SENAC JUNIDIAÍ

Curso Técnico em Informática

LUCAS HENRIQUE DE LIMA GEREMIAS

PROJETO FINAL

CONSULTÓRIO MÉDICO

Jundiaí

2020

LUCAS HENRIQUE DE LIMA GEREMIAS

PROJETO FINAL

CONSULTÓRIO MÉDICO

Apresentação do projeto integrador final do curso técnico em informática da instituição SENAC JUNDIAÍ

Orientador: Marcos A. F. Cozatti

Jundiaí

2020

Sumário

Sumário

[**Introdução** 4](#_Toc58106845)

[**Documentação** 5](#_Toc58106846)

[**Conexão com o Banco de Dados e Criação de Tabelas** 5](#_Toc58106847)

[**Criação dos Forms** 8](#_Toc58106848)

[**Códigos e Funcionalidades** 14](#_Toc58106849)

[**Login** 14](#_Toc58106850)

[**Tela Principal (sistema)** 16](#_Toc58106851)

[**Telas de Cadastro** 16](#_Toc58106852)

[**Manutenção de Usuário** 23](#_Toc58106853)

# **Introdução**

Este projeto integrador consiste em um sistema funcional para um consultório médico. Foi realizado com o intuito de apresentar tudo que foi passado ao decorrer do módulo de programação em C#.

O objetivo é que seja um sistema usual e simples, mas que atenda as necessidades de um consultório médico, facilitando assim, a interação entre médico, clínica e paciente.

Para a realização deste foram utilizadas as seguintes ferramentas:

-Visual Studio Community 2019

- Microsoft SQL Server Management Studio 18

-Excel

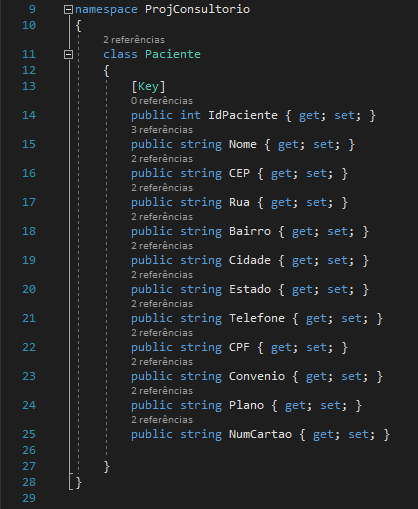
-Word

# **Documentação**

## **Conexão com o Banco de Dados e Criação de Tabelas**

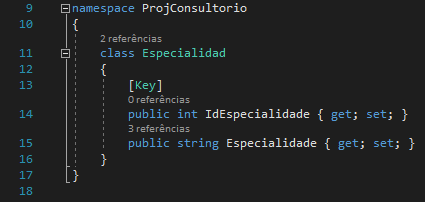
O projeto iniciou-se com a criação do banco (DbConsultorio), este não era meu objetivo, eu planejava utilizar a tecnologia de conexão Code First, mas tive certas dificuldades na criação do banco via código por isso foi utilizado o método Database First. Após isso a primeira coisa feita no Visual Studio foi a instalação da biblioteca EntityFramework versão estável 6.4.4 e com isso, iniciei o processo de conexão com o banco utilizando o “Fonte de Dados” e apelidando de “conexao” a string de conexão.

Depois desses passos meu sistema estava pronto para criação das primeiras 4 tabelas principais. Como dito anteriormente minhas tabelas foram criadas via código no Visual Studio, a primeira feita foi a tabela “Pacientes” contendo os seguintes campos:

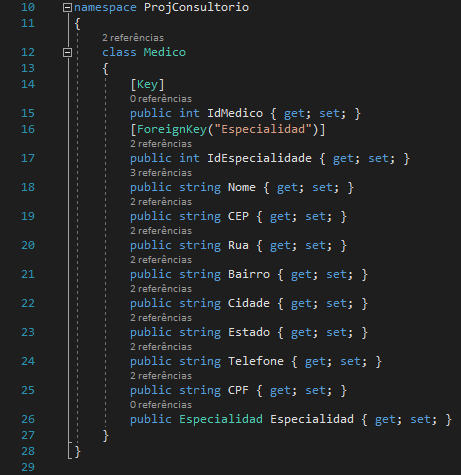


Para definir qual será a chave primária foi usado o “[Key ]”, com isso o “IdPaciente” foi criado como chave primária e com auto indentação.

