CRUD com Arquitetura MVC

fatec – senai centro | Rua Caetés 741, Belo Horizonte

quatro camadas com c# e sql server

Rubem Cândido dos Santos

2019

Sumário

[Introdução 2](#_Toc23690878)

[Criando o Banco de Dados 2](#_Toc23690879)

[Criando o Projeto 3](#_Toc23690880)

[Preparando os Formulários 4](#_Toc23690881)

[Formulário Login 4](#_Toc23690882)

[Formulário Cadastrar 7](#_Toc23690883)

[Conectando o Visual Studio com o SQL Server 8](#_Toc23690884)

[Criando classes para Conexão e Cadastro 11](#_Toc23690885)

[Classe Conexao 11](#_Toc23690886)

[Classe Cadastrar 12](#_Toc23690887)

[Codificando as Classes 14](#_Toc23690888)

[Classe Conexao código 14](#_Toc23690889)

[SqlConnection 14](#_Toc23690890)

[Construtor 16](#_Toc23690891)

[Método Conectar 19](#_Toc23690892)

[Método Desconectar 19](#_Toc23690893)

[Classe Cadastrar 20](#_Toc23690894)

[SqlCommand 21](#_Toc23690895)

[Criando Parâmetros 24](#_Toc23690896)

[Chamando a conexão 25](#_Toc23690897)

[ExecuteNonQuery 26](#_Toc23690898)

[Configurando o botão Salvar 27](#_Toc23690899)

[Github 29](#_Toc23690900)

[Conclusão 30](#_Toc23690901)

# Introdução

Pretendo com esse pequeno tutorial, demostrar uma das várias formas de se encriptar dados.

# Criando o Banco de Dados

Abra o Management e se conect no banco

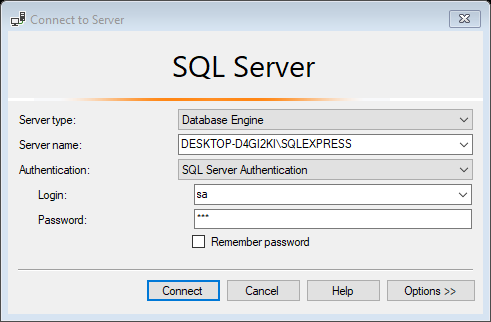


Figura 1. Connect to Server

Abra uma nova Consulta e vamos criar um banco chamado db\_encript\_login e tabela tb\_usuario com o script abaixo:

CREATE DATABASE db\_encrypt\_login;

GO

USE db\_encrypt\_login;

GO

CREATE TABLE tb\_usuario (

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

nome NVARCHAR(60),

email NVARCHAR(70),

senha NVARCHAR(150));

GO

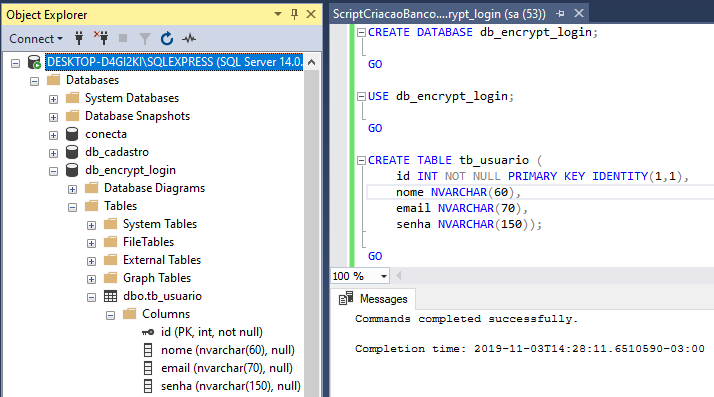


Figura 2. Script de criação do banco

# Criando o Projeto

Vamos criar um projeto chamado EncryptLogin.

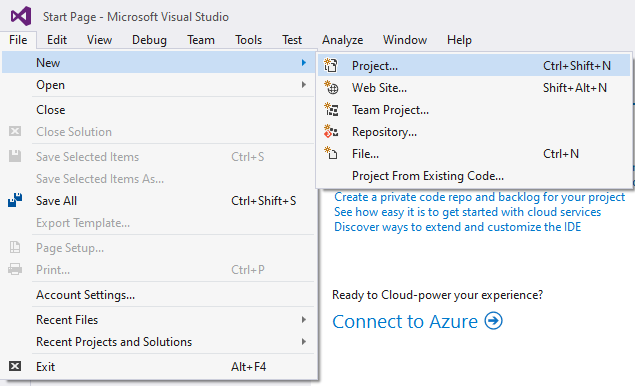


Figura 3. Criando o projeto

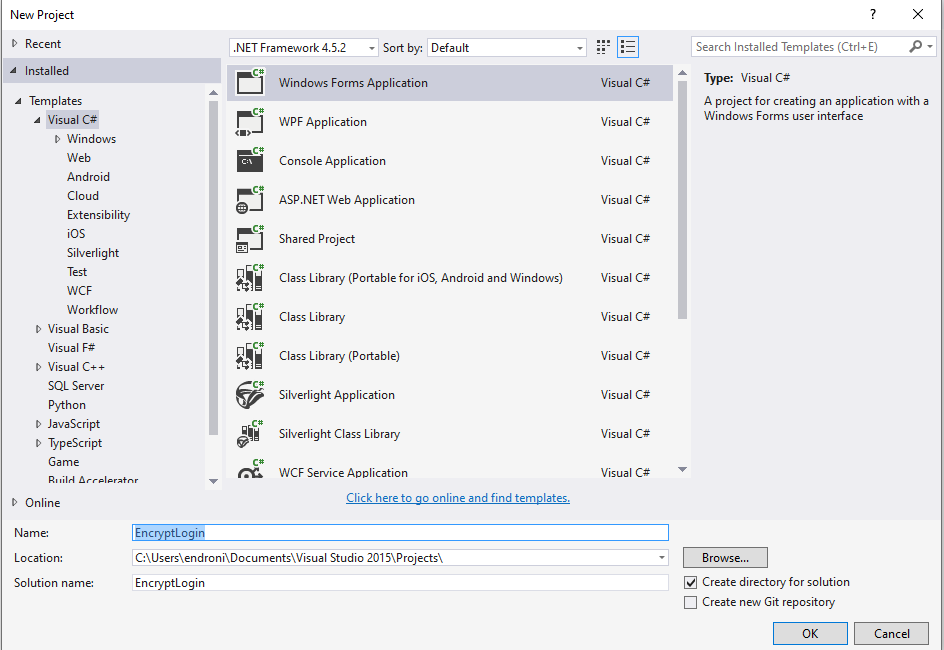


Figura 4. Windows Form Application EncryptLogin

# Preparando os Formulários

Vamos criar dois formulários. Um para Login e outro para cadastro

# Formulário Login

Vamos inserir no forumário duas labl’s, dois textbox, três botões e renomear o formulário

