

Aplicativo MVC – Parte 2

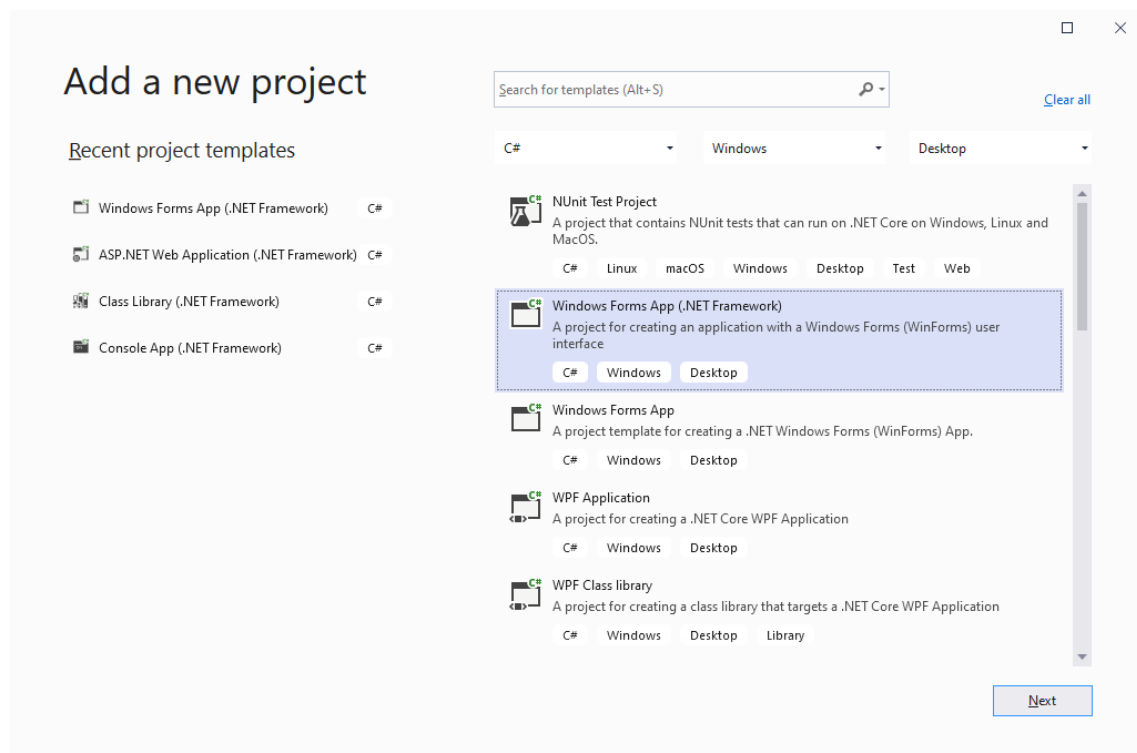
José Cassiano Grassi Gunji

Nosso aplicativo já possui uma interface gráfica com o usuário em modalidade web que acessa sua lógica implementada em um projeto de biblioteca de classes. Agora vamos criar a mesma interface na modalidade desktop.

É prática comum criar um sistema com interfaces gráficas tanto em modalidade web quanto em desktop. Em geral, as telas desktop são mais fáceis de se desenvolver (pelo menos para programadores) e podem realizar parte da lógica do sistema, diminuindo a carga dos servidores e o tráfego pela rede. Já as telas web podem ser executadas em terminais bastante simples e baratos. Além disso, a manutenção e a implantação são mais fáceis pois envolvem apenas os servidores (na maioria dos casos). Mas este tipo de implantação aumenta a carga de trabalho dos servidores e da rede. Por isso, a escolha pela modalidade de interface gráfica do usuário depende de cada implantação.

Para tanto, clique com o botão direito na solução Aplicativo MVC e escolha [Add] -> [New Project...].

Figura 1: Criando o projeto Windows Forms App.

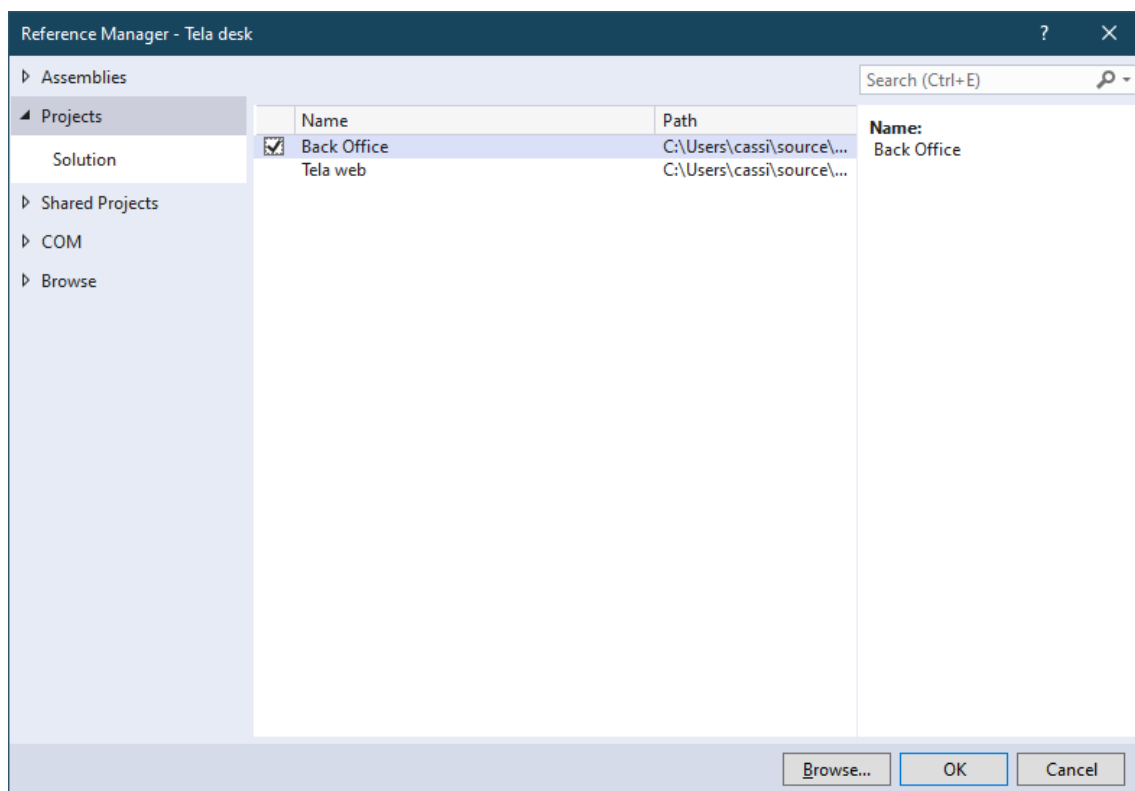


Na janela que se abre, mude o filtro para apresentar modelos de linguagem C#, plataforma Windows e modalidade Desktop. Escolha o modelo Windows Forms App (.NET Framework). Lembre-se de não escolher o modelo .NET Core, pois este não será compatível com o restante da solução.

Na janela a seguir, defina o nome do projeto como “Tela Desk” e finalize o assistente clicando em [Create]. Um novo projeto será adicionado à sua solução. Clique com o botão direito no projeto “Tela Desk” e selecione “Set as default Project”. Note que o projeto agora está em negrito e o botão de execução agora mudou de “IIS Express” para “Start”, indicando que se a solução for executada agora, ela será compilada para a modalidade desktop e não mais web.

