados
  - Atualizando registros na base de dados
  - Removendo registros na base de dados
  - Selecionando registros na base de dados
- Aplicação com acesso a banco simples
- Entity Model



# Configurando o Banco

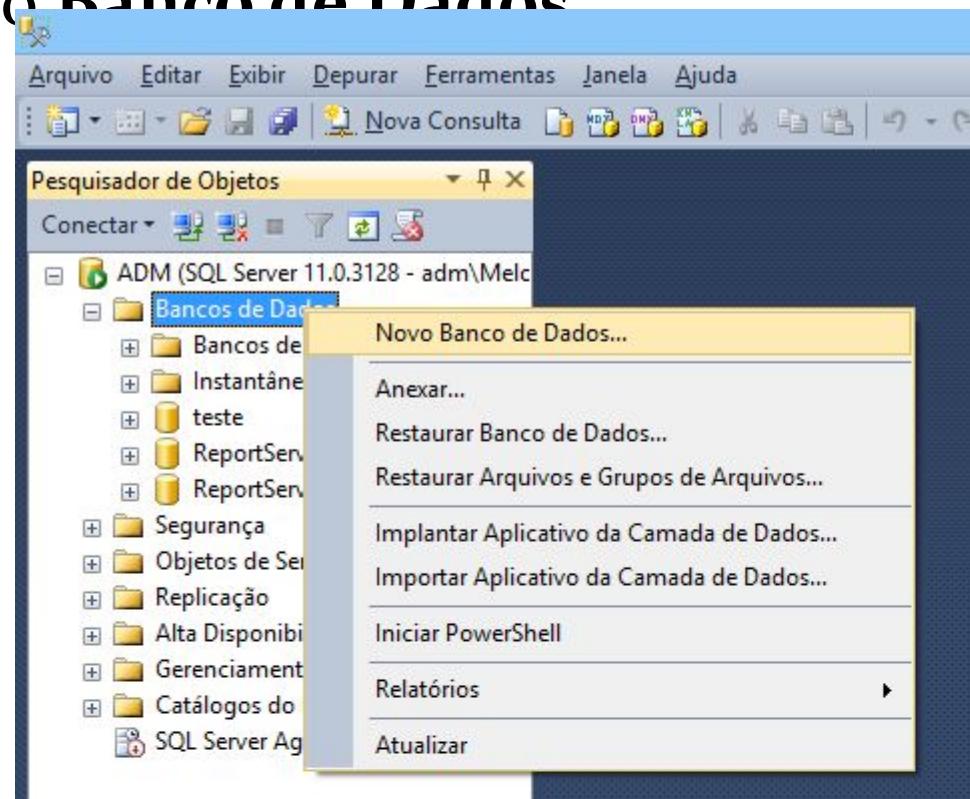
---

- Utilizaremos como SGBD o SQL Server 2012
- Passos:
  - Criação da base de dados
  - Criação do usuário para acesso a base de dados
  - Criação da tabela



# Criação da Base de Dados

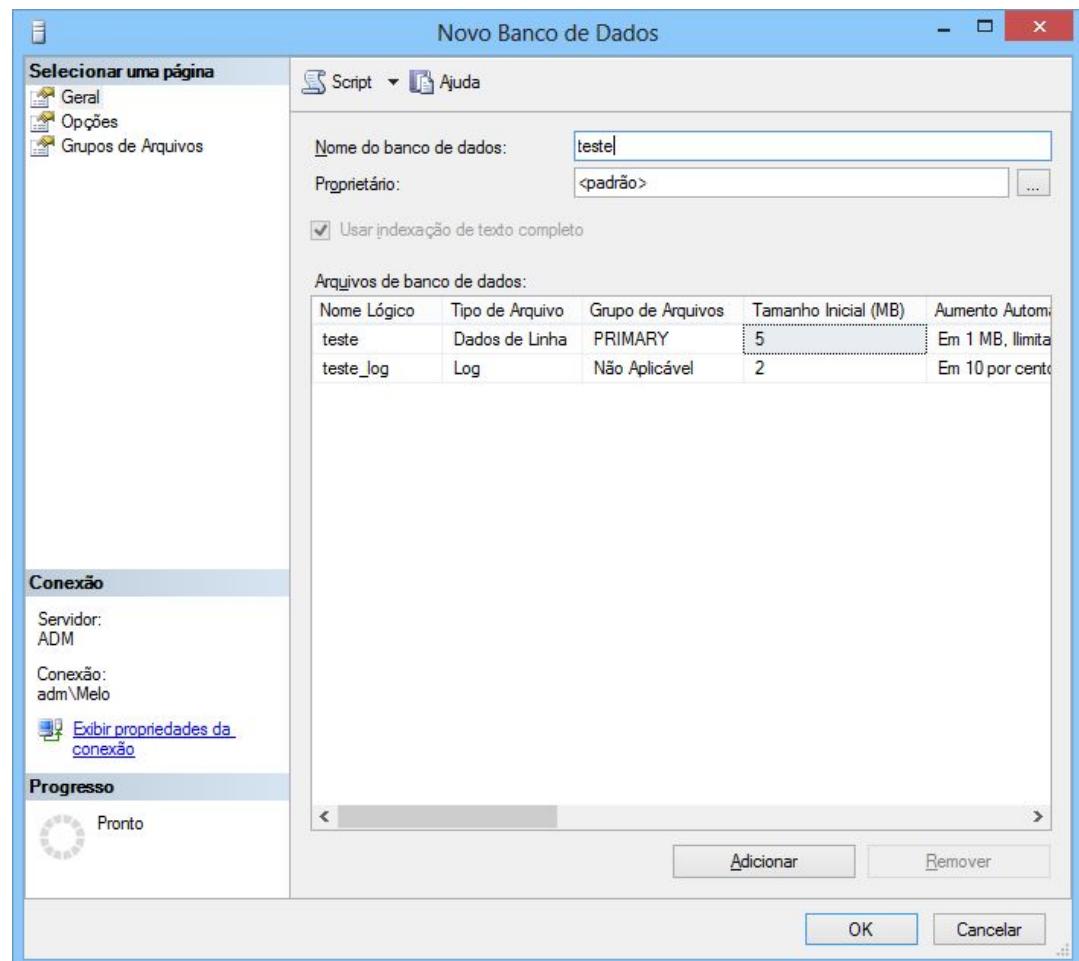
- Clicar com o botão direito do mouse em cima da pasta **Banco de Dados**
- Escolher a opção **Novo Banco de Dados**





# Criação da Base de Dados

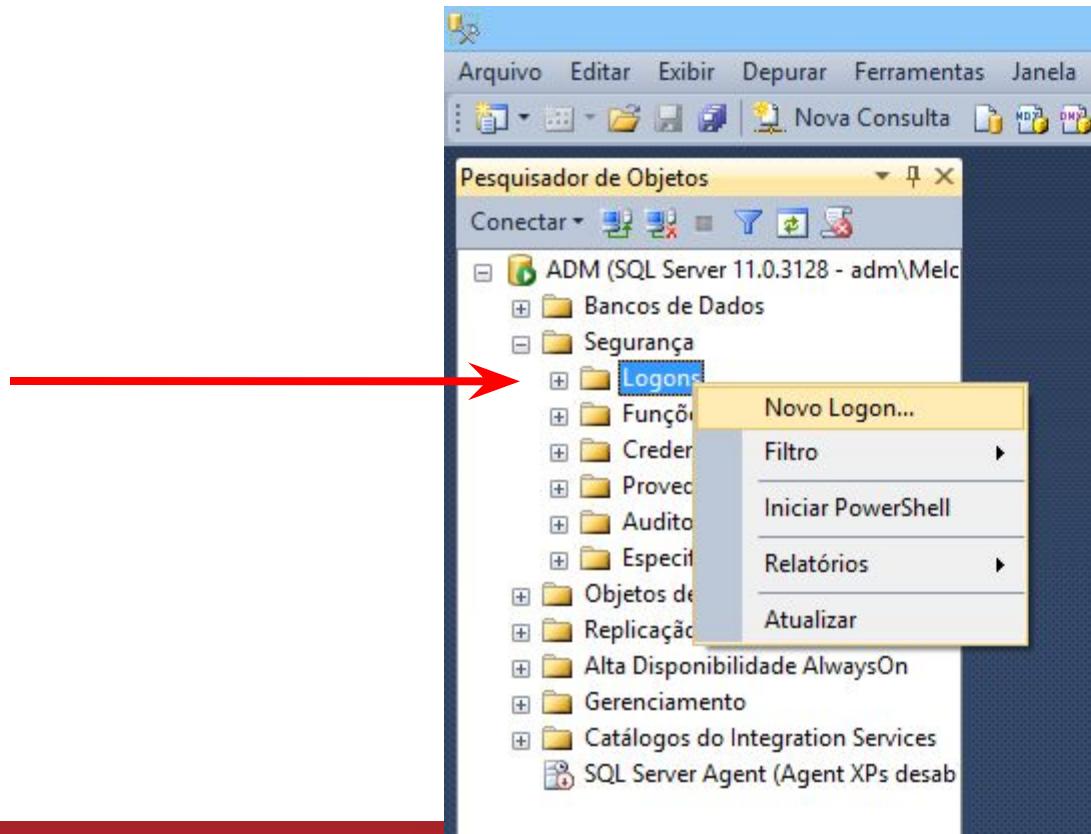
- Informar o nome do banco de dados e clicar no botão OK





# Criação do Usuário do Banco

- Clicar com o botão direito do mouse em cima da pasta Logons
- Opção **Novo Logon**

