

Universidade do planalto catarinense

Curso de sistemas de informação

Ricardo Taruhn Duarte

**Projeto de Desenvolvimento de Sistemas**

Lages (SC)

2020

Sumário

1 REgra de negócio 4

2 LEvantamento de requisitos 5

3 diagrama de classe 7

4 como criar repositório no github 9

5 Desenvolvimento 16

5.1 Criando uma solução 16

5.2 Criando os projetos 19

5.3 Organizando em pastas 19

5.4 Criando as telas do projeto 20

5.5 Criando um teste em uma classe de acesso a dados (DAO) 22

5.6 Criando uma base de dados no SQL Management Studio 23

6 adicionando um produto no banco de dados 29

6.1 Adicionando uma lista no Banco de dados 31

6.2 Criando uma tabela no SSMS 32

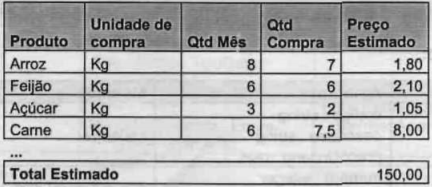
6.3 Usando o insert e values no código 34

6.4 Classe Produto 36

6.5 Configurando o Tabpage 38

1. REgra de negócio

O sistema de lista de compra tem como função objetiva, desenvolver uma solução em controle de gastos para manter a média de compras mensais, obter uma lista para ter o controle de dados, com ela ter uma média em quanto é gastado em cada mês e fazer uma valor estimado em o quanto e o que pode ser gastado, sendo abaixo do média ou acima, deve haver um relatório de todos os produtos comprados no mês.

1. Controle de Lista de Compras no Excel.

Fonte: Thiago Sartor (?)

1. LEvantamento de requisitos

Nesse levantamento, tem como função declarar em uma linguagem natural e com o diagrama as funcionalidades e as restrições sob as quais o sistema deve ser operado. Com esse documento escrito mostra de forma clara como é o conhecimento técnico detalhado do sistema levantado.

1. Requisitos do usuário na lista de compras.

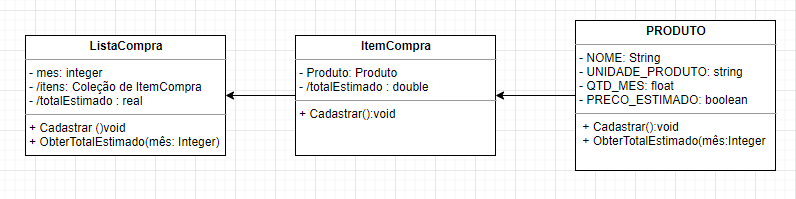
|  |  |  |
| --- | --- | --- |
| CÓDIGO | DESCRIÇÃO | PRIORIDADE |
| RF1 | **Condicionar uma sistematização de adicionar uma lista de compras.** | **Obrigatório** |
| RF1.1 | **A lista de compras deve adicionar o cadastro de um produto;** | **Obrigatório** |
| RF1.1.1 | **Os dados do cadastro do produto devem ser:** | **Obrigatório** |
| RF1.1.1.1 | **Nome do produto, unidade de compra, quantidade no mês, quantidade de compra, preço estimado.** | **Obrigatório** |
| RF2 | **A execução do programa deve ser operada em um computador** | **Obrigatório** |
| RF2.1 | **Quando o cliente liga a aplicação e não existe uma lista armazenada: permitir a criação de uma lista** | **Obrigatório** |
| RF2.1 | **Permitir o status atual da lista de compras Gerenciamento dos produtos** | **Obrigatório** |
| RF2.1.1 | **Listar o relatório da lista de compras feita no mês** | **Obrigatório** |
| RF2.1.2 | **Listar todos os produtos adicionados no mês** | **Obrigatório** |
| RF2.1.3 | **Listar as descrições do produto cadastrado** | **Obrigatório** |
| RF2.1.1.1 | **Permitir edição dos valores dos preços dos produtos em geral** | **Obrigatório** |
| RF2.1.1.2 | **Permitir edição da unidade de valor utilizada do produto** | **Obrigatório** |
| RF2.1.1.3 | **Permitir edição da quantidade limite dos produtos em geral** | **Obrigatório** |
| RF3.1 | **Deletar produtos que não vão agregar no relatório da lista do mês** | **Obrigatório** |
| RF4.1 | **Gerar o valor deve ser calculado na soma de todos os produtos listados** | **Obrigatório** |
| RF4.2 | **Gerar o relatório da lista de compras feita no mês** | **Obrigatório** |
| RNF5.1 | **Todas as variáveis de entrada terão valores default e tais valores serão usados sempre que dados de entrada estiverem faltando ou inválidos.** | **Obrigatório** |
| RNF6.1 | **O sistema deverá ser acessado via computador** | **Obrigatório** |
| RNF7.1 | **O sistema deverá atender às normas legais, tais como padrões de leis e etc...** | **Obrigatório** |
| RNF8.1 | **O sistema deve fazer documentação e coloca-los via backup de cada mês feito na lista de compras** | **Obrigatório** |

Fonte: Autor próprio

1. diagrama de classe

Nesse diagrama de classe para a lista de compras foi feito com os seguintes requisitos:

1. Diagrama de classe da lista de compras



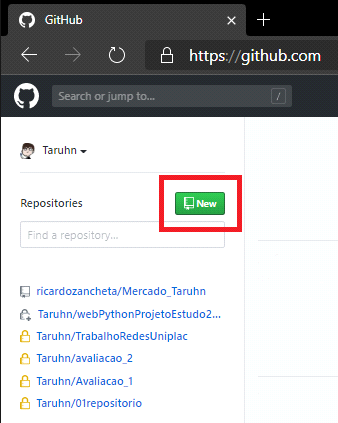
Fonte: Autor próprio

* *ListaCompra*: O *ListaCompra* tem como função de listar a compra do mês, mesmo não tendo algo incluso dentro dela. O atributo “*mes”* mostra qual mês ela está sendo listada, mesmo não havendo um específico mês em colocar. O atributo “*itens”*, ele herda tudo o que está no *ItemCompra* para coletar todos os dados que ele armazenou. E o totalEstimado é o cálculo de todos valores dos itens e gera um resultado somando totalmente ele. No método *cadastrar* está público na visibilidade dele, e o *ObterTotalEstimado* gera o valor total estimado que foi somado no mês
* *ItemCompra*: A classe *ItemCompra* tem o atributo *Produto*, que ele herda um certo produto com as descrições que apresenta. O *totalEstimado* é a soma do total estimado na qual o produto tende a chegar.
* No *Produto* ele descreve todos os atributos necessários para gerar a classe produto, nele tem o *nome* do produto, a *UNIDADE\_PRODUTO* mostra o tipo de unidade que esse produto usa, a *QTD\_MES* apresenta a quantidade dela comprada no mês, e o *PRECO\_ESTIMADO* exibe o valor que provavelmente a unidade que o produto deve valer.

1. como criar repositório no github

Na criação do repositório é necessário acessar o site do GitHub, fazer o login e entrar na conta, e clique na opção no botão "New".

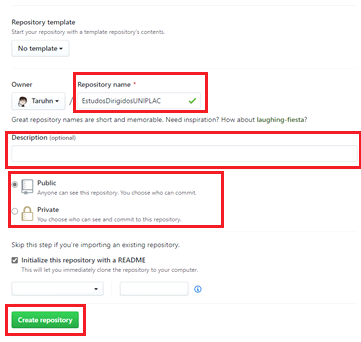
1. Criar novo repositório.



Fonte: Autor próprio

Depois de selecionado o botão, ele entrará em uma nova página ele irá pedir o preenchimento de alguns requisitos para a criação.

1. Configurar o repositório.



Fonte: Autor próprio

No "*repository name",* basta colocar o nome do repositório que deseja, em "*description*" descreva o que quer colocar como uma descrição sobre ele ou algum complemento na função que ele foi criado.

Em seguida tem as opções de público ou privado, sendo que quando o projeto do repositório é público fica acessível a todos quando todos podem ver o que está dentro dele, e o uso do *private* é usado para usuários que querem colocar permissões de acesso alterando a visibilidade do repositório. E para finalizar a criação do repositório clique em "*create repository*".