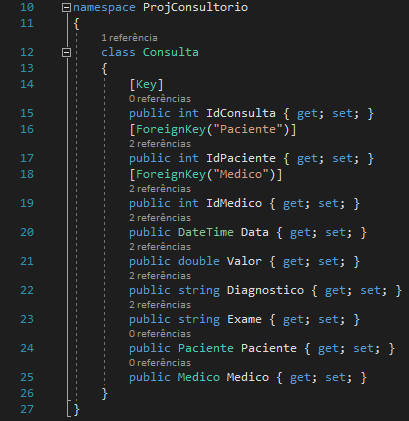
Seguida pela tabela “Especialidads”



Para que assim fosse possível a criação da tabela “Medico” com um Foreign Key da tabela “Especialidads”



E por último a tabela “Consultas” que contêm duas Foreign Key, uma da tabela “Pacientes” e outra da tabela “Medicos”

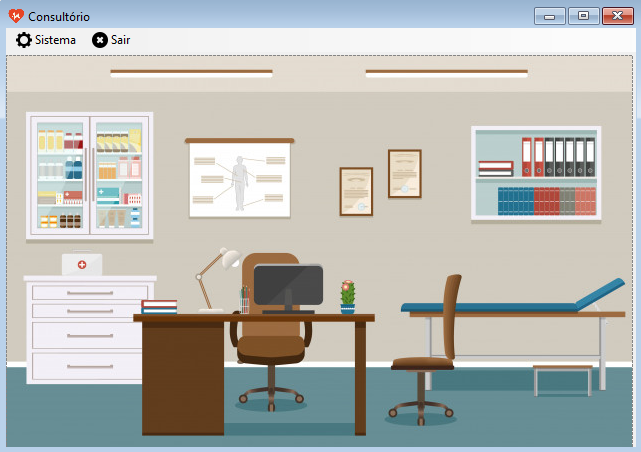


As foreign keys ou chaves estrangeiras foram muito importantes para que houvesse a conexão entre as tabelas, elas que realizam o relacionamento tornando o sistema mais prático e intuitivo.

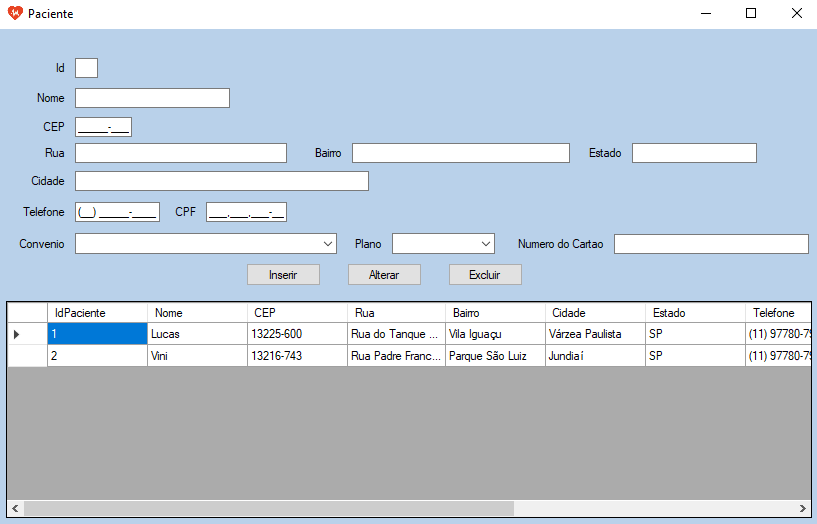
Feito isso só me restava fazer o update dessas tabelas no banco de dados, para isso, no Console do Gerenciador de Pacotes foram executados os seguintes 3 comandos para fazer a migração das tabelas para o banco: enable-migrations, Add-migration banco e update-database. Com isso, meu sistema estava conectado com o banco e minhas tabelas iniciais estavam criadas.

## **Criação dos Forms**

Essa foi umas das partes mais fácil do projeto, eu comecei criando a tela inicial do sistema onde a ferramenta Menu Strip mostra o caminho para as telas de cadastro e a tela de manutenção que foi criada posteriormente.

