

## C# - Exemplo MDI

Nesta aula criaremos uma aplicação simples de cadastro para avaliar o nível de dificuldade/facilidade que é desenvolver *softwares* com a tecnologia C# (lê-se Cê Sharp).

No Visual Studio Express 2013 for Desktop Windows, crie um novo projeto (aqui também chamado de *Solution*) (*File | New Project...*). Na 1 é apresentada a tela de criação de novo projeto. Escolha *Windows Forms Application*, no campo *Name* coloque o nome *ProjetoExemploMDI* que será o nome do formulário criado, no campo *Location* escolha um local adequado para salvar o seu projeto, no campo *Solution* deixe como está e no campo *Solution Name* dê o nome de "SolutionExemploMDI". Por fim clique no botão *Ok*.

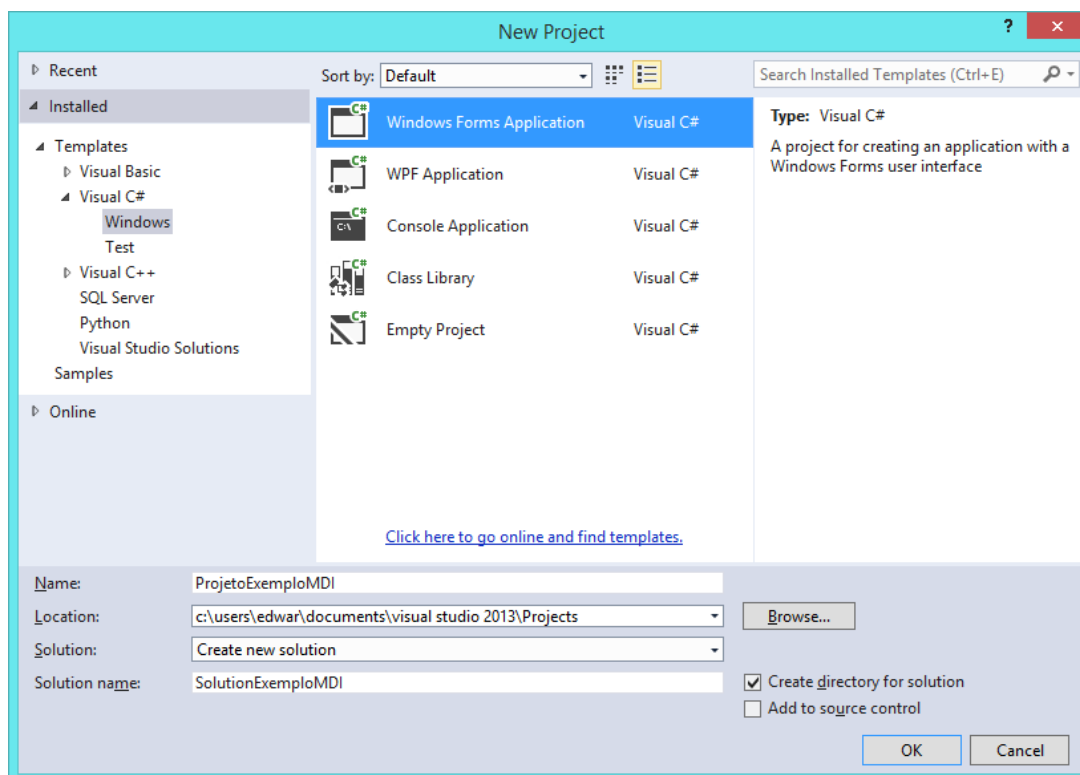


Figura 1: Tela de criação de novo projeto.

Na construção deste sistema simplificaremos ao máximo o nível de detalhamento das telas, pois, temos três fatores primordiais a serem levados em consideração:

- não é objetivo desta aula ensinar como se deve construir de sistemas de informação,
- o tempo é curto e
- quanto mais simples for o sistema, mais fácil será para avaliarmos o grau de dificuldade de utilização da tecnologia.

Sua tela do Visual Studio Express 2013 for Desktop Windows, doravante tratado apenas por VS, deve estar semelhante à apresentada na 2.

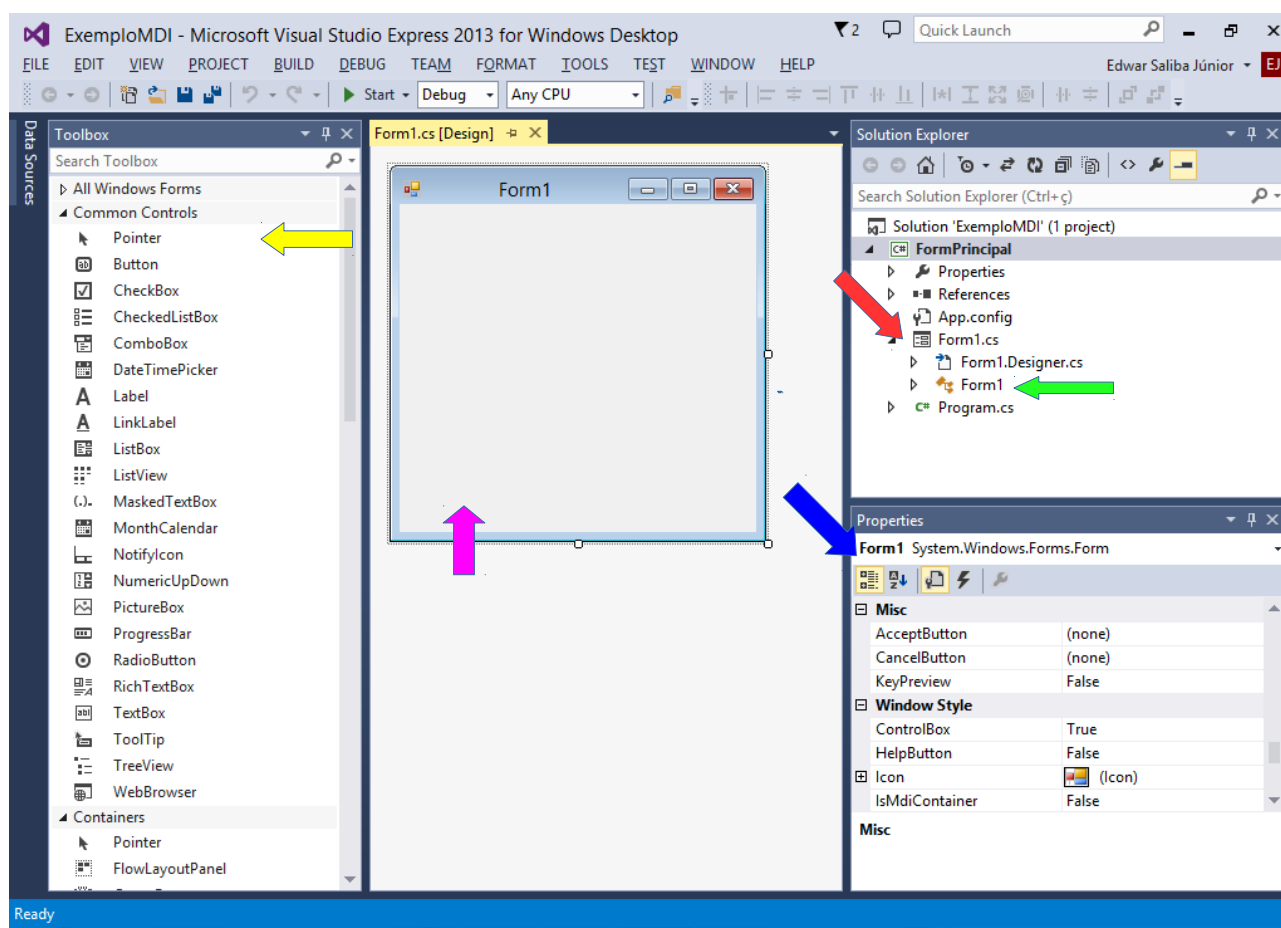


Figura 2: Criação do Projeto "ExemploMDI"

## Tela “Principal”

A tela Principal será constituída apenas de um *menu* para acesso aos demais formulários que comporão o sistema.

## Adicionando Componentes ao Formulário

Vamos começar a montar a nossa tela *Principal*. Para isto, vamos observar e entender alguns pontos importantes na IDE<sup>1</sup> VS:

- seta vermelha na 2, este arquivo representa um formulário visual, neste caso, o que está sendo apontado pela seta rosa. O arquivo apontado pela seta verde é exatamente a parte de codificação do formulário, ou seja, a classe com seus

<sup>1</sup> *Integrated Development Environment*, do inglês, Ambiente Integrado de Desenvolvimento.

atributos e métodos. Ambos arquivos estão situados na área da IDE conhecida como *Solution Explorer*;

- a seta amarela na 2, mostra-nos a área onde ficam os componentes visuais disponíveis na IDE, esta área é conhecida como *ToolBox*. E por último,
- a seta azul na 2 aponta exatamente para a área que leva o nome de *Properties* e que representa as propriedades do componente que estiver em foco.

