# Introdução

MVC é um padrão de arquitetura de software, separando suas aplicações em três camadas. Iremos fazer um simples CRUD para demonstrar sua aplicação. As três camadas significam:

VIEW – Visão: A camada de interação com o usuário

MODEL – Modelo: A camada de manipulação dos dados

CONTROLLER – Controle: A camada que controla

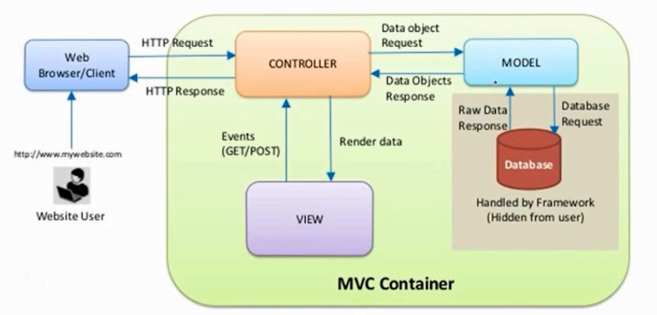


Figura1. Exemplo de aplicação web para MVC

# Ferramentas necessárias

## SQL SERVER

SQL Server (qualquer versão a partir da 2005)

* Sugestão SQL Server 2017 Express

Não precisa colocar senha. Pode utilizar a autenticação do Windows

Obs. Caso mostre **error de instancia**. Basta acessar o Sql Server Configuration Manager e reiniciar o **SQL Server (SQLEXPRESS)**

## Visual Studio

Visual Studio a partir do 2012

Colocar no google Baixar visual studio 2017 Comunidade “Comunit”

## Componentes do Visual Studio necessários

* Desenvolvimento para desktop com .NET
* Desenvolvimento com a Plataforma Universal do Windows
* ASP.NET e desenvolvimento Web
* Desenvolvimento desktop com C++
* Processamento e armazenamento de dados
* Desenvolvimento de extensão do Visual Studio
* Desenvolvimento multiplataforma com .NET Core

# O que é o modelo MVC

É uma forma de desenvolvimento reutilizável. Também conhecido como Modelo de quatro camadas ou Modelo estrutural.

View – São vistas, os formulários. A Tela de Login é um exemplo claro de View

## Criando a camada View

No visual studio vamos criar o projeto juntamente com a primeira camada a View. Clique em Arquivo > Novo > Projeto > Visual C# > Aplicativo do Windows Forms (.NET Framework)

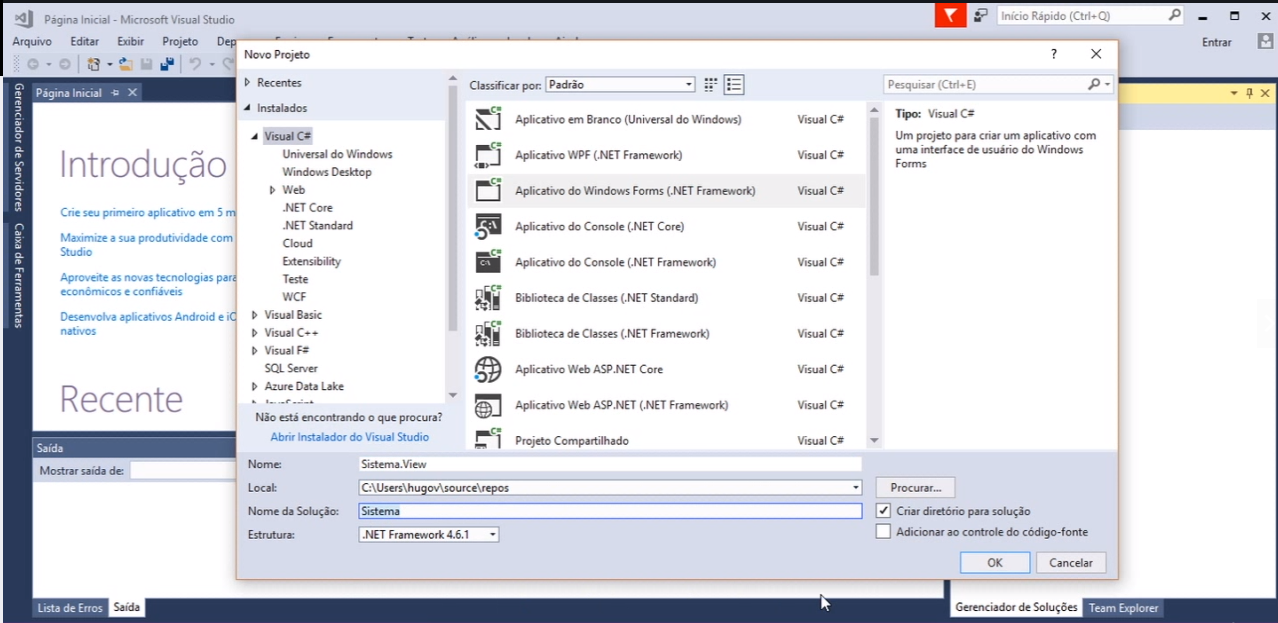


Figura 2. Criando um projeto e a primeira camada View

## Criando a camada Model

Botão direito sobre a Solução > Adicionar > Novo Projeto

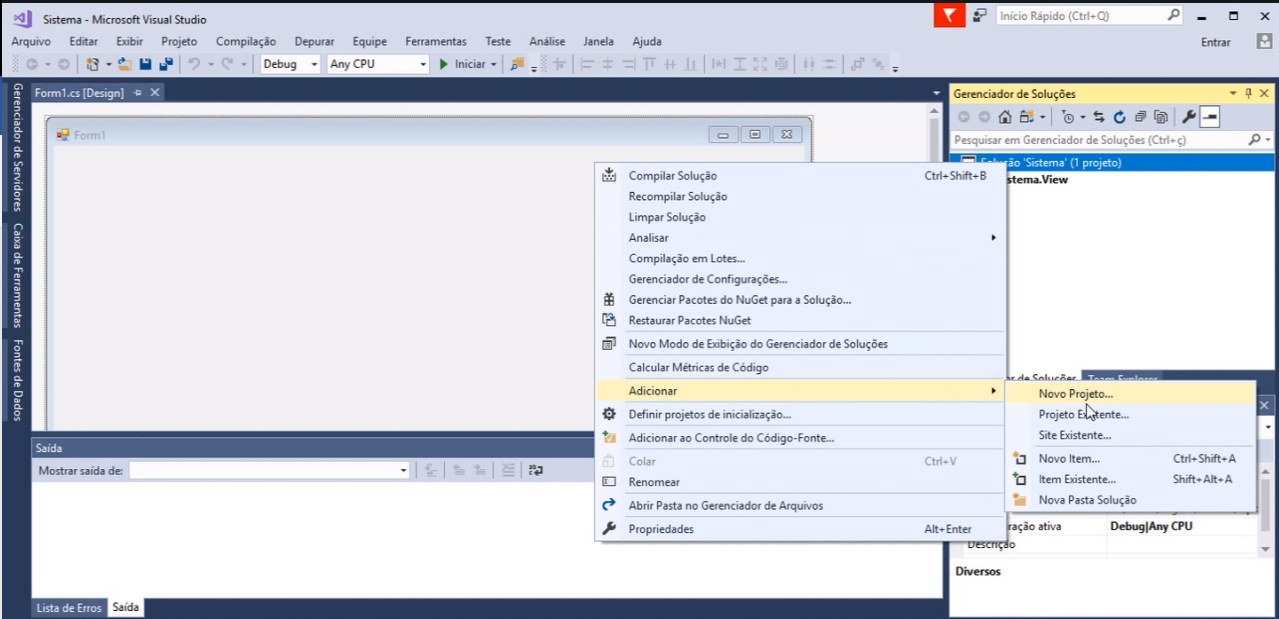


Figura 3­. Criando uma nova camada

A camada Model precisa ser um conjunto de bibliotecas. Portanto selecione **Biblioteca de Classes (.NET Framework)**

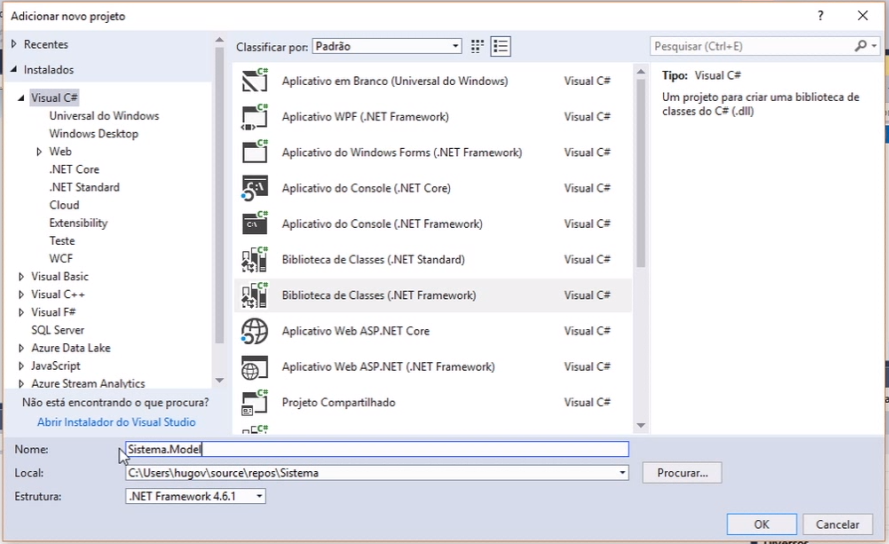


Figura 3. Criando a camada Model

## Criação da camada Control

A camada Control ou DAO, também precisa ser um conjunto de bibliotecas. Portanto selecione **Biblioteca de Classes (.NET Framework)**

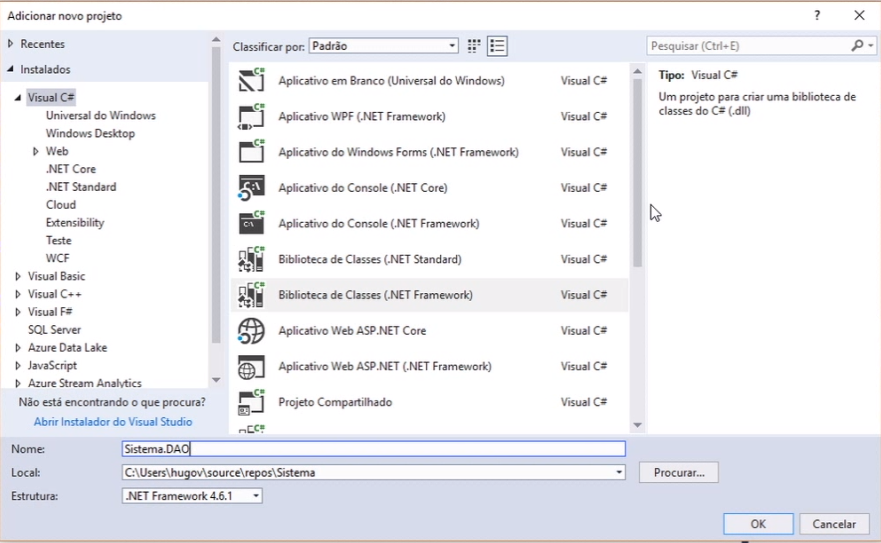


Figura 4. Criando a camada DAO Control

Criando a camada Classes ou Entidades

A camada Entidades ou Classes, também precisa ser um conjunto de bibliotecas. Portanto selecione **Biblioteca de Classes (.NET Framework)**

Toda entidade, toda tabela e seus atributos serão representados por suas respectivas classes que estarão dentro da camada Entidades, para que as outras camadas possam se comunicar com o Banco através dessas classes.

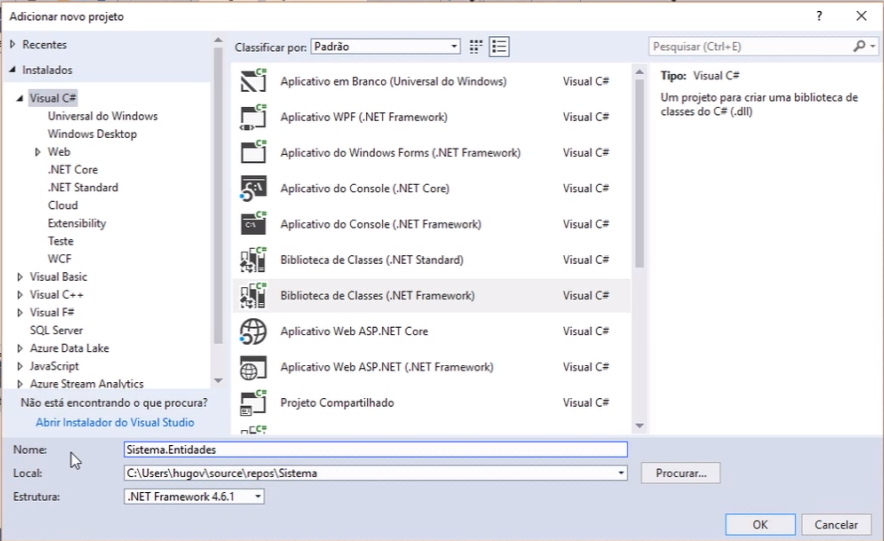


Figura 5. Criando a camada Entidades ou Classes

## Editando as Ferramentas do seu Visual Studio

Para acrescentar ou remover opções de ferramentas siga os passos. Exibir > Barra de Ferramentas

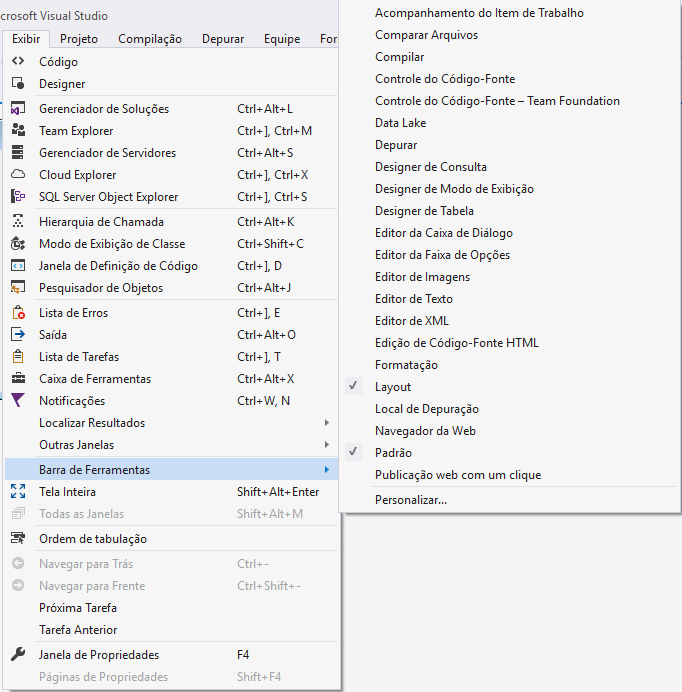


Figura 6. Opção do menu Exibir Barra de Ferramentas

## Restaurando o formato padrão do Visual Studio

Caso você tenha perdido alguma ferramenta e não está encontrando. Siga os passos para restaurar sua tela. Janela > Redefinir Layout da Janela

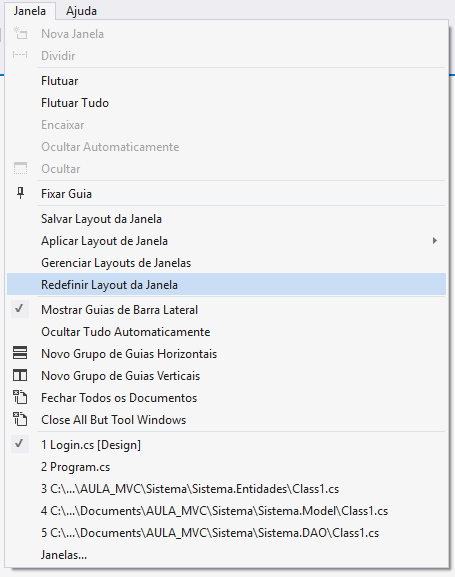


Figura 7. Redefinir Layout da Janela

## Criando a tela de cadastro de usuário

A tela de cadastro assim como o Login estarão na camada de View. Essas telas precisam ser inicializadas assim que o projeto inicia. Portanto caso não esteja em NEGRITO, precisamos clicar com o botão direito na camada e selecionar a opção **Inicializar Interativo com o Projeto**.

Renomear o Form1 por Login



Figura 8. Renomeando o formulário Form1.cs

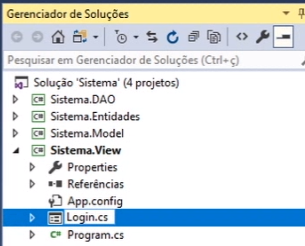


Figura 9. Renomeando o Form1.cs para Login.cs

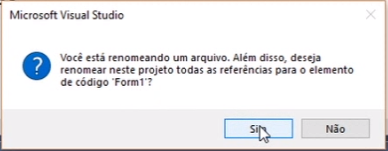


Figura 8. Autorizando a renomeação do formulário

Observe que ao alterar, todas as referências também serão alteradas.

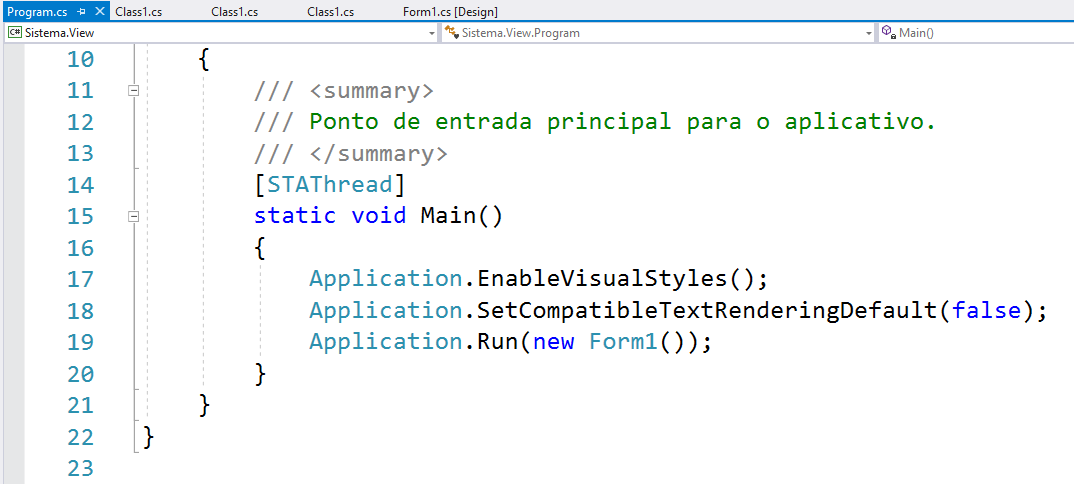


Figura 9. New Form1()

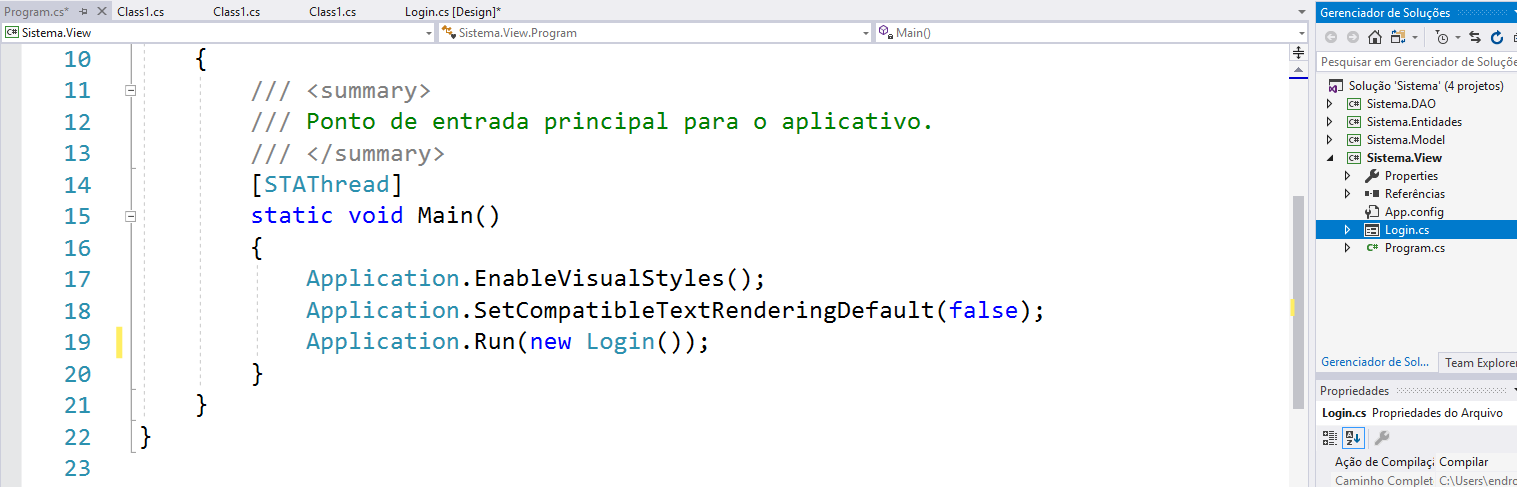


Figura 10. New Login()