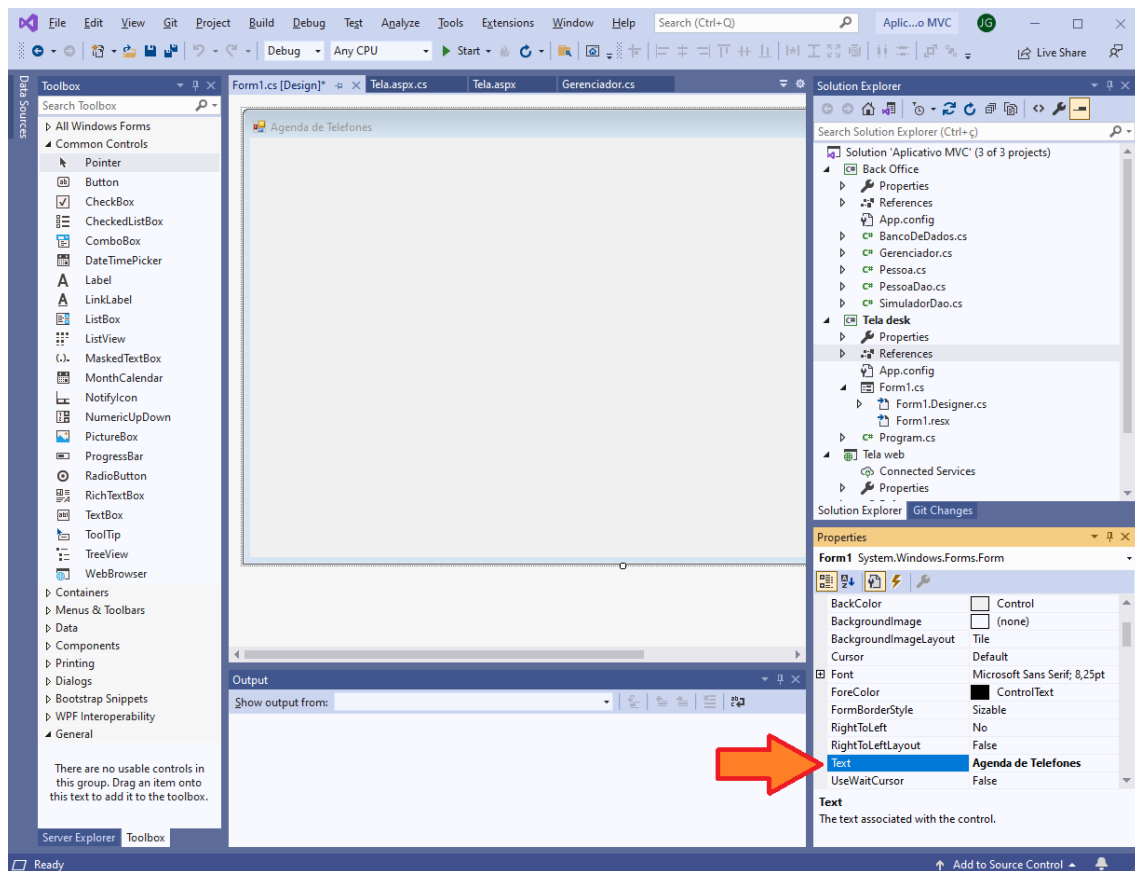
Para usarmos as classes e pacotes desenvolvidos para o back office, inclua como referência ao projeto “Tela Desk” o projeto “Back Office”. Clique com o botão direito sobre References de Tela Desk e selecione [Add reference...]. Note que agora a janela apresenta os dois projetos da solução. Escolha apenas o projeto “Back Office” e finalize o assistente.

Figura 2: Adicionando a referência ao projeto Back Office.



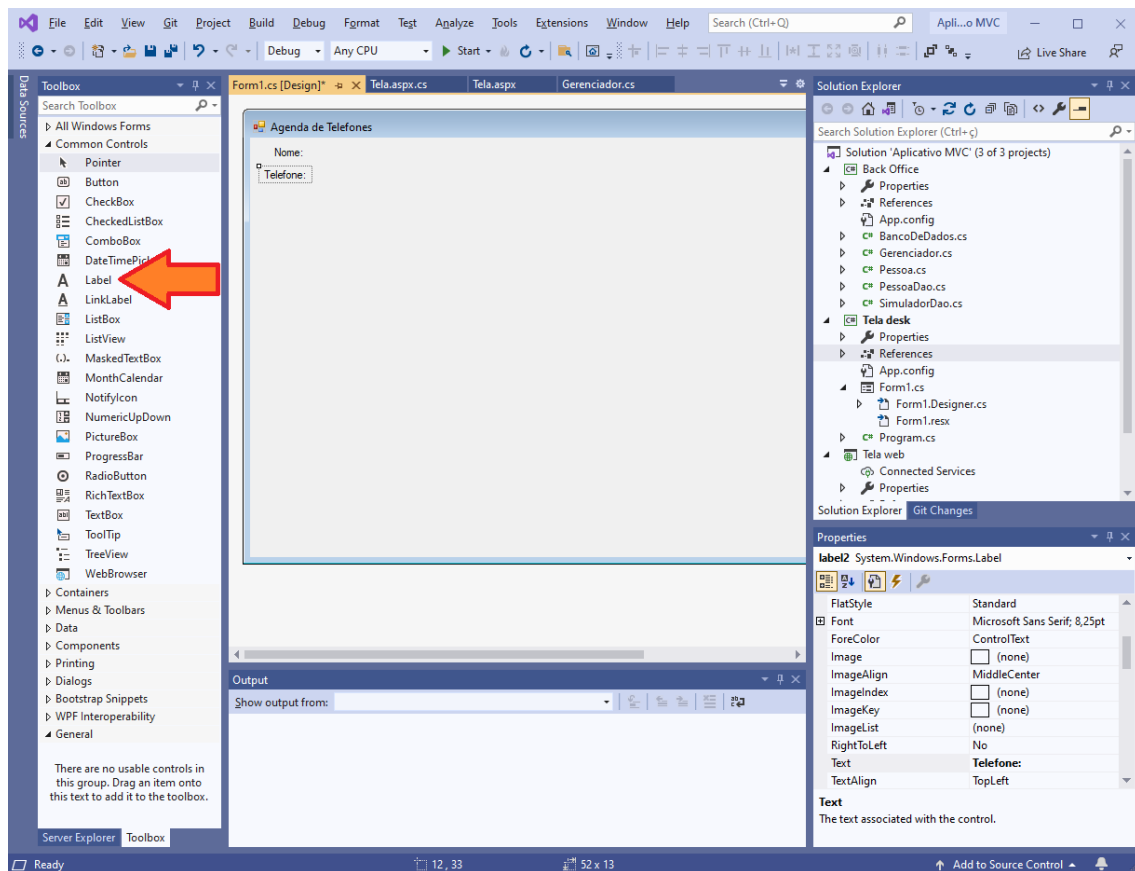
Clique sobre qualquer parte do formulário e altere sua propriedade Text para “Agenda de Telefones”.

Figura 3: Alterando o título da janela.



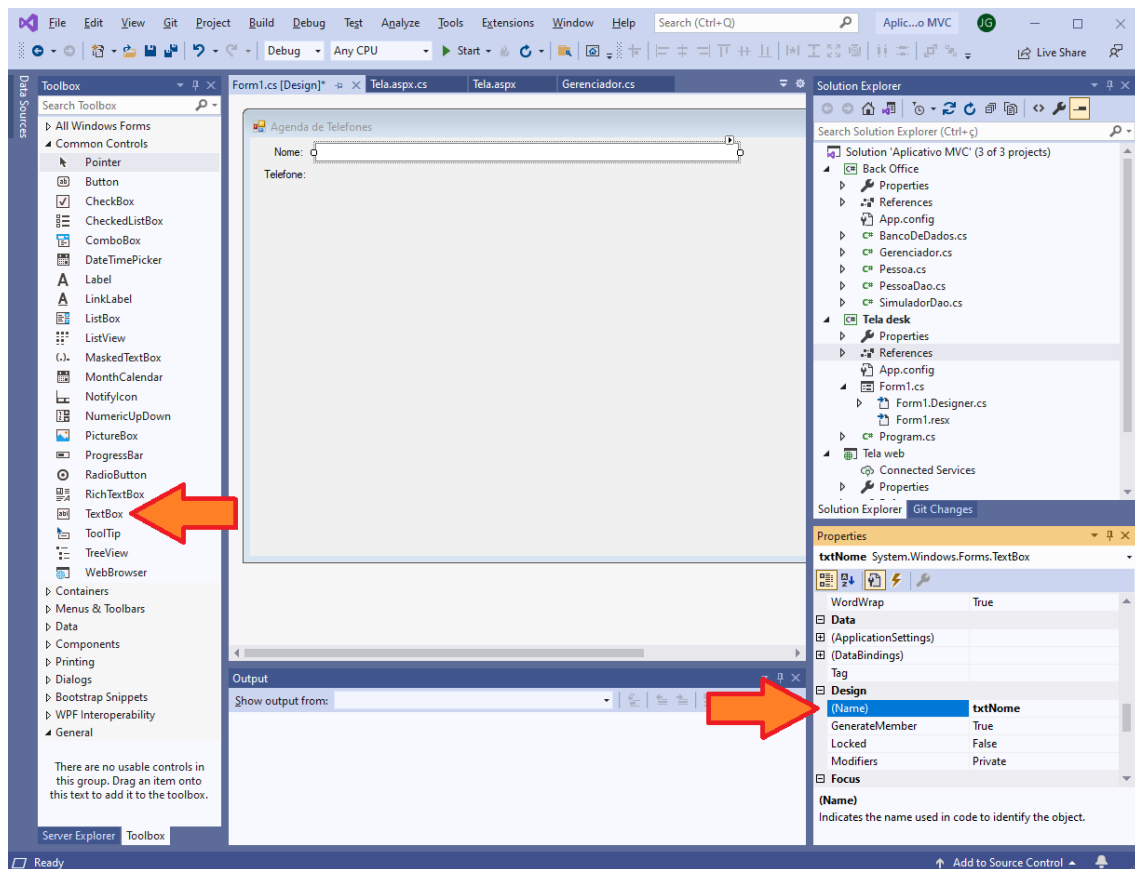
Insira dois Label no formulário, alterando suas propriedades Text para “Nome:” e “Telefone:”.