## ***Inclusão dos Formulários que Comporão o Sistema***

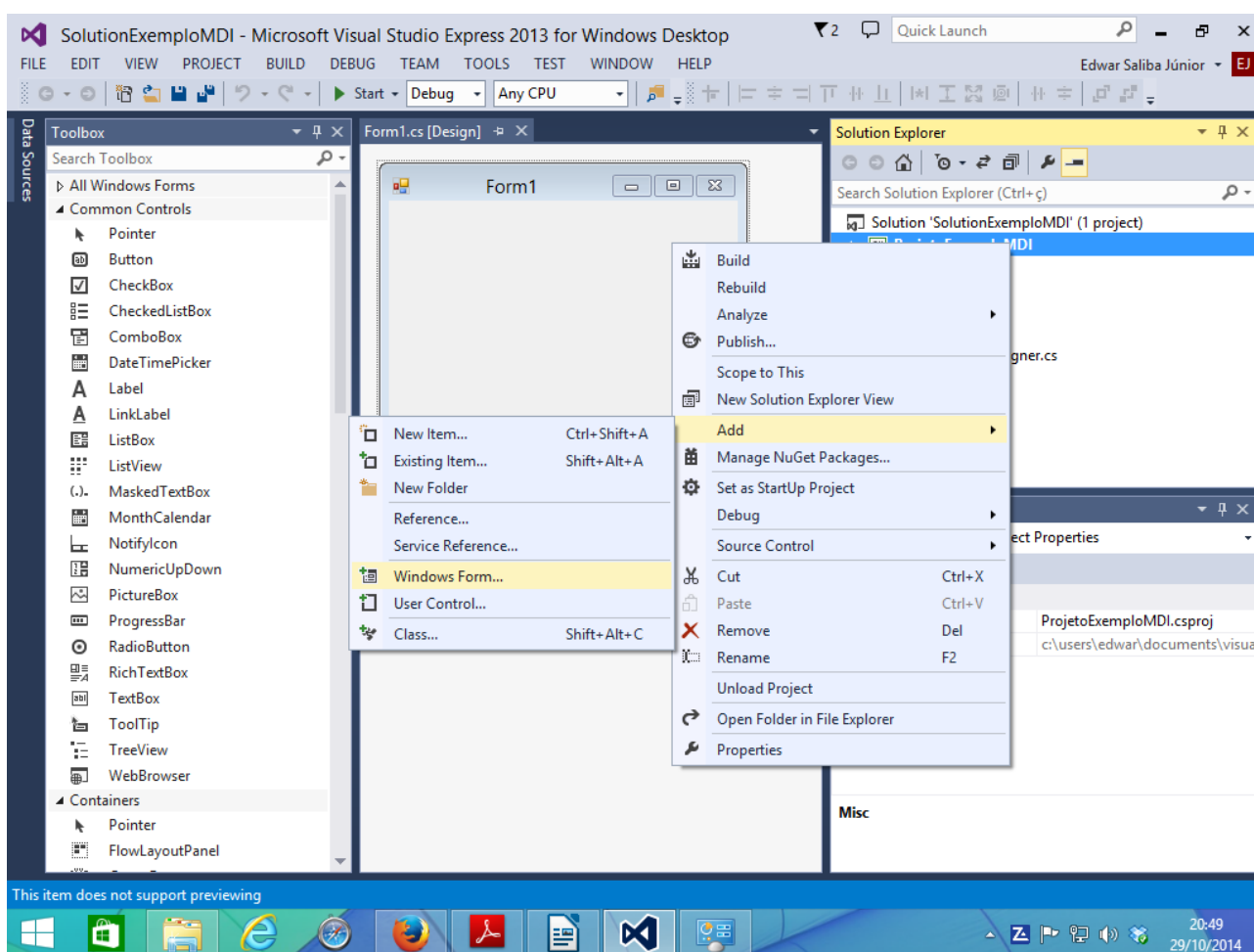


Figura 3: Adição de novos formulários ao projeto.

Vamos adicionar todos os formulários que comporão nosso projeto. Para isto, como mostrado na 3, clique com o botão direito do *mouse* sobre o nome do projeto e escolha *Add | Windows Form...*

Na tela que abrir escolha *Windows Form*, dê um nome para o formulário e aperte o botão *Add*.

Você deverá criar os seguintes formulários:



- *FormClienteEdicao,*
- *FormClientesVisao,*
- *FormEstadoEdicao,*
- *FormEstadosVisao e*
- *FormSobre.*

## **Modelando os Formulários com Componentes Visuais**

### **FormPrincipal**

Dando continuidade ao nosso trabalho, dê um clique duplo no formulário *FormPrincipal*. Ele será mostrado na área central do VS. Vamos até a *ToolBox* e selecionar um componente *MenuStrip* e colar no formulário. Na área que aparecer no componente *MenuStrip* digite a palavra *Arquivo*. Clique na palavra *arquivo* e adicione um *submenu* *Sair*.

Agora, do lado direito do *menu Arquivo*, digite *Cadastro* e do lado digite *Ajuda*. Clique no *menu Cadastro* e adicione os *submenus* *Clientes* e *Estados*. Clique no *menu Ajuda* e adicione o *submenu* *Sobre*.

Vamos aproveitar para configurar algumas propriedades do *FormPrincipal*, portanto, vá até *Properties* e na propriedade *Text* digite *Principal*, na propriedade *WindowState* escolha *Maximized* e na propriedade *isMDIContainer* modifique o valor para *True*.

Sua tela deverá estar semelhante à apresentada na 4.

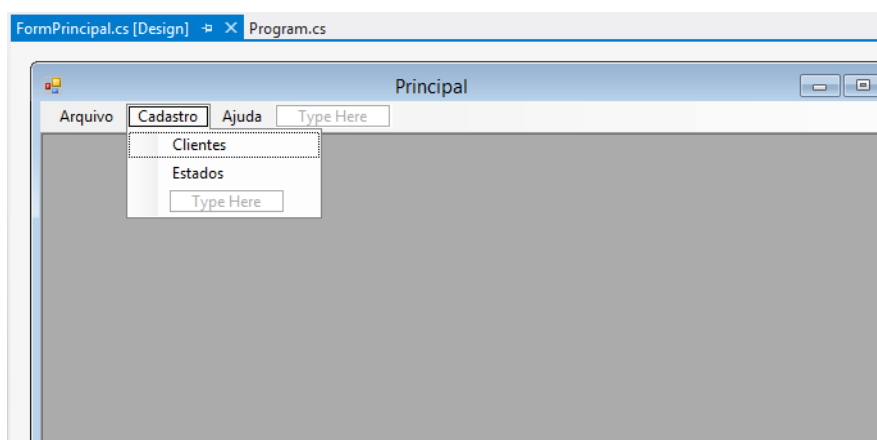


Figura 4: Adição de menu e propriedade ao *FormPrincipal*.

### **FormClienteVisao**

Agora dê um clique duplo no formulário *FormClientesVisao* (no *Solution Explorer*) para abri-lo na IDE e vamos fazer o seguinte:

- adicione a este formulário, no canto superior esquerdo, quatro componentes *Button*. Clique em um por um e modifique suas propriedades:
  - *Text* para: *Novo*, *Alterar*, *Excluir* e *Atualizar* e respectivamente a propriedade
  - *Name* para *btnNovo*, *btnAlterar*, *btnExcluir* e *btnAtualizar*.

- Adicione um componente *DataGridView* e altere as seguintes propriedades do mesmo:
  - *Name* para *dgvClientes*,
  - *AllowUserToAddRows* e *AllowUserToDeleteRows* para *False*,
  - *MultiSelect* para *False*,
  - *ReadOnly* para *True* e
  - *SelectionMode* para *FullRowSelect*.

Após toda esta operação, seu formulário deverá estar semelhante ao apresentado na 5.

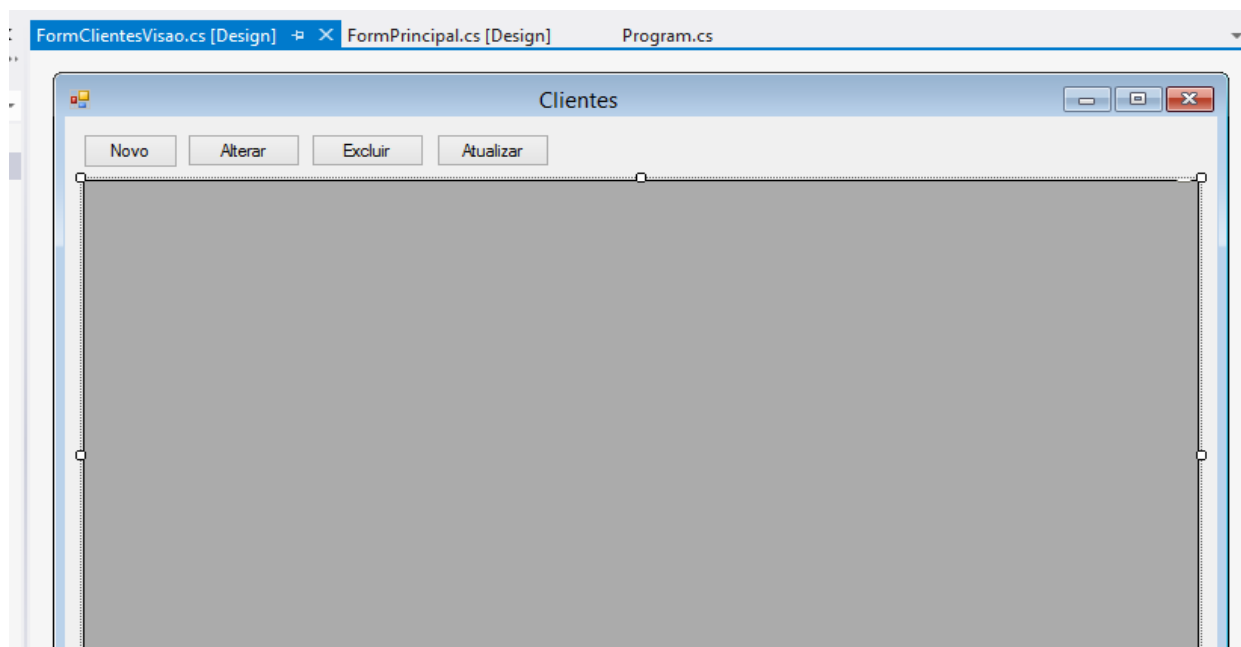


Figura 5: *FormClientesVisao* após colocação dos componentes.