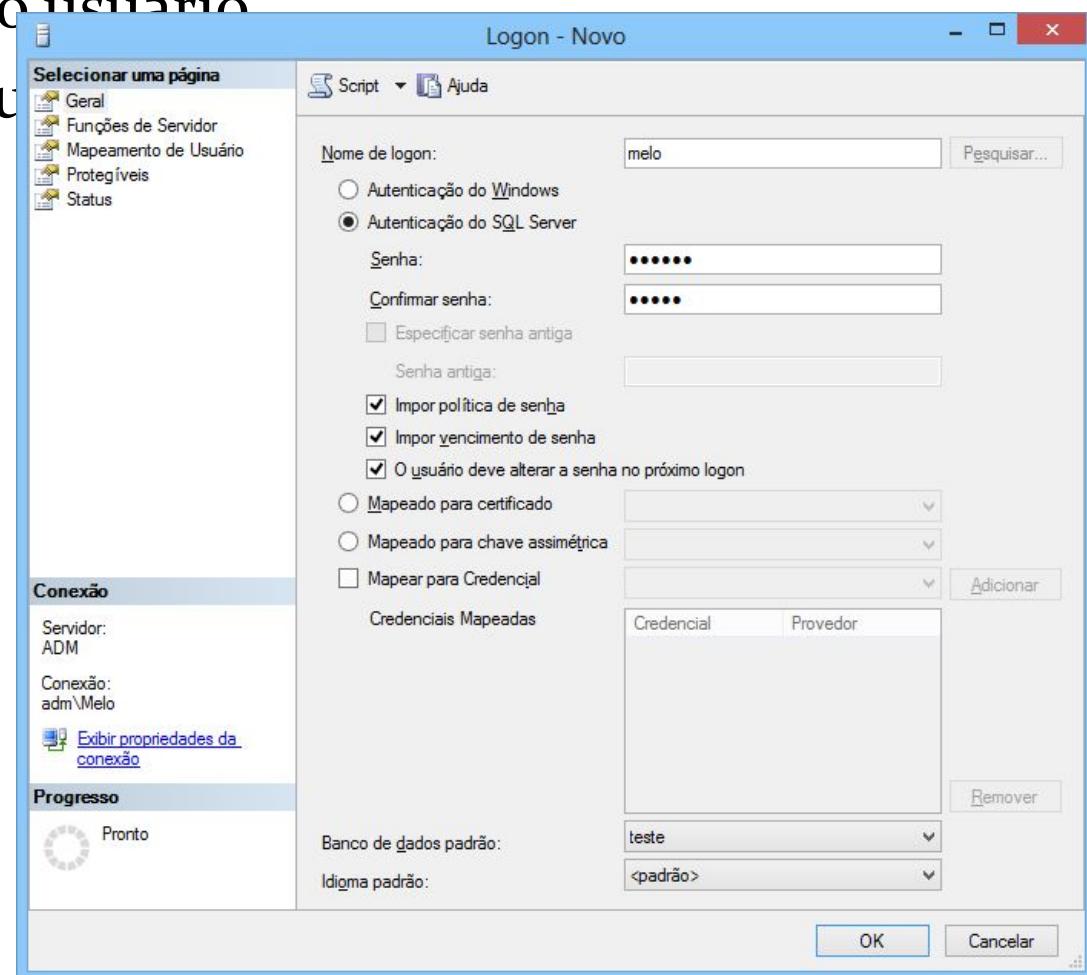




# Criação do Usuário do Banco

- Na aba Geral

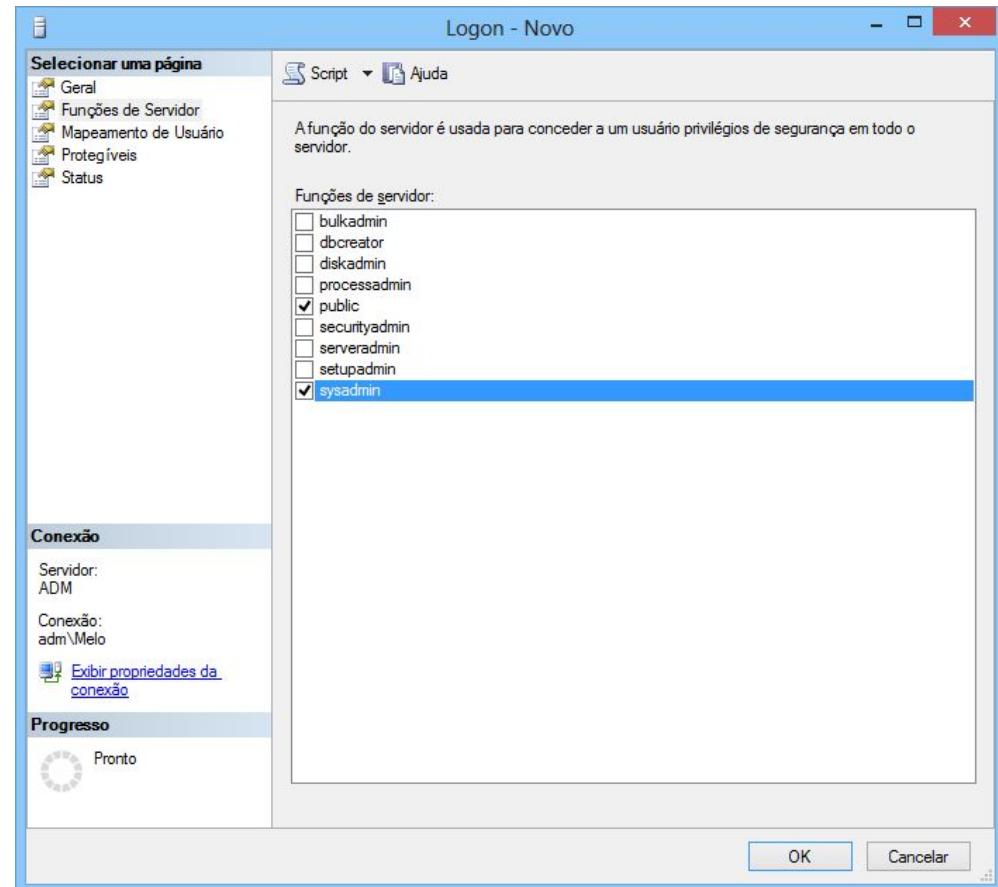
- Informar o nome do usuário
- Escolher a opção Autenticação do SQL Server
- Informar a senha





# Criação do Usuário do Banco

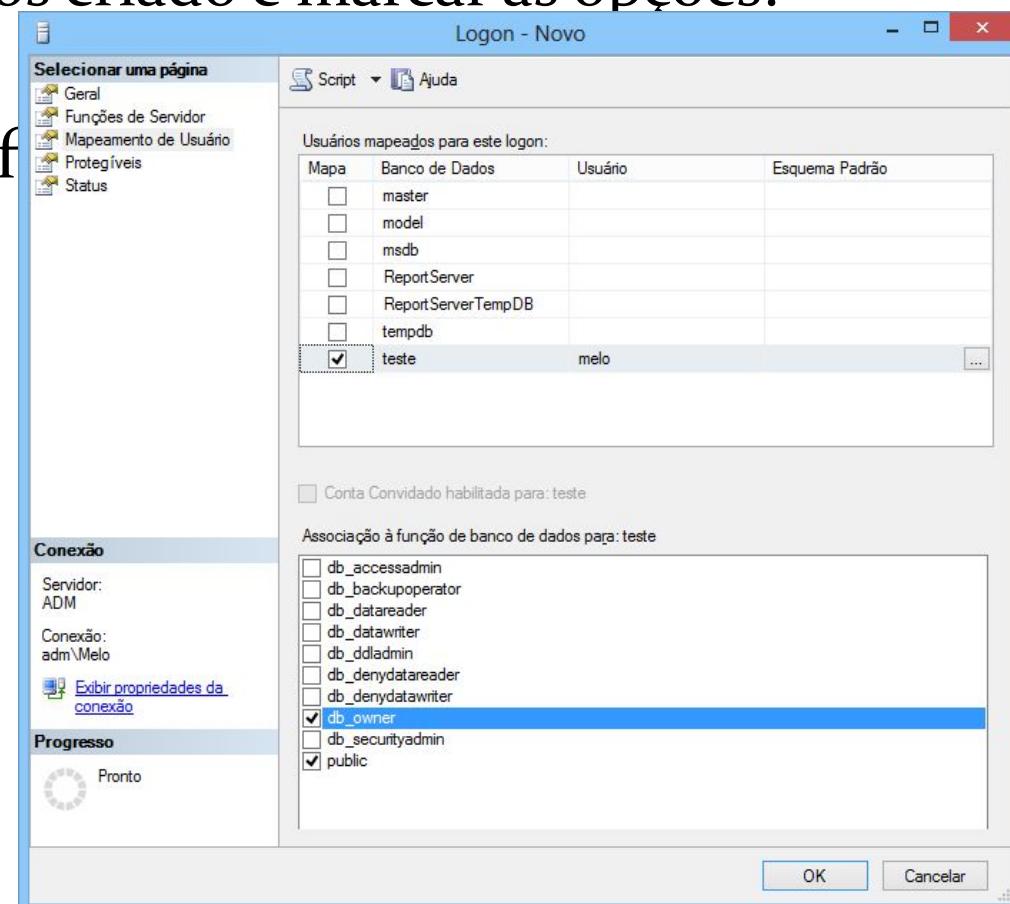
- Na aba Funções do Servidor
  - Escolher as opções: **public** e **sysadmin**