Figura 4: Inserindo os rótulos.



Insira um TextBox ao lado do rótulo “Nome:” e altere o nome do componente para txtNome. É com este nome que vamos referenciar o objeto no código.

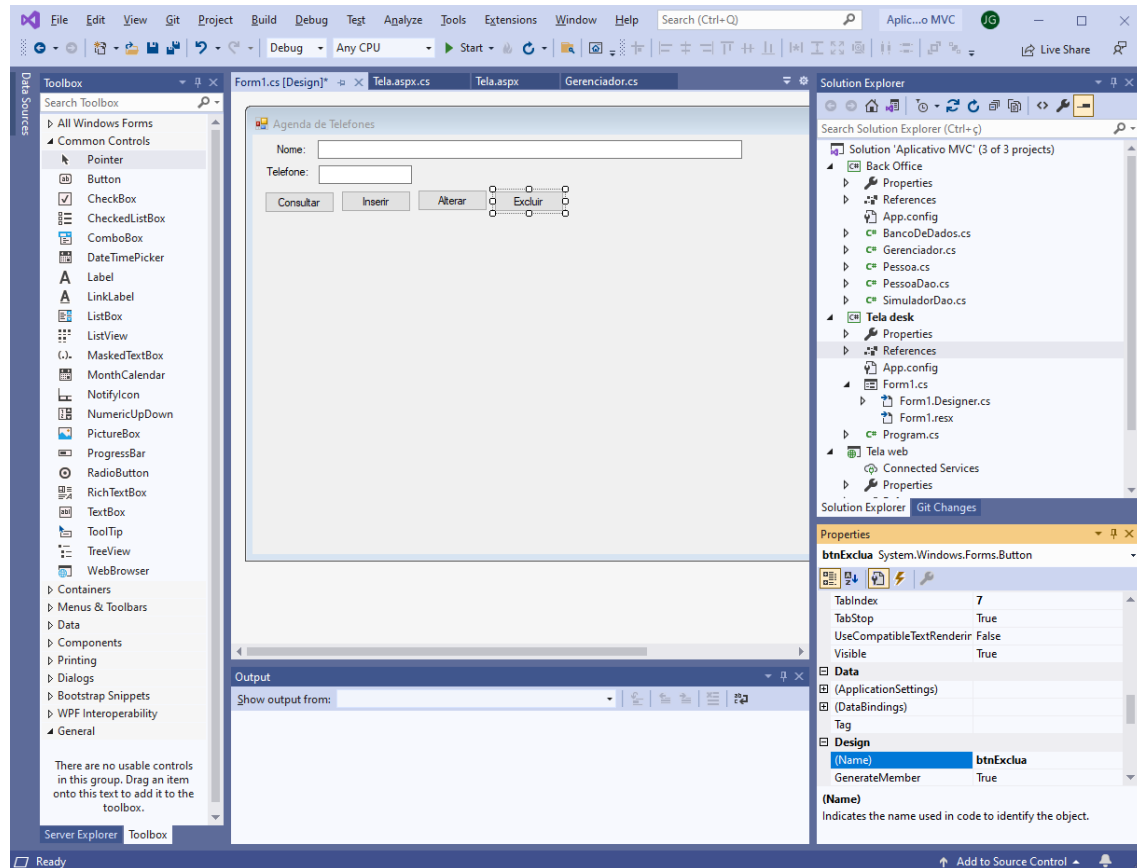
Figura 5: Inserindo o campo de texto para o nome.



Repita o processo e insira um segundo TextBox ao lado do rótulo “Telefone:”. Altere seu nome para txtTelefone.

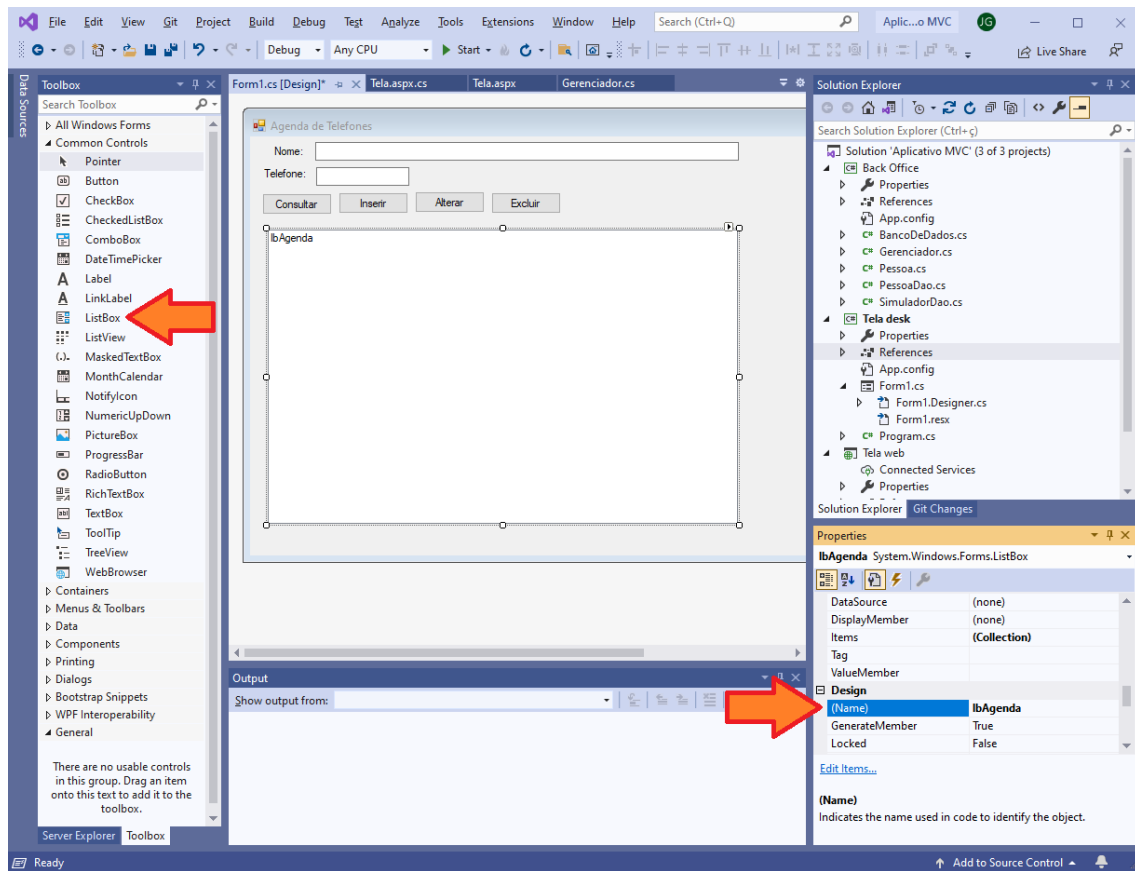
Insira quatro botões. Altere seus atributos Text para “Consultar”, “Inserir”, “Alterar” e “Excluir”. Altere também seus nomes para “btnConsulte”, “btnInsira”, “btnAltere” e “btnExclua”.

Figura 6: Inserindo e configurando os botões.



Insira um ListBox e altere seu nome para lbAgenda.

Figura 7: Inserindo o ListBox.