## FormEstadosVisao

Abra o formulário *FormEstadosVisao* e da mesma forma como foi feito com o *FormClientesVisao*, faça com este. Ou seja, pegue quatro *Buttons* e um *DataGridView* e coloque no formulário.

Configure as mesmas propriedades de forma idêntica ao do formulário anterior, não se esquecendo de modificar apenas o nome.

E seu formulário deverá, depois, de toda a operação realizada, se aparentar ao mostrado na 6.

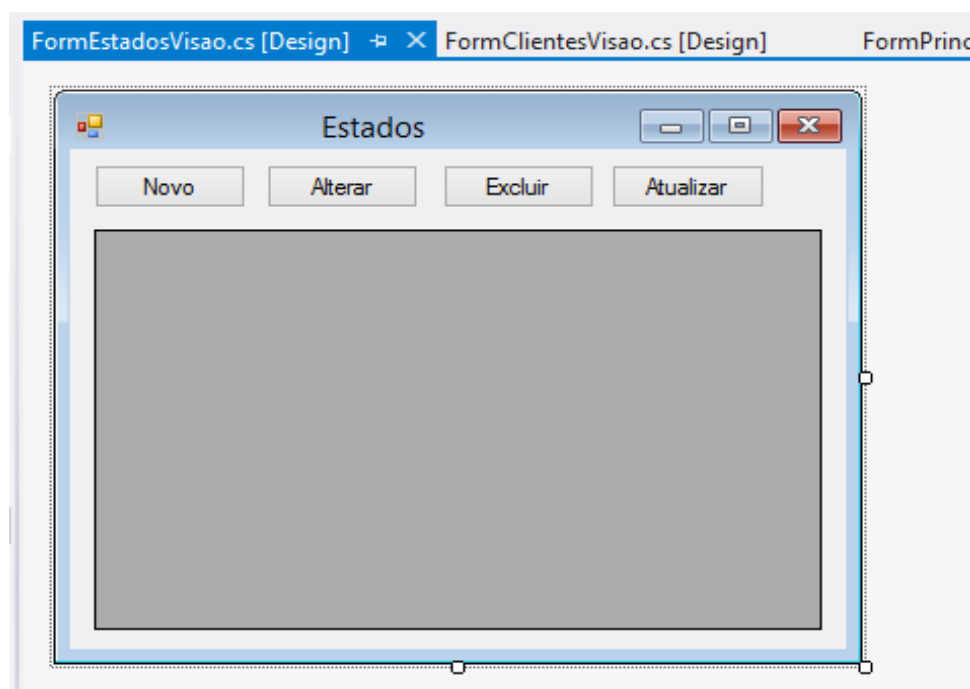


Figura 6: *FormEstadosVisao* após colocação e configuração dos componentes.

## FormClienteEdicao

Abra o *FormClientesEdicao* e comece a construção. Para este formulário serão necessários oito *Label*'s, sete *Text Box*'s, um *ComboBox* e dois *Button*'s.

Disponha os sete componentes *Text Box* juntamente com sete componentes *Label* no formulário. Lembrando que são para preenchimento dos seguintes campos: Código, Nome, Logradouro, Número, Bairro, Município, C.E.P. e o último *Label* para o *ComboBox*. Mude as seguintes propriedades:

Nos *Label*'s:

- *Text* – coloque o um texto significativo (e. g. para o campo código, coloque *Código*)
- *Name* – coloque o mnemônico representativo do tipo do componente concatenado a um nome significativo (e. g. *lblCodigo*, o mnemônico sempre em letras minúsculas)

Nos *Text Box*'s:

- *Name* – coloque o mnemônico representativo do tipo do componente concatenado a um nome significativo (e. g. *tbxCodigo*).

Feito isto, adicione também um componente do tipo *ComboBox*, este será destinado a

escolha do estado. Nele você deverá alterar as seguintes propriedades:

- *Name* – coloque o mnemônico representativo do tipo do componente concatenado a um nome significativo (e. g. *cbxEstado*)
- *Sorted* – passe o valor desta propriedade para *True*.

Estamos acabando, agora só falta os dois *Button's*, adicione-os ao formulário um botão para *Salvar* e outro para *Fechar*. Mude as seguintes propriedades:

- *Name* – coloque o mnemônico representativo do tipo do componente concatenado a um nome significativo (e. g. *btnSalvar*),
- *Text* – coloque um texto significativo para a função que o botão desempenhará na tela.

Terminamos! Seu formulário deve estar semelhante ao apresentado na 7.

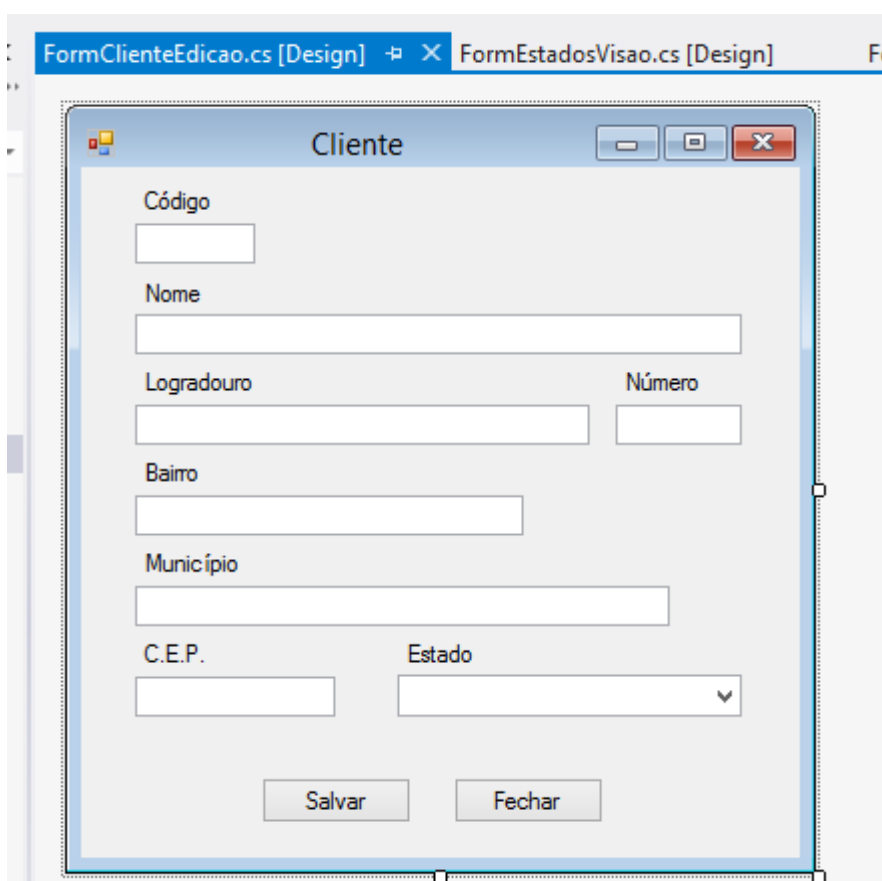


Figura 7: *FormClientesEdicao* após colocação e configuração dos componentes.

## FormEstadoEdicao

Para o *FormEstadoEdicao* desenvolva o mesmo processo de componentização que foi feito para o *FormClienteEdicao*. Lembrando que para este formulário só teremos três componentes do tipo *Text Box*, juntamente com três componentes do tipo *Label* e dois componentes do tipo *Button*.

Os campos a serem moldados no formulário são: *Código*, *Nome* e *Sigla* do estado. E os botões serão: *Salvar* e *Fechar*.

Ao final seu formulário deverá estar semelhante ao apresentado na 8.