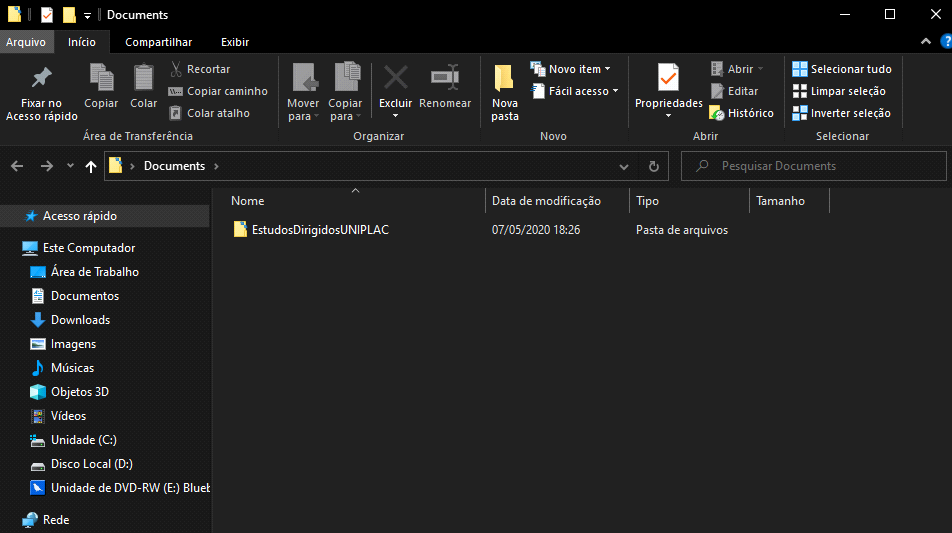
1. Pasta do repositório no GitHub.



Fonte: Autor próprio

Com a criação do repositório feita, é necessário criar uma pasta dedicada para colocar o repositório do GitHub para o computador pessoal. Para isso precisa criar uma pasta vazia com o nome e o local desejado.

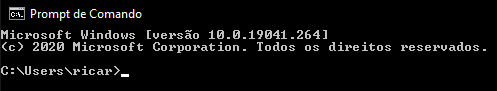
1. Pasta criada para o repositório.



Fonte: Autor próprio

Depois de criada a pasta, entre no prompt de comando do computador.

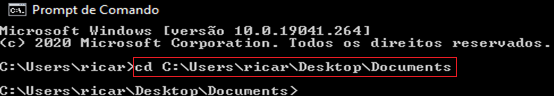
1. Dentro do Prompt de comando.



Fonte: Autor próprio

Em seguida digite o comando "cd" para ir no diretório que a pasta está e depois a posição que ele está presente.

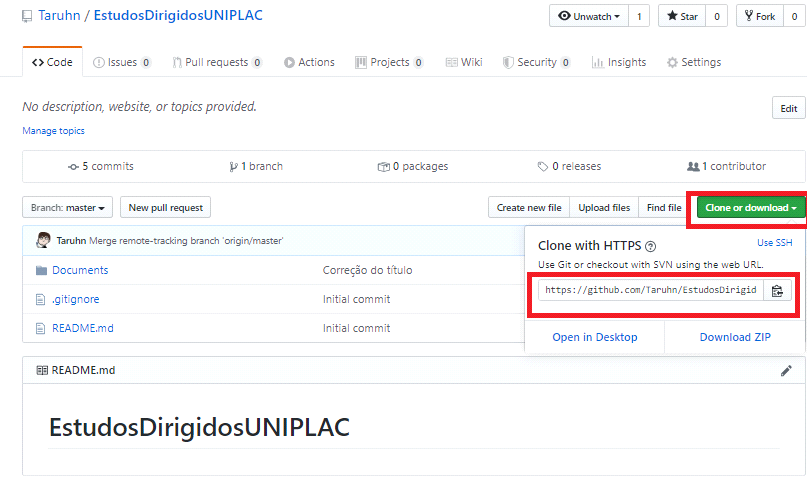
1. Inserindo o comando “cd”.



Fonte: Autor próprio

Para fazer pegar o clone do repositório é necessario ir no link dele e clicar no botão "Clone or Download", depois de clicado ele vai mostrar um link de endereço onde o repositório está, e copie-o.

1. Copiando o link no GitHub.



Fonte: Autor próprio

Agora no prompt de comando, se usa o comando "git clone", com ele se obtém uma cópia atual do repositório que está no GitHub.

1. Usando o comando “git clone”.



Fonte: Autor próprio

Quando é posto todos os comandos, pressione *enter*.

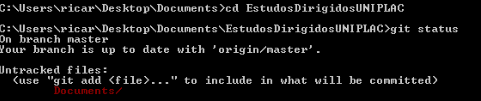
1. Confirmação após o “git clone”.



Fonte: Autor próprio

Depois de clonado os arquivos do repositório dentro do documento, é necessário colocar o comando "*git status*", com ele consegue ver as mudanças que foram feitas dentro do repositório, se algum documento foi alterado ele vai ficar em vermelho.

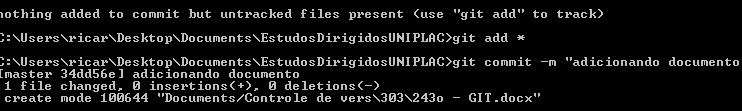
1. Após usar o comando “git status”.



Fonte: Autor próprio

Para colocar os arquivos dentro do repositório, é necessário está na posição em que o diretório está presente dentro do computador e usar o comando *"git add \*"*, com esse comando ele adiciona as mudanças do repositório nas quais que foram adicionadas dentro dela, e em seguida digite "*git commit -m "mensagem explicando o que está fazendo" ",* com ele confirma do que está mandando para o servidor, a partir disso é necessário fazer outra mudança.

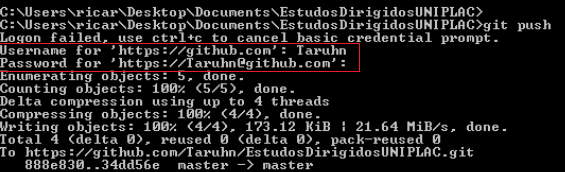
1. Usando o comando “git add” e “git commit -m”.



Fonte: Autor próprio

E por fim, para mandar o documento para o repositório, se usa o comando "*git push*", faz com que mande o commit localmente do computador para o repositório.

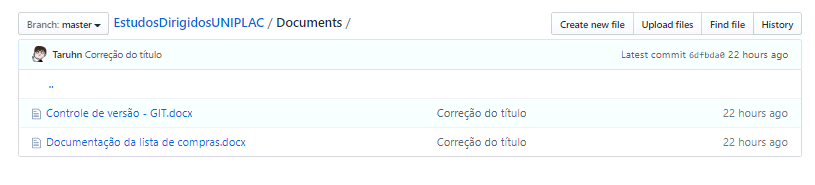
1. Colocando o e-mail e senha.



Fonte: Autor próprio

Para a confirmação da transferência do arquivo, é requerido colocar o e-mail da conta do GitHub e a senha do criador ou colaborador (no caso se o repositório esteja privado), para confirma a transferência, e depois de finalizar a ação ele estará presente dentro do servidor.

1. Documento no repositório depois da transferência.



Fonte: Autor próprio

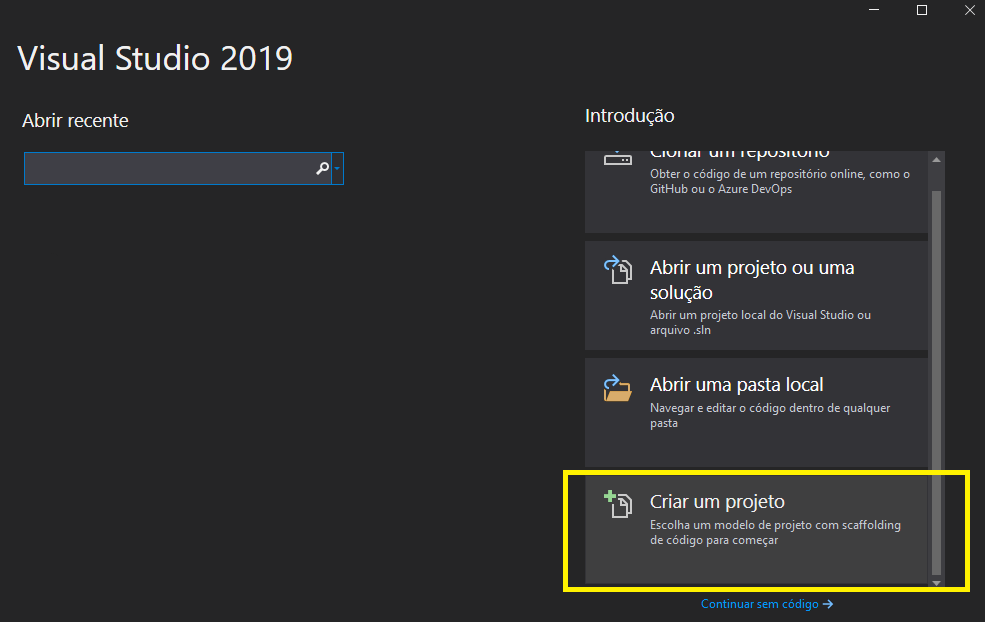
1. Desenvolvimento

Na criação desse desenvolvimento de um sistema de lista de compras é utilizado o Visual Studio 2019 para o desenvolvimento de software dedicado ao .NET FRAMEWORK utilizando a linguem de programação em C#, sendo que a tela de interface será em *desktop*, com criação com um banco de dados usando o SQL Management Studio.