Dê um clique duplo em alguma área em branco do formulário ou em sua barra de título. O Visual Studio irá criar o método Form1-Load() e associá-lo ao evento de carregamento da página. Neste método, vamos instanciar um objeto da classe Gerenciador que será armazenada em um atributo da classe deste formulário. Não se esqueça de importar os pacotes modelo e controle do pacote aplicativoMVC.

Figura 8: Instanciando o objeto gerenciador.

```
1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.Data;
5  using System.Drawing;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9  using System.Windows.Forms;
10 using aplicativoMVC.controle;
11 using aplicativoMVC.modelo;
12
13 namespace Tela_desk
14 {
15     3 references
16     public partial class Form1 : Form
17     {
18         private Gerenciador gerenciador;
19         1 reference
20         public Form1()
21         {
22             InitializeComponent();
23
24         1 reference
25         private void Form1_Load(object sender, EventArgs e)
26         {
27             gerenciador = new Gerenciador(BancoDeDados.Simulador);
28         }
29     }
30 }
```



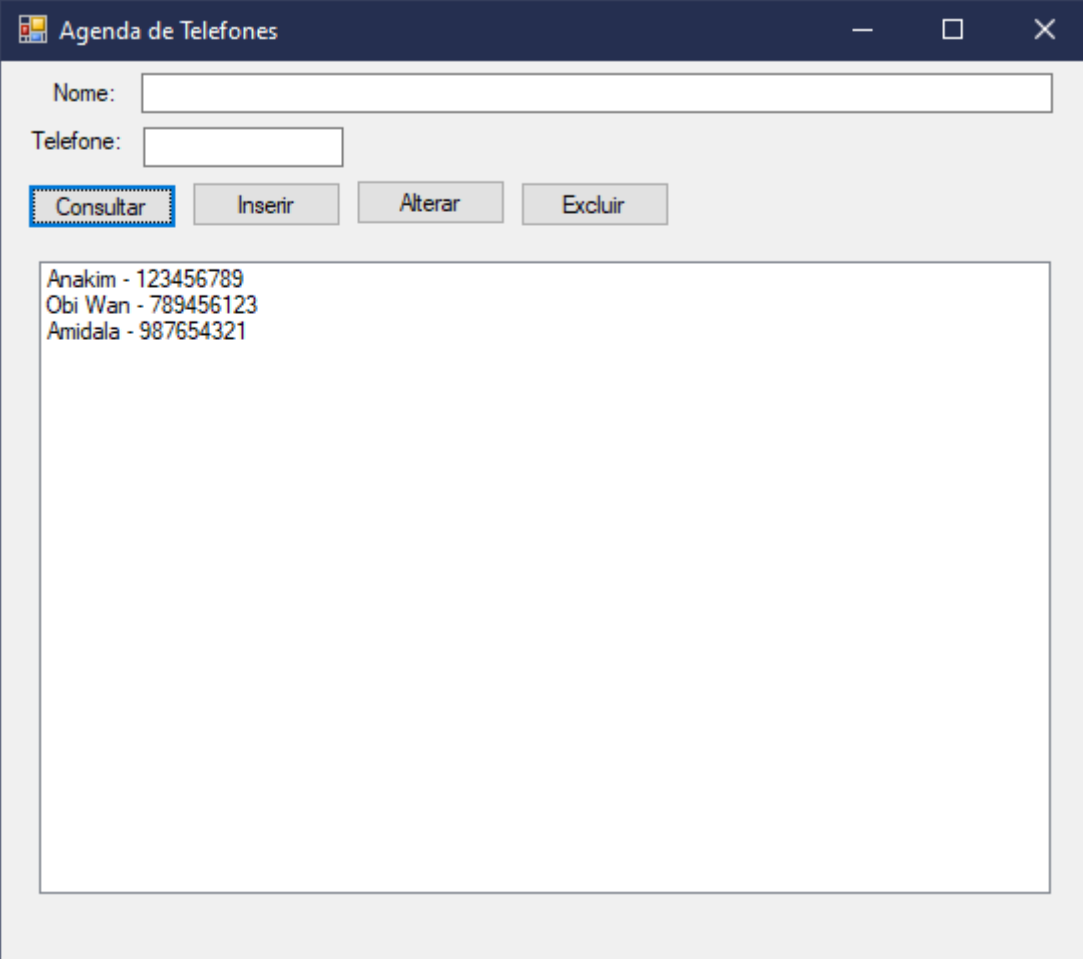
Dê um clique duplo em cada um dos botões para criar e associar os métodos que serão executados quando cada um dos botões for clicado. Implemente o comportamento dos métodos da mesma maneira que fizemos na interface gráfica web.

Figura 9: Código da classe associada ao formulário desktop.

```
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.Data;
5 using System.Drawing;
6 using System.Linq;
7 using System.Text;
8 using System.Threading.Tasks;
9 using System.Windows.Forms;
10 using aplicativoMVC.controle;
11 using aplicativoMVC.modelo;
12
13 namespace Tela_desk
14 {
15     3 references
16     public partial class Form1 : Form
17     {
18         1 reference
19         private Gerenciador gerenciador;
20         public Form1()
21         {
22             InitializeComponent();
23         }
24
25         1 reference
26         private void Form1_Load(object sender, EventArgs e)
27         {
28             gerenciador = new Gerenciador(BancoDeDados.Simulador);
29         }
30
31         1 reference
32         private void btnConsulte_Click(object sender, EventArgs e)
33         {
34             List<Pessoa> pessoas;
35             if (txtNome.Text.Equals(""))
36             {
37                 pessoas = gerenciador.dao.consulte();
38             }
39             else
40             {
41                 pessoas = gerenciador.dao.consulte(txtNome.Text);
42             }
43             lbAgenda.Items.Clear();
44             foreach (Pessoa p in pessoas)
45             {
46                 lbAgenda.Items.Add(p.ToString());
47             }
48         }
49
50         1 reference
51         private void btnInsira_Click(object sender, EventArgs e)
52         {
53             Pessoa p = new Pessoa();
54             p.nome = txtNome.Text;
55             p.telefone = Convert.ToInt32(txtTelefone.Text);
56             gerenciador.dao.insira(p);
57             txtNome.Text = "";
58             txtTelefone.Text = "";
59         }
60
61         1 reference
62         private void btnAltere_Click(object sender, EventArgs e)
63         {
64             Pessoa p = new Pessoa();
65             p.nome = txtNome.Text;
66             p.telefone = Convert.ToInt32(txtTelefone.Text);
67             gerenciador.dao.altere(p);
68         }
69
70         1 reference
71         private void btnExclua_Click(object sender, EventArgs e)
72         {
73             Pessoa p = new Pessoa();
74             p.nome = txtNome.Text;
75             p.telefone = Convert.ToInt32(txtTelefone.Text);
76             gerenciador.dao.exclua(p);
77         }
78     }
79 }
```

Neste ponto você pode executar sua solução e verificar se sua interface gráfica com o usuário em modalidade desktop está funcionando.

Figura 10: Interface gráfica com o usuário em modalidade desktop.