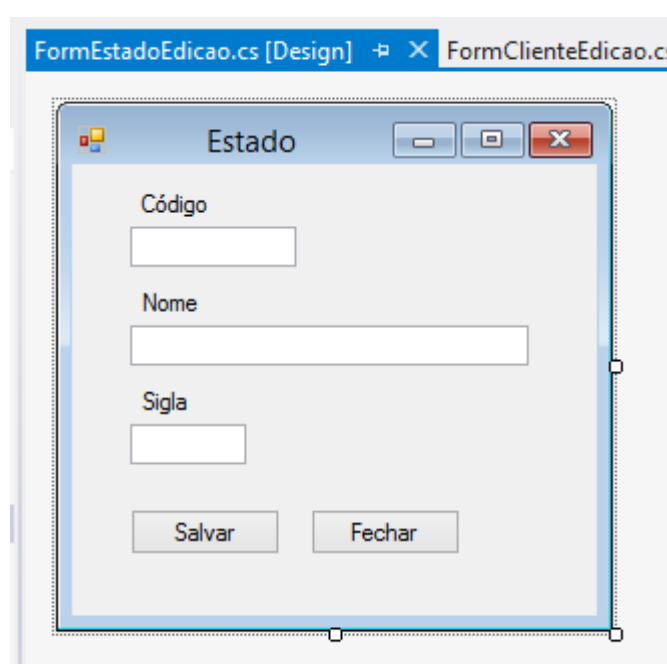


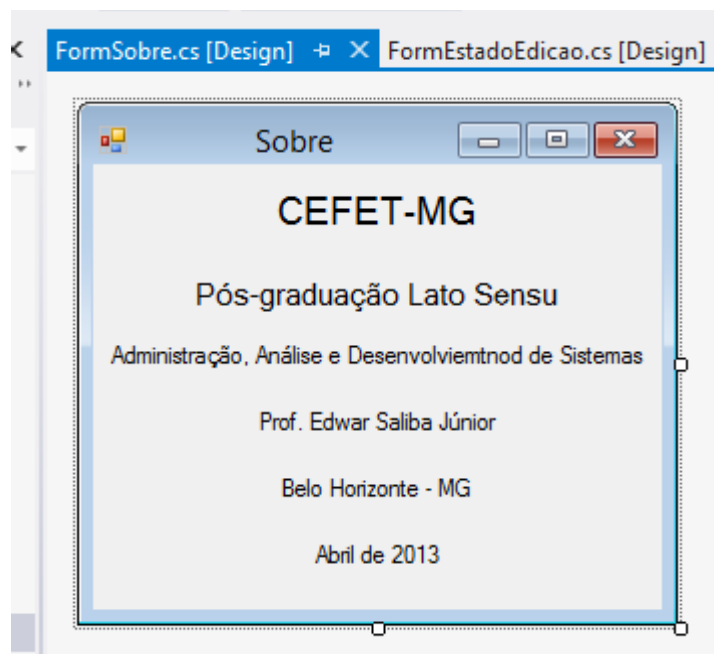
Figura 8: *FormEstadoEdicao* após colocação e configuração dos componentes.

## FormSobre

O *FormSobre* é o mais simples de todos. Ele conterá apenas seis componentes do tipo *Label*, a não ser que você queira adicionar mais informações.

Em princípio, faça o *FormSobre* semelhante ao mostrado na 9. No lugar do nome do professor coloque o seu nome, pois, afinal de contas, é você quem está desenvolvendo o sistema.

Qualquer dúvida pergunte ao professor.



*Figura 9: FormSobre após colocação e configuração dos componentes.*

## **Conectando o Software ao SGBD<sup>2</sup>**

Para preenchermos o componente *DataGridView* dos formulários precisamos de dados. Estes estão armazenados nas tabelas criadas em nosso SGBD.

Para acessarmos o SGBD precisamos criar uma conexão com o mesmo. Para isto, logo adiante vamos criar uma classe especial em nosso sistema que se encarregará desta conexão.

Antes de criarmos a conexão com o SGBD precisamos de três bibliotecas para que o VS saiba como “dialogar” com o PostgreSQL. São elas: *Npgsql.dll*, *Mono.Security.dll* e *policy.2.0.Npgsql.dll*. E também os arquivos: *Npgsql.pdb*, *Npgsql.xml* e *policy.2.0.Npgsql.config*. Estes arquivos serão fornecidos pelo professor.

Estes arquivos e bibliotecas deverão ser referenciados pelo projeto. Para isto, você deverá copiá-los para a pasta *bin* do projeto.

Em seguida, vá ao VS e observe que no projeto há uma pasta chamada *References* (seta vermelha na 10). Clique com o botão direito do *mouse* sobre esta pasta e escolha a opção *Add Reference...*

Na janela que abrirá, aperte o botão *Browse...* (seta azul na 10) e, na nova janela, vá até a pasta onde você colocou as bibliotecas, selecione as três e aperte o botão *Add* (seta amarela na 10) e em seguida o botão *Ok* (seta verde na 10). E pronto! Desta forma as devidas referências foram adicionadas.

---

<sup>2</sup> Sistema Gerenciador de Banco de Dados.

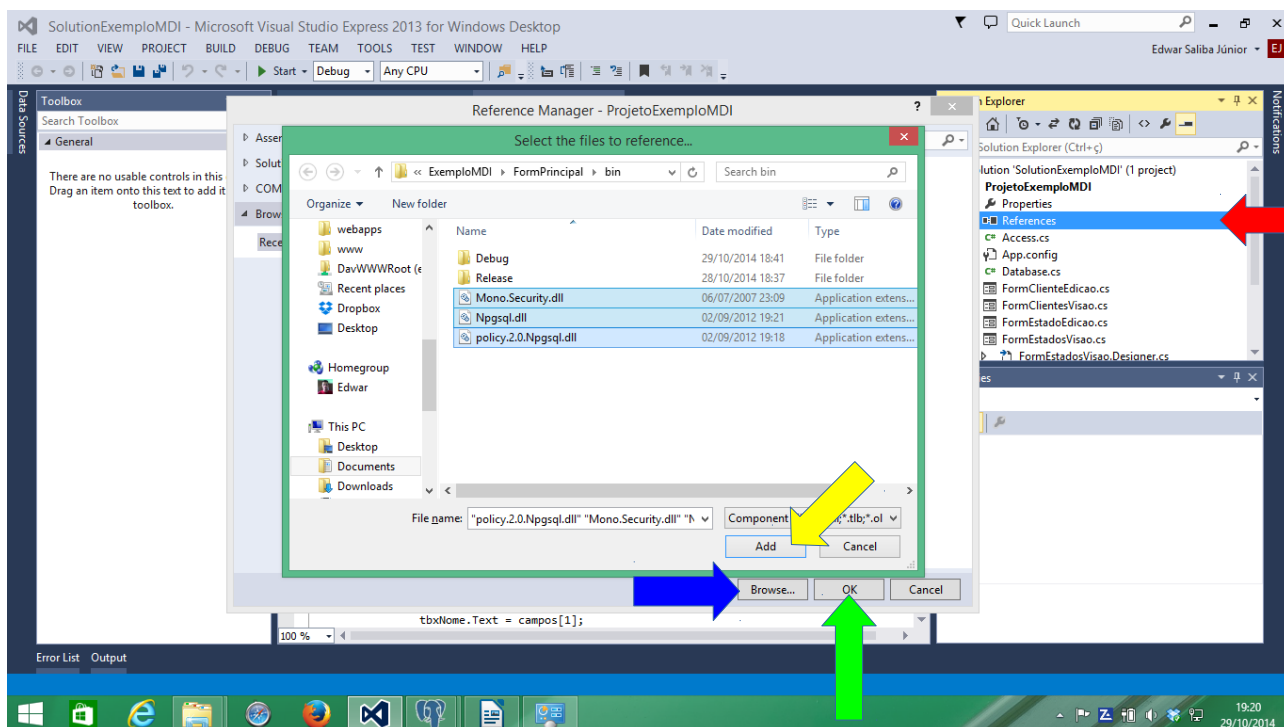


Figura 10: Adicionando as bibliotecas do Npgsql.

## Classe Access

Esta classe criará a *String de Conexão* para facilitar nossa vida. Então com o botão direito do *mouse* clique sobre o nome do projeto (11), escolha a opção *Add* e em seguida *Class...*

Na tela que aparecer escolha *Class* (seta verde na 12), dê o nome *Access.cs* (seta vermelha na 12) e aperte o botão *Add* (seta amarela na 12) para que o arquivo seja criado.

Assim que criado, veja o código-fonte mostrado nas Figuras 13 e 14. Para diminuir o risco de erros na produção do *software*, esta classe será fornecida pelo professor.

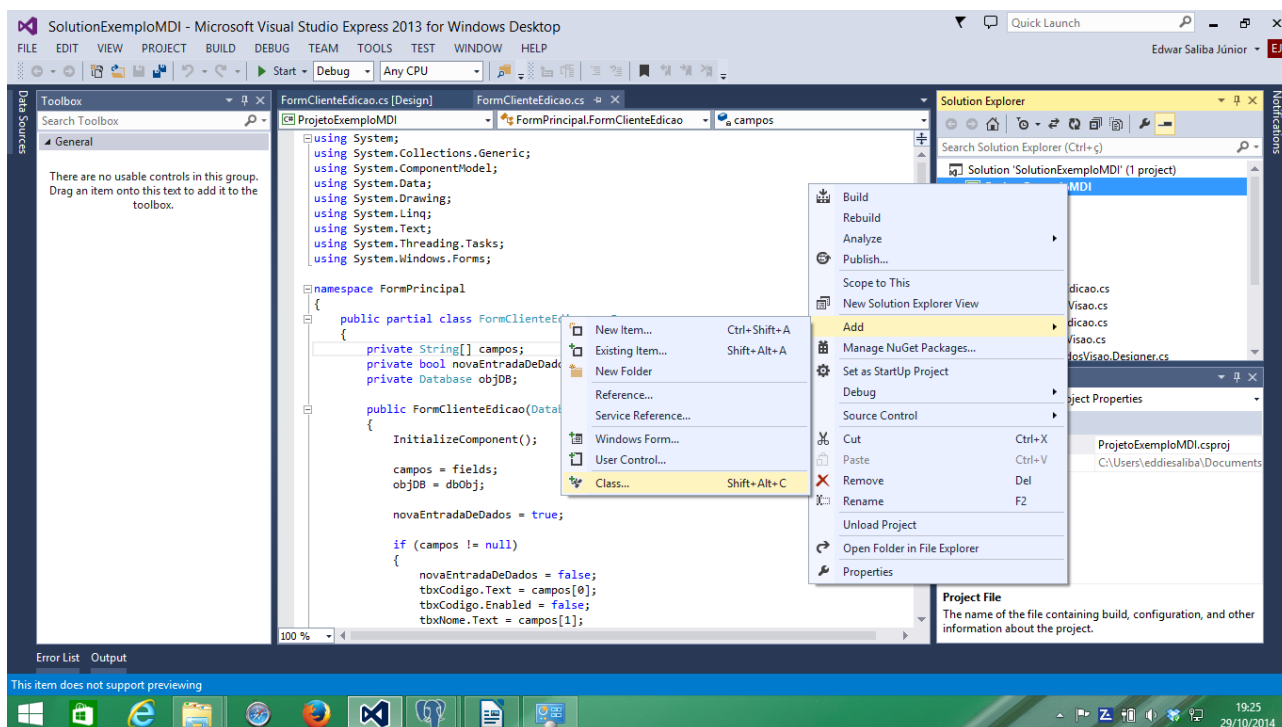


Figura 11: Caminha para criar uma nova classe no projeto.