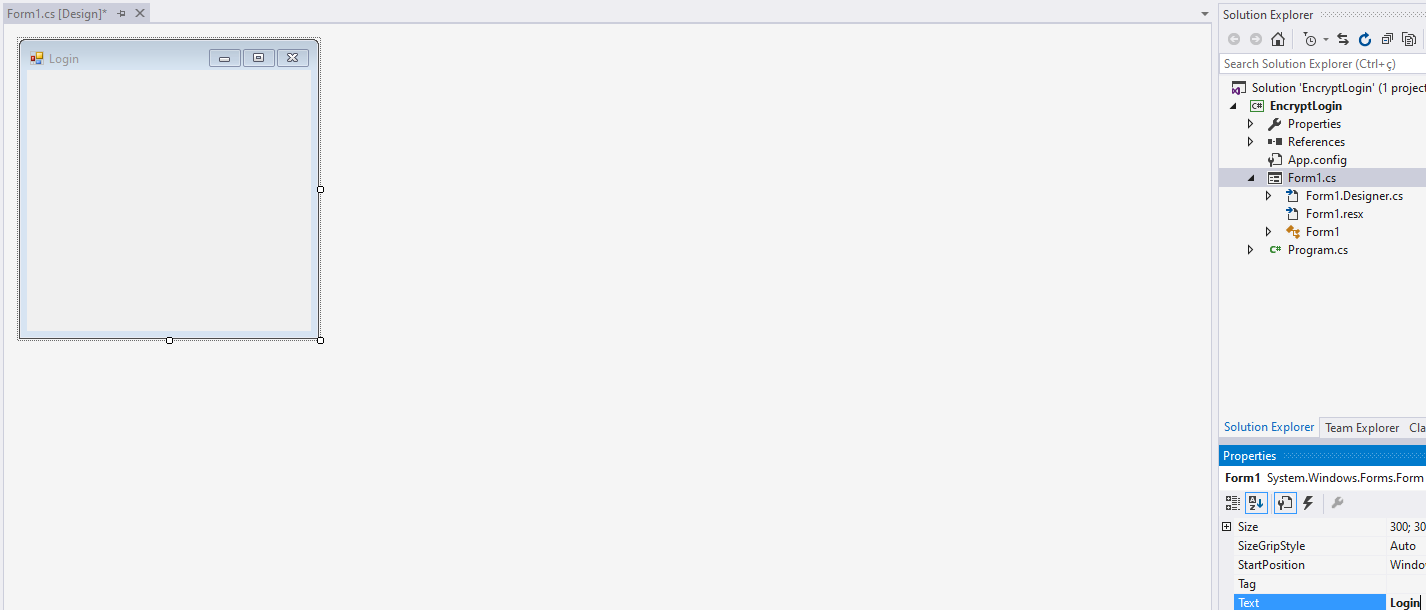


Figura 5. Formulário renomeado para Login

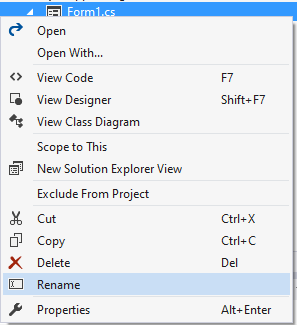


Figura 6. Renomeando a classe do Form1.cs

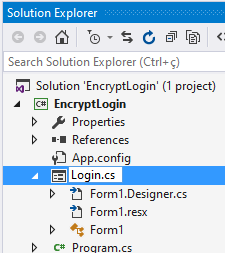


Figura 7. Login.cs

Clique em yes para aceite a alteração

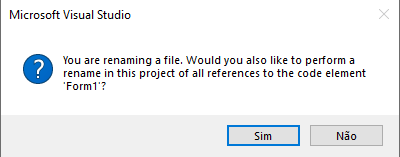


Figura 8. Permitindo que o código seja renomeado

Adicione duas Label’s

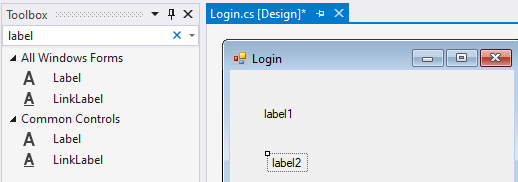


Figura 9. Label’s adicionadas

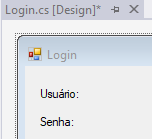


Figura 10. Label’s renomeadas

Elementos inseridos

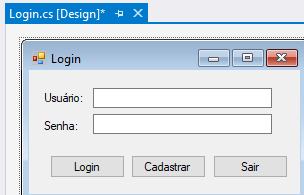


Figura 11. Formulário concluído

Duas labl’s = Usuário e Senha

Dois TextBox = txtUsuario e txtSenha

Três Button = btnLogin, btnCadastrar e btnSair

# Formulário Cadastrar

Clique com o botão direito sobre a solução EncryptLogin > Add > New Item

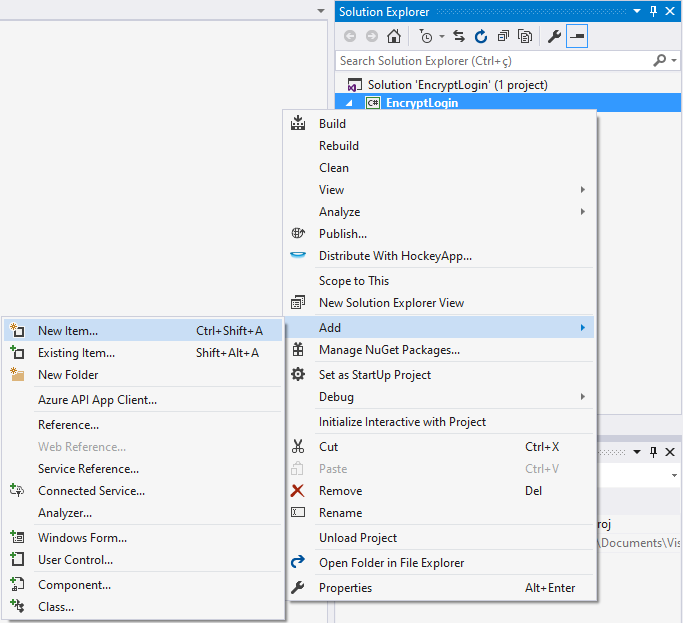


Figura 12 Menu New Item

Selecione Windows Forms

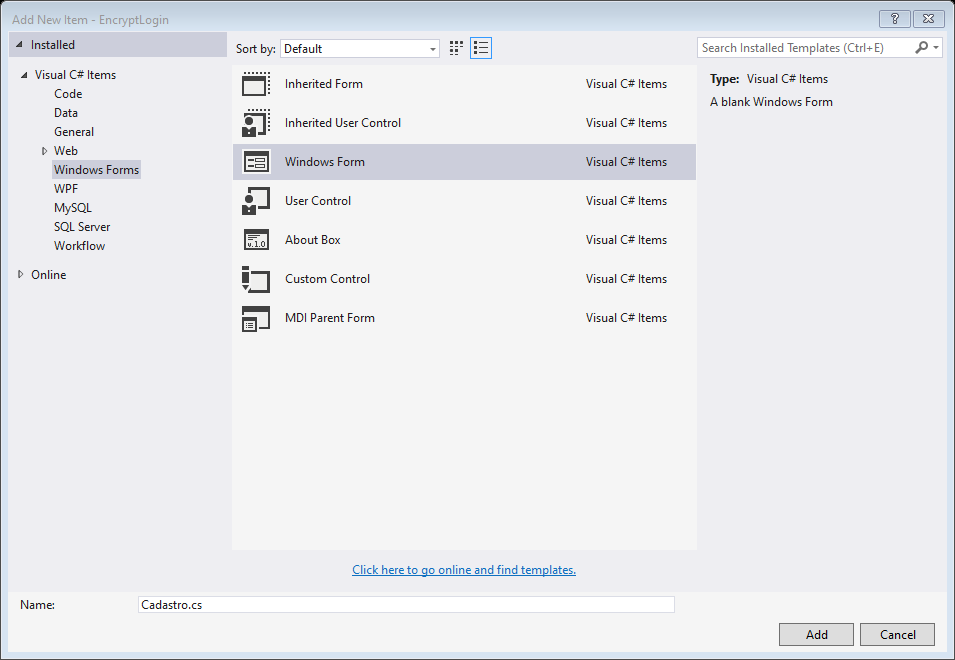


Figura 13. Formulário Cadastro sendo criado

Adicione três label’s, três textbox’s e dois button’s

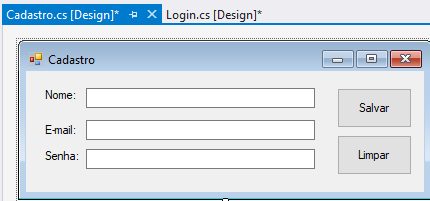


Figura 14. Formulário Cadastrar

Três labl’s = Nome, E-mail e Senha

Três TextBox = txtNome, txtEmail e txtSenha

Três Button = btnLogin, btnCadastrar e btnSair

# Conectando o Visual Studio com o SQL Server

No Visual Studio, clique em Server Explorer e adicione uma nova conexão

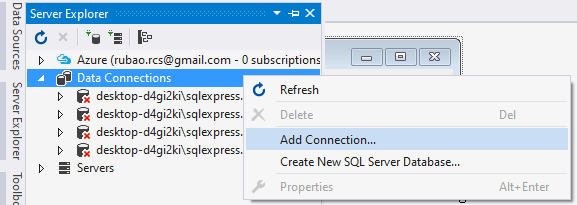


Figura 15. Add Connection

Copie o nome do servidor da conexão com o Banco de Dados

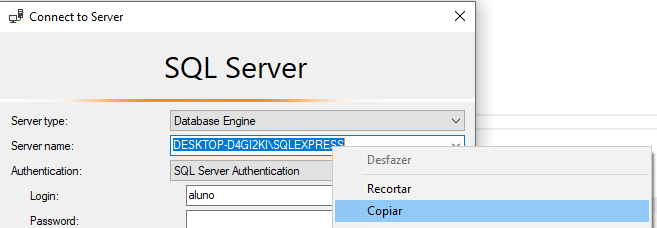


Figura 16. Server name

Cole no nome do servidor na tela de adição de conexão, ajuste o modo de autenticação de acordo com o seu servidor e selecione o banco de dados db\_encrypt\_login

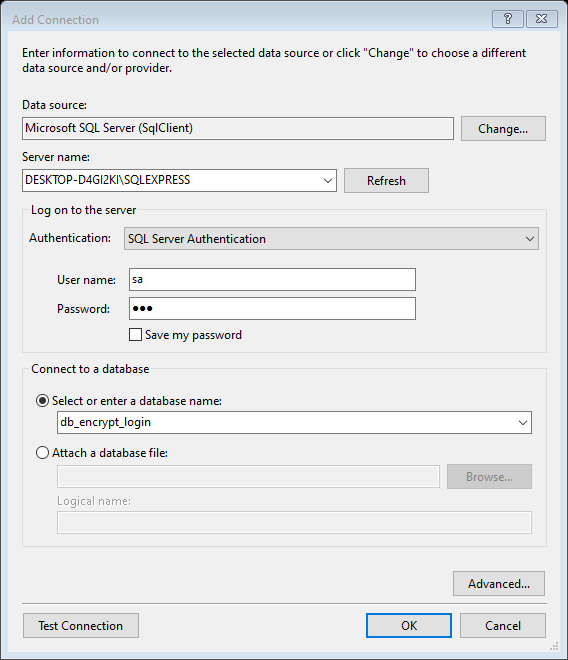


Figura 17. Add Connection

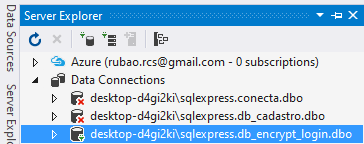


Figura 18. Conexão adicionada

# Criando classes para Conexão e Cadastro

Agora vamos criar uma classe de conexão com o banco de dados.