# Criação do Usuário do Banco

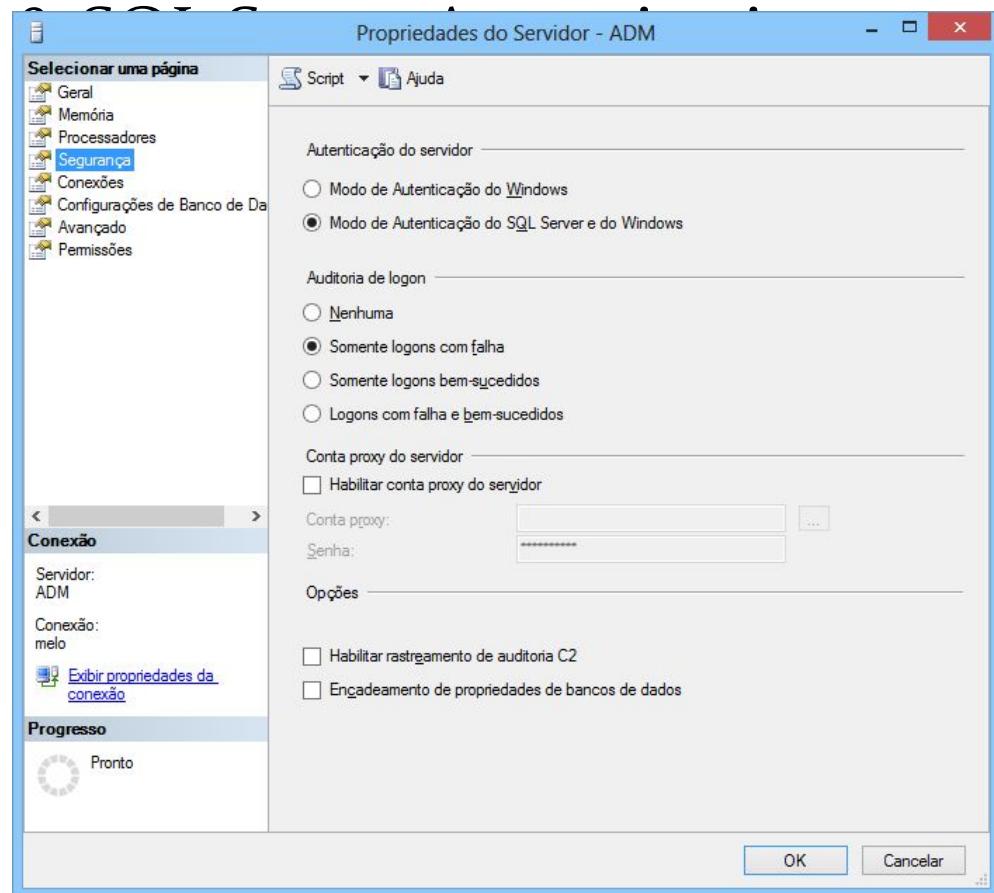
- Na aba mapeamento do usuário
  - Escolher o banco de dados criado e marcar as opções: **db\_owner** e **public**
- Clicar no botão **OK** para finalizar





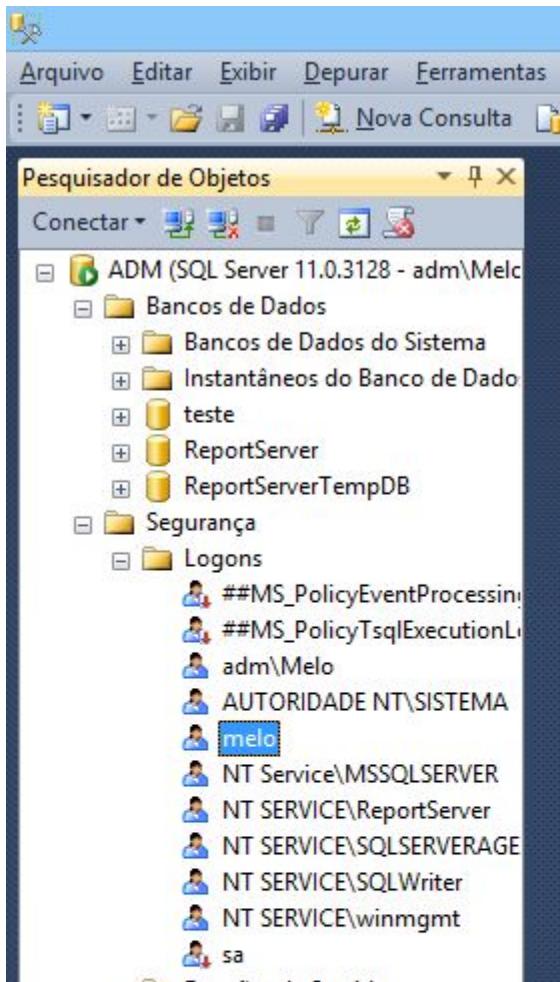
# Configurando o Usuário

- Clique sobre o seu servidor
- Vá em Properties, Security
- Mude para Windows





# Criação do Usuário do Banco





# Criação da Tabela no Banco de Dados

```
- create table aluno(  
    matricula int primary key,  
    nome varchar(100) not null  
);|
```



# Acesso ao Banco Via C#

---

- Em uma nova solução ou em uma solução já existente, vamos adicionar duas classes, uma que representará os atributos de um setor, e outra com as rotinas de acesso ao banco de dados



# Criação da Classe Aluno

---

- Criaremos a classe Aluno assemelhando-se a tabela criada na base de dados.
- A tabela terá duas colunas (matricula e nome)



# Classe de Conexão

The screenshot shows a code editor window with the following details:

- Tab Bar:** Shows "ConexaoSqlServer.cs" (selected), "Aluno.cs", and a "Biblioteca" tab.
- Namespace:** The code is part of the namespace `Biblioteca.conexao`.
- Class Definition:** A public class `ConexaoSqlServer` is defined with the following structure:

```
using System.Text;
using System.Threading.Tasks;
//importação do pacote para conexão com o sqlserver
using System.Data.SqlClient;

namespace Biblioteca.conexao
{
    public class ConexaoSqlServer
    {
        variáveis
        string de conexão
        abertura da conexão
        fechamento da conexão
    }
}
```
- Code Completion:** A tooltip labeled "variáveis" is displayed over the code, listing four items: "string de conexão", "abertura da conexão", and "fechamento da conexão".



```
#region variáveis
//tipo responsável para se trabalhar com o sqlserver
public SqlConnection sqlConn;
//máquina no qual estará o banco de dados
private const string local = "localhost";
//nome do banco de dados no qual desejamos nos comunicar
private const string banco_de_dados = "aluno";
//usuário que tenha os privilégios para utilizar o banco de dados
private const string usuario = "jersonbrito";
//senha do usuario
private const string senha = "123";
#endregion
```



```
#region string de conexão
//string de conexão obtida para o sql sever
string connectionStringSqlServer = @"Data Source=" + local+
    ";Initial Catalog=" + banco_de_dados+
    ";UID=" + usuario+
    ";Password=" + senha+";";
#endregion
```



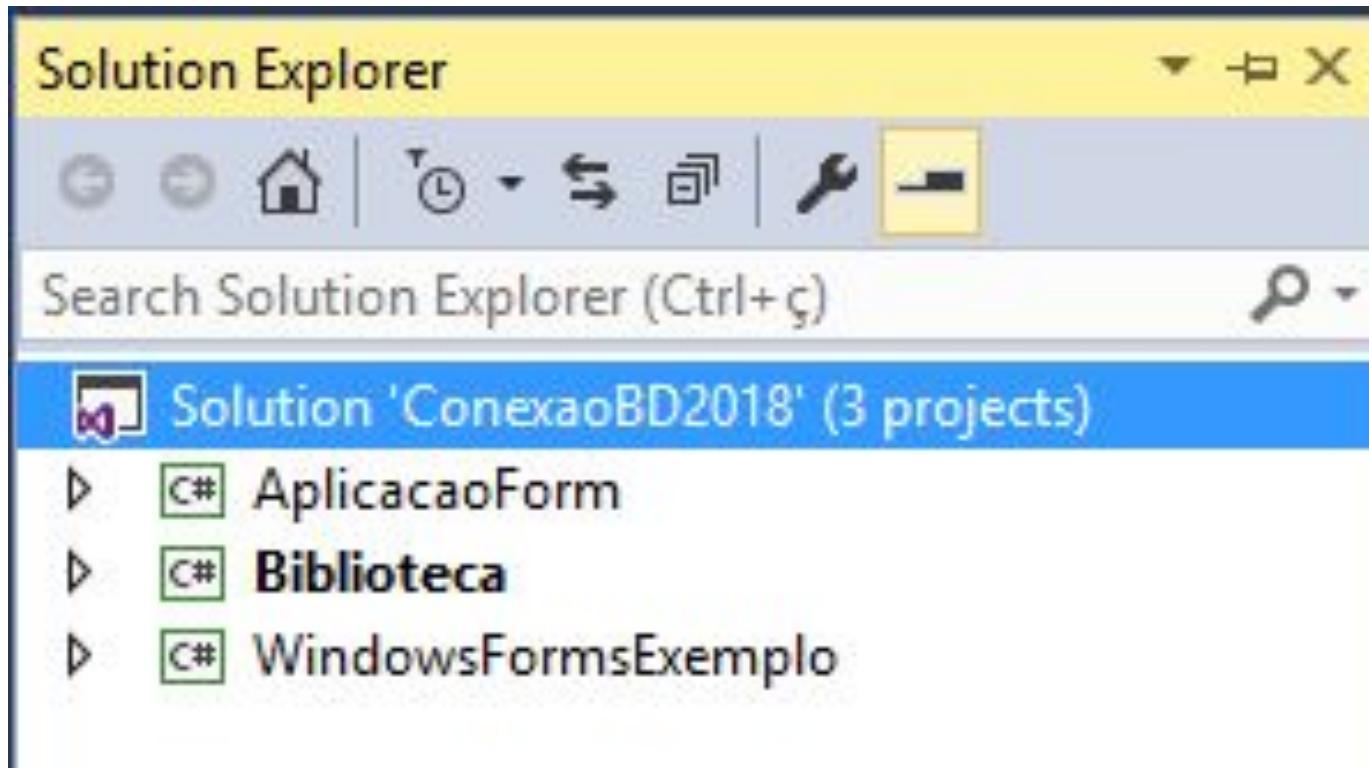
---

```
#region abertura da conexão
0 references
public void AbrirConexao()
{
    //iniciando uma conexão com o sql server,
    //utilizando os parâmetros da string de conexão
    this.sqlConn = new SqlConnection(connectionStringSqlServer);
    //abrindo a conexão com a base de dados
    this.sqlConn.Open();
}
#endregion
```

```
#region fechamento da conexão
0 references
public void FecharConexao()
{
    //fechando a conexao com a base de dados
    this.sqlConn.Close();
    //liberando a conexao da memoria
    this.sqlConn.Dispose();
}
#endregion
```