Criando a tela de login

Duplo clique em Login.cs e altere o Text de Form1 para Login

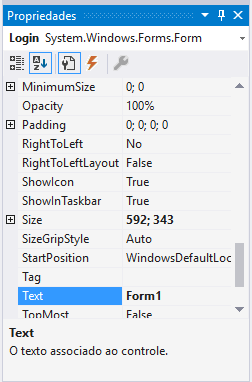


Figura 11. Text Form1

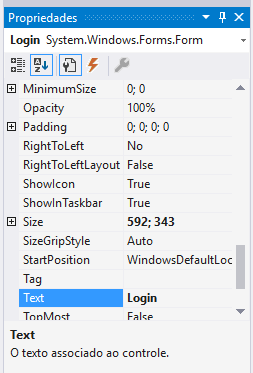


Figura 12. Text agora Login

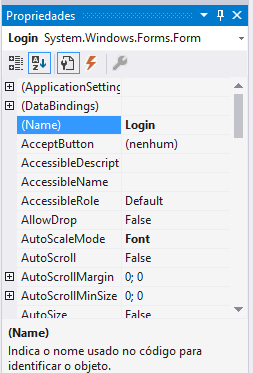


Figura 13. Name Login

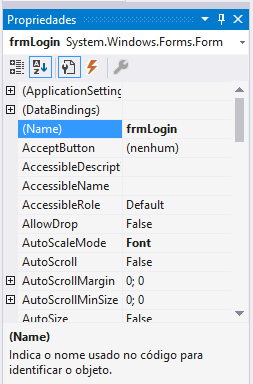


Figura 14. Name renomeado para frmLogin

Inserindo textos

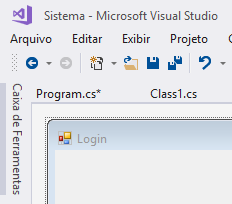


Figura 15. Caixa de Ferramentas

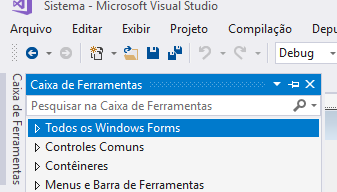


Figura 16. Todos os Windows Forms

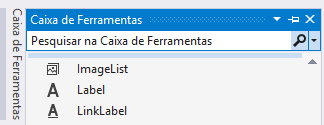


Figura 17. Label

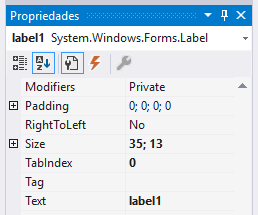


Figura 18. Text label1

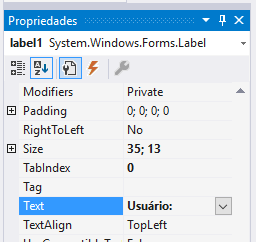


Figura 19. Text agora como Usuário:

Podemos copiar e colar uma label

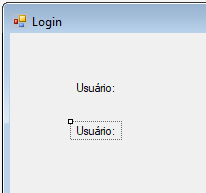


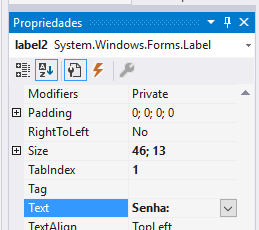
Figura 20. Label Usuário copiado. Alterar o nome da label para Senha:

Figura 21. Text Senha:

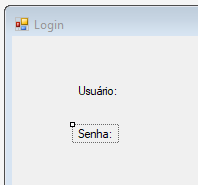


Figura 22. Senha

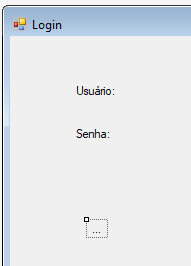


Figura 23. Uma label para mensagens futuras

Vamos alterar o name para lblMensagem

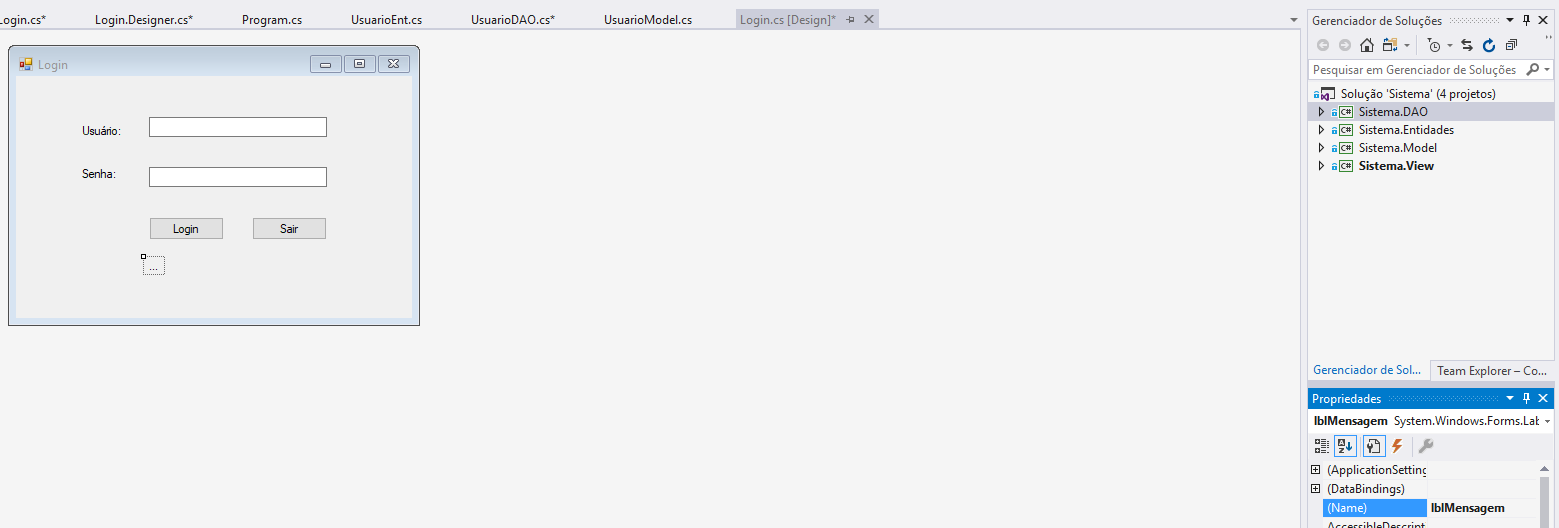


Figura 24. Alterando o name de uma label

Acrescentando uma caixa de texto

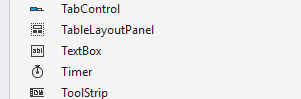


Figura 24. TextBox

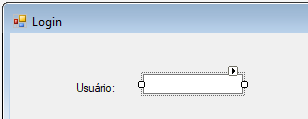


Figura 25. Inserindo um TextBox

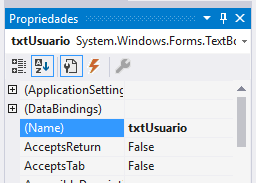


Figura 26. Nomeando a caixa de texto

Copiar e colar a label caixa de texto e renomear

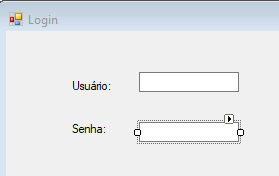


Figura 27. Duas labels

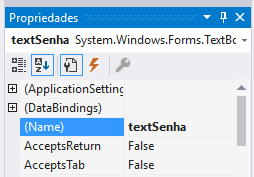


Figura 28. Label renomeada para txtSenha

Coloque um asterisco no campo PasswordChar. Para criar uma mascara no campo.

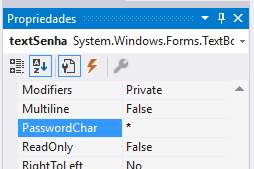


Figura 29. PassordChar com uma mascara

Inserindo os botões

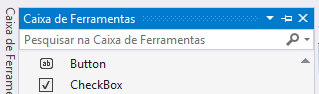


Figura 30. Inserindo um Button

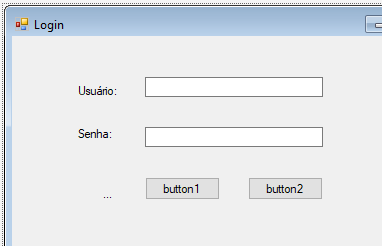


Figura 31. Dois botões

Altere o Text dos botões para Login e Sair

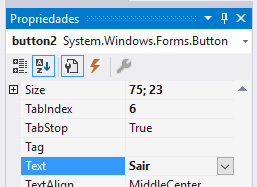


Figura 32. Text do button2

Está ficando assim

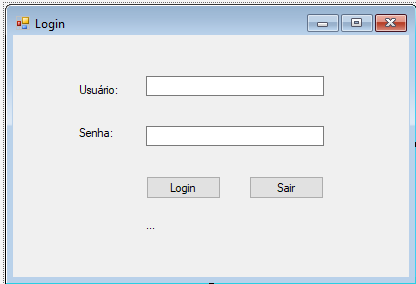


Figura 33. Visualização da tela de login

Altere os name dos botões

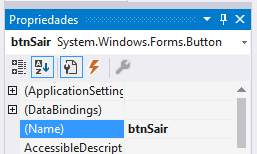


Figura 34. Renomeando os botões

Uma visão de como está ficando

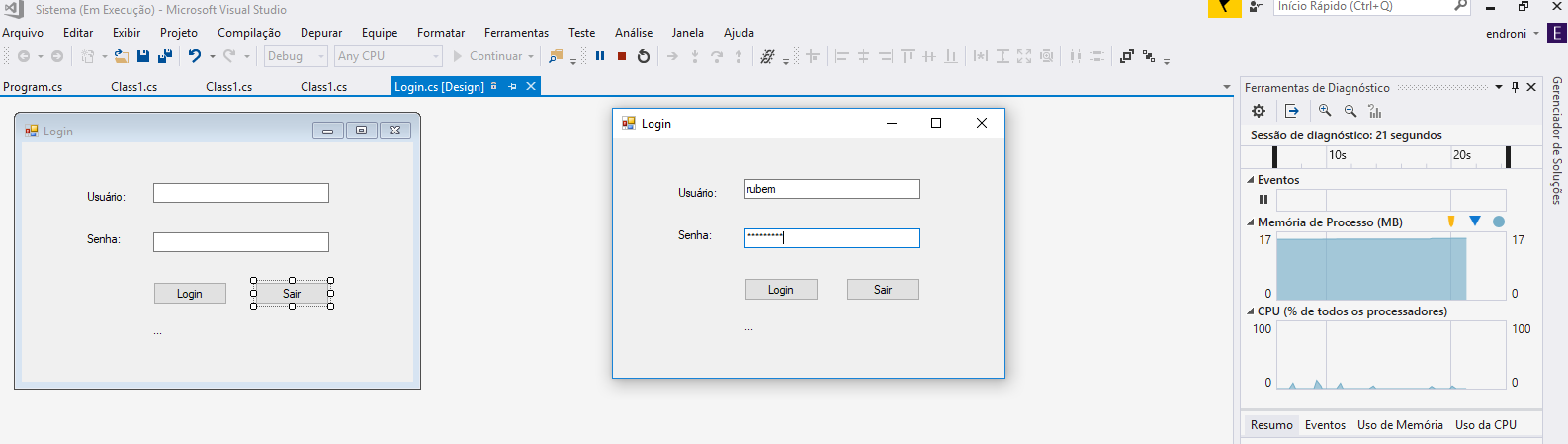


Figura 35. Tela de login em execução

# Camada Entidades

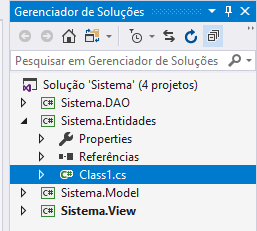
Para trabalharmos com o banco de dados iremos focar na camada Entidades. Vamos iniciar renomeando a Classe Cass1.cs para Usuario.cs 

Figura 36. Primeira Classe

Botão direito na Classe e clique em renomear e após alterar o nome, aceite a alteração clicando em Sim

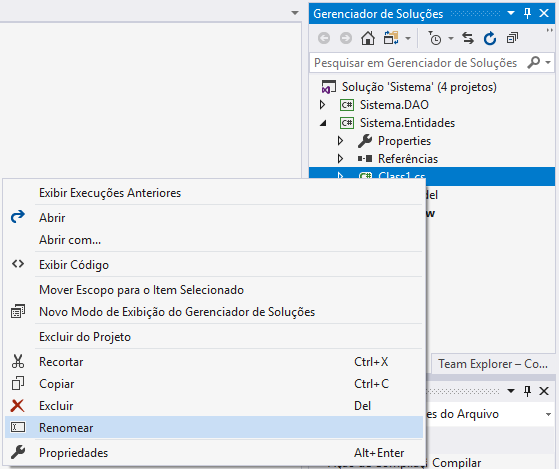


Figura 37. Renomeando Classe

Importante. Todos esses objetos da classe precisam estar como público para que as outras camadas possam utilizar livremente.

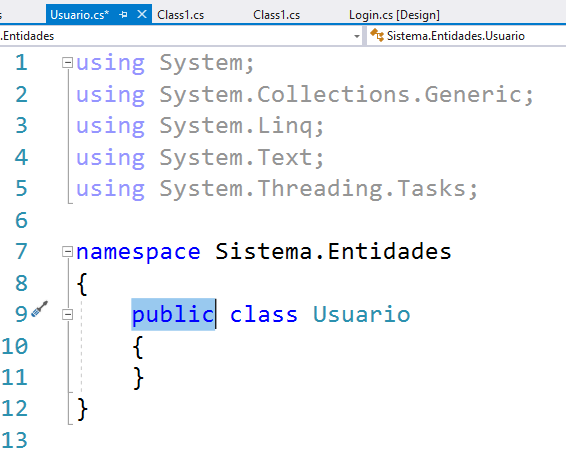


Figura 38. Classe Usuario publica

Dentro das classes os campos criados podem ser privados. Crie então os campo privados. Conforme imagem

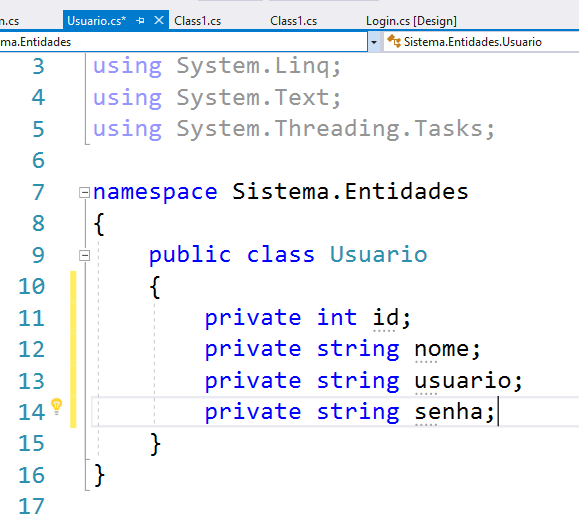


Figura 39. Incluindo campos privados na classe Usuario

## Métodos Get e Set

Tratando de banco de dados, precisamos pensar em envio e recebimento de dados. Para tanto, iremos criar métodos adequados.

Get é um método para recebimento de dados – Recebe dados do Banco.

Set é um método para envio de dados – Insere dados no Banco.

Clique com o botão direito do mouse no campo desejado e clique em **“Ações Rápidas e Refatoração”**. Em algumas versões pode estar apenas **“Refatoração”**

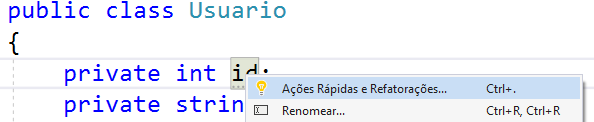


Figura 40. Opção - Ações Rápidas e Refatorações

Em seguida selecione a segunda opção. “Encapsular Campo: “id” (e usar propriedades)”.

Estaremos encapsulando um campo de uma tabela.

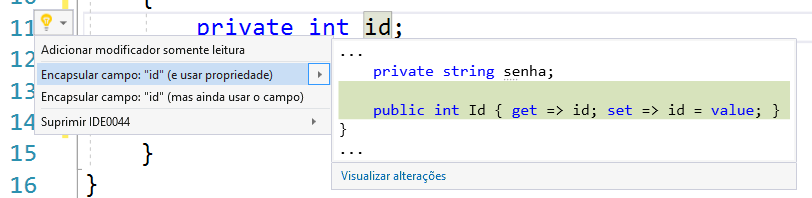


Figura 41. Criando um método

Irá ficar assim

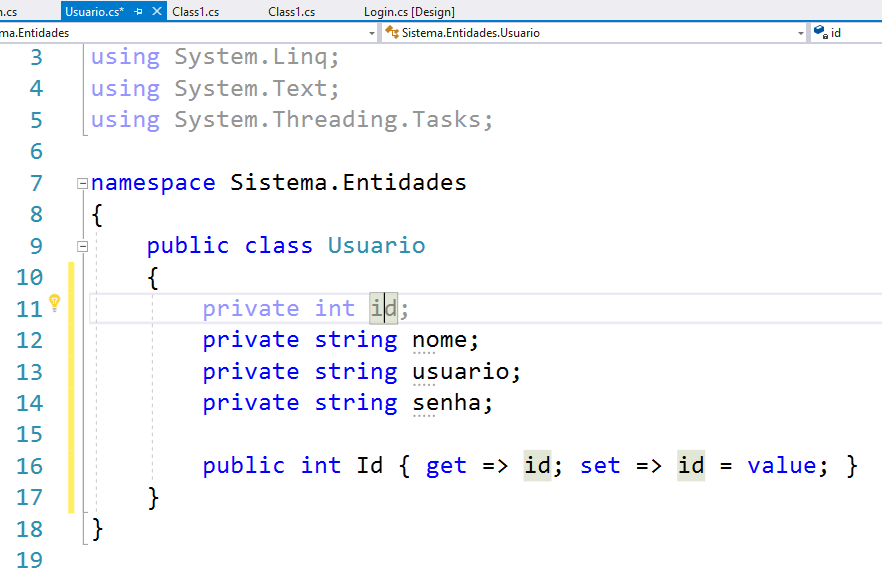


Figura 42. Método público criado

Repita a ação para os outros campos. Irá ficar assim

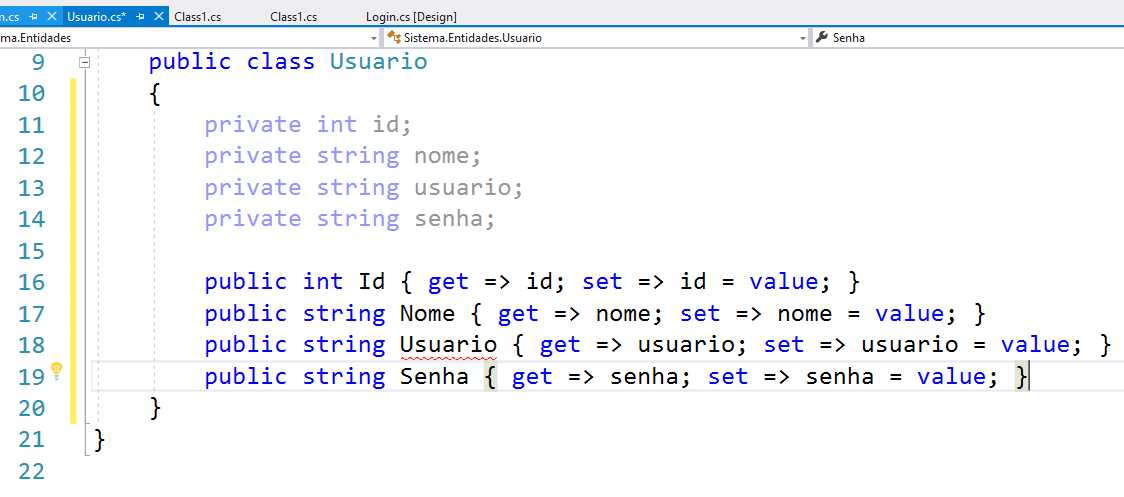


Figura 43. Todos os campos da classe agora estão encapsulados.