# Classe Conexao

Clique com o botão direito do mouse sobre a solução EncryptLogin > Add > Class

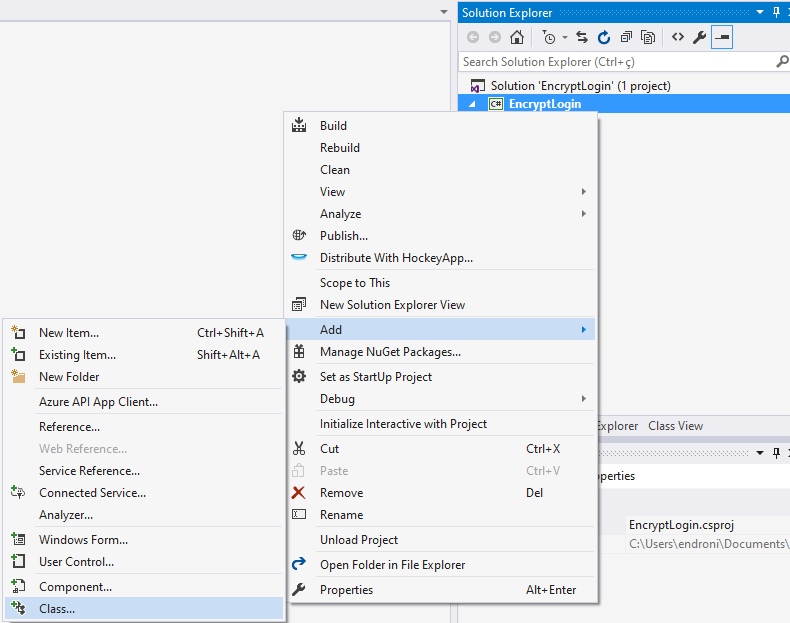


Figura 19. Adicionando uma classe à solução

Iremos chama-la de Conexao

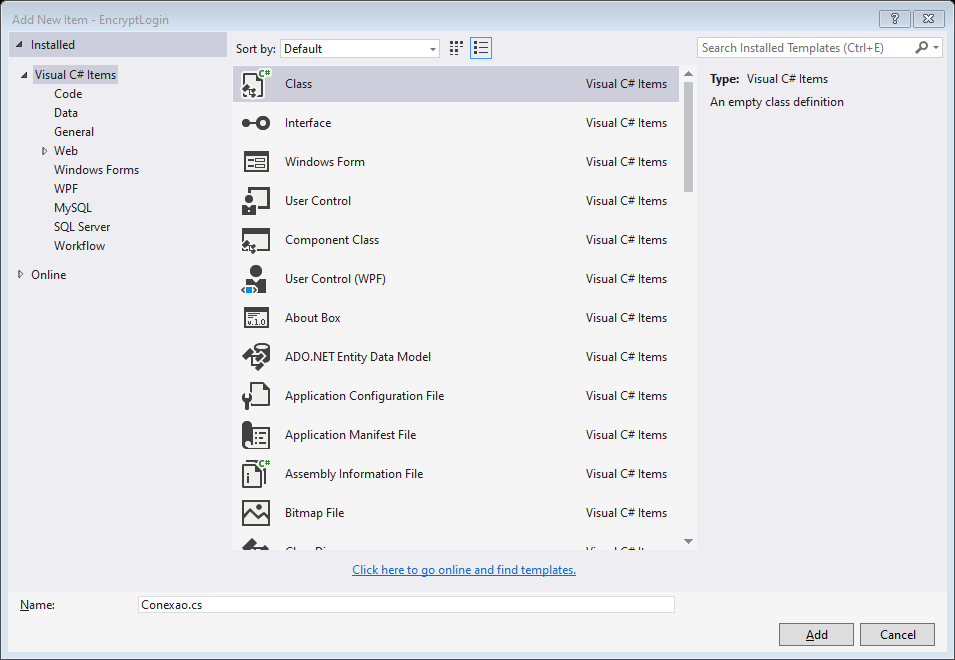


Figura 20. Class Conexao

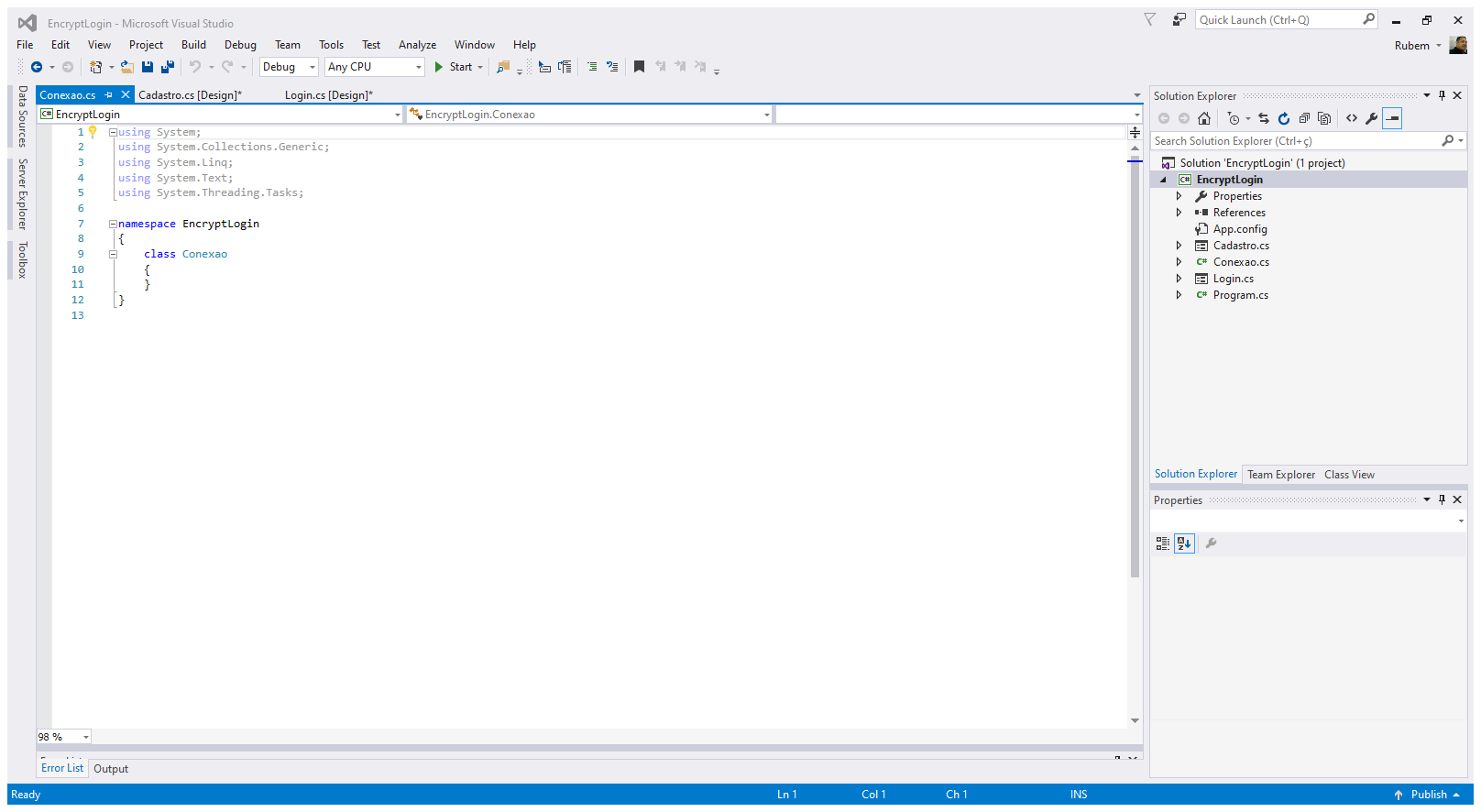


Figura 21. Classe Conexao criada

# Classe Cadastrar

Clique com o botão direito do mouse sobre a solução EncryptLogin > Add > Class

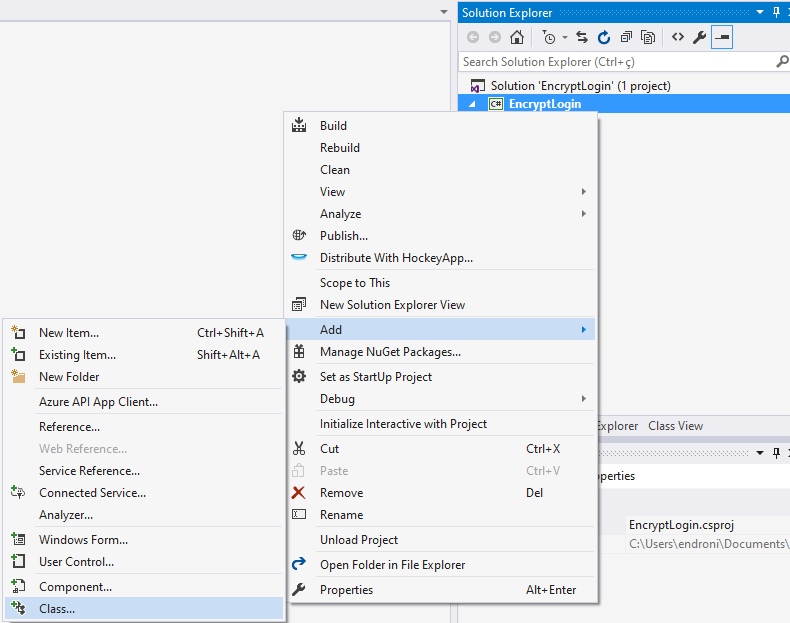


Figura 22. Adicionando uma classe à solução

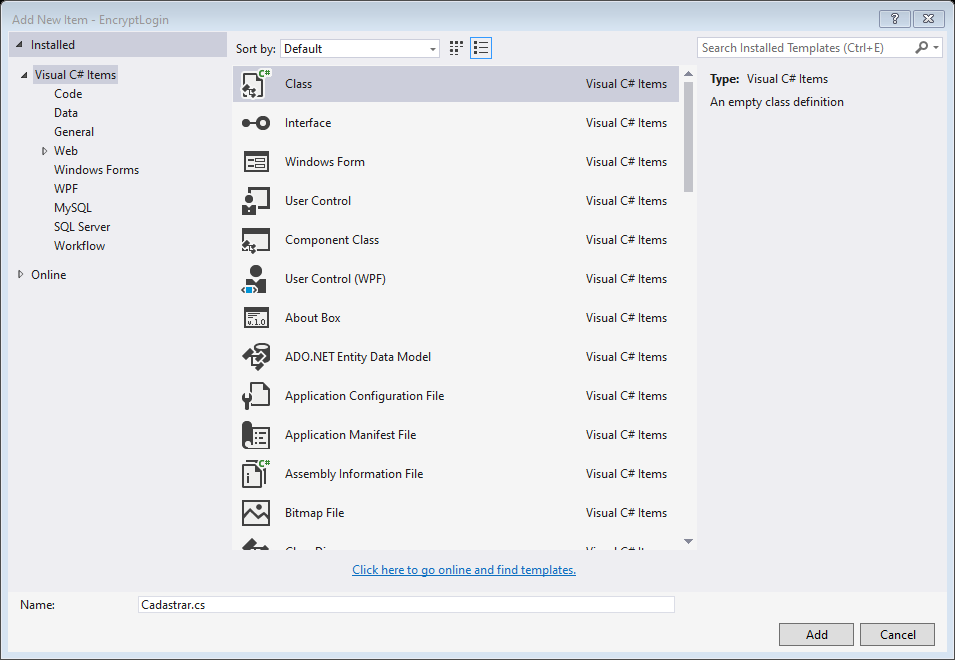


Figura 23. Classe Cadastrar sendo criada