* 1. Criando uma solução

Para adicionar uma solução no Visual Studio 2019, precisa inicia-lo e escolher a opção “criar novo projeto” no campo inferior direito para iniciar uma nova solução.

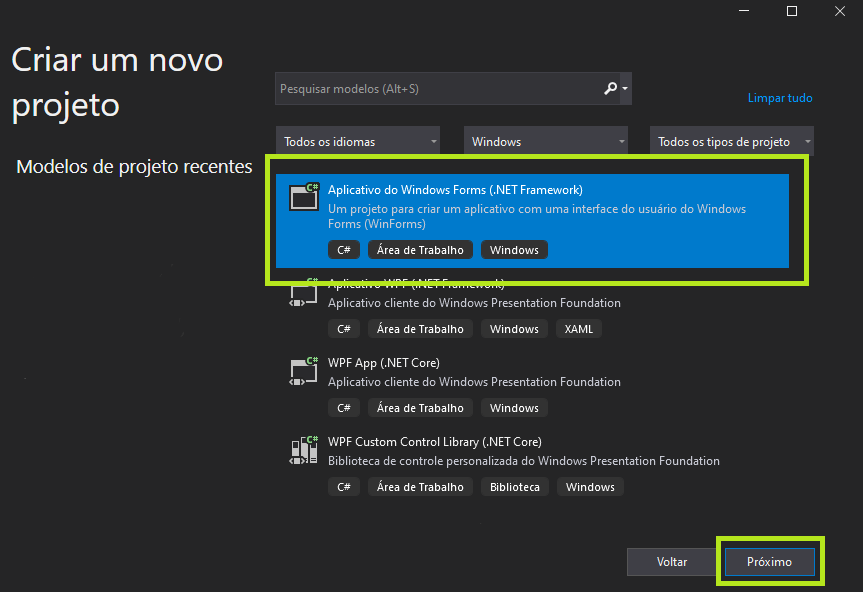
1. Menu de introdução no Visual Studio.



Fonte: Autor próprio

Depois de selecionado vai parecer inúmeros tipos de projetos para iniciar no programa, a aplicação que vai ser utilizada vai ser o Windows Forms (.NET Framework), pois ela fornece acesso a funcionalidade mínima de um elemento de interface de usuário, tendo textos, fontes, tamanho, e eventos comuns como o clique e arraste/queda do mouse.

1. Selecionando o Windows Forms (.NET framework).

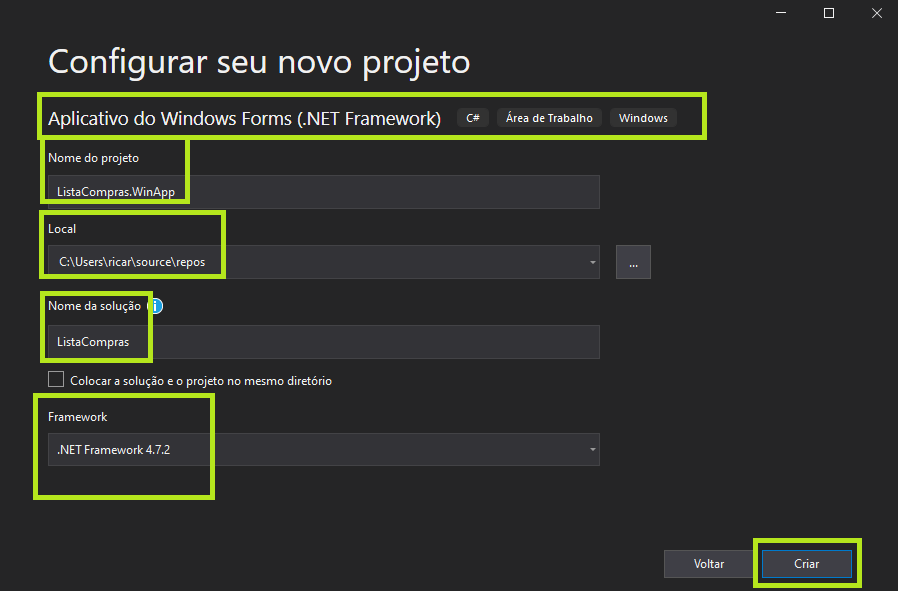


Fonte: Autor próprio

Assim que foi selecionado, pede para dar continuidade com a criação, depois que escolheu o tipo do projeto, ele mostra na tela como confirmação se é esse o projeto a ser criado, com ele, mostra qual linguagem ele suporta, onde vai ser aplicado e qual o sistema operacional que vai suporta-lo.

Abaixo pede para colocar o nome do projeto, em seguida o local onde o repositório do arquivo deve ficar, o nome da solução, significa que ele é um container para um ou mais projetos no Visual Studio, e por fim qual a versão do .NET Framework que vai ser utilizado, e com isso com o botão “criar” ele fez um novo projeto para trabalhar.

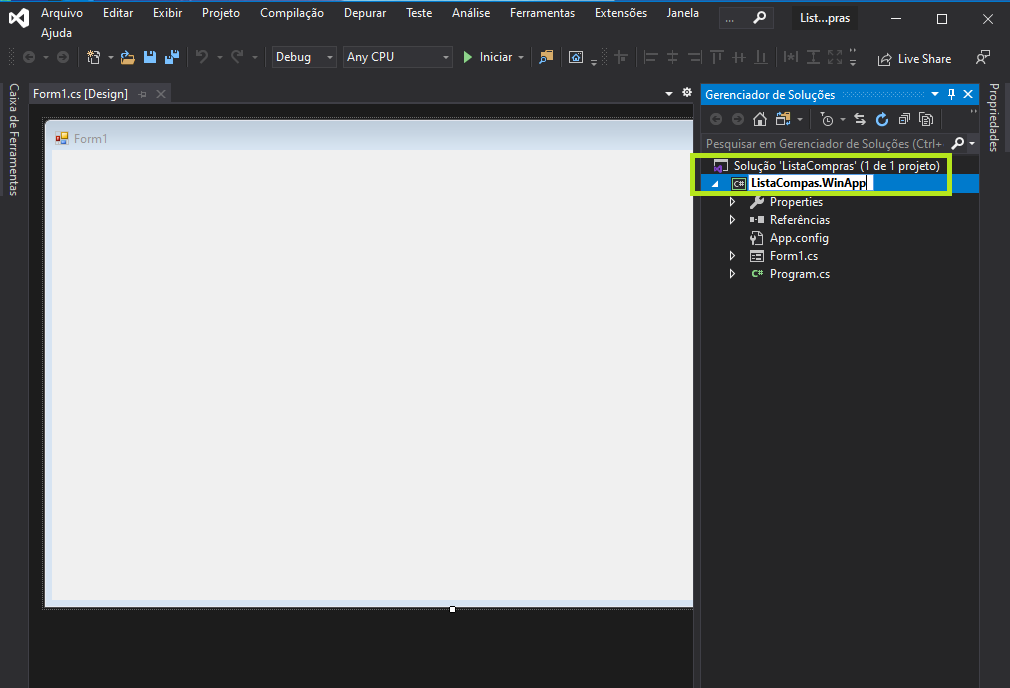
1. Configurando o projeto no Visual Studio



Fonte: Autor próprio

Quando iniciado ele vai para aparecer a área de trabalho do Visual Studio, ao lado direito da tela aparece o nome da solução criada e também o ícone com o nome da criação do Windows Forms.

1. Tela da área de trabalho do Visual Studio

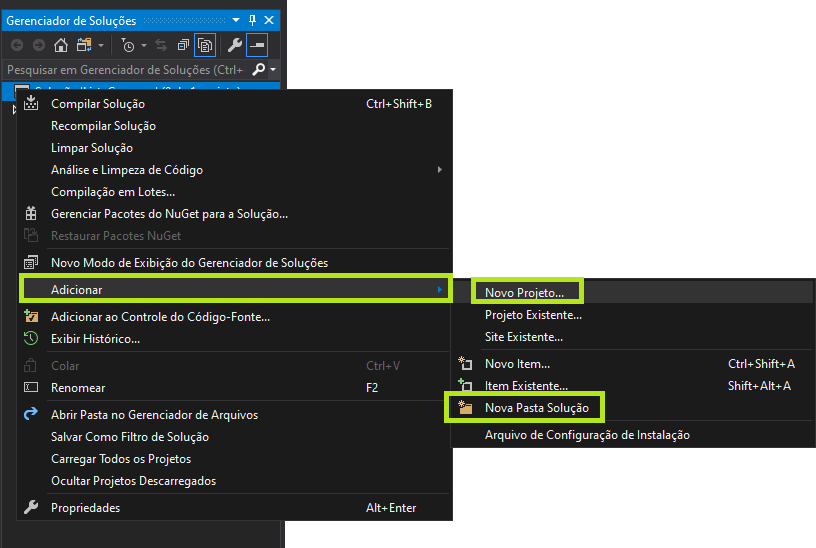


Fonte: Autor próprio

* 1. Criando os projetos

Para criar novos projetos dento da solução feita, deve-se ir no nome da solução dentro do gerenciador de soluções e dentro dele tem o nome da solução e o números de projetos que tem dentro dela, para adicionar um novo deve-se clicar o botão direito, ir em adicionar e pro fim no “Novo projeto...”.