Atenção. O Usuario em vermelho. Demostrando um erro. Isso aconteceu porque o nome da classe igual ao da entidade. Para resolver esse problema, iremos renomear a classe **Usuario** para **UsuarioEnt.**

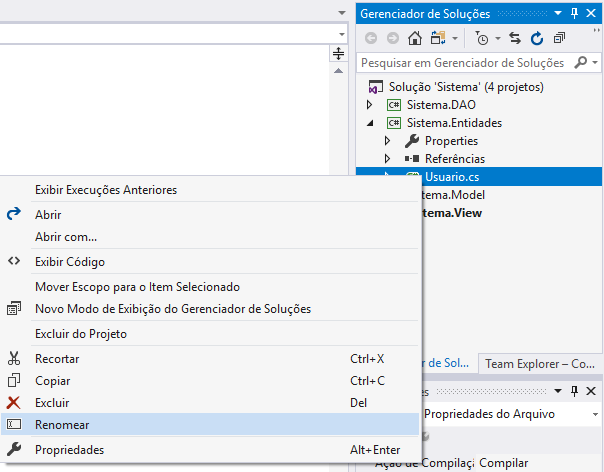


Figura 44. Renomeando nome da classe

Clique em Sim para alterar

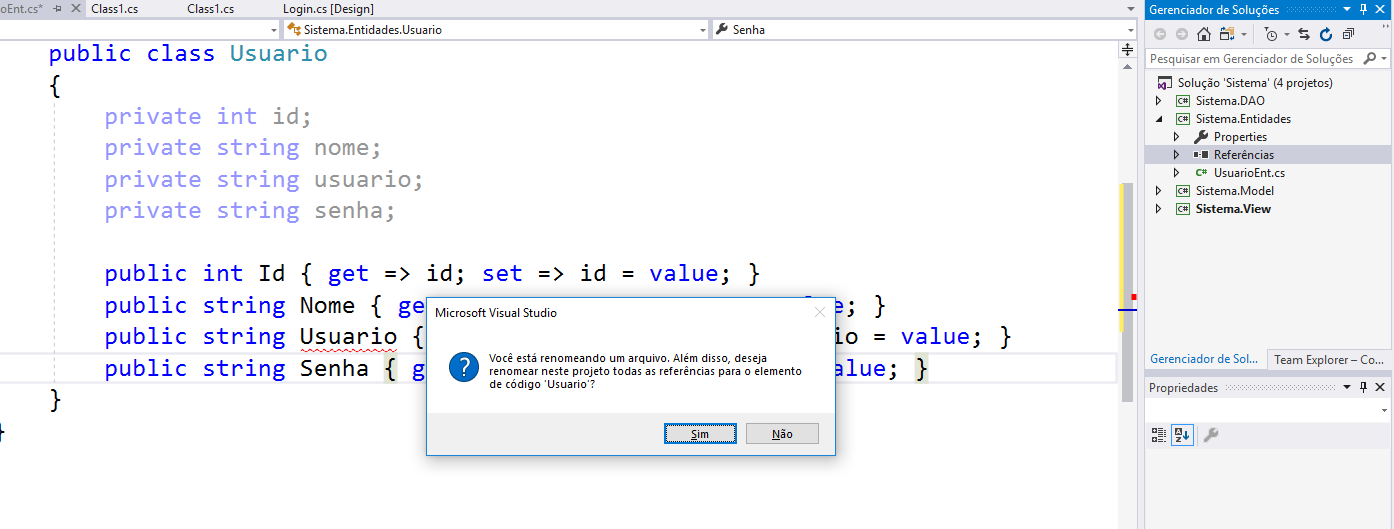


Figura 45. Classe renomeada

# Camada Model

Vamos começar renomeando a classe

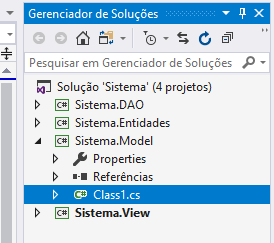


Figura 46. Uma classe da camada Model

Vamos renomea-la

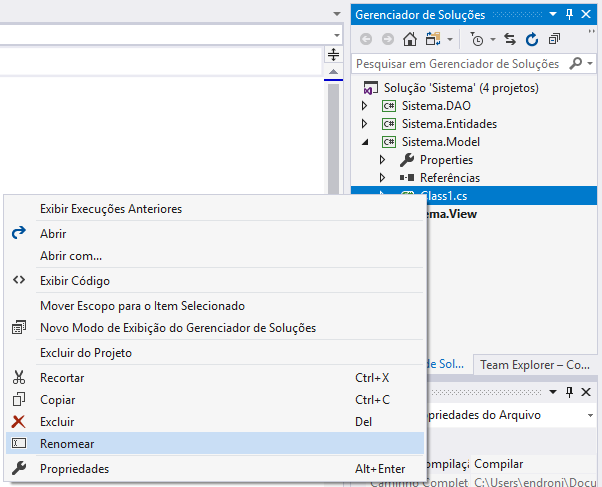


Figura 47. Renomeando uma classe da camada Model

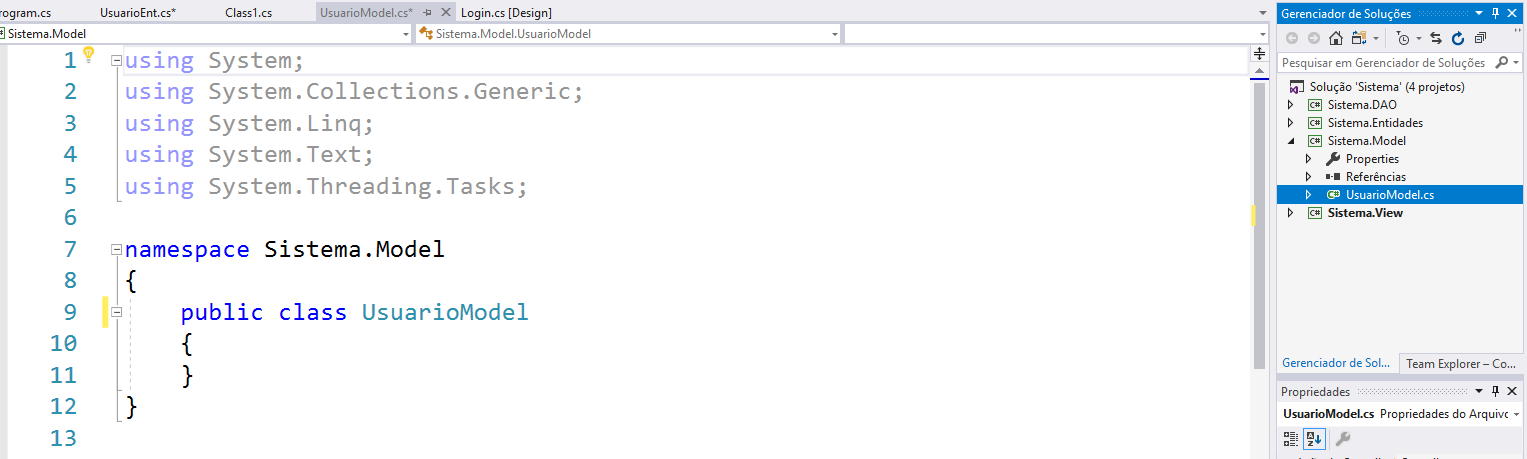


Figura 48. Classe renomeada para UsuarioModel

Caso sua camada não possua uma classe. Podemos criar. Clicando com o botão direito na classe que desejamos criar > Adicionar > Classe

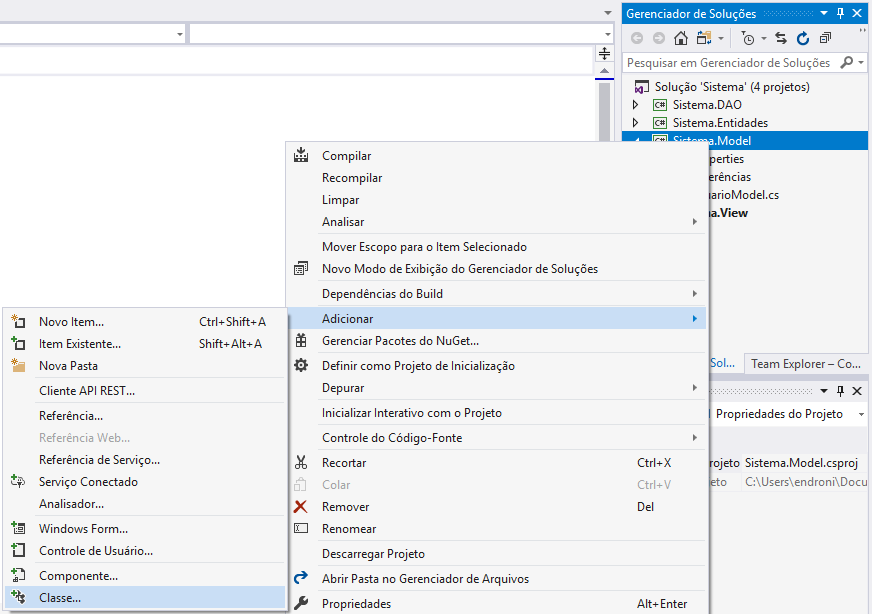


Figura 49. Adicionando uma classe em uma camada

# Camada Control DAO

Vamos renomear a classe existente na camada DAO clicando com o botão direito na classe e selecionando a opção renomear. De class1.cs para **UsuarioDAO.cs**

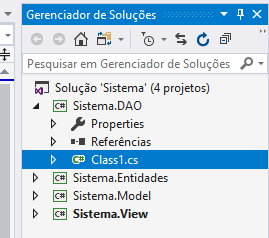


Figura 50. Uma classe da camada DAO (Control)



Figura 51. Renomeando uma classe da camada DAO (Control)

Renomeie para UsuarioDAO.cs e aceite a alteração clicando em SIM



Figura 52. Classe renomeada.

# Relacionando camadas Criando Referências

A forma em que cada uma das camadas se relacionam é de suma importância para um projeto em formato MVC.

A Camada View precisa se relacionar com as Entidades e com o Model, mas não necessita se comunicar com o DAO (Control). A Model comunica com a DAO (Control).

O DAO (Control) se comunica com o Banco de Dados e pode se comunicar com o Model que por sua vez se recebe dados do Control.

Colocar aqui o Desenho

A camada Entidade, também conhecida como a quarta camada, pode se relaciona com todas as outras camadas

## Adicionando referências na camada View

A camada View precisa se comunicar com a camada Entidades e Model

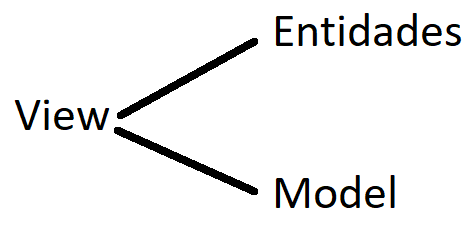


Figura 53. Camada View se relacionanado com as camadas Entidades e Model

Clique com o botão direito na camada View > Adicionar > Referência...

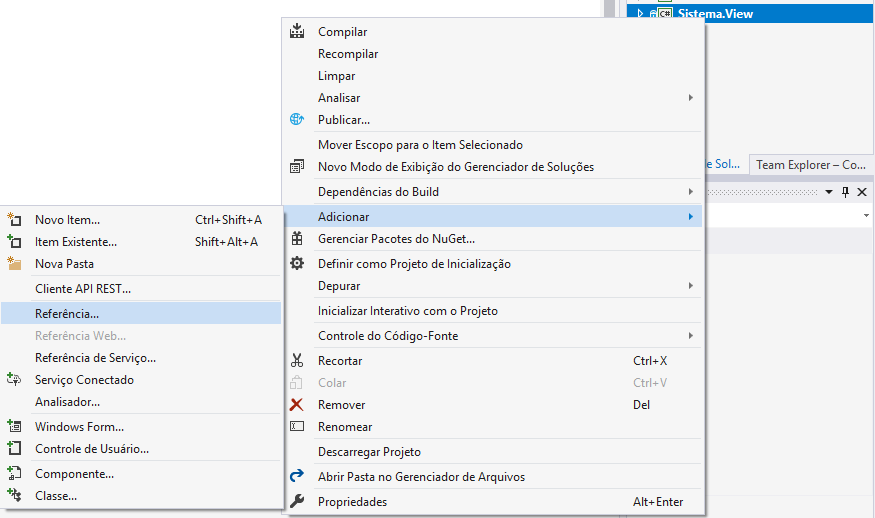


Figura 54. Botão para Adicionando Referência

Clique em Projetos

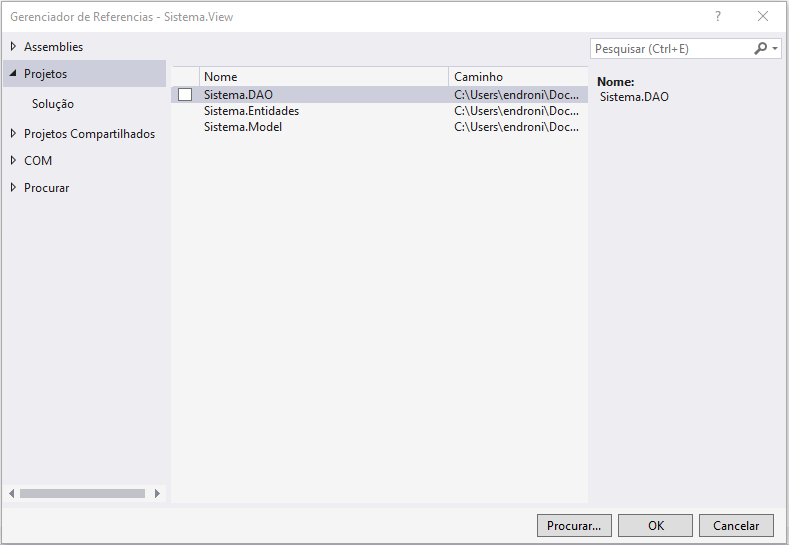


Figura 55. Tela de Referências - Opção Projetos

Selecione o Sistema.Entidades, o Sistema.Model e clique em OK

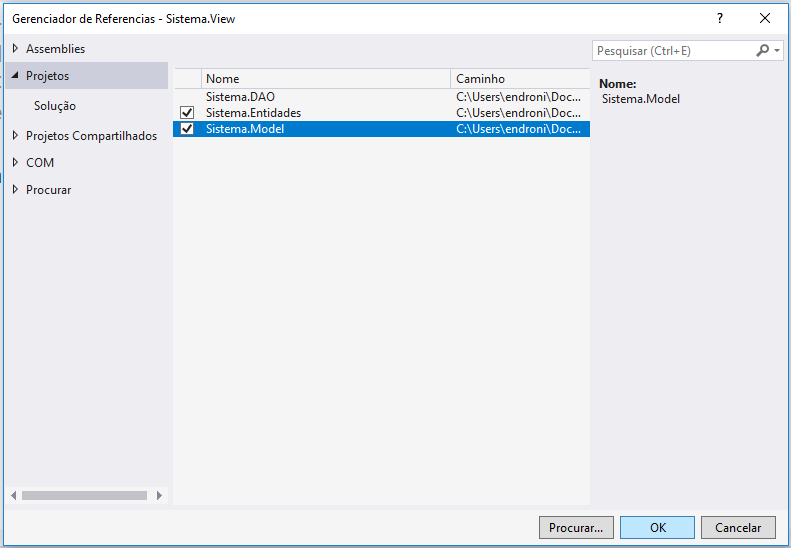


Figura 56. Adicionando referências entre camadas

Podemos abrir o Formulário clicando em Login.cs

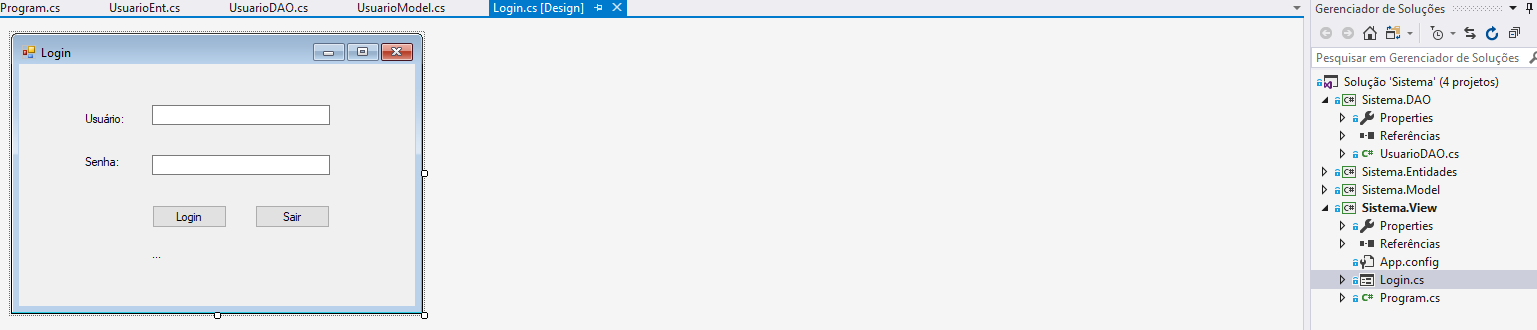


Figura 57. Abrindo o formulário com duplo clique em Login.cs

Para abrir a classe precisamos expandir o Login.cs clicando na setinha e dar um duplo clique em Login.Designer.cs

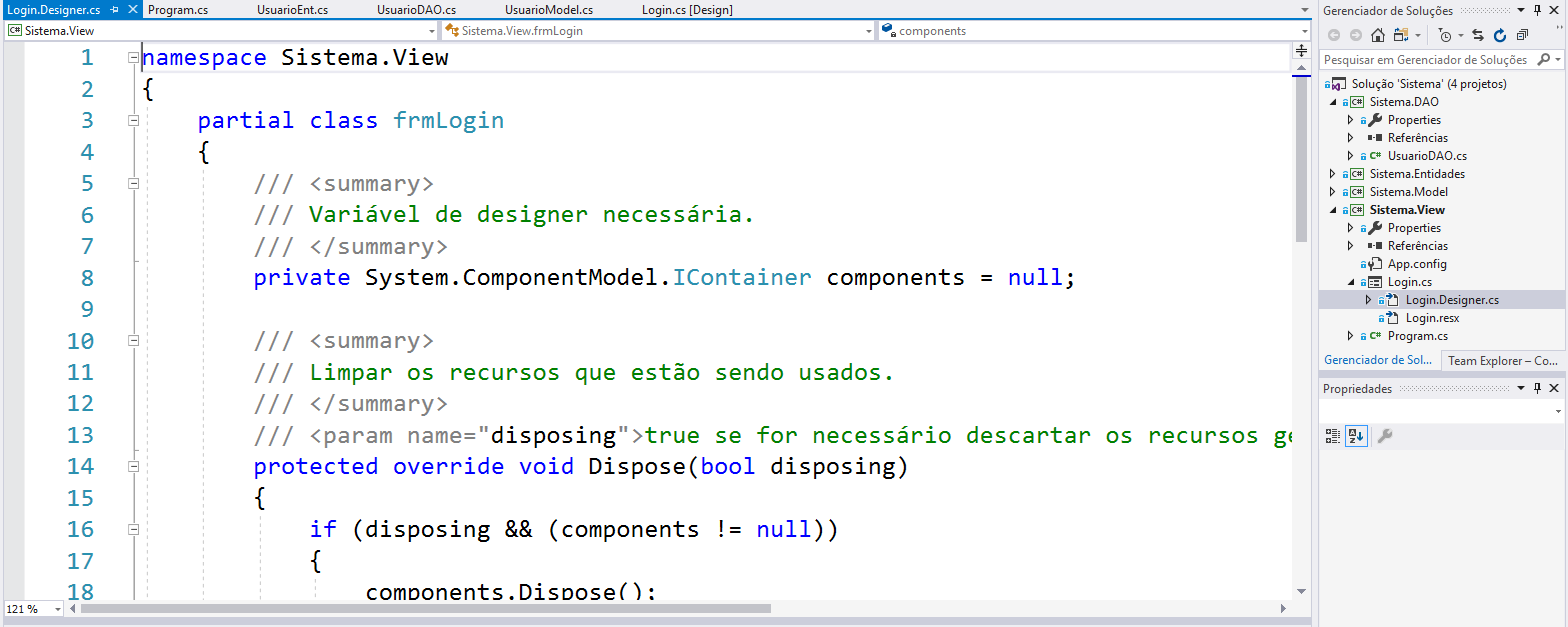


Figura 58. Classe Login aberta.

O Login.resx possui informações de comunicação com o banco e alguns outros recursos.

Expandindo o Login.Designer.cs do Visual Studio e dando um duplo clique em frmLogin conseguimos ter acessos internos quanto ao design do formulário.

Para testar. Vamos tentar adicionar dois using, um para Sistema.Model e um Sistema.Entidades. Basta começar a digitar que já aparece a opção com as classes relacionadas.