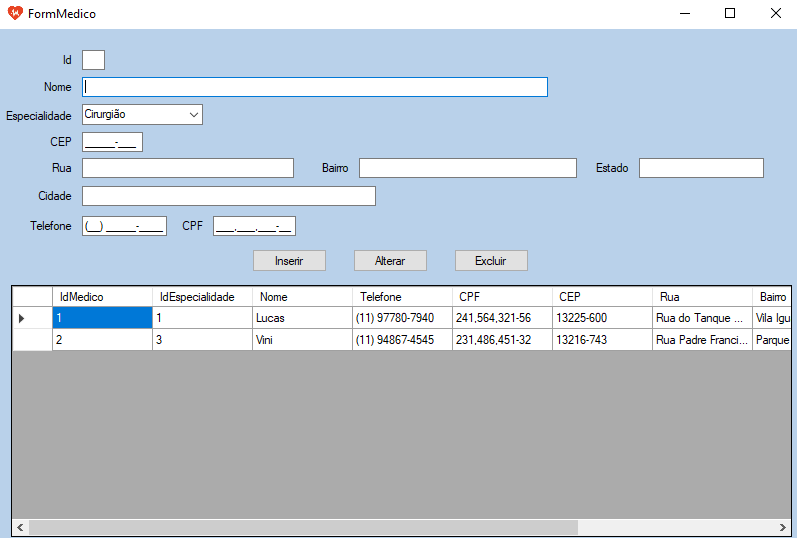


Em seguida eu criei a tela de cadastro de pacientes:



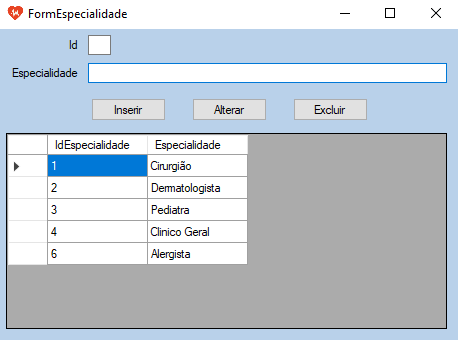
Essa tela também foi bem simples eu utilizei algumas Labels (que são os texto), alguns TextBox (que são essas caixas em branco), MaskedTextBox (são como as TextBox a diferença é que elas mostram alguns caracteres especiais que deixam o sistema com uma interface mais intuitiva, um exemplo é o Telefone que tem os parênteses onde fica o DDD, ela também foi usada no campo CEP e CPF), ComboBox (são campos com o funcionalidade de dropdown, possibilitando que eu coloque varias opções de escolha para o usuário, nesse campo não é permitido a escrita), a GridView (é esse campo cinza que mostra os dados da tabela e ele facilita a funcionalidade do botão excluir e alterar pois com um click ele carrega os dados da tabela nos campos em branco) e por ultimo os botões que são de grande importância para que seja feito o cadastro e futuras modificações.

O Form de cadastro de médicos:



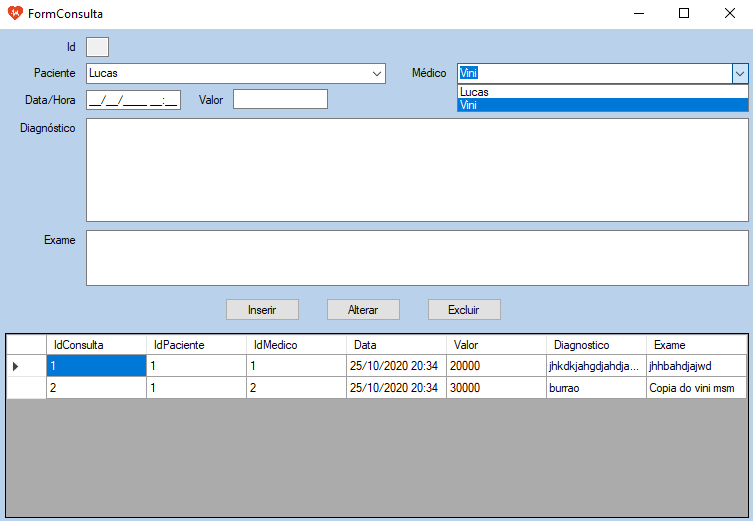
A tela do médico contém os mesmos campos e mesmas funcionalidades da tela de pacientes o único diferencial é o ComboBox de Especialidades que realiza uma conexão com o banco de dados onde os campos que estão contidos nele são os mesmos campos que estão e/ou podem ser adicionados na tabela de especialidades, quando uma nova especialidade é adicionada o ComboBox é atualizado.

Também tem a tela de Especialidade:



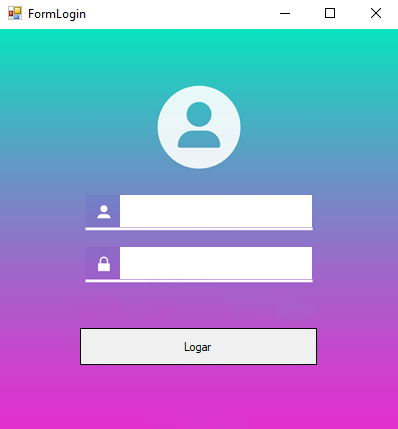
De todas, essa sem dúvidas foi a mais fácil.