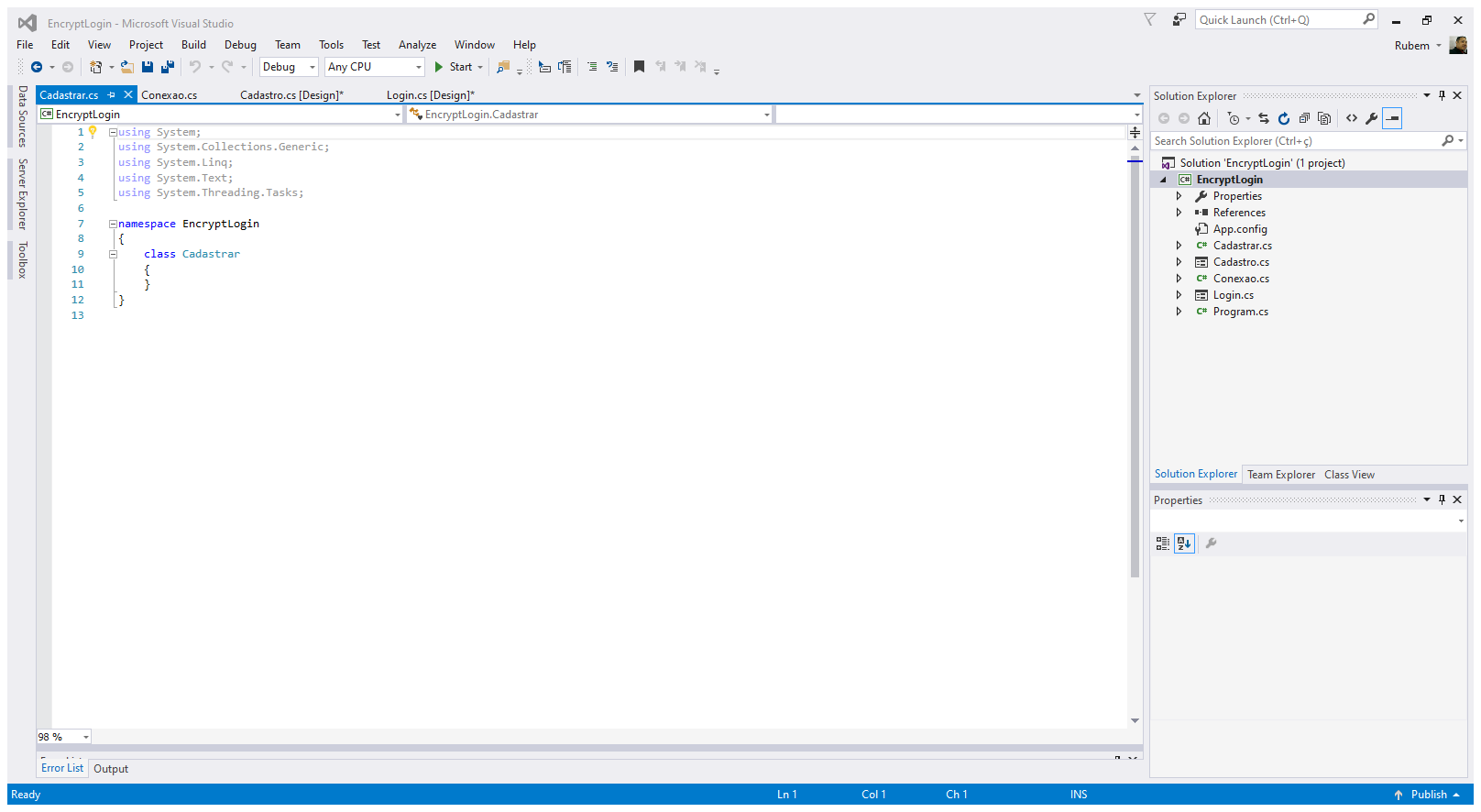


Figura 24. Classe Cadastrar criada

# Codificando as Classes

Vamos começar com a classe Conexao

## Classe Conexao código

Coloque a classe como pública

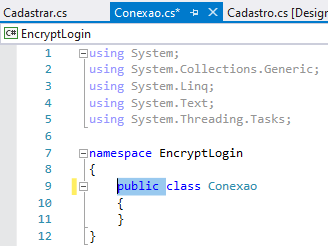


Figura 25. Classe Conexao pública

## SqlConnection

Vamos instanciar

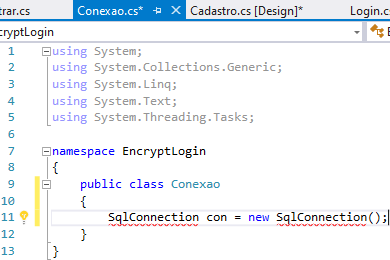


Figura 26. SqlConnection

Como o SqlConnection não foi referenciado, irá apresentar esse erro.

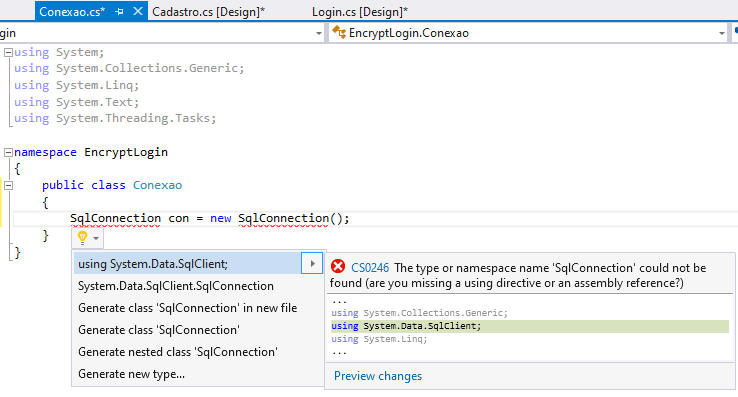


Figura 27. Adicionando referência

O erro irá desaparecer e a bliblioteca System.Data.SqlClient será adicionado.