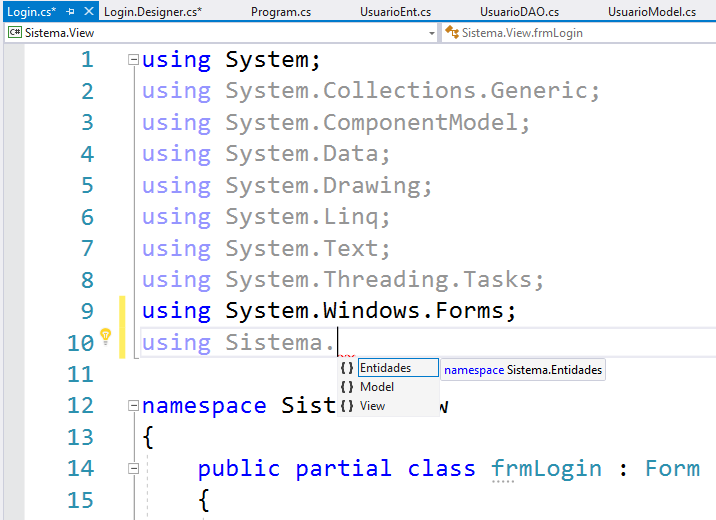


Figura 62. Adicionando using de outra classe

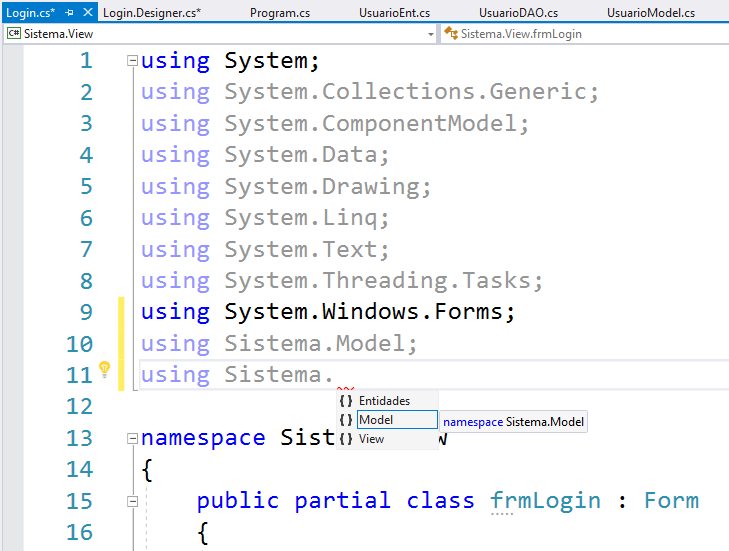


Figura 63. Adicionando using de outra camada. Irá ficar assim

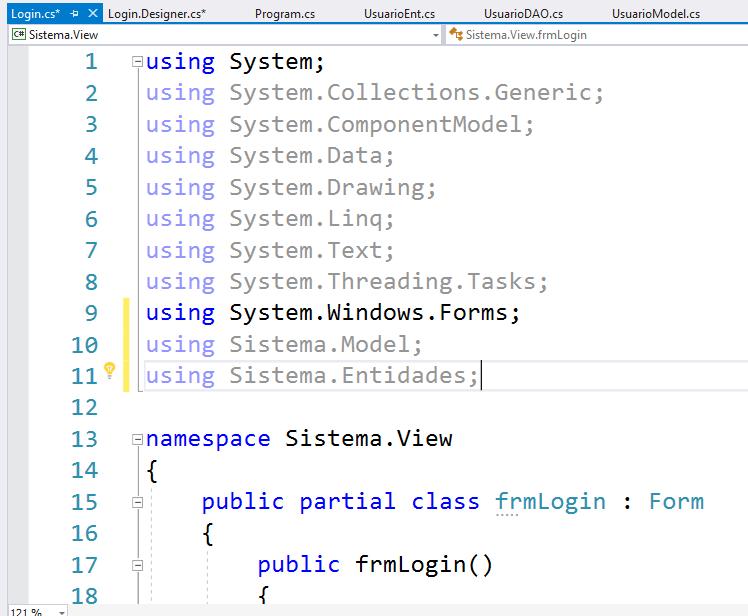


Figura 64. Recursos adicionados à classe frmLogin

## Atenção erro!

Um duplo clique no objeto cria um evento que deve ser excluído caso não se tenha a intenção de trabalhar com esse. Exemplo:

Dê um duplo clique no formulário conforme figura abaixo

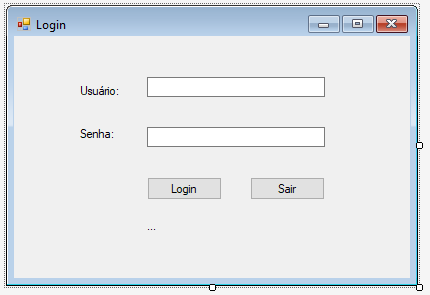


Figura 59. Formulário de Login

Esse evento selecionado da figura abaixo, foi gerado devido ao duplo clique no Formulário de Login. Não se esqueça de excluí-lo.

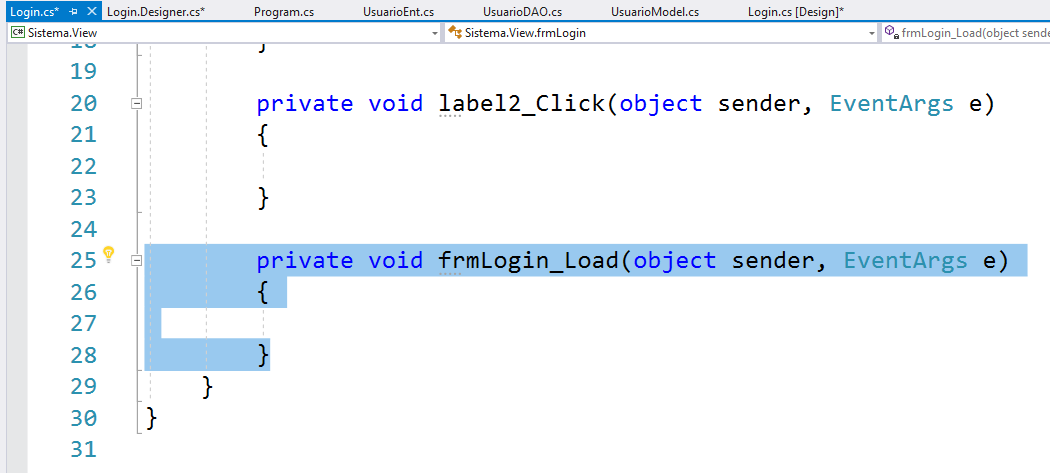


Figura 60. Evento criado devido duplo clique no Formulário de Login

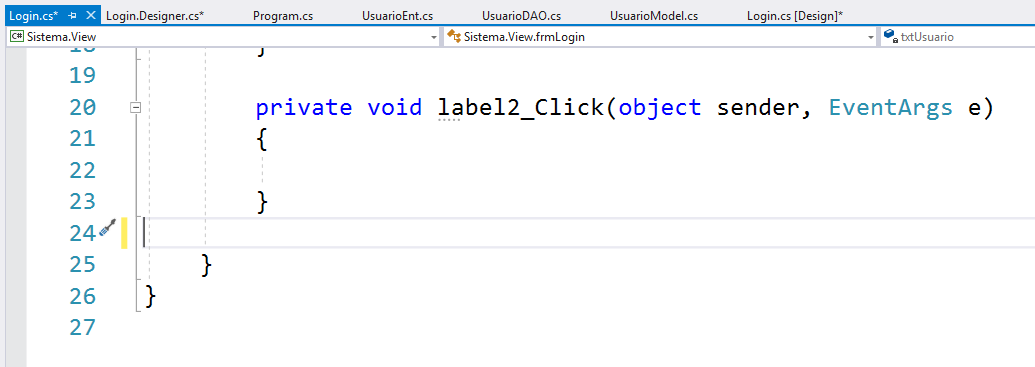


Figura 61. Evento da classe Login.cs excluído

Excluir esse evento faz gerar um erro que é apresentado na tela do formulário do login

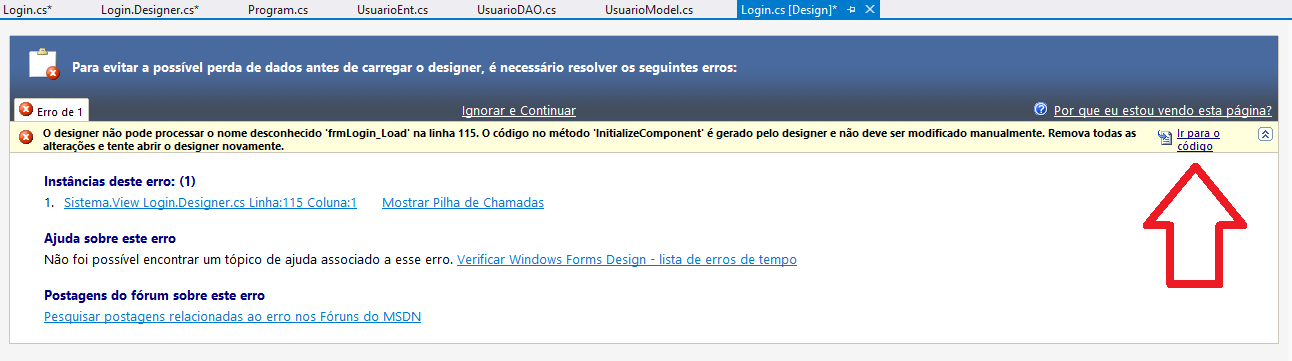


Figura 62. Mensagem de erro

Para remover esse erro, clique em ir ao código (botão aparece na figura acima).

E seremos direcionados para a linha onde está o erro. Selecione a linha, exclua e salve com o Ctrl S .

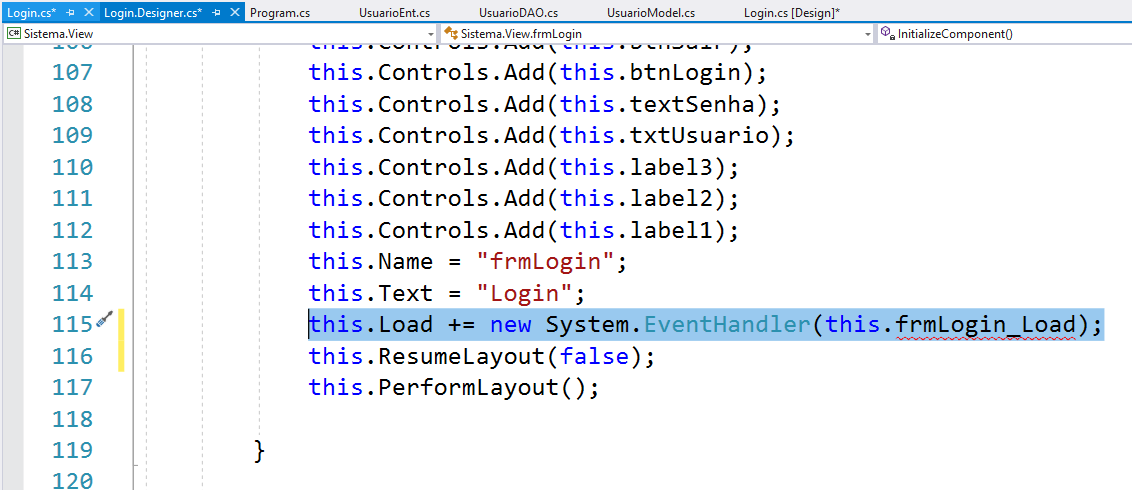


Figura 63. Linha fazendo referência a um evento já não mais existente.

Isso aconteceu porque foi excluído o evento, mas não foi excluído a instancia do evento.

# Adicionando Referência na camada Model

A camada Model precisa se comunicar com a Control (DAO) e a Entidades

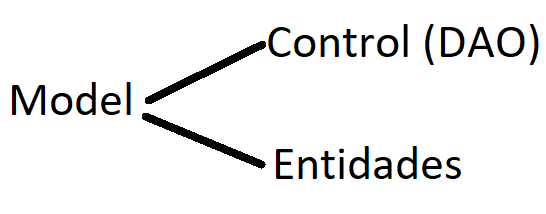


Figura 64. Camada Model se relaciona com a DAO e a Entidades

Clique com o botão direito na camada Model > Adicionar > Referência...

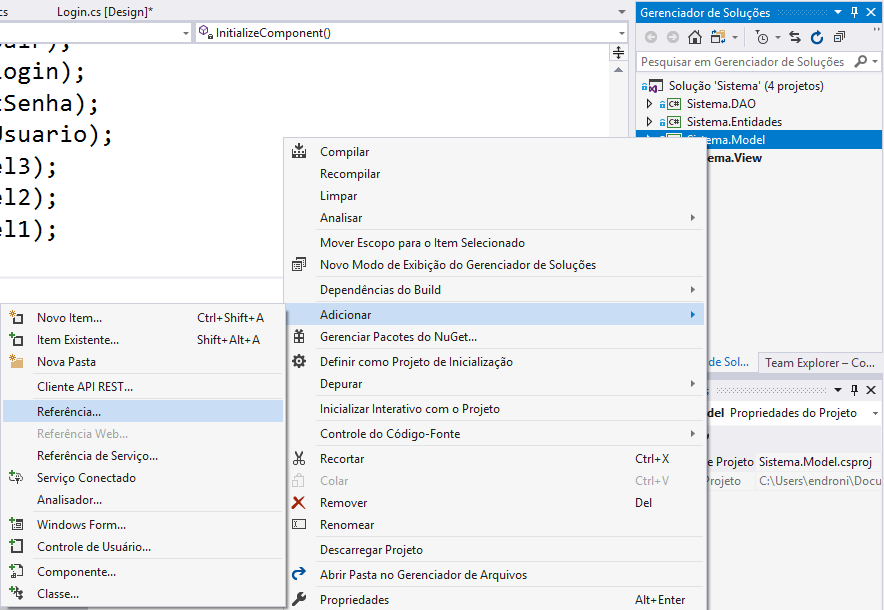


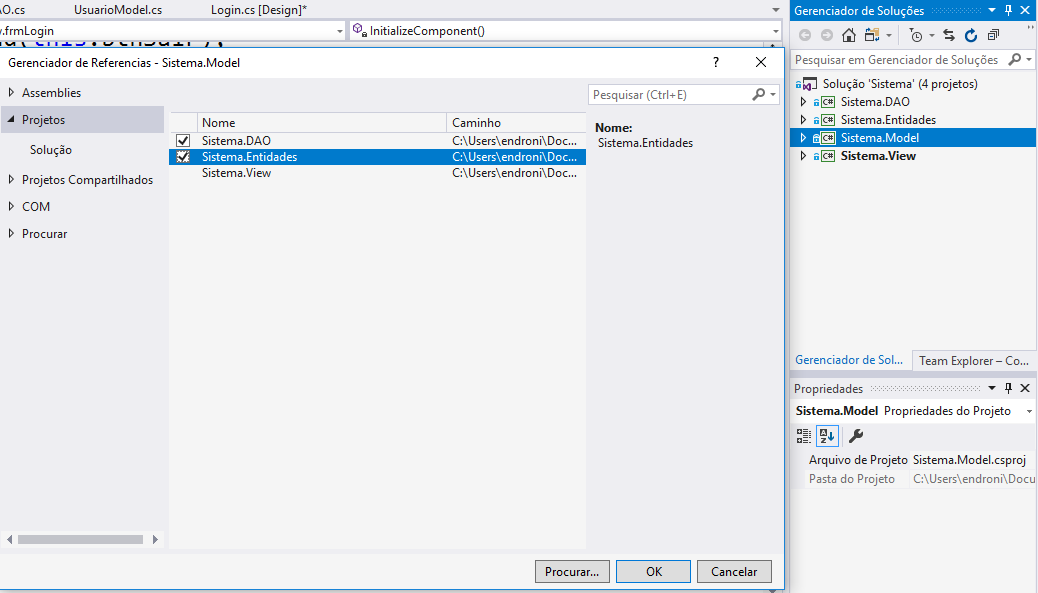
Figura 64. Adicionando Referência na camada Model

Figura 65. Adicionando referências a uma classe

Agora precisamos adicionar os using na classe UsuarioModel

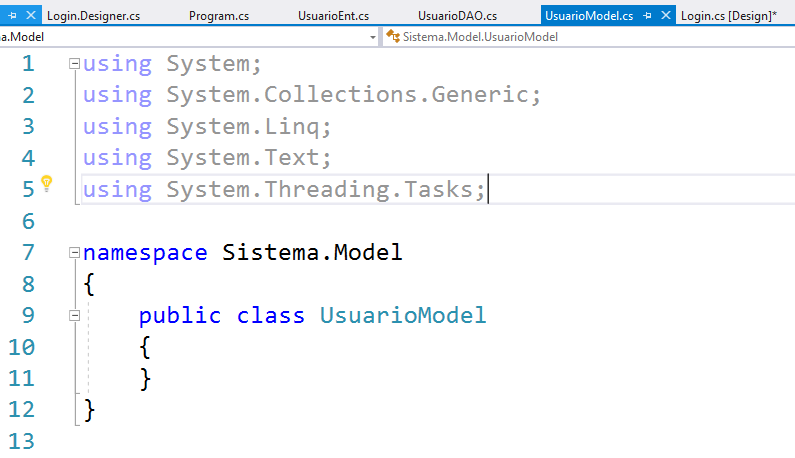


Figura 66. Classe UsuarioModel.cs

Irá ficar assim a classe UsuarioModel.cs

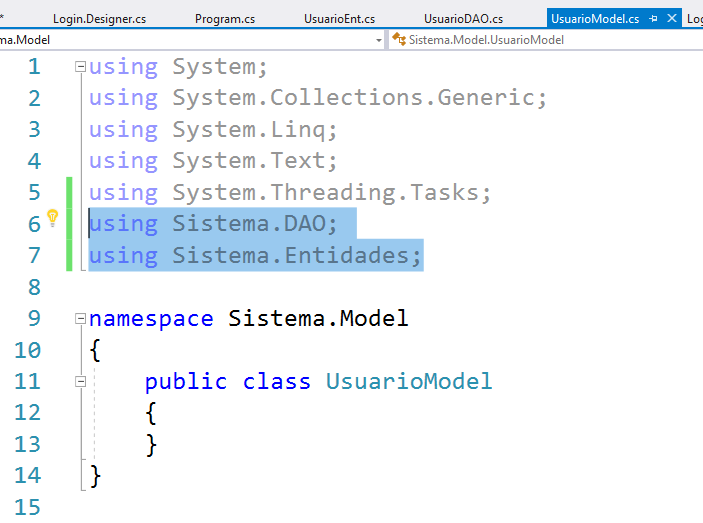


Figura 67. Recursos Adicionados à classe UsuarioModel

# Adicionando Referências à classe DAO (Control)

A classe Control só precisa se relacionar com a Entidades



Figura 68. Camada Control se relacionando com a camada Entidades

Clique com o botão direito na camada DAO > Adicionar > Referência...

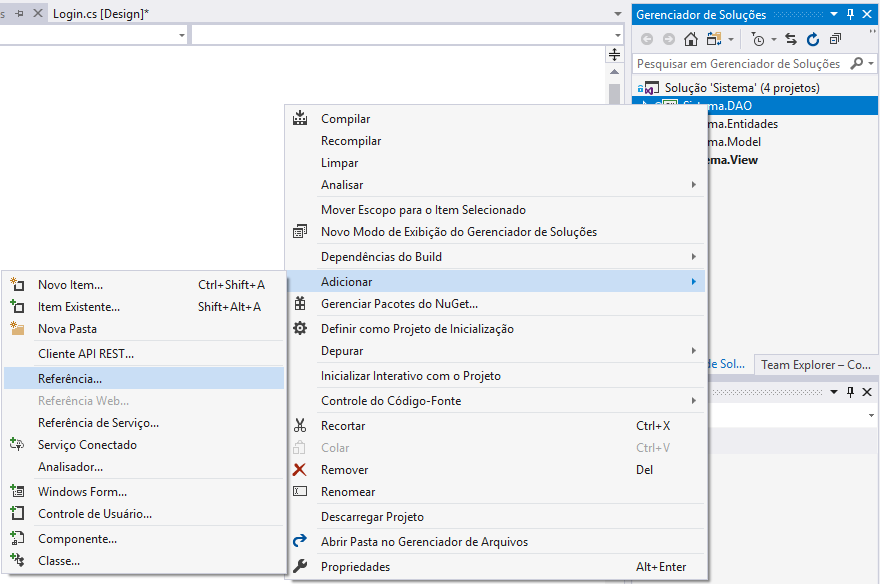


Figura 69. Adicionando uma referência à camada DAO

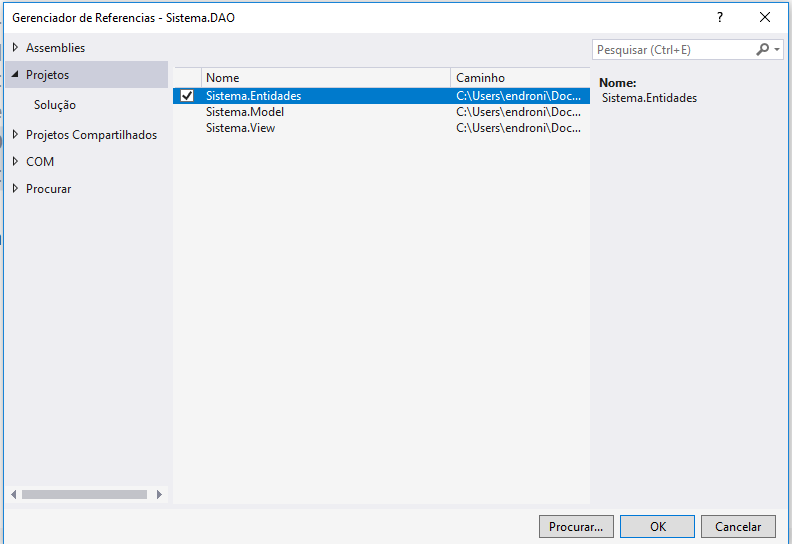


Figura 70. Adicionando uma referência à uma camada

Concluindo, as referências

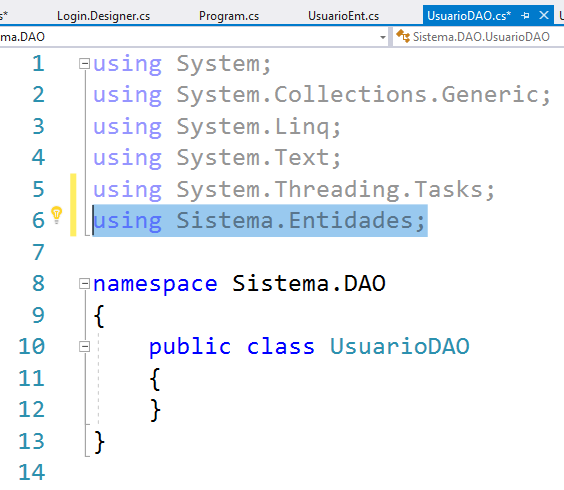


Figura 71. Sistema.Entidades em uso

# Criando o Banco de Dados

Abra o Microsoft SQL Management e se conect na sua instancia. Clique em New Query – Nova Consulta e execute o script abaixo para criar o banco e a tabela

CREATE DATABASE db\_mvc;

USE db\_mvc;

GO

CREATE TABLE usuarios

(

id INT NOT NULL IDENTITY(1,1) PRIMARY KEY,

nome NVARCHAR(50),

usuario NVARCHAR(50),

senha NVARCHAR(50)

);

Agora no Visual Studio, clique com o botão direito em **Server Explorer** depois com o botão direito em **Data Connections** e **Add Connection**.