# Estrutura da Solução





# Biblioteca

```
▲ C# Biblioteca
  ▷ Properties
  ▷ References
  ▲ classesBasicas
    ▷ C# Aluno.cs
  ▲ conexao
    ▷ C# ConexaoSqlServer.cs
  ▲ dados
    ▷ C# AlunoBD.cs
  ▲ interfaces
    ▷ C# InterfaceAluno.cs
```



# Classe Básica Aluno

```
namespace Biblioteca.classesBasicas
{
    28 references
    ]   public class Aluno
        {
            private int matricula;

            14 references
            ]       public int Matricula
                    {
                        get { return matricula; }
                        set { matricula = value; }
                    }

            private string nome;

            13 references
            ]       public string Nome
                    {
                        get { return nome; }
                        set { nome = value; }
                    }
                }
}
```



# Interface Aluno

```
]namespace Biblioteca.interfaces
{
    1 reference
]    public interface InterfaceAluno
    {
        3 references
        void Insert(Aluno aluno);
        1 reference
        void Update(Aluno aluno);
        1 reference
        void Delete(Aluno aluno);
        1 reference
        bool VerificaDuplicidade(Aluno aluno);
        3 references
        List<Aluno> Select(Aluno filtro);
    }
}
```



# Classe de Dados do Aluno

```
namespace Biblioteca.dados
{
    //esta classe herda da classe de conexão
    //e implementa suas respectivas funcionalidades

    segunda forma de passar parametros
    8 references
    public class AlunoBD : ConexaoSqlServer, InterfaceAluno
    {
        Método Insert
        Método Update
        Método Delete
        Método Verifica Duplicidade
        Método Select
    }
}
```

```
#region Método Insert
3 references
public void Insert(Aluno aluno)
{
    try
    {
        //abrir a conexão
        this.AbrirConexao();
        string sql = "insert into aluno (matricula,nome) ";
        sql += "values(@matricula,@nome)";
        //instrucao a ser executada
        SqlCommand cmd = new SqlCommand(sql, this.sqlConn);

        cmd.Parameters.Add("@matricula", SqlDbType.Int);
        cmd.Parameters["@matricula"].Value = aluno.Matricula;

        cmd.Parameters.Add("@nome", SqlDbType.VarChar);
        cmd.Parameters["@nome"].Value = aluno.Nome;

        //executando a instrucao
        cmd.ExecuteNonQuery();
        //liberando a memoria
        cmd.Dispose();
        //fechando a conexao
        this.FecharConexao();
    }
    catch (Exception ex)
    {
        throw new Exception("Erro ao conectar e inserir " + ex.Message);
    }
}
#endregion
```

```
public void Update(Aluno aluno)
{
    try
    {
        //abrir a conexão
        this.AbrirConexao();
        string sql = "update aluno set ";
        sql += " matricula = @matricula ,";
        sql += " nome = @nome ";
        sql += " where matricula = @matricula2";
        //instrucao a ser executada
        SqlCommand cmd = new SqlCommand(sql, this.sqlConn);

        cmd.Parameters.Add("@matricula", SqlDbType.Int);
        cmd.Parameters["@matricula"].Value = aluno.Matricula;

        cmd.Parameters.Add("@nome", SqlDbType.VarChar);
        cmd.Parameters["@nome"].Value = aluno.Nome;

        cmd.Parameters.Add("@matricula2", SqlDbType.Int);
        cmd.Parameters["@matricula2"].Value = aluno.Matricula;

        //executando a instrucao
        cmd.ExecuteNonQuery();
        //liberando a memoria
        cmd.Dispose();
        //fechando a conexao
        this.FecharConexao();
    }
    catch (Exception ex)
    {
        throw new Exception("Erro ao conectar e atualizar " + ex.Message);
    }
}
```

```
#region Método Delete
1 reference
public void Delete(Aluno aluno)
{
    try
    {
        //abrir a conexão
        this.AbrirConexao();
        string sql = "delete from aluno where matricula = @matricula";
        //instrucao a ser executada
        SqlCommand cmd = new SqlCommand(sql, this.sqlConn);
        cmd.Parameters.Add("@matricula", SqlDbType.Int);
        cmd.Parameters["@matricula"].Value = aluno.Matricula;
        //executando a instrucao
        cmd.ExecuteNonQuery();
        //liberando a memoria
        cmd.Dispose();
        //fechando a conexao
        this.FecharConexao();
    }
    catch (Exception ex)
    {
        throw new Exception("Erro ao conectar e remover " + ex.Message);
    }
}
#endif
```

```
public bool VerificaDuplicidade(Aluno aluno)
{
    bool retorno = false;
    try
    {
        this.AbrirConexao();
        //instrucao a ser executada
        string sql = "SELECT matricula,nome FROM aluno where matricula = @matricula";
        SqlCommand cmd = new SqlCommand(sql, sqlConn);
        cmd.Parameters.Add("@matricula", SqlDbType.Int);
        cmd.Parameters["@matricula"].Value = aluno.Matricula;
        //executando a instrucao e colocando o resultado em um leitor
        SqlDataReader DbReader = cmd.ExecuteReader();
        //lendo o resultado da consulta
        while (DbReader.Read())
        {
            retorno = true;
            break;
        }
        //fechando o leitor de resultados
        DbReader.Close();
        //liberando a memoria
        cmd.Dispose();
        //fechando a conexao
        this.FecharConexao();
    }
    catch (Exception ex)
    {
        throw new Exception("Erro ao conectar e selecionar " + ex.Message);
    }
    return retorno;
}
```



```
public List<Aluno> Select(Aluno filtro)
{
    List<Aluno> retorno = new List<Aluno>();
    try
    {
        this.AbrirConexao();
        //instrucao a ser executada
        string sql = "SELECT matricula,nome FROM aluno where matricula = matricula ";
        //se foi passada uma matricula válida, esta matricula entrará como critério de filtro
        if (filtro.Matricula > 0)
        {
            sql += " and matricula = @matricula";
        }
        //se foi passada um nome válido, este nome entrará como critério de filtro
        if (filtro.Nome != null && filtro.Nome.Trim().Equals("") == false)
        {
            sql += " and nome like @nome";
        }
        SqlCommand cmd = new SqlCommand(sql, sqlConn);

        //se foi passada uma matricula válida, esta matricula entrará como critério de filtro
        if (filtro.Matricula > 0)
        {
            cmd.Parameters.Add("@matricula", SqlDbType.Int);
            cmd.Parameters["@matricula"].Value = filtro.Matricula;
        }
        //se foi passada um nome válido, este nome entrará como critério de filtro
        if (filtro.Nome != null && filtro.Nome.Trim().Equals("") == false)
        {
            cmd.Parameters.Add("@nome", SqlDbType.VarChar);
            cmd.Parameters["@nome"].Value = "%" + filtro.Nome + "%";
        }
    }
```

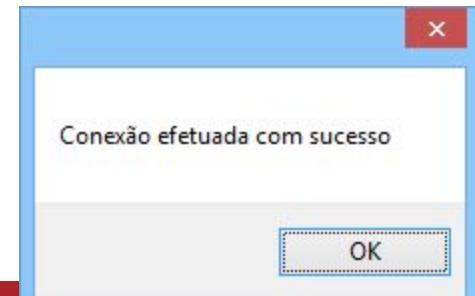
```
//executando a instrucao e colocando o resultado em um leitor
SqlDataReader DbReader = cmd.ExecuteReader();
//lendo o resultado da consulta
while (DbReader.Read())
{
    Aluno aluno = new Aluno();
    //acessando os valores das colunas do resultado
    aluno.Matricula = DbReader.GetInt32(DbReader.GetOrdinal("matricula"));
    aluno.Nome = DbReader.GetString(DbReader.GetOrdinal("nome"));
    retorno.Add(aluno);
}
//fechando o leitor de resultados
DbReader.Close();
//liberando a memoria
cmd.Dispose();
//fechando a conexao
this.FecharConexao();
}
catch (Exception ex)
{
    throw new Exception("Erro ao conectar e selecionar " + ex.Message);
}
return retorno;
}
```



# Testando a Conexão

- Colocaremos no clique de um botão em um formulário o código fonte a baixo e depois basta testá-lo

```
try
{
    ConexaoSqlServer conn = new ConexaoSqlServer();
    conn.AbrirConexao();
    MessageBox.Show("Conectou");
    conn.FecharConexao();
    MessageBox.Show("Desconectou");
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
```





# Testando o Select

```
try
{
    AlunoBD dados = new AlunoBD();
    Aluno filtro = new Aluno();
    if (textBoxMatricula.Text.Trim().Equals("") == false)
    {
        filtro.Matricula = Int32.Parse(textBoxMatricula.Text.Trim());
    }
    filtro.Nome = textBoxNome.Text;
    listaAlunos = dados.Select(filtro);
    listViewAlunos.Items.Clear();
    foreach (Aluno a in listaAlunos)
    {
        ListViewItem linha = listViewAlunos.Items.Add(a.Matricula.ToString());
        linha.SubItems.Add(a.Nome);

    }
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
```



# Testando o Insert

```
try
{
    Aluno a = new Aluno();
    a.Matricula = Int32.Parse(textBoxMatricula.Text.Trim());
    a.Nome = textBoxNome.Text;
    AlunoBD dados = new AlunoBD();
    dados.Insert(a);
    MessageBox.Show("Aluno cadastrado com sucesso");
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
```



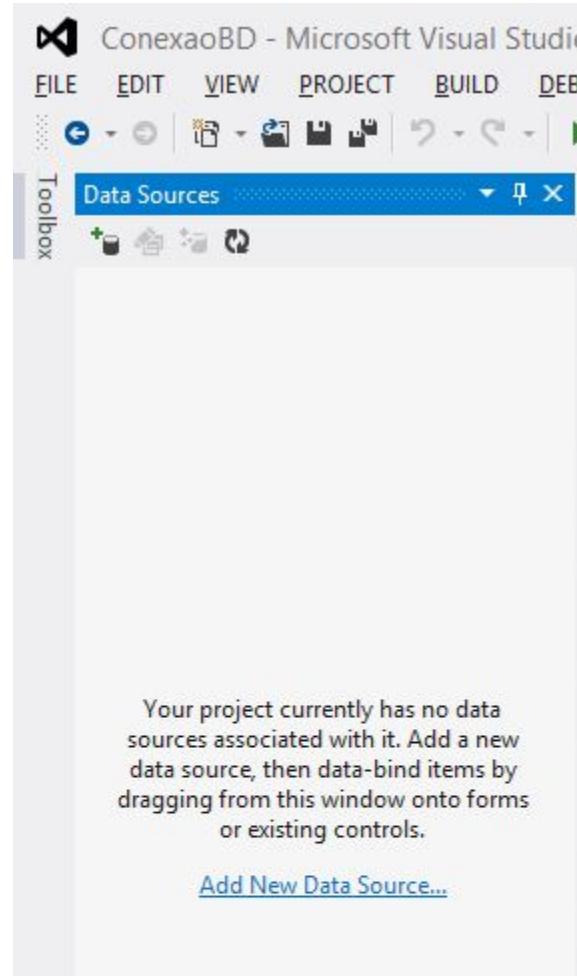
# Aplicação Rápida Com Acesso a Banco

- Passos:

- Criar um novo projeto
- Adicionar uma fonte de dados ao projeto
- Vincular a fonte de dados ao formulário



# Fonte de Dados





# Aplicação Rápida

Data Source Configuration Wizard

Choose a Data Source Type

Where will the application get data from?