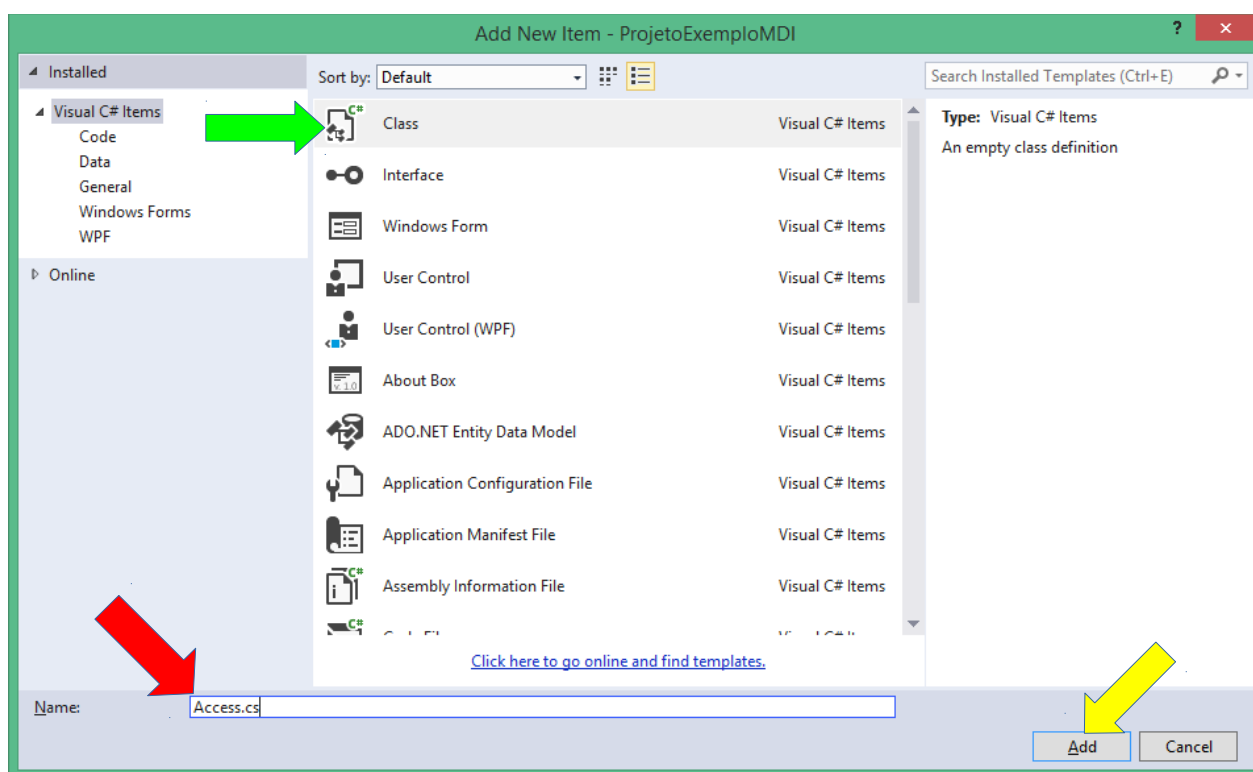


Figura 12: Tela de adição de novo item.

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace ExemploMDI
8 {
9     class Access
10    {
11        private String nomeSGBD;
12        private String host;
13        private String porta;
14        private String nomeBD;
15        private String usuario;
16        private String senha;
17        private String nomeConexaoODBC = null;
18
19        // Construtor para PostgreSQL via ODBC.
20        public Access(String nomeConexaoODBC, String nomeBD, String senha, String host)
21        {
22            this.nomeSGBD = "postgresql";
23            if ((host == null) || (host.Trim().Equals(""))){
24                this.host = "localhost";
25            }
26            else{
27                this.host = host;
28            }
29            this.porta = "5432";
30            this.nomeBD = nomeBD;
31            this.usuario = "postgres";
32            this.senha = senha;
33            this.nomeConexaoODBC = nomeConexaoODBC;
34        }
35
36        // Construtor para PostgreSQL via Npgsql.
37        public Access(String host, String porta, String usuario, String senha, String nomeBD)
38        {
39            this.nomeSGBD = "postgresql";
40            this.host = host;
41            this.porta = porta;
42            this.nomeBD = nomeBD;
43            this.usuario = usuario;
44            this.senha = senha;
45        }
46
47        public String getJDBCStringDeConexao()
48        {
49            return "jdbc:" + nomeSGBD + "://" + host + ":" + porta + "/" + nomeBD;
50        }
51
52        public String getNpgsqlStringDeConexao()
53        {
54            return "Server=" + host + ";Port=" + porta + ";User Id=" + usuario +
55                ";Password=" + senha + ";Database=" + nomeBD + ";Preload Reader = true;";
56        }
57
58        public String getODBCStringDeConexao()
59        {
60            return "DSN=" + nomeConexaoODBC + ";UID=" + usuario + ";PWD=" + senha + ";";
61        }
62    }
```

Figura 13: Classe Access - Parte 01/02.

```
63 public String getNomeBD()
64 {
65     return nomeBD;
66 }
67
68 public String getUsuario()
69 {
70     return usuario;
71 }
72
73 public String getSenha()
74 {
75     return senha;
76 }
77
78 public String getNomeConexaoODBC()
79 {
80     return nomeConexaoODBC;
81 }
82 }
83 }
```

Figura 14: Classe Access - Parte 02/02.

Feito isto, vamos agora para a classe que contém funções auxiliares para que nosso trabalho de manipulação de dados seja menos árduo.

## Classe Database

Esta classe contém funções e procedimentos aplicados ao SGBD para nos poupar trabalho.

Então dê uma olhada e uma boa estudada nesta classe (Figuras 15, 16, 17 e 18), pois nós a usaremos no *software* inteiro.

Para diminuir o risco de erros na produção do *software*, esta classe também será fornecida pelo professor. No entanto, o código-fonte será mostrado nas páginas a seguir.

```
1 using System;
2 using System.Collections;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6 using System.Data;
7 using System.Data.Odbc;
8 using Npgsql;
9
10 namespace ExemploMDI
11 {
12     public class Database
13     {
14         private Access informationDB;
15         private NpgsqlConnection conn;
16         private bool enableMessages;
17
18         public Database(){
19             String porta = "5432",
20                 dataBaseName = "ExemploMDI",
21                 password = "123456",
22                 host = "localhost",
23                 usuario = "postgres";
24
25             this.informationDB = new Access(host, porta, usuario, password, dataBaseName);
26             this.connectionToDB();
27
28             this.enableMessages = true;
29         }
30
31         public void connectionToDB() {
32             try {
33                 conn = new NpgsqlConnection(informationDB.getNpgsqlStringDeConexao());
34                 conn.Open();
35
36                 if (this.enableMessages)
37                     System.Console.WriteLine("Conexão Npgsql completada com sucesso.");
38             } catch (OdbcException e) {
39                 System.Console.WriteLine("Exceção ocorrida na conexão. Mensagem: " + e.Message +
40                                         "\n\nStackTrace: " + e.StackTrace);
41             }
42         }
43
44         public void insertValues(String tableName, String[] fieldsNames, String[] fieldsValues){
45             NpgsqlCommand queryDB;
46
47             String query = "INSERT INTO \"" + tableName + "\" (" +
48                 returnFieldsNames(fieldsNames) + ")";
49             query += " VALUES(" +
50                 returnValues(fieldsValues, true) + ")";
51
52             if (this.enableMessages)
53                 System.Console.WriteLine(query);
54
55             queryDB = new NpgsqlCommand(query, conn);
56             queryDB.ExecuteNonQuery();
57             queryDB.Dispose();
58             queryDB = null;
59         }
60     }
```

Figura 15: Classe Database - Parte 01/04.