E por último a tela de Consultas:



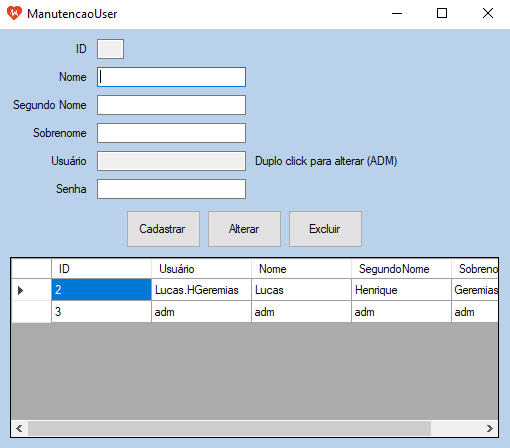
Essa tela também não tem nada de especial em relação às outras telas

A tela de login foi a que mais chamou atenção em relação ao seu design:

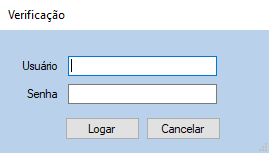


Essa foi a tela que ficou mais bonita, porém também não tem nada de especial em relação aos campos das outras tabelas.

A tela de manutenção de usuário onde pode ser feitas alterações nos cadastros dos usuários:



E por último uma simples tela de Login para habilitar a permissão de edição de usuário:

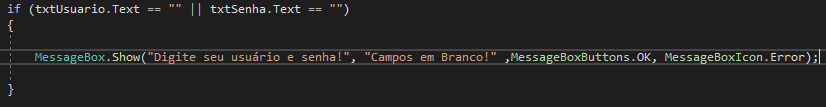


## **Códigos e Funcionalidades**

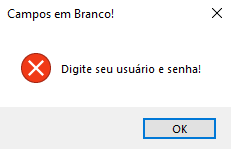
### **Login**

Agora a parte que faz o sistema funcionar, começando pela primeira tela que aparece, a tela de Login.

Primeiro, quando são pressionados o botão “Logar” ou a tecla “Enter” é feita uma verificação se os campos estão preenchidos.

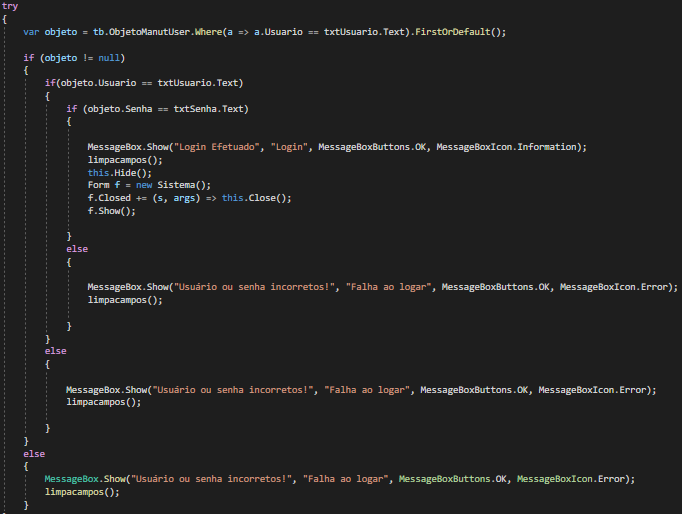


Caso não estejam, aparecerá o seguinte:

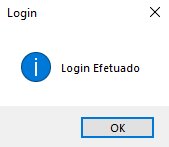


Caso estejam, seguirá para o processo de validação de usuário.

A validação do usuário é feita pelo seguinte código:

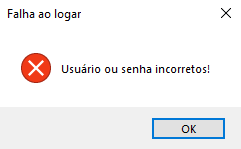


A primeira linha faz com que o sistema verifique se o usuário que foi escrito no TextBox de usuário (txtUsuario) existe no banco de dados, caso esteja escrito corretamente, ele verifica se a senha é a mesma cadastrada para aquele usuário, se todas essas informações forem confirmadas, os campos TextBox serão limpos, aparecerá a seguinte mensagem:



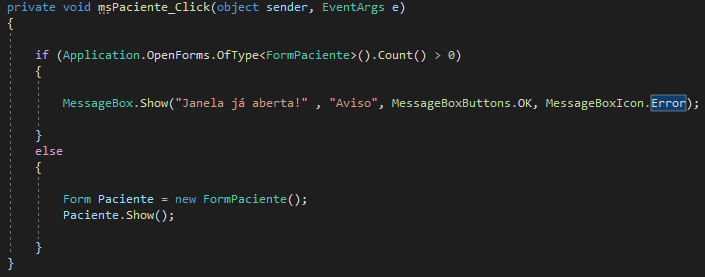
E logo após, tanto a tela de login quanto esta última tela, serão fechadas e abrirá a tela principal do sistema.