Database      Service      Object      SharePoint

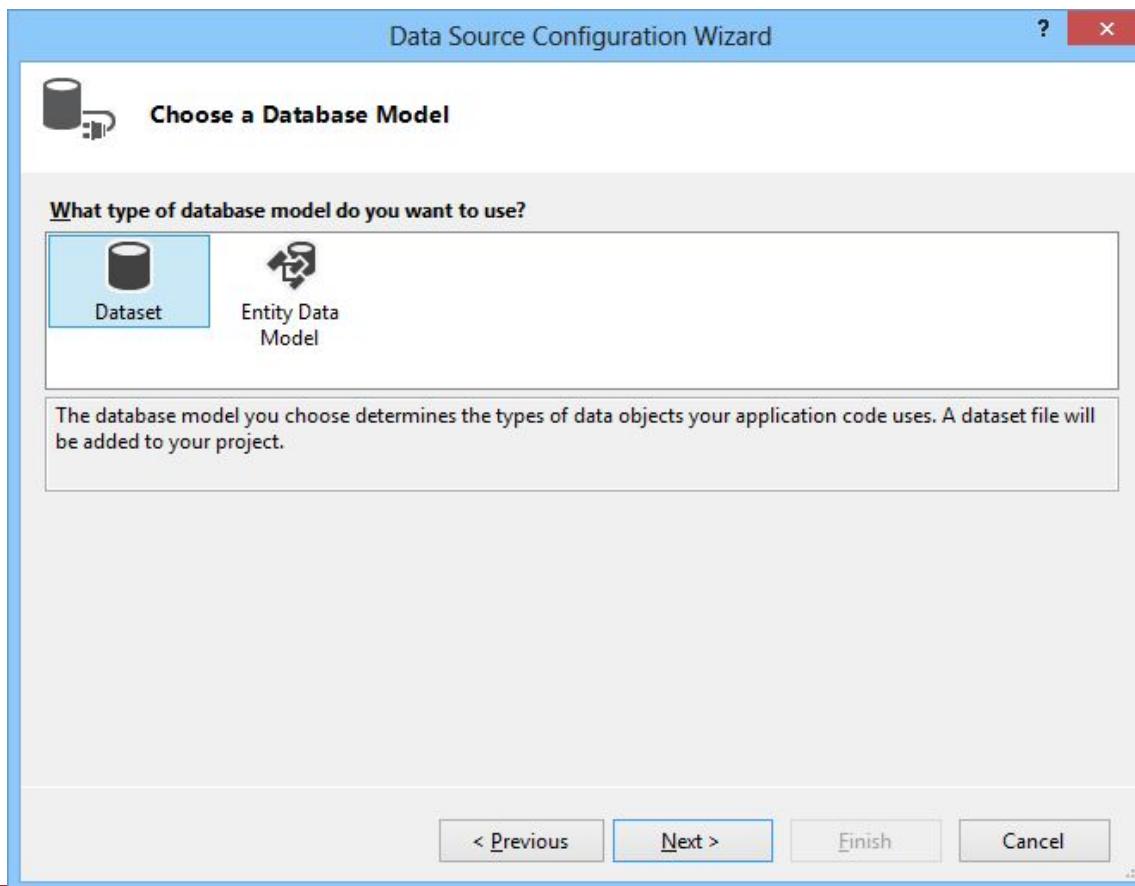
Lets you connect to a database and choose the database objects for your application.