1. Adicionado um projeto no gerenciador de soluções



Fonte: Autor próprio.

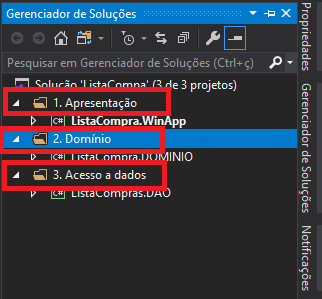
Assim que aparecer a tela de novo projeto, vai ser criado dois novos projetos de biblioteca de classe (.NET Framework), tendo eles como *ListaCompra.Dominio* e *ListaCompras.DAO*.

* 1. Organizando em pastas

Para obter mais planejamento com os projetos que estão dentro das soluções, se usa pastas para separar e gerenciar com mais controle os projetos, conforme a FIGURA 19, ela segue o mesmo caminho para criar uma pasta ao invés de um novo projeto, é selecionado “Nova Pasta Solução”.

Dentro da solução ListaCompras tem 3 classes sendo cada uma separada em 3 pastas, sendo elas, "1.Apresentação" com a função de ser o ponto de entrada principal para a aplicação, "2.Domínio" dentro dela vai ficar as classes e a conexão para o SQL SERVER, o "3.Acesso a dados" nessa pasta terá o acesso aos dados de conexão ao banco de dados.

1. O uso da criação de pastas.

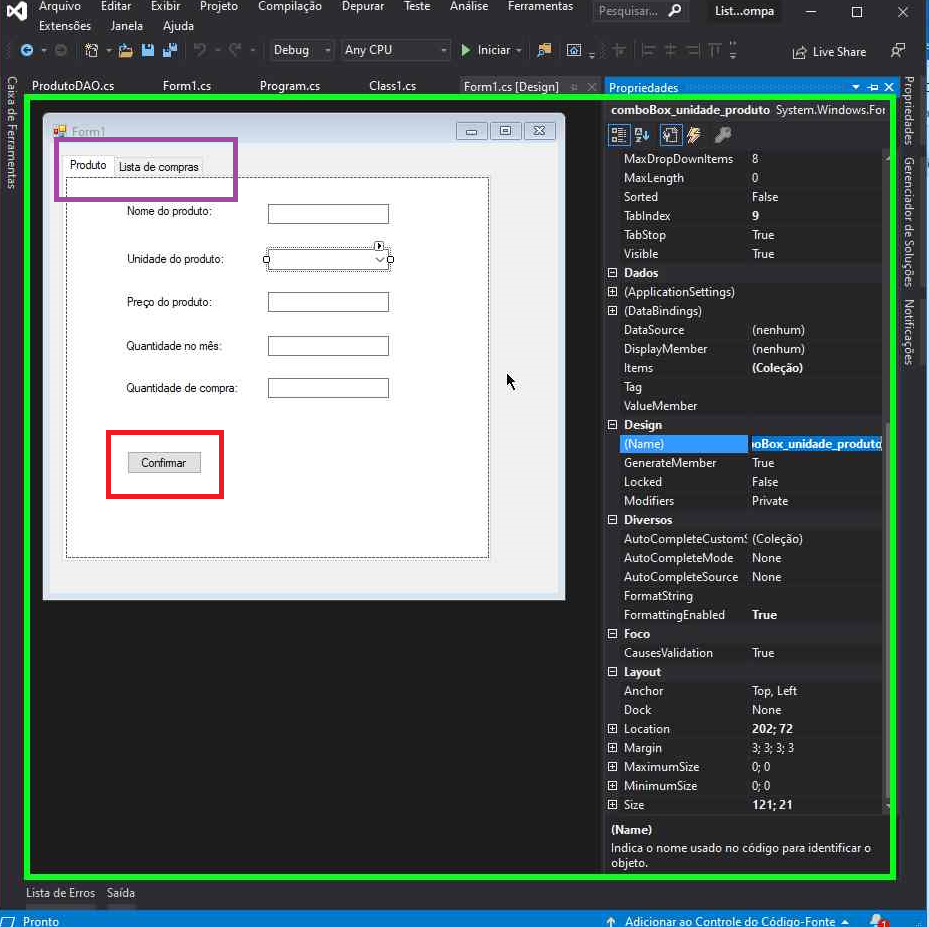


Fonte: Autor próprio

* 1. Criando as telas do projeto

Como padrão na tela de Design do Windows Forms ele aparece uma janela em branco, para o desenvolvimento desse projeto vai ser utilizado a ferramenta TabControl, com ela cria tabelas paralelas e separa os tipos de preenchimentos que vai ser gerenciado. Para essa solução vai ser usadas duas tabelas e são referenciadas conforme o Quadro 1 e o Quadro 2, que usa os diagramas de classe e os levantamentos para adicionar o que deve ter nas tabelas criadas as quais são “Produto” e “Lista de Compras”.

1. Tabela Produto

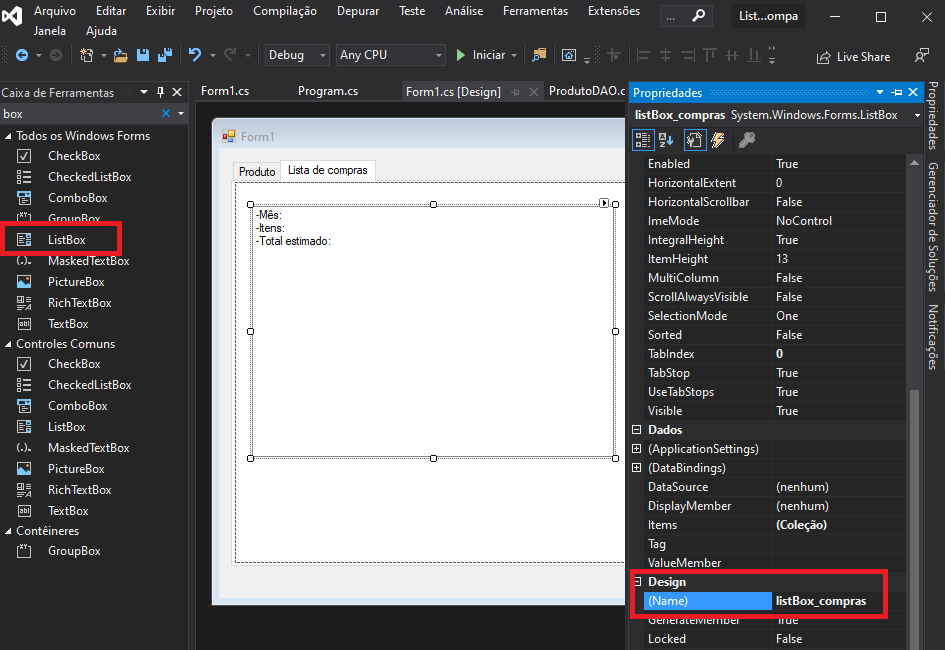


Fonte: Autor próprio

As nomenclaturas de cada ferramenta colocada dentro do código, tem como padrão colocar o nome e o número da ferramenta criada, para facilitar a localização da ferramenta colocada é posto a abreviatura da ferramenta e o nome da função que vai exercer.

Conforme a FIGURA 22, a tabela “Lista de compras”, é uma tabela que recebe a listagem dos produtos e a descrição de cada e um botão para adicionar os produtos, os itens vai ser a quantidade de produtos que vão ser adicionados na tela *Listbox*.

1. Tabela “Lista de compras”.



Fonte: Autor próprio

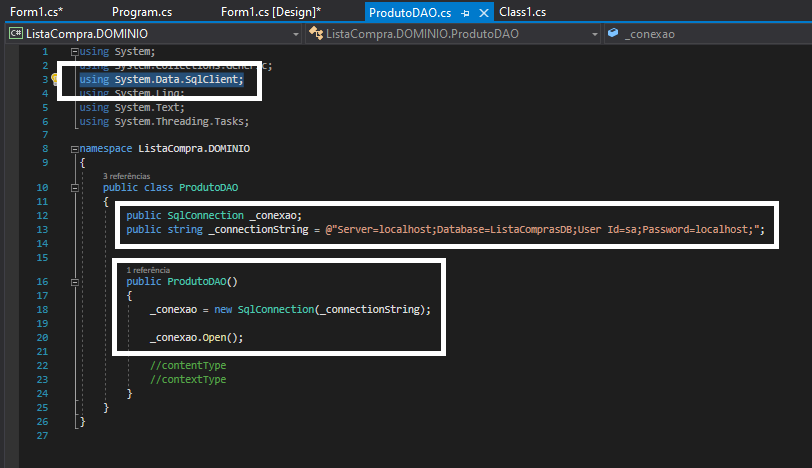
* 1. Criando um teste em uma classe de acesso a dados (DAO)

No projeto de banco de dados vai ser colocado um "DAO" para cada classe que vai ter, sendo assim são duas no projeto "produto.DAO" e "lista\_produto.DAO". O2 produto DAO, é responsável por abrir e fechar conexão com o banco de dados, onde vai ter as “Queue<T>” que vai fazer a comunicação com o banco de dados e fazer inserção, conexão e afins.