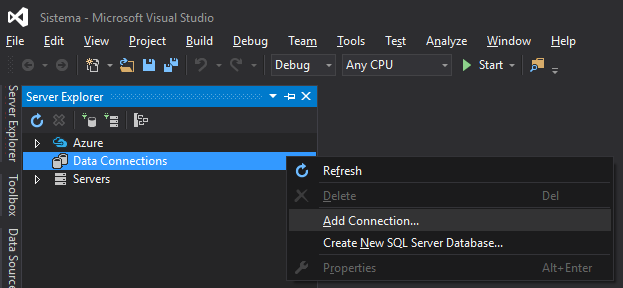


Figura 72. Server Explorer

Selecione o Microsoft SQL Server

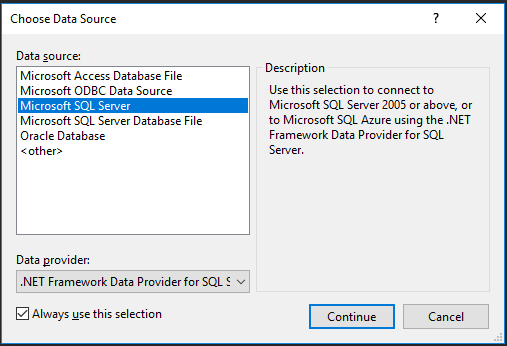


Figura 73. Selecionando o banco de dados

Selecione o nome do servidor. Para saber qual é, podemos desconectar o Microsoft SQL Management e reconectar

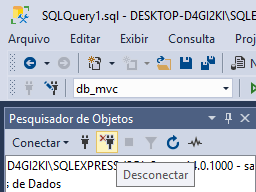


Figura 74. Desconectando o Microsoft SQL Management

E clique em reconectar

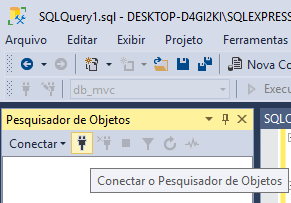


Figura 75. Reconectando o Microsoft SQL Management

Agora copie o Nome do Servidor

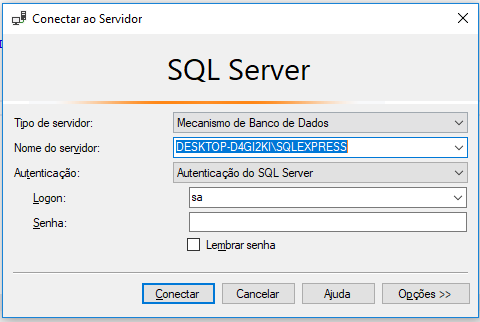


Figura 76. Janela de Conexão com o SQL Server

Selecione o nome e o tipo de autenticação do banco de dados

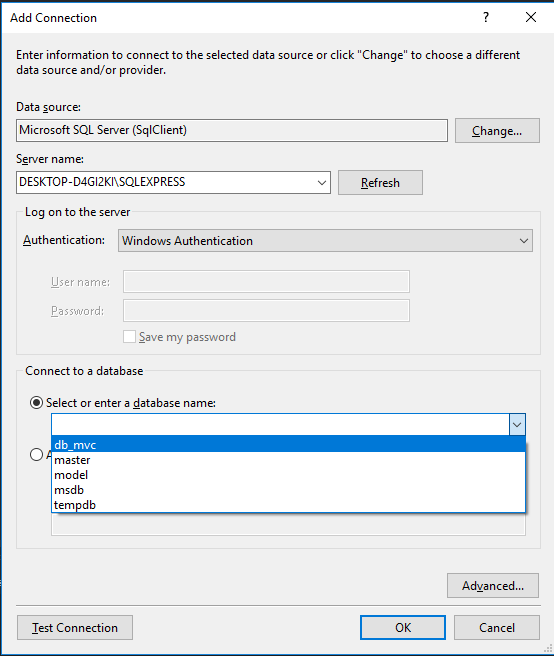


Figura 77. Selecionando o banco

Ao selecionar o banco clique em OK



Figura 78. Dabase selecionado

Com isso foi criado uma cadeia de conexão com o banco de dados

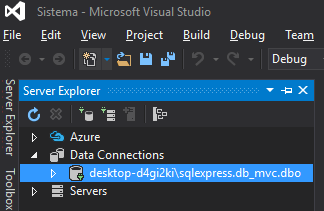


Figura 78. Cadeia de conexão criada

# Criando Tela de Cadastro

No Visual Studio, clique com o botão direito em Sistema.View > Add > New Item

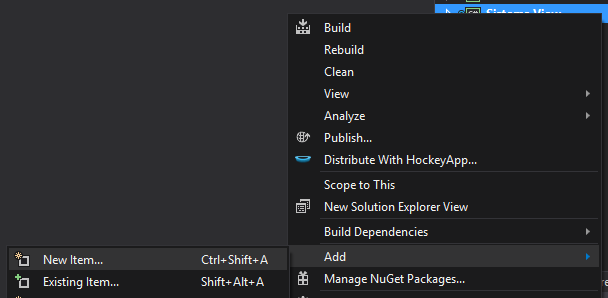


Figura 79. Criando um novo item dentro da camada View

O nome desse item será frmCadusuario.cs

Figura 79. Criando um novo Item frmCadUsuario.cs

Objeto criado

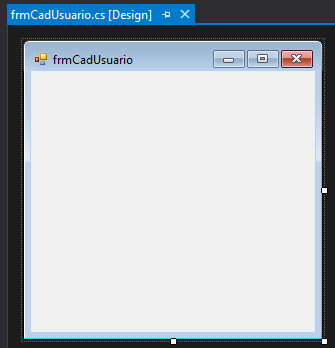


Figura 80. Formulário de cadastro

Vamos criar os rótulos para o cadastro com as Label’s

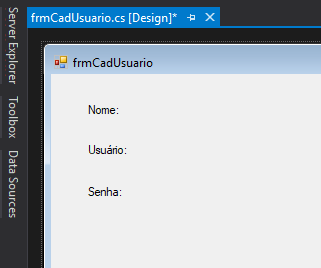


Figura 80. Três rótulos

Adicione os textBox e Button

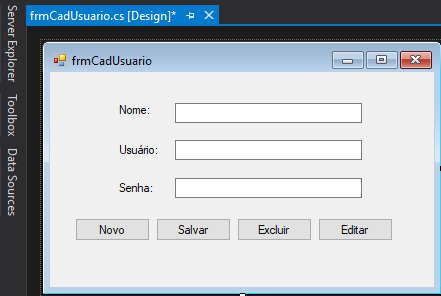


Figura 81. Formulário com as label’s, button’s e txtBox

Vamos adicionar ao formulário uma DataGridView

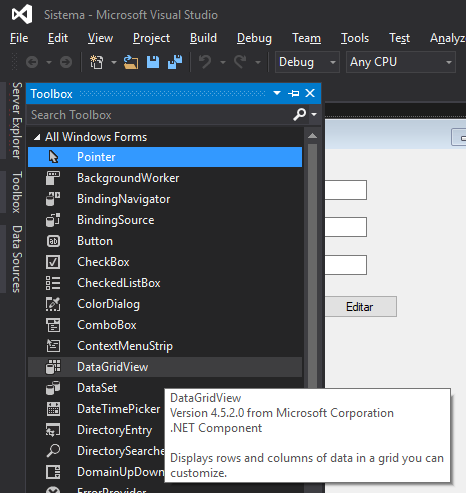


Figura 81. Adicionando uma DataGridView

Desmarque as opções

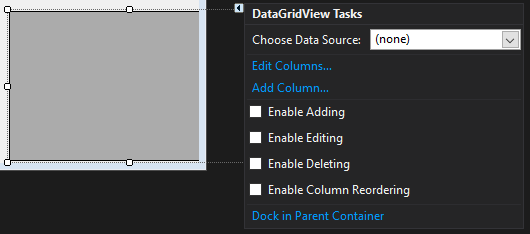


Figura 82. Opções do DataGridView desabilitadas

A tela está ficando assim

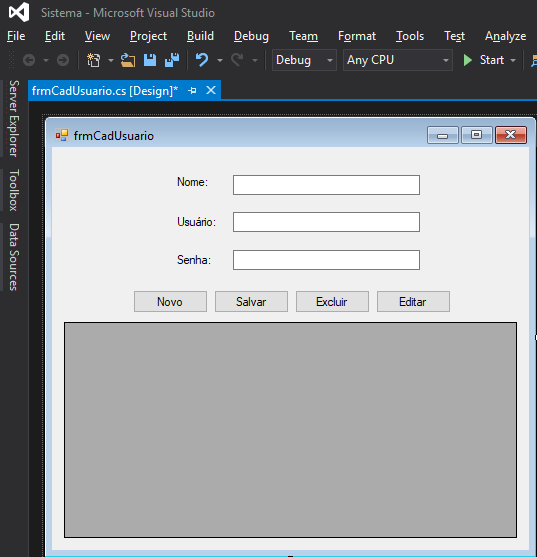


Figura 83. frmCadUsuario.cs

Selecione os textBox e selecionem a opção False para o Enabled

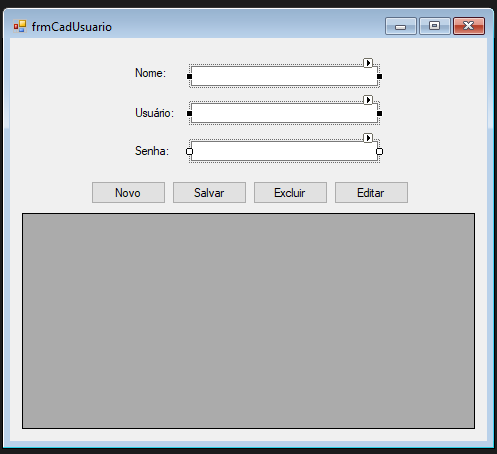


Figura 83. Campos TextBox selecionados

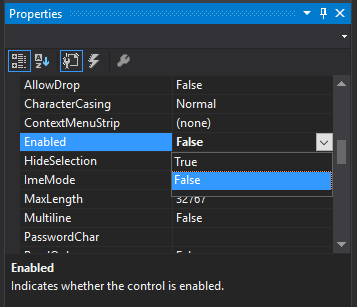


Figura 84. Propriedade Enabled dos TextBox em False

Assim, quando o formulário de cadastro iniciar, os campos estarão desabilitados.

## Criando eventos no Botão Novo

Dê um duplo clique no botão Novo

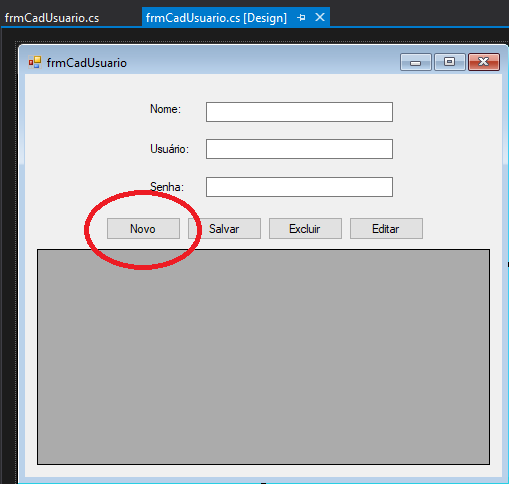


Figura 88. Botão Novo do frmCadUsuario.cs

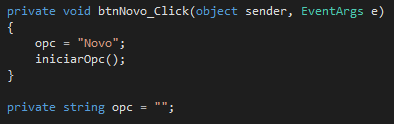
e vamos criar eventos nele.

Dentro do evento do botão Novo vamos criar uma variável opc recebendo o valor “Novo” e inicar o método iniciarOpc().



Figura 85. Evendo btnNovo\_click recebendo uma string Novo

Vamos declarar a variável opc como string recebendo um valor nulo.



Fiura 86. private string opc

Agora vamos criar um método iniciarOpc e dentro dele um switch

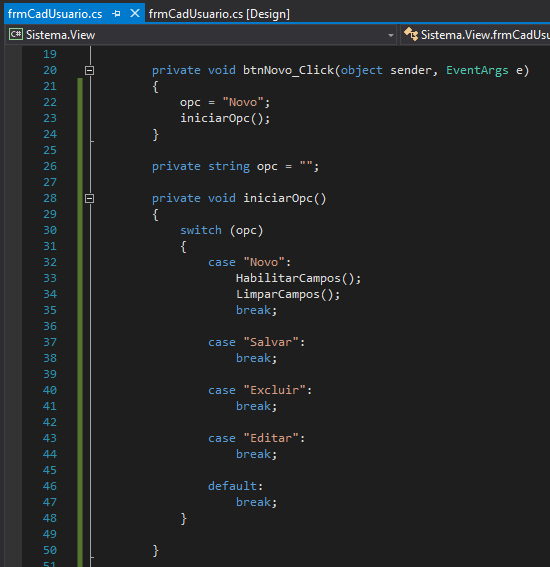


Figura 87. Um switch dentro do método iniciarOpc

Abaixo do método iniciarOpc vamos criar dois métodos, um para habilitar os campos e outro para limpar os campos

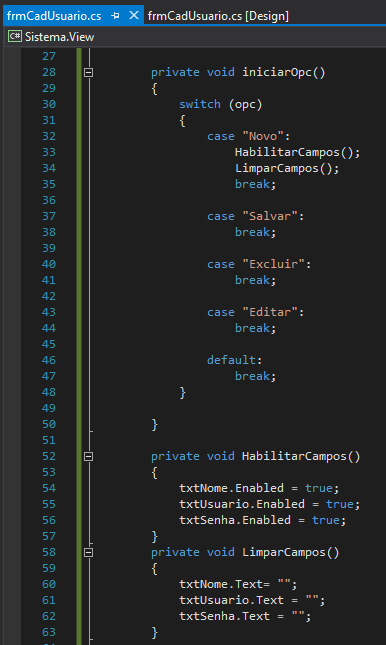


Figura 88. Métodos de HabilitarCampo e LimparCampos criado

Agora vamos criar eventos para os botões de Salvar, Excluir e Editar

Dê um duplo clique no botão Salvar

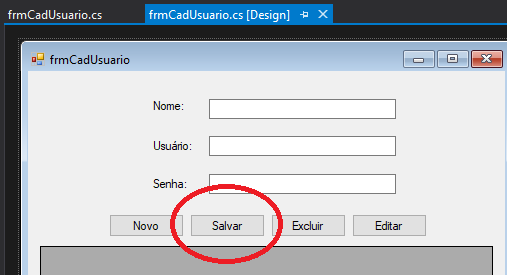


Figura 89. Botão salvar do frmCadUsuario

Acrescente as duas linhas no evento

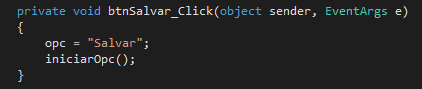


Figura 89. Evento btnSalvar\_click recebendo uma variável e um método

Duplo clique no botão Excluir

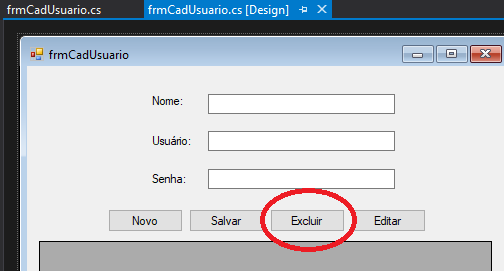
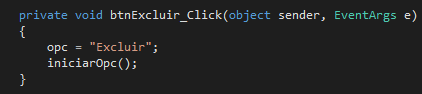


Figura 90. Botão Excluir

Acrescente as duas linhas no evento



Figrua 91. Evento btnSalvar\_click recebendo uma variável e um método

Duplo clique no botão Editar

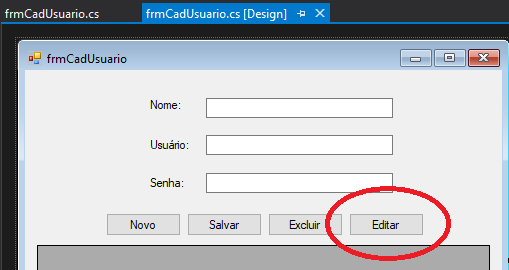


Figura 92. Botão Editar

Acrescente as duas linhas no evento

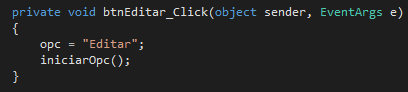


Figura 93. Evento btnSalvar\_click recebendo uma variável e um método

## Um pequeno teste

Vamos criar um evento no botão Login

Dê um duplo clique no botão Login

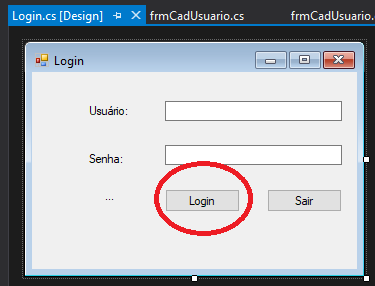


Figura 94. Botão Login no formulário Login

Agora dentro do evento criado, vamos instanciar e chamar o formulário de cadastro com as linhas abaixo.

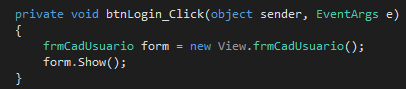


Figura 95. Evento btnLogin\_click chamando formulário frmCadUsuario

Efeito do teste será isso

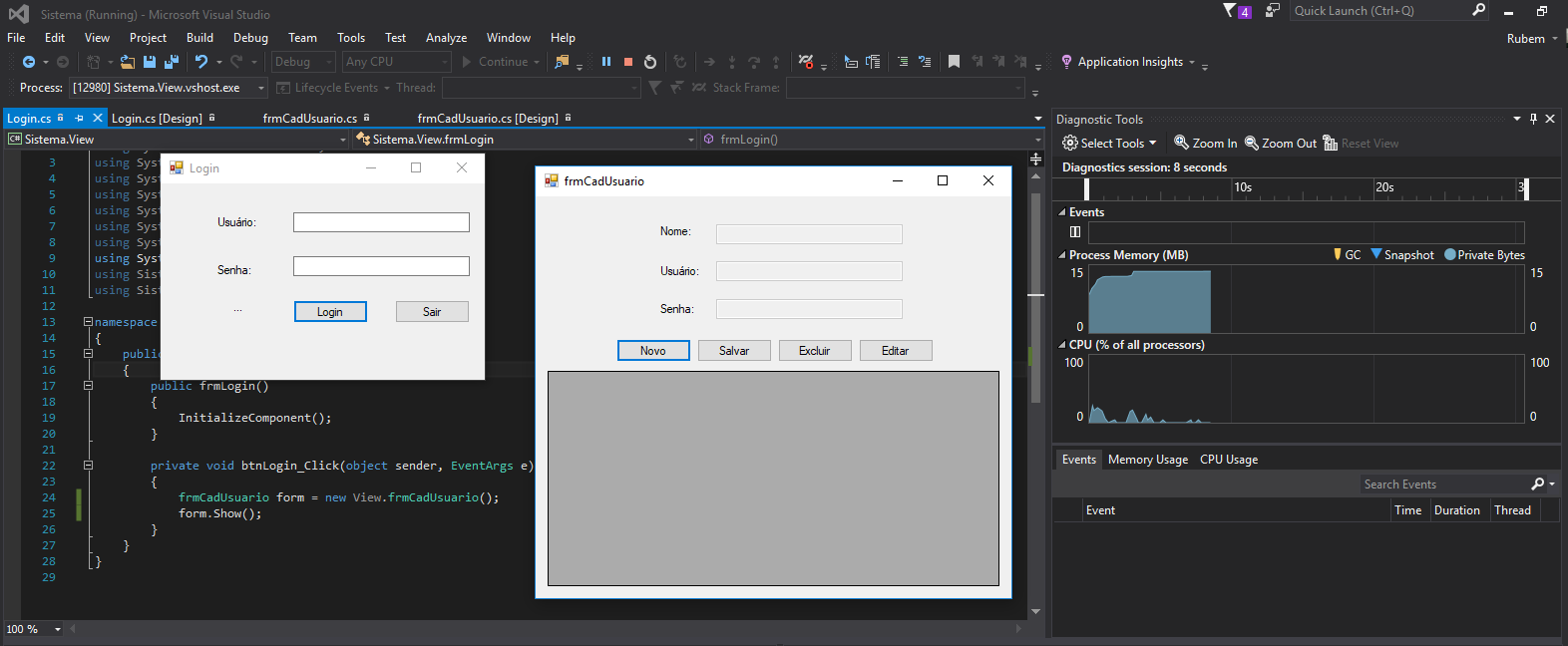


Figura 96. Testando o formulário frmCadUsuario

# Criando a Função Inserir

Antes de qualquer coisa, vamos adicionar os dois imports das camadas Model e Entidade ao formulário frmCadUsuario

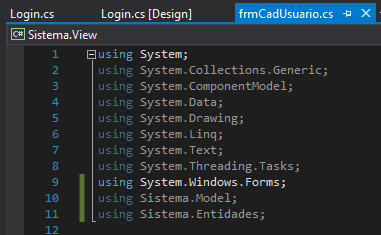


Figura 97. Using no frmCadUsuario

No inicio do código, vamos criar um **objTabela** para instanciar à classe UsuarioEnt.