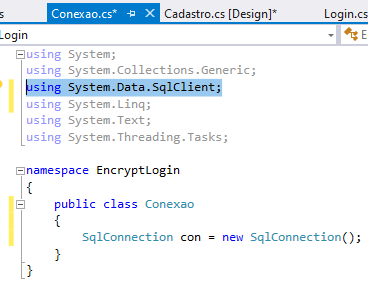


Figura 28. Biblioteca adicionada

## Construtor

Vamos criar uma conexão

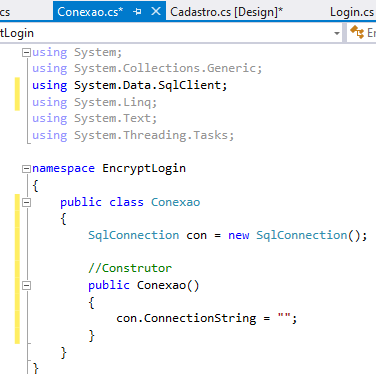


Figura 29. Construtor

Agora precisamos encontrar a string de conexão. No Server Explorer, clique sobre a conexão já criada anteriormente.

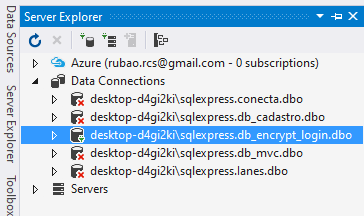


Figura 30. Conexão com o banco

Selecione com o cursor do mouse Connection String das propriedades

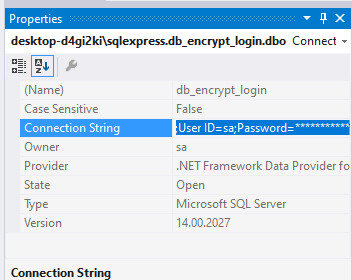


Figura 31. Connection String

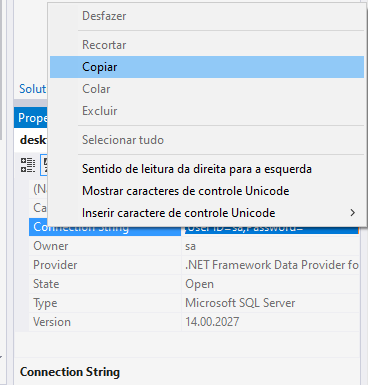


Figura 32. Copiando String de Conexão

Agora cole entre as aspas

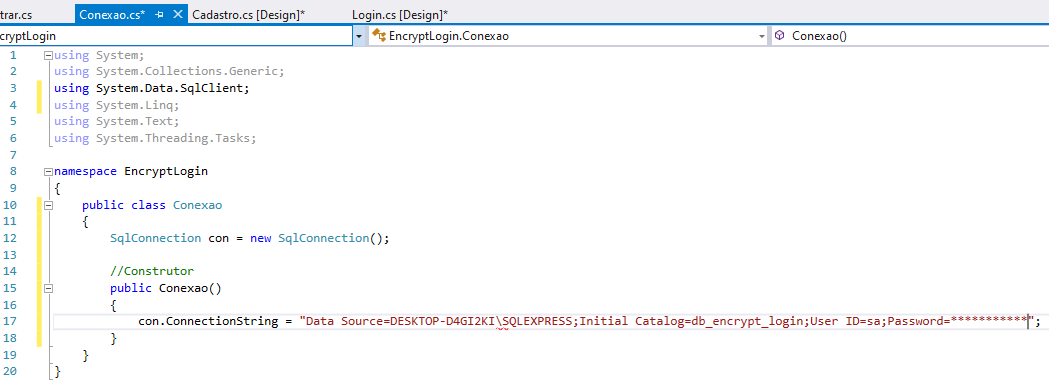


Figura 33. Conexão preenchida

No meu caso, eu me conectei ao banco utilizando uma conta do Sql Server, por isso preciso alterar os asteriscos pela minha senha segura.



Figura 34. Senha alterada

Adicione uma arroba @ antes da string para corrigir o problema do contra barra.

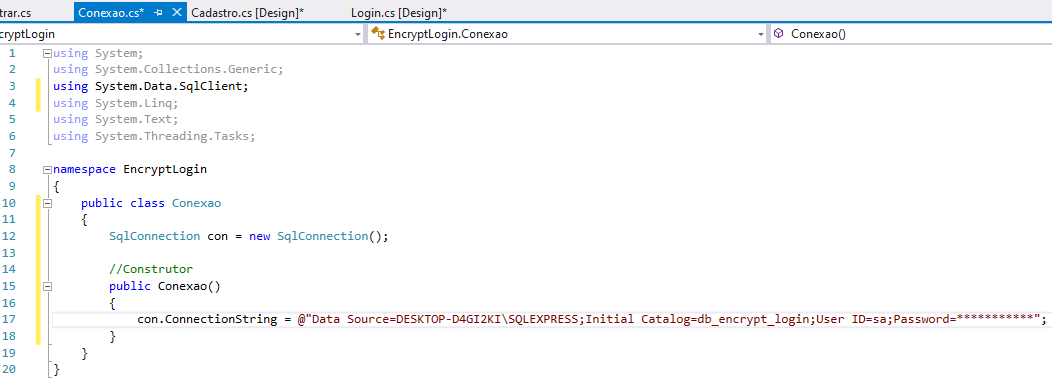


Figura 35. @ adicionado e erro sanado

## Método Conectar

Iremos utilizar o SqlConnection

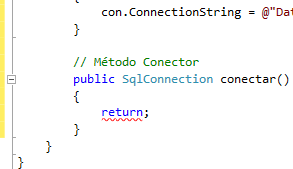


Figura 36. Método Conector

Vamos inserir um if para confirmar que a conexão ainda não está aberta.

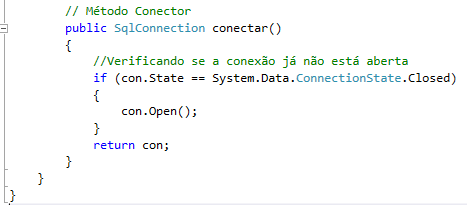


Figura 37. Abrindo conexão com o con.Open()

## Método Desconectar

O método desconectar será parecido

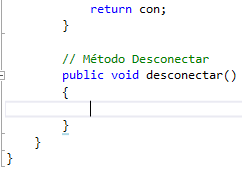


Figura 38. Metodo desconectar

Vamos verificar se a conexão está aberta e fechar.

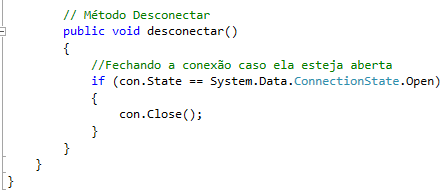


Figura 39. Metodo desconectar

## Classe Cadastrar

Vamos deixar como pública a classe cadastrar

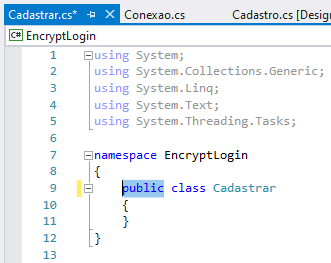


Figura 40. Classe Cadastrar public

## SqlCommand

Agora vamos instanciar as duas classes Cadastrar

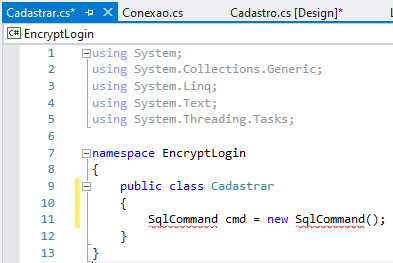


Figura 41. SqlCommand instanciada

Para corrigir o erro, precisamos inserir a referência

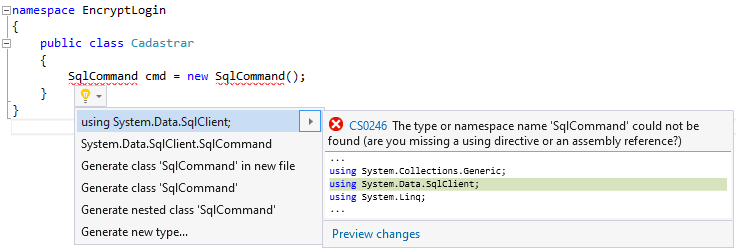


Figura 42. Inserindo referência

A bliblioteca System.Data.SqlClient será instalada

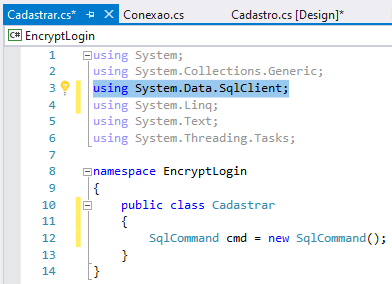


Figura 43. System.Data.SqlClient inserida e erro corrigido

Instancie a classe Conexao também

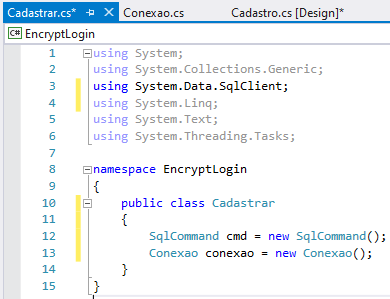


Figura 44. Conexao instanciada

Iremos precisar de uma string para apresentr alguma mensagem de erro, case o cadastro não aconteça.