```
61 public void deleteValues(String tableName, String condition){
62     NpgsqlCommand queryDB;
63
64     String query = "DELETE FROM \"" + tableName + "\"";
65
66     if (!condition.Equals("")) {
67         query += " WHERE " + condition;
68     }
69
70     if (this.enableMessages)
71         System.Console.WriteLine(query);
72
73     queryDB = new NpgsqlCommand(query, conn);
74     queryDB.ExecuteNonQuery();
75     queryDB.Dispose();
76     queryDB = null;
77 }
78
79 public void updateValues(String tableName, String[] fields, String[] values, String condition)
80     NpgsqlCommand queryDB;
81
82     String query = "UPDATE \"" + tableName + "\" SET ";
83     query += this.returnSetValues(fields, values);
84
85     if (!condition.Equals("")) {
86         query += " WHERE " + condition;
87     }
88
89     if (this.enableMessages)
90         System.Console.WriteLine(query);
91
92     queryDB = new NpgsqlCommand(query, conn);
93     queryDB.ExecuteNonQuery();
94     queryDB.Dispose();
95     queryDB = null;
96 }
97
98 public ArrayList selection(String table, bool putQuotationMarksOnTheFields,
99     String[] fields, String condition){
100     // Este é o data adapter o objeto que entende os banco de dados SQL.
101     NpgsqlDataAdapter da = new NpgsqlDataAdapter();
102
103     String query = "SELECT ";
104     if(putQuotationMarksOnTheFields)
105         query += returnFieldsNames(fields);
106     else
107         query += returnValues(fields, putQuotationMarksOnTheFields);
108
109     query += " FROM " + table;
110
111     if ((condition != null) && (!condition.Equals("")) {
112         query += " WHERE " + condition;
113     }
114
115     if (this.enableMessages)
116         System.Console.WriteLine(query);
117
118     da.SelectCommand = new NpgsqlCommand(query, conn);
119     DataSet ds = new DataSet();
120     da.Fill(ds);
```

Figura 16: Classe Database - Parte 02/04.

```

121
122     ArrayList resultsList = new ArrayList();
123
124     int j = 0,
125         qtdLinhas = ds.Tables[0].Rows.Count,
126         qtdColunas = ds.Tables[0].Columns.Count;
127
128     while (j < qtdLinhas) {
129         String[] row = new String[qtdColunas];
130         for (int i = 0; i < qtdColunas; i++)
131         {
132             row[i] = ds.Tables[0].Rows[j][i].ToString();
133         }
134         resultsList.Add(row);
135     }
136     da.Dispose();
137     da = null;
138
139     return resultsList;
140 }
141
142 public DataSet selection(String table, String[] fields, bool putQuotationMarksOnTheFields,
143     String condition) {
144     String query = "SELECT ";
145     // Este é o data adapter o objeto que entende os banco de dados SQL.
146     NpgsqlDataAdapter da = new NpgsqlDataAdapter();
147
148     if (putQuotationMarksOnTheFields)
149         query += returnFieldsNames(fields);
150     else
151         query += returnValues(fields, putQuotationMarksOnTheFields);
152
153     if (putQuotationMarksOnTheFields)
154         query += " FROM \"" + table + "\" ";
155     else
156         query += " FROM " + table + " ";
157
158     if (!condition.Equals("")) {
159         query += " WHERE " + condition;
160     }
161
162     if (this.enableMessages)
163         System.Console.WriteLine(query);
164
165     da.SelectCommand = new NpgsqlCommand(query, conn);
166     DataSet ds = new DataSet();
167     da.Fill(ds);
168     da.Dispose();
169     da = null;
170
171     return(ds);
172 }
173
174 public DataSet selection(String query){
175     // Este é o data adapter o objeto que entende os banco de dados SQL.
176     NpgsqlDataAdapter da = new NpgsqlDataAdapter();
177
178     da.SelectCommand = new NpgsqlCommand(query, conn);
179     DataSet ds = new DataSet();
180     da.Fill(ds);

```

Figura 17: Classe Database - Parte 03/04.

```
181         da.Dispose();
182         da = null;
183         return(ds);
184     }
185
186     public String returnValues(String[] values, bool putQuotationMarks) {
187         String vals = "";
188
189         if(putQuotationMarks){
190             for (int i = 0; i < values.Length - 1; i++) {
191                 vals += "\"" + values[i] + "\", ";
192             }
193
194             vals += "\"" + values[values.Length - 1] + "\"";
195         }
196         else{
197             for (int i = 0; i < values.Length - 1; i++) {
198                 vals += values[i] + ", ";
199             }
200
201             vals += values[values.Length - 1];
202         }
203
204         return vals;
205     }
206
207     public String returnFieldsNames(String[] values) {
208         String vals = "\"";
209
210         for (int i = 0; i < values.Length - 1; i++) {
211             vals += values[i] + "\", \";
212         }
213
214         vals += values[values.Length - 1] + "\"";
215
216         return vals;
217     }
218
219     public String returnSetValues(String[] fields, String[] values)
220     {
221         String vals = "";
222
223         for (int i = 0; i < values.Length - 1; i++) {
224             vals += "\"" + fields[i] + "\" = " +
225                 (values[i].Equals("") ? "\\''" : "\"" + values[i] + "\"") + ", ";
226         }
227
228         vals += "\"" + fields[fields.Length - 1] + "\" = " +
229             (values[values.Length - 1].Equals("") ? "\\''" : "\"" +
230             values[values.Length - 1] + "\"");
231
232         return vals;
233     }
234
235     public void closeDBConnection(){
236         this.conn.Close();
237     }
238 }
239
240 }
```

Figura 18: Classe Database - Parte 04/04.

## Criando os Eventos nos Formulários

### FormClienteEdicao

Primeiramente, dê um clique duplo no botão *Salvar* do formulário. Será automaticamente criado o evento *btnSalvar\_Click*. Volte ao formulário e dê um clique duplo no botão *Fechar* para que o evento *btnFechar\_Click* também seja criado.

Agora acompanhe o código da classe que será postado a seguir (Figuras 19, 20 e 21) e faça as modificações necessárias na classe *Database* que você criou no seu *software* para que ambas fiquem semelhantes. Ou seja, crie os atributos necessários, reescreva o método construtor, pois, ele terá que ser modificado para que atenda as necessidades do *software*, escreva o código-fonte dos métodos *btnSalvar\_Click* e *btnFechar\_Click* e ainda, escreva os demais métodos que foram criados para preenchimento de componentes e manipulação de dados.

```
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
11 namespace ExemploMDI
12 {
13     public partial class FormClienteEdicao : Form
14     {
15         private String[] campos;
16         private bool novaEntradaDeDados;
17         private Database objDB;
18
19         public FormClienteEdicao(Database dbObj, String[] fields)
20         {
21             InitializeComponent();
22
23             campos = fields;
24             objDB = dbObj;
25
26             novaEntradaDeDados = true;
27
28             /* Se o formulário for aberto para alteração, então os campos
29              * aparecerão preenchidos e o campo "Código" aparecerá desabilitado.
30              * Caso contrário, se o formulário for aberto para cadastro de um
31              * novo estado, então todos os campos aparecerão vazios.
32              */
33             if (campos != null)
```

Figura 19: Classe *FormClienteEdicao* – Parte 01/03.

```
34         {
35             novaEntradaDeDados = false;
36             tbxCodigo.Text = campos[0];
37             tbxCodigo.Enabled = false;
38             tbxNome.Text = campos[1];
39             tbxLogradouro.Text = campos[2];
40             tbxNumero.Text = campos[3];
41             tbxBairro.Text = campos[4];
42             tbxMunicipio.Text = campos[5];
43             tbxCEP.Text = campos[6];
44             cbxEstado.FindString(campos[7]);
45         }
46
47         // Coloca o cursor no primeiro componente habilitado do formulário.
48         if (tbxCodigo.Enabled)
49         {
50             tbxCodigo.Focus();
51         }
52         else
53         {
54             tbxNome.Focus();
55         }
56
57         preencheComboBoxEstados();
58     }
59
60     private void btnSalvar_Click(object sender, EventArgs e)
61     {
62         try
63         {
64             if (novaEntradaDeDados)
65             {
66                 gravaNovoRegistro();
67             }
68             else
69             {
70                 atualizaRegistro();
71             }
72
73             MessageBox.Show("Dados salvos com sucesso!");
74             btnFechar_Click(sender, e);
75         }
76         catch (Exception msn)
77         {
78             MessageBox.Show(msn.ToString());
79         }
80     }
81
82     private void btnFechar_Click(object sender, EventArgs e)
83     {
84         FormClienteEdicao.ActiveForm.Close();
85     }
86
87     private void gravaNovoRegistro()
88     {
89         String[] fields = { "cli_cod", "cli_nom", "cli_lgd", "cli_num",
90                             "cli_bai", "cli_mun", "cli_cep", "est_cod" },
91             values = { tbxCodigo.Text.Trim(),
92                      tbxNome.Text.Trim(),
93                      tbxLogradouro.Text.Trim(),
```

Figura 20: Classe FormClienteEdicao – Parte 02/03.

```
94         tbxNumero.Text.Trim(),
95         tbxBairro.Text.Trim(),
96         tbxMunicipio.Text.Trim(),
97         tbxCEP.Text.Trim(),
98         cbxEstado.SelectedValue.ToString() };
99
100     objDB.insertValues("Cliente", fields, values);
101 }
102
103 private void atualizaRegistro()
104 {
105     String[] fields = { "cli_nom", "cli_lgd", "cli_num",
106                       "cli_bai", "cli_mun", "cli_cep", "est_cod" },
107     values = { tbxNome.Text.Trim(),
108              tbxLogradouro.Text.Trim(),
109              tbxNumero.Text.Trim(),
110              tbxBairro.Text.Trim(),
111              tbxMunicipio.Text.Trim(),
112              tbxCEP.Text.Trim(),
113              cbxEstado.SelectedValue.ToString() };
114
115     objDB.updateValues("Cliente", fields, values, "cli_cod = " + tbxCodigo.Text.Trim());
116 }
117
118 private void preencheComboBoxEstados()
119 {
120     DataSet ds = objDB.selection(" SELECT est_cod, est_nom " +
121                                " FROM \"Estado\" " +
122                                " ORDER BY est_nom");
123
124     cbxEstado.DataSource = ds.Tables[0];
125     cbxEstado.DisplayMember = "est_nom";
126     cbxEstado.ValueMember = "est_cod";
127 }
128 }
129 }
```

Figura 21: Classe FormClienteEdicao - Parte 03/03.

## FormEstadoEdicao

Neste formulário dê um clique duplo no botão *Salvar* e em seguida no botão *Fechar*. Desta forma serão criados os eventos: *btnSalvar\_Click* e *btnFechar\_Click*.

Agora faça as modificações necessárias no código-fonte comparando-o com o apresentado a seguir (Figuras 22 e 23).