Caso algum campo esteja em branco e/ou contenham dados que não existem no banco de dados será mostrada esta tela:



### **Tela Principal (sistema)**

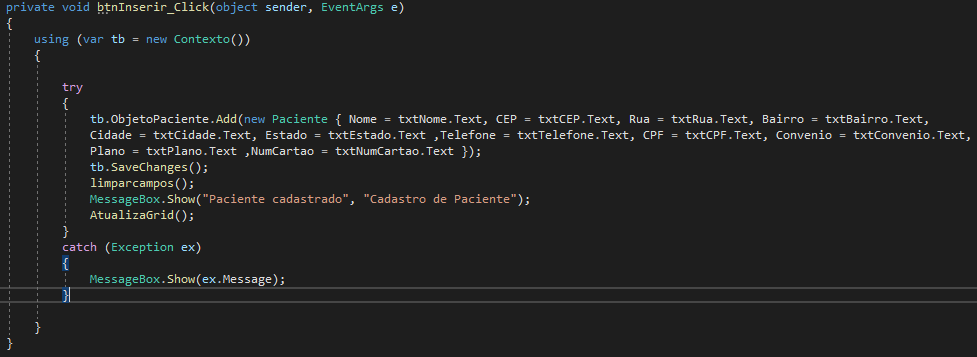
A tela principal é uma tela mais simples, possui um menu strip que redireciona para as paginas de cadastro e manutenções, o único código usado é para não permitir que seja aberta novamente uma página que já está em execução.



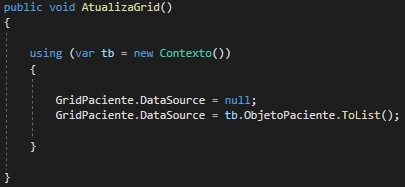
### **Telas de Cadastro**

As telas de cadastro em si têm códigos muito parecidos pois suas funções são parecidas. Usarei um como exemplo, mas destacarei funcionalidades especiais de cada tela.

Começando pelo botão “Inserir”, graças à biblioteca EntityFramework o código ficou bem mais compacto, essa biblioteca faz com que o código que seria escrito em 3 linhas seja encapsulado em poucos comandos, um exemplo no meu código é o comando “.Add” na primeira linha do evento “try” e o “.SaveChanges” na quarta linha.

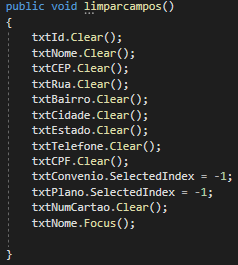


O código funciona da seguinte forma, o comando “.Add” indica o caminho onde serão inserido os dados escritos pelo usuário (“Nome = txtNome.Text” o campo Nome na tabela receberá os dados escritos no TextBox txtNome por exemplo), em seguida o comando “.SaveChange” realiza o update no banco, com isso o “limparcampos()” apaga os escritos nos TextBox facilitando uma nova inserção, é aberta uma tela confirmando o cadastro ou uma tela informando o erro ocorrido e por fim o GridView será atualizado pelo comando “AtualizaGrid()”.



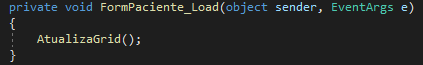
Aqui ele faz com que os dados existentes sejam apagados da Grid e os dados do banco carregados novamente.

Outra função usada é a “limparcampos”.

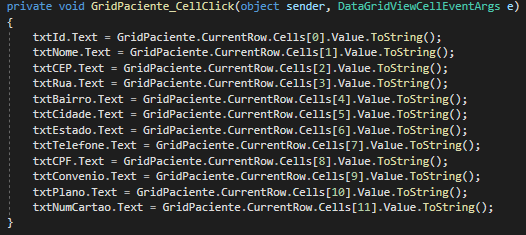


Como o próprio nome diz, ela faz a limpeza dos campos definidos usando o “.Clear”

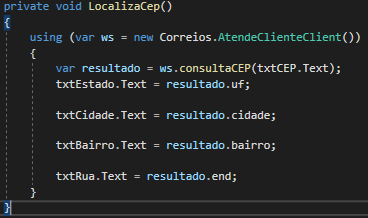
O comando “AtualizaGrid()” também está contido no evento load, ou seja, toda vez que a tela for aberta o GridView será atualizado.