Figura 98. objTabela instanciado

Dentro do case Salvar vamos criar um try catch digitando **try** e pressionando a tecla tabe duas vezes

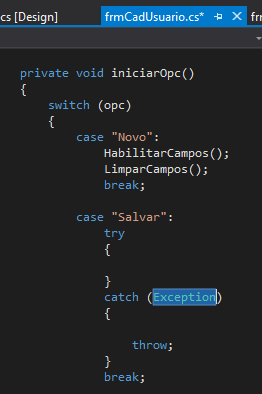


Figura 99. Try catch criado dentro do case Salvar

Vamos criar uma MessageBox para tratar o erro com a variável ex do tipo Exception



Figura 100. MessageBox.Show tratando o erro

Vamos concatenar a mensagem à variável ex para termos informações sobre o erro.

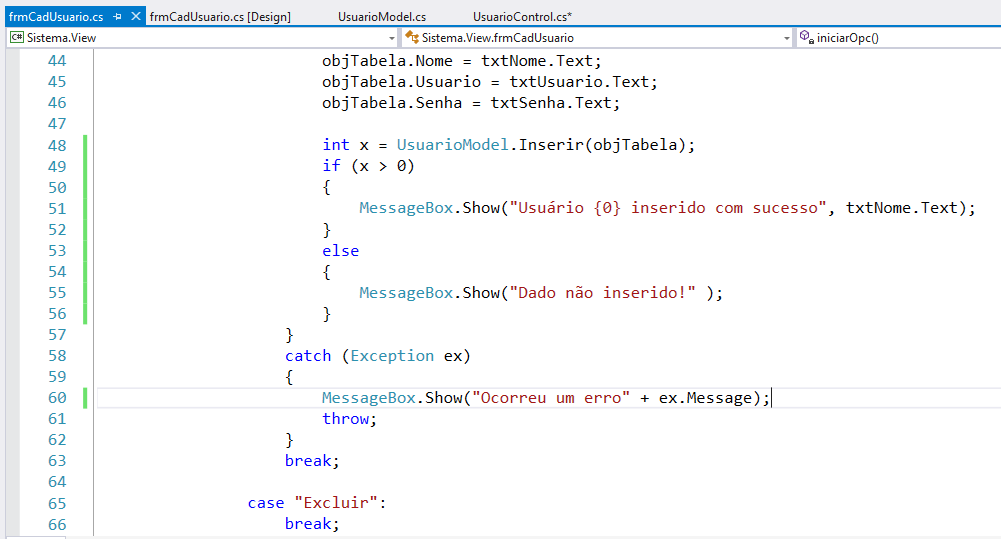


Figura 101. Mensagem de erro concatenada com ex

Agora vamos recuperar os dados que forem digitados dentro do TextBox txtNome, será associado ao campo objTabela.Nome

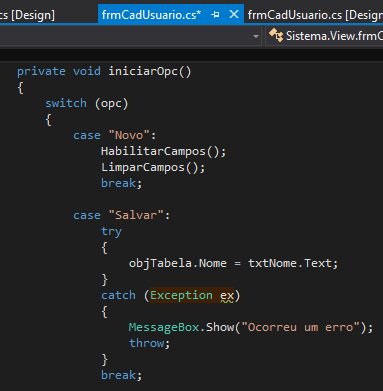


Figura 101. Acrescentado um objeto no try

E vamos repetir para as outras caixas, ficando como a figura abaixo

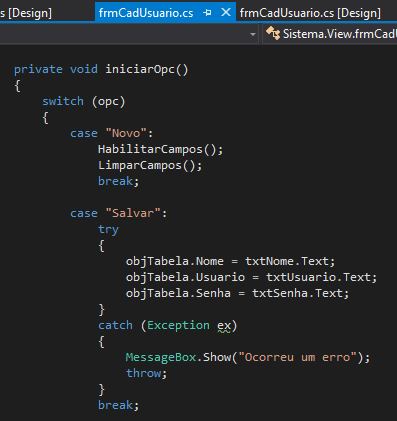


Figura 102. Objetos recebendo dados das caixas de texto

## Criando uma função inserir

Definindo os dados criando um método inserir.

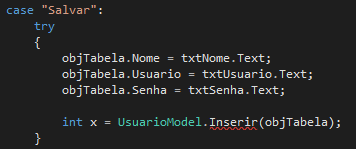


Figura 103. Criando uma variável do tipo inteiro

Como o método Inserir ainda não existe dentro da classe usuarioModel, podemos passar o cursor do mouse em **Inserir**, clicar na setinha ao lado da lâmpada e em gerar método.

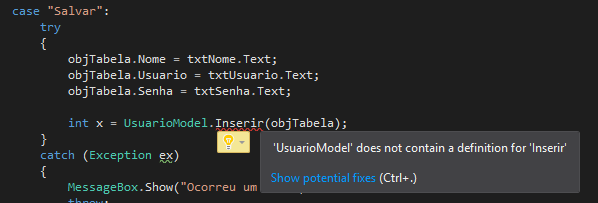


Figura 104. Passando o cursor do mouse em inserir

Clique em Generate method conforme figura abaixo.

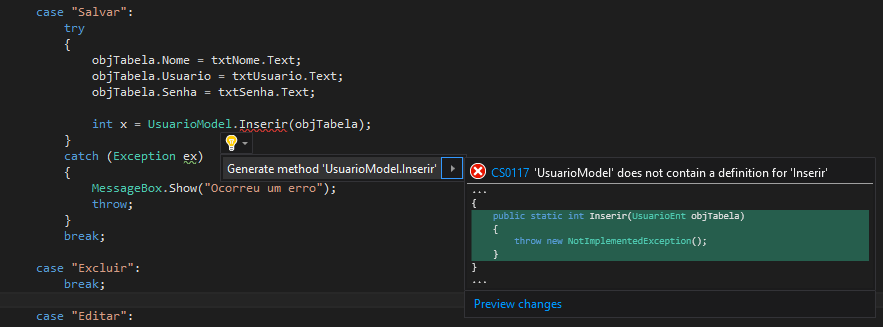


Figura 105. Opção automática para gerar método

Observe que o erro desapareceu



Figura 106. Inserir sem o sublinhado de erro

Observe também que o método foi criado dentro da classe UsuarioModel.cs

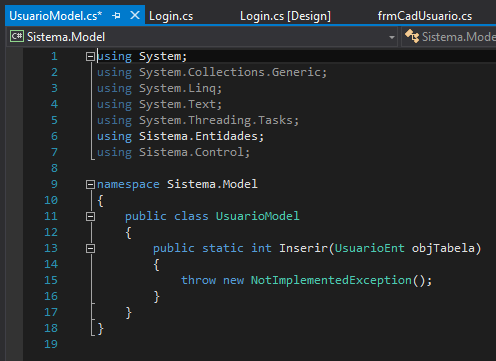


Figura 107. Método Inserir criado dentro da classe UsuarioModel

Esse método não pode receber essa exceção. Portanto precisamos substituí-la por um retorno que irá passar os mesmos dados que serão cadastrados na tabela.

Vamos gerar um método na entidade Controle. Passe o mouse sobre o objeto Inserir depois dê um clique na setinha ao lado da lâmpada e clique em Generate method ‘Usuario.Control.Inserir’

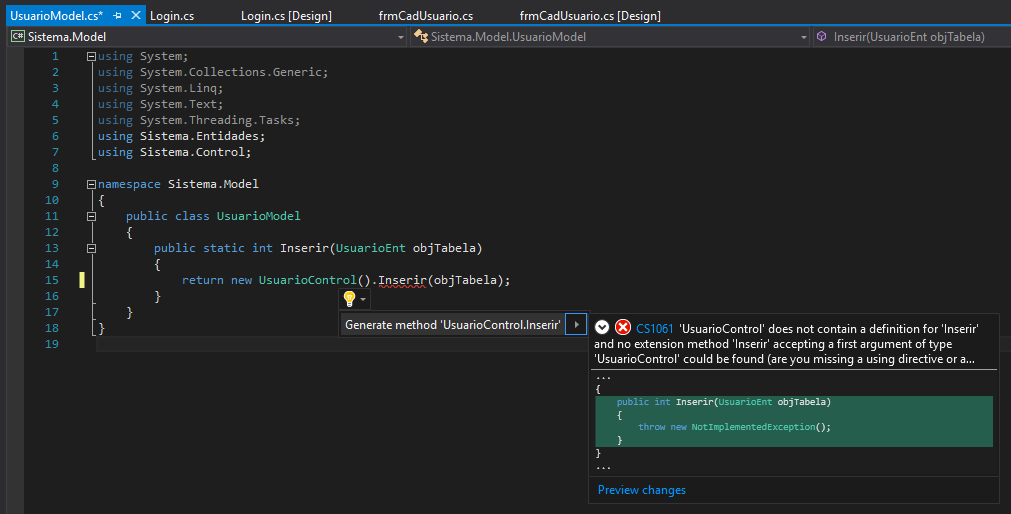


Figura 108. Criando método na classe Usuario

Observe que o erro desaparece

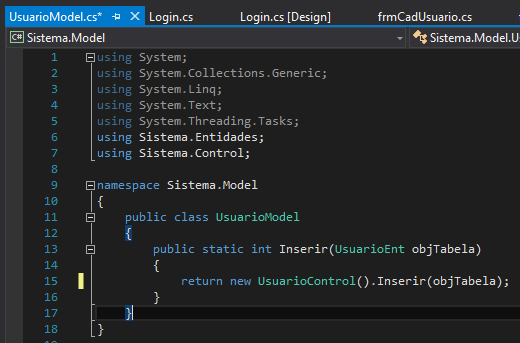


Figura 109. Objeto Inserir sem erro

Dê um duplo clique em UsuarioControl.cs dentro da camada Sistema.Control

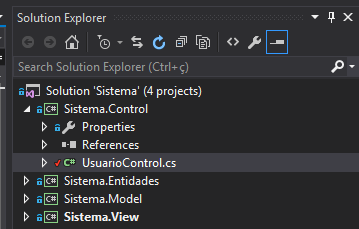


Figura 110. Classe UsuarioControl.cs

O método foi criado na classe UsuarioControl.cs

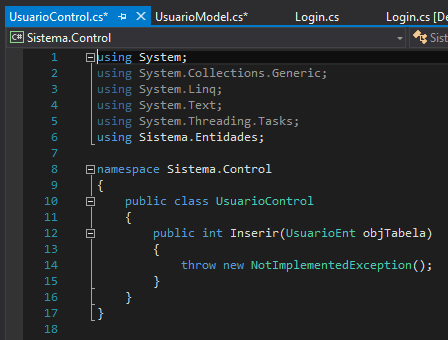


Figura 111. Método criado dentro da classe UsuarioControl.cs

Porém também não preciso da linha de retorno gerada. Apague a linha: **throw new NotImplementedException();**



Figura 112. Método criado e exception excluída

## Comunicando com o Banco

Para comunicar com o banco iremos precisar de uma Conection String.

Acrescente a linha using (SQLConection)

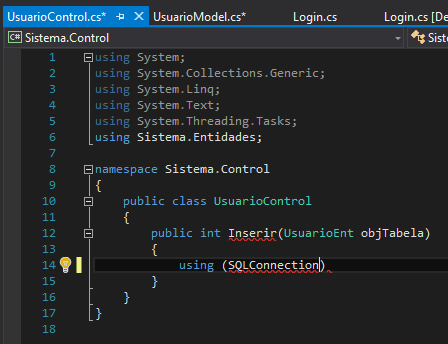


Figura 113. SQLConnection

Essa conexão necessita de uma biblioteca e para isso precisamos adicioná-la passando o mouse em cima do objeto SQLConnection e clicando na setinha ao lado da lâmpada e dando um duplo clique em cima de using System.Data.SqlClient conforme figura abaixo.

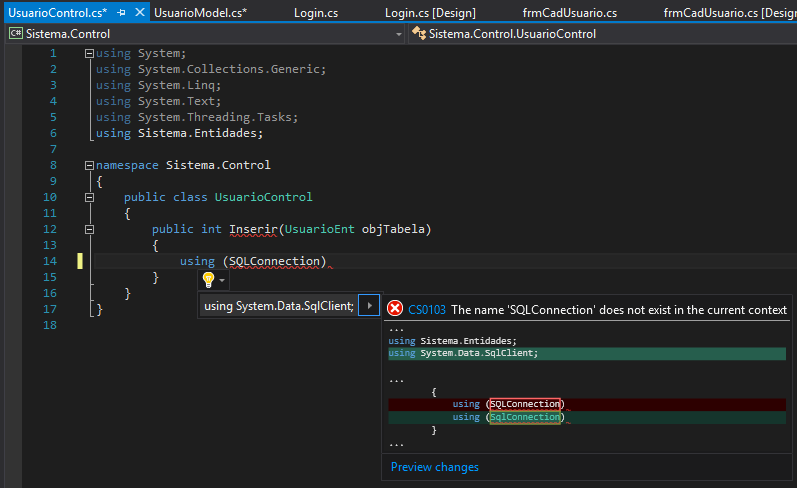


Figura 114. Adicionando biblioteca de conexão com o banco

A biblioteca será adicionada

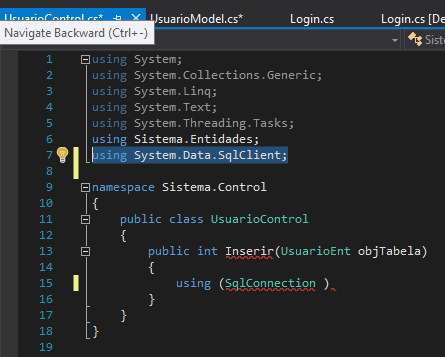


Figura 115. Biblioteca adicionada em destaque

Crie uma variável. No caso utilizamos o nome con e instanciamos.

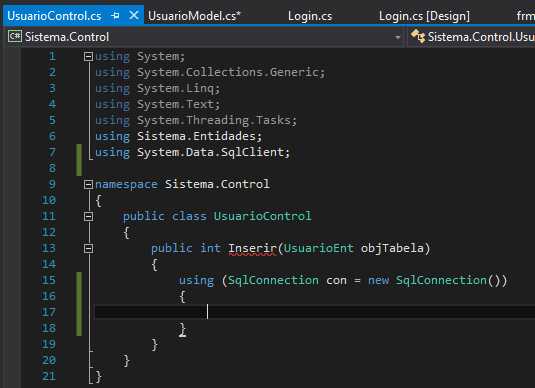


Figura 116. Variável com instanciada com o SqlConnection

Agora vamos fazer uma conexão para a classe Control clicando com o botão direito sobre a camada Control e selecionando Propriedades

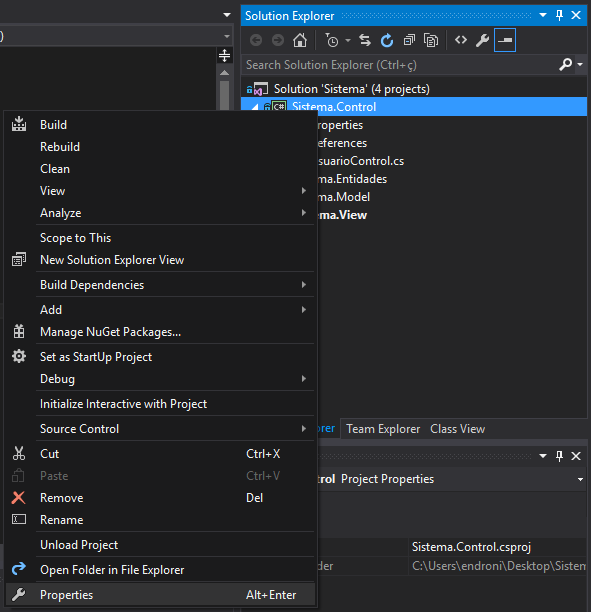


Figura 117. Botão de Propriedades da camada Sistema.Control

Irá abrir as configurações

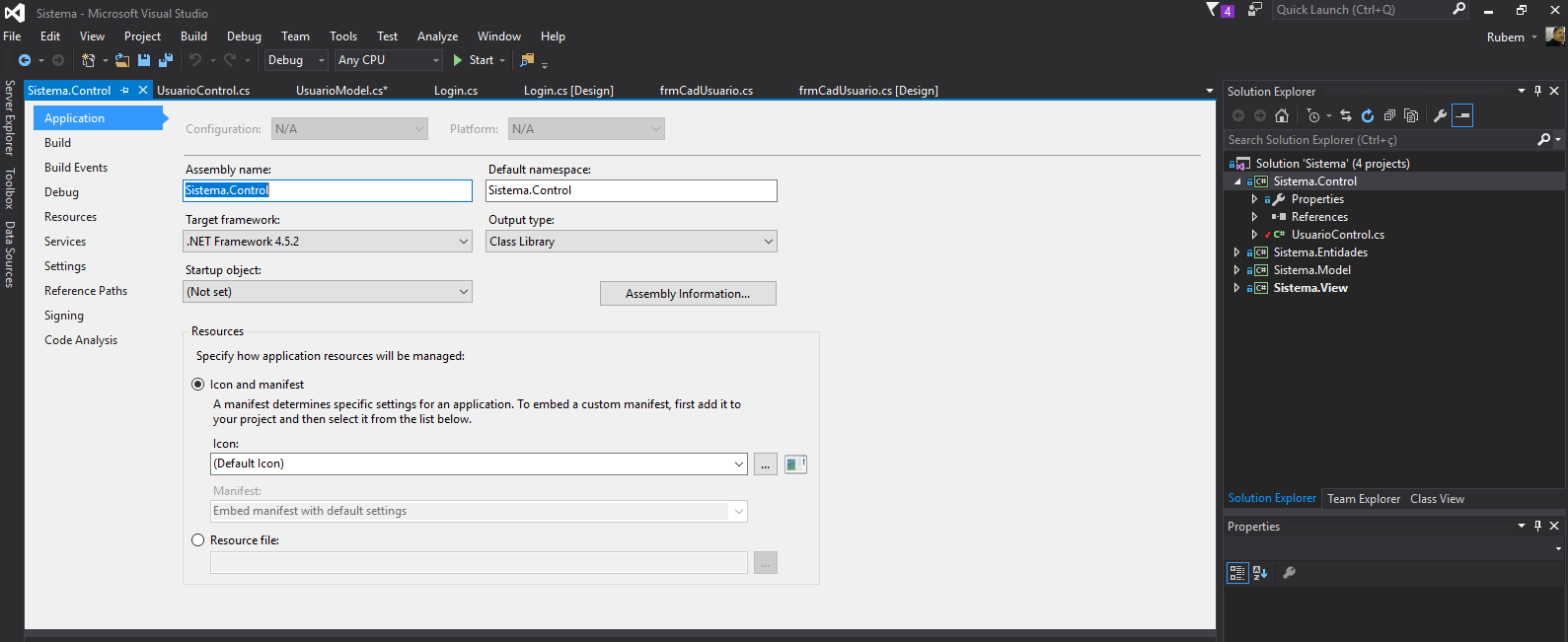


Figura 118. Propriedades da camada Sistema.Control

Na opção de Settings não consta nenhuma configuração. A mensagem diz “Este projeto não contém um arquivo de configurações padrão. Clique aqui para criar um”. Clique em **This Project does not contain a defaut settings file. Click here to create one.**

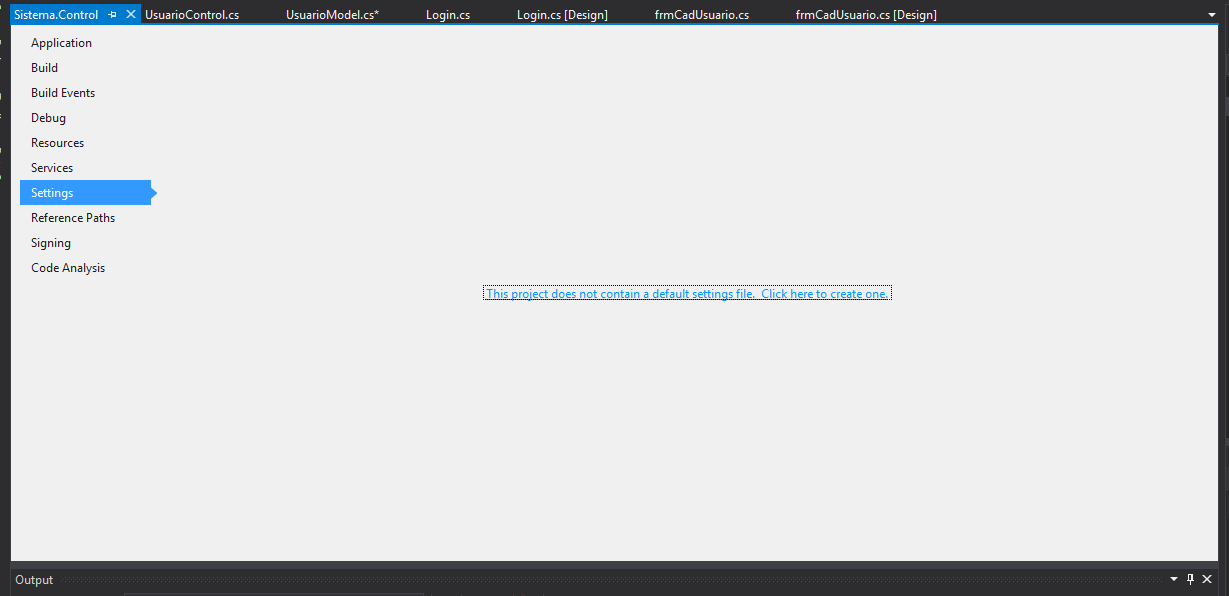


Figura 119. Settings – This Project does not contain ...