< Previous      Next >      Finish      Cancel

c.edu.br



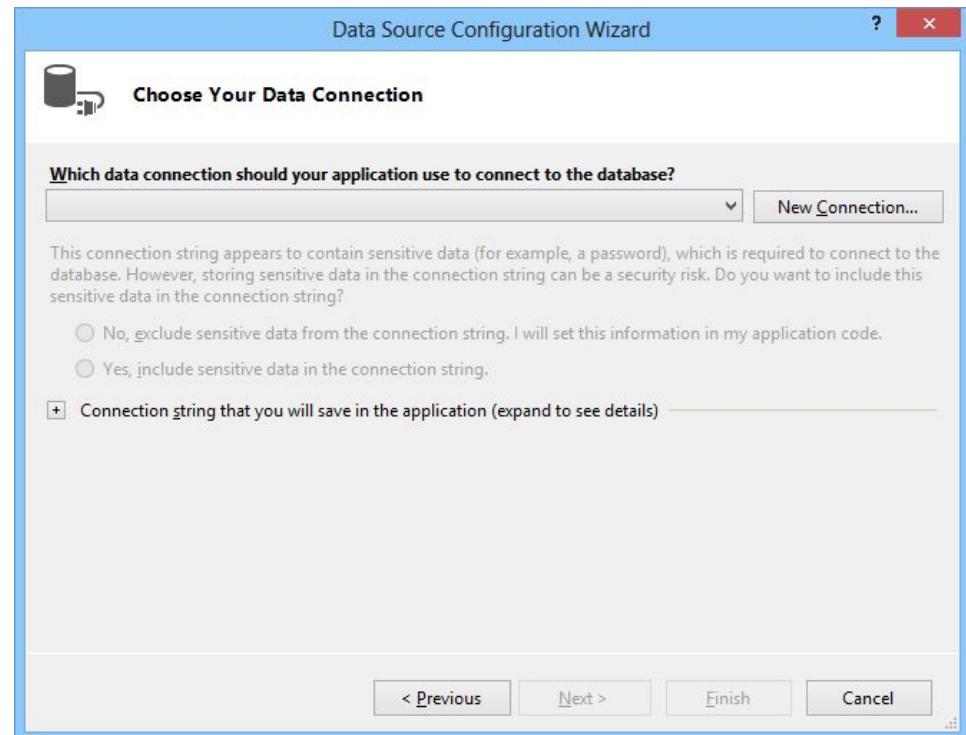
# Aplicação Rápida





# Aplicação Rápida

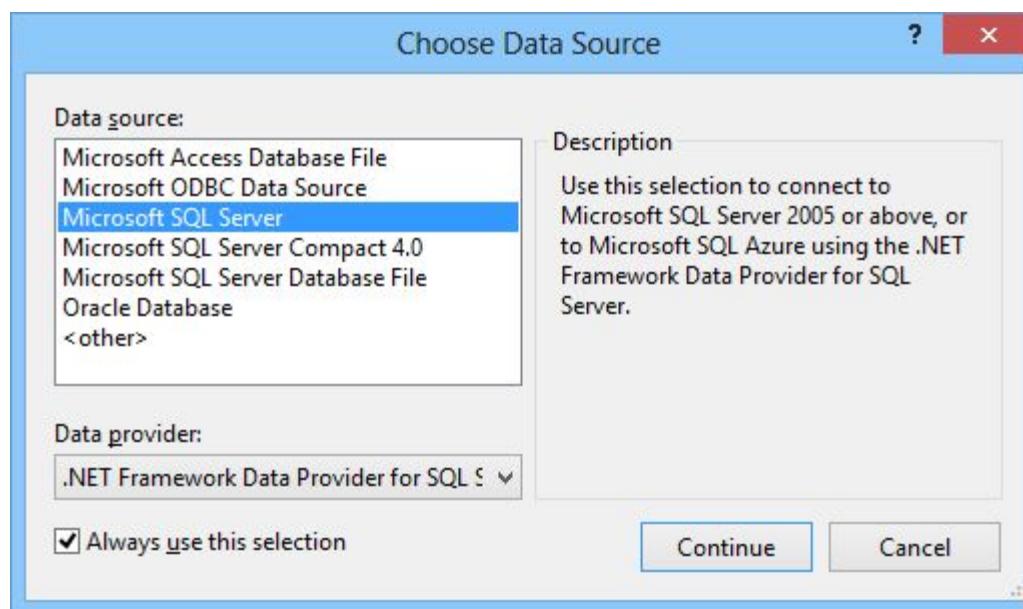
- Criando a conexão





# Aplicação Rápida

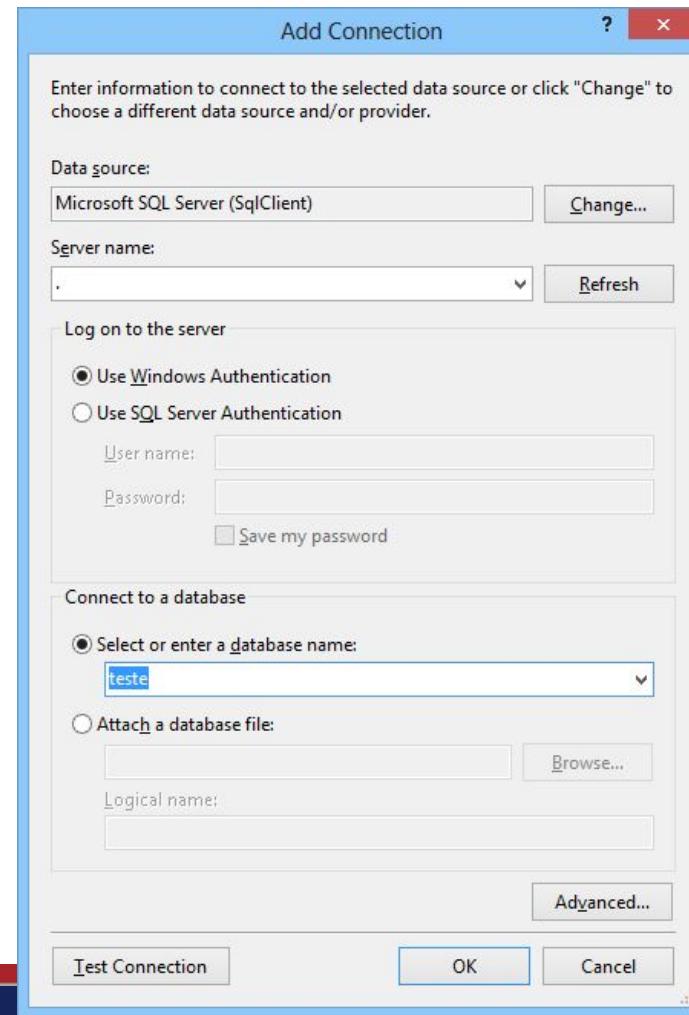
- Escolhendo a forma de conexão





# Aplicação Rápida

- Selecionando o servidor e o banco de dados





# Aplicação Rápida

**Data Source Configuration Wizard**

**Choose Your Data Connection**

**Which data connection should your application use to connect to the database?**

adm.teste dbo

This connection string appears to contain sensitive data (for example, a password), which is required to connect to the database. However, storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?

No, exclude sensitive data from the connection string. I will set this information in my application code.

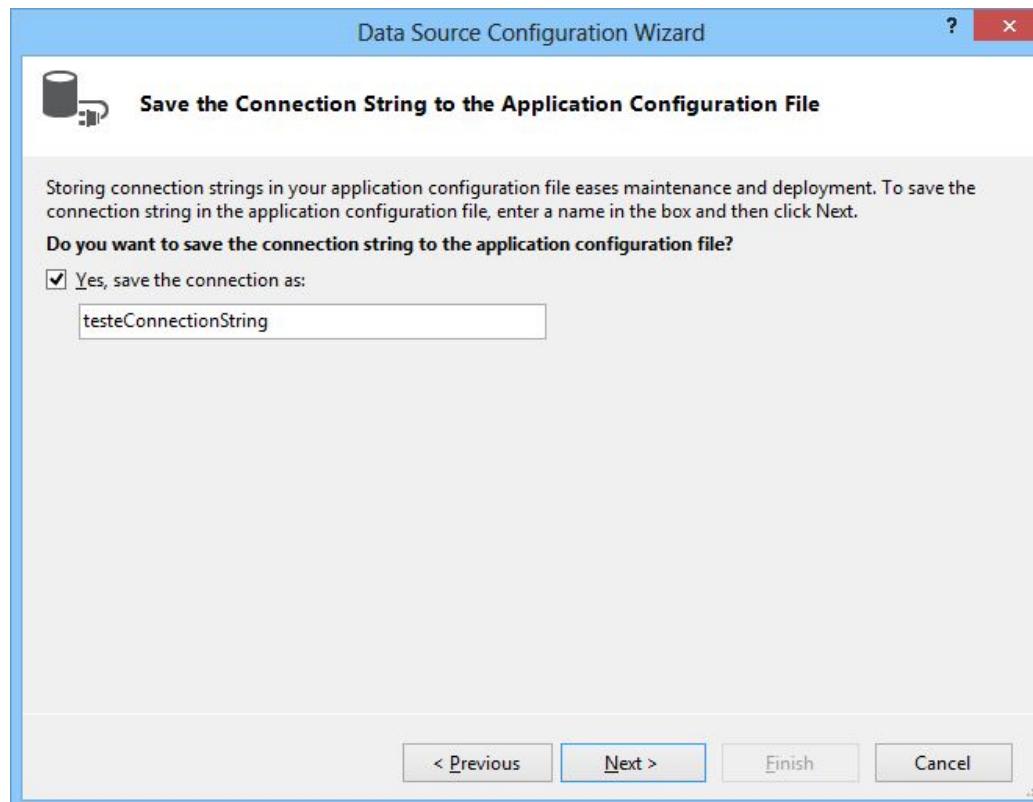
Yes, include sensitive data in the connection string.

Connection string that you will save in the application (expand to see details)

< Previous Next > Finish Cancel



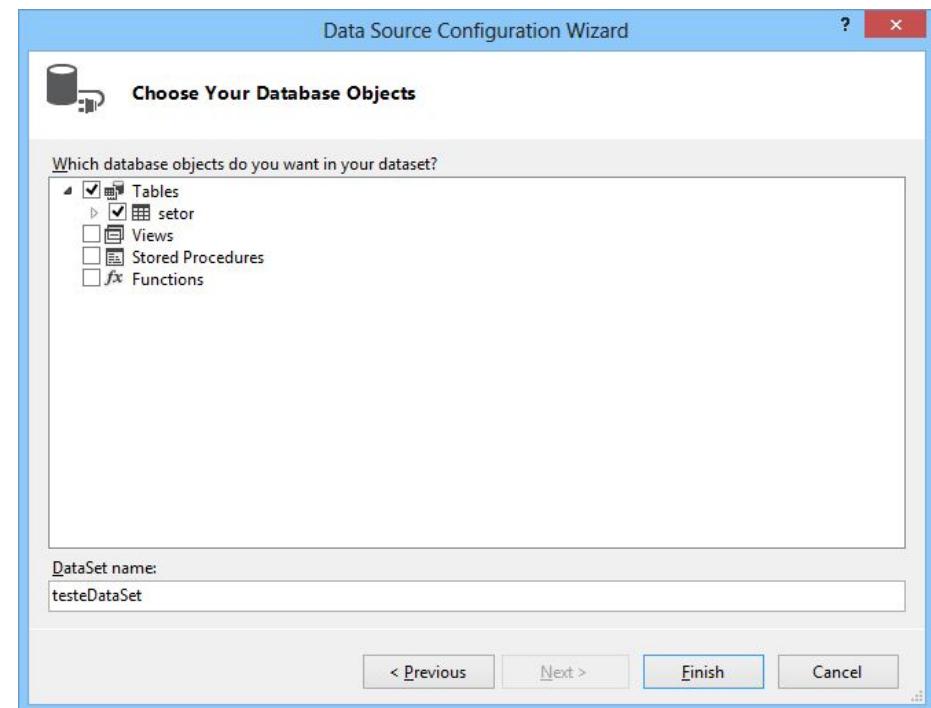
# Aplicação Rápida





# Aplicação Rápida

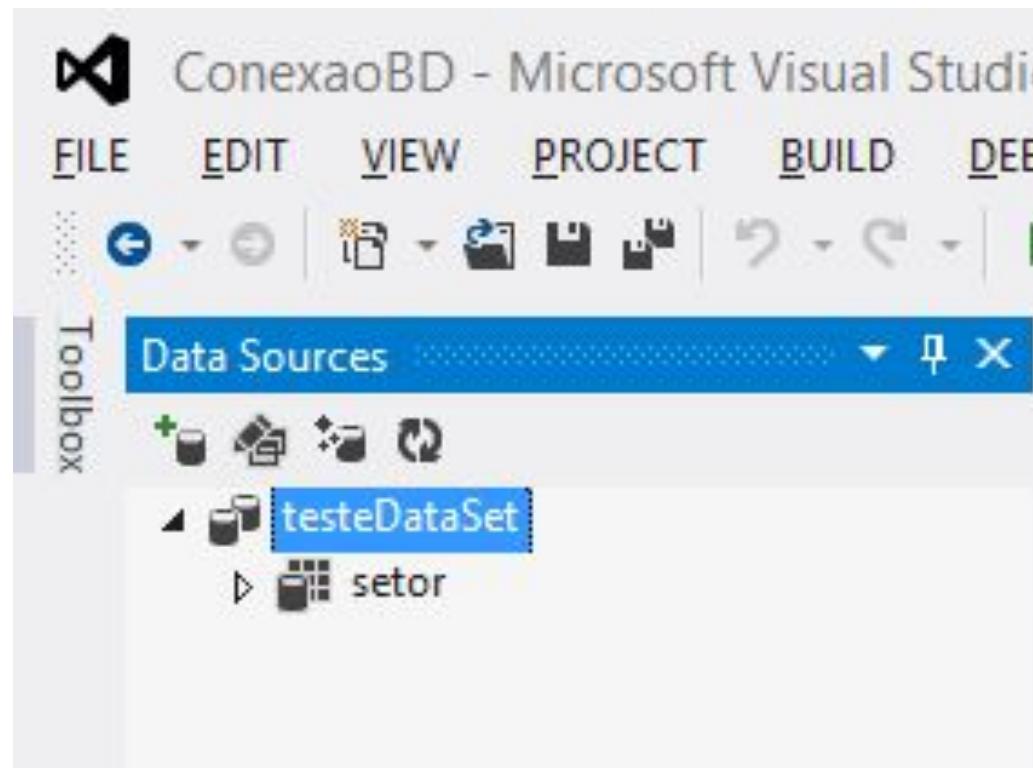
- Selecionado as tabelas que farão parte da fonte de dados