```
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
11 namespace ExemploMDI
12 {
13     public partial class FormEstadoEdicao : Form
14     {
15         private String[] campos;
16         private bool novaEntradaDeDados;
17         private Database objDB;
18
19         public FormEstadoEdicao(Database dbObj, String[] fields)
20         {
21             InitializeComponent();
22
23             campos = fields;
24             objDB = dbObj;
25
26             novaEntradaDeDados = true;
27
28             /* Se o formulário for aberto para alteração, então os campos
29              * aparecerão preenchidos e o campo "Código" aparecerá desabilitado.
30              * Caso contrário, se o formulário for aberto para cadastro de um
31              * novo estado, então todos os campos aparecerão vazios.
32              */
33             if (campos != null)
34             {
35                 novaEntradaDeDados = false;
36                 tbxCodigo.Text = campos[0];
37                 tbxCodigo.Enabled = false;
38                 tbxNome.Text = campos[1];
39                 tbxSigla.Text = campos[2];
40             }
41
42             // Coloca o cursor no primeiro componente habilitado do formulário.
43             if (tbxCodigo.Enabled)
44             {
45                 tbxCodigo.Focus();
46             }
47             else
48             {
49                 tbxNome.Focus();
50             }
51         }
52
53         private void btnSalvar_Click(object sender, EventArgs e)
54         {
55             try
56             {
57                 if (novaEntradaDeDados)
58                 {
59                     gravaNovoRegistro();
60                 }
61             }
62             catch { }
63         }
64     }
65 }
```

Figura 22: FormEstadoEdicao - Parte 01/02.

```
61         else
62         {
63             atualizaRegistro();
64         }
65
66         MessageBox.Show("Dados salvos com sucesso!");
67         btnFechar_Click(sender, e);
68     }
69     catch (Exception msn)
70     {
71         MessageBox.Show(msn.ToString());
72     }
73 }
74
75 private void btnFechar_Click(object sender, EventArgs e)
76 {
77     FormEstadoEdicao.ActiveForm.Close();
78 }
79
80 private void gravaNovoRegistro()
81 {
82     String[] fields = { "est_cod", "est_nom", "est_sgl" },
83     values = { tbxCodigo.Text.Trim(),
84               tbxNome.Text.Trim(),
85               tbxSigla.Text.Trim() };
86
87     objDB.insertValues("Estado", fields, values);
88 }
89
90 private void atualizaRegistro()
91 {
92     String[] fields = { "est_nom", "est_sgl" },
93     values = { tbxNome.Text.Trim(),
94               tbxSigla.Text.Trim() };
95
96     objDB.updateValues("Estado", fields, values, "est_cod = " + tbxCodigo.Text.Trim());
97 }
98 }
99 }
```

Figura 23: FormEstadoEdicao - Parte 02/02.

## FormClientesVisao

Para este formulário faça o seguinte:

- dê um clique simples numa área vazia do mesmo. Após o clique, vá até as propriedades do formulário e mude o nome do formulário para *frmClientes*, da mesma forma que fizemos com os outros objetos (Veja que podemos modificar também, se quisermos, os nomes dos formulários!). Clique na aba eventos e procure o evento *Load*. Ao achá-lo dê um clique duplo na área logo a frente do nome do evento. Então o evento *frmClientes\_Load* será criado no código-fonte;
- agora dê um clique duplo no botão *Novo*, *Alterar*, *Excluir* e *Atualizar*. Serão criados no código-fonte, respectivamente, os seguintes eventos: *btnNovo\_Click*, *btnAlterar\_Click*, *btnExcluir\_Click* e *btnAtualizar\_Click*. E por último,
- dê um clique simples sobre o *DataGridView* e vá até as propriedades deste

componente. Acesse a aba eventos e procure o evento *SelectionChanged*. Ao encontrá-lo dê um clique duplo na área vazia a frente do nome do evento e então o evento *dgvClientes\_SelectionChanged* será criado no código-fonte.

Feito isto, compare o código-fonte do seu formulário com o apresentado nas Figuras 24, 25 e 26 e faça as devidas modificações para que o seu código funcione harmonicamente com o restante do sistema. Você terá que, além de modificar o método construtor da classe e criar alguns atributos, criar também alguns outros métodos que o auxiliarão na manipulação dos dados no formulário. Veja o código-fonte a seguir:

```
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.Data;
5 using System.Data.Common;
6 using System.Data.Odbc;
7 using System.Data.OleDb;
8 using System.Drawing;
9 using System.Linq;
10 using System.Text;
11 using System.Threading.Tasks;
12 using System.Windows.Forms;
13
14 namespace ExemploMDI
15 {
16     public partial class FormClientesVisao : Form
17     {
18         private String clienteSelecionado;
19         private Database objDB;
20         private FormClienteEdicao dlgClienteEdicao;
21
22         public FormClientesVisao(Database dbObj)
23         {
24             InitializeComponent();
25             objDB = dbObj;
26             clienteSelecionado = null;
27         }
28
29         private void frmClientes_Load(object sender, EventArgs e)
30         {
31             preencheVisaoClientes();
32         }
33
34         private void btnNovo_Click(object sender, EventArgs e)
35         {
36             dlgClienteEdicao = new FormClienteEdicao(objDB, null);
37             dlgClienteEdicao.ShowDialog();
38         }
39
40         private void btnAlterar_Click(object sender, EventArgs e)
41         {
42             if (dgvClientes.CurrentRow.Index >= 0)
43             {
44                 String[] valores = obterLinhaDoComponenteDataGrid();
45             }
46         }
47     }
48 }
```

Figura 24: *FormClientesVisao - Parte 01/03.*

```

46         dlgClienteEdicao = new FormClienteEdicao(objDB, valores);
47         dlgClienteEdicao.ShowDialog();
48     }
49     else
50         MessageBox.Show("Antes de tentar editar, selecione uma linha na tabela.");
51 }
52
53 private void btnExcluir_Click(object sender, EventArgs e)
54 {
55     if (dgvClientes.CurrentRow.Index >= 0)
56     {
57         if (MessageBox.Show("Confirma exclusão?",
58             "Confirmação de Exclusão", MessageBoxButtons.YesNo) == DialogResult.Yes)
59         {
60             try
61             {
62                 clienteSelecioneado = dgvClientes.CurrentRow.Cells[0].Value.ToString();
63                 objDB.deleteValues("Cliente", "cli_cod = " + clienteSelecioneado);
64                 MessageBox.Show("Exclusão efetuada com sucesso!");
65                 preencheVisaoClientes();
66             }
67             catch (Exception m)
68             {
69                 MessageBox.Show(m.ToString());
70             }
71         }
72         else
73         {
74             MessageBox.Show("Exclusão não efetuada.");
75         }
76     }
77 }
78
79 private void btnAtualizar_Click(object sender, EventArgs e)
80 {
81     preencheVisaoClientes();
82 }
83
84 private void preencheVisaoClientes()
85 {
86     String[] fields = { "c.\"cli_cod\"",
87         "c.\"cli_nom\"",
88         "c.\"cli_lgd\"",
89         "c.\"cli_num\"",
90         "c.\"cli_bai\"",
91         "c.\"cli_mun\"",
92         "c.\"cli_cep\"",
93         "e.\"est_nom\" };
94
95     try
96     {
97         DataSet ds = new DataSet();
98         ds = objDB.selection("\\"Cliente\\" c, \\"Estado\\" e ",
99             fields,
100             false,
101             "c.est_cod = e.est_cod " +
102             "ORDER BY c.\"cli_nom\"");
103
104         dgvClientes.DataSource = ds.Tables[0];
105
106         // Renomeando o título das colunas do dataGridView.

```

Figura 25: FormClientesVisao - Parte 02/03.