Irá aparecer essa tela:

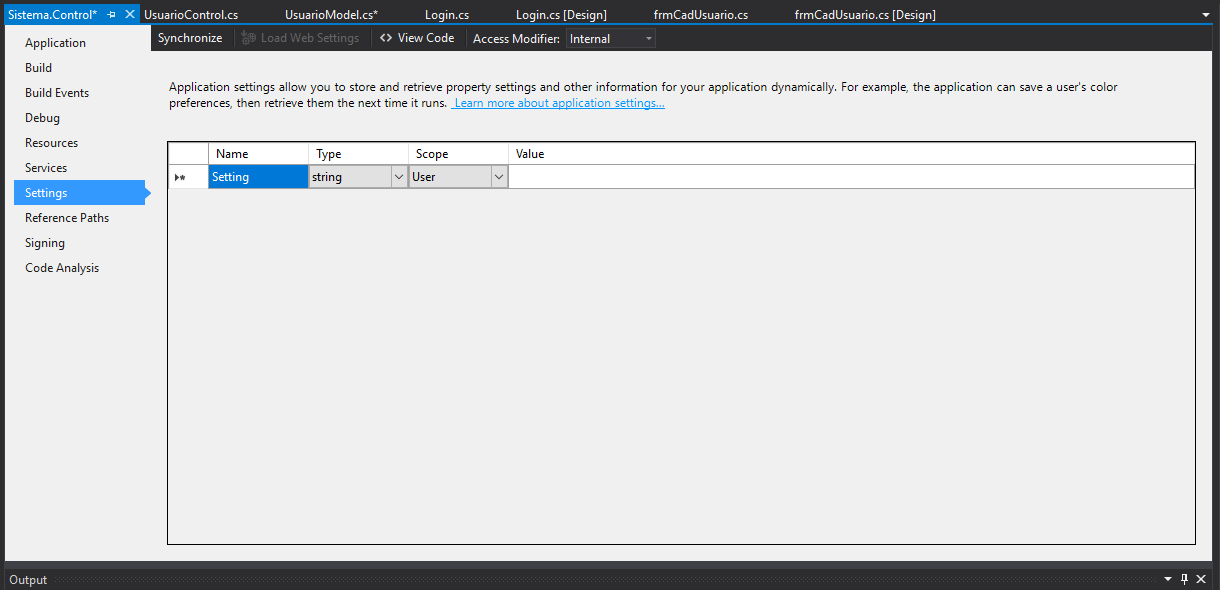


Figura 120. Painel de configurações

Em **Name**, selecione Setting

Em **Type**, selecione (Connection string)

Em **Scope**, selecione Aplication

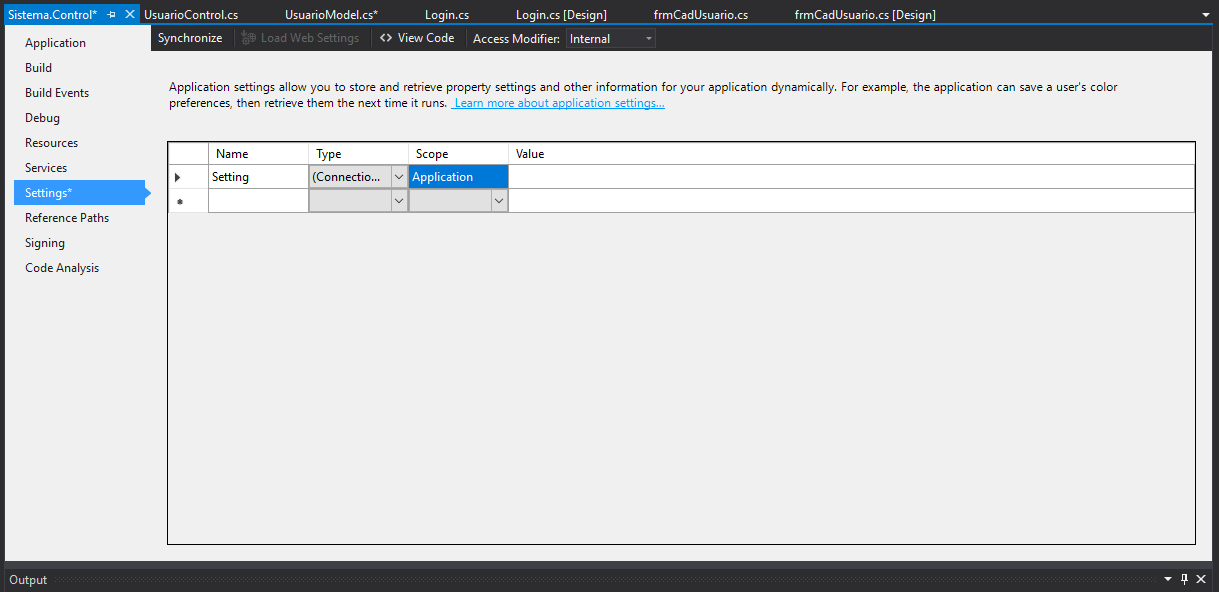


Figura 121. Configurando a conexão

Podemos renomear para banco conforme figura abaixo

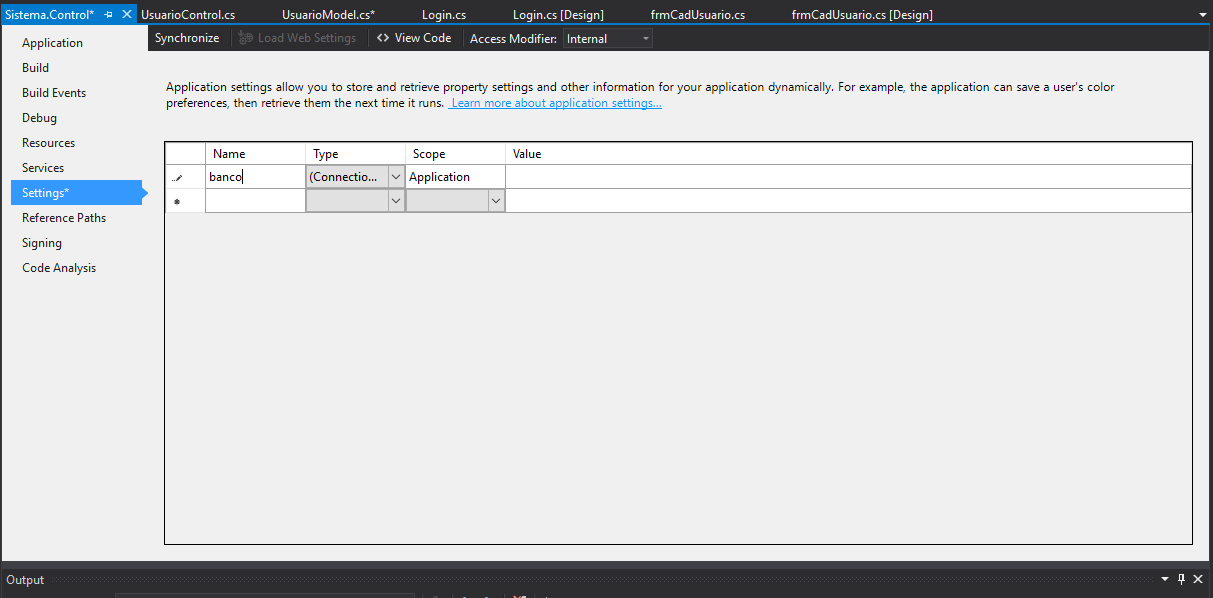
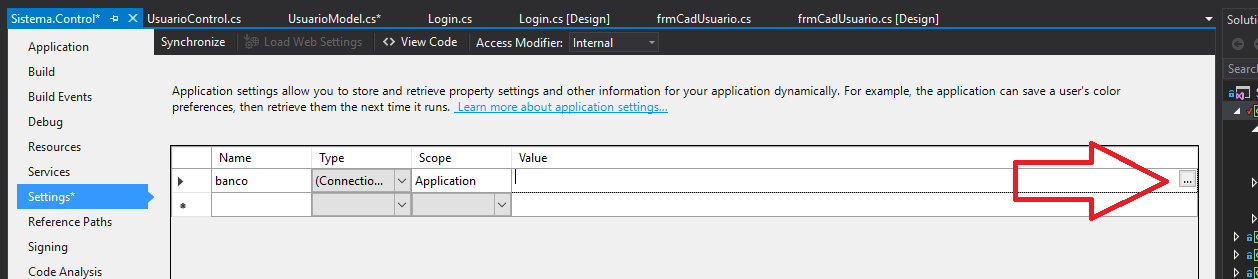


Figura 122. Name alterado para banco

Clique nos três pontinhos

  
Figura 123. Abrindo painel de configurações

Irá aparecer um painel parecido com o de conexão com o Servidor de Banco de Dados.

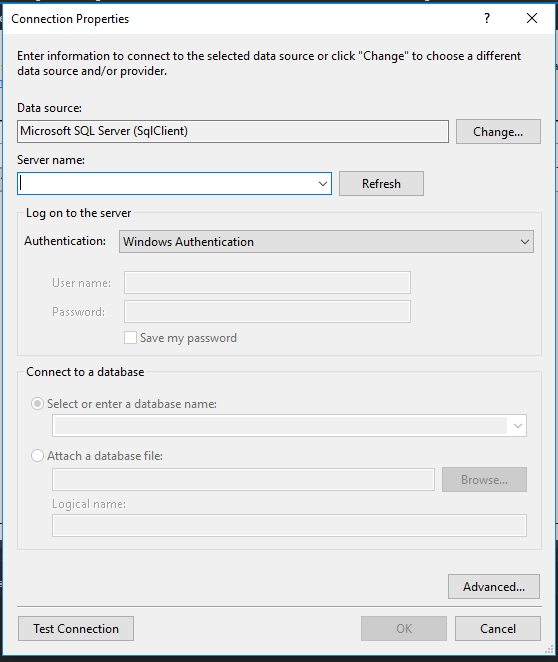


Figura 124. Propriedades de Conexão

O Server name será o mesmo que aparece na conexão com o banco

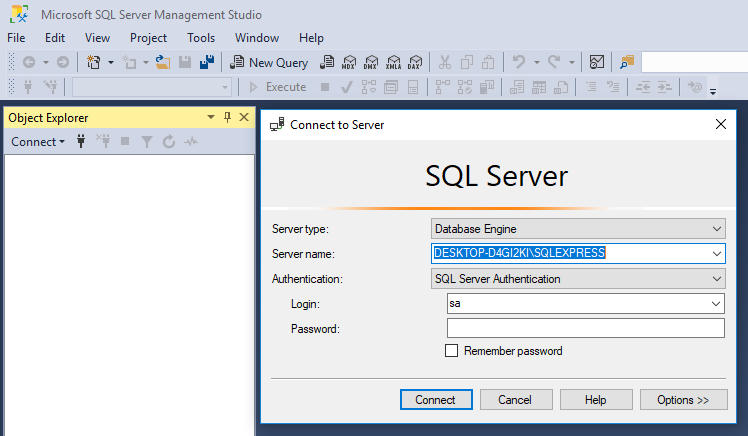


Figura 125. Conexão com o SQL Server

Copie o Server name da tela de Connect to Server e cole no painel de conexão. Selecione seu banco de dados e clique em OK.

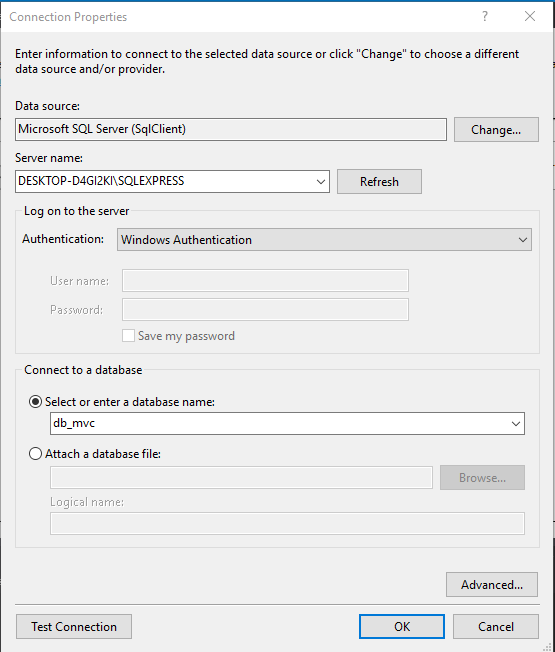


Figura 126. Connection proprieties configurado

Irá ficar assim:



Figura 127. Connection String – Cadeia de conexão preenchida.

Podemos agora fechar o Sistema.Control e clicar em Sim para salvar as alterações.

Um arquivo .xml foi criado com o nome app.config e contém informações da cadeia de conexão.

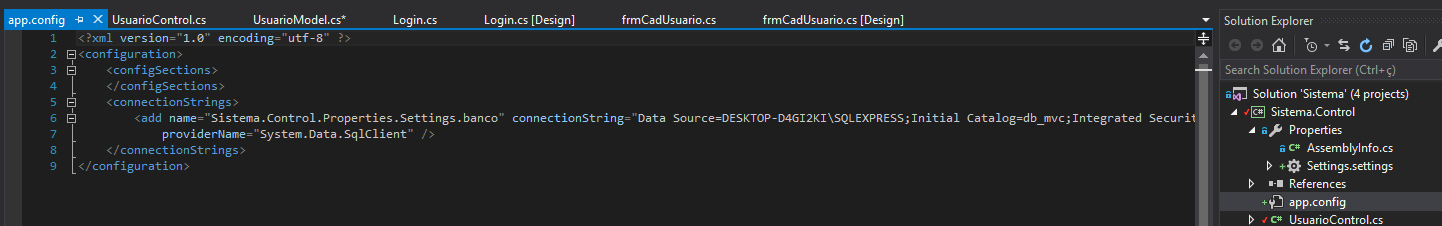


Figura 128. Xml com informações da cadeia de conexão

Se você estiver utilizando autenticação do SQL Server, o final da ConnectionString estará diferente.

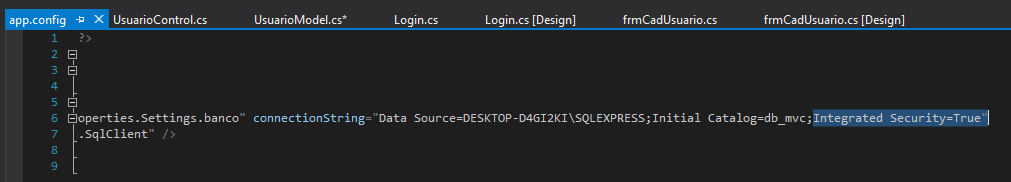


Figura 129. Configuração de senha com autenticação do Windows

Ficará parecido com isso

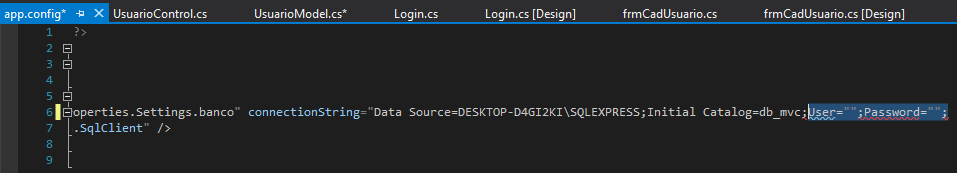


Figura 130. Cadeia de conexão com usuário e senha

Agora vamos apontar para a conexão criada para que a inserção possa ser realizada. Para isso, vamos pegar o objeto con e associar à ConnectionString já criada.

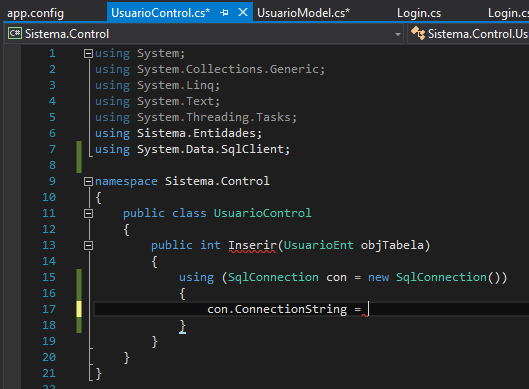


Figura 131. Objeto con sendo associada a ConnectionString

Podemos copiar a associação do xml criado

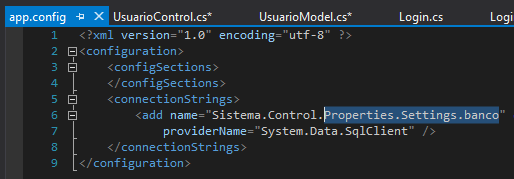


Figura 132. Associação criada no arquivo xml

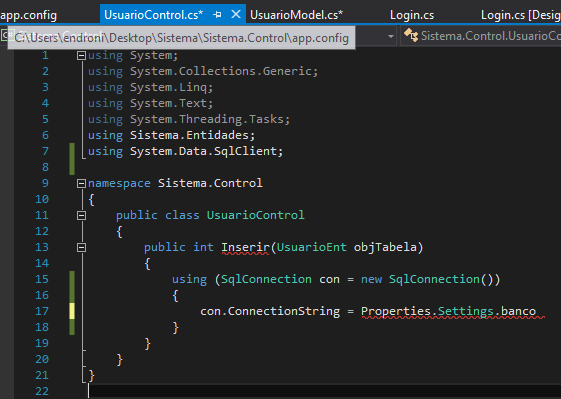


Figura 133. Colando a associação criada

Como é uma conexão padrão, precisamos acrescentar o default.

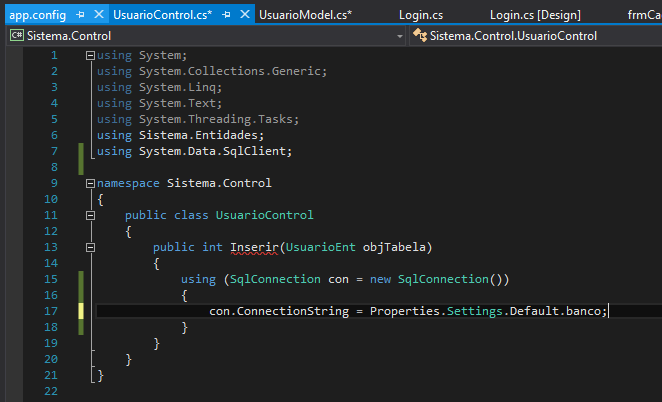


Figura 134. Connection String criada e relacionada

## Inserção

Adicione a bliblioteca System.Data

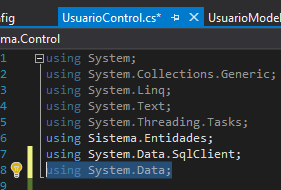


Figura 132. Adicionando biblioteca System.Data

Adicionando os objetos de inserção

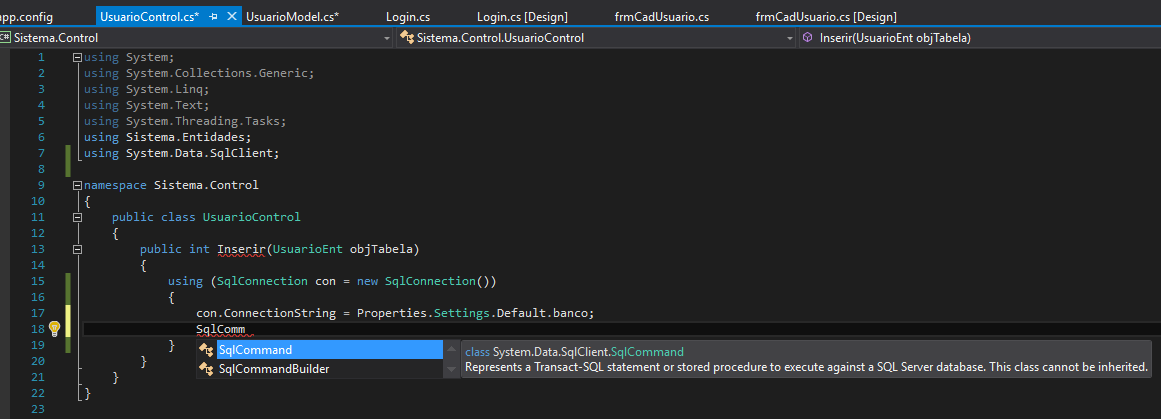


Figura 135. Objeto SqlCommand

Variável instanciada abaixo



Figura 136. Variável cn

Passando o comando CommandType

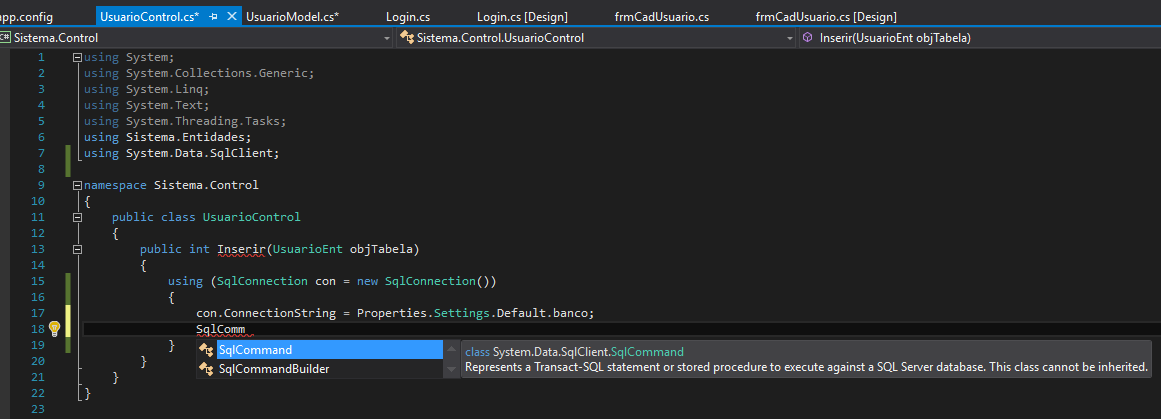


Figura 137. CommandType sendo inserido

Como o **cn** precisa ser do tipo texto, precisamos informar atribuindo o cn.CommandType ao CommandType.Text.