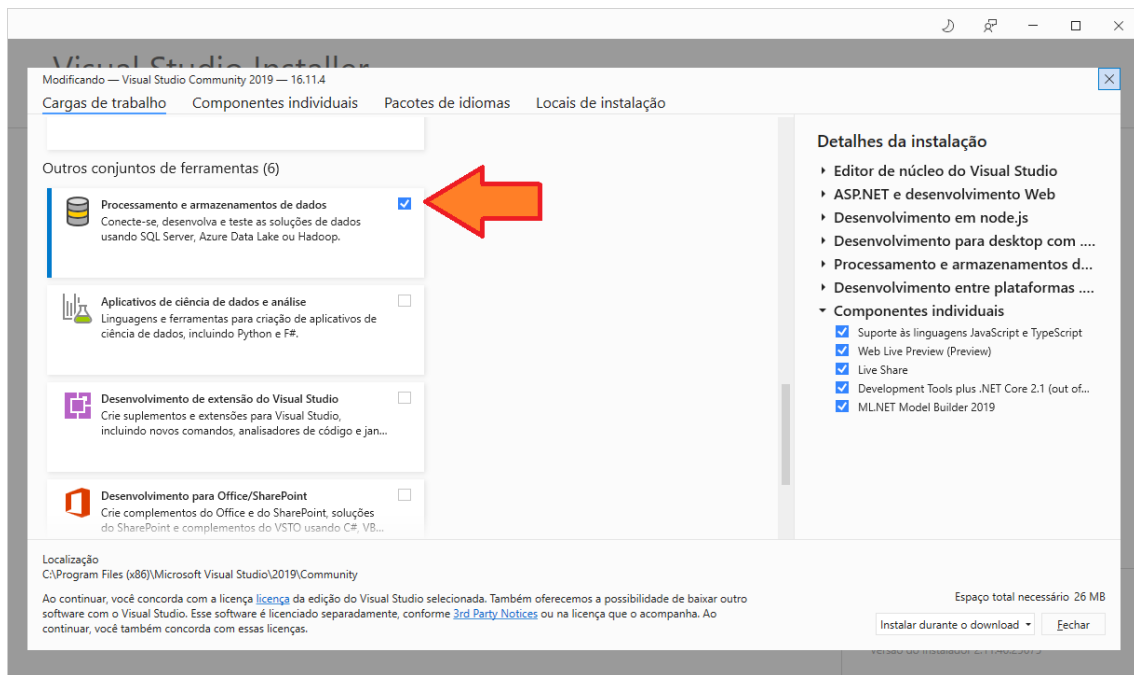
The image shows a desktop application window titled "Agenda de Telefones". The window has a dark blue title bar with standard Windows window controls (minimize, maximize, close). Below the title bar, there are two text input fields: "Nome:" and "Telefone:". Below these fields, there are four buttons: "Consultar" (highlighted with a blue border), "Inserir", "Alterar", and "Excluir". Below the buttons, there is a large text area displaying a list of contacts:

- Anakim - 123456789
- Obi Wan - 789456123
- Amidala - 987654321

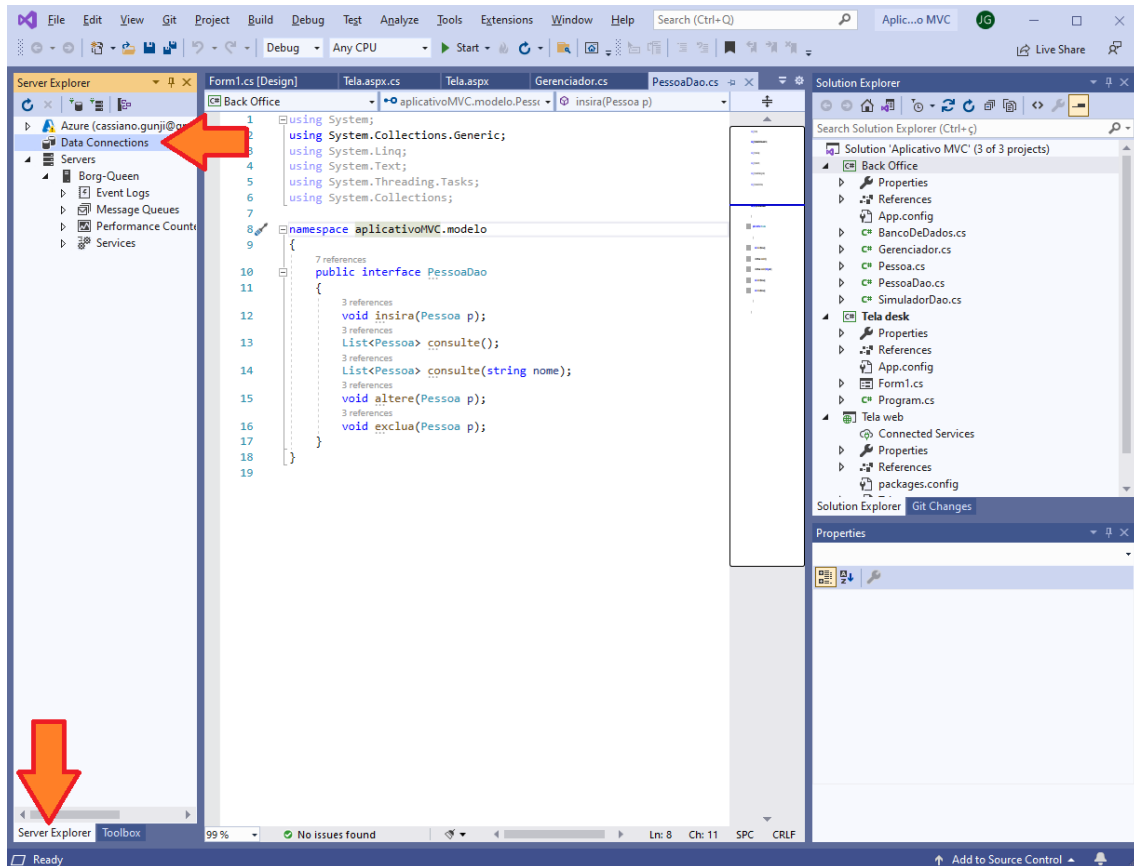
Já temos um aplicativo com duas modalidades de acesso, desktop e web, que consegue simular acesso a banco de dados. Vamos agora criar um banco de dados para armazenar esta agenda de telefones. Vamos também criar uma nova classe DAO para prover acesso a tal banco de dados. Como esta nova classe DAO vai respeitar a interface PessoaDao que criamos justamente para padronizar as classes DAO, nosso sistema vai conseguir usar esta nova classe facilmente para acessar este banco de dados real.

Abra o seu gerenciador de instalação do Visual Studio e verifique se você já instalou as ferramentas de gerenciamento de banco de dados.

Figura 11: Instalando as ferramentas de processamento e armazenamento de dados.



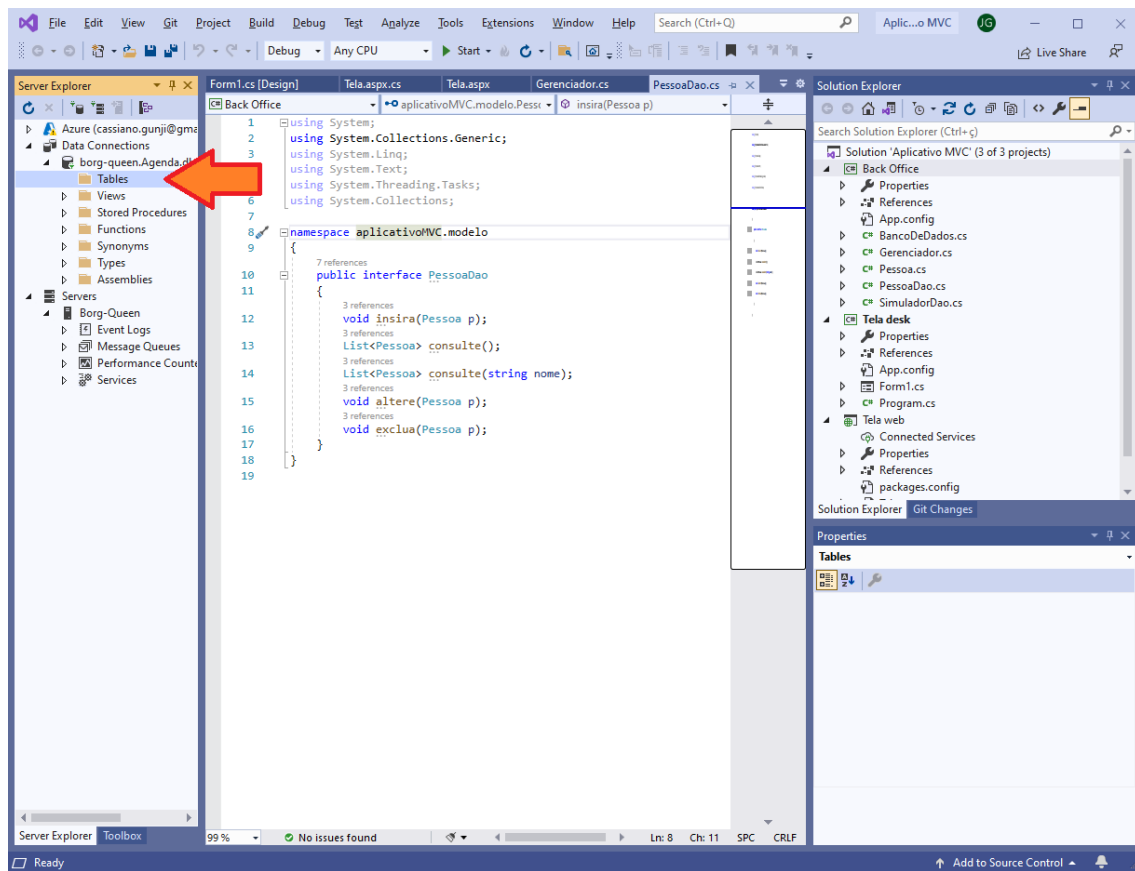
Agora, clique na aba “Server Explorer” e clique com o botão direito em “Data Connections”. Clique em [Create New SQL Server Database...].