```

106         dgvClientes.Columns["cli_cod"].HeaderText = "Código";
107         dgvClientes.Columns["cli_nom"].HeaderText = "Nome";
108         dgvClientes.Columns["cli_lgd"].HeaderText = "Logradouro";
109         dgvClientes.Columns["cli_num"].HeaderText = "Número";
110         dgvClientes.Columns["cli_bai"].HeaderText = "Bairro";
111         dgvClientes.Columns["cli_mun"].HeaderText = "Município";
112         dgvClientes.Columns["est_nom"].HeaderText = "Estado";
113         dgvClientes.Columns["cli_cep"].HeaderText = "C.E.P.";
114     }
115     catch (Exception msn)
116     {
117         MessageBox.Show(msn.ToString());
118         throw;
119     }
120 }
121
122 private void dgvClientes_SelectionChanged(object sender, EventArgs e)
123 {
124     if (dgvClientes.CurrentRow != null)
125     {
126         if (dgvClientes.CurrentRow.Index >= 0)
127         {
128             clienteSelecionado = dgvClientes.CurrentRow.Index.ToString();
129         }
130     }
131 }
132
133 private String[] obterLinhaDoComponenteDataGrid()
134 {
135     String[] campos = null;
136
137     if (dgvClientes.CurrentRow.Index >= 0)
138     {
139         campos = createEditableData(dgvClientes.CurrentRow.Cells[0].Value.ToString(),
140                                     dgvClientes.CurrentRow.Cells[1].Value.ToString(),
141                                     dgvClientes.CurrentRow.Cells[2].Value.ToString(),
142                                     dgvClientes.CurrentRow.Cells[3].Value.ToString(),
143                                     dgvClientes.CurrentRow.Cells[4].Value.ToString(),
144                                     dgvClientes.CurrentRow.Cells[5].Value.ToString(),
145                                     dgvClientes.CurrentRow.Cells[6].Value.ToString(),
146                                     dgvClientes.CurrentRow.Cells[7].Value.ToString());
147     }
148
149     return campos;
150 }
151
152 private String[] createEditableData(String codigo, String nome, String logradouro,
153                                     String numero, String bairro, String municipio, String cep, String estado) {
154     return (new String[] { codigo, nome, logradouro, numero, bairro, municipio, cep, estado })
155 }
156 }
157 }

```

Figura 26: FormClientesVisao - Parte 03/03.

## FormEstadosVisao

Para este formulário, da mesma forma que no formulário anterior, faça o seguinte:

- dê um clique simples numa área vazia do mesmo. Mude o nome do formulário para *frmEstados*. Vá até as propriedades do formulário e na aba eventos, procure o evento *Load*. Ao achá-lo, dê um clique duplo na área logo a frente do nome do evento. Então o evento *frmEstados\_Load* será criado no código-fonte;
- então dê um clique duplo no botão *Novo*, *Alterar*, *Excluir* e *Atualizar*. Serão criados, respectivamente, os seguintes eventos: *btnNovo\_Click*, *btnAlterar\_Click*, *btnExcluir\_Click* e *btnCarregarEstados\_Click*. E para finalizar a criação de eventos,
- dê um clique simples sobre o *DataGridView* e vá até as propriedades deste componente. Acesse a aba eventos e procure o evento *SelectionChanged*. Ao encontrá-lo dê um clique duplo na área vazia a frente do nome do evento e então o evento *dgvEstados\_SelectionChanged* será criado no código-fonte.

Feito isto, compare o código-fonte do seu formulário com o apresentado nas Figuras 27, 28 e 29 e faça as devidas modificações para que o seu código funcione perfeitamente com o restante do sistema. Você terá que, além de modificar o método construtor da classe e criar alguns atributos, criar também alguns outros métodos que facilitarão a manipulação dos dados no formulário. Código-fonte a seguir:

```
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
11 namespace ExemploMDI
12 {
13     public partial class FormEstadosVisao : Form
14     {
15         private Database objDB;
16         private FormEstadoEdicao dlgEstadoEdicao;
17         private String estadoSelecioneado;
18
19         public FormEstadosVisao(Database db)
20         {
21             InitializeComponent();
22             objDB = db;
23             estadoSelecioneado = null;
24         }
25     }
```

Figura 27: FormEstadosVisao - Parte 01/03.

```
26 private void frmEstados_Load(object sender, EventArgs e)
27 {
28     preencheVisaoEstados();
29 }
30
31 private void btnCarregarEstados_Click(object sender, EventArgs e)
32 {
33     frmEstados_Load(sender, e);
34 }
35
36 private void btnNovo_Click(object sender, EventArgs e)
37 {
38     dlgEstadoEdicao = new FormEstadoEdicao(objDB,null);
39     dlgEstadoEdicao.ShowDialog();
40 }
41
42 private void btnAlterar_Click(object sender, EventArgs e)
43 {
44     if (dgvEstados.CurrentRow.Index >= 0)
45     {
46         String[] valores = obterLinhaDoComponenteDataGrid();
47
48         dlgEstadoEdicao = new FormEstadoEdicao(objDB, valores);
49         dlgEstadoEdicao.ShowDialog();
50     }
51     else
52         MessageBox.Show("Antes de tentar editar, selecione uma linha na tabela.");
53 }
54
55 private void btnExcluir_Click(object sender, EventArgs e)
56 {
57     if (dgvEstados.CurrentRow.Index >= 0)
58     {
59         if (MessageBox.Show("Confirma exclusão?",
60             "Confirmação de Exclusão", MessageBoxButtons.YesNo) == DialogResult.Yes)
61         {
62             try
63             {
64                 estadoSelecioneado = dgvEstados.CurrentRow.Cells[0].Value.ToString();
65                 objDB.deleteValues("Estado", "est_cod = " + estadoSelecioneado);
66                 MessageBox.Show("Exclusão efetuada com sucesso.");
67                 preencheVisaoEstados();
68             }
69             catch (Exception m)
70             {
71                 MessageBox.Show(m.ToString());
72             }
73         }
74         else
75         {
76             MessageBox.Show("Exclusão não efetuada.");
77         }
78     }
79 }
80
81 private void preencheVisaoEstados(){
82     String[] fields = { "e.\"est_cod\"",
83         "e.\"est_nom\"",
84         "e.\"est_sgl\""};
85     try{
```

Figura 28: FormEstadosVisao - Parte 02/03.

```
86         DataSet ds = new DataSet();
87         ds = objDB.selection("\Estado\" e",
88                             fields,
89                             false,
90                             "1 = 1 " +
91                             "ORDER BY e.\\"est_nom\\"");
92
93         dgvEstados.DataSource = ds.Tables[0];
94
95         // Renomeando o título das colunas do dataGridView.
96         dgvEstados.Columns["est_cod"].HeaderText = "Código";
97         dgvEstados.Columns["est_nom"].HeaderText = "Nome";
98         dgvEstados.Columns["est_sgl"].HeaderText = "Sigla";
99     }
100     catch(Exception msn){
101         MessageBox.Show(msn.ToString());
102         throw;
103     }
104 }
105
106 private void dgvEstados_SelectionChanged(object sender, EventArgs e)
107 {
108     if (dgvEstados.CurrentRow != null)
109     {
110         if (dgvEstados.CurrentRow.Index >= 0)
111         {
112             estadoSelecionado = dgvEstados.CurrentRow.Index.ToString();
113         }
114     }
115 }
116
117 private String[] obterLinhaDoComponenteDataGrid()
118 {
119     String[] campos = null;
120
121     if (dgvEstados.CurrentRow.Index >= 0)
122     {
123         campos = createEditableData(dgvEstados.CurrentRow.Cells[0].Value.ToString(),
124                                     dgvEstados.CurrentRow.Cells[1].Value.ToString(),
125                                     dgvEstados.CurrentRow.Cells[2].Value.ToString());
126     }
127
128     return campos;
129 }
130
131 private String[] createEditableData(String codigo, String nome, String sigla)
132 {
133     return new String[] { codigo, nome, sigla };
134 }
135 }
136 }
```

Figura 29: FormEstadosVisao - Parte 03/03.

## FormPrincipal

No *FormPrincipal*, você deverá fazer o seguinte:

- dê um clique duplo na opção de *menu* “Arquivo | Sair”. Será criado então o evento *sairToolStripMenuItem\_Click*;
- repita esta mesma operação para os demais itens de *menu* existentes. Ou seja “Cadastro | Clientes”, “Cadastro | Estados” e “Ajuda | Sobre”.

Feito isto, os respectivos eventos serão criados. E agora, para finalizar seu *software*, compare sua classe *FormPrincipal* com a apresentada a seguir (Figuras 30 e 31). Reescreva as diferenças para que a sua fique semelhante a apresentada neste tutorial.

Terminada esta etapa, teste seu *software* para averiguar se está tudo funcionando perfeitamente. E assim se encerra mais uma etapa deste curso.

```
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
11 namespace ExemploMDI
12 {
13     public partial class FormPrincipal : Form
14     {
15         private FormClientesVisao fc;
16         private FormEstadosVisao fe;
17         private FormSobre fs;
18         private Database dbObj;
19
20         public FormPrincipal()
21         {
22             InitializeComponent();
23             dbObj = new Database();
24         }
25
26         private void sairToolStripMenuItem_Click(object sender, EventArgs e)
27         {
28             // Encerra aplicativo.
29             FormPrincipal.ActiveForm.Close();
30         }
31
32         private void clientesToolStripMenuItem_Click(object sender, EventArgs e)
33         {
```

Figura 30: *FormPrincipal* - Parte 01/02.

```
34         if (fc == null || fc.IsDisposed)
35         {
36             fc = new FormClientesVisao(dbObj);
37             fc.MdiParent = this;
38             fc.Show();
39         }
40         else
41         {
42             fc.BringToFront();
43         }
44     }
45
46     private void estadosToolStripMenuItem_Click(object sender, EventArgs e)
47     {
48         if (fe == null || fe.IsDisposed)
49         {
50             fe = new FormEstadosVisao(dbObj);
51             fe.MdiParent = this;
52             fe.Show();
53         }
54         else
55         {
56             fe.BringToFront();
57         }
58     }
59
60     private void sobreToolStripMenuItem1_Click(object sender, EventArgs e)
61     {
62         if (fs == null || fs.IsDisposed)
63         {
64             fs = new FormSobre();
65             fs.MdiParent = this;
66             fs.Show();
67         }
68         else
69         {
70             fs.BringToFront();
71         }
72     }
73 }
74
```

Figura 31: FormPrincipal - Parte 02/02.