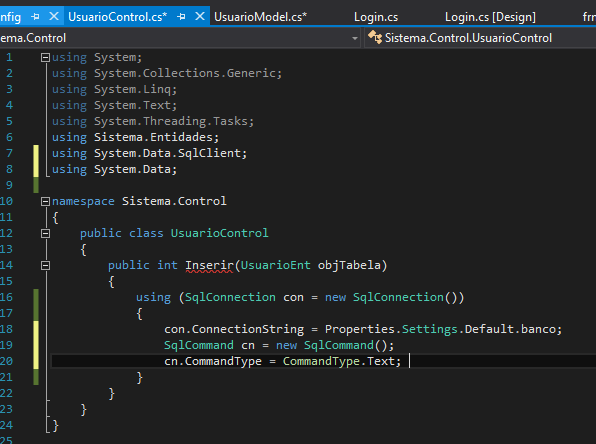


Figura 138. CommandType.Text

Vamos iniciar, abrir a conexão agora com a linha con.Open()

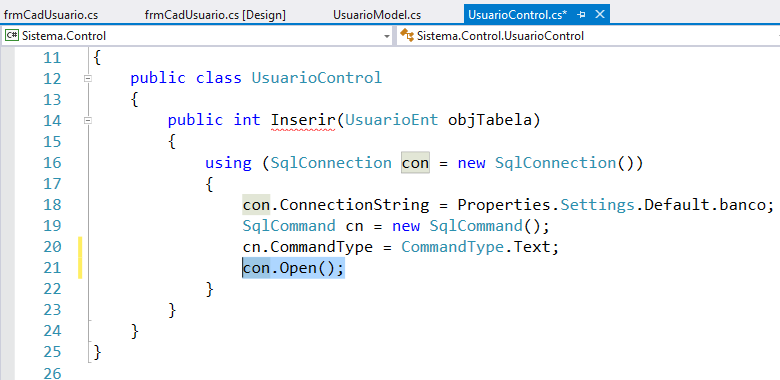


Figura 139. con.Open() inserida

Agora precisamos passar o comando de inserção

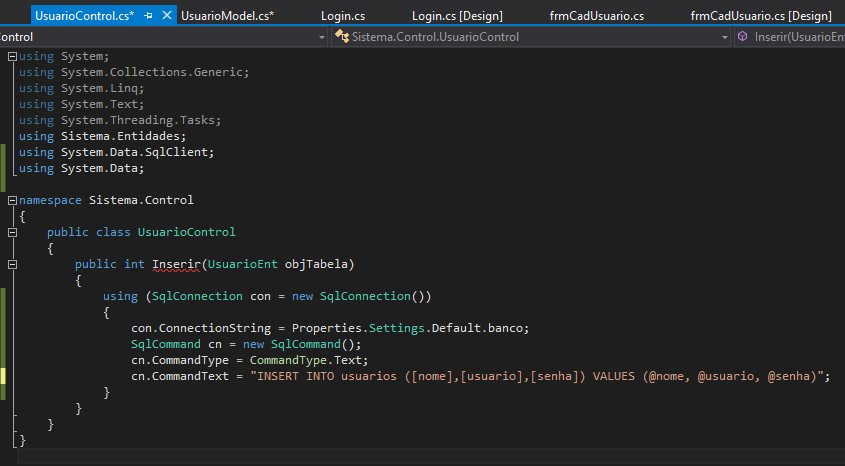


Figura 139. Comando de inserção

Precisamos criar agora os parâmetros, fazer referências para cada campo.

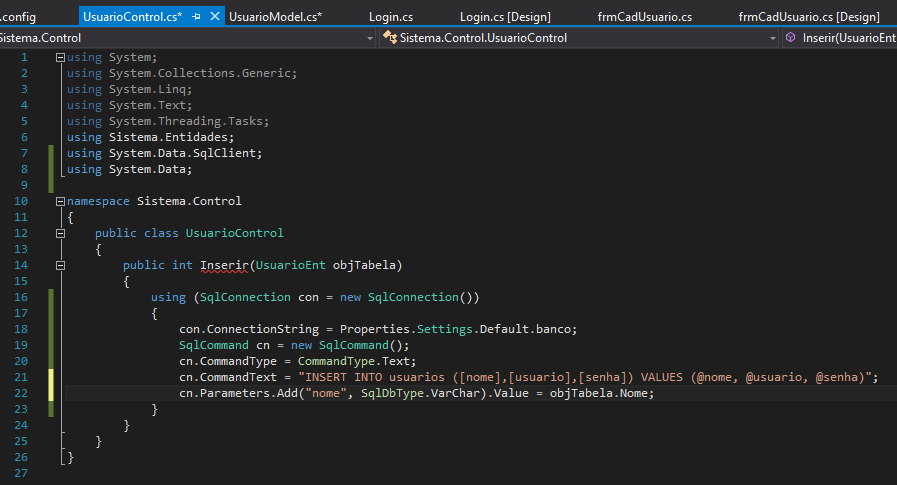
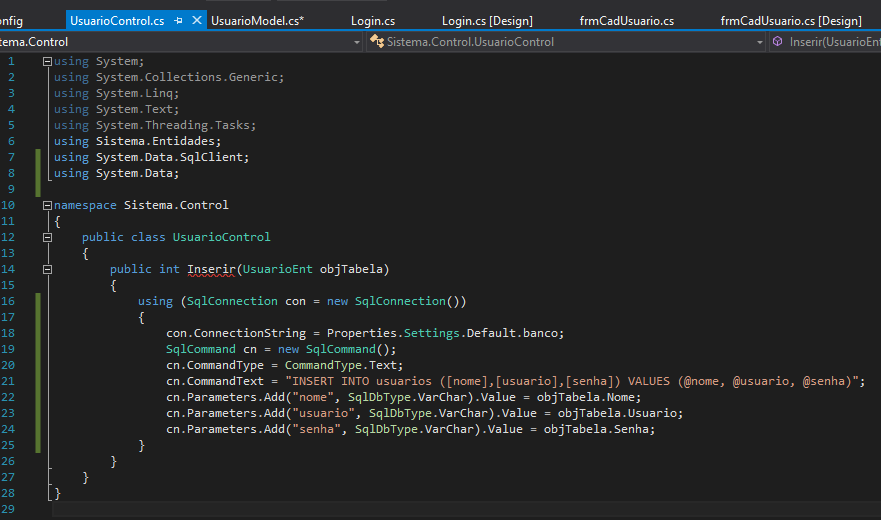


Figura 140. Criação do Parâmetro para o campo nome

Agora é só acrescentar o Parâmetro dos outros campos



Figua 141. Parâmetros criados

Vamos iniciar também o SQL Command

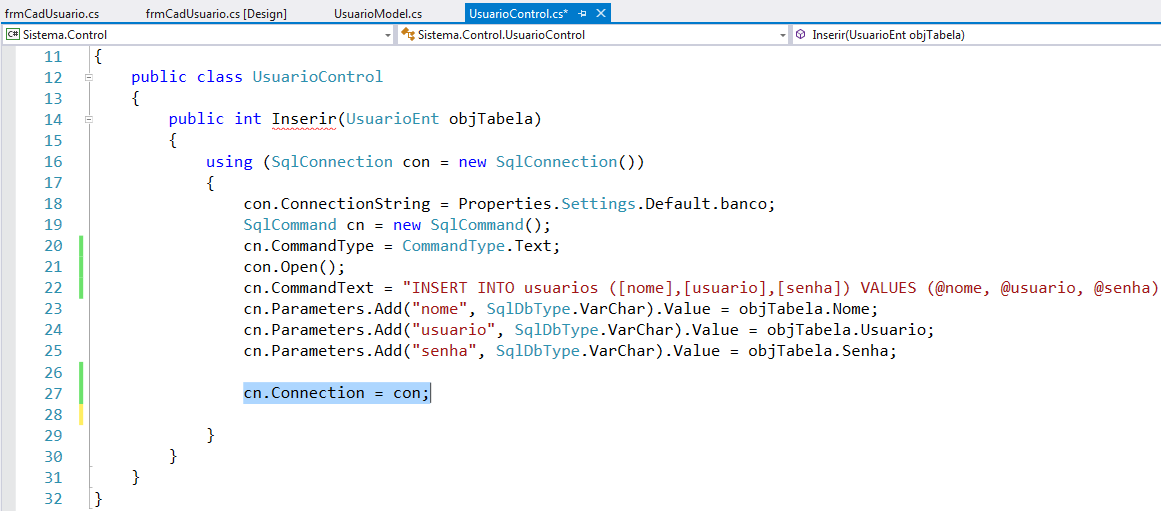


Figura 142. Inicializando o SqlConnection

Agora precisamos dar uma função que executa o comando. ExecuteNonQuery

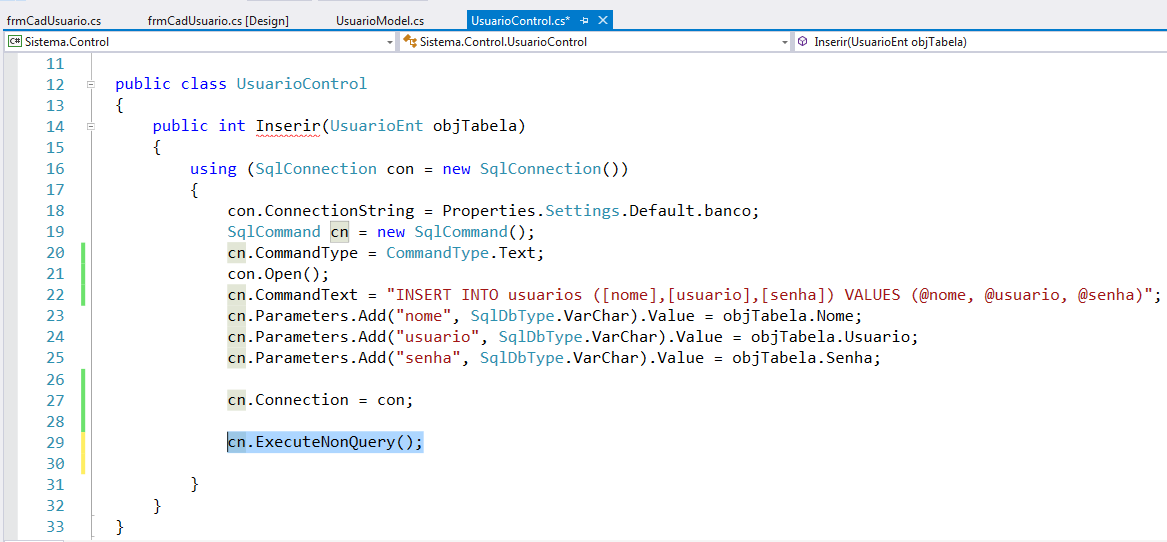


Figura 142. Executando a função com o ExecuteNonQuery

Obs. Para fazer o Inserir parar de dar erro. Precisamos dar um retorno para ele

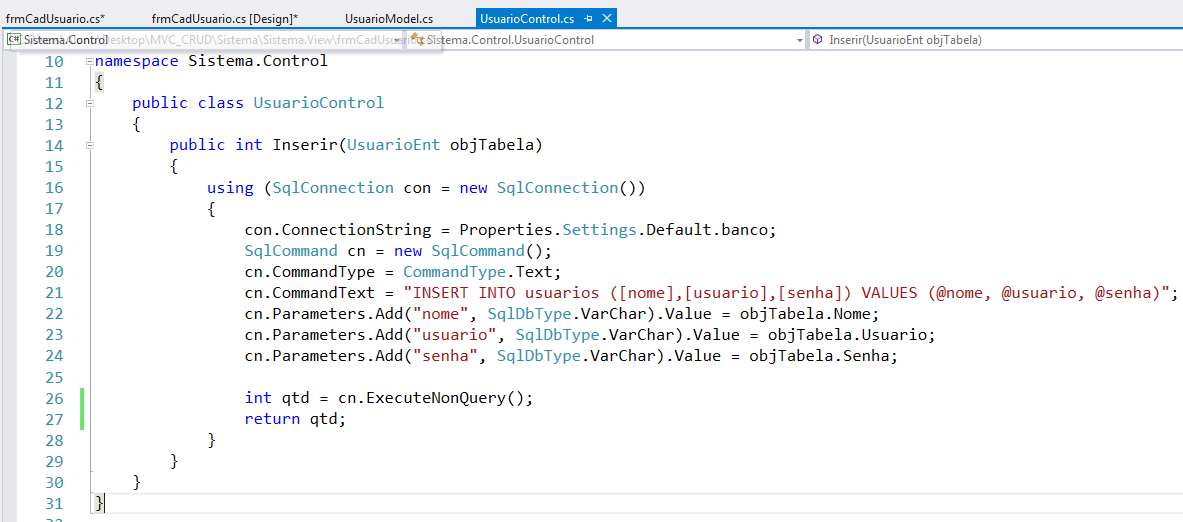


Figura 143. return qtd – Dando um retorno para o objeto

Agora vamos exibir o retorno com o Console.Write

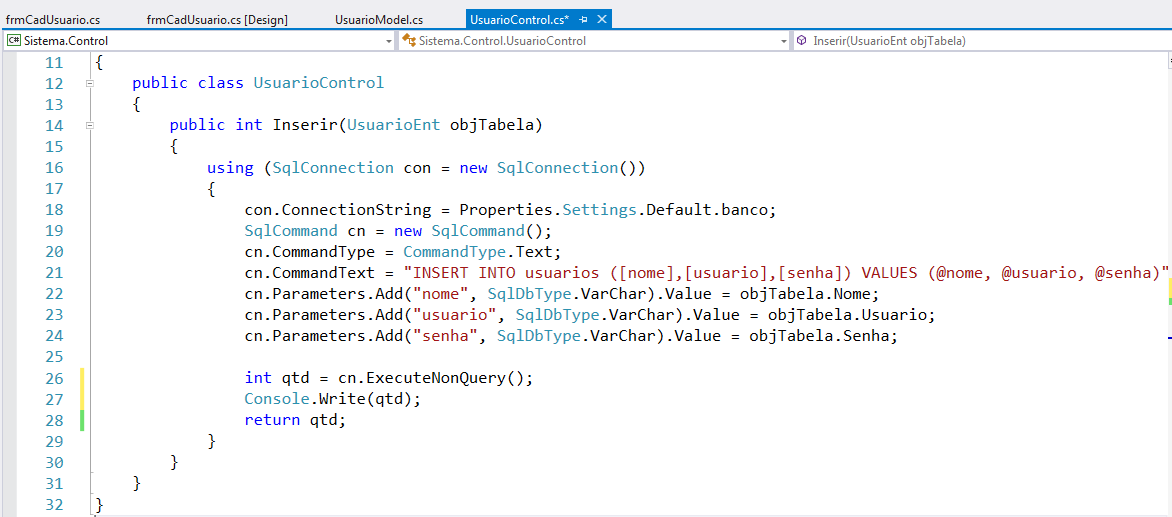


Figura 144. Console.Write

Precisamos criar uma variável que irá receber o retorno do banco. Esse retorno será um número de linhas afetadas. 1 para ok ou 0 para falha.

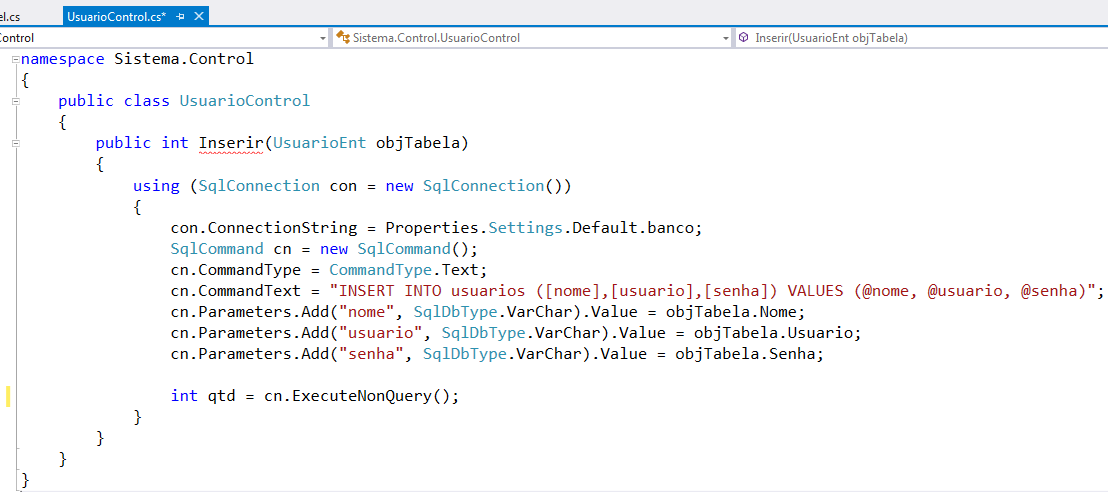


Figura 143. Variável **qtd** irá receber retorno de linhas afetadas do banco

Bem. No **frmCadUsuaruio.cs** nós já estávamos aguardando receber um valor do tipo inteiro na variável x.



Figura 144. Variável x do tipo inteiro

Se esse valor for maior que **0**, quer dizer que o registro foi inserido. Vamos inserir duas mensagens para exibir os resultados.

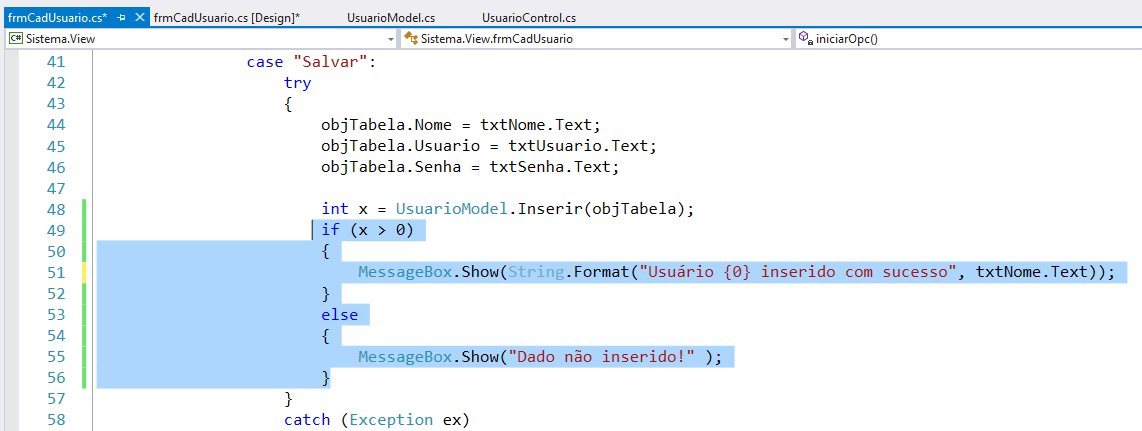


Figura 145. Mensagens de inserido ou não

## Teste de Inserção

Iniciando o serviço com Ctrl + F5 > Login > Novo

Inserindo dados e clicando em Salvar

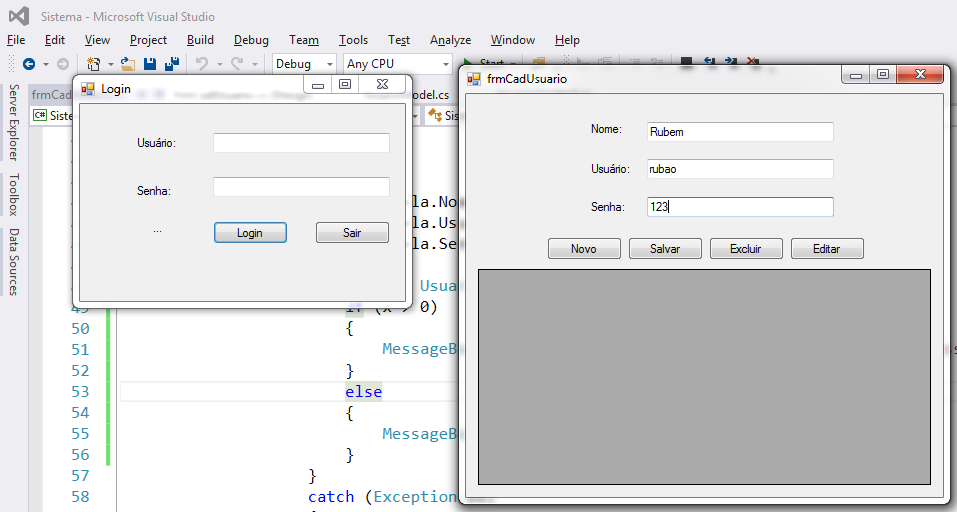


Figura 150. Inserindo dados