- O “.dao" é uma dll de banco de dados que faz comunicação da classe domínio com as tabelas do banco de dados, com isso ele vai fazer o gerenciamento da tela de interface.

Para fazer a conexão é usado o “SqlConnection”, que representa uma conexão ao banco de dados do SQL Server, com um sistema de banco de dados cliente/servidor, é equivalente a uma conexão de rede com o servidor.

1. Aplicação do SqlConnection no Visual Studio



Fonte: Autor próprio

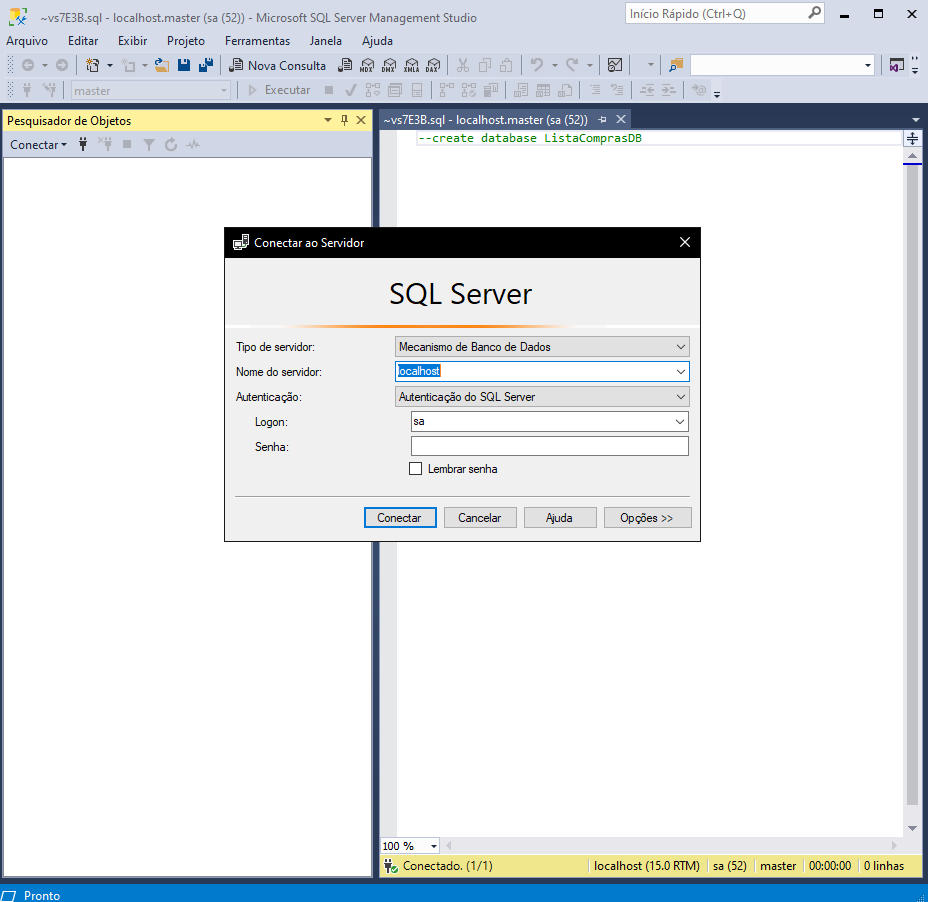
Em seguida é necessário um Connection String para fazer a cadeia de conexão com o banco de dados, nesse caso irá ser usado uma conexão via SSMS, com ele tem que colocar o nome do servidor, o nome do banco de dados, o nome do usuário e a senha.

Para ele abrir uma conexão se usa o comando “.Open();” com isso ele vai abrir conexão para acessar o SSMS e o banco de dados atribuídos.

* 1. Criando uma base de dados no SQL Management Studio

O SSMS (SQL Server Management Studio), é um ambiente integrado feito para gerenciar seja qual for a infraestrutura SQL. O uso dele tem como função de acessar, configurar, gerenciar, desenvolver, administrar todos os componentes e fornecer um grupo de ferramentas gráficas para proporcionar o acesso ao SQL Server.

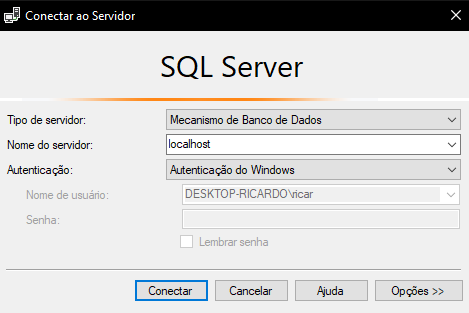
1. Tela Principal do SSMS



Fonte: Autor próprio

Para acessar o autenticador do SQL server, precisa entrar primeiro na autenticação do Windows.

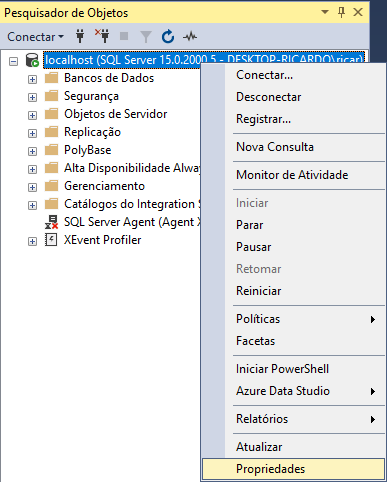
1. Autenticação do Windows no SQL Server



Fonte: Autor próprio

Com o acesso do banco de dados via Windows, é necessário ativar para que ele aceite autenticações via SQL, para ativar precisa ir nas propriedades indo pela instância do SQL Server.

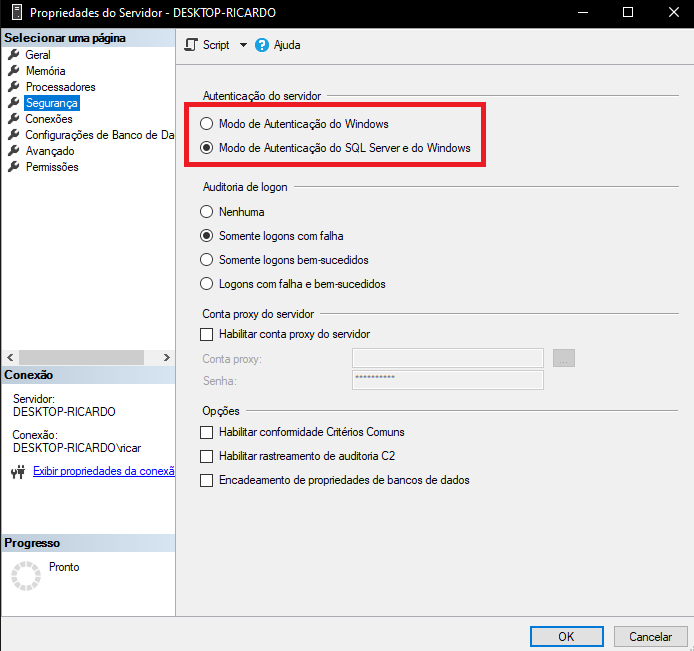
1. Propriedades do SQL Server



Fonte: Autor próprio

Dentro das propriedades, entrar na opção de segurança e mudar o autenticador de Windows para SQL Server e do Windows.