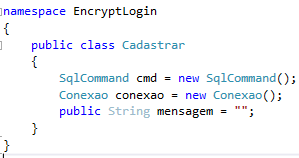


Figura 45. Public String mensagem

Vamos passar as três strings de nome email e senha.

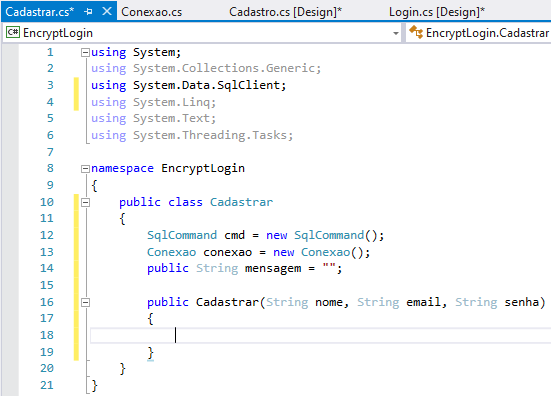


Figura 46. Strings nome, email e senha

Vamos chamar o método SqlCommand para os comandos que podem ser INSERT, UPDAT ou DELETE

Vamos utilizar o CommandText para inserir dados

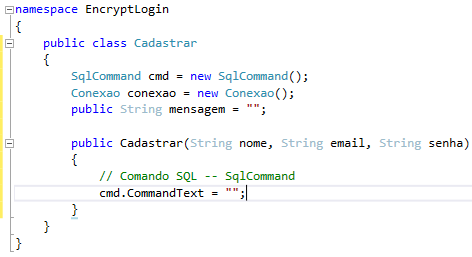


Figura 47. Cmd.CommandText

Vamos passar o comando de inserção dentro do cm.CommandText. Fique atento aos nomes das tabelas e colunas.

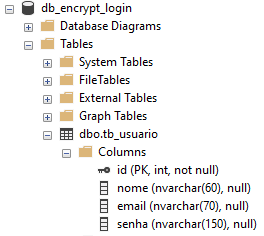


Figura 48. Banco, tabela e suas colulas

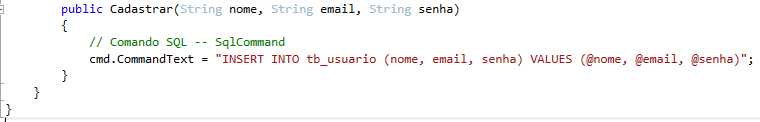


Figura 49. Comando de INSERT

## Criando Parâmetros

Vamos inserir os parâmetros para os três dados do formulário

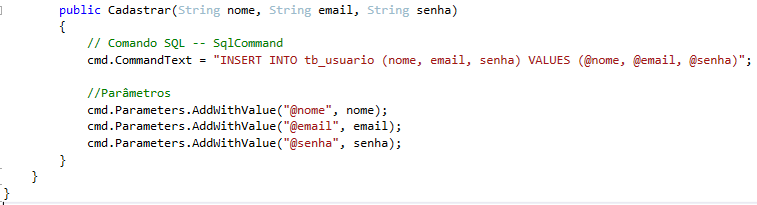


Figura 50. Parâmetros passados

## Chamando a conexão

Vamos utilizar o cmd.Connection e para isso, vamos inserir um try catch para tratar possíveis erros.

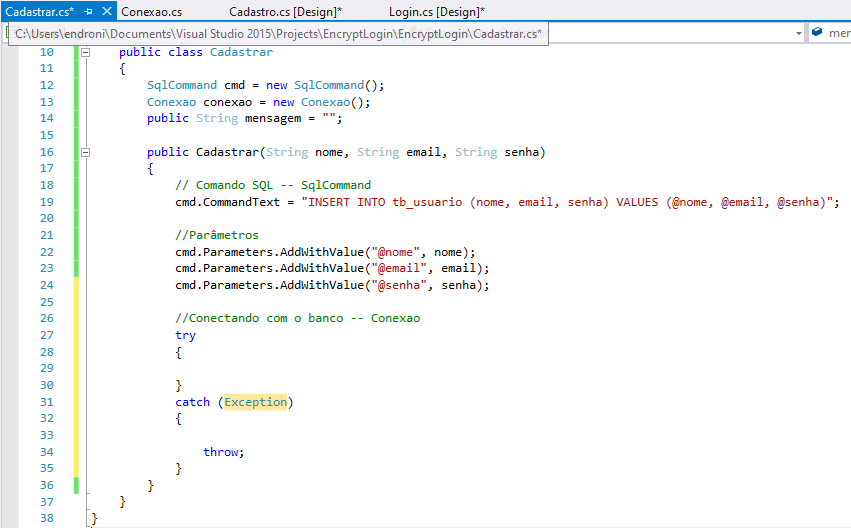


Figura 51. Try catch inserido

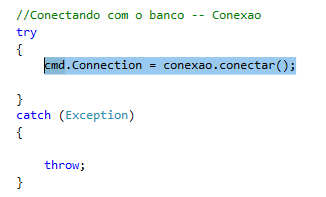


Figura 52. cmd.Connection

## ExecuteNonQuery

Vamos chamar o ExecuteNonQuery para executar o comando

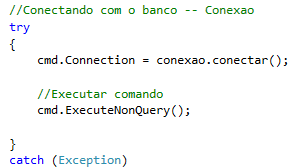


Figura 53. ExecuteNonQuery

Após passar o comando, vamos fechar a conexão

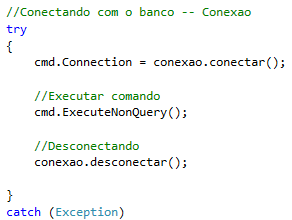


Figura 54. Chamando função desconectar

Agora vamos exibir uma mensagem de sucesso

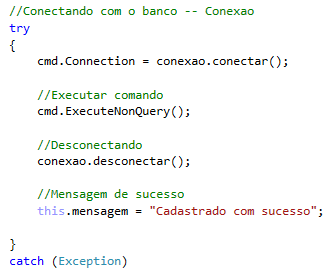


Figura 55. Mensagem de cadastro com sucesso

Precisamos agora tratar a exceção, o erro caso não tenhamos sucesso. Ao invés do **Exception,** vamos utilizar o **SqlException**.

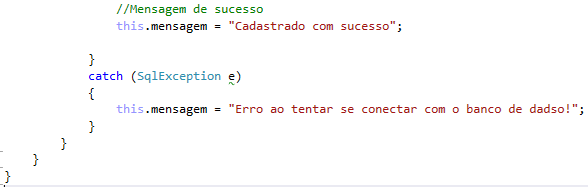


Figura 56. Mensagem de não sucesso.

# Configurando o botão Salvar

No formulário de cadastro, dê um duplo clique no botão de Salvar para abrir o evento e podermos codifica-lo.

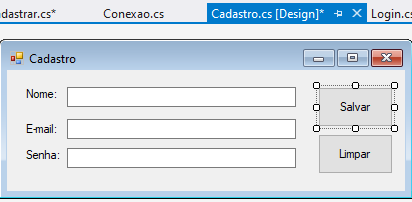


Figura 57. Botão Salvar do formulário Cadastro

O evento para o botão será gerado irá se abrir o código

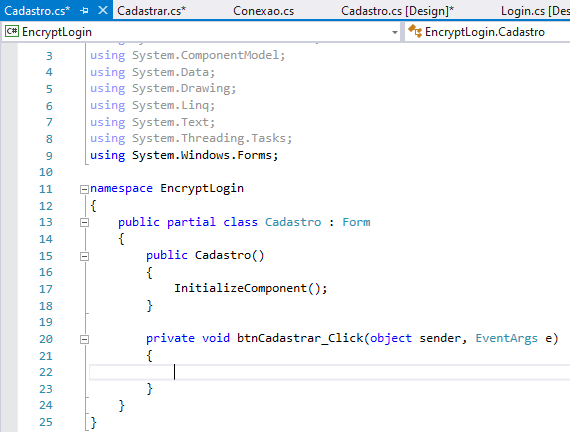


Figura 58. Evento do botão Salvar

Insira os códigos para buscar dos TextBox os dados inseridos pelo usuário

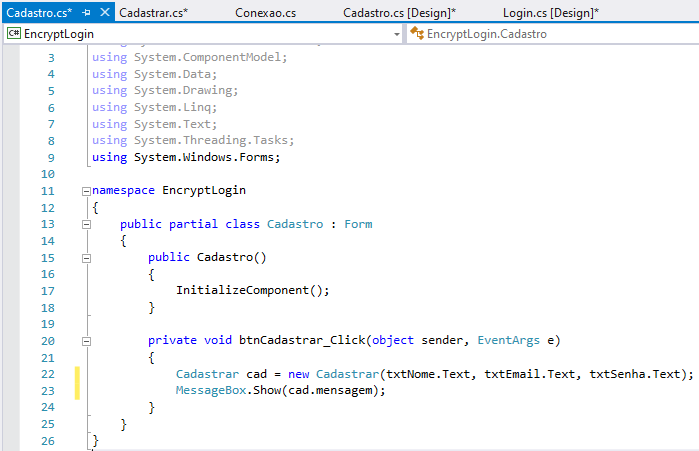


Figura 59. Dados vindos dos textbox.