No ComboBox, selecione uma instância de SQL Server que esteja acessível de seu computador. Deve haver ao menos uma instância em seu próprio computador. Para o nosso exemplo, vamos manter a autenticação do Windows. Ao implantar o sistema em um ambiente de produção, a autenticação deverá ser enfatizada por um servidor de domínio ou com a autenticação do próprio SQL Server. Finalmente, defina o nome do banco de dados para “Agenda”.

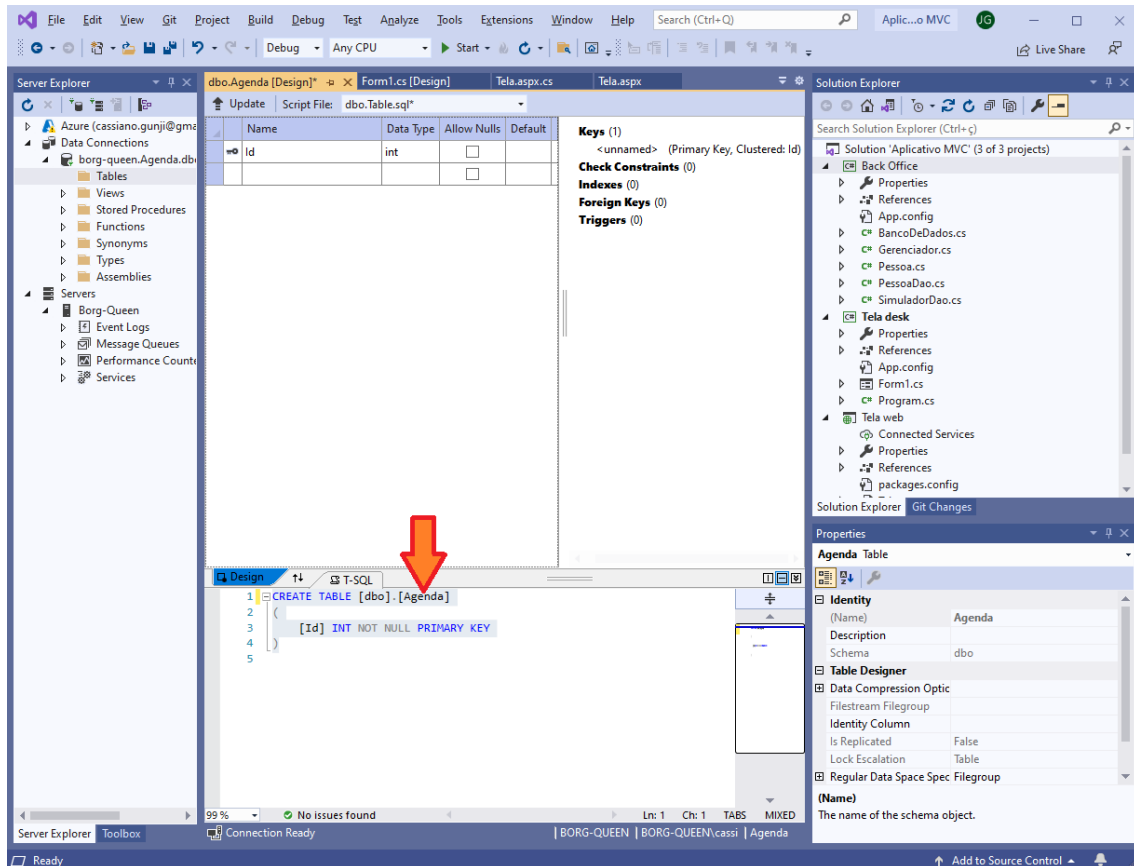
No banco de dados criado, clique com o botão direito sobre a pasta “Tables” e selecione [Add new table].

Figura 12: Criando uma nova tabela.



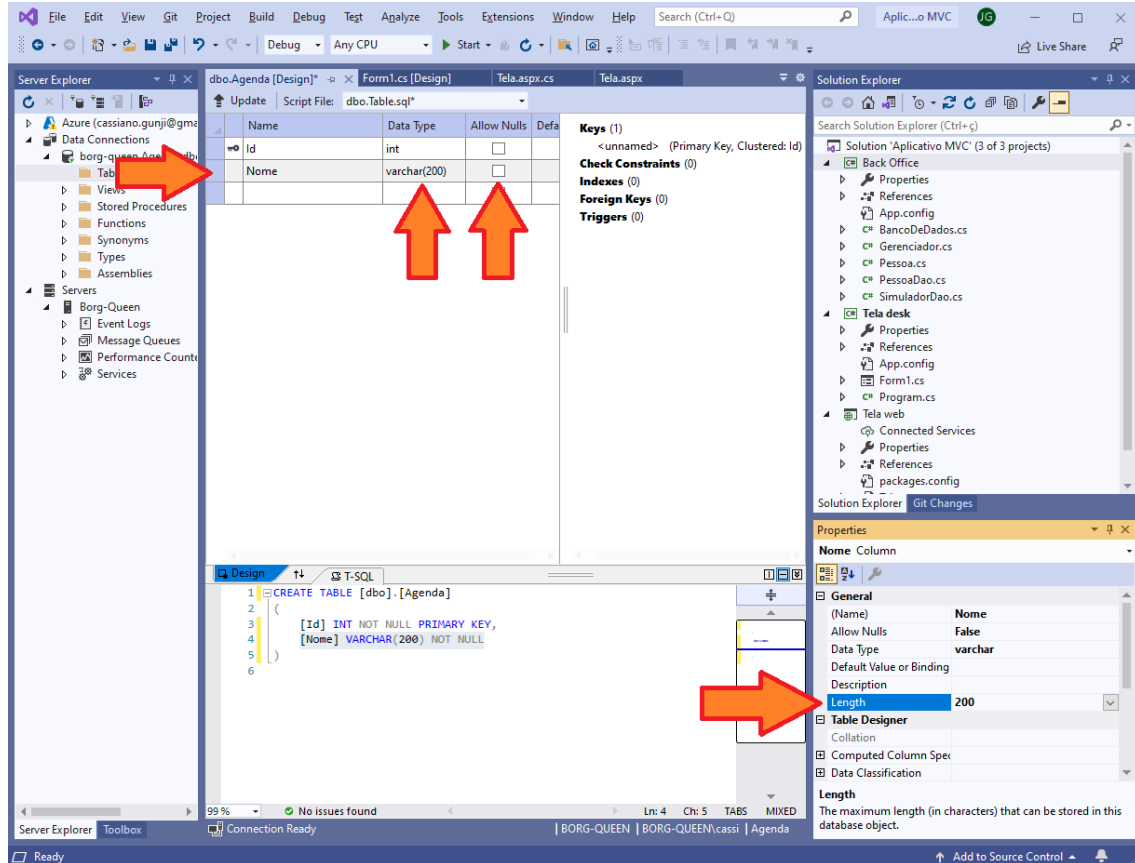
Altere o nome da tabela para “Agenda” no editor de código.

Figura 13: Alterando o nome da tabela.