# Aplicação Rápida

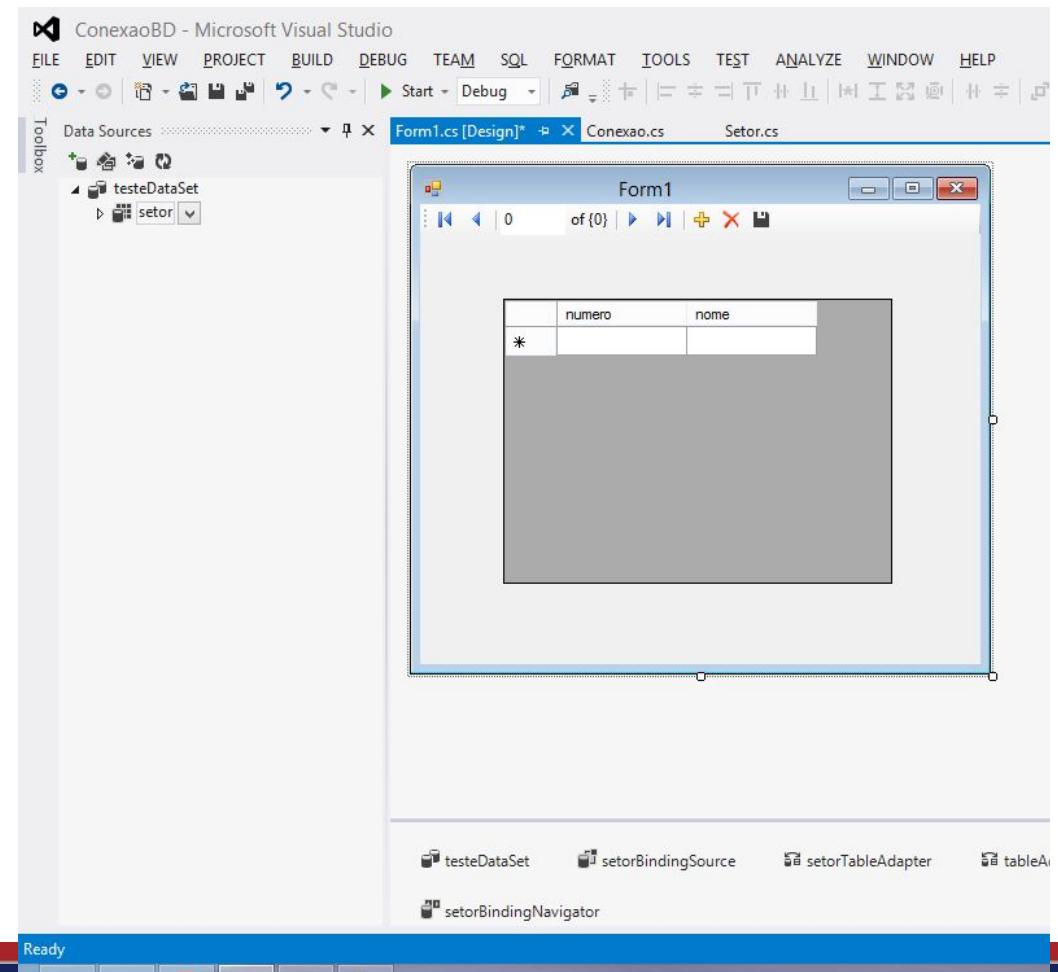
- Fonte de dados criada





# Aplicação Rápida

- Clicar e arrastar a tabela da fonte de dados para o formulário





# Executando o Form

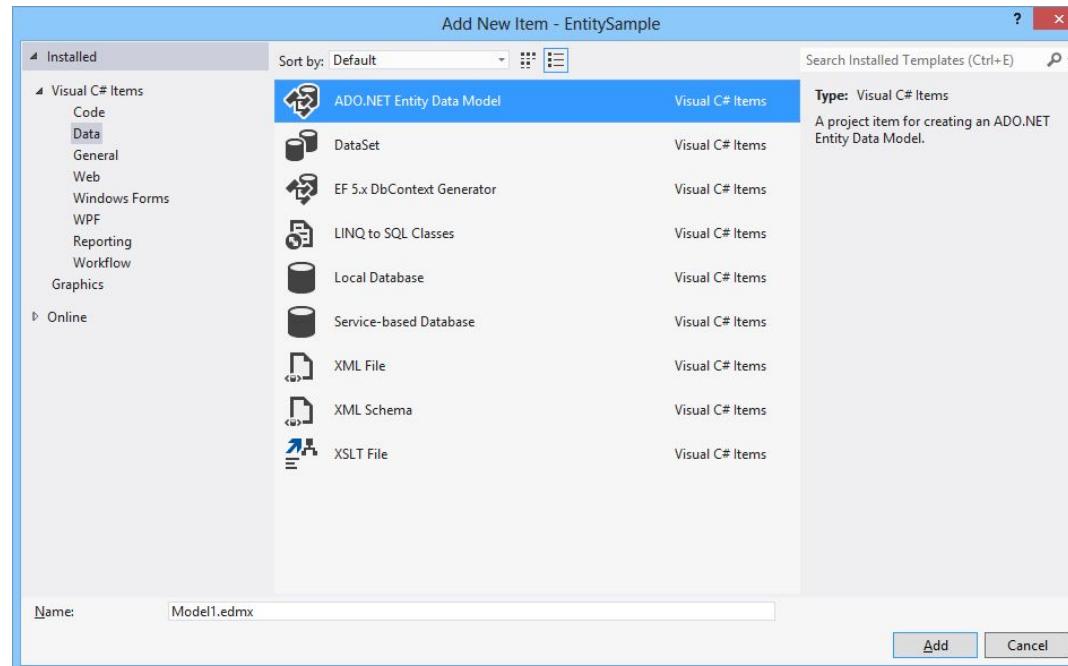
The screenshot shows a Windows application window titled "Form1". At the top, there is a toolbar with various icons: back, forward, search, and other standard window controls. Below the toolbar is a table control displaying data. The table has two rows:

	numero	nome
▶	1	RH
*		



# Entity Framework

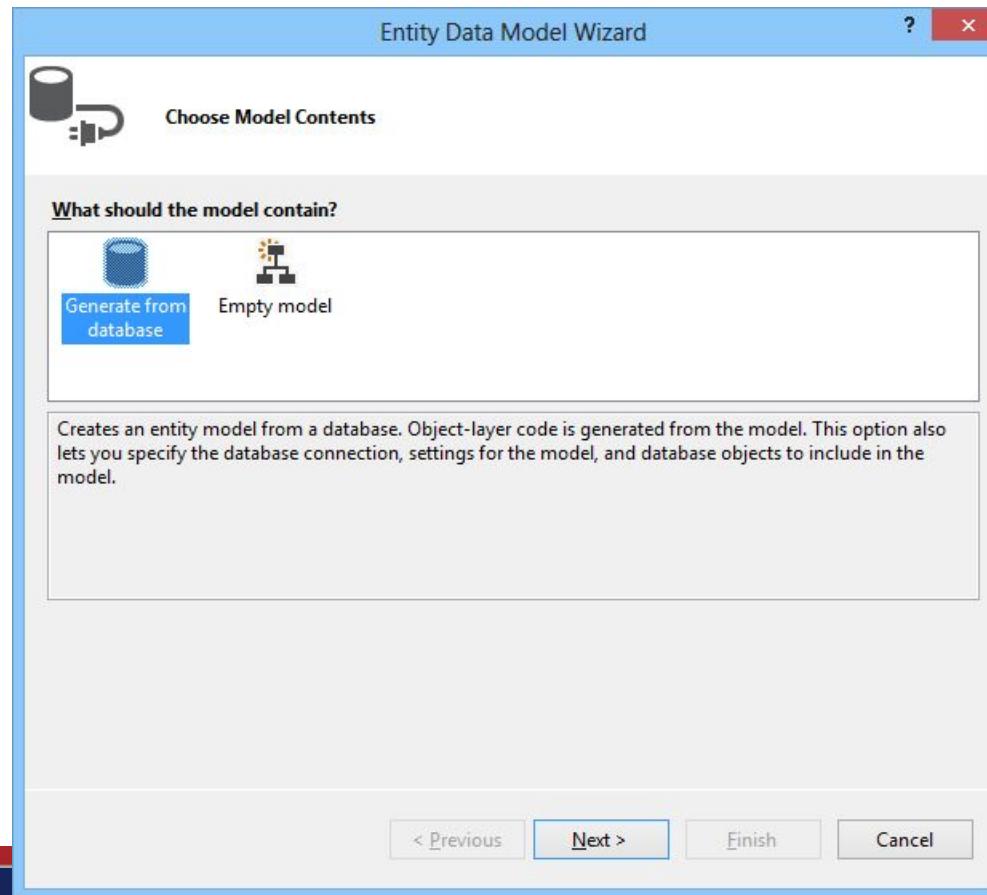
- Adicionar outro projeto na solução
- Adicionar o Entity Framework neste projeto