# Botão de Cadastrar

Vamos fazer o botão Cadastrar chamar a tela de cadastro. No formulário Login dê um duplo clique no botão Cadastrar

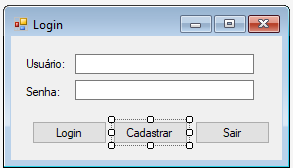


Figura 60. Botão Cadastrar

Irá gerar um evento e abrir o código do botão Cadastrar

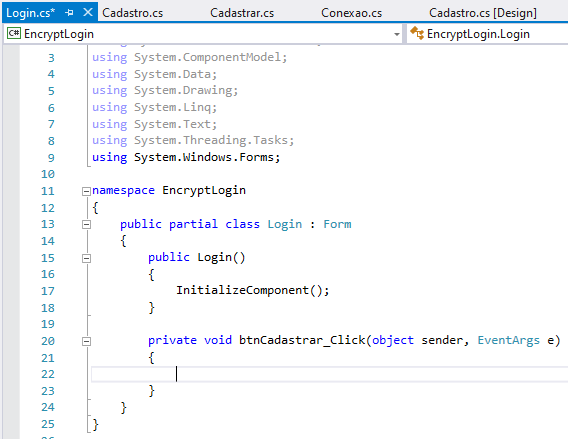


Figura 61. Evento do botão Cadastrar

Vamos instanciar a tela de cadastro

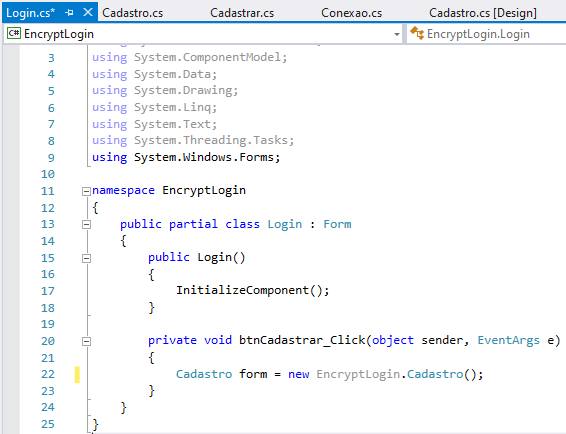


Figura 62. Cadastro instanciado

Vamos fazer a tela aparecer agora

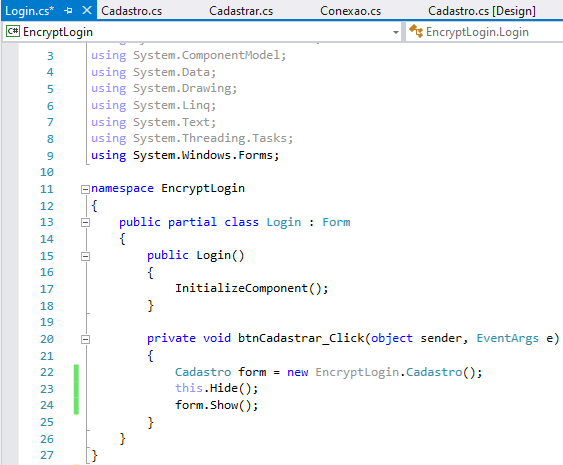


Figura 63. this.Hide

Testando

Vamos Iniciar o programa e inserir dados para testar e clique em Cadastrar no formulário de Login

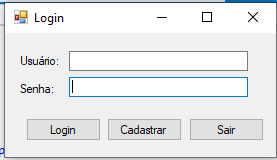


Figura 64. Tela de Login

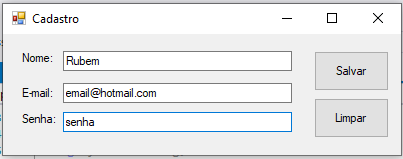


Figura 65. Dados digitados

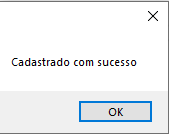


Figura 66. Cadastrado com sucesso

## Verificando dados no Banco

Rode um Select no Management

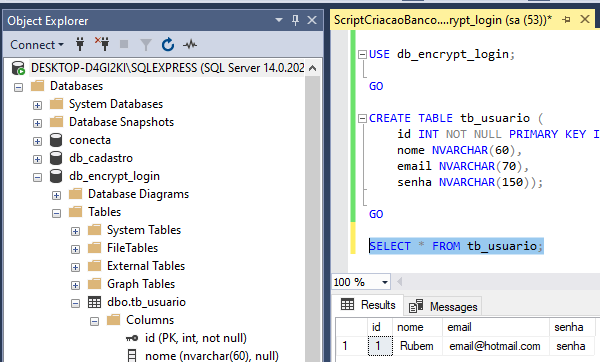


Figura 67. Dados inseridos checados no banco

# Encriptando o campo da Senha

Vamos adicionar uma classe em nossa solução. Clique com o botão direito na solução > Add > Class