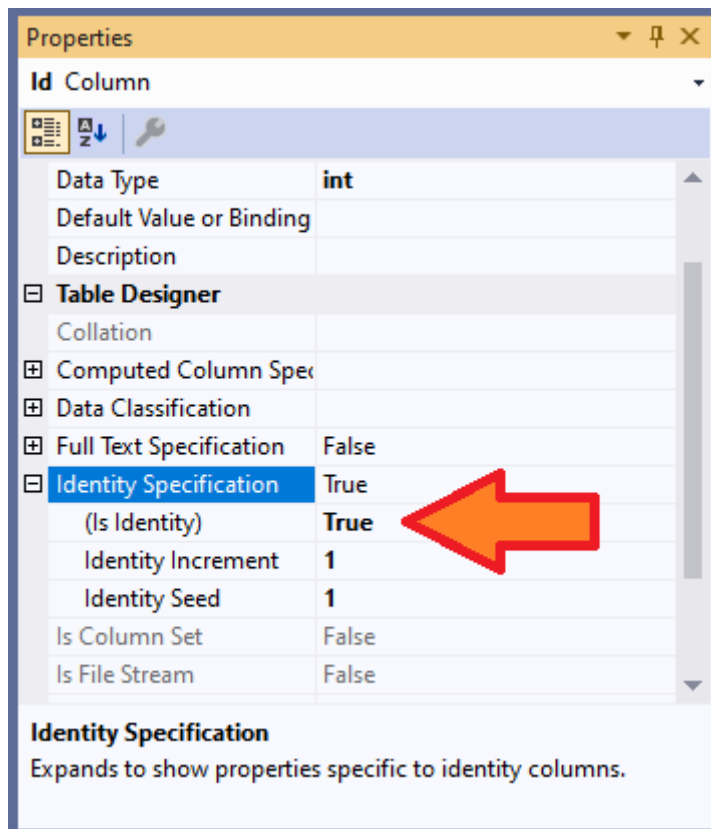
Clique no espaço em branco abaixo de “Id” e digite o nome da nova coluna “Nome”. Defina seu tipo de dado para nchar, desmarque a permissão para nulos e defina o comprimento máximo do campo para 200.

Figura 14: Criando uma nova coluna.



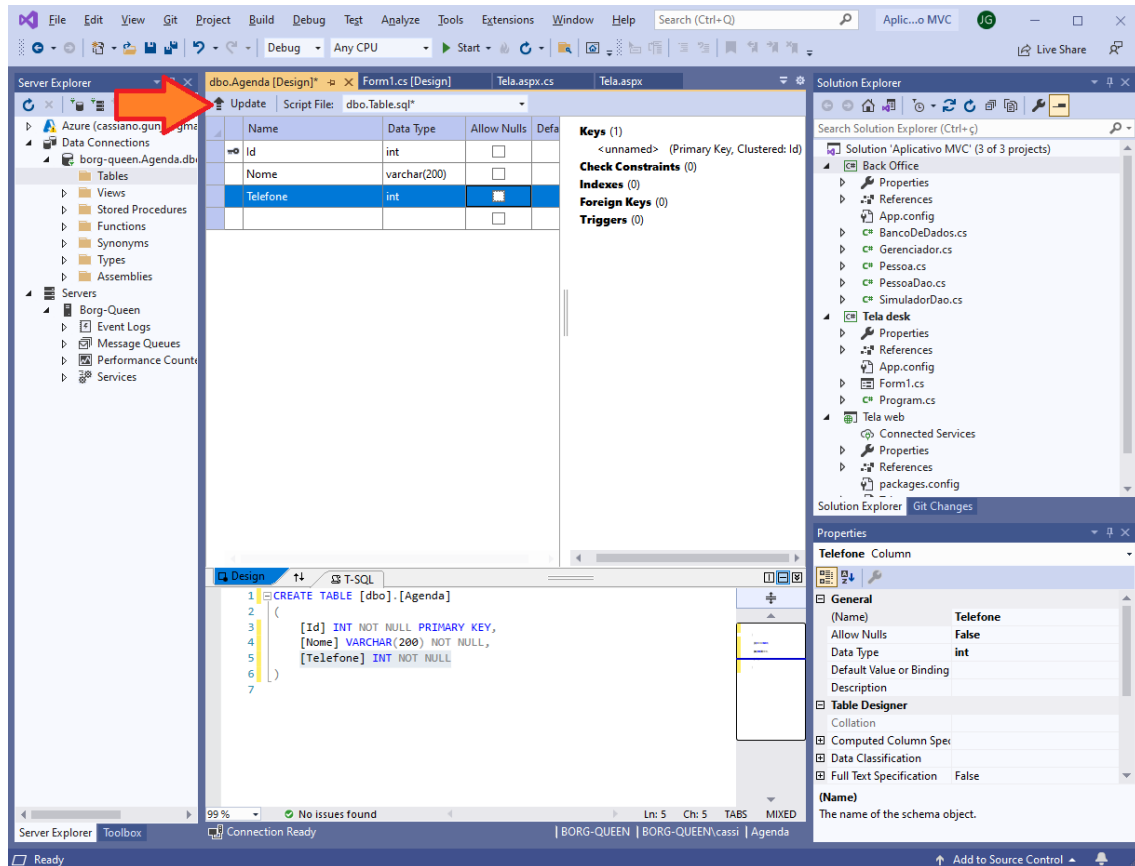
Defina o campo Id como Identity. Desta forma, o SQL Server irá criar valores incrementais automaticamente para esta coluna.

Figura 15: Definindo o campo Id como identidade.



Repita o processo para criar a coluna “Telefone” de tipo int e que não aceite nulos. Clique no botão “Update” para submeter o comando de criação ao banco de dados e criar a tabela.

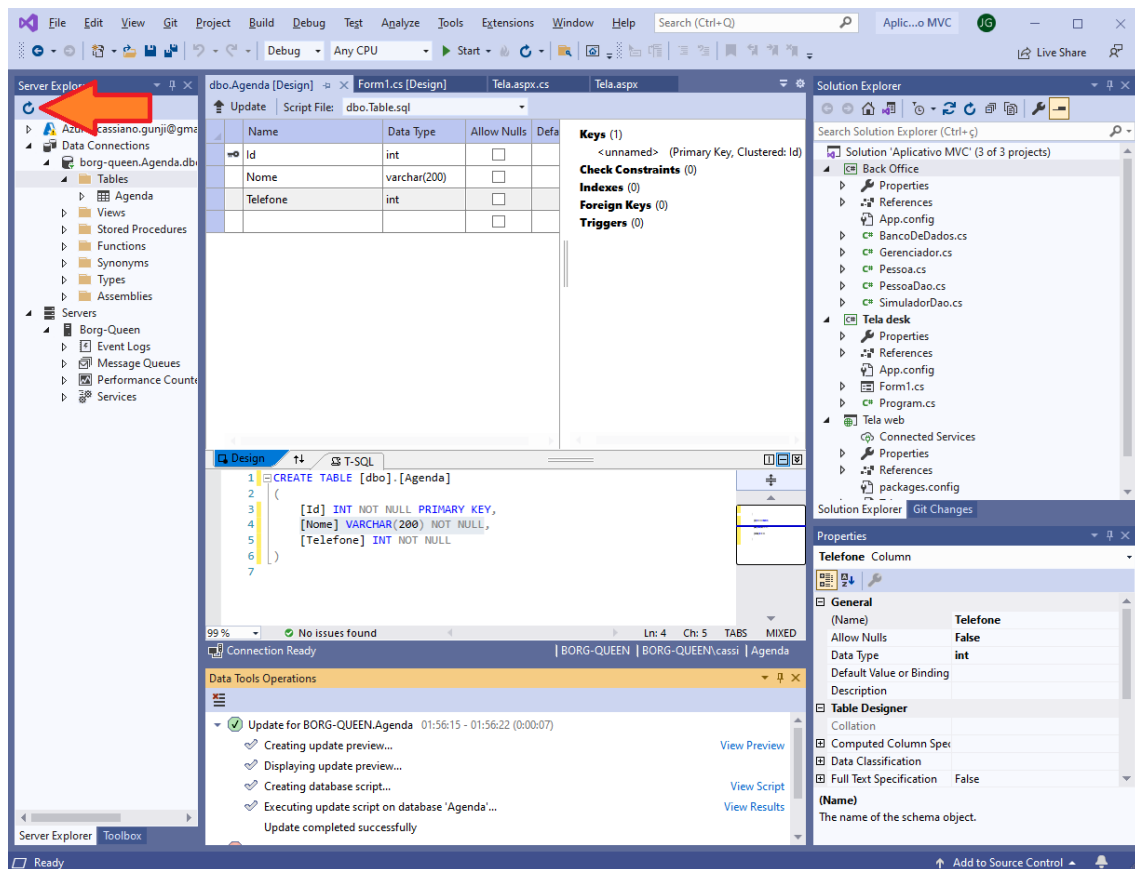
Figura 16: Criando a tabela no banco de dados.



Na janela que se abre a seguir, clique no botão [Update Database] para submeter o comando ao banco de dados e criar a tabela no seu SQL Server local.

Com a pasta Tables selecionada, clique no botão para atualizar o painel com as informações atualizadas do banco de dados. Sua nova tabela deve aparecer.

Figura 17: Atualizando o painel de servidores com as informações atuais.