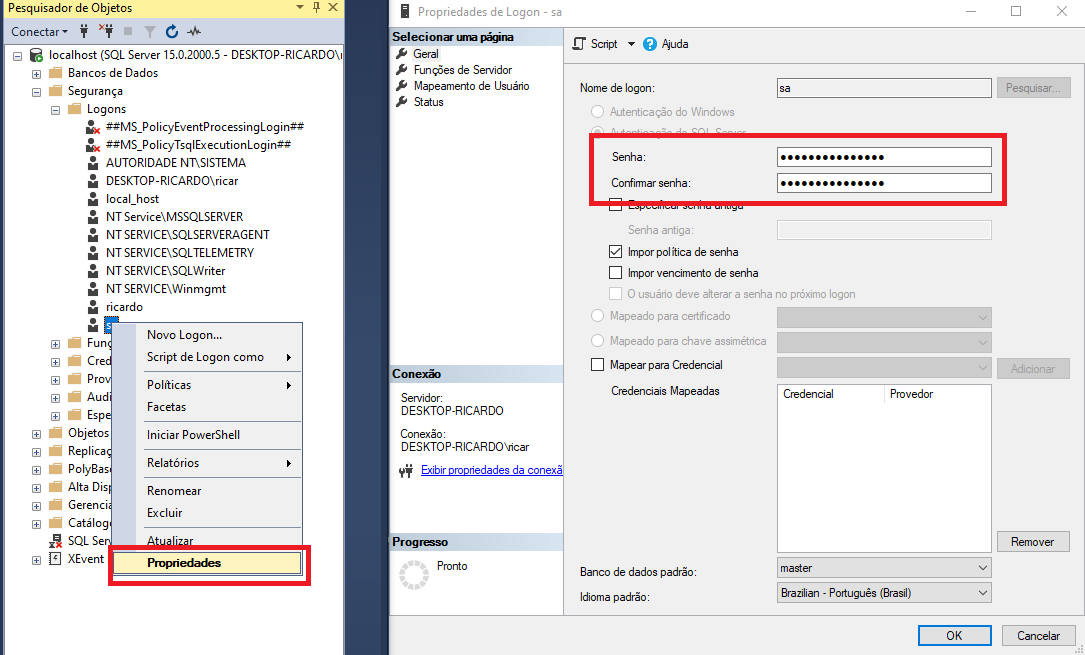
1. Mudando o modo de autenticação no SSMS



Fonte: Autor próprio

Quando selecionado e depois de confirmar a opção o SSMS mostrará um alerta que houve a mudança. Agora é necessário habilitar o usuário SA. Para habilitar o usuário SA, Expanda a opção “Segurança” e depois expanda a opção “Logons”, ao visualizar o usuário SA, é preciso ir nas propriedades dele.

1. Mudança de senha do “sa”

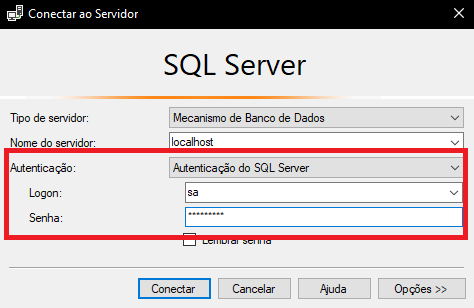


Fonte: Autor próprio

Na FIGURA 27 dentro das propriedades, o SSMS gera uma senha *default* para que haja uma troca de senha pelo usuário do programa, assim que é trocada a senha, o SSMS precisa reiniciar o banco de dados.

Após ele ser reiniciado, ele irá pedir a autenticação de conexão, ao invés de usar o autenticador do Windows, agora configurado vai ser usado o SQL, com isso ele pedirá o nome do usuário (“sa”) e a senha configurada.

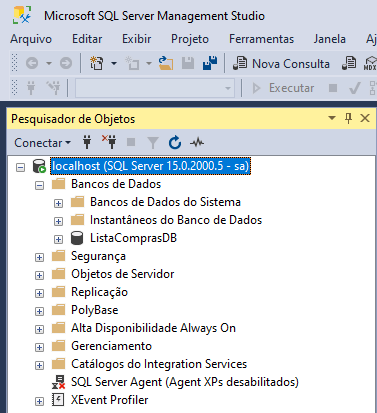
1. Fazendo o Login com o Autenticador do SQL Server



Fonte: Autor próprio

E com isso conforme a FIGURA 29 o banco de dados estará pronto para ser usado.

1. Banco de dados configurado e conectado.

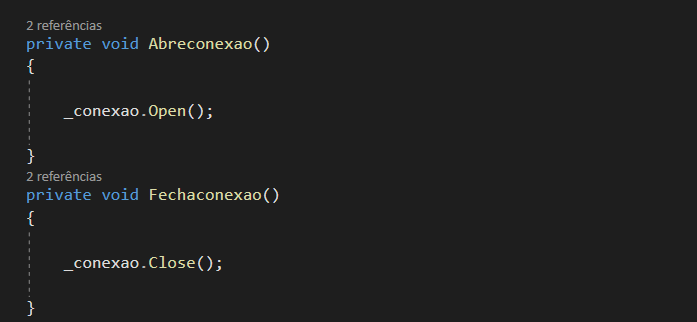


Fonte: Autor próprio

1. adicionando um produto no banco de dados

Para começar com que possa adicionar a lista de compras no banco de dados precisa fazer certos requisitos de preparação e é necessário que ele acesse o banco de dados e que armazene os parâmetros corretamente em suas posições. Começando em criar métodos de abrir uma conexão “.*Open()*;” e outra de fechar “.*Close();*” que toda vez que é aberta uma conexão e depois de ter feito os comandos tem que fecha-la.

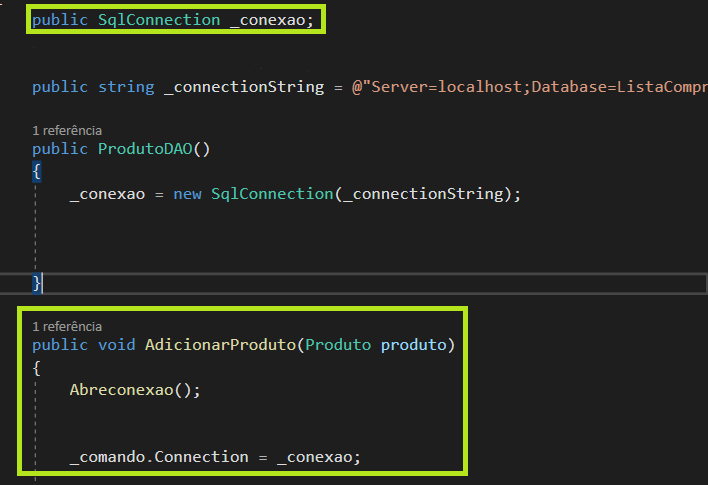
1. Métodos de abrir e fechar conexão



Fonte: Autor próprio

Após a criação dos métodos de conexão, sendo necessário um SqlConnection, é uma representação de uma conexão e representa uma sessão exclusiva para uma fonte de banco de dados SQL Server, e também de ter um método de adicionar um produto (“*AdicionarProduto*();”) em uma lista com o nome preço e unidade para salvar o serviço.

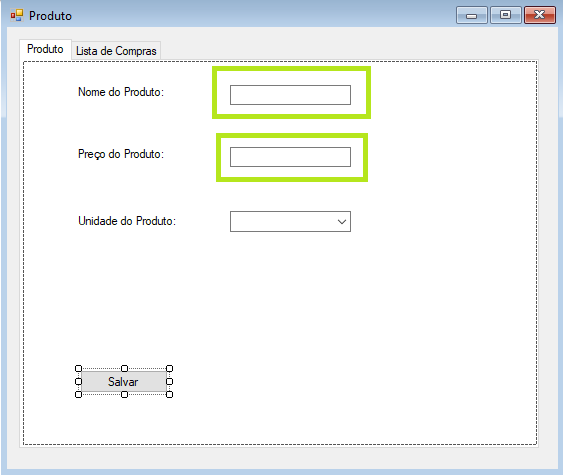
1. Adicionando o produto



Fonte: Autor próprio

Com o objetivo de adicionar é usado o fluxo reto, que quando é usado para adicionar algo sem que haja alterações no código e o produto que precisa passar os dados para colocar no banco de dados, na variável local que tem que receber as descrições do produto, dento do WindowsForms no “Textbox”.

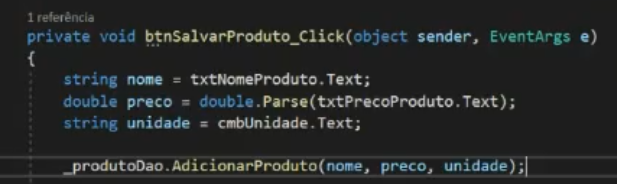
1. TextBox na interface



Fonte: Autor próprio

Dentro do botão, a diferença entre eles é o que vai ser adicionado, no caso do preço vai ser colocado um double para representar números reais e o string para o preço, no caso do C# é necessário fazer uma conversão, e também deve ser passado por parâmetro usando as mesmas descrições, sendo usado na mesma ordem que ela foi criada e colocada.