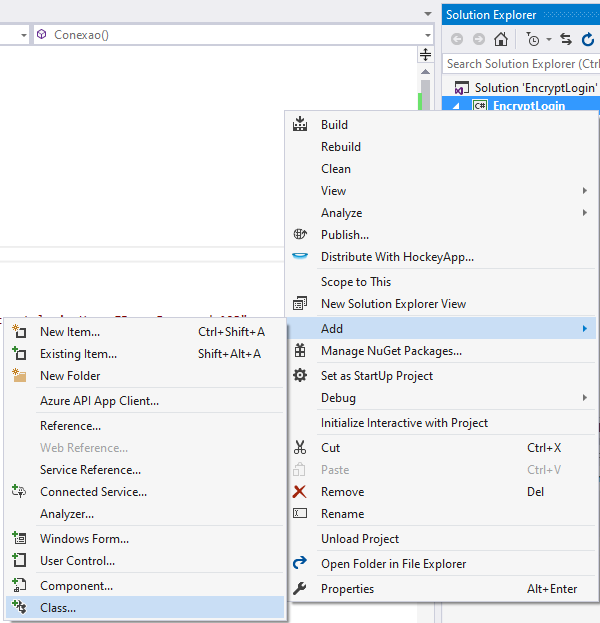


Figura 68. Adicionando uma classe

Vamos chama-la de HashCode

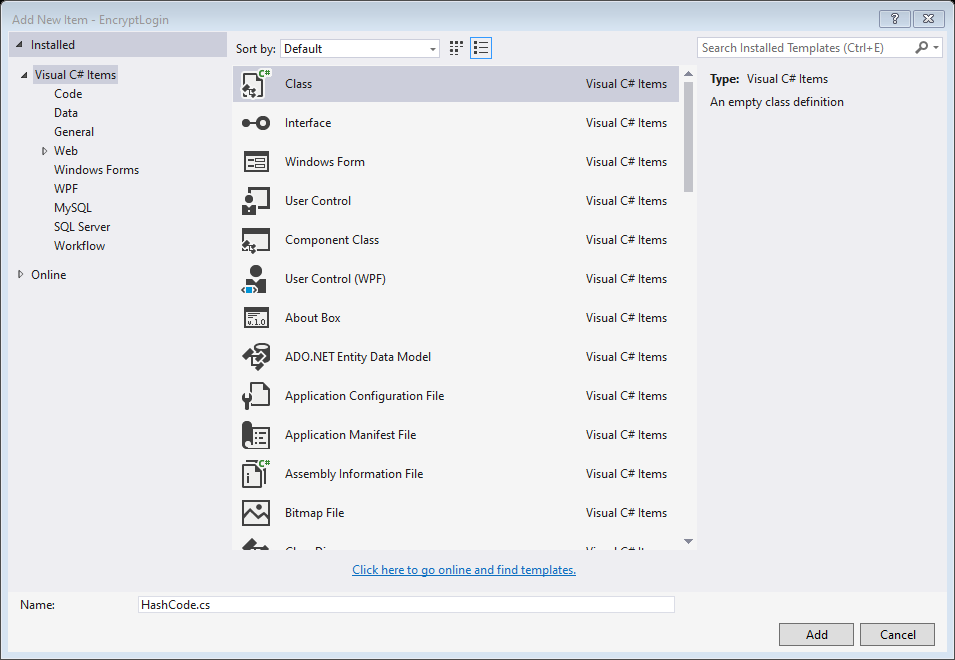


Figura 69. Classe HashCode.cs

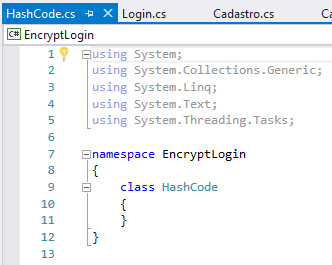


Figura 70. HashCode criada

Adicione a biblioteca

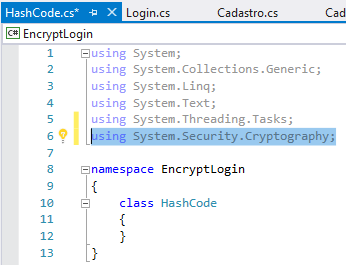


Figura 71. Biblioteca inserida

Vamos inserir uma string publica

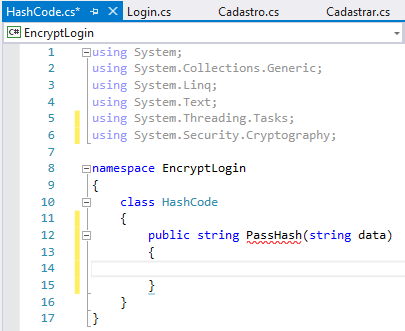


Figura 72. PassHash

Insira o código

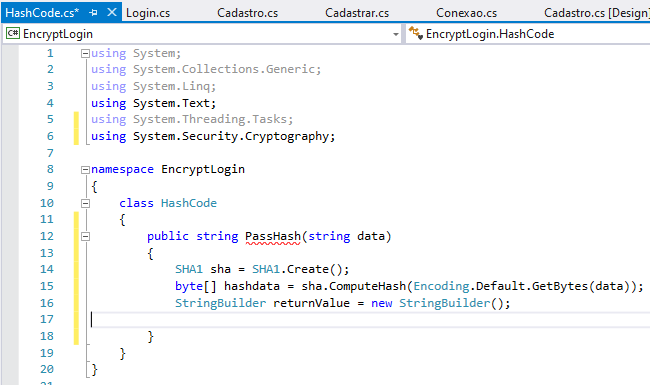


Figura 73. PassHash

Insira um for

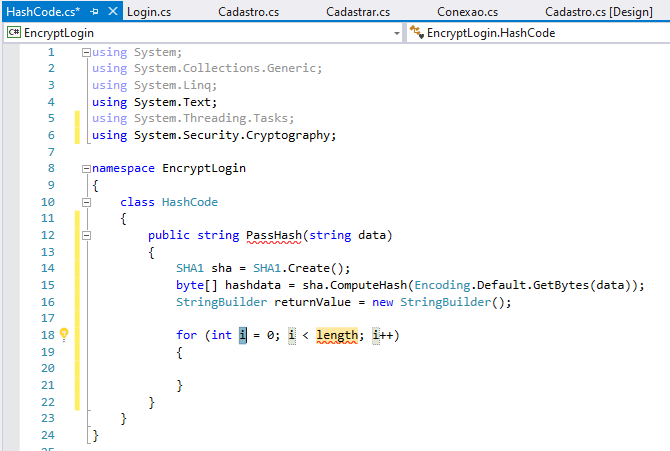


Figura 74. For inserido

Altere o for conforme código abaixo

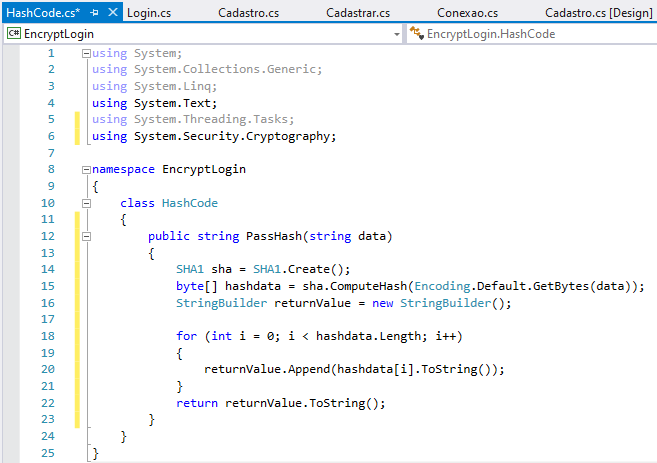


Figura 75. Classe HashCode concluída

Vamos para a classe Cadastro.cs e instanciar HashCode

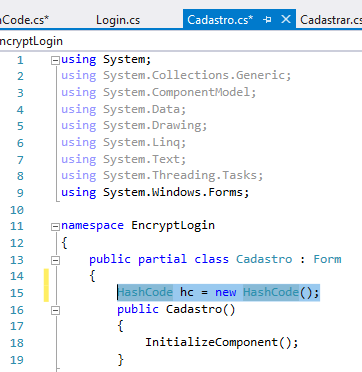


Figura 76. HashCode instanciado

Acrescente o hc.PassHash no campo da senha

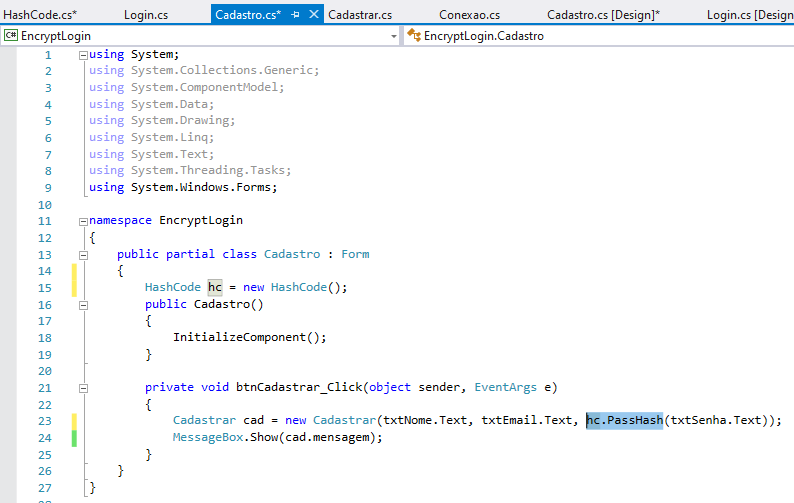


Figura 77. hc.PassHash

Vamos agora em Cadastrar.cs

Acrescente a instancia

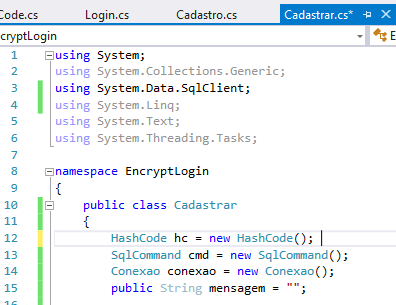


Figura 78. HashCode sendo instanciado

Em Cadastrar.cs acrescente o hc.PassHash

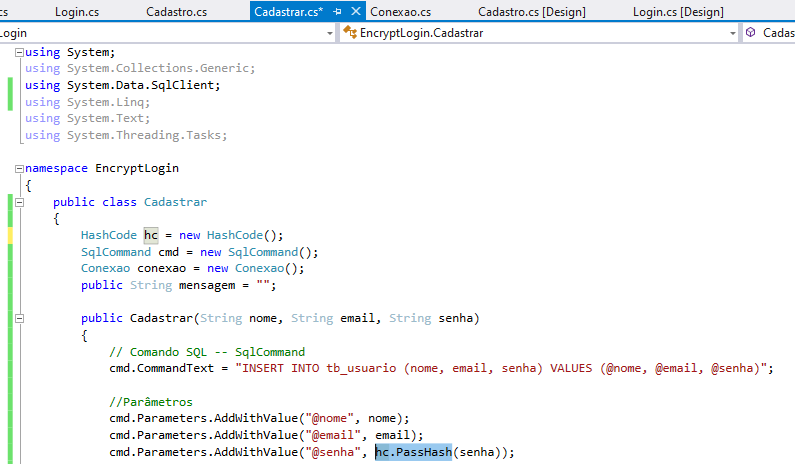


Figura 79. hc.PassHash

Testando

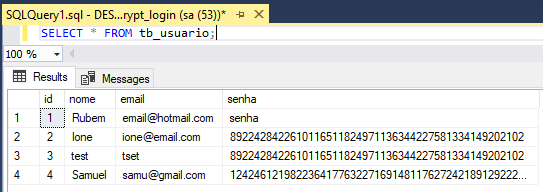


Figura 80. Campo senha encriptado

# Github

https://github.com/endroni/Encrypt\_Login.git

# Conclusão

Uma estruturação em quatro camadas pode ser um tanto complexo no início, porém possibilita um grande reaproveitamento em trabalhos que possuem funções semelhantes.

Espero que tenha valia para seus estudos.

**BIBLIOGRAFIA BÁSICA:**

PRESSMAN, Roger S. **Engenharia de Software: uma abordagem profissional**. 7. ed. São Paulo: Pearson Makron Books, 2011.

SOMMERVILLE, Ian. **Engenharia de Software**. 8. ed. São Paulo: Pearson Addison Wesley, 2007.

TONSIG, Sergio Luiz. **Engenharia de Software: análise e projeto de sistemas**. 2. ed. Rio de Janeiro: Ciência Moderna, 2008.

**BIBLIOGRAFIA COMPLEMENTAR:**

BEZERRA, Eduardo. **Princípios de Análise e Projeto de Sistemas com UML**. Rio de Janeiro: Elsevier, 2007.

Hugo Vasconcelos. **Programando em MVC com C#.** Udemy: Desenvolvimento com CSharp. Disponível em: < https://www.udemy.com/course/programando-em-mvc-com-c-4-camadas/>. Acesso em: 10 de setembro de 2019.