Clickando em uma célula na GridView os dados da mesma serão carregados nos respectivos TextBox graças ao evento “CellClick”, possibilitando alterações no cadastro.

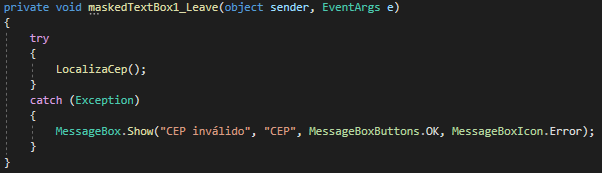


Essa tela, assim como a de cadastro de médicos, possui um serviço dos Correios que contêm os endereços e conforme o CEP for informado os demais campos do endereço serão preenchidos.



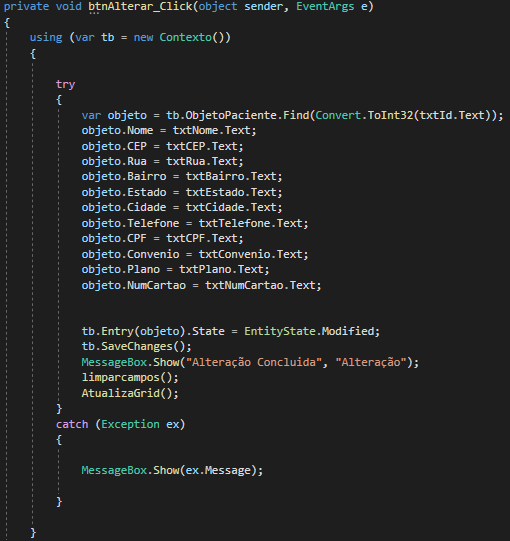
No código eu apenas indico onde será inserido o CEP e onde cada valor deve ficar.

Quando o CEP for informado e somente após o usuário sair deste TextBox, os demais campos de endereço serão preenchidos com as informações do CEP graças ao evento “Leave”.



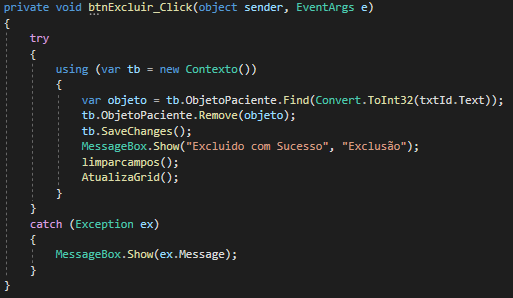
Ele faz com que determinado comando seja executado após o usuário sair do TextBox com esse evento

O código a seguir pertence ao botão “Alterar” apesar de ser maior, ele é simples igual o botão anterior.



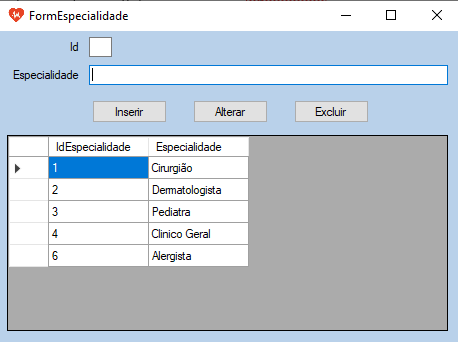
Após ter os dados carregados nos TextBox o comando “Find” procura no banco de dados dentro da tabela especificada (tb.ObjetoPaciente) o usuário a ser alterado pelo campo pré-definido (no meu caso txtId), em seguida é definido os campos que serão alterados e assim como no botão de inserção, os dados serão salvos, os TextBox serão limpos, será mostrada uma tela confirmando a alteração ou informando um possível erro e por fim a GridView receberá a atualização.