Vamos agora criar nossa classe DAO para acessar este banco de dados. No projeto “Back Office”, clique com o botão direito e selecione [Add] -> [New item...] e escolha Class, dando o nome “SqlServerDao” para a sua classe. Indique que a classe realiza a interface PessoaDao. Você pode usar a solução automática para implementar todos os métodos definidos na interface.

Os detalhes de como realizar as operações de banco de dados em C# ultrapassam as intenções deste tutorial. Estes detalhes são discutidos em mais detalhes em outra disciplina. Mas é suficiente saber que o processo de acesso a banco de dados em C# (e Java) segue uma mesma “receita de bolo”:

1. Defina um string de conexão com o banco de dados;
2. Obtenha um objeto que representa uma conexão e abra a conexão com ele;
3. Obtenha um objeto que representa um comando a partir do objeto conexão.

Configure o comando SQL neste objeto;

- a. Se o comando for um comando de alteração (INSERT, UPDATE ou DELETE), execute o comando como um comando de não consulta;
- b. Se o comando for uma consulta, execute-o como tal e armazene o leitor de BD resultante. Itere sobre o leitor para obter os dados da consulta. Feche o leitor.

Termine a implementação de sua classe `SqlServerDao` conforme o código a seguir.

Figura 18: Inclusões, declaração da classe `SqlServerDao`, seu atributo e construtor.

```
1  using System;
2  using System.Collections.Generic;
3  using System.Data.SqlClient;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace aplicativoMVC.modelo.dal
9  {
10     public class SqlServerDao : PessoaDao
11     {
12         // Armazena uma única instância de conexão com o BD.
13         private static SqlConnection con;
14
15         // Construtor
16         public SqlServerDao()
17         {
18             // Se não existe uma conexão aberta, cria uma.
19             if (con == null)
20             {
21                 try
22                 {
23                     // Objeto que auxilia a configurar uma conexão com BD.
24                     SqlConnectionStringBuilder cs = new SqlConnectionStringBuilder();
25                     cs.DataSource = "BORG-QUEEN";
26                     cs.InitialCatalog = "Agenda";
27                     cs.Authentication = SqlAuthenticationMethod.ActiveDirectoryIntegrated;
28                     cs.TrustServerCertificate = true;
29
30                     // Instancia e abre a conexão.
31                     con = new SqlConnection(cs.ConnectionString);
32                     con.Open();
33                 }
34                 catch (SqlException e)
35                 {
36                     Console.Error.WriteLine("Ocorreu uma exceção de BD: " + e.Message);
37                     throw new ApplicationException("Ocorreu uma exceção de BD: " + e.Message);
38                 }
39             }
40         }
41     }
```

Figura 19: Sobrecargas do método `consulte()` da classe `SqlServerDao`.

```
42 3 references
43 void PessoaDao.altere(Pessoa p)
44 {
45     SqlCommand cmd = con.CreateCommand();
46     cmd.CommandText = "UPDATE Agenda SET Telefone = " + p.telefone + " WHERE Nome = '" +
47         p.nome + "'";
48     cmd.ExecuteNonQuery();
49 }
50 3 references
51 List<Pessoa> PessoaDao.consulte()
52 {
53     List<Pessoa> pessoas = new List<Pessoa>();
54
55     SqlCommand cmd = con.CreateCommand();
56     cmd.CommandText = "SELECT Nome, Telefone FROM Agenda;";
57     SqlDataReader reader = cmd.ExecuteReader();
58     while (reader.Read())
59     {
60         Pessoa pessoa = new Pessoa();
61         pessoa.nome = reader.GetString(0);
62         pessoa.telefone = reader.GetInt32(1);
63         pessoas.Add(pessoa);
64     }
65     reader.Close();
66
67     return pessoas;
68 }
69 3 references
70 List<Pessoa> PessoaDao.consulte(string nome)
71 {
72     List<Pessoa> pessoas = new List<Pessoa>();
73
74     SqlCommand cmd = con.CreateCommand();
75     cmd.CommandText = "SELECT Nome, Telefone FROM Agenda WHERE Nome = '"
76         + nome + "'";
77     SqlDataReader reader = cmd.ExecuteReader();
78     while (reader.Read())
79     {
80         Pessoa pessoa = new Pessoa();
81         pessoa.nome = reader.GetString(0);
82         pessoa.telefone = reader.GetInt32(1);
83         pessoas.Add(pessoa);
84     }
85     reader.Close();
86
87     return pessoas;
88 }
```

Figura 20: Métodos `exclua()` e `insira()` da classe `SqlServerDao`.

```
89 3 references
    void PessoaDao.exclua(Pessoa p)
90  {
91      SqlCommand cmd = con.CreateCommand();
92      cmd.CommandText = "DELETE FROM Agenda WHERE Nome = '" + p.nome
93          + "' AND Telefone = " + p.telefone + "'";
94      cmd.ExecuteNonQuery();
95  }
96
97 3 references
    void PessoaDao.insira(Pessoa p)
98  {
99      SqlCommand cmd = con.CreateCommand();
100     cmd.CommandText = "INSERT INTO Agenda (Nome, Telefone)" +
101         " VALUES ('" + p.nome + "', " + p.telefone + ")";
102     cmd.ExecuteNonQuery();
103 }
104 }
105 }
```

Altere o construtor da classe `Gerenciador` para instanciar corretamente a classe `SqlServerDao`.

Figura 21: Alterando a classe `Gerenciador`.

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using aplicativoMVC.modelo;
7  using aplicativoMVC.modelo.dal;
8
9  namespace aplicativoMVC.controle
10 {
11     5 references
    public class Gerenciador
12     {
13         public PessoaDao dao;
14         2 references
        public Gerenciador(BancoDeDados bd)
15         {
16             switch (bd)
17             {
18                 case BancoDeDados.Simulador:
19                     dao = new SimuladorDao();
20                     break;
21                 case BancoDeDados.MySql:
22                     throw new NotImplementedException("DAO para MySql não implementado");
23                 case BancoDeDados.SqlServer:
24                     dao = new SqlServerDao();
25                     break;
26             }
27         }
28     }
29 }
```

Em sua classe Tela.aspx.cs, altere o método Page_Load() para que ele instancie um gerenciador usando o banco de dados SQL Server.

Figura 22: Alterando a interface gráfica web para usar o banco de dados SQL Server.

```
1  + using ...
9
10 - namespace Tela_web
11 {
12     2 references
13     public partial class Tela : System.Web.UI.Page
14     {
15         Gerenciador gerenciador;
16         0 references
17         protected void Page_Load(object sender, EventArgs e)
18         {
19             gerenciador = new Gerenciador(BancoDeDados.SqlServer);
20         }
21     }
22     0 references
23     protected void btnConsulte_Click(object sender, EventArgs e) ...
24     0 references
25     protected void btnInsira_Click(object sender, EventArgs e) ...
26     0 references
27     protected void btnAltere_Click(object sender, EventArgs e) ...
28     0 references
29     protected void btnExclua_Click(object sender, EventArgs e) ...
30 }
31 }
```



Faça o mesmo para configurar sua classe Form1.cs

Figura 23: Alterando a interface gráfica desktop para usar o banco de dados SQL Server.

```
1  + using ...
12
13  namespace Tela_desk
14  {
15      3 references
16      public partial class Form1 : Form
17      {
18          private Gerenciador gerenciador;
19          1 reference
20          public Form1()...
21
22          1 reference
23          private void Form1_Load(object sender, EventArgs e)
24          {
25              gerenciador = new Gerenciador(BancoDeDados.SqlServer);
26          }
27
28          1 reference
29          private void btnConsulte_Click(object sender, EventArgs e)...
30
31          1 reference
32          private void btnInsira_Click(object sender, EventArgs e)...
33
34          1 reference
35          private void btnAltere_Click(object sender, EventArgs e)...
36
37          1 reference
38          private void btnExclua_Click(object sender, EventArgs e)...
39
40      }
41  }
```

Neste ponto você pode executar novamente seu aplicativo em ambas as modalidades. Repare que os dados inseridos agora são persistentes, ou seja, estão sendo mantidos pelo seu banco de dados gerenciado pelo SQL Server instalado em seu computador.

Caso você esteja encontrando dificuldades, você pode consultar o código que foi desenvolvido no seguinte repositório: <https://github.com/Cassiano-Gunji/Aplicativo-MVC>.