# Entity Model

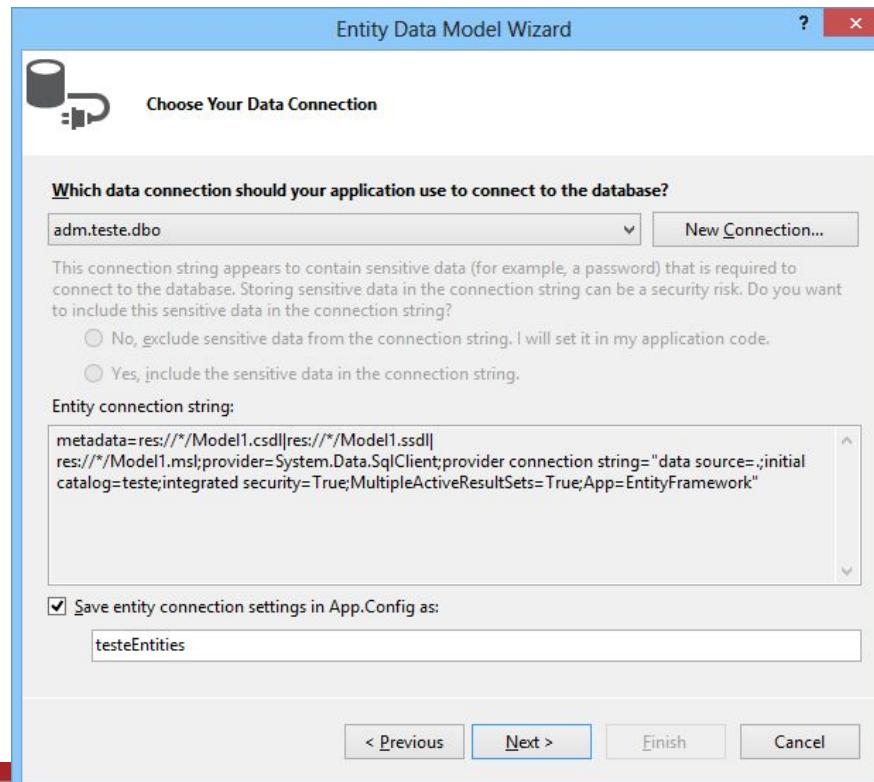
- Configurando a conexão





# Entity Model

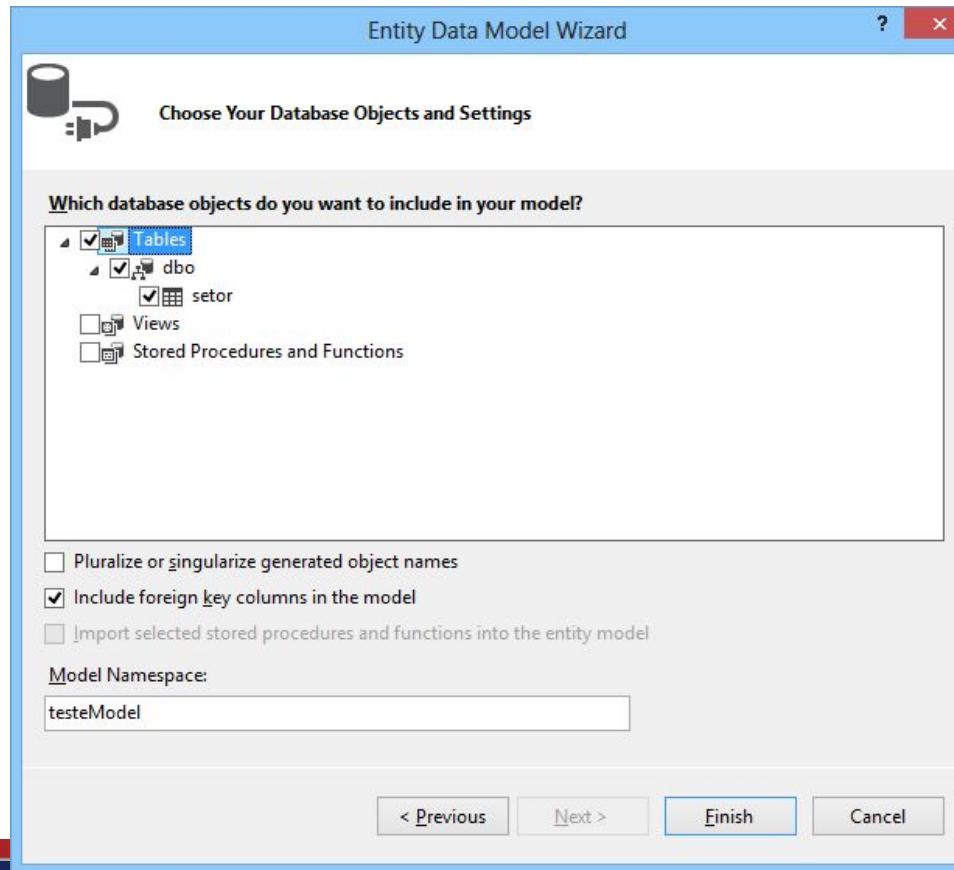
- Configurando a conexão





# Entity Model

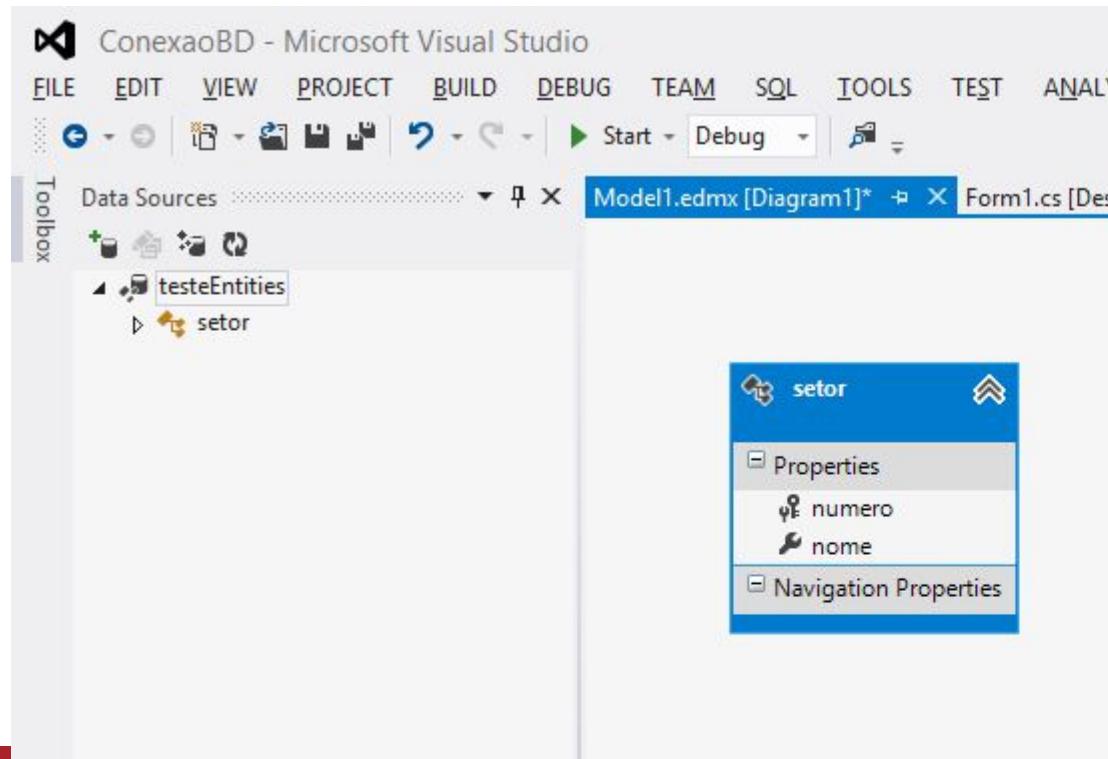
- Selecionando as tabelas





# Entity

- Automaticamente é criada a classe que se refere a tabela setor





# Inserindo Entity Model

```
try
{
    testeEntities bd = new testeEntities();
    setor s = new setor();
    s.numero = 4;
    s.nome = "Secretaria";
    bd.setor.Add(s);
    bd.SaveChanges();
    MessageBox.Show("Setor cadastrado");
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
```



# Removendo Entity Model

```
try
{
    testeEntities bd = new testeEntities();
    setor s = bd.setor.ToList()[listBox1.SelectedIndex];
    bd.setor.Remove(s);
    bd.SaveChanges();
    MessageBox.Show("Removido com sucesso");
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
```



# Atualizando Entity Modelo

```
try
{
    testeEntities bd = new testeEntities();
    setor s = bd.setor.ToList()[listBox1.SelectedIndex];
    s.nome = "juridico";
    bd.SaveChanges();
    MessageBox.Show("Removido com sucesso");
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
```



# Selecionando Entity Model

```
try
{
    testeEntities bd = new testeEntities();
    listBox1.Items.Clear();
    foreach (var item in bd.setor)
    {
        listBox1.Items.Add(item.nome);
    }
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
```



# Selecionando Entity Model

```
testeEntities bd = new testeEntities();
var consulta = (from s in bd.setor
                orderby s.nome ascending
                select s);

listBox1.Items.Clear();
foreach (var item in consulta.ToList())
{
    listBox1.Items.Add(item.nome);
}
```



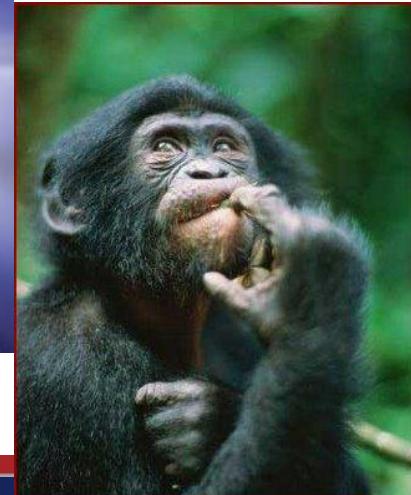
# Selecionando Entity Model

```
testeEntities bd = new testeEntities();
var consulta = (from s in bd.setor
                where s.nome.Contains("o")
                orderby s.nome ascending
                select s);

listBox1.Items.Clear();
foreach (var item in consulta.ToList())
{
    listBox1.Items.Add(item.nome);
}
```



# Perguntas





# Referências

Bibliografia Básica (títulos, periódicos, etc.)						
Título/Periódico	Autor	Edição	Local	Editora	Ano	LT/BV <sup>1</sup>
Microsoft Visual C# 2008: passo a passo	SHARP, John	1 <sup>a</sup>	Porto Alegre	Bookman	2008	LT
Desenvolvimento em camadas com C# .Net	CAMACHO JÚNIOR, Carlos Olavo de Azevedo	1 <sup>a</sup>	Florianópolis	Visual Books	2008	LT
C#: como programar	DEITEL, H. M.	1 <sup>a</sup>	São Paulo	Pearson	2003	LT
Bibliografia Complementar (títulos, periódicos, etc.)						
Título/Periódico	Autor	Edição	Local	Editora	Ano	LT/BV <sup>1</sup>
Use a cabeça C#	STELLMAN, Andrew; GREENE, Jeniffer	2 <sup>a</sup>	Rio de Janeiro	Alta Books	2011	LT
Treinamento avançado em XML	FARIA, Rogério Amorim De	1 <sup>a</sup>	São Paulo	Digerati	2005	LT
Projeto de Banco de Dados com XML	GRAVES, Mark	1 <sup>a</sup>	São Paulo	Pearson	2003	BV