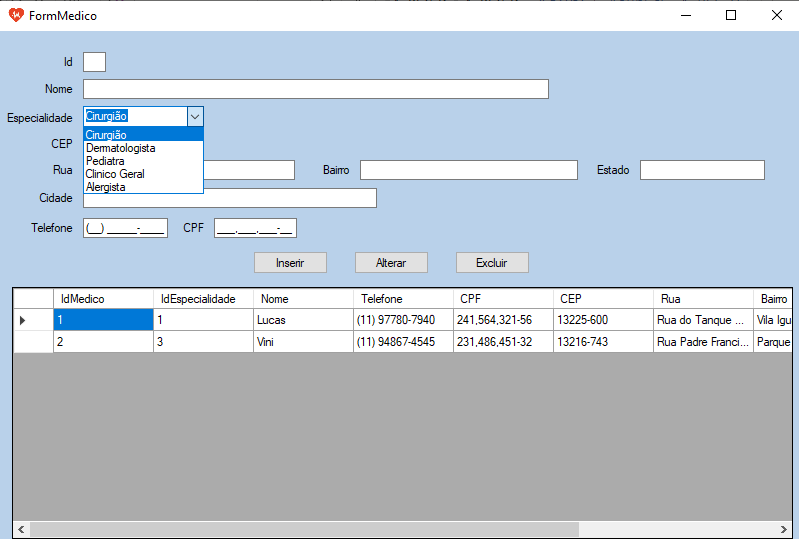
Da mesma forma é possível ser feita a exclusão de um cadastro.



A diferença desse botão, é que o “Find” só precisa achar o id que será excluído e com isso o comando “.Remove” faz com que todos os dados daquele id sejam deletados e assim, é enviado esse update para o banco de dados, os TextBox serão limpos, será exibida uma tela confirmando a exclusão ou informando um possível erro e em seguida a GridView receberá esta atualização.

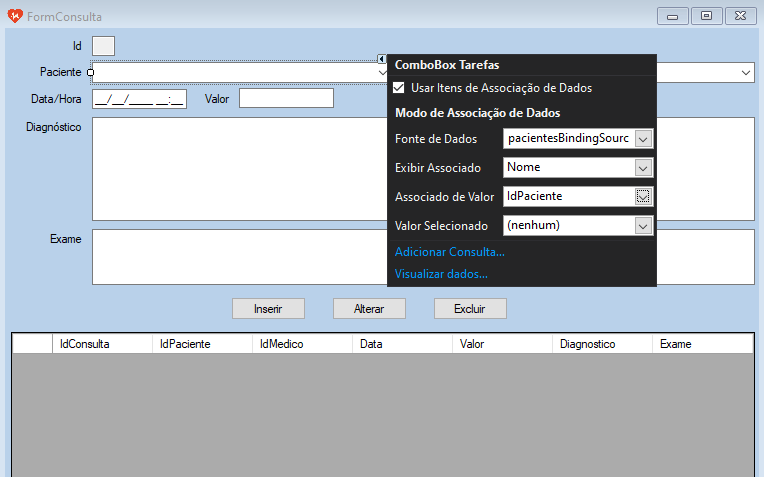
Algumas telas possuem um ComboBox que faz conexões com um ou mais tabelas de outras telas. A tela do cadastro de médicos por exemplo faz ligação com a tabela “Especialidads”, o objetivo é que todas as especialidades cadastradas na tela de especialidades, apareçam no ComboBox da tela de médicos.



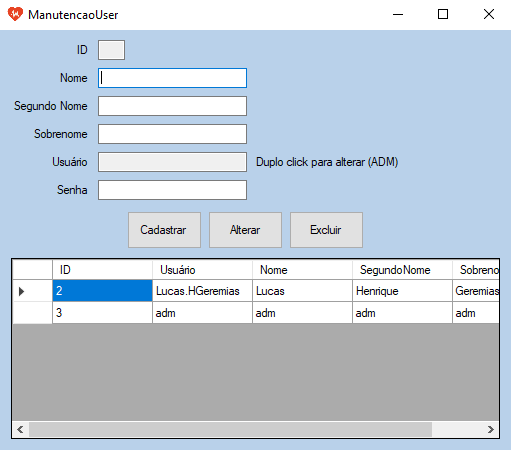


O mesmo acontece na tela de consultas, mas lá há duas conexões, uma com a tabela “Medicos” e outra com a tabela “Pacientes”.

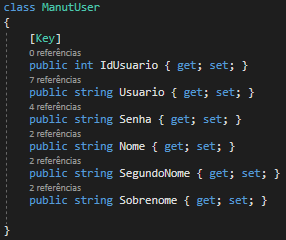
Para tal, na tela de tarefas do ComboBox eu habilitei a associação de dados e realizei a conexão com o banco de dados de uma forma bem parecida com a que foi feita na conexão do sistema com o banco, utilizei o “Fontes de Dados” onde eu define em qual tabela os dados seriam pegos, depois disso em “Exibir Associado” eu escolhi qual campo da tabela eu gostaria que aparecesse e em “Associado de Valor” eu mostrei qual seria o parâmetro de busca.