1. Representações no botão “salvar”

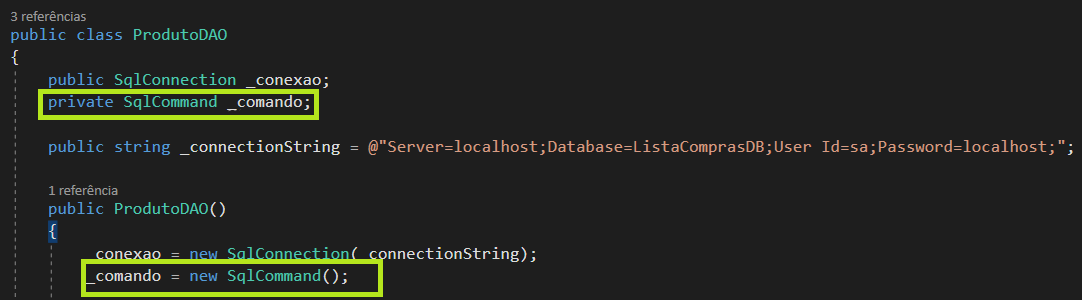


* 1. Adicionando uma lista no Banco de dados

A primeira coisa a ser colocada para adicionar o produto é necessário que abra a conexão para que pegue os dados que o cliente inseriu na tela de interface do programa para se comunicar com o banco de dados.

Em questão de começar a trabalhar com essa parte é adicionado o comando SqlCommand, que serve para criar os comandos (private SqlCommand \_comando;) e um objeto que trabalhe com ele, para abrir tem colocar a conexão que está sendo trabalhada para informar que o comando que se situa-se sendo efetuado está sendo puxado para o banco de dados assim ele reconhece o SSMS e fazer esse tipo de comunicação.

1. Colocando o SqlCommand

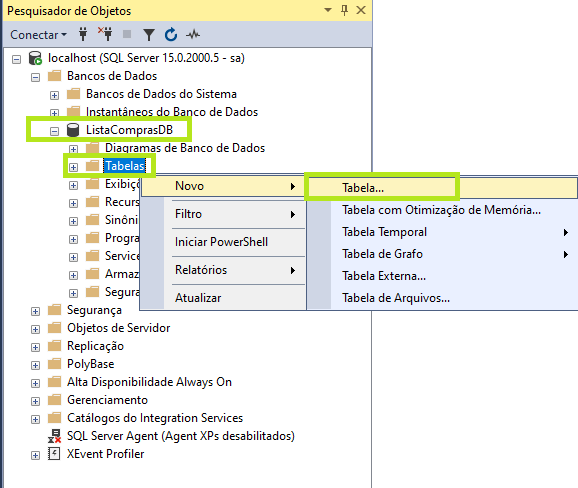


Fonte: Autor próprio

* 1. Criando uma tabela no SSMS

Na adição de um dado no SQL é preciso colocar um *insert* e nele introduzir uma tabela no SSMS, após finalizado é colocado um nome na tabela (TB\_Produto) usando uma padronização do *SQLServer*.

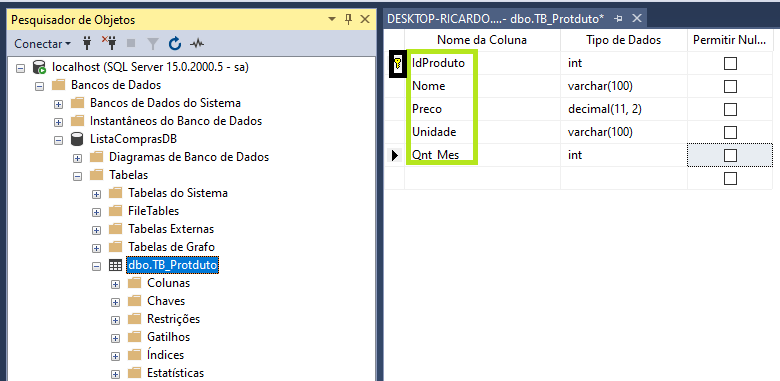
1. Adicionando uma nova tabela



Fonte: Autor próprio

depois de finalizado as configurações é necessário atualizar ela e podendo adicionar um dado dentro dela.

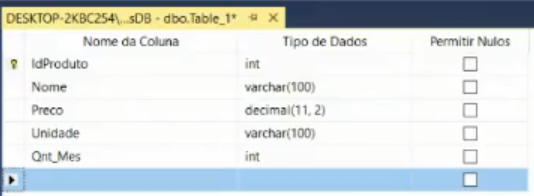
1. Tabela do SSMS criada



Fonte: Autor próprio

Quando foi criado um script, é preciso fazer um requerimento de testar ele, usando a função do SSMS de criar uma nova consulta usando o script da tabela criada para o insert.

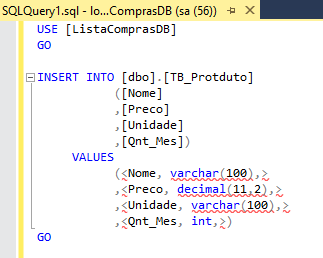
1. Script criado



Fonte: Autor próprio

Conforme a FIGURA 38, a tabela de banco de dados à qual a linha ela será adicionada, nas colunas cujo o conteúdo desejado em adicionar e finalizando com o valor ou a expressão a ser inserida nas colunas individuais.

1. O *insert* e *values* dentro do SSMS

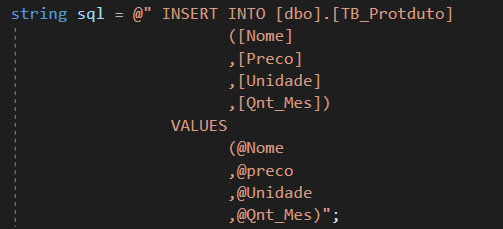


Fonte: Autor próprio

* 1. Usando o insert e values no código

com o insert pronto ele vai ser colocado na Queue de adicionar o produto: e nele é colocado o nome, preço, unidade e quantidade no mes, quando ele tem um auto incremento ele o ID não precisa ser inserido.

1. Insert e values dentro do ProdutoDao



Voltando para a tabela no SSMS, a tabela só vai querer são os valores que vão ser inseridos, sendo na parte do insert ele referência e com o valor se colocar o que vai ser inserido no banco, esses dados que vão ser postos na tela da lista de compras.

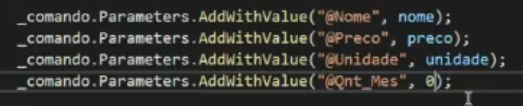
1. Tabela da lista de compras



Fonte: Autor próprio

O comando (“\_comando.Parameters.AddWithValue”) serve para pegar o valor adicionado no tipo de parâmetro que ele atribui dentro da lista de compras. que vai pegar o que está dentro do campo na listagem de compras, e com ele vai passar o valor dos parâmetros.

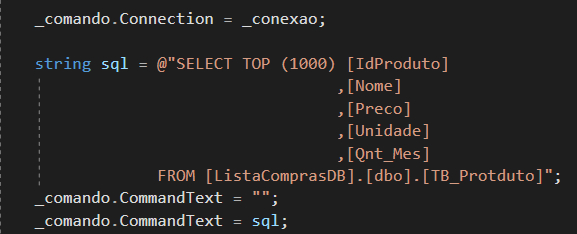
1. Complementando o (“AddWithValue”)



Fonte: Autor próprio

Após isso precisa definir que com tudo que foi acrescentado em criar os parâmetros, script, tem que adicionar que ele é o comando de texto (“*\_comando.CommandText = sql;*”) que está sendo feito, e no final do da adição do produto, deve criar um comando que tudo vai ser executado no banco de dados, fazendo que esse meio campo de pegar do C#, executar esse script dentro da tabela automaticamente. após fazer as ações que são prescritas para essa função é necessária para fechar o processo de acesso a conexão.

1. Comando de texto *CommandText*



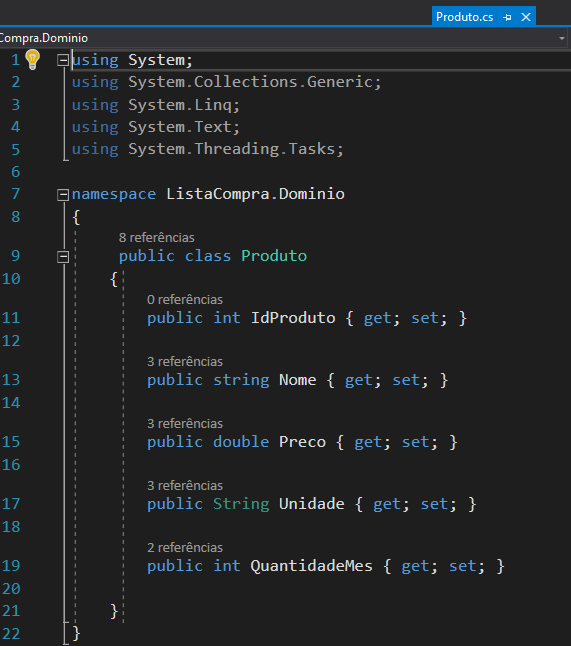
Fonte: Autor próprio

Dentro do domínio a necessidade de modificá-la tem o objetivo que foi criado as variáveis para acrescentar os valores e alterar elas para virarem um objeto, com isso precisa ser adicionada uma classe dentro do domínio (“produto.cs”).