### **Manutenção de Usuário**

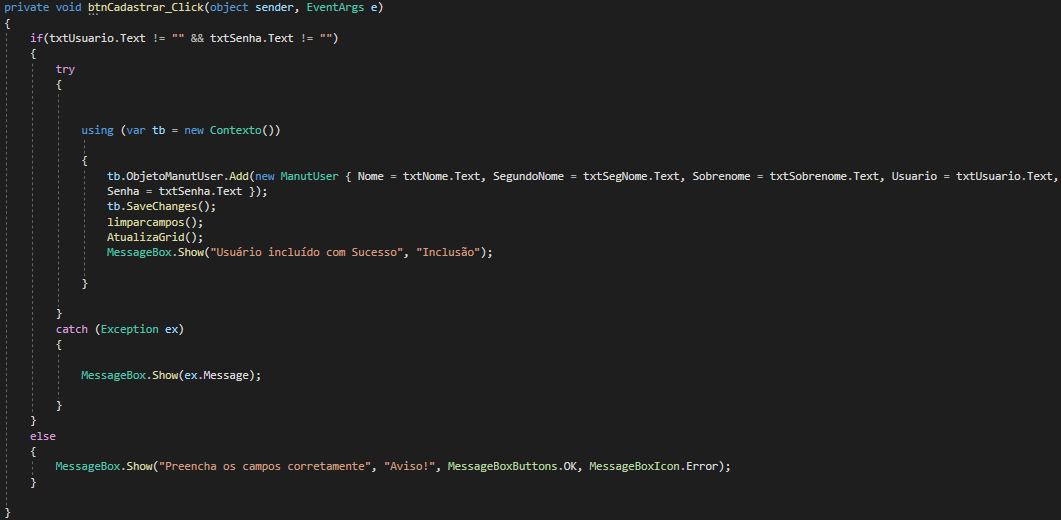


Foi uma das últimas telas a ser feitas, aqui é onde são criados os usuários que serviriam para os funcionários do consultório fazerem o login no sistema.

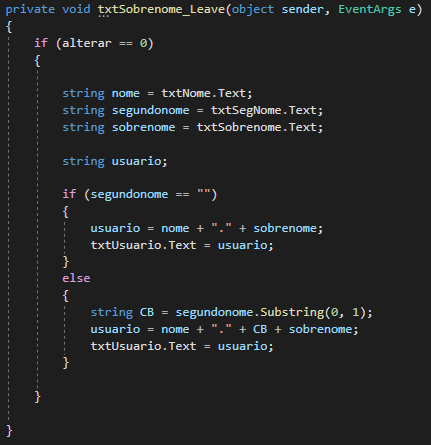


Logo de início criei uma nova tabela e a upei no banco com aqueles mesmos três comandos de migração usados nas primeiras tabelas.

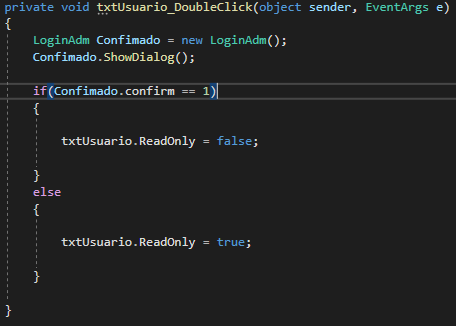
O código dessa tela é um pouco diferente dos demais códigos de cadastro. Primeiro é verificado se os campos usuário e senha estão preenchidos, em caso negativo será aberta uma tela avisando e o cadastro não será efetuado, do contrário ocorrerá a concatenação do nome, um ponto, a primeira letra do segundo nome (caso o usuário tenha) e o último nome para a criação do usuário que será escrito para fazer o login. Foi usado o comando “.Substring(0 ,1)” para que seja pega somente a primeira letra do segundo nome, o código da concatenação esta contido no evento “leave” do TextBox do sobrenome



Assim que o usuário sair deste campo, automaticamente serão concatenados os dados e exibido o nome de usuário que será criado.



Importante ressaltar que o nome de usuário só pode ser alterado pelo administrador do sistema. Para isso será necessário clicar duas vezes no TextBox de usuário e isso redirecionará o usuário à uma nova tela de login.



Quando aberta a tela de login de administrador, após digitar o usuário e senha, o sistema verificará no banco se os dados digitados batem com o login de administrador, caso não, será exibida uma tela de falha no login, caso contrário a variável “confirm” receberá o valor 1 e no código acima mostra que o TextBox do usuário somente sairá do modo “ReadOnly”(apenas leitura) quando o valor de “confirm” for igual a 1 ,ou seja, o usuário estará apto à realizar a alteração.