* 1. Classe Produto

Essa classe é classificada como burra pois não contém nenhum tipo de inteligência dentro dela, por hora não está sendo utilizado pois os itens dentro dele adiciona e é listado, pois por hora não vai a classe que fazer esse tipo de função, vai ser utilizado o (“ProdutoDAO”), isso serve como explicação que não tem adicionar ou remover dentro da classe burra, e também ela precisa ser pública que com ela vai poder ter o acesso para fora desse (“projetoDominio”).

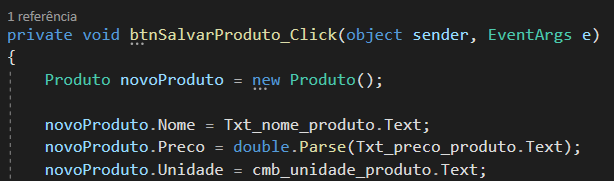
1. Classe burra



Fonte: Autor próprio

A classe (”produto”) vai ser instanciada e ficará como local dentro do botão de salvar pois a classe vai ser modificada para que ele ocorra uma instancia com a classe "produto" para puxar de adicionar e listar as informações colocadas na lista de compras do produto, prosseguindo ele o (“\_produtoDao”) também pode ser modificado com o objeto novoProduto.

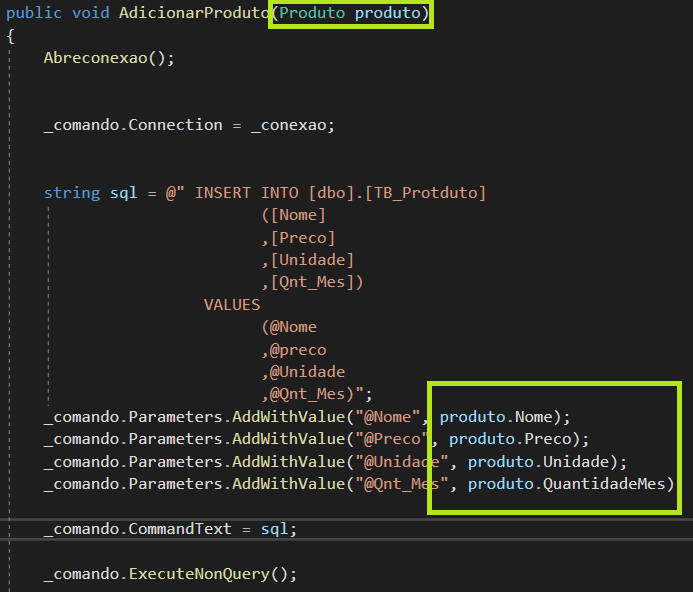
1. Instanciando Produto no botão



Fonte: Autor próprio

No parâmetro (“\_comando.Parameters.AddWithValue”) deve ter essa modificação de colocar o objeto do tipo "produto" que quando for executado o programa e de ter adicionado um produto para colocar na lista, na depuração do botão ele vai definir que esse produto é um novo produto a ser colocado e adicionas as especificações do produto, sendo assim ela vai ser adicionado no (“\_produtoDao.AdicionarProduto(novoProduto);”), que assim vai para o produtoDao abrir a conexão e adicionar os parâmetros para o banco de dados.

1. Adicionando (“produto”) no AddWithValue



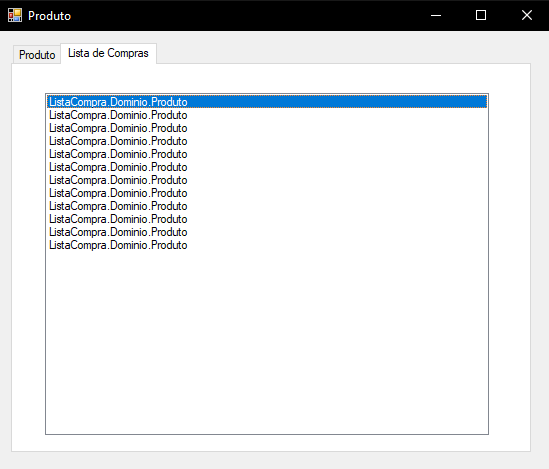
Fonte: Autor próprio

Com o método (“AdicionarProduto(Produto produto)”), o ID ele está zerado pois não é definido no começo da depuração, o que vai dar auto incremento vai ser no banco de dados.

* 1. Configurando o Tabpage

Dentro da *tabpage*  da lista de compra, é criada uma ListBox mostrando a lista dos produtos no banco de dados e nela mostrará na *tabpage* produto sendo gerenciado o produto (item de compra adicionado), por hora não está adicionando na lista, para ter essa lista ela tem que guardar dentro do banco de dados.

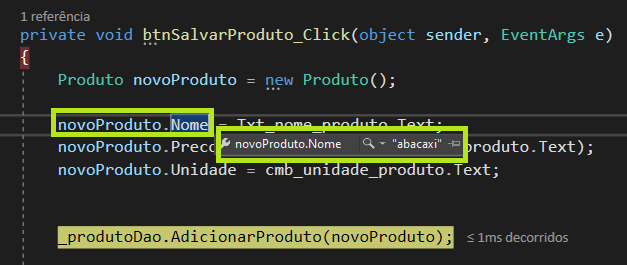
1. *Tabpage* da lista de compras



o *tabpage* produto tem que ir para *tabpage* de lista de compras, sendo que deve ir para o banco de dados primeiro e depois que adicionado ele ser listado nessa *tabpage*, com isso em mente precisa adicionar um método que lista a lista dos produtos, dentro dele é necessário que para acessar a lista no banco de dados tem que abrir e fechar uma conexão para que só seja usado quando puxa essa lista e depois disso ele fechar a conexão.

Com a chamada no banco e de como deve ser listado, o *script* pego deve listar todos os produtos dentro do banco de dados, em seguida se coloca um objeto fazendo a adição de comandos (“SqlCommand”), e outro que deve ler os dados do banco (“SqlReader”) comandos de insert, updates e etc...

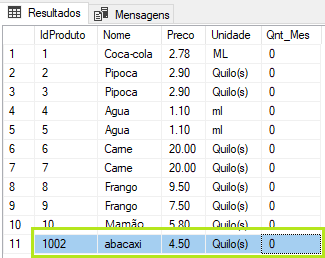
1. Depurando um novo produto a ser adicionado



Fonte: Autor próprio

Conforme a FIGURA 45, ela irá receber uma a própria execução com uma lista no SqlServer e ele vai trazer a lista e vai ter um leitor, depois se coloca um repetidor que lista os produtos e vai adicionar cada item que foi colocado no insert, como o "produto" foi adicionado como classe e atribuiu a ele um novo valor para ser adicionado na lista e os itens que estão dentro vem do banco de dados e deve ser passado pela ordem dos parâmetros em ordem correta e assim ele vai está adicionando a lista de produtos.

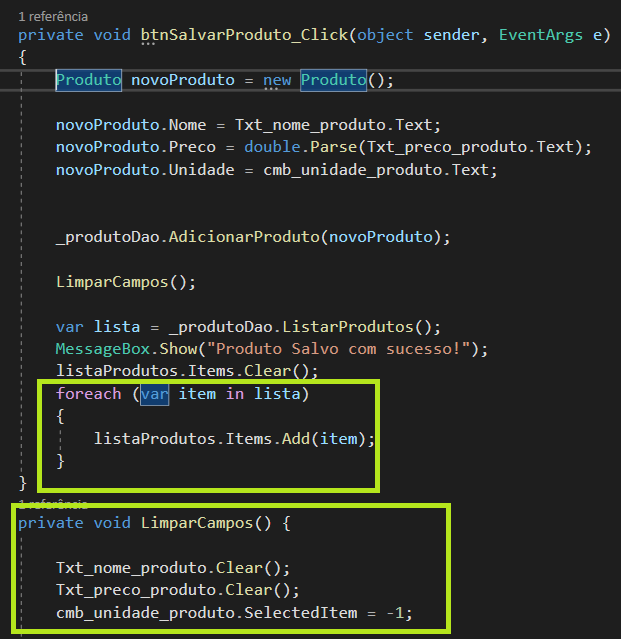
1. Produto adicionado no banco de dados



Fonte: Autor próprio

Após isso é voltado ao botão (“salvar”), ele terá um *foreach* e será um lasso de repetição que deve passar por todo ele, e lista completa que vem do banco de dados, e será adicionado um item dentro dela.

1. Usando a função *clear* e repetidor



Fonte: Autor próprio

Por fim os campos vão ser limpados toda vez que o produto é adicionado junto com a listagem na lista de produtos, resumindo, toda a vez que é adicionado um produto ele vai limpar, depois listar os produtos e por fim atualizar e no banco de dados, o preenchimento de números não e aceito o número vazio o index 0 é o primeiro do combo box, e o -1 é o vazio.